

# XSS/CSRF Attacks

## IMPORTANT NOTES:

**Study lecture materials at least 1 hour and prepare Lab Section 3 prior to the lab session to install DVWA. Prepared questions will be discussed in the lab session.**

## 1 Overview

The learning objective of this lab is for students to get familiar with vulnerabilities of and attacks targeting Web applications by using the XSS and CSRF attacks.

## 2 Lab Environment

In this lab we will exploit XSS/CSRF vulnerabilities on an intentionally vulnerable web server using the cloud VM. If you are using the local VM, you can skip Section 3, since the DVWA is already installed.

## 3 Install DVWA

Copy /srv/fit2093files/dvwa.zip to /var/www/html/ and unzip the archive. Rename the decompressed directory to DVWA. Change the owner and group of the decompressed directory and all files inside the directory to www-data and www-data, respectively (both the user and group have the same name www-data).

```
sudo cp /srv/fit2093files/dvwa.zip /var/www/html/
cd /var/www/html/
sudo unzip dvwa.zip
sudo mv DVWA-master/ DVWA/
sudo chown www-data:www-data -R DVWA/
```

Follow the instructions in “Database Setup” and “Other Configuration” Sections at <https://github.com/digininja/DVWA>. Here we summarise the necessary steps. Connect to the MySQL server and create the database for DVWA.

```
sudo mysql
```

```
MariaDB [(none)]> create database dvwa;
Query OK, 1 row affected (0.000 sec)
```

```
MariaDB [(none)]> create user dvwa@localhost identified by 'p@ssw0rd';
Query OK, 0 rows affected (0.033 sec)
```

```
MariaDB [(none)]> grant all on dvwa.* to dvwa@localhost;
Query OK, 0 rows affected (0.000 sec)
```

```
MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.000 sec)
```

```
MariaDB [(none)]> exit;
```

Change the file permissions:

```
sudo chmod +w /var/www/html/DVWA/hackable/uploads/  
sudo chmod +w /var/www/html/DVWA/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt
```

Change the following PHP configurations in /etc/php/7.4/apache2/php.ini. You may run a command line based text editor such as vi or nano with sudo to edit the file.

```
allow_url_include = On  
allow_url_fopen = On  
display_errors = On
```

Rename /var/www/html/DVWA/config/config.inc.php.dist to  
/var/www/html/DVWA/config/config.inc.php:

```
sudo mv /var/www/html/DVWA/config/config.inc.php.dist /var/www/html/DVWA/config/config.inc.php
```

Restart the Apache web server to make the changes above effective:

```
sudo /etc/init.d/apache2 restart
```

Open a web browser and access <http://127.0.0.1/DVWA/setup.php>. The information shown on the webpage should be similar to the following screenshot. If not, please check if any of the above steps is missing. Click the **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/DVWA/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

Web Server SERVER\_NAME: 127.0.0.1

Operating system: \*nix

PHP version: 7.4.3  
PHP function display\_errors: **Enabled** (Easy Mode!)  
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**

Backend database: **MySQL/MariaDB**  
Database username: **dvwa**  
Database password: **\*\*\*\*\***  
Database database: **dvwa**  
Database host: **127.0.0.1**  
Database port: **3306**

reCAPTCHA key: **Missing**

[User: www-data] Writable folder /var/www/html/DVWA/hackable/uploads/: **Yes**  
[User: www-data] Writable file /var/www/html/DVWA/external/phpids/0.6/lib/IDS/tmp/phpids\_log.txt: **Yes**

[User: www-data] Writable folder /var/www/html/DVWA/config: **Yes**  
**Status in red**, indicate there will be an issue when trying to complete some modules.

If you see disabled on either `allow_url_fopen` or `allow_url_include`, set the following in your php.ini file and restart Apache.

