

Silver Vulnerability Report

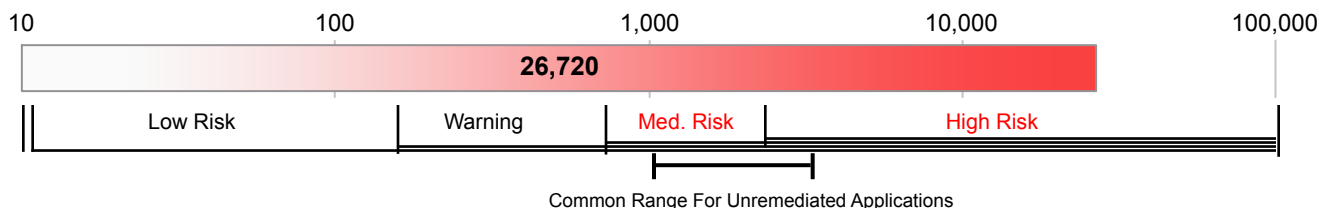
URL Assessed: http://crackme.cenzic.com

Summary

An assessment was performed on the target application, using 16 SmartAttacks™ to identify vulnerabilities. As shown below, a large number of issues were found with the application that resulted in a **very high HARM™ score of 26,720**, which is commonly seen when there is at least one broadly repeated significant vulnerability. Typical HARM scores for complete assessments of individual non-remediated applications range from 1000 to 4500. **It is recommended that action be taken to remediate these vulnerabilities.** The tests performed, as well as specific vulnerabilities to be remediated, and their contribution to the total HARM score, can be viewed in the "SmartAttack Results (by HARM)" section of this report below.

[HARM Scoring](#)

Total HARM™ Score: 26720 = 26720 (Raw Scores) x 1.0 (App Risk Factor)

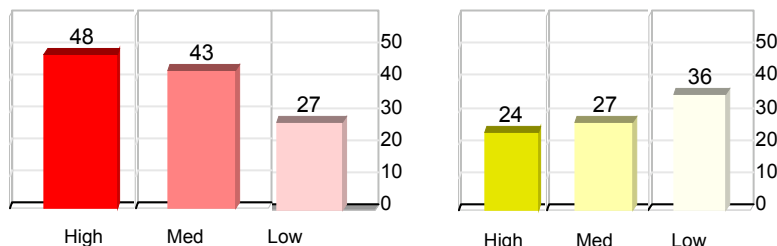


The 'Total HARM Score', above, is a sum of the HARM scores for all the SmartAttack assessments included in this report. SmartAttacks have different HARM scores based on the risks associated with each kind of vulnerability. The charts reflect the raw HARM scores without application specific risk adjustments.

Severity of Findings

[Severity Drill Down](#)

[Severity Drill-down w/Info Items](#)



Pages Tested	100
Attack Count	12136

Note: **High, Med. & Low** relate to the severity of the findings. **Warnings** are findings for which there is less confidence of being real vulnerabilities.

High Severity

SmartAttack.	Vuln.	Warn.
Credit Card Disclosure	20	0
Cross-Site Scripting	26	0
Non-SSL Password	2	0
Non-SSL Form	0	18
Cross Site Request Forgery	0	6

Medium and Low Severity

SmartAttack.	Vuln.	Warn.
SQL Error Message	33	0
Directory Browsing	8	0
Form Caching	0	26
Password Autocomplete	1	0
Check HTTP Methods	0	1
Web Server Vulnerabilities	0	36
HTML & JavaScript Comments	1	0
HTML & JavaScript Comments	27	0

SmartAttack Results (by HARM)

[SmartAttack Drill-down](#)[SmartAttack Drill-down w/Info Items](#)

	<u>Status</u>	<u>Severity</u>	<u>HARM</u>	<u>Vuln</u>	<u>Warning</u>	<u>Info</u>
Cross-Site Scripting	Alert	High	8320	26	0	0
Credit Card Disclosure	Alert	High	11520	20	0	1
Non-SSL Password	Alert	High	1280	2	0	0
Non-SSL Form	Alert	High	600	0	18	0
Cross Site Request Forgery	Alert	High	462	0	6	0
.....
SQL Error Message	Alert	Medium	1500	33	0	0
HTML & JavaScript Comments	Note	Medium	200	1	0	0
"		+ Low		27	0	
Directory Browsing	Alert	Medium	1280	8	0	0
Password Autocomplete	Note	Medium	210	1	0	0
Form Caching	Alert	Medium	1000	0	26	0
Check HTTP Methods	Note	Medium	24	0	1	1
.....
Web Server Vulnerabilities	Alert	Low	324	0	36	0
.....
Remote File Inclusion	Ok	High	0	0	0	0
File & Directory Discovery	Ok	Medium	0	0	0	2
Application Exception	Ok	Medium	0	0	0	3
URL In Query	Ok	Low	0	0	0	0

Note: Detailed individual findings start on the next page.

Detail (by HARM / SmartAttack)

Attack: Credit Card Disclosure	Ver: 1.0.11	CWE-359	Observations: 20
Descr: Credit Card Disclosure vulnerabilities are disclosures of credit card numbers in responses from the Web application. The SmartAttack observes all responses and reports any pages where a full credit card number is observed. The significance of such findings is dependent on the intention and other security factors related to this display.			
Severity: High HARM: 576 Total HARM: 11520			

Impact

An attacker browsing through a Web application that displays valid credit card numbers on a page may store and use the displayed credit card numbers to assume identities of the persons to whom the credit cards belong. This may lead to the attacker discovering personal information about the owner or making use of the identities for his benefit, such as authorizing transactions using the credit card, requesting a replacement credit card on a new address, *etc.*

Even if the display of a credit card number by a Web application is protected by logins, they may still be disclosed to an attacker peeping over the shoulder of an unsuspecting user of the application.

Credit Card Disclosure Vulnerable Findings

Assessment: Silver, **Run:** 6/6/2011 8:38:25PM, **Traversal:** [Default Spider]

Credit Card Disclosure**1. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>**

Message: Valid credit card number(s) found on page.
4386 2425 0139 5127

Credit Card Disclosure**2. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>**

Message: Valid credit card number(s) found on page.
4386 2425 0139 5127

Credit Card Disclosure**3. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>**

Message: Valid credit card number(s) found on page.
4346 7841 0044 0795

Credit Card Disclosure**4. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>**

Message: Valid credit card number(s) found on page.
4346 7841 0044 0795

Credit Card Disclosure**5. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>**

Message: Valid credit card number(s) found on page.
4346 7841 0044 0795

Credit Card Disclosure**6. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>**

Message: Valid credit card number(s) found on page.
4346 7841 0044 0795

Credit Card Disclosure**7. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>**

Message: Valid credit card number(s) found on page.
5264 1902 0028 5277

Credit Card Disclosure**8. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>**

Message: Valid credit card number(s) found on page.
4129 0380 8370 8477

Credit Card Disclosure

9. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Valid credit card number(s) found on page.
4129 0380 8370 8477

Credit Card Disclosure

10. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Valid credit card number(s) found on page.
4346 7841 0044 0910

Credit Card Disclosure

11. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: Valid credit card number(s) found on page.
4386 2425 0139 5127

Credit Card Disclosure

12. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: Valid credit card number(s) found on page.
4386 2425 0139 5127

Credit Card Disclosure

13. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: Valid credit card number(s) found on page.
4346 7841 0044 0795

Credit Card Disclosure

14. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: Valid credit card number(s) found on page.
4346 7841 0044 0795

Credit Card Disclosure

15. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: Valid credit card number(s) found on page.
4346 7841 0044 0795

Credit Card Disclosure

16. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: Valid credit card number(s) found on page.
4346 7841 0044 0795

Credit Card Disclosure

17. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: Valid credit card number(s) found on page.
5264 1902 0028 5277

Credit Card Disclosure

18. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: Valid credit card number(s) found on page.
4129 0380 8370 8477

Credit Card Disclosure

19. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: Valid credit card number(s) found on page.
4129 0380 8370 8477

Credit Card Disclosure

20. Vulnerable (High, HARM: 576) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: Valid credit card number(s) found on page.
4346 7841 0044 0910

Remediation Tips

Make sure that your Web application does not disclose credit card numbers on any of the pages, whether inside or outside a login.

References

Credit card fraud at Wikipedia: http://en.wikipedia.org/wiki/Credit_card_fraud

Remediation References

OWASP card handling guidelines based on the PCI DSS:

http://www.owasp.org/index.php/Handling_E-Commerce_Payments

The PCI Data Security Standard: https://www.pcisecuritystandards.org/security_standards/pci_dss.shtml

Attack: Cross-Site Scripting	Ver: 2.0.39	CWE-79	Observations: 26
Descr: Cross-site Scripting vulnerabilities allow malicious scripts to execute in the context of a trusted session with a web site. The SmartAttack alters the inputs to the web application to send benign versions of such malicious scripts, and detects the actual execution or unfiltered reflection of such scripts.			
Severity: High HARM: 320 Total HARM: 8320			

Impact

Cross-Site Scripting enables an attacker to run scripts inside a victim's browser. Using such a script, the attacker can modify the look-and-feel of a page, deface page contents and even steal user credentials and session information. If an application uses cookies for session management, then a Cross-Site Scripting vulnerability also assists the attacker in exploiting certain session-based attacks such as Session Fixation, if present.

Many Web applications display user input on their Web pages. Depending on whether the input is stored by the application for repeated use (e.g. user comments), a Cross-Site Scripting vulnerability may be *reflected* - i.e. usually one time - or *persistent*. A persistent Cross-Site Scripting vulnerability has greater impact than a reflected Cross-Site Scripting vulnerability, because a large number of users are affected without elaborate actions on the victims' part.

The effects of a Cross-Site Scripting vulnerability may range from simple defacement of Web pages to serious identity theft. For example, a Cross-Site Scripting vulnerability on a page that displays user-uploaded images could enable an attacker to show offensive images as if they were uploaded by a legitimate user, while a Cross-Site Scripting vulnerability on a banking Web site may expose the credentials of customers of the bank. While the offensive image may only affect a handful of users and the effect would be more annoyance than real harm, exposure of the credentials poses the threat of the attacker stealing money from them.

Cross-Site Scripting Vulnerable Findings

Assessment: Silver, **Run:** 6/6/2011 8:38:25PM, **Traversal:** [Default Spider]

Cross-Site Scripting

1. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/register/register.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: UserId
Injection value: >'><script>alert(13074180.137)</script>
Detection value: 13074180.137
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

2. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/view/updateloanrequest.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: txtFirstName
Injection value: >'><script>alert(13074180.4907)</script>
Detection value: 13074180.4907
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

3. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/view/updateloanrequest.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: txtAnnualIncome
Injection value: >'><script>alert(13074180.8007)</script>
Detection value: 13074180.8007
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

4. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/login.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: hUserType
Injection value: "><script>alert(13074180.9377)</script>
Detection value: 13074180.9377
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

5. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/accttransaction.php>

Message: Cross-site scripting vulnerability found
 Injected item: POST: FromDate
 Injection value: "><script>alert(13074180.11897)</script>
 Detection value: 13074180.11897
 This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

6. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/accttransaction.php>

Message: Cross-site scripting vulnerability found
 Injected item: POST: ToDate
 Injection value: "><script>alert(13074180.11917)</script>
 Detection value: 13074180.11917
 This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

7. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/login.php>

Message: Cross-site scripting vulnerability found
 Injected item: POST: hUserType
 Injection value: "><script>alert(13074180.12737)</script>
 Detection value: 13074180.12737
 This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

8. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
 Injected item: POST: hUserId
 Injection value: "><script>alert(13074180.16057)</script>
 Detection value: 13074180.16057
 This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

9. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
 Injected item: POST: hUserId
 Injection value: "><script>alert(13074180.16487)</script>
 Detection value: 13074180.16487
 This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

10. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
 Injected item: POST: hUserId
 Injection value: "><script>alert(13074180.16587)</script>
 Detection value: 13074180.16587
 This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

11. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
 Injected item: POST: hUserId
 Injection value: "><script>alert(13074180.17057)</script>
 Detection value: 13074180.17057
 This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

12. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
 Injected item: POST: hUserId
 Injection value: "><script>alert(13074180.17527)</script>
 Detection value: 13074180.17527
 This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

13. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: hUserId
Injection value: "><script>alert(13074180.17997)</script>
Detection value: 13074180.17997
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

14. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: hUserId
Injection value: "><script>alert(13074180.18467)</script>
Detection value: 13074180.18467
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

15. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: hUserId
Injection value: "><script>alert(13074180.18937)</script>
Detection value: 13074180.18937
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

16. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: hUserId
Injection value: "><script>alert(13074180.19407)</script>
Detection value: 13074180.19407
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

17. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: hUserId
Injection value: "><script>alert(13074180.19877)</script>
Detection value: 13074180.19877
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

18. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: hUserId
Injection value: "><script>alert(13074180.20347)</script>
Detection value: 13074180.20347
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

19. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: hUserId
Injection value: "><script>alert(13074180.20817)</script>
Detection value: 13074180.20817
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

20. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: hUserId
Injection value: "><script>alert(13074180.21287)</script>
Detection value: 13074180.21287
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Cross-Site Scripting

21. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: hUserId
Injection value: "><script>alert(13074180.21757)</script>

Detection value: 13074180.21757

This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

[Cross-Site Scripting](#)

22. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: hUserId
Injection value: "><script>alert(13074180.22227)</script>
Detection value: 13074180.22227
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

[Cross-Site Scripting](#)

23. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/transfer.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: Amount
Injection value: "><script>alert(13074180.31147)</script>;//
Detection value: 13074180.31147
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

[Cross-Site Scripting](#)

24. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/transfer.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: ToAccountNo
Injection value: "><script>alert(13074180.31137)</script>;//
Detection value: 13074180.31137
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

[Cross-Site Scripting](#)

25. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/login.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: hLoginType
Injection value: "><script>alert(13074180.36937)</script>
Detection value: 13074180.36937
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

[Cross-Site Scripting](#)

26. Vulnerable (High, HARM: 320) at: <http://crackme.cenzic.com/Kelev/php/login.php>

Message: Cross-site scripting vulnerability found
Injected item: POST: hLoginType
Injection value: "><script>alert(13074180.39487)</script>
Detection value: 13074180.39487
This is a reflected XSS vulnerability, detected in an alert that was an immediate response to the injection.

Remediation Tips

The following general recommendations can help mitigate the risk associated with Cross-Site Scripting vulnerabilities. This is a complex problem area so there is no one simple fix or solution:

- Ensure that your web application validates all forms, headers, cookie fields, hidden fields, and parameters, and converts scripts and script tags to a non-executable form.
- Ensure that any executables on your server do not return scripts in executable form when passed scripts as malformed command parameters.
- Consider converting JavaScript and HTML tags into alternate HTML encodings (such as “<” to “<”).
- If your site runs online forums or message boards, disallow the use of HTML tags and Scripting in these areas.
- Keep up with the latest security vulnerabilities and bugs for all production applications and servers.
- Update your production servers with the latest XSS vulnerabilities by downloading current patches, and perform frequent security audits on all deployed applications.

