

## 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`

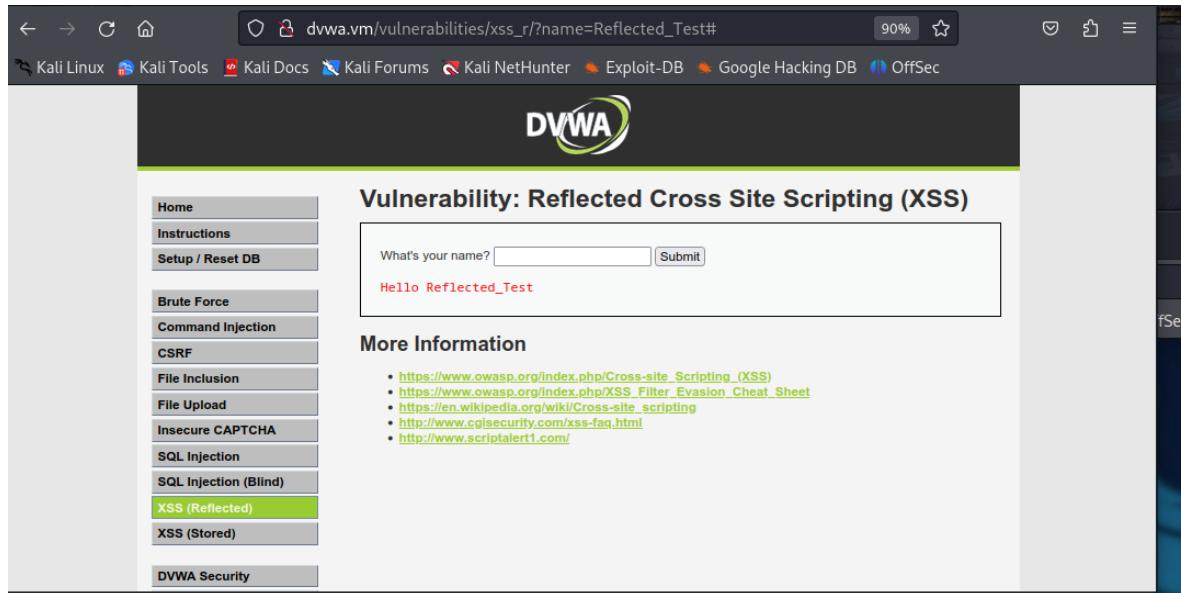
You will see the message `Hello Reflected_Test` appear. In the second screenshot

The screenshot shows a web browser window for the DVWA application. The URL is `dvwa.vm/vulnerabilities/xss_r/`. The main content area displays the title "Vulnerability: Reflected Cross Site Scripting (XSS)". Below the title is a form with a text input field containing the value "Reflected\_Test". To the right of the input field is a "Submit" button. Further down, there is a "More Information" section with a list of links related to XSS:

- [https://www.owasp.org/index.php/Cross-site\\_Scripting\\_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- [https://www.owasp.org/index.php/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)
- [https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

The left sidebar contains a navigation menu with the following items:  
Home  
Instructions  
Setup / Reset DB  
Brute Force  
Command Injection  
CSRF  
File Inclusion  
File Upload  
Insecure CAPTCHA  
SQL Injection  
SQL Injection (Blind)  
**XSS (Reflected)**  
XSS (Stored)  
DVWA Security

Screenshot 1



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.