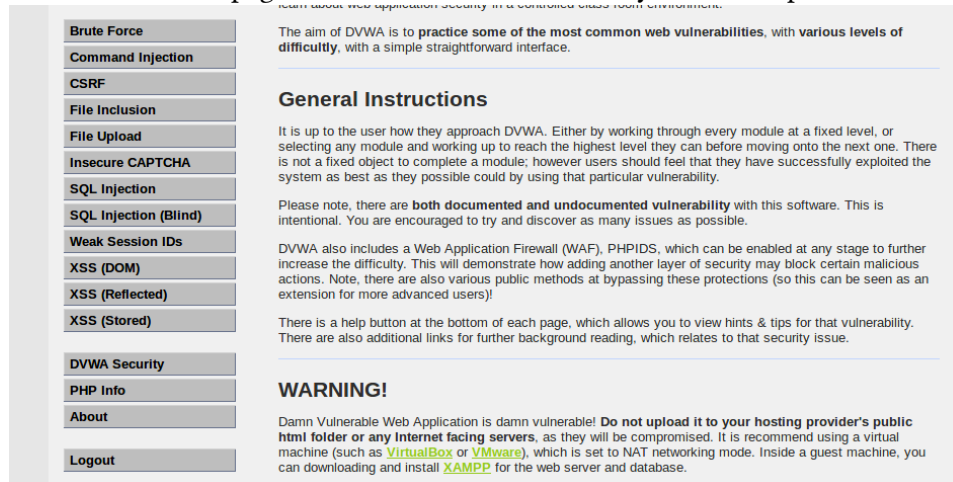
```
allow_url_fopen = On  
allow_url_include = On
```

These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.

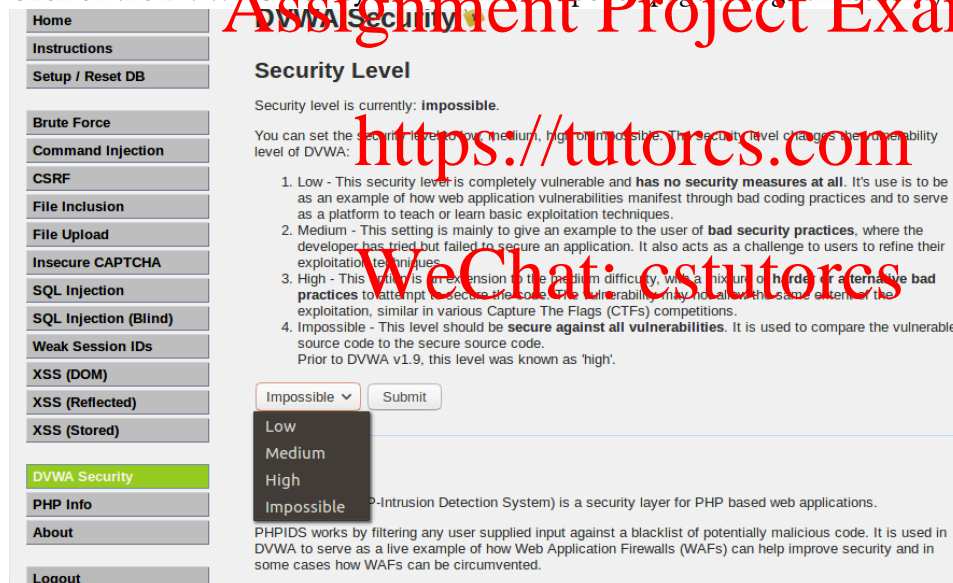
**Create / Reset Database**

## 4 Lab Tasks

Open the Firefox web browser and type `http://127.0.0.1/DVWA` in the address bar. Please note that the url is case sensitive. This should open the login page. Use admin as username and password as password. Once logged in scroll down the page and find the **DVWA Security** on the left pane.



Click on the **DVWA Security** and within the opened page change the security level to low and then click **Submit**.



### 4.1 CSRF

Read about the different types of CSRF attacks:

<https://owasp.org/www-community/attacks/csrf>

Make sure the **Security Level** is set to low. On the left pane click on the **CSRF** and perform the following steps.