The root cause of Cross-Site Scripting is a failure to filter hazardous characters from web application’s input and output. The two most critical programming practices you can institute to guard against Cross-Site Scripting are:

- Validate Input
- Encode output

Always filter data originating from outside your application by disallowing the use of special characters. Only display output to the browser that has been sufficiently encoded. When possible, avoid simple character filters and write routines that validate user input against a set of allowed, safe characters. Use regular expressions to confirm that data conforms to the allowed character set. This enhances application security and makes it harder to bypass input validation routines.

There are different tools you can use to validate and encode your data, depending upon your development environment. Your goal in Cross-Site Scripting attacks remediation should be to filter and encode all potentially dangerous characters so that the application does not return data that the browser will interpret as executable. Any non-escaped or non-encoded data that is returned to the browser is a potential security risk.

The following characters can be harmful and should be filtered whenever they appear in the application input or output. In output, you should translate these characters to their HTML equivalents before returning data to the browser.

> < () [] ' " ; : / |

PHP

The following PHP functions help mitigate Cross-Site Scripting **Vulnerabilities**:

Strip_tags() removes HTML and PHP scripting tags from a string.

Utf8_decode() converts UTF-8 encoding to single byte ASCII characters. Decoding Unicode input prior to filtering it can help you detect attacks that the attacker has obfuscated with Unicode encoding.

Htmlspecialchars() turns characters such as &, >, <, ” into their HTML equivalents. Converting special characters to HTML prevents them from being executable within browsers when sent by an application.

Strtr() filters any characters you specify. Make sure to filter “; : ()” characters so that attackers cannot craft strings that generate alerts. Many XSS attacks are possible without the use of HTML characters, so filtering and encoding parentheses mitigates these attacks. For example:

```
" style="background:url(Javascript:alert(Malicious Content));
```

ASP.NET

With ASP.NET, you can use the following functions to help prevent Cross-Site Scripting:

- Constrain input submitted via server controls by using ASP.NET validator controls, such as *RegularExpressionValidator*, *RangeValidator*, and *System.Text.RegularExpressions.Regex*. Using these methods as server-side controls to limit data input to only allowable character sequences by validating input type, length, format, and character range.
- Use the *HtmlUtility.HtmlEncode* method to encode data if it originates from either a user or from a database. *HtmlEncode* replaces special characters with their HTML equivalents, thus preventing the output from being executable in the browser. Use *HtmlUtility.UrlEncode* when writing URLs that may have originated from user input or stored database information.
- Use the *HttpOnly* cookie option for added protection.
- As a best practice, you should use regular expressions to constrain input to known safe characters. Do not rely solely on ASP.NET *validateRequest*, but use it in addition to your other input validation and encoding mechanisms.

Java

When it comes to writing some validation code, there are two main choices: filtering and encoding.

Vulnerable code:

Consider the following code:

```
<% String sid = request.getParameter("sid"); %>
```

...

```
Student ID: <%= sid %>
```

This code functions correctly when the values of name are as expected. Since there is no validation of this, things can go awry if the inputs are not as expected. For example, a javascript can be made to execute that steals cookies.

Remediation to prevent these problems is outlined below.

Secure code:

Filtering:

There are two types of filtering: positive and negative filtering.

Positive Filtering:

The safest and most prevalent method of preventing against attack is to only accept data that is valid and reject everything else. For example, if the data is expected to be alphanumeric, then any input that is not should be rejected.

```
String Str = request.getParameter("input");
```

```
String Pattern = "\\d+$";
```

```
if (!Str.matches(Pattern))
```

```
    /* invalid input, take appropriate action*/
```

Negative Filtering:

Even though this is the ideal mechanism, it might not be practical to reject all data. For example, in blogging applications, it might be a requirement to allow the user to use html format to input data. In this case, check for the existence of special characters within the data. These characters can be replaced with other characters, such as a space.

```
/* regular expression that
```

```
* tests for the existence of malicious characters
```

```
* and replaces them with a space.*/
```

```
String Pattern="[<>{}\\[\\];\\&]";
```

```
String Str = s.replaceAll(Pattern, " ");
```

Encoding:

To ensure that the generated pages are properly encoded for a Web server, use a simple mechanism rather than an application. Pass each character in the dynamic content through an encoding function where the scripting tags in the dynamic content are encoded.

A tag library is made up of one or more classes and an XML tag library description file, which dictates the new tag names and valid attributes for those tags. Tag handlers determine how the tags, their attributes, and their bodies are interpreted and processed at request time from inside a JSP page.

Examples of Encoding Functions are below:

```
public int Encode () throws Exception {
    StringBuffer sbuf = new StringBuffer();
    char[] chars = property.toCharArray();
    for (int i = 0; i < chars.length; i++)
        sbuf.append("&#" + (int) chars[i]);
    try
    {
        pageContext.getOut().print(sbuf.toString());
    } catch (IOException ex) {
        throw new JspException(ex.getMessage());
    }
    return;
}
```

Java HTML Encoding Function

```
public static String HTMLEncode(String aTagFragment){
    final StringBuffer result = new StringBuffer();
    final StringCharacterIterator iterator = new
        StringCharacterIterator(aTagFragment);
```

```

char character = iterator.current();
while (character != StringCharacterIterator.DONE )
{
    if (character == '<')
        result.append("&lt;");
    else if (character == '>')
        result.append("&gt;");
    else if (character == '\\"')
        result.append("&quot;");
    else if (character == '\\')
        result.append("&#039;");
    else if (character == '\\')
        result.append("&#092;");
    else if (character == '&')
        result.append("&amp;");
    else {
        //the char is not a special one
        //add it to the result as is
        result.append(character);
    }
    character = iterator.next();
}
return result.toString();
}

```

ColdFusion

Generic

ColdFusion provides a number of ways to filter or validate user input. The security measures span both client-side and server-side filtering. Where possible, implement your security controls on the server-side to avoid tampering of your controls via a Man-In-The-Middle (MITM) proxy. An attacker can easily bypass client-side controls by using such a program to modify the underlying HTTP Request as it passes between the proxy and the web application. Additionally, we recommend that sites using ColdFusion should configure their application using the recommended security features of the Adobe ColdFusion 8 Developers Guide, or the guide relevant to your version of ColdFusion. ColdFusion Data validation allows you to control the type of data that is allowed as well as to ensure that user-supplied data corresponds to the correct form. Attentive data validation procedures can have the following benefits:

- Enhance the security of your application by ensuring that malicious users cannot input data that exploits a security vulnerability, such as SQL Injection, XSS, or buffer overflows.
- Enhance application resilience by rejecting invalid data on the server-side prior to processing the input.
- Enhance application usability by providing the user with feedback that allows them to correct their mistakes, while not generating verbose error messages.

The list below gives you an overview of the available data validation tags, as well as their validation type (server vs. client side) methods. For a more detailed explanation consult your ColdFusion Developers resources on the CFML language and its security features:

- **Mask, client:** Applies to cinput tags on the client-side. The use of Mask creates a Javascript or ActionScript control that verifies that input corresponds to a specified pattern. For example: nnn-*n*-nnn-*n* where “*n*” is an integer. Note, this is a client-side control that can be easily bypassed.
- **onBlur, client:** Applies to cinput and ctextarea tags. *onBlur* creates a JavaScript that runs in the browser and checks that user supplied data matches a corresponding pattern. Can be bypassed by a MITM proxy.
- **onSubmit, client:** Applies to the Web browser when the user clicks submit. Checks that the data passed from the browser corresponds to a specified pattern. Can be bypassed by MITM proxy.
- **onServer, server:** Applies to server-side data after the form is submitted. ColdFusion checks the form data of cinput and ctextarea tags and generates an error page if the data is not valid. Use this tag in conjunction with the cferror tag to specify the validation error page. Note: a failure to specify an error page will result in an information leak in your error handling routine, as ColdFusion errors are verbose.

- *IsValid*, **server**: Tests an input variable to determine if the content of the variable meets internal validation rules. The *IsValid* function returns true or false for the variable.
- *Cfparam*, **server**: Tests an input variable to determine if the variable meets validation criteria. If the variable does not meet the criteria an expression exception is generated.
- *Cfqueryparam*, **server**: Evaluates the content of a HTTP query string to validate whether the string meets validation criteria. This tag is useful for scrubbing HTTP query strings prior to further processing.

ColdFusion Scriptprotect

In addition to the data validation techniques available in the CFML language ColdFusion also provides a Scriptprotect setting to further assist in the prevention of Cross-Site Scripting. The ColdFusion administrator should configure the Scriptprotect method in Application.cfm, under the Enable Global Script Protection setting. Using this setting will help to protect user input from Cross-Site Scripting, however, use of the global Scriptprotect method should not be a substitute for individual form and parameter validation techniques.

References

OWASP Frequently Asked Questions on Web Application Security: <http://www.owasp.org/documentation/appsecfaq>
Detection of SQL Injection and Cross-site Scripting Attacks: <http://www.securityfocus.com/infocus/1768>
The Cross-Site Scripting FAQ: <http://www.cgisecurity.com/articles/xss-faq.shtml>
CERT Advisory on Malicious HTML Tags: <http://www.cert.org/advisories/CA-2000-02.html>
Cross-Site Scripting Security Exposure Executive Summary:
<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/topics/ExSumCS.asp>
Understanding the cause and effect of CSS Vulnerabilities: <http://www.technicalinfo.net/papers/CSS.html>

Remediation References

OWASP Guide to Building Secure Web Applications: <http://www.owasp.org/>
Preventing the Cross-Site Scripting Vulnerability: http://www.giac.org/practical/GSEC/Deyu_Hu_GSEC.pdf
CERT "Understanding Malicious Content Mitigation": http://www.cert.org/tech_tips/malicious_code_mitigation.html
OWASP Guide to Building Secure Web Applications and Web Services, Chapter 8: Data Validation:
<http://www.owasp.org/documentation/guide/>
How to Build an HTTP Request Validation Engine (J2EE validation with Stinger):
<http://www.owasp.org/columns/jeffwilliams/jeffwilliams2>

Attack: Non-SSL Form	Ver: 1.2.12	CWE-201	Observations: 11
Descr: Non-SSL Form is a vulnerability caused by allowing submission of sensitive form data without using SSL encryption. The SmartAttack observes and reports any form that is not submitted via SSL. The significance of such findings is dependent on the sensitivity of the submitted data.			
Severity: High HARM: 288 Total HARM: 1551			

Impact

The impact of this vulnerability is entirely dependent on the sensitivity of the data involved. Even common data associated with a person, such as address, email, phone should commonly be considered sensitive.

Accessing a Web application having a Non-SSL Form vulnerability can leak this potentially private information. An attacker with access to communication infrastructure between the user and the website can easily and automatically collect such information. Here, which information is considered as private is entirely dependent on the context set by the Web application.

For example, an attacker who has access to a shared switch or router at an Internet Cafe may be able to collect sensitive data of users of applications with this vulnerability.

Non-SSL Form Warning Findings

Assessment: Silver, **Run:** 6/6/2011 8:38:25PM, **Traversal:** [Default Spider]

Non-SSL Form**1. Warning (High, HARM: 86) at: <http://crackme.cenzic.com/Kelev/register/register.php>**

Message: Number of forms submitted without using SSL =1

```
<form method=post id=frm action=register.php name=frm >
<input size=30 value= tabindex=2 id=txtFirstName class=InputBox1 name=FirstName >
<input size=30 value= tabindex=2 id=txtLastName class=InputBox1 name=LastName >
<input size=30 value= tabindex=2 id=txtUserId class=InputBox1 name=UserId >
<input type=password maxlength=20 size=20 value= tabindex=2 id=txtpass class=InputBox1
name=Password >
<input maxlength=20 size=20 value= tabindex=2 id=txtDOB class=InputBox1 name=DOB >
<input size=49 value= tabindex=2 id=txtAddress class=InputBox1 name=Address >
<input size=30 value= tabindex=2 id=txtCity class=InputBox1 name=txtCity >
<input name=drpState class=InputBox1 >
<input maxlength=20 size=15 value= tabindex=2 id= txtTelephoneNo class=InputBox1
name=TelephoneNo >
<input maxlength=40 size=25 value= tabindex=2 id=txtEmail class=InputBox1 name=Email >
/form>
```

Non-SSL Form**2. Warning (High, HARM: 86) at: <http://crackme.cenzic.com/Kelev/register/register.php>**

Message: Number of forms submitted without using SSL =1

```
<form method=post id=frm action=register.php name=frm >
<input size=30 value= tabindex=2 id=txtFirstName class=InputBox1 name=FirstName >
<input size=30 value= tabindex=2 id=txtLastName class=InputBox1 name=LastName >
<input size=30 value= tabindex=2 id=txtUserId class=InputBox1 name=UserId >
<input type=password maxlength=20 size=20 value= tabindex=2 id=txtpass class=InputBox1
name=Password >
<input maxlength=20 size=20 value= tabindex=2 id=txtDOB class=InputBox1 name=DOB >
<input size=49 value= tabindex=2 id=txtAddress class=InputBox1 name=Address >
<input size=30 value= tabindex=2 id=txtCity class=InputBox1 name=txtCity >
<input name=drpState class=InputBox1 >
<input maxlength=20 size=15 value= tabindex=2 id= txtTelephoneNo class=InputBox1
name=TelephoneNo >
<input maxlength=40 size=25 value= tabindex=2 id=txtEmail class=InputBox1 name=Email >
/form>
```

Non-SSL Form**3. Warning (High, HARM: 86) at: <http://crackme.cenzic.com/Kelev/view/feedback.php>**

Message: Number of forms submitted without using SSL =1

```
<form method=post id=frm action=feedback.php name=frm >
<input type=TextBox1 size=30 value= tabindex=2 id=txtFirstName name=Email >
<input rows=10 cols=30 name=Feedback >
/form>
```

Non-SSL Form

4. Warning (High, HARM: 86) at: <http://crackme.cenzic.com/Kelev/view/feedback.php>

Message: Number of forms submitted without using SSL =1

```
<form method=post id=frm action=feedback.php name=frm >
<input type=TextBox1 size=30 value= tabindex=2 id=txtFirstName name=Email >
<input rows=10 cols=30 name=Feedback >
/form>
```