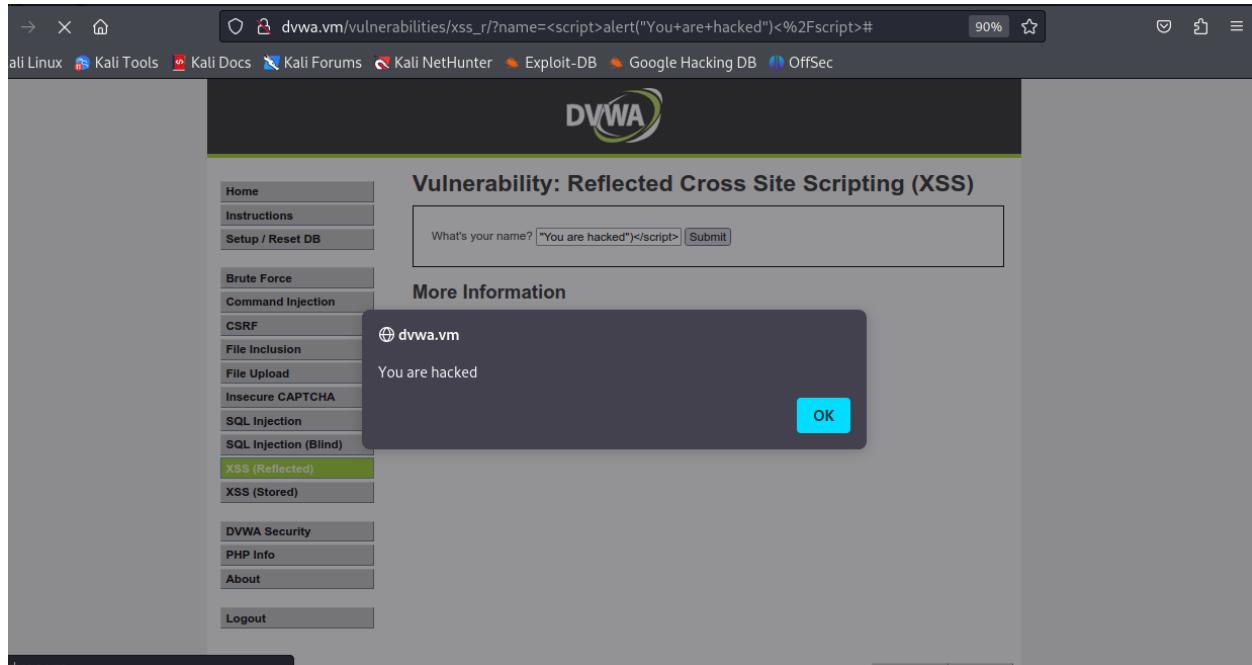
A screenshot of a browser's developer tools showing the page source code. The URL in the address bar is 'view-source:http://dvwa.vm/vulnerabilities/xss\_r/?name=Reflected\_Test#'. The source code highlights the reflected XSS payload 'Hello Reflected\_Test' in red. The code includes HTML structure like 

, 

# , , , and ``` , along with a list of external links under a 'More Information' section. ```

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.



## 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.

The screenshot shows a browser window for the DVWA application at the URL `dvwa.vm/vulnerabilities/xss_r/?name=<script>alert("You+are+hacked")<%2Fscripts>#`. The page title is "Vulnerability: Reflected Cross Site Scripting (XSS)". On the left, there's a sidebar menu with various security test categories. The "XSS (Reflected)" option is highlighted in green. The main content area contains a form with a "Submit" button and a red alert message "Hello alert('You are hacked')". Below the form, under "More Information", is a list of links related to XSS. At the bottom, it shows the user is logged in as "admin".

What's your name?  Submit

Hello alert("You are hacked")