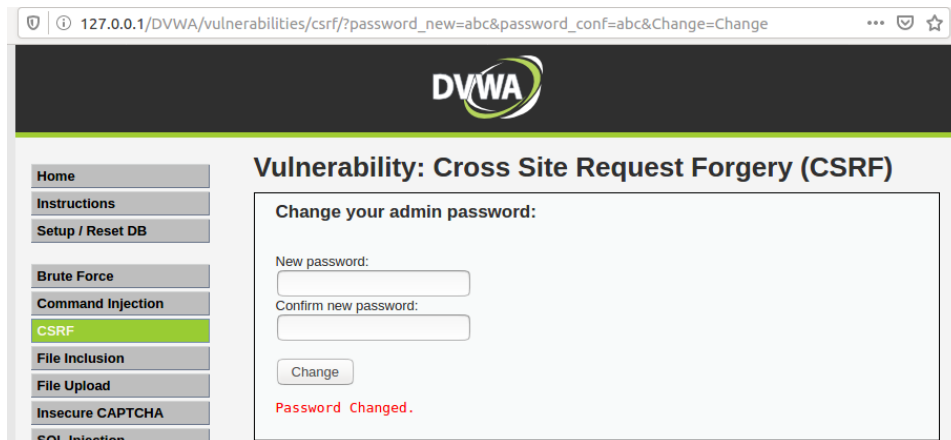
1. Enter the same password in both textboxes and click the “Change” button. Notice the change in the URL. How to forge such a link to trigger the changing admin password behaviour? You may need to logout and re-login to verify if the admin password has actually been changed.

For example, enter “test” in both textboxes and click the “Change” button. The URL in the address bar becomes:

[http://127.0.0.1/DVWA/vulnerabilities/csrf/?password\\_new=test&password\\_conf=test&Change=Change#](http://127.0.0.1/DVWA/vulnerabilities/csrf/?password_new=test&password_conf=test&Change=Change#)

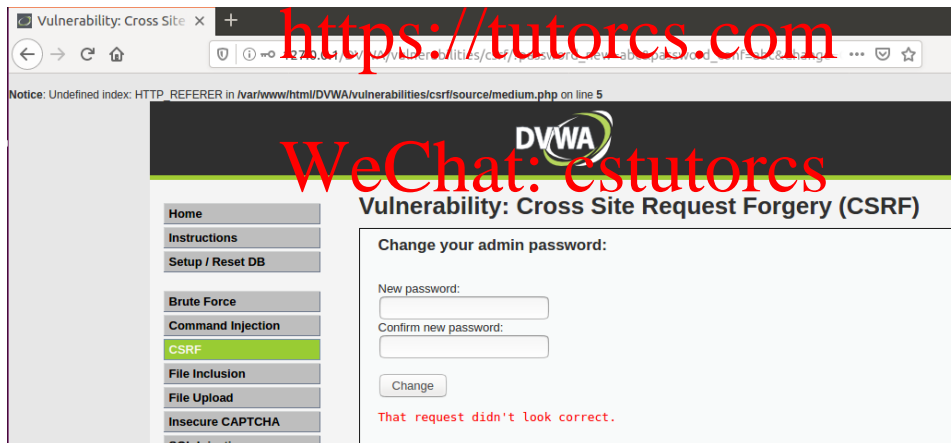
Now the attacker can just forge a similar URL, by replacing “test” with the password the attacker wants: e.g. directly access the following URL to change the admin password to “abc”:

[http://127.0.0.1/DVWA/vulnerabilities/csrf/?password\\_new=abc&password\\_conf=abc&Change=Change](http://127.0.0.1/DVWA/vulnerabilities/csrf/?password_new=abc&password_conf=abc&Change=Change)



2. Now set the Security Level to medium and repeat the previous step. View the source using the provided button. What check does the given code perform to prevent the previous simple CSRF attack?