Non-SSL Form

5. Warning (High, HARM: 86) at: <http://crackme.cenzic.com/Kelev/view/loanrequest.php>

Message: Number of forms submitted without using SSL =1

```
<form action=updateloanrequest.php method=post id=frm name=frm >
<input size=30 value= tabindex=2 id=txtFirstName class=TextBox1 name=txtFirstName >
<input size=30 value= tabindex=2 id=txtLastName class=TextBox1 name=txtLastName >
<input size=30 value= tabindex=2 id=txtSocialSecurityNo class=TextBox1 name=txtSocialSecurityNo >
<input maxlength=20 size=20 value= tabindex=2 id=txtDOB class=TextBox1 name=txtDOB >
<input size=49 value= tabindex=2 id=txtAddress class=TextBox1 name=txtAddress >
<input size=30 value= tabindex=2 id=txtCity class=TextBox1 name=txtCity >
<input name=drpState class=TextBox1 >
<input maxlength=20 size=15 value= tabindex=2 id=txtTelephoneNo class=TextBox1
name=txtTelephoneNo >
<input maxlength=40 size=25 value= tabindex=2 id=txtEmail class=TextBox1 name=txtEmail >
<input maxlength=20 size=15 value= tabindex=2 id=txtAnnualIncome class=TextBox1
name=txtAnnualIncome >
<input class=TextBox1 name=drpLoanType id=drpLoanType >
<input type=submit value=Submit name=sendbutton1 >
/form>
```

Non-SSL Form

6. Warning (High, HARM: 86) at: <http://crackme.cenzic.com/Kelev/php/login.php>

Message: Number of forms submitted without using SSL =2

```
<form method=post name=frmlogin >
<input type=hidden value= id=hUserType name=hUserType >
/form>
<form action=login.php method=post id=frm name=frm >
<input type=hidden value= id=hLoginType name=hLoginType >
<input type=hidden value=accttransaction.php id=hPageName name=hPageName >
<input type=hidden value=Welcome to CrackMeBank Investments id=whoislog name=whoislog >
<input type=hidden value=0 id=hUserId name=hUserId >
<input type=text id=LoginName size=20 value= class=TextBox1 name=LoginName >
<input type=password id=Password size=20 class=TextBox1 name=Password >
<input type=submit value=Login name=sendbutton1 onclick=Javascript:loginclick1(); >
/form>
```

Non-SSL Form

7. Warning (High, HARM: 86) at: <http://crackme.cenzic.com/Kelev/php/login.php>

Message: Number of forms submitted without using SSL =2

```
<form method=post name=frmlogin >
<input type=hidden value= id=hUserType name=hUserType >
/form>
<form action=login.php method=post id=frm name=frm >
<input type=hidden value= id=hLoginType name=hLoginType >
<input type=hidden value=accttransaction.php id=hPageName name=hPageName >
<input type=hidden value=Welcome to CrackMeBank Investments id=whoislog name=whoislog >
<input type=hidden value=0 id=hUserId name=hUserId >
<input type=text id=LoginName size=20 value= class=TextBox1 name=LoginName >
<input type=password id=Password size=20 class=TextBox1 name=Password >
<input type=submit value=Login name=sendbutton1 onclick=Javascript:loginclick1(); >
/form>
```

Non-SSL Form

8. Warning (High, HARM: 86) at: <http://crackme.cenzic.com/Kelev/php/accttransaction.php>

Message: Number of forms submitted without using SSL =1
<form action=accttransaction.php method=post id=frm name=frm >
<input type=hidden value=7 id=hUserId name=hUserId >
<input type=text size=15 value= name=FromDate class=InputBox1 >
<input type=text size=15 value= name=ToDate class=InputBox1 >
<input type=submit value=Get Statement name=sendbutton1 onclick=Javascript: dispAcctTrn(); >
/form>

Non-SSL Form**9. Warning (High, HARM: 86) at: <http://crackme.cenzic.com/Kelev/php/loanrequestlist.php>**

Message: Number of forms submitted without using SSL =1
<form method=post id=frm name=frm >
<input type=hidden value=1 id=hUserId name=hUserId >
<input type=hidden value= id=hRequestId name=hRequestId >
/form>

Non-SSL Form**10. Warning (High, HARM: 86) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>**

Message: Number of forms submitted without using SSL =1
<form method=post id=frm name=frm >
<input type=hidden value=1 id=UserId name=hUserId >
/form>

Non-SSL Form**11. Warning (High, HARM: 691) at: URL-not-available**

Message: Limit exceeded. 18 pages found with forms submitted without SSL.
The vulnerabilities reported individually have been limited to 10 items.
To change the number of reported items allowed adjust the ReportItemLimit parameter .

Remediation Tips

HTTPS uses Secure Socket Layer (SSL) encryption between client and server to protect data confidentiality. It is a recommended practice that any secure portion of your web application designed only for HTTPS access should be inaccessible via HTTP. Ensure that forms in which potentially sensitive information is requested are served via SSL to prevent private information from being sent in a clear text format. Submit passwords via SSL to prevent exposure.

Attack: Non-SSL Password	Ver: 1.2.6	CWE-201	Observations: 2
Descr: Non-SSL Password is a vulnerability caused by a failure to submit passwords via SSL. The SmartAttack inspects all requests where passwords are submitted and reports those where SSL is not used for the submission.			
Severity: High HARM: 640 Total HARM: 1280			

Impact

An attacker sniffing traffic on a computer network may notice and consequently steal usernames and passwords if they are sent unencrypted to a Web application. The attacker may then use this information to use the accounts corresponding to it for various malicious purposes, ranging from disclosure of personal information to identity theft.

Although the use of a packet sniffer or network monitoring tool is required to capture unencrypted traffic on a network, such tools are widely available and may be easy to install. This means that the complexity involved in capturing unencrypted traffic can be very less. Depending on which place the attacker is successfully able to install a sniffer, the scope of such an attack may be large, thereby increasing the chance of the attacker succeeding to get more passwords.

Non-SSL Password Vulnerable Findings

Assessment: Silver, **Run:** 6/6/2011 8:38:25PM, **Traversal:** [Default Spider]

Non-SSL Password

1. Vulnerable (High, HARM: 640) at: <http://crackme.cenzic.com/Kelev/register/register.php>

Message: The action of a form containing a password field is not using SSL

Non-SSL Password

2. Vulnerable (High, HARM: 640) at: <http://crackme.cenzic.com/Kelev/php/changepass.php>

Message: The action of a form containing a password field is not using SSL

Remediation Tips

HTTPS uses Secure Socket Layer (SSL) encryption between client and server to protect data confidentiality. It is a recommended practice that any secure portion of your web application designed only for HTTPS access should be inaccessible via HTTP.

In addition to this, especially for login requests, it is highly recommended that requests containing username and password are sent over HTTPS, *i.e.* using SSL. To achieve this, ensure one of the following:

1. The username and password are being submitted to a URL which is explicitly an `https://` URL
2. If the username and password are being submitted to a URL relative to the URL of that page, then the page itself has an explicitly `https://` URL.

Attack: Cross Site Request Forgery	Ver: 1.0.10	Observations: 6
Descr: Cross-site Request Forgery vulnerabilities allow unauthorized requests from a victim's machine to improperly initiate transactions using an existing authenticated session. The SmartAttack, in combination with utilities used to identify session ids, identifies cases where the application is using only cookies to maintain and communicate session ids, and is therefore vulnerable.		
Severity: High HARM: 256 Total HARM: 462		

Impact

If a Web application is vulnerable to CSRF, an attacker can make the victim's browser send unauthorized HTTP requests on behalf of the victim without her knowledge. This allows an attacker to perform all the legitimate actions which a legitimate user can perform after a log-in. All Web applications which use only HTTP cookies to store session information are vulnerable to CSRF.

Such applications expose their users to a typical CSRF attack every time they log in to the application. For example, a banking application which is vulnerable to CSRF may allow an attacker to transfer funds from a victim's account to his own account. If the victim is using an e-mail application vulnerable to CSRF, the attacker can send malicious e-mails using the victim's account without her knowledge.

Cross Site Request Forgery Warning Findings

Assessment: Silver, **Run:** 6/6/2011 8:38:25PM, **Traversal:** [Default Spider]

Cross Site Request Forgery

1. Warning (High, HARM: 77) at: <http://crackme.cenzic.com/Kelev/php/accttransaction.php>

Message: There is no session token present in GET/POST parameter or in path parameter to uniquely identify legitimate private request.

Cross Site Request Forgery

2. Warning (High, HARM: 77) at: <http://crackme.cenzic.com/Kelev/php/loanrequestlist.php>

Message: There is no session token present in GET/POST parameter or in path parameter to uniquely identify legitimate private request.

Cross Site Request Forgery

3. Warning (High, HARM: 77) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: There is no session token present in GET/POST parameter or in path parameter to uniquely identify legitimate private request.

Cross Site Request Forgery

4. Warning (High, HARM: 77) at: <http://crackme.cenzic.com/Kelev/php/transfer.php>

Message: There is no session token present in GET/POST parameter or in path parameter to uniquely identify legitimate private request.

Cross Site Request Forgery

5. Warning (High, HARM: 77) at: <http://crackme.cenzic.com/Kelev/php/changepass.php>

Message: There is no session token present in GET/POST parameter or in path parameter to uniquely identify legitimate private request.

Cross Site Request Forgery

6. Warning (High, HARM: 77) at: URL-not-available

Message: Limit exceeded. 6 Private requests found with cross site request forgery vulnerability. The vulnerabilities reported individually have been limited to 5 items. To change the number of reported items allowed, adjust the parameter value.

Remediation Tips

Use of persistent and transient authentication method

Web application should use both persistent authentication method (e.g. Cookie or HTTP authentication) as well as transient authentication method (or a hidden field provided on every form) to defend against this attack.

This can be done using following ways.

1. Add a per request nonce to URL and all forms in addition to the standard session.
2. Add a per session nonce to URL and all forms.
3. Add hash (session-id, function name, server-side secret) to URL and all forms.

Accept only POST requests

Although JavaScript can be used to forge POST requests with ease, requests that execute the business logic should always use POST. The reason is that POST does not leave a trail of variable data in web server and proxy server logs, while GET does leave such a trail. Therefore, it is best to use POST when coding in a defence-in-depth approach.

Please, refer to “Forms Submitted without Using Post” and “GET for POST” SmartAttacks for more details.

HTTP Referrer Check

Checking HTTP referrer details can help in mitigating the attack but this check does not provide bullet proof solution. There are two reasons for this. First is, browser sometimes omit the referrer header because of user's privacy settings and second is, if CSRF attack is used in combination with XSS on the original site, then this mechanism will not provide any protection. But, still it is recommended that application should check HTTP referrer header to validate the request.

Please, refer to “Phishing Referrer Trust” SmartAttack for more details.

References

OWASP Top Ten 2007

https://www.owasp.org/index.php/Top_10_2007 <<http://www.owasp.org/documentation/top10.html>>

OWASP: CSRF Guard

http://www.owasp.org/index.php/CSRF_Guard <<http://www.cgisecurity.com/owasp/html/>>

[Secure Coding: Principles & Practices](#)

<<http://www.securecoding.org/>>

Attack: Form Caching	Ver: 1.1.20	CWE-525	Observations: 11
Descr: Form Caching is a vulnerability caused by allowing the browser to cache sensitive form field values which could later be displayed to a different person using the same browser. The SmartAttack observes caching directives specified for pages, and reports each page that contains one or more forms while allowing such caching. The significance of such findings is dependent on the sensitivity of the cached data.			
Severity: Medium HARM: 640 Total HARM: 4992			

Impact

The impact of this vulnerability is entirely dependent on the sensitivity of the data involved. Even common data associated with a person, such as address, email, and phone should commonly be considered sensitive. Accessing a Web application having a Form Caching vulnerability will cause content in HTML forms to be stored on the machine from which the browsing happens. An attacker who has access to such a machine may be able to make the browser give away those cached forms by visiting its cache. If the previously filled form entries contain sensitive information like Credit Card numbers, usernames *etc.* then the attacker can easily steal and misuse such information.

An attacker who has access to a number of shared machines, such as at an Internet Cafe or by posing as a computer maintenance professional, he may be able to collect sensitive data of users of applications with this vulnerability. It may also be possible for an attacker to install malware on victims' computers which will compromise the browser, thus making the attacker's presence at the machine unnecessary.

Form Caching Warning Findings

Assessment: Silver, **Run:** 6/6/2011 8:38:25PM, **Traversal:** [Default Spider]

Form Caching**1. Warning (Medium, HARM: 192) at: <http://crackme.cenzic.com/Kelev/register/register.php>**

Message: No caching directives found.
The application should use BOTH 'Pragma:no-cache' as well as 'Cache-Control:no-Store,no-Cache' headers to prevent caching.

Number of forms in the page cache =2

```
<form method=post name=frmlogin>
</form>
```

```
<form method=post id=frm action=register.php name=frm>
<input size=30 tabindex=2 id=txtFirstName class=InputBox1 name=FirstName >
<input size=30 tabindex=2 id=txtLastName class=InputBox1 name=LastName >
<input size=30 tabindex=2 id=txtUserId class=InputBox1 name=UserId >
<input type=password maxlength=20 size=20 tabindex=2 id=txtpass class=InputBox1 name=Password >
<input maxlength=20 size=20 tabindex=2 id=txtDOB class=InputBox1 name=DOB >
<input size=49 tabindex=2 id=txtAddress class=InputBox1 name=Address >
<input size=30 tabindex=2 id=txtCity class=InputBox1 name=txtCity >
<input name=drpState class=InputBox1 >
<input maxlength=20 size=15 tabindex=2 id=txtTelephoneNo class=InputBox1 name=TelephoneNo >
```

Form Caching**2. Warning (Medium, HARM: 192) at: <http://crackme.cenzic.com/Kelev/register/register.php>**

Message: No caching directives found.
The application should use BOTH 'Pragma:no-cache' as well as 'Cache-Control:no-Store,no-Cache' headers to prevent caching.

Number of forms in the page cache =2

```
<form method=post name=frmlogin>
</form>
```

```
<form method=post id=frm action=register.php name=frm>
<input size=30 tabindex=2 id=txtFirstName class=InputBox1 name=FirstName >
```

```

<input size=30 tabindex=2 id=txtLastName class=InputBox1 name=LastName >
<input size=30 tabindex=2 id=txtUserId class=InputBox1 name=UserId >
<input type=password maxlength=20 size=20 tabindex=2 id=txtpass class=InputBox1 name=Password
>
<input maxlength=20 size=20 tabindex=2 id=txtDOB class=InputBox1 name=DOB >
<input size=49 tabindex=2 id=txtAddress class=InputBox1 name=Address >
<input size=30 tabindex=2 id=txtCity class=InputBox1 name=txtCity >
<input name=drpState class=InputBox1 >
<input maxlength=20 size=15 tabindex=2 id=txtTelephoneNo class=InputBox1 name=TelephoneNo >

```

Form Caching

3. Warning (Medium, HARM: 192) at: <http://crackme.cenzic.com/Kelev/view/feedback.php>

Message: No caching directives found.
The application should use BOTH 'Pragma:no-cache' as well as 'Cache-Control:no-Store,no-Cache' headers to prevent caching.