More Information

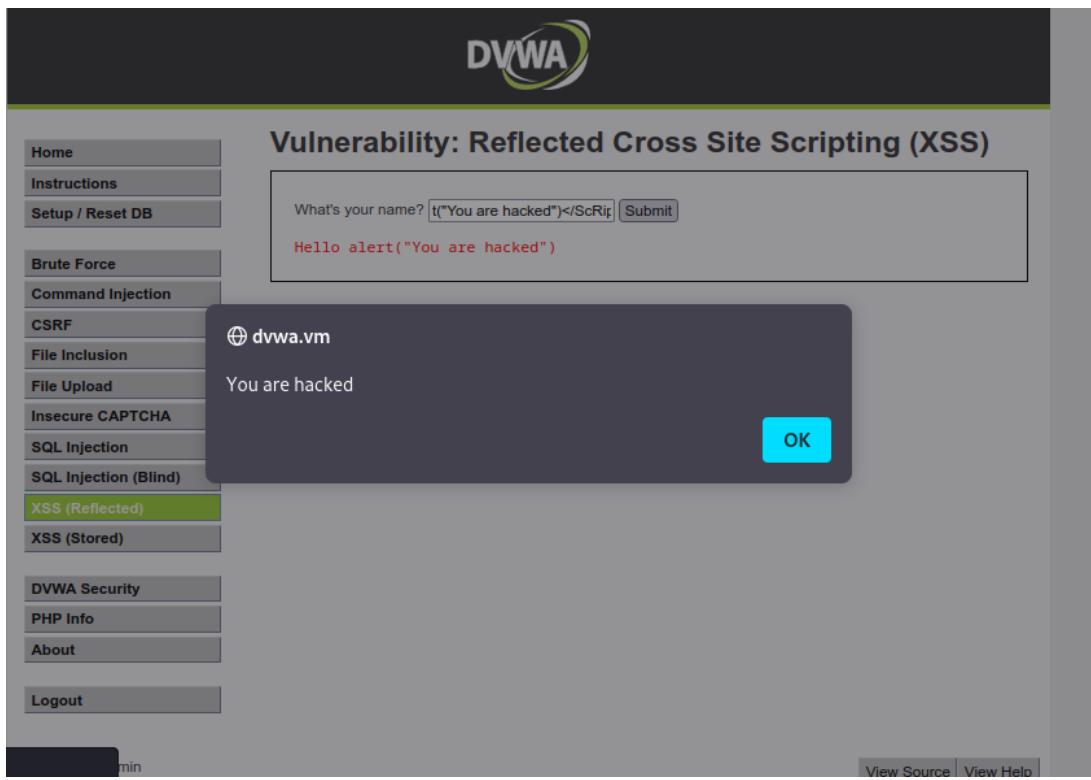
- [https://www.owasp.org/index.php/Cross-site\\_Scripting\\_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- [https://www.owasp.org/index.php/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)
- [https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

Username: admin

[View Source](#) | [View Help](#)

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.



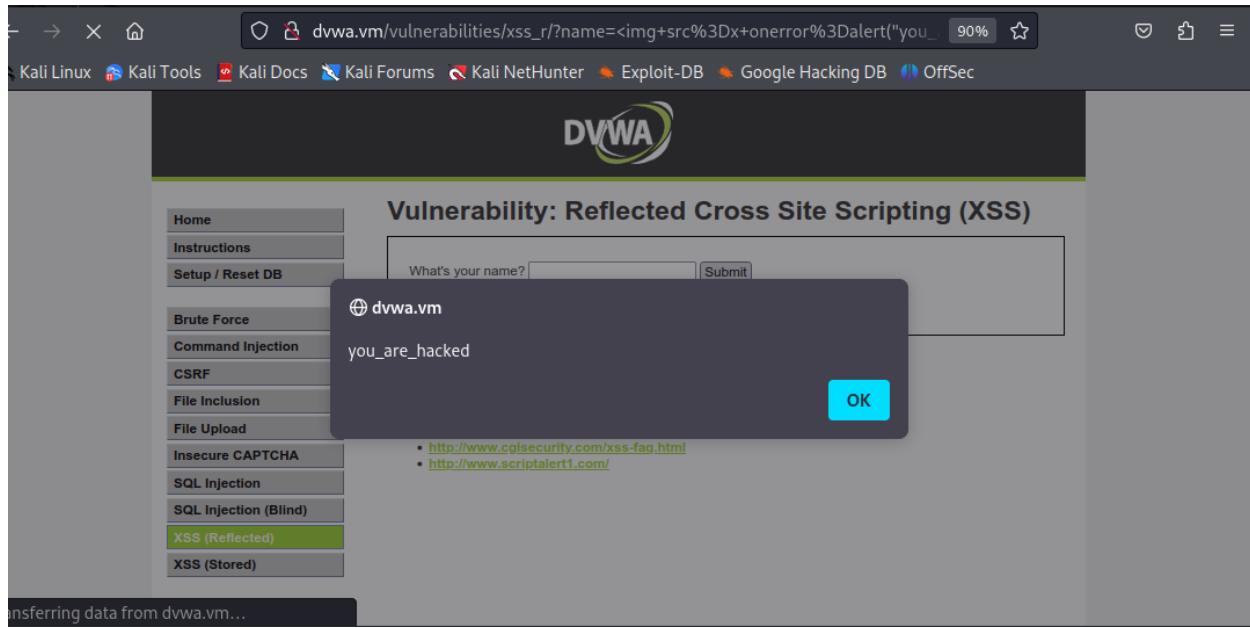
## Perform a Reflective XSS attack at High security level.

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.



## 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

The screenshot shows the DVWA Stored XSS page. On the left, a sidebar lists various vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), XSS (Reflected), and XSS (Stored). The XSS (Stored) option is selected and highlighted in green. The main content area displays a form for signing a guestbook. The 'Name' field contains 'test' and the 'Message' field contains 'Message: This is a test comment.'. Below the form, five entries from the database are listed:

- 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

At the bottom, there's a 'More Information' section with links to various XSS resources.

Screenshot 1

The screenshot shows the browser's developer tools with the 'view-source' tab selected. The page content is displayed as HTML source code. A red oval highlights the injected XSS payloads in the database entries. The payloads are:  
1. Name: test<br />Message: This is a test comment.<br /></div>  
2. Name: Omar<br />Message: &lt;script&gt;alert('omar')&lt;/script&gt;<br /></div>  
3. Name: Omar<br />Message: &lt;script&gt;alert("omar")&lt;/script&gt;<br /></div>  
4. Name: omae<br />Message: alert("omar")<br /></div>  
5. Name: XSS Test#1<br />Message: Stored XSS Test<br /></div>  
6. Name: XSS Test#1<br />Message: Stored XSS Test<br /></div>

Screenshot 2

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

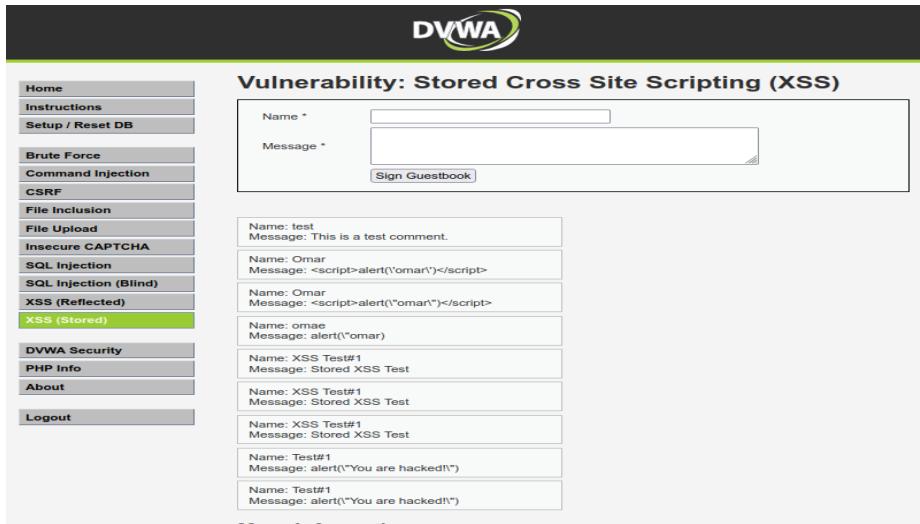
Enter **Test#1** in the **Name \*** box and enter the following payload in the **Message \*** box and click **Sign Guestbook**.

```
<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.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The URL in the browser is `dvwa.vm/vulnerabilities/xss_s/`. The main content area displays a form titled "Vulnerability: Stored Cross Site Scripting (XSS)". The "Name \*" field contains "Test#1". The "Message \*" field contains the payload "`<script>alert("You are hacked!")</script>`". Below the form is a "Sign Guestbook" button. To the right of the form, a list of messages is shown, each consisting of a "Name:" and "Message:" pair. The last message in the list is identical to the ones above it, indicating that the input was sanitized. The sidebar on the left lists various vulnerability types, with "XSS (Stored)" currently selected. The DVWA logo is at the top right.

Screenshot 1



The screenshot shows the DVWA application interface. On the left is a sidebar menu with various security test categories. The 'XSS (Stored)' category is highlighted with a green background. The main content area is titled 'Vulnerability: Stored Cross Site Scripting (XSS)'. It features a form with two input fields: 'Name \*' and 'Message \*'. Below the form, a list of guestbook entries is displayed, showing Name and Message pairs. The first entry is 'Name: test, Message: This is a test comment.'. Subsequent entries show various user names and messages, including one where the message contains a script tag.

Name	Message
test	<script>alert('This is a test comment.')</script>
Omar	<script>alert('Omar')</script>
Omar	<script>alert("omar")</script>
omar	alert("omar")
XSS Test#1	Stored XSS Test
XSS Test#1	Stored XSS Test
XSS Test#1	Stored XSS Test
Test#1	Message: alert("You are hacked!")
Test#1	Message: alert("You are hacked!")

Screenshot 2

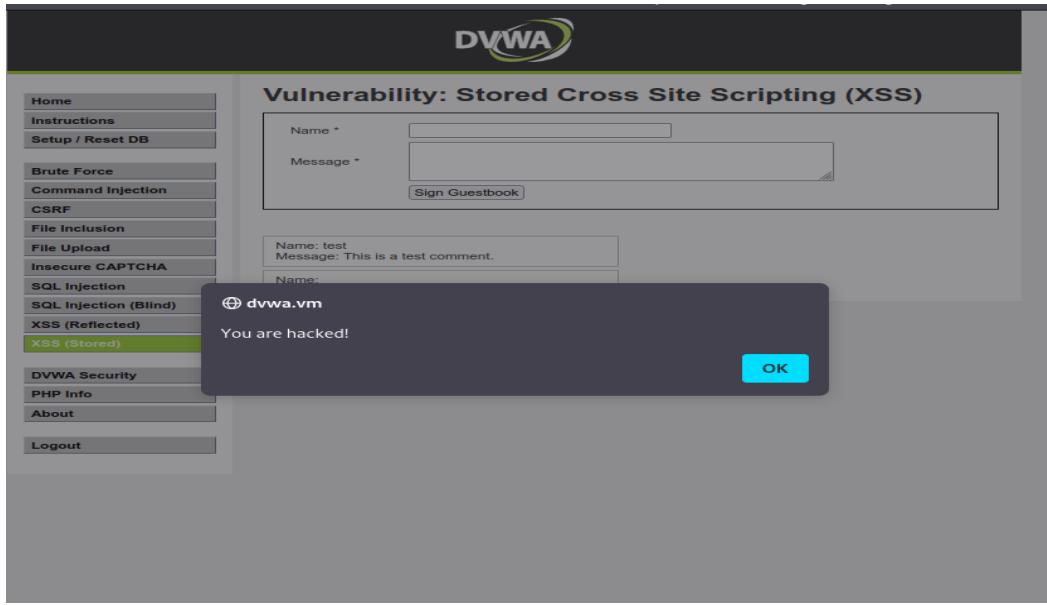
in the **Name \*** field enter the payload `<ScRipt>alert("You are hacked!")</ScRipt>`

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



## 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

Screenshot 1 shows the DVWA Stored XSS page. The left sidebar lists various vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), XSS (Reflected), XSS (Stored) (which is selected), DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: Stored Cross Site Scripting (XSS)". It contains a form with fields for "Name" and "Message", and a "Sign Guestbook" button. Below the form, a message box displays "Name: test" and "Message: This is a test comment.". Another message box below it shows "Name: Test#1" and "Message: Stored XSS Test". A red arrow points from the text "Message: Stored XSS Test" to the "More Information" section. The "More Information" section lists several links related to XSS, including OWASP and Wikipedia entries. At the bottom, there are "View Source" and "View Help" buttons.

Screenshot 1

Screenshot 2 shows the browser developer tools' "View Source" tab for the DVWA XSS page. The page's HTML source code is displayed, highlighting the user input "Name: Test#1" and "Message: Stored XSS Test" which were submitted via the "Sign Guestbook" button. These inputs appear in the "Message" field of the database table row. A red arrow points from the highlighted text in the source code to the same text in the DVWA screenshot above.

```
53     <div id="main_body">
54
55
56 <div class="body_padded">
57   <h1>Vulnerability: Stored Cross Site Scripting (XSS)</h1>
58
59   <div class="vulnerable_code_area">
60     <form method="post" name="guestform" onsubmit="return validate_form(this)">
61       <table width="550" border="0" cellpadding="2" cellspacing="1">
62         <tr>
63           <td width="100">Name </td>
64           <td><input name="txtName" type="text" size="30" maxlength="10"></td>
65         </tr>
66         <tr>
67           <td width="100">Message </td>
68           <td><textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea></td>
69         </tr>
70         <tr>
71           <td width="100">&ampnbsp</td>
72           <td><input name="btnSign" type="submit" value="Sign Guestbook" onClick="return checkForm();"></td>
73         </tr>
74       </table>
75
76     </form>
77
78   </div>
79   <br />
80
81   <div id="guestbook_comments">Name: test<br />Message: This is a test comment.<br /></div>
82   <div id="guestbook_comments">Name: Test#1<br />Message: Stored XSS Test<br /></div>
83   <div id="guestbook_comments">Name: Test#1<br />Message: Stored XSS Test<br /></div>
84
85   <br />
86
87   <h2>More Information</h2>
88   <ul>
89     <li><a href="http://hiderefer.com/?https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)" target="_blank">https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)</a></li>
90     <li><a href="http://hiderefer.com/?https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet" target="_blank">https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet</a></li>
91     <li><a href="http://hiderefer.com/?https://en.wikipedia.org/wiki/Cross-site_scripting" target="_blank">https://en.wikipedia.org/wiki/Cross-site_scripting</a></li>
92     <li><a href="http://hiderefer.com/?http://www.cgisecurity.com/xss-faq.html" target="_blank">http://www.cgisecurity.com/xss-faq.html</a></li>
93     <li><a href="http://hiderefer.com/?http://www.scriptalert1.com/" target="_blank">http://www.scriptalert1.com/</a></li>
94   </ul>
```

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.

The screenshot shows the DVWA application's interface. On the left is a sidebar with various security test categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), XSS (Reflected), and XSS (Stored). The XSS (Stored) option is highlighted. The main content area has a title "Vulnerability: Stored Cross Site Scripting (XSS)". It contains two input fields: "Name \*" and "Message \*". Below these is a "Sign Guestbook" button. To the right of the buttons is a list of guestbook entries. The last entry, which corresponds to the "Test#1" input, includes the message "Message: alert('You are hacked!')". At the bottom of the main content area is a "More Information" section with a bulleted list of links related to XSS.

- [https://www.owasp.org/index.php/Cross-site\\_Scripting\\_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- [https://www.owasp.org/index.php/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)
- [https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://www.colosecurity.com/xss-faq.html>
- <http://www.scripttalent1.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.

Screenshot 1 shows the DVWA XSS (Stored) page. The left sidebar menu is visible, with 'XSS (Stored)' highlighted. The main content area displays a guestbook entry where the 'Message' field contains the value: <iframe src="http://h4cker.org"></iframe>. Below this, four other entries are shown, each with a different name and message. A 'More Information' section at the bottom provides links to various XSS resources.

Screenshot 2 shows the same DVWA XSS (Stored) page. The 'Message' field now contains the value: Name: iframe Message:. The page displays an error message: "Hmm. We're having trouble finding that site".

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.

The screenshot shows the DVWA application interface. On the left is a sidebar with various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), XSS (Reflected), and XSS (Stored). The XSS (Stored) item is highlighted. The main content area has a title 'Vulnerability: Stored Cross Site Scripting (XSS)'. It contains two input fields: 'Name \*' with 'Cookie' typed in, and 'Message \*' with the XSS payload '<script>alert(document.cookie)</script>'. Below these is a 'Sign Guestbook' button. Underneath the form, there's a preview box showing 'Name: test' and 'Message: This is a test comment.' At the bottom, there's a 'More Information' section with a bulleted list of links related to XSS, and buttons for 'View Source' and 'View Help'.

The screenshot shows a web browser window for the DVWA application, specifically the XSS (Stored) module. The URL is `dvwa.vm/vulnerabilities/xss_s/`. The main content area displays a guestbook form with the following fields:

- Name:
- Message:

Below the form, a message box shows the stored data:

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

A modal dialog box is displayed, indicating the exploit was successful:

⊕ dvwa.vm  
PHPSESSID=qj1v3ea0pdvb2alelrd5p98u6; security=low

**OK**

The left sidebar menu lists various DVWA modules, with "XSS (Stored)" currently selected. The bottom status bar shows the user is "admin" with "Security Level: low" and "PHPIDS: disabled".