The previous simple CSRF attack will not work anymore:



This is because the source code checks the “Referer” field in the HTTP request header:

```

<?php
if( isset( $_GET[ 'Change' ] ) ) {
    // Checks to see where the request came from
    if( eregi( $_SERVER[ 'SERVER_NAME' ], $_SERVER[ 'HTTP_REFERER' ] ) ) {
        // Get input
        $pass_new = $_GET[ 'password_new' ];
        $pass_conf = $_GET[ 'password_conf' ];

        // Do the passwords match?
        if( $pass_new == $pass_conf ) {
            // They do!
            $pass_new = mysql_real_escape_string( $pass_new );
            $pass_new = md5( $pass_new );

            // Update the database
            $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "'";
            $result = mysql_query( $insert ) or die( "<pre>" . mysql_error() . "</pre>" );

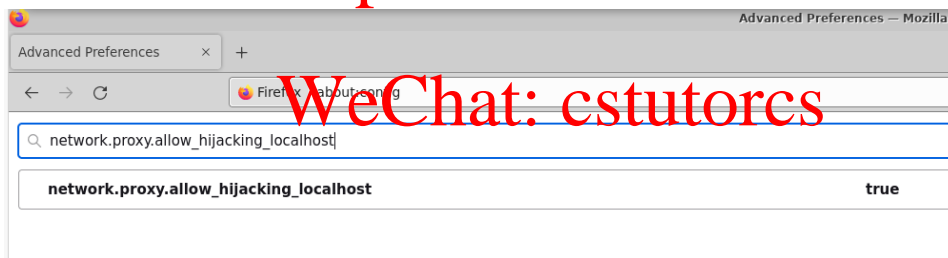
            // Feedback for the user
            echo "<pre>Password Changed.</pre>";
        }
        else {
            // Issue with passwords matching
            echo "<pre>Passwords did not match.</pre>";
        }
    }
    else {
        // Didn't come from a trusted source
        echo "<pre>That request didn't look correct.</pre>";
    }
}

mysql_close();
}
?>

```

To get around this check, we need to forge a malicious HTTP request. We can use the Burpsuite tool, which will create a proxy server in order to intercept and modify the HTTP request sent from the web browser to the server.

In order to use a local proxy server in Firefox, we need to change the settings. Access about:config from the Firefox web browser. Change network.proxy.allow\_hijacking\_localhost to true.



To perform the task, first, we need to install the Burpsuite. Copy the installer /srv/fit2093files/burpsuite\_community\_linux\_v2021\_12\_1.sh to your home directory. Make this file executable and run the installer. Follow the instructions.

```

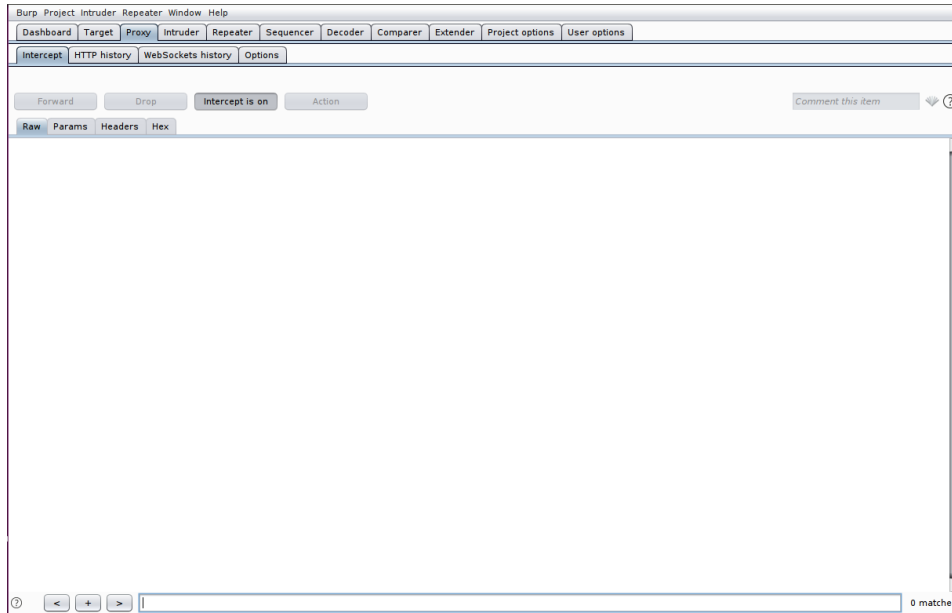
cp /srv/fit2093files/burpsuite_community_linux_v2021_12_1.sh ~
cd
chmod +x burpsuite_community_linux_v2021_12_1.sh
./burpsuite_community_linux_v2021_12_1.sh