Number of forms in the page cache =2

```

<form method=post name=frmlogin>
</form>

```

```

<form method=post id=frm action=feedback.php name=frm>
<input type=InputBox1 size=30 tabindex=2 id=txtFirstName name=Email >
<input rows=10 cols=30 name=Feedback >
</form>

```

Form Caching

4. Warning (Medium, HARM: 192) at: <http://crackme.cenzic.com/Kelev/view/feedback.php>

Message: No caching directives found.
The application should use BOTH 'Pragma:no-cache' as well as 'Cache-Control:no-Store,no-Cache' headers to prevent caching.

Number of forms in the page cache =2

```

<form method=post name=frmlogin>
</form>

```

```

<form method=post id=frm action=feedback.php name=frm>
<input type=InputBox1 size=30 tabindex=2 id=txtFirstName name=Email >
<input rows=10 cols=30 name=Feedback >
</form>

```

Form Caching

5. Warning (Medium, HARM: 192) at: <http://crackme.cenzic.com/Kelev/view/kelev2.php>

Message: No caching directives found.
The application should use BOTH 'Pragma:no-cache' as well as 'Cache-Control:no-Store,no-Cache' headers to prevent caching.

Number of forms in the page cache =1

```

<form method=post name=frmlogin>
</form>

```

Form Caching

6. Warning (Medium, HARM: 192) at: <http://crackme.cenzic.com/Kelev/loans/carloanmain.php>

Message: No caching directives found.
The application should use BOTH 'Pragma:no-cache' as well as 'Cache-Control:no-Store,no-Cache' headers to prevent caching.

Number of forms in the page cache =1

```

<form method=post name=frmlogin>
</form>

```

Form Caching

7. Warning (Medium, HARM: 192) at: <http://crackme.cenzic.com/Kelev/view/rate.php>

Message: No caching directives found.
The application should use BOTH 'Pragma:no-cache' as well as 'Cache-Control:no-Store,no-Cache' headers to prevent caching.

Number of forms in the page cache =1

```
<form method=post name=frmlogin>
</form>
```

Form Caching

8. Warning (Medium, HARM: 192) at: <http://crackme.cenzic.com/Kelev/view/loanrequest.php>

Message: No caching directives found.
The application should use BOTH 'Pragma:no-cache' as well as 'Cache-Control:no-Store,no-Cache' headers to prevent caching.

Number of forms in the page cache =2

```
<form method=post name=frmlogin>
</form>
```

```
<form action=updateloanrequest.php method=post id=frm name=frm>
<input size=30 tabindex=2 id=txtFirstName class=InputBox1 name=txtFirstName >
<input size=30 tabindex=2 id=txtLastName class=InputBox1 name=txtLastName >
<input size=30 tabindex=2 id=txtSocialScurityNo class=InputBox1 name=txtSocialScurityNo >
<input maxlength=20 size=20 tabindex=2 id=txtDOB class=InputBox1 name=txtDOB >
<input size=49 tabindex=2 id=txtAddress class=InputBox1 name=txtAddress >
<input size=30 tabindex=2 id=txtCity class=InputBox1 name=txtCity >
<input name=drpState class=InputBox1 >
<input maxlength=20 size=15 tabindex=2 id=txtTelephoneNo class=InputBox1 name=txtTelephoneNo >
<input maxlength=40 size=25 tabindex=2 id=txtEmail class=InputBox1 name=txtEmail >
<input maxlength=20 size=15 tabindex=2 id=txtAnnualIncome class=InputBox1 name=txtAnnualIncome >
```

Form Caching

9. Warning (Medium, HARM: 192) at: <http://crackme.cenzic.com/Kelev/view/updatesloanrequest.php>

Message: No caching directives found.
The application should use BOTH 'Pragma:no-cache' as well as 'Cache-Control:no-Store,no-Cache' headers to prevent caching.

Number of forms in the page cache =1

```
<form method=post name=frmlogin>
</form>
```

Form Caching

10. Warning (Medium, HARM: 192) at: <http://crackme.cenzic.com/Kelev/loans/homeloan.php>

Message: No caching directives found.
The application should use BOTH 'Pragma:no-cache' as well as 'Cache-Control:no-Store,no-Cache' headers to prevent caching.

Number of forms in the page cache =1

```
<form method=post name=frmlogin>
</form>
```

Form Caching

11. Warning (Medium, HARM: 3072) at: URL-not-available

Message: Limit exceeded. 16 more Warning item(s) were reported.
The total items reported individually have been limited to 10 items.
To change the number of reported items allowed, adjust the ReportItemLimit parameter.

Remediation Tips

This SmartAttack analyzes forms within the application to determine which forms allow content caching. When a form allows content to be cached, values previously entered into the form will be stored in the client's web browser. You can disable caching on a page by setting the "Pragma: No-cache" and "Cache-control: No-cache, No-Store" HTTP Header values. If you are using Active Server Pages, you can prevent the caching of form content with the following script:

```
<% Response.CacheControl = "no-cache,no-Store" %>  
<% Response.AddHeader "Pragma", "no-cache" %>  
<% Response.Expires = -1 %>
```

You can also set cache control in the HTML Header using META Tags as follows:

```
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">  
<META HTTP-EQUIV="Cache-Control" CONTENT="no-cache, no-Store">
```

NOTE: Be advised that if you use this method, pages might still be cached in Internet Explorer's temporary internet files. In order to resolve this issue, you must add an additional header to the page, as discussed in the Microsoft Knowledge Base article, *Pragma: No-cache Tag May Not Prevent Page from Being Cached* (<http://support.microsoft.com/default.aspx?kbid=222064>).

References

"Pragma: No-cache" Tag May Not Prevent Page from Being Cached
<http://support.microsoft.com/default.aspx?kbid=222064>:

Attack: SQL Error Message	Ver: 1.5.7	CWE-209	Observations: 33
Descr: SQL Error Message, or SQL Exception, is a vulnerability caused by a Web application inserting user input in a SQL query without validation and failing to suppress error messages that may result from use of such input. This SmartAttack injects SQL characters in order to cause errors in SQL execution, and looks for evidence of such errors.			
Severity: Medium HARM: 120 Total HARM: 3960			

Impact

An attacker might gain administrative control of your web application or database by using specially crafted SQL queries. It is also possible to gain remote access to restricted information via queries to your database.

A SQL Error Message vulnerability helps the attacker in formulating the correct query depending on information disclosed in the error message. Such an error message may also disclose information about the deployment of the database, such as the database server used, server-side technology used, *etc.*

Many Web applications use SQL databases to store important information which can be disclosed through error messages deliberately obtained by an attacker. For example, an error generated by a Web-based form for searching employee details of an organization may lead to disclosure of personal information stored along with. Likewise, errors generated by a Web-based inventory management app may divulge important information about products yet to be released.

SQL Error Message Vulnerable Findings

Assessment: Silver, **Run:** 6/6/2011 8:38:25PM, **Traversal:** [Default Spider]

SQL Error Message

1. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/view/updateloanrequest.php>

Message: SQL Error Message vulnerability found
Injectable request #: 3
Injected item: POST: txtAnnualIncome
Injection value: 'OR
Detection value: error in your sql syntax

SQL Error Message

2. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/accttransaction.php>

Message: SQL Error Message vulnerability found
Injectable request #: 6
Injected item: POST: ToDate
Injection value: 'OR
Detection value: warning:

SQL Error Message

3. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/accttransaction.php>

Message: SQL Error Message vulnerability found
Injectable request #: 6
Injected item: POST: FromDate
Injection value: 'OR
Detection value: warning:

SQL Error Message

4. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 10
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

5. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 12
Injected item: POST: hRequestId
Injection value: '

Detection value: warning:

SQL Error Message

6. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 13
Injected item: POST: hRequestId
Injection value: '
Detection value: warning:

SQL Error Message

7. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 14
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

8. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 15
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

9. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 16
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

10. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 17
Injected item: POST: hRequestId
Injection value: '
Detection value: warning:

SQL Error Message

11. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 18
Injected item: POST: hRequestId
Injection value: '
Detection value: warning:

SQL Error Message

12. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 19
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

13. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 20
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

14. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 21
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

15. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 22
Injected item: POST: hRequestId
Injection value: '
Detection value: warning:

SQL Error Message

16. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 23
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

17. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/loanrequestdetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 24
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

18. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 27
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

19. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 28
Injected item: POST: hRequestId
Injection value: ' 1
Detection value: warning:

SQL Error Message

20. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 29
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

21. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 30
Injected item: POST: hRequestId
Injection value: '\
Detection value: warning:

SQL Error Message

22. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 31

Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

23. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 32
Injected item: POST: hRequestId
Injection value: '
Detection value: warning:

SQL Error Message

24. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 33
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

25. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 34
Injected item: POST: hRequestId
Injection value: ' 1
Detection value: warning:

SQL Error Message

26. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 35
Injected item: POST: hRequestId
Injection value: '
Detection value: warning:

SQL Error Message

27. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 36
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

28. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 37
Injected item: POST: hRequestId
Injection value: '\
Detection value: warning:

SQL Error Message

29. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 38
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

30. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 39
Injected item: POST: hRequestId
Injection value: '\
Detection value: warning:

SQL Error Message

31. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 40
Injected item: POST: hRequestId
Injection value: ' 1
Detection value: warning:

SQL Error Message

32. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/approveloanpagedetail.php>

Message: SQL Error Message vulnerability found
Injectable request #: 41
Injected item: POST: hRequestId
Injection value: 'OR
Detection value: warning:

SQL Error Message

33. Vulnerable (Medium, HARM: 120) at: <http://crackme.cenzic.com/Kelev/php/login.php>

Message: SQL Error Message vulnerability found
Injectable request #: 47
Injected item: POST: hUserId
Injection value: \"
Detection value: warning:

Remediation Tips

The following recommendations will help to mitigate the risk of SQL injection attacks:

- Monitor your production applications for the latest security vulnerabilities and bugs.
- Keep up to date on patches and security fixes as they are released by the vendor or maintainer.
- Limit the types of characters and strings that can be passed as application parameters by configuring native application filters.
- Avoid using external SQL interpreters.
- Audit your applications frequently for points where HTML input can access interpreters.
- Ensure proper input validation is performed for user supplied data, regardless of the application's relationship to a back-end database.
- Avoid use of dynamic SQL or PL/SQL and use bound variables whenever possible.
- Enforce strict limitations on the rights to database access.
- Remove any sample applications or demo scripts that allow remote database queries.

Use structured exception handling to catch errors and prevent their exposure to the client.

If errors occur while the user is connecting to the database, be sure that you provide only limited information to the user. Detailed error SQL error messages help a malicious user reverse engineer your database structure and determine the presence or absence of database tables. Such errors also help them to refine their attacks against your database.

In order to protect against exposing sensitive information you need to implement specific security measures to catch SQL based errors and redirect users to a generic error page.

PHP

For PHP, you can control whether or not detailed error messages are presented to the client by setting the following directives. The configuration below will log errors to your server's error log, but not display the errors to the client:

```
log_errors = On
display_errors = Off
```

See also the PHP Manual's section Error Handling and Logging Functions: <http://us2.php.net/errorfunc>.

ASP.NET

Implementing a global exception handling mechanism can enhance the security and usability of a web application by redirecting users to a custom error page whenever handled and unhandled exceptions occur. With ASP.NET you can handle errors either on the Application level, via `global.asax`, or on the page level via specific error handling code.

Also employ mechanisms to redirect users to a custom error page whenever an exception occurs.

To set up a custom error page it is necessary to edit `web.config` and add a `customerrors` tag if it does not already exist.

```
<configuration>
  <customerrors mode="On" defaultredirect="error.aspx" />
</configuration>
```

The mode attribute has three settings: "On," "Off," and "RemoteOnly." When set to "On" the custom error page will be used anytime an exception occurs. When the attribute is set to RemoteOnly, it will only redirect to the custom error page in the event the client does not originate from the local machine. Local users will continue to see detailed error messages.

Structured Exception Handling

Use code-level structured exception handling to trap exceptions and prevent diagnostic or detailed error messages from being displayed. Implement a global exception handling mechanism in `global.asax` to catch any exceptions that are not handled in your code.

Java

There are two kinds of exceptions in Java, namely checked and unchecked exceptions.

If the client code cannot do anything, then make it an unchecked exception. If the client code will take some useful recovery action based on information in the exception, make it a checked exception.

Vulnerable code:

An example of vulnerable code is below.

```
public void dataAccess(){
  try{
    ..some code that throws SQLException
  }catch(SQLException ex){
    throw new SQLException(ex);
  }
}
```

In this example, the details of the exception are propagated, which might cause a leak of private information.

Secure code:

1. Encapsulation

Never let implementation-specific exceptions propagate to the higher layers. For example, do not propagate SQLException from the data access code to the business objects layer.

```
public void dataAccess(){
    try{
        ..some code that throws SQLException
    }catch(SQLException ex){
        throw new RuntimeException(ex);
    }
}

public void dataAccessCode(){
    try{
        ..some code that throws LDAPException
    }catch(javax.naming.directory.Exception ex){
        throw new RuntimeException(ex);
    }
}
```

This example converts SQLException or LDAPException to RuntimeException. If SQLException or LDAPException occurs, the catch clause throws a new RuntimeException. The execution thread is suspended and the exception gets reported. However, this does not corrupt the business object layer with unnecessary exception handling, since specific exceptions related to LDAP or SQL should be handled in the tier of code that deals with it.

In addition to the above scheme, production systems should NEVER use the default error page. A custom error page should be created using the <error-page> directive in web.xml. For example, place the following within your <web-app> to catch all JSP compilation/runtime exceptions and redirect them to your custom error page:

```
<error-page><exception >java.lang.Exception</exception> <location>/safeErrorPage.html</location></error-page>
```

More information can be found at:

http://www.oracle.com/technology/sample_code/tech/java/codesnippet/servlets/handlingServletExceptions/handlingServletExceptions.html#how1

http://www.objectsource.com/j2eechapters/Ch18-Exception_Handling.htm

2. Clean up

Resources like database connections or network connections should be cleaned up. Even if they are only unchecked exceptions, clean up the resources using try - finally blocks.

```
public void dataAccess(){
    Connection conn = null;
    try{
        conn = getConnection();
        ..some code that throws SQLException
    }catch(SQLException ex){
        logger.error("error in sql");
    } finally{
        DBUtil.closeConnection(conn);
    }
}

class DBUtil{
    public static void closeConnection
    (Connection conn){
        try{
            conn.close();
        } catch(SQLException ex){
            logger.error("Cannot close connection");
            throw new RuntimeException(ex);
        }
    }
}
```

```

    }
  }
}

```

3. Identify sources of exception

Identify all possible uncaught runtime exceptions early in the development lifecycle, then examine and modify the code to ensure that it does not provide any opportunities for hackers.

