

Stored Cross-Site Scripting

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The left sidebar has a green highlight under 'Setup / Reset DB'. The main content area is titled 'Database Setup' with a wrench icon. It contains a note about creating or resetting the database and a warning about clearing existing data. Below this is a 'Setup Check' section with various system status indicators. At the bottom is a red 'Status' message and a 'Create / Reset Database' button.

Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in: `/var/www/html/config/config.inc.php`

If the database already exists, **it will be cleared and the data will be reset**.
You can also use this to reset the administrator credentials ("admin // password") at any stage.

Setup Check

Operating system: *nix
Backend database: MySQL
PHP version: 5.5.9-1ubuntu4.21

Web Server SERVER_NAME: dvwa.example.com

PHP function display_errors: **Disabled**
PHP function safe_mode: **Disabled**
PHP function allow_url_include: **Enabled**
PHP function allow_url_fopen: **Enabled**
PHP function magic_quotes_gpc: **Disabled**
PHP module gd: **Installed**
PHP module mysql: **Installed**
PHP module pdo_mysql: **Installed**

MySQL username: root
MySQL password: *blank*
MySQL database: dvwa
MySQL host: 127.0.0.1

reCAPTCHA key: 6LdK7xIAzzAAJQfI7fu6I-0aPi8KHieAT_yJg

[User: root] Writable folder /var/www/html/hackable/uploads/: **Yes**
[User: root] Writable file /var/www/html/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt: **Yes**

Status in red, indicate there will be an issue when trying to complete some modules.

Create / Reset Database

CYSE 230 - Computer Networking

Spring 2023

dvwa.example.com/setup.php

The screenshot shows the DVWA Database Setup page. On the left, a sidebar menu includes 'Setup DVWA' (selected), 'Instructions', and 'About'. The main content area has a 'Database Setup' section with a note about creating or resetting the database. Below it is a 'Setup Check' section displaying various system details and configuration status. A red note at the bottom indicates potential issues. At the bottom right is a 'Create / Reset Database' button.

Setup DVWA

Instructions

About

Database Setup ↗

Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in: `/var/www/html/config/config.inc.php`

If the database already exists, **it will be cleared and the data will be reset**.
You can also use this to reset the administrator credentials ("admin // password") at any stage.

Setup Check

Operating system: *nix
Backend database: MySQL
PHP version: 5.5.9-1ubuntu4.21

Web Server SERVER_NAME: dvwa.example.com

PHP function display_errors: **Disabled**
PHP function safe_mode: **Disabled**
PHP function allow_url_include: **Enabled**
PHP function allow_url_fopen: **Enabled**
PHP function magic_quotes_gpc: **Disabled**
PHP module gd: **Installed**
PHP module mysql: **Installed**
PHP module pdo_mysql: **Installed**

MySQL username: **root**
MySQL password: ***blank***
MySQL database: **dvwa**
MySQL host: **127.0.0.1**

reCAPTCHA key: **6LdK7xITAAzAAJQTfL7fu6I-oaPI8KHieAT_yJg**

[User: root] Writable folder /var/www/html/hackable/uploads/: **Yes**
[User: root] Writable file /var/www/html/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt: **Yes**

Status in red, indicate there will be an issue when trying to complete some modules.

Create / Reset Database

For this task, I need to log in to the DVWA and set the security level to low. Then I clicked the XSS (Stored) button on the left column.

The screenshot shows the DVWA XSS (Stored) page. The left sidebar menu includes 'XSS (DOM)', 'XSS (Reflected)', 'XSS (Stored)' (selected), 'DVWA Security' (highlighted in green), 'PHP Info', and 'About'. The main content area has a dropdown for security level set to 'Low' with a 'Submit' button. Below it is a 'PHPIDS' section with a brief description and a note about its purpose.

XSS (DOM)

XSS (Reflected)

XSS (Stored)

DVWA Security

PHP Info

About

Low ▾ Submit

PHPIDS

PHPIDS v0.6 (PHP-Intrusion Detection System) is a security system that filters user supplied input against known attacks.

PHPIDS works by filtering any user supplied input against DVWA to serve as a live example of how Web Application Firewalls (WAFs) can be circumvented.

- In the form, I typed a name in the name submission box. Then I typed a message wrapped in a html h1 tag: `<h1>message here</h1>` and then clicked on the **Sign Guestbook** button.

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: Darth V

Message:

Welcome to the Dark
Side

Notice that the `<h1>` tag has changed the message to be a heading. This means we can inject HTML. Let's try a script tag next. Type the following: `<script>alert("Welcome to the Dark Side")</script>` and click on the **Sign Guestbook** button.