```

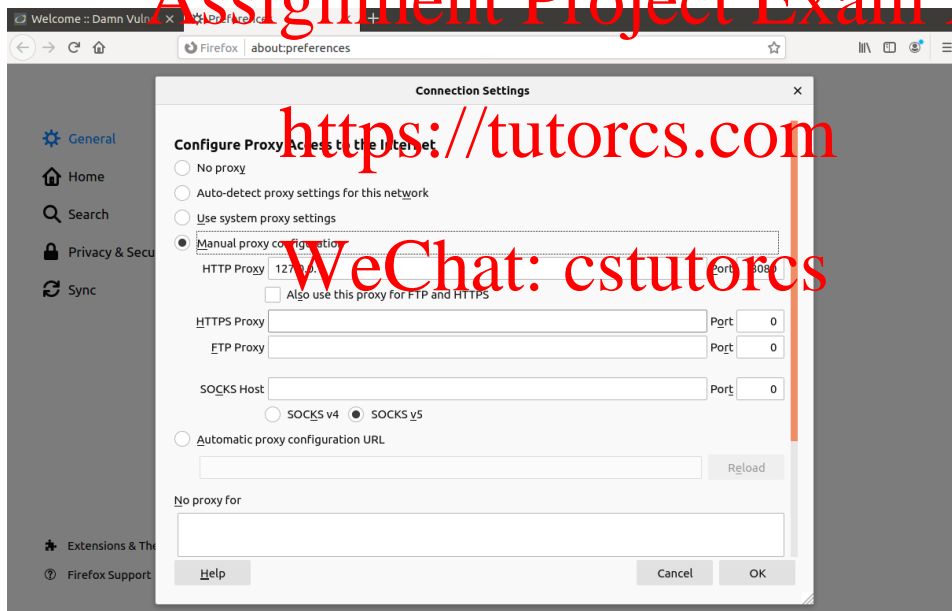
Then, open a terminal and run the following command to launch the Burpsuite tool:

```
java -jar ~/BurpSuiteCommunity/burpsuite_community.jar
```

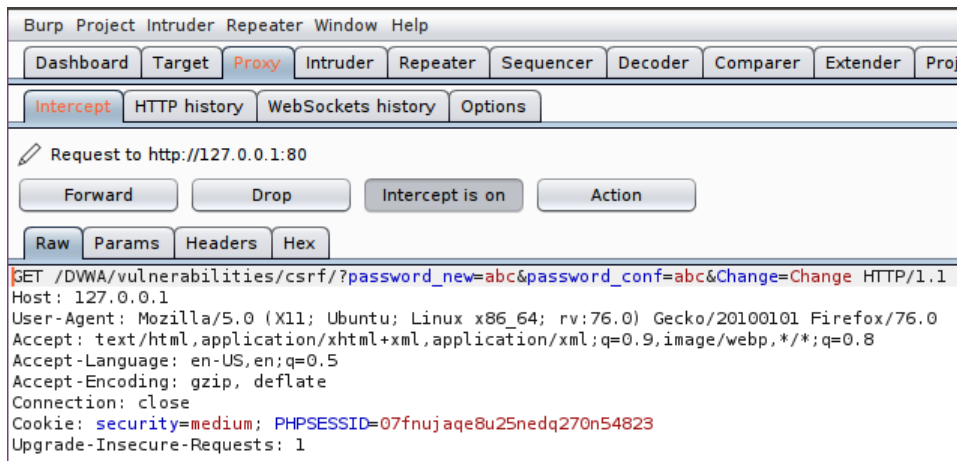
Ignore all the warnings. Use the default settings to create a temporary project and start Burpsuite. Click the "Proxy" tab. Make sure the button is showing "Intercept is on".