Disable Trace Output

Also in web.config, set pageOutput="false" on the <trace> element to prevent trace output from being displayed. You should make the changes to a machine level web.config file to ensure that the configuration is used for all your applications on the server. Place the <trace> element inside a <location> element and set allowOverride to false:

```

<location path="" allowOverride="false">
  <system.web>
    <trace pageOutput="false" ... />
  </system.web>
</location>

```

Coldfusion

Generic

ColdFusion provides a number of ways to filter or validate user input. The security measures span both client-side and server-side filtering. Where possible, implement your security controls on the server-side to avoid tampering of your controls via a Man-In-The-Middle (MITM) proxy. An attacker can easily bypass client-side controls by using such a program to modify the underlying HTTP Request as it passes between the proxy and the web application. Additionally, we recommend that sites using ColdFusion should configure their application using the recommended security features of the Adobe ColdFusion 8 Developers Guide, or the guide relevant to your version of ColdFusion. ColdFusion Data validation allows you to control the type of data that is allowed as well as to ensure that user-supplied data corresponds to the correct form. Attentive data validation procedures can have the following benefits:

- Enhance the security of your application by ensuring that malicious users cannot input data that exploits a security vulnerability, such as SQL Injection, XSS, or buffer overflows.
- Enhance application resilience by rejecting invalid data on the server-side prior to processing the input.
- Enhance application usability by providing the user with feedback that allows them to correct their mistakes, while not generating verbose error messages.

The list below gives you an overview of the available data validation tags, as well as their validation type (server vs. client side) methods. For a more detailed explanation consult your ColdFusion Developers resources on the CFML language and its security features:

- *Mask*, **client**: Applies to cinput tags on the client-side. The use of Mask creates a Javascript or ActionScript control that verifies that input corresponds to a specified pattern. For example: nnn-*n*-nnn-*n* where “*n*” is an integer. Note, this is a client-side control that can be easily bypassed.
- *onBlur*, **client**: Applies to cinput and ctextarea tags. *onBlur* creates a JavaScript that runs in the browser and checks that user supplied data matches a corresponding pattern. Can be bypassed by a MITM proxy.
- *onSubmit*, **client**: Applies to the Web browser when the user clicks submit. Checks that the data passed from the browser corresponds to a specified pattern. Can be bypassed by MITM proxy.
- *onServer*, **server**: Applies to server-side data after the form is submitted. ColdFusion checks the form data of cinput and ctextarea tags and generates an error page if the data is not valid. Use this tag in conjunction with the cferror tag to specify the validation error page. Note: a failure to specify an error page will result in an information leak in your error handling routine, as ColdFusion errors are verbose.
- *IsValid*, **server**: tests an input variable to determine if the content of the variable meets internal validation rules. The *IsValid* function returns true or false for the variable.
- *Cfparam*, **server**: tests an input variable to determine if the variable meets validation criteria. If the variable does not meet the criteria an expression exception is generated.
- *Cfqueryparam*, **server**: evaluates the content of a HTTP query string to validate whether the string meets validation criteria. This tag is useful for scrubbing HTTP query strings prior to further processing.

SQL Security

When data from a user-supplied parameter is passed within a SQL query, if proper security mechanisms are not in place a malicious user can modify the underlying query and conduct SQL Injection attacks.

Vulnerable Code

Consider the code below, which is vulnerable to SQL Injection:

```
<cfquery name="GetMembers" datasource="example">
  SELECT FirstName, LastName,
  From Members
  WHERE EmpID='#Form.EmpID#'
</cfquery>
```

The sample code above could be called by a malicious URL in the following way:

```
Server/script.cfm?EmpID=0%20Malicious%20SQL%20Query
```

Which would result in following SQL Expression, as if the form consisted of:

```
<cfquery name="GetMembers" datasource="example">
  SELECT FirstName, LastName,
  From Members
  WHERE EmpID= 0 MALICIOUS SQL QUERY
</cfquery>
```

An attacker could abuse this vulnerability to delete all data from tables or read or modify the underlying database, or authenticate without a username or password.

Secure Code

The **cfqueryparam** tag can be used to evaluate the string parameters prior to processing by the database. You can specify the type of data for the corresponding database columns used in the select statement and reject input of invalid data types. Cfqueryparam, when used in conjunction with **cfsqltype**, can also use a wide range of input validation functions, as well as constrain the length of input on the server-side, preventing SQL Injection.

For example:

```
<cfquery name="GetFirstName" datasource="example">
  SELECT * FROM Employees
  WHERE EmpID = <cfqueryparam value = "#EmpID#"
  Cfsqltype = "cf_sql_integer">
</cfquery>
```

This code thus prevents SQL Injection of the EmpID parameter value by enforcing an integer data type. Note that you must enforce proper error handling or this security measure could introduce an information leak via verbose SQL Error Messages.

When manipulating strings, you can use other security measures to enforce a string data type, limit maximum length of the string, and escape the string values within single quotations to ensure it is examined as a single value by the database. For additional information on the use of cfqueryparam and cfsqltype, consult your ColdFusion documentation. In addition to the use of these measures, you should also enforce input validation to reject queries that contain special characters. This prevents an attacker from manipulating SQL expressions via special characters used by delimiters in database queries.

References

OWASP Guide to Building Secure Web Applications: <http://www.owasp.org/>
 OWASP Frequently Asked Questions on Web Application Security: <http://www.owasp.org/documentation/appsecfaq>
 Detection of SQL Injection and Cross-site Scripting Attacks: <http://www.securityfocus.com/infocus/1768>
 SQL Injection Walkthrough: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
 SQL Injection in Oracle: <http://www.integrigy.com/info/IntegrigyIntrotoSQLInjectionAttacks.pdf>

Remediation References

SQL Server Security Site: <http://www.sqlsecurity.com>
 Detecting SQL Injection in Oracle: <http://www.securityfocus.com/infocus/1714>
 Thwarting SQL Web Hacks: Fixing insecure Web code is up to site owners:
<http://www.varbusiness.com/sections/News/breakingnews.asp?ArticleID=49029>

Attack: HTML & JavaScript Comments	Ver: 1.2.19	CWE-615	Observations: 12
Descr: HTML & JavaScript Comments vulnerabilities are disclosures of internal information caused by any comments in HTML or JavaScript present in pages. The SmartAttack examines responses containing HTML or JavaScript, and reports any comments observed therein. The significance of such vulnerabilities depends on the specific disclosures involved and the policies of the organization.			
Severity: Medium HARM: 60 Total HARM: 1680			

Impact

An attacker may get sensitive information out of a Web application or even gain unintended control or access if he is aware of implementation details, unexposed URLs or even other simple details such as developers' identities.

Unnecessary comments in the HTML source of the pages or in the Javascript code used will help an attacker become aware of one or more of the aspects mentioned above. Such comments may also disclose other information such as the tools used to create the Web application, names of various pieces of code of the backend, *etc.*

Sometimes, developers habitually add their personal details or details about the code otherwise not guessable in comments. For example, if an attacker finds the details of the main developer of a banking application, he may use them to gain more information about the application to ultimately exploit otherwise unknown vulnerabilities for monetary gains.

HTML & JavaScript Comments Vulnerable Findings

Assessment: Silver, **Run:** 6/6/2011 8:38:25PM, **Traversal:** [Default Spider]

HTML & JavaScript Comments**1. Vulnerable (Medium, HARM: 60) at: <http://crackme.cenzic.com:80/Kelev/scripts/kelevcommon.js>**

Message: Potentially insecure comments found with the following keywords:
uncomment,

No. of JavaScript comments in page = 22

```

1. //alert(getErrorMessage(allowNegative, allowDecimal));
2. //return date string in mm/dd/yyyy format by referring to system date format.
3. // case "dd/mmm/yy" :
4. // lretDateStr = sDate.substr(3,2) + "/" + Mid(MonthName(CLng(Mid(sDate, 1, 2))), 1, 3) + "/" + Mid(sDate, 9,
2)
5. // break;
6. // case "dd/mmm/yyyy" :
7. // lretDateStr = Mid(sDate, 4, 2) + "/" + Mid(MonthName(CLng(Mid(sDate, 1, 2))), 1, 3) + "/" + Mid(sDate, 7)
8. // break;
9. // case "dd-mmm-yy" :
10. // lretDateStr = Mid(sDate, 4, 2) + "-" + Mid(MonthName(CLng(Mid(sDate, 1, 2))), 1, 3) + "-" + Mid(sDate, 9,
2)
11. // break;
12. // case "dd-mmm-yyyy" :
13. // lretDateStr = Mid(sDate, 4, 2) + "-" + Mid(MonthName(CLng(Mid(sDate, 1, 2))), 1, 3) + "-" + Mid(sDate, 7)
14. // break;

```

HTML & JavaScript Comments**2. Vulnerable (Low, HARM: 60) at: <http://crackme.cenzic.com/Kelev/register/register.php>**

Message: No. of comments in page = 20

```

1. <!-- TOP HORIZONTAL BAR -->
2. <!-- USER NAME AND LOGO -->
3. <!-- USER NAME AND LOGO END-->
4. <!-- FEEDBACK AND OTHER LINKS TABLE -->
5. <!-- FEEDBACK AND OTHER LINKS TABLE END-->
6. <!-- TOP HORIZONTAL BAR END -->
7. <!-- LEFT SIDE MAIN MENU TABLE -->
8. <!-- LEFT SIDE MAIN MENU TABLE END -->
9. <!-- BILLS ONLINE TABLE -->
10. <!-- BILLS ONLINE TABLE END -->

```

```

11. <!-- MAIN AREA FORM TABLE -->
12. <!-- CAR LOAN SECTION -->
13. <!-- CAR LOAN SECTION END-->
14. <!-- HOME LOAN SECTION -->
15. <!-- HOME LOAN SECTION END-->
16. <!-- BUTTONS SUBMIT AND CANCEL -->
17. <!-- <tr>

```

```
<td></td>
```

HTML & JavaScript Comments

3. Vulnerable (Low, HARM: 60) at: <http://crackme.cenzic.com/Kelev/view/feedback.php>

Message: No. of comments in page = 19

```

1. <!-- TOP HORIZONTAL BAR -->
2. <!-- USER NAME AND LOGO -->
3. <!-- USER NAME AND LOGO END-->
4. <!-- FEEDBACK AND OTHER LINKS TABLE -->
5. <!-- FEEDBACK AND OTHER LINKS TABLE END-->
6. <!-- TOP HORIZONTAL BAR END -->
7. <!-- LEFT SIDE MAIN MENU TABLE -->
8. <!-- LEFT SIDE MAIN MENU TABLE END -->
9. <!-- BILLS ONLINE TABLE -->
10. <!-- BILLS ONLINE TABLE END -->
11. <!-- MAIN AREA FORM TABLE -->
12. <!-- CAR LOAN SECTION -->
13. <!-- CAR LOAN SECTION END-->
14. <!-- HOME LOAN SECTION -->
15. <!-- HOME LOAN SECTION END-->
16. <!-- BUTTONS SUBMIT AND CANCEL -->
17. <!-- BUTTONS SUBMIT AND CANCEL END -->
18. <!-- MAIN AREA FORM TABLE END -->
19. <!-- FOOTER ROW -->

```

HTML & JavaScript Comments

4. Vulnerable (Low, HARM: 60) at: <http://crackme.cenzic.com/Kelev/view/kelev2.php>

Message: No. of comments in page = 27

```

1. <!-- TOP HORIZONTAL BAR -->
2. <!-- USER NAME AND LOGO -->
3. <!-- USER NAME AND LOGO END-->
4. <!-- FEEDBACK AND OTHER LINKS TABLE -->
5. <!-- FEEDBACK AND OTHER LINKS TABLE END-->
6. <!-- TOP HORIZONTAL BAR END -->
7. <!-- LEFT SIDE MAIN MENU TABLE -->
8. <!-- LEFT SIDE MAIN MENU TABLE END -->
9. <!-- BILLS ONLINE TABLE -->
10. <!-- BILLS ONLINE TABLE END -->
11. <!-- WHOLE CENTRAL DATA AREA -->
12. <!-- CAR LOAN SECTION -->
13. <!-- CAR LOAN SECTION END-->
14. <!-- HOME LOAN SECTION -->
15. <!-- HOME LOAN SECTION END-->
16. <!-- PRE APPROVED LOAN SECTION -->
17. <!-- HPRE APPROVED LOAN SECTION -->
18. <!-- REST ASSURE SECTION -->
19. <!-- REST ASSURE SECTION END-->

```

HTML & JavaScript Comments

5. Vulnerable (Low, HARM: 60) at: <http://crackme.cenzic.com/Kelev/loans/carloanmain.php>

Message: No. of comments in page = 11

```

1. <!-- TOP HORIZONTAL BAR -->
2. <!-- USER NAME AND LOGO -->
3. <!-- USER NAME AND LOGO END-->
4. <!-- FEEDBACK AND OTHER LINKS TABLE -->
5. <!-- FEEDBACK AND OTHER LINKS TABLE END-->
6. <!-- TOP HORIZONTAL BAR END -->
7. <!-- LEFT SIDE MAIN MENU TABLE -->

```

8. <!-- LEFT SIDE MAIN MENU TABLE END -->
9. <!-- BILLS ONLINE TABLE -->
10. <!-- BILLS ONLINE TABLE END -->
11. <!-- FOOTER ROW -->

HTML & JavaScript Comments

6. Vulnerable (Low, HARM: 60) at: <http://crackme.cenzic.com/Kelev/view/rate.php>

- Message: No. of comments in page = 16
1. <!-- TOP HORIZONTAL BAR -->
 2. <!-- USER NAME AND LOGO -->
 3. <!-- USER NAME AND LOGO END-->
 4. <!-- FEEDBACK AND OTHER LINKS TABLE -->
 5. <!-- FEEDBACK AND OTHER LINKS TABLE END-->
 6. <!-- TOP HORIZONTAL BAR END -->
 7. <!-- LEFT SIDE MAIN MENU TABLE -->
 8. <!-- LEFT SIDE MAIN MENU TABLE END -->
 9. <!-- BILLS ONLINE TABLE -->
 10. <!-- BILLS ONLINE TABLE END -->
 11. <!-- #EndEditable -->
 12. <!-- #BeginEditable "NCUA" -->
 13. <!-- #EndEditable -->
 14. <!-- #BeginEditable "FooterText" -->
 15. <!-- #EndEditable -->
 16. <!-- FOOTER ROW -->