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

dvwa.example.com says

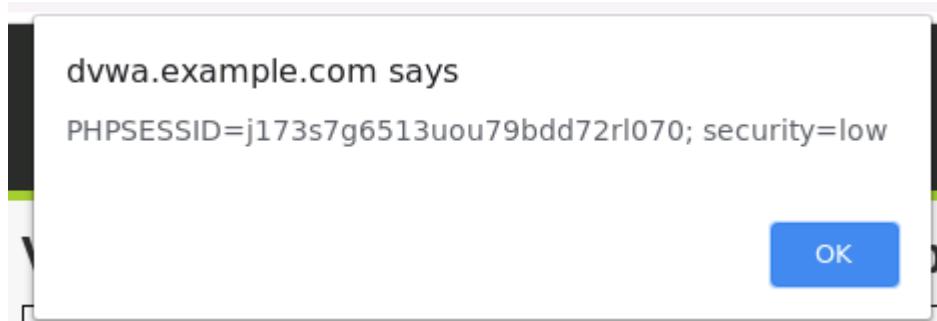
Welcome to the Dark Side

OK

I then clicked on the **OK** button. From here, it seems logical to grab the session cookie. I typed `<script>alert(document.cookie)</script>` and clicked on the **Sign Guestbook** button. Received both prompts as they are stored on the page.

Vulnerability: Stored Cross Site Scripting (XSS)

Name *	<input type="text" value="Darth V "/>
Message *	<input type="text" value="<script>alert(document.cookie)</script>"/>
<input type="button" value="Sign Guestbook"/>	



To be sure the XSS is stored on the page, I clicked on any other DVWA page and then clicked on the XSS (Stored) page again. Both of the XSS prompts appeared one at a time after clicking OK each time I saw a pop-up. If an attacker can get Stored XSS on a site, they can potentially gather all session cookies and even credentials by setting up a JavaScript keylogger.

Question 1: do some research online regarding basic JavaScript syntax. And then submit another message to display the image (<https://www.google.com/favicon.ico>) on the Stored XSS webpage. Explain what you have done to make it happen, and provide a screenshot of the webpage showing you have successfully inserted the image into the page.

To display an image on a Stored XSS vulnerability on the DVWA webpage, I created an `` element with the `src` attribute set to the URL of the desired image (<https://www.google.com/favicon.ico>). This crafted payload was then injected into a vulnerable input field on the webpage, exploiting the XSS vulnerability. When the page is loaded, the browser interprets the injected HTML code and renders the image specified in the payload.

The screenshot shows a guestbook form titled "Vulnerability: Stored Cross Site Scripting (XSS)". The "Name" field contains "Darth V". The "Message" field contains "". Below the form are three entries in the guestbook:

- Name: Darth V Message:
- Name: Darth V Message:
- Name: Darth V Message: G

A link "More Information" is visible at the bottom.

Task 2: DVWA Stored XSS on Medium Security

I reset the DB as shown on the first page. After resetting, I went to the XSS(stored) page and verified all previous messages I had posted had been removed.

On the sidebar menu, I clicked on the DVWA Security button and used the dropdown menu to change the security settings to Medium. Sometimes I have to do this more than once for it to work. I then checked the bottom of the page to be sure the settings had changed to medium.

The screenshot shows the DVWA sidebar with "DVWA Security" selected. On the right, under "PHPIDS", the security level is set to "Medium".

PHPIDS v0.6 (PHP Intrusion Detection System) is currently disabled.

You can enable PHPIDS across this site for the duration of your session. PHPIDS is currently: disabled. [Enable PHPIDS] [Simulate attack] - [View IDS log]

Username: admin
Security Level: medium
PHPIDS: disabled

I tried to use the previous technique by directly inserting a script, i.e., putting `<script>alert("Welcome to the Dark Side")</script>` in the message box, but I found that the script will not be executed any longer.

CYSE 230 - Computer Networking Spring 2023

I then navigated to the Stored XSS page and clicked OK on the alerts I created in the first task. If I look at the source by clicking on the **View Source** button located on the bottom right of the page, I can see there are a few changes to the security settings. The name submission box has been filtered by replacing the string `<script>` with double quotations.

The screenshot shows the DVWA Stored Cross Site (XSS) vulnerability page. On the left, there's a form with fields for 'Name' and 'Message', and a 'Sign Guestbook' button. Below the form, a guestbook entry is displayed: 'Name: test' and 'Message: This is a test comment.' To the right, a large block of PHP source code is shown, which includes several security sanitization and MySQL escaping mechanisms. The code uses trim(), stripslashes(), str_replace(), and mysql_real_escape_string() functions to filter user input.

```
<?php
if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = ((isset($GLOBALS["__mysqli_ston"])) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $message) : ((is_array($message)) ? MySQLConverterToo() Fix the mysql_escape_string() call! This code does not work., E_USER_ERROR) ? $message = htmlspecialchars( $message ) : $message;

    // Sanitize name input
    $name = str_replace( '<script>', '"', $name );
    $name = ((isset($GLOBALS["__mysqli_ston"])) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $name) : ((is_array($name)) ? MySQLConverterToo() Fix the mysql_escape_string() call! This code does not work., E_USER_ERROR) ? $name = htmlspecialchars( $name ) : $name;

    // Update database
    $query = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query) or die( '<pre>' . ((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) : ((is_array($GLOBALS["__mysqli_ston"])) ? implode("\n", $GLOBALS["__mysqli_ston"]) : $GLOBALS["__mysqli_ston"])) . "</pre>" );
    //mysql_close();
}
?>
```

In order to still exploit this, I need to use an exploit that does not require `<script>`. I tried to place a script into the name submission box.

- I closed out of viewing the source.
- In the name submission box, I typed: `<body onload=alert("medium")>`

The screenshot shows the DVWA Stored Cross Site Scripting (XSS) vulnerability page. The 'Name' field contains the value `<body onload=alert("medium")>`. The 'Message' field is empty. Below the form is a 'Sign Guestbook' button.

As you can see, the name submission box is only allowing 10 characters. I may be able to modify this using the developer tools.

- I right-clicked on the Name submission box and clicked on Inspect.

Vulnerability: Stored Cross Site Scripting (XSS)

The screenshot shows a web page with a guestbook form. The 'Name *' field contains the value '<body onlo'. A context menu is open over this field, with the 'Inspect' option highlighted. Below the form, three entries are visible in a list:

- Name: test
Message: This is a test comment.
- Name: Darth V
Message:
- Name: Darth V
Message:

Below the list, the HTML source code for the table row is displayed:

```
<tr>
  <td width="100">Name *</td>
  <td>
    <input name="txtName" type="text" size="30" maxlength="10">
  </td>
</tr>
```

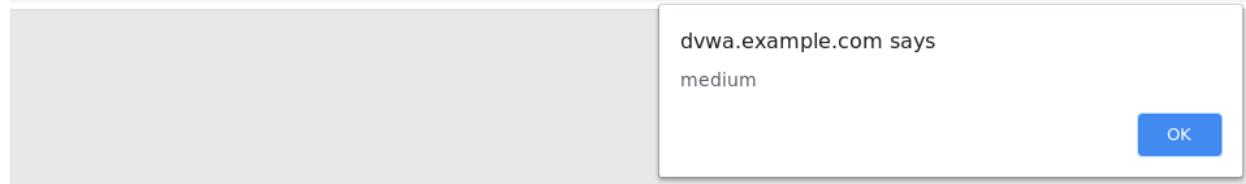
I took note of the **maxlength="10"**. Double-clicked on the 10 and changed it to 100. Since the check for this is on the client side and not the server side I can bypass the limitation. I tried to exploit again.

- In the name submission box, I typed `<body onload=alert("medium")>`, and in the message submission box, I type any message liked. Finally, I press the **Sign Guestbook** button.

The screenshot shows the same guestbook form. The 'Name *' field now contains the value '`<body onload=alert("medium")>`'. The 'Message *' field contains the value 'Blah'. The 'Sign Guestbook' button is visible at the bottom.

Once again, we can test to see if the XSS is stored by navigating to any other page, then back to the XSS (Stored) page.

dvwa.example.com/vulnerabilities/xss_s/



Question 2: Do a research online, find a second way, other than the <body> tag given in the example, that can also get the script executed. Explain what you have find, show that you have successfully override the length limit, and show that you have successfully have the script execute and the “medium” message popped out (you need to provide both verbal explanation and screenshots). Hint: you can get some inspiration from here <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>

A screenshot of the DVWA XSS module. The URL is dvwa.example.com/vulnerabilities/xss_s/. A modal dialog box is open, displaying the message "dvwa.example.com says medium" with an "OK" button. The background shows the guestbook form with the payload "" entered into the Name field. To the left, a portion of the page source code is visible, showing the injected script: <td width="100">Name *</td> <td> <input name="txtName" type="text" size="30" maxlength="100">

The script is a concise yet potent payload for exploiting Stored Cross-Site Scripting (XSS) vulnerabilities. By injecting this script into a vulnerable input field on the DVWA Stored XSS page, attackers can store it on the server. When other users access the page, the script triggers a JavaScript alert() function with the message "medium" whenever their mouse pointer hovers over the invisible image.