Now go to the “Preferences” of the Firefox web browser. Change the “Network Settings” and add the following proxy server:



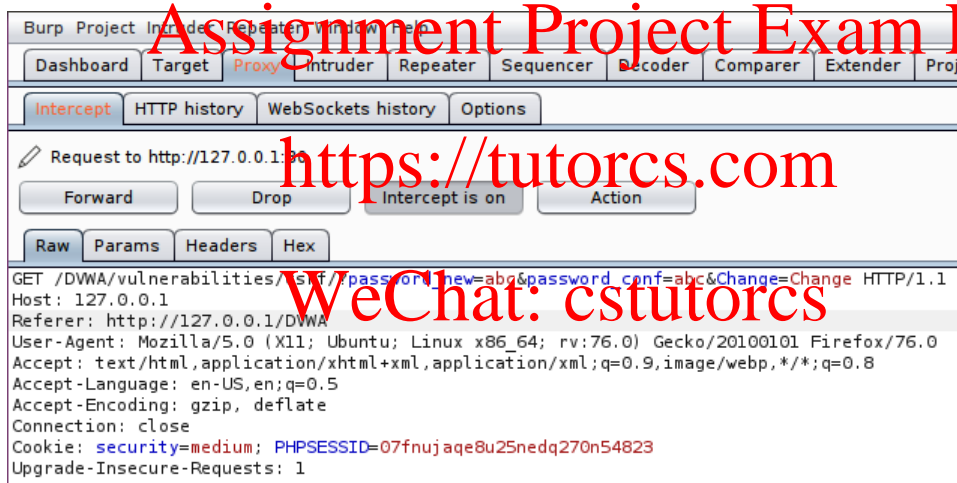
Access the URL forged by the previous simple CSRF attack in the web browser. Note that Burpsuite will intercept the HTTP request sent from the web browser to the server:



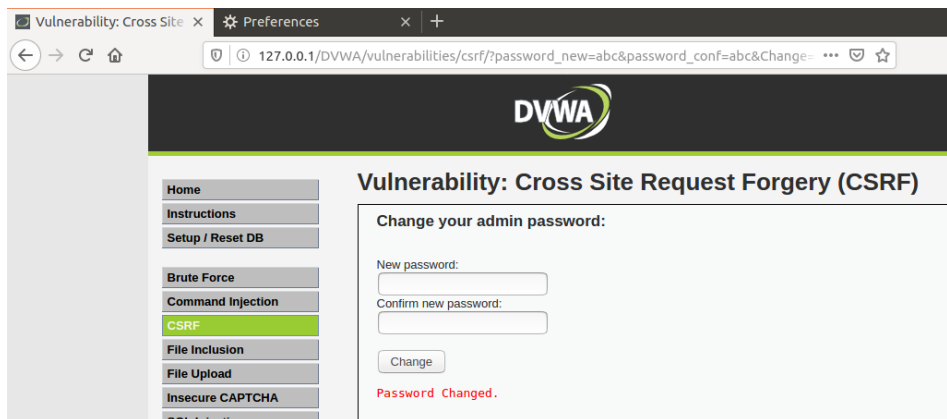
Based on your finding from the source code, what field in the HTTP request header should you add/modify to get around the check?

Add the "Referer" field in the HTTP request header, with the value being the URL of the website:

Referer: http://127.0.0.1/DVWA



Click the "Forward" button. Now the web page will show "Password Changed".



## 4.2 Reflected XSS Attack

Read about the different types of XSS attacks:

[https://www.owasp.org/index.php/Cross-site\\_Scripting\\_\(XSS\)#Stored\\_and\\_Reflected\\_XSS\\_Attacks](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)#Stored_and_Reflected_XSS_Attacks)

Open the DVWA. Make sure the **Security Level** is set to low. On the left pane click on the **XSS (Reflected)** and perform the following steps.