HTML & JavaScript Comments

7. Vulnerable (Low, HARM: 60) at: <http://crackme.cenzic.com/Kelev/view/loanrequest.php>

- Message: No. of comments in page = 21
1. <!-- TOP HORIZONTAL BAR -->
 2. <!-- USER NAME AND LOGO -->
 3. <!-- USER NAME AND LOGO END-->
 4. <!-- FEEDBACK AND OTHER LINKS TABLE -->
 5. <!-- FEEDBACK AND OTHER LINKS TABLE END-->
 6. <!-- TOP HORIZONTAL BAR END -->
 7. <!-- LEFT SIDE MAIN MENU TABLE -->
 8. <!-- LEFT SIDE MAIN MENU TABLE END -->
 9. <!-- BILLS ONLINE TABLE -->
 10. <!-- BILLS ONLINE TABLE END -->
 11. <!-- MAIN AREA FORM TABLE -->
 12. <!-- CAR LOAN SECTION -->
 13. <!-- CAR LOAN SECTION END-->
 14. <!-- HOME LOAN SECTION -->
 15. <!-- HOME LOAN SECTION END-->
 16. <!-- BUTTONS SUBMIT AND CANCEL -->
 17. <!-- <tr>
- <td></td>

HTML & JavaScript Comments

8. Vulnerable (Low, HARM: 60) at: <http://crackme.cenzic.com/Kelev/view/updatesloanrequest.php>

- Message: No. of comments in page = 13
1. <!-- TOP HORIZONTAL BAR -->
 2. <!-- USER NAME AND LOGO -->
 3. <!-- USER NAME AND LOGO END-->
 4. <!-- FEEDBACK AND OTHER LINKS TABLE -->
 5. <!-- FEEDBACK AND OTHER LINKS TABLE END-->
 6. <!-- TOP HORIZONTAL BAR END -->
 7. <!-- LEFT SIDE MAIN MENU TABLE -->
 8. <!-- LEFT SIDE MAIN MENU TABLE END -->
 9. <!-- BILLS ONLINE TABLE -->
 10. <!-- BILLS ONLINE TABLE END -->
 11. <!-- MAIN AREA FORM TABLE -->
 12. <!-- MAIN AREA FORM TABLE END -->
 13. <!-- FOOTER ROW -->

HTML & JavaScript Comments

9. Vulnerable (Low, HARM: 60) at: <http://crackme.cenzic.com/Kelev/loans/homeloan.php>

Message: No. of comments in page = 11

1. <!-- TOP HORIZONTAL BAR -->
2. <!-- USER NAME AND LOGO -->
3. <!-- USER NAME AND LOGO END-->
4. <!-- FEEDBACK AND OTHER LINKS TABLE -->
5. <!-- FEEDBACK AND OTHER LINKS TABLE END-->
6. <!-- TOP HORIZONTAL BAR END -->
7. <!-- LEFT SIDE MAIN MENU TABLE -->
8. <!-- LEFT SIDE MAIN MENU TABLE END -->
9. <!-- BILLS ONLINE TABLE -->
10. <!-- BILLS ONLINE TABLE END -->
11. <!-- FOOTER ROW -->

HTML & JavaScript Comments

10. Vulnerable (Low, HARM: 60) at: <http://crackme.cenzic.com/Kelev/loans/studentloan.php>

Message: No. of comments in page = 11

1. <!-- TOP HORIZONTAL BAR -->
2. <!-- USER NAME AND LOGO -->
3. <!-- USER NAME AND LOGO END-->
4. <!-- FEEDBACK AND OTHER LINKS TABLE -->
5. <!-- FEEDBACK AND OTHER LINKS TABLE END-->
6. <!-- TOP HORIZONTAL BAR END -->
7. <!-- LEFT SIDE MAIN MENU TABLE -->
8. <!-- LEFT SIDE MAIN MENU TABLE END -->
9. <!-- BILLS ONLINE TABLE -->
10. <!-- BILLS ONLINE TABLE END -->
11. <!-- FOOTER ROW -->

HTML & JavaScript Comments

11. Vulnerable (Low, HARM: 60) at: <http://crackme.cenzic.com/Kelev/view/netbanking.php>

Message: No. of comments in page = 12

1. <!-- TOP HORIZONTAL BAR -->
2. <!-- USER NAME AND LOGO -->
3. <!-- USER NAME AND LOGO END-->
4. <!-- FEEDBACK AND OTHER LINKS TABLE -->
5. <!-- FEEDBACK AND OTHER LINKS TABLE END-->
6. <!-- TOP HORIZONTAL BAR END -->
7. <!-- LEFT SIDE MAIN MENU TABLE -->
8. <!-- LEFT SIDE MAIN MENU TABLE END -->
9. <!-- BILLS ONLINE TABLE -->
10. <!-- BILLS ONLINE TABLE END -->
11. <!-- Image -->
12. <!-- FOOTER ROW -->

HTML & JavaScript Comments

12. Vulnerable (Low, HARM: 1020) at: URL-not-available

Message: Limit exceeded for pages with less severe comments.
 Number of pages with less severe comments: 27
 The vulnerabilities reported for individual pages with less severe comments have been limited to 10.
 To change the number of reported items allowed for pages with less severe comment , adjust the Report Item Limit parameter.

Remediation Tips

HTML and Javascript comments should be removed from pages on production servers. Comments left by developers and designers can help an attacker make inferences about vulnerabilities in your application or site configuration

When you audit your scripts or pages, remember the following checklist:

- Remove any incidental references to server names, network topology, or trust relationships.

- Ensure comments don't reveal potential vulnerabilities, like types of input that crash the script or produce anomalous results.

- Remove all references to the script version, OS versions, or application versions. These comments help an attacker identify vulnerabilities in your platform and software.

- Ensure the comments do not reveal internal application logic, or how particular input is parsed or handled.

References

OWASP Top Ten: <http://www.owasp.org/documentation/topten.html>

OWASP: A Guide to Building Secure Web Applications: <http://www.cgisecurity.com/owasp/html/>

Attack: Directory Browsing	Ver: 1.3.5	CWE-548	Observations: 8
Descr: Directory Browsing is a vulnerability caused by unintentionally disclosing directory listings to users. The SmartAttack attempts to retrieve and identify such listings and reports them as vulnerabilities based on the assumption that the listings are unintended.			
Severity: Medium HARM: 160 Total HARM: 1280			

Impact

If a Web application is vulnerable to directory browsing, an attacker can gain information about the web application by browsing directory listings that reveal files and folder hierarchy in the application. These resources may store sensitive information about web applications and operational systems, such as source code, credentials, internal network addressing, and so on which can be used to exploit vulnerabilities in the web application.

Information leakage occurs when a web site reveals sensitive data, such as authentication information, absolute or relative paths, which may aid an attacker in exploiting the system. While leakage through such directory listings does not necessarily represent a breach in security, it does give an attacker useful guidance for future exploitation. Leakage of sensitive information may carry various levels of risk and should be limited whenever possible.

For example, any attacker can guess file and directory names not intended for public viewing. The files/paths often have common naming convention and reside in standard locations. Hence by making educated guesses an attacker can attain absolute path.

Directory Browsing Vulnerable Findings

Assessment: Silver, **Run:** 6/6/2011 8:38:25PM, **Traversal:** [Default Spider]

Directory Browsing

1. Vulnerable (Medium, HARM: 160) at: <http://crackme.cenzic.com/Kelev/scripts/>

Message: Directory listing found.

Directory Browsing

2. Vulnerable (Medium, HARM: 160) at: <http://crackme.cenzic.com/Kelev/include/>

Message: Directory listing found.

Directory Browsing

3. Vulnerable (Medium, HARM: 160) at: <http://crackme.cenzic.com/Kelev/cgi-bin/>

Message: Directory listing found.

Directory Browsing

4. Vulnerable (Medium, HARM: 160) at: <http://crackme.cenzic.com/Kelev/images/>

Message: Directory listing found.

Directory Browsing

5. Vulnerable (Medium, HARM: 160) at: <http://crackme.cenzic.com/icons/small/>

Message: Directory listing found.

Directory Browsing

6. Vulnerable (Medium, HARM: 160) at: <http://crackme.cenzic.com/icons/>

Message: Directory listing found.

Directory Browsing

7. Vulnerable (Medium, HARM: 160) at: <http://crackme.cenzic.com/Kelev/register/>

Message: Directory listing found.

Directory Browsing

8. Vulnerable (Medium, HARM: 160) at: <http://crackme.cenzic.com/Kelev/loans/>

Message: Directory listing found.

Remediation Tips

Obtaining directory lists gives an attacker useful information when planning attacks against your server or your application. Follow these guidelines to prevent unintended information disclosure:

Check the access permissions on the affected directories and configure them to prevent access.

Configure the server to disallow directory listing on the affected directories.

Examine your applications and if the directory list was obtained by exploiting a known bug or vulnerability, contact the vendor or maintainer for a patch.

If the directory listing is exploitable in a custom application then review the code and prevent malformed strings or tricked URI's from bypassing the filters or input validation you are applying to directory GET requests.

References

OWASP Top Ten

[<http://www.owasp.org/documentation/topten.html>](http://www.owasp.org/documentation/topten.html)

OWASP: A Guide to Building Secure Web Applications

[<http://www.cgisecurity.com/owasp/html/>](http://www.cgisecurity.com/owasp/html/)

Apache Directory Listing Vulnerability

[<http://cve.mitre.org/cgi-bin/cvename?name=CAN-2001-0729>](http://cve.mitre.org/cgi-bin/cvename?name=CAN-2001-0729)

Microsoft IIS 5.0 Directory Listing with WEBDAV

[<http://www.securiteam.com/windowsntfocus/6P00R0K06Q.html>](http://www.securiteam.com/windowsntfocus/6P00R0K06Q.html)

Attack: Password Autocomplete	Ver: 1.1.10	CWE-525	Observations: 1
Descr: Password Autocomplete is a vulnerability caused by allowing caching of passwords by browsers. The SmartAttack inspects responses from the application to identify if autocomplete is explicitly set to "off".			
Severity: Medium HARM: 210 Total HARM: 210			

Impact

Accessing a Web application with the Password Autocomplete vulnerability will cause users' passwords to be stored on the machine from where the browsing happens. An attacker who has access to such a machine may be able to make the browser give away those passwords by visiting the pages with the login forms, or by means of compromising the browser. He may be able to use those passwords for identity theft and/or performing malicious actions on behalf of the users.

An attacker who has access to a number of shared machines, such as at an Internet Cafe or posing as a computer maintenance professional may be able to collect passwords of users of applications with this vulnerability. It may also be possible for an attacker to install malware on victims' computers which will compromise the browser, thus making the attacker's presence at the machine unnecessary.

Password Autocomplete Vulnerable Findings

Assessment: Silver, **Run:** 6/6/2011 8:38:25PM, **Traversal:** [Default Spider]

Password Autocomplete**1. Vulnerable (Medium, HARM: 210) at: <http://crackme.cenzic.com/Kelev/register/register.php>**

Message: Page contains password input fields without autocomplete='off': 1 forms.

```
<form method=post id=frm action=register.php name=frm >
<input size=30 value= tabindex=2 id=txtFirstName class=InputBox1 name=FirstName >
<input size=30 value= tabindex=2 id=txtLastName class=InputBox1 name=LastName >
<input size=30 value= tabindex=2 id=txtUserId class=InputBox1 name=UserId >
<input type=password maxlength=20 size=20 value= tabindex=2 id=txtpass class=InputBox1
name=Password >
<input maxlength=20 size=20 value= tabindex=2 id=txtDOB class=InputBox1 name=DOB >
<input size=49 value= tabindex=2 id=txtAddress class=InputBox1 name=Address >
<input size=30 value= tabindex=2 id=txtCity class=InputBox1 name=txtCity >
<input name=drpState class=InputBox1 >
<input maxlength=20 size=15 value= tabindex=2 id=txtTelephoneNo class=InputBox1
name=TelephoneNo >
<input maxlength=40 size=25 value= tabindex=2 id=txtEmail class=InputBox1 name=Email >
</form>
```

Remediation Tips

Turn off the AUTOCOMPLETE attribute in any HTML FORM/INPUT element that is used for passwords. This can be accomplished for a single field (such as a password field) by modifying the HTML source and adding the following line:

```
<INPUT TYPE = password NAME = value AUTOCOMPLETE = "off">
```

Alternately, you can use the <FORM AUTOCOMPLETE = "off"> attribute to disable autocomplete on every form field within a page.

If the page containing the password field is served over HTTPS and it was served with headers preventing caching of the data in the form, then Autocomplete is turned off for the form in Internet Explorer.

References

Microsoft Knowledge Base article, *Using AutoComplete in HTML Forms*

[http://msdn.microsoft.com/en-us/library/ms533032\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms533032(VS.85).aspx)

MSDN Library page on the Autocomplete attribute in HTML and CSS

[http://msdn.microsoft.com/en-us/library/ms533486\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms533486(VS.85).aspx)

Attack: Check HTTP Methods	Ver: 1.1.11	CWE-650	Observations: 1
<p>Descr: A Web application supporting a few potentially dangerous HTTP methods like PUT, DELETE, COPY, MOVE, OPTIONS can aid an attacker to tamper the Web server of the application or craft further advanced attacks. Hence, this is a vulnerability we will call as Check HTTP Methods. This SmartAttack checks the HTTP Methods supported by the server and reports a failure when the web server responds to dangerous HTTP methods.</p>			
Severity: Medium HARM: 80 Total HARM: 24			

Impact

For typical Web applications, HTTP methods like GET and POST are sufficient. HTTP Methods such as PUT, DELETE, COPY, MOVE allow user to modify, delete, copy, move resources on the Web server. If the web application is supporting these methods, an attacker can leverage details disclosed by these methods for performing various illegitimate operations and attacks.

For example, HTTP TRACE method enables an attacker to leverage the resources on the web server. The TRACE method returns the contents of client HTTP requests in the entity-body of the TRACE response. Attackers could leverage this behavior to access sensitive information, such as cookies or authentication data, contained in the HTTP headers of the request.

Check HTTP Methods Warning Findings

Assessment: Silver, Run: 6/6/2011 8:38:25PM, Traversal: [Default Spider]

[Check HTTP Methods](#)

1. Warning (Medium, HARM: 24) at: <http://crackme.cenzic.com:80/>

Message: TRACE Method found enabled on crackme.cenzic.com:80

Remediation Tips

Potentially harmful HTTP methods for the Web application are PUT, DELETE, TRACE, TRACK, COPY, MOVE, LOCK, UNLOCK, PROPFIND, PROPPATCH, SEARCH, MKCOL. All these methods should be disabled if not required by the application.

Disabling HTTP Methods on IIS:

Microsoft provides the following tools to apply certain criteria and turning off unnecessary features:

- URLScan: <http://www.microsoft.com/technet/security/tools/urlscan.mspx>
- IIS Lockdown: <http://www.microsoft.com/technet/security/tools/locktool.mspx>

Example: Disable TRACE method (called a *verb* in Microsoft documentation) using URLScan

- Install URLScan filter.
- Open - %WINDOWS_ROOT%\system32\inetrv\urlscan\urlscan.ini.
- If the value of the **UseAllowVerbs** option under the **[Options]** section is:
 - 1, then make sure that the verb TRACE does *not* appear under the **[AllowVerbs]** section.
 - 0, then make sure that the verb TRACE appears under the **[DenyVerbs]** section.
- Restart the IIS Server.

Disabling HTTP Methods on Apache:

Apache httpd.conf provides the facility to turn off HTTP methods.

Example: Disable Trace method on Apache

- Open httpd.conf file.
- Add/uncomment : LoadModule rewrite_module modules/mod_rewrite.so
- Enter the following in the httpd.conf file
- RewriteEngine On
- RewriteCond %{REQUEST_METHOD} ^TRACES
- RewriteRule .* - [F]

References

<<http://www.w3.org/Protocols/rfc2616/rfc2616.html>>
<<http://asg.web.cmu.edu/rfc/rfc2518.html>>
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/e2k3/e2k3/_webdav_methods.asp>
<<http://www.microsoft.com/technet/security/tools/urlscan.mspx>>
<<http://www.microsoft.com/technet/security/tools/locktool.mspx>>
<<http://www.kb.cert.org/vuls/id/867593>>.

Attack: Web Server Vulnerabilities	Ver: 1.1.21	Observations: 37
<p>Descr: Web Server Vulnerabilities are a variety of CVE style vulnerabilities regarding known security flaws in known versions of various software infrastructures, such as Apache, PHP, Oracle, etc. This SmartAttack does hundreds of tests looking for versions and resources with evidence pointing to known CVEs.</p>		
Severity: Low HARM: 30 Total HARM: 333		

Impact

An attacker may gain administrative control of your application or launch other advanced attacks such as Command Execution, Buffer Overflows if he has an accurate knowledge of the vulnerabilities in your Web and application servers. He may also be able to influence configuration settings of your servers if certain common configuration files are kept accessible.

Web Server Vulnerabilities, if present, help an attacker plan advanced attacks based on the information disclosed through such vulnerabilities. These vulnerabilities not only disclose the Web and application servers being used, but may also disclose their versions, server-side technologies used, configuration settings and sensitive information about the application and its users.

Many Web servers advertise themselves by adding headers to each HTTP response that they send out. Many servers also come with typical configurations for ease of use. These details are publicly available most of the times, making an attacker's job easier. For example, if he comes to know the version number and configuration of a server used by a banking application, he may try to exploit any of the vulnerabilities reported in the public domain for that version of that server for manipulating accounts of the customers of the bank.

Web Server Vulnerabilities Warning Findings

Assessment: Silver, **Run:** 6/6/2011 8:38:25PM, **Traversal:** [Default Spider]

Web Server Vulnerabilities

1. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache HTTP Server Multiple Remote Denial of Service Vulnerabilities

Apache HTTP Server is prone to multiple remote denial-of-service vulnerabilities. An attacker can exploit these issues to deny service to legitimate users.

Detail information can be looked at
<http://www.securityfocus.com/bid/41963/info>

Solution:
 The vendor released updates. Please see the references for more information.
<http://www.securityfocus.com/bid/41963/references>

Web Server Vulnerabilities

2. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache 'mod_isapi' Memory Corruption Vulnerability

Apache is prone to a memory-corruption vulnerability. Attackers can leverage this vulnerability to execute arbitrary code with SYSTEM privileges; failed attacks may result in denial-of-service conditions. Apache versions prior to 2.2.15 are affected.

Detail information can be looked at
<http://www.securityfocus.com/bid/38494/info>

Solution:
 Updates are available. Please see the references for details.
<http://www.securityfocus.com/bid/38494/references>

Web Server Vulnerabilities

3. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache HTTP Server 'mod_status' Cross-Site Scripting Vulnerability

The Apache HTTP Server 'mod_status' module is prone to a cross-site scripting vulnerability because the application fails to properly sanitize user-supplied input.
 An attacker may leverage this issue to execute arbitrary script code in the browser of an unsuspecting user in the context of the affected site.
 This may allow the attacker to steal cookie-based authentication credentials and to launch other attacks. Reportedly, attackers can also use this issue to redirect users' browsers to arbitrary locations.
 Attacker can leverage this fact to perform phishing attack.

Following Apache version is affected
 Apache/1.3.20 to Apache/1.3.29
 Apache/1.3.31 to Apache/1.3.39
 Apache/2.0.40 to Apache/2.0.49
 Apache/2.0.50 to Apache/2.0.54

Detail information can be looked at
<http://www.securityfocus.com/bid/27237/>

Solution:

Web Server Vulnerabilities

4. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache and WebSphere Application Server Mod_Rewrite Off-By-One Buffer Overflow Vulnerability

Apache mod_rewrite is prone to an off-by-one buffer-overflow condition.
 The vulnerability arising in the mod_rewrite module's ldap scheme handling allows for potential memory corruption when an attacker exploits certain rewrite rules.
 An attacker may exploit this issue to trigger a denial-of-service condition.
 Reportedly, arbitrary code execution may be possible as well.

Affected versions are
 Apache/1.3.3 , Apache/1.3.4 , Apache/1.3.6, Apache/1.3.7 ,Apache/1.3.9
 Apache/1.3.31 to Apache/1.3.35
 Apache/2.0.46 to Apache/2.0.56
 WebSphere Application Server/6.0.2.1, WebSphere Application Server/6.0.2.3, WebSphere Application Server/6.0.2.5
 WebSphere Application Server/6.0.2.7, WebSphere Application Server/6.0.2.9

Detail information can be looked at
<http://www.securityfocus.com/bid/19204/>

Web Server Vulnerabilities

5. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache 'mod_proxy_ftp' Undefined Charset UTF-7 Cross-Site Scripting Vulnerability

Apache 'mod_proxy_ftp' is prone to a cross-site scripting vulnerability because the application fails to properly sanitize user-supplied input.
 An attacker may leverage this issue to execute arbitrary script code in the browser of an unsuspecting user in the context of the affected site.
 This may help the attacker steal cookie-based authentication credentials and launch other attacks.

Affected versions are
 Apache/1.3.1, Apache 1.3.3, Apache 1.3.4, Apache 1.3.6, Apache 1.3.9
 Apache/1.3.11, Apache 1.3.12, Apache 1.3.14, Apache 1.3.17, Apache 1.3.19
 Apache/1.3.22, Apache 1.3.24, Apache 1.3.26, Apache 1.3.27, Apache 1.3.28, Apache 1.3.29
 Apache/1.3.31 to Apache/1.3.39
 Apache/2.2.3 to Apache/2.2.6
 Apache/2.0.41 to Apache/2.0.49

Detail information can be looked at
<http://www.securityfocus.com/bid/27234/>

Solution:

Web Server Vulnerabilities

6. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache 'mod_proxy_ftp' Undefined Charset UTF-7 Cross-Site Scripting Vulnerability

Apache 'mod_proxy_ftp' is prone to a cross-site scripting vulnerability because the application fails to properly sanitize user-supplied input.
An attacker may leverage this issue to execute arbitrary script code in the browser of an unsuspecting user in the context of the affected site.
This may help the attacker steal cookie-based authentication credentials and launch other attacks.

Affected versions are

Apache/1.3.1, Apache 1.3.3, Apache 1.3.4, Apache 1.3.6, Apache 1.3.9
Apache/1.3.11, Apache 1.3.12, Apache 1.3.14, Apache 1.3.17, Apache 1.3.19
Apache/1.3.22, Apache 1.3.24, Apache 1.3.26, Apache 1.3.27, Apache 1.3.28, Apache 1.3.29
Apache/1.3.31 to Apache/1.3.39
Apache/2.2.3 to Apache/2.2.6
Apache/2.0.41 to Apache/2.0.49

Detail information can be looked at
<http://www.securityfocus.com/bid/27234/>

Solution:

[Web Server Vulnerabilities](#)

7. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache and IBM HTTP Server(Powered by Apache) 413 Error HTTP Request Method Cross-Site Scripting Weakness

Apache and IBM HTTP Server(powered by Apache) is prone to a cross-site scripting weakness when handling HTTP request methods that result in 413 HTTP errors.
An attacker may exploit this issue to steal cookie-based authentication credentials and launch other attacks.

Detail information can be looked at
<http://www.securityfocus.com/bid/26663>

Solution:

This can be looked at
<http://www.securityfocus.com/bid/26663/solution>

[Web Server Vulnerabilities](#)

8. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache Mod_AutoIndex.C Undefined Charset Cross-Site Scripting Vulnerability

Apache is affected by a vulnerability that may cause certain web pages to be prone to a cross-site scripting attack.
This issue stems from a lack of a defined charset on certain generated pages.
Web pages generated by the affected source code may be prone to a cross-site scripting issue.
This vulnerability is hard to exploit when application is running on Windows operating system.

Affected versions are

Apache/2.2.0 to Apache/2.2.4
Apache/2.1.1 to Apache/2.1.8
Apache/2.0.50 to Apache/2.0.59
Apache/2.0.40 to Apache/2.0.49

Detail information can be looked at
<http://www.securityfocus.com/bid/25653/exploit>

Solution:

[Web Server Vulnerabilities](#)

9. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache

Versions of Apache prior to 2.0.55 are affected by the following vulnerability:

"A flaw occurred when using the Apache server as a HTTP proxy . A remote attacker could send a HTTP request with both a "Transfer-Encoding: chunked" header and a Content-Length header, causing Apache to incorrectly handle and forward the body of the request in a way that causes the receiving server to process it as a separate HTTP request . This could allow the bypass of web application firewall protection or lead to cross-site scripting (XSS) attacks."

Quoted from Apache.org (see below).

The following CVE Names are associated with these security issues:

CVE-2005-2088

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-2088>

Web Server Vulnerabilities

10. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache Byterange filter DoS

Versions of Apache prior to 2.0.55 are affected by the following vulnerability:

"A flaw in the byterange filter would cause some responses to be buffered into memory . If a server has a dynamic resource such as a CGI script or PHP script which generates a large amount of data, an attacker could send carefully crafted requests in order to consume resources, potentially leading to a Denial of Service."

Quoted from Apache.org (see below).

The following CVE Names are associated with these security issues:

CVE-2005-2728

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-2728>

Vulnerability Information at Apache.org

http://httpd.apache.org/security/vulnerabilities_20.html

Web Server Vulnerabilities

11. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache CRL off-by-one Overflow

Versions of Apache prior to 2.0.55 are affected by the following vulnerability:

"An off-by-one stack overflow was discovered in the mod_ssl CRL verification callback . In order to exploit this issue the Apache server would need to be configured to use a malicious certificate revocation list (CRL)"

Quoted from Apache.org (see below).

The following CVE Names are associated with these security issues:

CVE-2005-1268

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-1268>

Vulnerability Information at Apache.org

http://httpd.apache.org/security/vulnerabilities_20.html

Web Server Vulnerabilities

12. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache PCRE Integer Overflow

Versions of Apache prior to 2.0.55 are affected by the following vulnerability:

The following CVE Names are associated with these security issues:

"An integer overflow flaw was found in PCRE, a Perl-compatible regular expression library included within httpd. A local user who has the ability to create .htaccess files could create a maliciously crafted regular expression in such a way that they could gain the privileges of a httpd child."

Quoted from Apache.org (see below).

CVE-2005-2491

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-2491>

Web Server Vulnerabilities

13. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache SSL VerifyClient bypass

Versions of Apache prior to 2.0.55 are affected by the following vulnerability:

"A flaw in the mod_ssl handling of the "SSLVerifyClient" directive. This flaw would occur if a virtual host has been configured using "SSLVerifyClient optional" and further a directive "SSLVerifyClient required" is set for a specific location. For servers configured in this fashion, an attacker may be able to access resources that should otherwise be protected, by not supplying a client certificate when connecting."

Quoted from Apache.org (see below).

The following CVE Names are associated with these security issues:

CVE-2005-2700

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-2700>

Vulnerability Information at Apache.org

http://httpd.apache.org/security/vulnerabilities_20.html

Web Server Vulnerabilities

14. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache mod_imap Referer Cross-Site Scripting

Versions of Apache prior to 2.0.58 are affected by the following vulnerability:

"A flaw in mod_imap when using the Referer directive with image maps. In certain site configurations a remote attacker could perform a cross-site scripting attack if a victim can be forced to visit a malicious URL using certain web browsers"

Quoted from Apache.org (see below).

The following CVE Names are associated with these security issues:

CVE-2005-3352

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-3352>

Vulnerability Information at Apache.org

http://httpd.apache.org/security/vulnerabilities_20.html

Web Server Vulnerabilities

15. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache mod_ssl access control DoS

Versions of Apache prior to 2.0.58 are affected by the following vulnerability:

"A NULL pointer dereference flaw in mod_ssl was discovered affecting server configurations where an SSL virtual host is configured with access control and a custom 400 error document. A remote attacker could send a carefully crafted request to trigger this issue which would lead to a crash. This crash would only be a denial of service if using the worker MPM." Quoted from Apache.org (see below).

The following CVE Names are associated with these security issues:

CVE-2005-3357

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-3357>

Vulnerability Information at Apache.org

http://httpd.apache.org/security/vulnerabilities_20.html

Web Server Vulnerabilities

16. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/>

Message: Apache MPM memory leak

Versions of Apache prior to 2.0.55 are affected by the following vulnerability:

"A memory leak in the worker MPM would allow remote attackers to cause a denial of service (memory consumption) via aborted connections, which prevents the memory for the transaction pool from being reused for other connections."

Quoted from Apache.org (see below).

The following CVE Names are associated with these security issues:

CVE-2005-2970

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-2970>

Vulnerability Information at Apache.org

http://httpd.apache.org/security/vulnerabilities_20.html

Web Server Vulnerabilities

17. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP GD Graphics Library '_gdGetColors' Remote Buffer Overflow Vulnerability

GD Graphics is prone to a remote buffer-overflow vulnerability because the software fails to perform adequate boundary checks on user-supplied data.

An attacker can exploit this issue to execute arbitrary code with the privileges of the user running an application that relies on the affected library.

Failed exploit attempts will result in a denial-of-service condition.

Following Apache versions are affected

PHP/3.0.0 to PHP/3.0.18

PHP/4.0.0 to PHP/4.0.7

PHP/4.3.1 to PHP/4.3.9

PHP/5.0.0 to PHP/5.0.5

PHP/5.1.1 to PHP/5.1.5

PHP/5.2.1 to PHP/5.2.9

Detail information can be looked at

<http://www.securityfocus.com/bid/36712/info>

Solution:

Web Server Vulnerabilities

18. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP Interruptions and Calltime Arbitrary Code Execution Vulnerability

PHP is prone to an arbitrary-code-execution vulnerability.

An attacker can exploit this issue to execute arbitrary with the privileges of the user running the affected application.

Successfully exploiting this issue will compromise the affected application and possibly underlying computers.

Detail information can be looked at

<http://www.securityfocus.com/bid/35867/>

Solution:
 More information can be looked at
<http://www.securityfocus.com/bid/35867/references>

Web Server Vulnerabilities

19. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP 'mb_ereg_replace()' String Evaluation Vulnerability

The 'mb_ereg_replace()' function of PHP is prone to a vulnerability that can result in the improper evaluation of user-supplied input.

Exploiting this issue may allow attackers to execute arbitrary PHP commands in the context of the affected application.

Detail information can be looked at
<http://www.securityfocus.com/bid/34873/>

Solution:
 More information can be looked at
<http://www.securityfocus.com/bid/34873/references>

Web Server Vulnerabilities

20. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP .htaccess Safe_Mode and Open_Basedir Restriction-Bypass Vulnerability

PHP is prone to a 'safe_mode' and 'open_basedir' restriction-bypass vulnerability. Successful exploits could allow an attacker to write files in unauthorized locations. These vulnerabilities would be an issue in shared-hosting configurations where multiple users can create and execute arbitrary PHP script code, with the 'safe_mode' and 'open_basedir' restrictions assumed to isolate the users from each other.

Detail information can be looked at
<http://www.securityfocus.com/bid/24661/>

Solution:
 Vendor has released the update. More information can be looked at
<http://www.securityfocus.com/bid/24661/solution>

Web Server Vulnerabilities

21. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP 'popen()' Function Buffer Overflow Vulnerability

PHP is prone to a buffer-overflow vulnerability because it fails to perform boundary checks before copying user-supplied data to insufficiently sized memory buffers.

An attacker can exploit this issue to execute arbitrary machine code in the context of the affected webserver.

Failed exploit attempts will likely crash the webserver, denying service to legitimate users.

Following PHP version is affected
 PHP/4.2.0 to PHP/4.2.3
 PHP/4.3.1 to PHP/4.3.9
 PHP/4.4.1 to PHP/4.4.9
 PHP/5.0.1 to PHP/5.0.5
 PHP/5.1.1 to PHP/5.1.6
 PHP/5.2.1 to PHP/5.2.8

Detail information can be looked at
<http://www.securityfocus.com/bid/33216/>

Solution:

Web Server Vulnerabilities

22. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP 'mbstring' Extension Buffer Overflow Vulnerability

PHP is prone to a buffer-overflow vulnerability because it fails to perform boundary checks before copying user-supplied data to insufficiently sized memory buffers.

The issue affects the 'mbstring' extension included in the standard distribution.
 An attacker can exploit this issue to execute arbitrary machine code in the context of the affected webserver.
 Failed exploit attempts will likely crash the webserver, denying service to legitimate users.

Following PHP version is affected
 PHP/5.1.1 to PHP/5.1.6
 PHP/5.0.0 to PHP/5.0.5
 PHP/4.4.1 to PHP/4.4.9
 PHP/4.3.1 to PHP/4.3.9

Detail information can be looked at
<http://www.securityfocus.com/bid/32948/>

Solution:
 More information can be looked at

Web Server Vulnerabilities

23. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP Zip_Entry_Read() Integer Overflow Vulnerability

PHP is prone to an integer-overflow vulnerability because it fails to ensure that integer values aren't overrun.
 Attackers may exploit this issue to cause a heap-based buffer overflow.
 Exploiting this issue may allow attackers to execute arbitrary machine code in the context of the affected application.
 Failed exploit attempts will likely result in a denial-of-service condition.

Detail information can be looked at
<http://www.securityfocus.com/bid/23169/>

Solution:
 More information can be looked at
<http://www.securityfocus.com/bid/23169/solution>

Web Server Vulnerabilities

24. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP Perl Compatible Regular Expression(PCRE) Heap Overflow Vulnerability

PHP PCRE is prone to a heap-overflow vulnerability.
 This issue is due to the library's failure to properly perform boundary checks on user-supplied input before copying data to an internal memory buffer.
 The impact of successful exploitation of this vulnerability depends on the application and the user credentials using the vulnerable library.
 A successful attack may ultimately permit an attacker to control the contents of critical memory control structures and write arbitrary data to arbitrary memory locations.

Affected version is
 PHP/4.0 to PHP/4.7
 PHP/4.2.1 to PHP/4.2.3
 PHP/4.3.1 to PHP/4.3.9
 PHP/5.0.1 to PHP/5.0.5

Detail information can be looked at
<http://www.securityfocus.com/bid/14620/>

Solution:

Web Server Vulnerabilities

25. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP GD Extension WBMP File Integer Overflow Vulnerabilities

PHP's GD extension is prone to two integer-overflow vulnerabilities because it fails to ensure that integer values aren't overrun.
 Successfully exploiting these issues allows attackers to crash the affected application , potentially denying service to legitimate users.
 Due to the nature of the issues, code execution may also be possible, but this has not been confirmed.

Affected versions are
PHP/4.0.0 to PHP/4.0.7
PHP/4.1.0 to PHP/4.1.2
PHP/4.2.0 to PHP/4.2.3
PHP/4.3.1 to PHP/4.3.11
PHP/4.4.0 to PHP/4.4.6
PHP/5.0.0 to PHP/5.0.5
PHP/5.1.1 to PHP/5.1.6

Detail information can be looked at
<http://www.securityfocus.com/bid/23357/>

[Web Server Vulnerabilities](#)

26. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP Folded Mail Headers Email Header Injection Vulnerability

PHP is prone to an email-header-injection vulnerability because it fails to properly sanitize user-supplied input when constructing email messages.
Exploiting this issue allows a malicious user to create arbitrary email headers.
It results into creation and transmission of spam messages from the affected computer .

Affected versions are
PHP/4.0.1 to PHP/4.0.7
PHP/4.2.1 to PHP/4.2.3
PHP/4.3.1 to PHP/4.3.11
PHP/5.0.1 to PHP/5.0.5
PHP/5.1.1 to PHP/5.1.6

Detail information can be looked at
<http://www.securityfocus.com/bid/23145/>

Solution:
Vendor has released the patch to address this issue .
More information can be looked at

[Web Server Vulnerabilities](#)

27. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP memory_limit Remote Code Execution Vulnerability

PHP modules compiled with memory_limit support are affected by a remote code-execution vulnerability.
This issue occurs because the PHP module fails to properly handle memory_limit request termination.
An attacker can leverage this issue by exploiting the Apache ap_escape_html Memory Allocation Denial Of Service Vulnerability .
The attacker can cause premature termination during critical code execution.
Attackers can exploit this issue to execute arbitrary code on an affected computer within the context of the vulnerable application, facilitating unauthorized access.

Affected version is
PHP/3.0.0 to PHP/3.0.16
PHP/4.0.0 to PHP/4.0.7
PHP/4.3.1 to PHP/4.3.7

Detail information can be looked at
<http://www.securityfocus.com/bid/10725/>

Solution:

[Web Server Vulnerabilities](#)

28. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP Mail Function ASCIIZ Message Truncation Weakness

PHP is prone to a weakness that allows attackers to truncate email text .
Successful exploits may allow attackers to truncate email text to manipulate message content.

This may potentially assist in phishing or other attacks.

Affected versions are
PHP 4.0.0 to PHP 4.0.7
PHP 4.1.0 to PHP 4.1.2
PHP 4.2.0 to PHP 4.2.3
PHP 4.3.1 to PHP 4.3.11
PHP 4.4.0 to PHP 4.4.6
PHP 5.0.0 to PHP 5.0.5
PHP 5.1.1 to PHP 5.1.6
PHP 5.2.1

Detail information can be looked at
<http://www.securityfocus.com/bid/23146/>

Web Server Vulnerabilities

29. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP EXT/Session HTTP Response Header Injection Vulnerability

PHP is prone to an HTTP-response-header-injection vulnerability because it fails to sanitize user-supplied input.

An attacker can exploit this issue to inject additional cookie attributes into session cookies.

This may lead to other attacks.

Vulnerable PHP versions are

1. PHP 5.2.1 to PHP 5.2.3
2. PHP 5.0.0 to PHP 5.0.5
3. PHP 4.4.0 to PHP 4.4.7
4. PHP 4.3.1 to PHP 4.3.11
5. PHP 4.0.0 to PHP 4.0.3

Detail Information can be looked at
<http://www.securityfocus.com/bid/24268/>
<http://www.php-security.org/MOPB/PMOPB-46-2007.html>

Web Server Vulnerabilities

30. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP Soap Engine Make_HTTP_Soap_Request Weak Nonce HTTP Authentication Weakness

PHP Soap Engine is prone to an authentication weakness.

Successfully exploiting this issue would allow an attacker to obtain information about the nonce used for the digest authentication.

Information obtained may allow the attacker to bypass certain security restrictions and potentially gain unauthorized access to the affected application.

Detail Information Is Available at

<http://www.securityfocus.com/bid/24034/>
<http://blog.php-security.org/archives/80-Watching-the-PHP-CVS.html>

Web Server Vulnerabilities

31. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP FTP_Putcmd Function HTTP Response Splitting Vulnerability

PHP is prone to an HTTP-response-splitting vulnerability because it fails to sanitize user-supplied input.

A remote attacker may exploit this vulnerability to influence or misrepresent how web content is served, cached, or interpreted. This could aid in various attacks that attempt to entice client users into a false sense of trust.

Detail information is available at
<http://www.securityfocus.com/bid/23818>
<http://nvd.nist.gov/nvd.cfm?cvename=CVE-2007-2509>

Web Server Vulnerabilities

32. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: Number of vulnerabilities were reported in PHP. A remote or local user may be able to execute arbitrary code on the target system.

1. It is possible to exploit the 16bit reference counter of PHP 4. It was only exploitable with local access. However because PHP does not protect against these overflows anywhere there are other exploit vectors. With unserialize() it is triggerable remotely because many popular PHP applications still use unserialize() on user supplied data [CVE-2007-1286].

For example phpBB2.

Detail information is available at

<http://php-security.org/MOPB/MOPB-04-2007.html>

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-1287>

2. The zip_read_entry() function that is used to read the content of a file stored inside a .ZIP archive is vulnerable to an integer overflow in memory allocation that leads to an exploitable bufferoverflow [CVE-2007-1777].

Detail information is available at

<http://php-security.org/MOPB/MOPB-35-2007.html>

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-1777>

Web Server Vulnerabilities

33. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: Several vulnerabilities were reported in PHP. A remote or local user may be able to execute arbitrary code on the target system or cause denial of service conditions.

Several buffer overflows exist in some PHP functions [CVE-2007-0906].

The PHP session extension, the str_replace() function, the imap_mail_compose() function, the sqlite_udf_decode_binary() function and the maxsize parameter of the msg_receive() functions are affected.

A remote user may be able to submit specially crafted values to a program that uses the affected function to trigger an overflow and execute arbitrary code .

Detail information is available at

<http://www.php-security.org/MOPB/MOPB-39-2007.html>

<http://www.php-security.org/MOPB/MOPB-40-2007.html>

<http://www.php-security.org/MOPB/MOPB-41-2007.html>

<http://www.php-security.org/MOPB/MOPB-43-2007.html>

A buffer underflow in the sapi_header_op() function may let users cause denial of service conditions [CVE-2007-0907].

The wddx extension can be exploited using certain WDDX input packets to view random

Web Server Vulnerabilities

34. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: Number of vulnerabilities were reported in PHP. A remote user can cause denial of service (DoS) condition.

1. A remote user can submit a specially crafted variable with a deeply nested array to cause deep recursion in the variable destruction routines in the Zend Engine.

The PHP application may crash.

The original advisory is available at:

<http://www.php-security.org/MOPB/MOPB-03-2007.html>

Solution: No solution was available at the time of this entry.

2. Whenever a PHP application goes into a very deep recursion it will crash when it runs out of stack. The original advisory is available at:

<http://www.php-security.org/MOPB/MOPB-02-2007.html>

Solution: No solution was available at the time of this entry.

Web Server Vulnerabilities

35. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP Session Data Deserialization Arbitrary Code Execution Vulnerability

When register_globals is activated the deserialization of the session data can overwrite any global variable, including the _SESSION array. Because of its special implementation this can result in arbitrary code execution.

Under normal situations this vulnerability can only be exploited locally. However it might be possible for a remote attacker to use an application vulnerability to inject a session data file onto the server.

Through this vulnerability it is possible to execute arbitrary code on servers running such applications. The Suhosin Extension will protect you from this kind of attack in the default config, because session data is encrypted on the server and cannot be easily modified.

Detail information is available at
<http://www.php-security.org/MOPB/MOPB-31-2007.html>

Web Server Vulnerabilities

36. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP scanf.c Buffer Overflow
 Date Disclosed: Oct 5 2006

Versions of PHP prior to 4.4.3 allow an attacker to execute arbitrary code via the sscanf function, that may increment an index past the end of an array, triggering an over-read. An attacker can exploit this flaw to execute arbitrary commands on the host.

Sites running affected versions of PHP are encouraged to upgrade to a fixed version as soon as it becomes available.

Additional information is available at the link below:

<http://cve.mitre.org/cgi-bin/cvename?name=CVE-2006-4020>

Web Server Vulnerabilities

37. Warning (Low, HARM: 9) at: <http://crackme.cenzic.com/Kelev/view/home.php>

Message: PHP Race Condition via open_basedir
 Date Disclosed: Oct 4 2006

A design flaw in PHP versions 4.X and 5.X was discovered by the Hardened PHP Project that allows an attacker to bypass access control restrictions configured with openbasedir. Usage of PHP's symlink function can be employed to exploit a race condition that allows the attacker to access any directory protected by open_basedir. There is no known solution to this vulnerability, although workarounds are available.

Consult the original advisory to determine the suitability of workarounds for your environment. Additionally, it is recommended that sites using PHP rely on operating system access controls, such as chroots, as PHP's openbasedir is insecure by design.

Advisory: PHP open_basedir Race Condition Vulnerability
<http://securitytracker.com/alerts/2006/Oct/1016977.html>

Remediation Tips

A solution is described in the vulnerability report made by the SmartAttack. The solution might involve configuring the web server appropriately to eliminate the vulnerability. Other solutions are to upgrade the web application component to a new version in which the vulnerability has been removed, or to apply a patch released by the vendor.

References

The SecurityTracker archives: <http://www.securitytracker.com>

The BugTraq vulnerability database at SecurityFocus: <http://www.securityfocus.com>

The Open Source Vulnerability Database: <http://www.osvdb.com>