We can use the examples from:

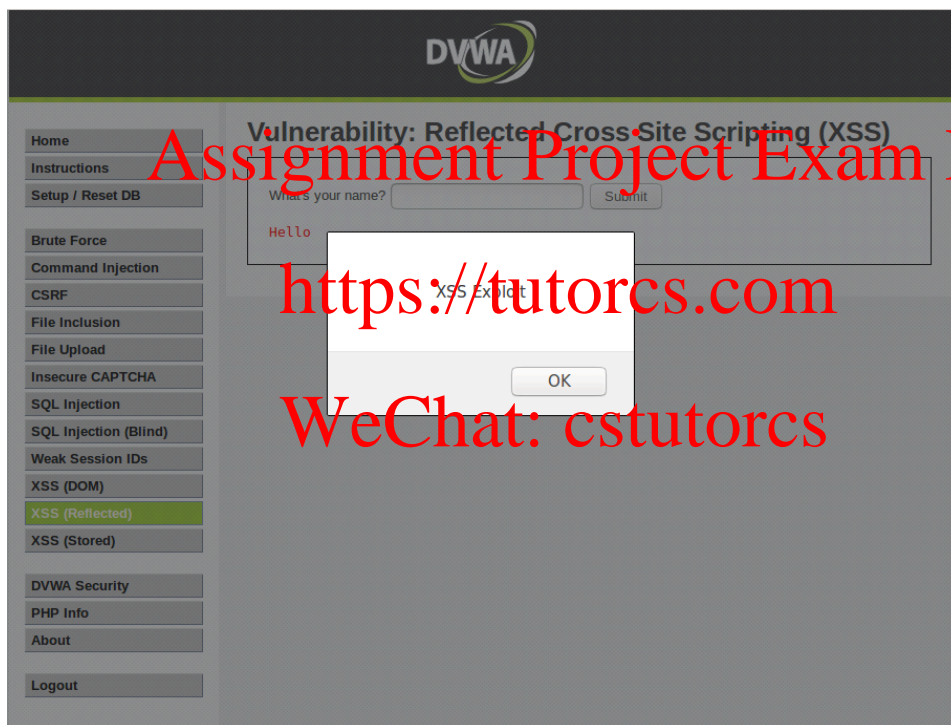
[https://www.owasp.org/index.php/Testing\\_for\\_Reflected\\_Cross\\_site\\_scripting\\_\(OTG-INPVAL-001\)](https://www.owasp.org/index.php/Testing_for_Reflected_Cross_site_scripting_(OTG-INPVAL-001)) to construct our tests.

1. Enter the following in the text box:

```
<script>alert("XSS Exploit")</script>
```

and submit. Explain the result.

This will execute a Java script and run the alert function which will open a window with provided message.



2. Now set the **Security Level** to medium and repeat the previous step. View the source using the provided button. What function in the given code prevents the use of `<script>`? Enter the following in the text box:

```
<SCRIPT>alert("XSS Exploit");</SCRIPT>
```

and submit. Explain the result.

This time the `<script>` keyword is removed and the code of the script appears as the content (name of the visitor).



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

What's your name?

Hello alert("XSS Exploit")

**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  
Security Level: medium  
PHPIDS: disabled

View Source View Help

Assignment Project Exam Help

<https://tutorcs.com>

The `str_replace()` function performs this task.

**Reflected XSS Source**

```
<?php
header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = str_replace( '<script>', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

?>
```

Compare All Levels

Since the function only looks for `<script>` keyword we can try to enter the script keyword differently so it

will not match the exact `<script>`. We can change `<script>` to uppercase:  
`<SCRIPT>alert("XSS Exploit");</SCRIPT>`.

- If you succeeded in bypassing the medium security level, now set the **Security Level** to high. Does your previous work around succeed at this level? Click on the **View source** button and observe the change. Enter the following in the text box:

```
<img src=1 href=1 onerror="alert('XSS')"></img>
```

and submit. Explain the result.

The screenshot shows the 'Reflected XSS Source' for the High security level. The PHP code uses `preg_replace` to filter out script tags. A large red watermark 'Assignment Project Exam Help' is overlaid on the image.

```
<?php
header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = preg_replace( '/<(.*s.*)c(.*?)i(.*?)p(.*?)t/i', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

?>
```

Compare All Levels

In high security level the function `preg_replace` looks for one or more opening bracket for script keyword and applies some additional level of regular express. The keyword replacement however is only looking for script and neglects other keywords that may load scripts. Now we can use other keywords for instance:

```
<img src=1 href=1 onerror="alert('XSS')"></img>
```

- If you succeeded in exploiting the high security level then change the **Security Level** to impossible. Try the exploit from previous step. Does it succeed in this security level? Look at the source what changes are made?

It does not succeed. Looking at the source:

The screenshot shows the 'Reflected XSS Source' for the Impossible security level. The PHP code uses `htmlspecialchars` to escape the input. A large red watermark 'WeChat: cstutorcs' is overlaid on the image.

```
<?php
// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $name = htmlspecialchars( $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

// Generate Anti-CSRF token
generateSessionToken();

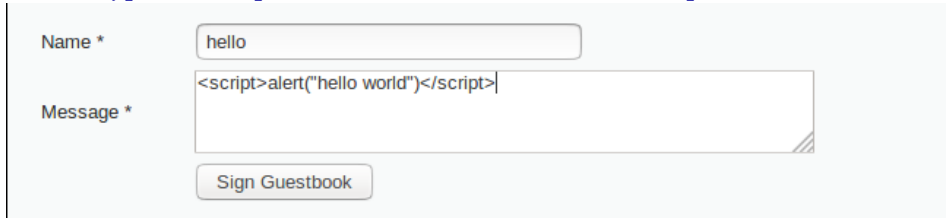
?>
```

The function `htmlspecialchars()` takes care of any keyword that can be interpreted by html.

## 5 Further Exploitation: Stored XSS Attack

Now set the **Security Level** back to low and click on the **XSS (Stored)** on the left pane. Could you try to launch a stored XSS attack by using a similar script as in the reflected XSS task?

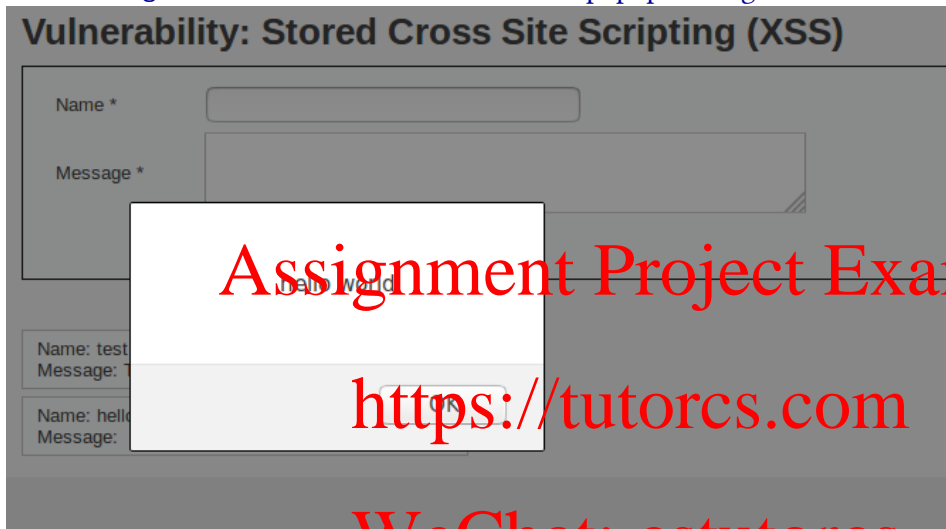
We can type in a script in the Message textbox. For example: `<script>alert("hello world")</script>`.



Name \*

Message \*

Click on Sign Guestbook and there should be a popup message:



Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs