Assignment 4

Zhicheng Zhao: 015221102

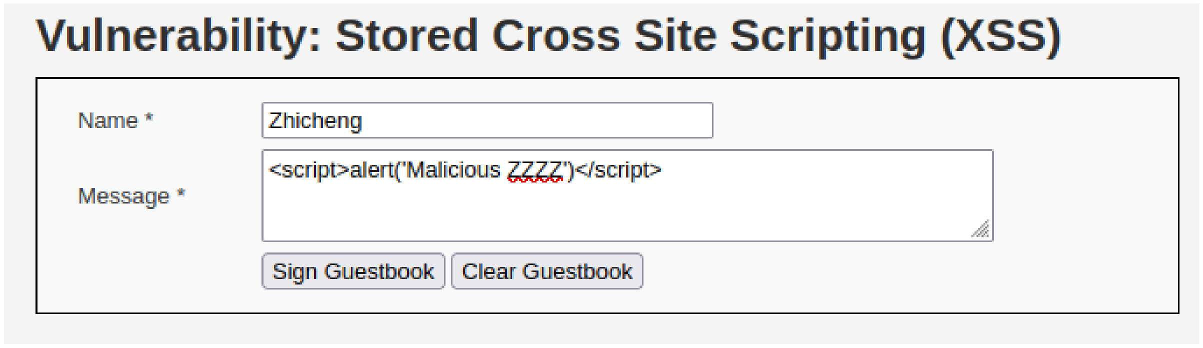
Xuewei Zheng: 012066314

Stored XSS

Question 1: Describe the attack you used. How did it work?

Stored XSS is one of the XSS attack. The main characteristic for stored XSS is, it stores the input and embeds the input into a later response in an unsafe manner.

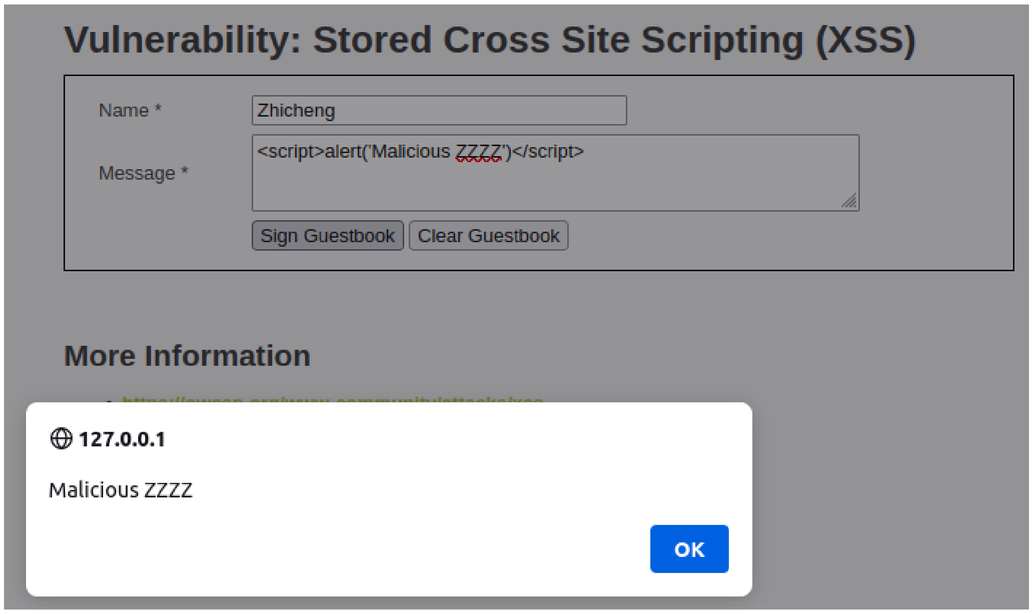
Set security level to low, select XSS (Stored) section on the left side. We can input characters inside the name and malicious script inside the message text box.


Vulnerability: Stored Cross Site Scripting (XSS)

Name * Zhicheng

Message * <script>alert('Malicious ZZZZ')</script>

Then click Sign Guestbook button, and we will get the pop-up alert message


Vulnerability: Stored Cross Site Scripting (XSS)

Name * Zhicheng

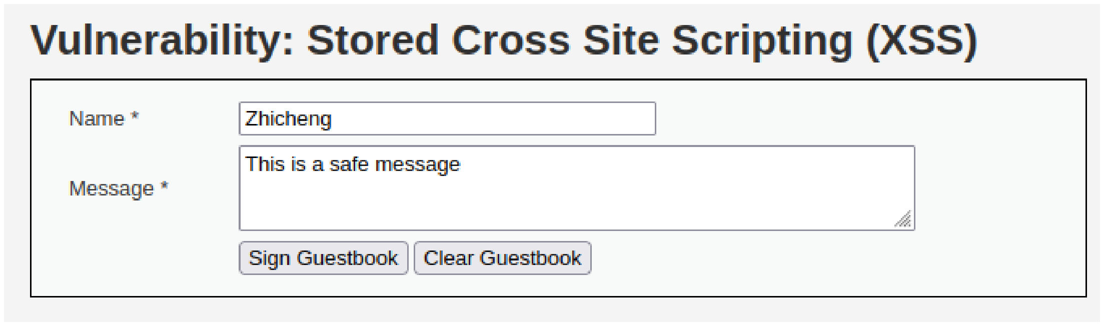
Message * <script>alert('Malicious ZZZZ')</script>

More Information

127.0.0.1

Malicious ZZZZ

Then let’s try input just characters inside the Message textbox

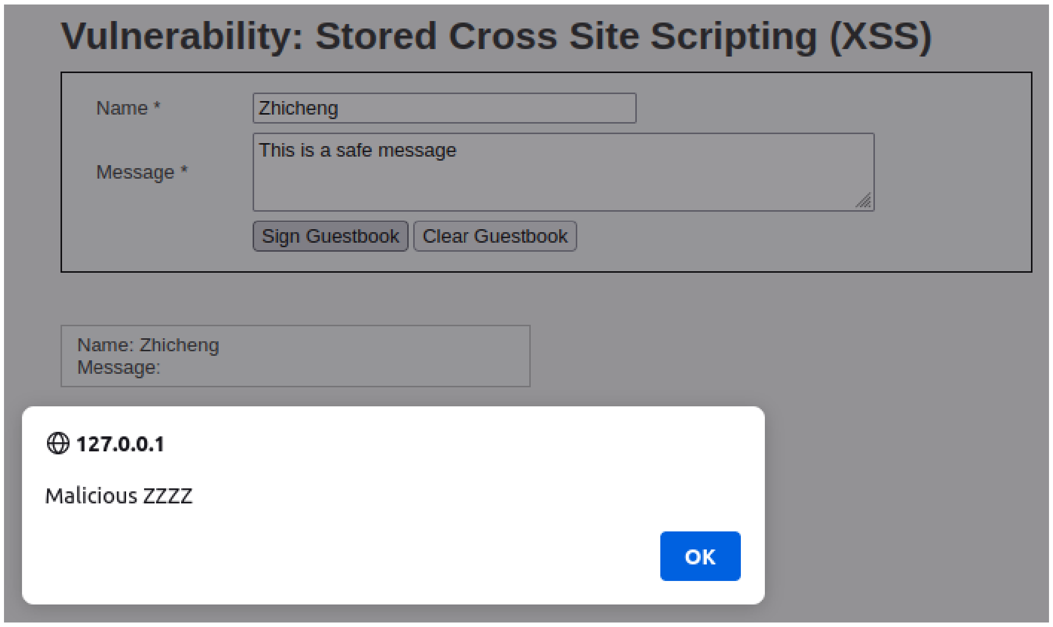


Vulnerability: Stored Cross Site Scripting (XSS)

Name * Zhicheng

Message * This is a safe message

Then click Sign Gustbook button, and we will still get the malicious pop-up alert message, instead of the character inside Message textbox. Which means the malicious script was stored and embeds the input into a later response. To clean up the stored malicious script, we need to click the Clear Guestbook button.



Vulnerability: Stored Cross Site Scripting (XSS)

Name * Zhicheng

Message * This is a safe message

Name: Zhicheng
Message:

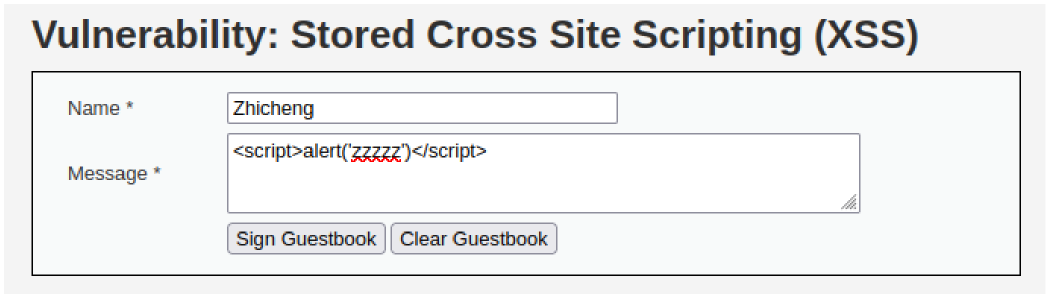
127.0.0.1

Malicious ZZZZ

Question 2: Does your attack work in “Medium” security level?

It doesn’t work as security level is medium.

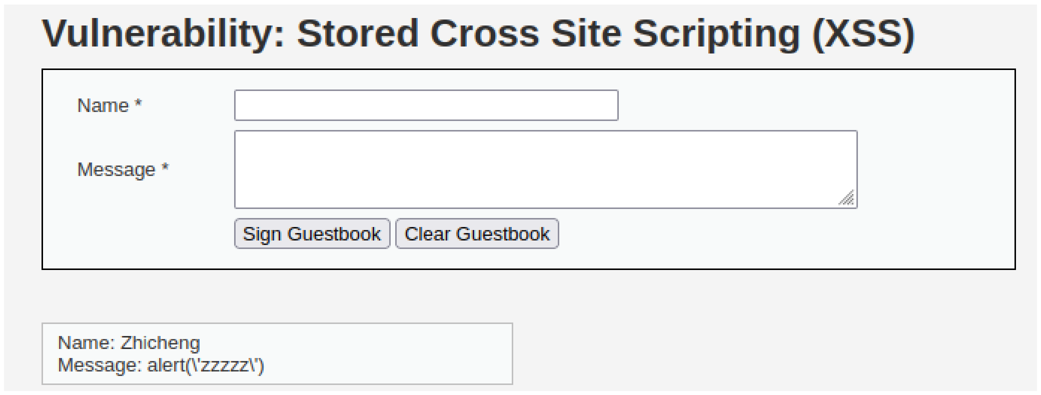
Input the script as bellow,


Vulnerability: Stored Cross Site Scripting (XSS)

Name * Zhicheng

Message * <script>alert('zzzzz')</script>

Then click Sign Guestbook, and we get


Vulnerability: Stored Cross Site Scripting (XSS)

Name *

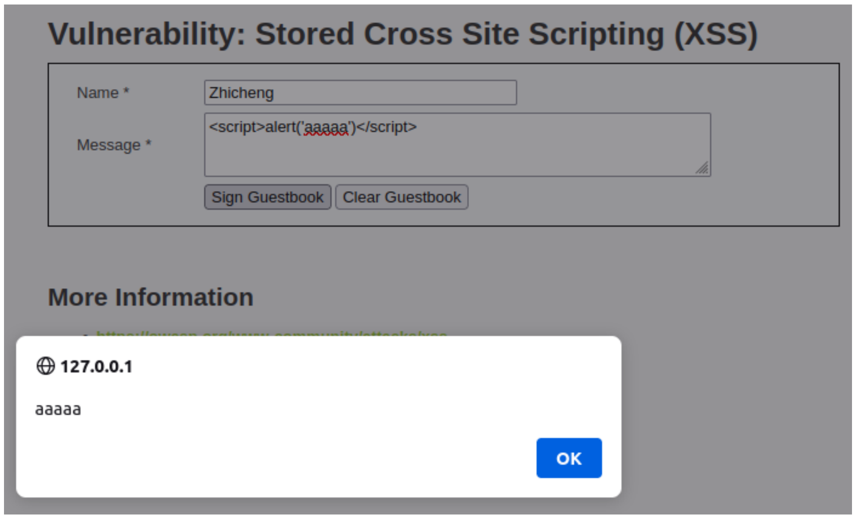
Message *

Name: Zhicheng
Message: alert(\'zzzzz\')

The block inside <script> tag was escaped, and output as characters instead of execute as code.

Question 3: Set the security mode to "Low" and examine the code that is vulnerable, and then set the security mode to "High" and reexamine the same code. What changed? How do the changes prevent the attack from succeeding?

Set security level to low, input the following and execute, we will get the alert message.


Vulnerability: Stored Cross Site Scripting (XSS)

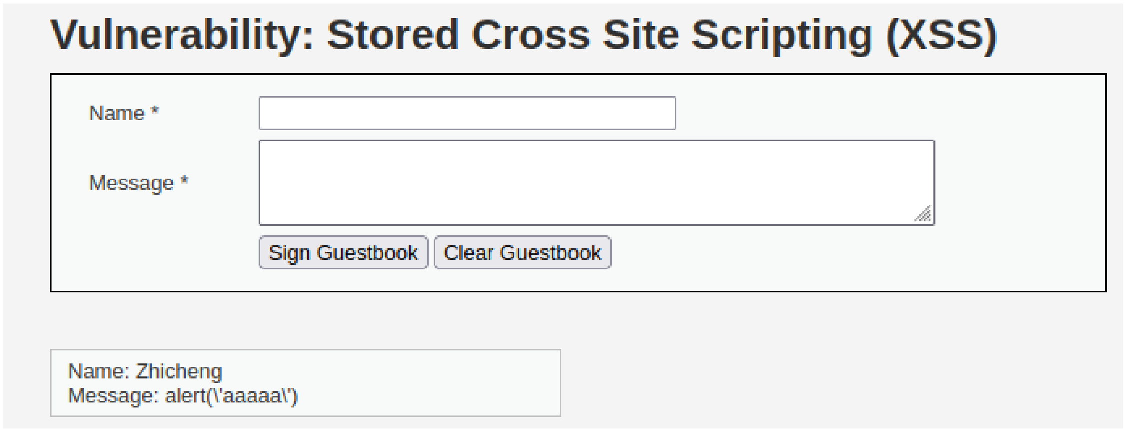
Name * Zhicheng

Message * <script>alert('aaaaa')</script>

127.0.0.1

aaaaa

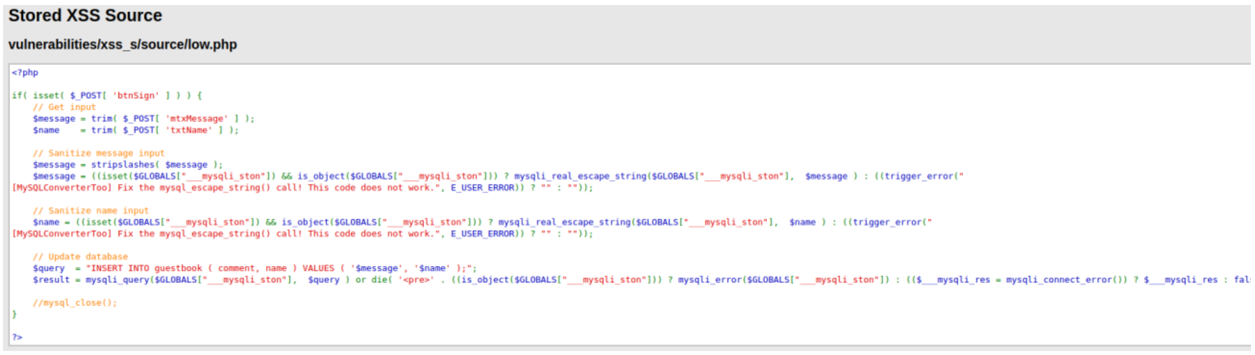
Set security level to high, input the same script and execute, we will not get the alert



Vulnerability: Stored Cross Site Scripting (XSS)

Name: Zhicheng
Message: alert(\'aaaaa\')

Low security level code examine

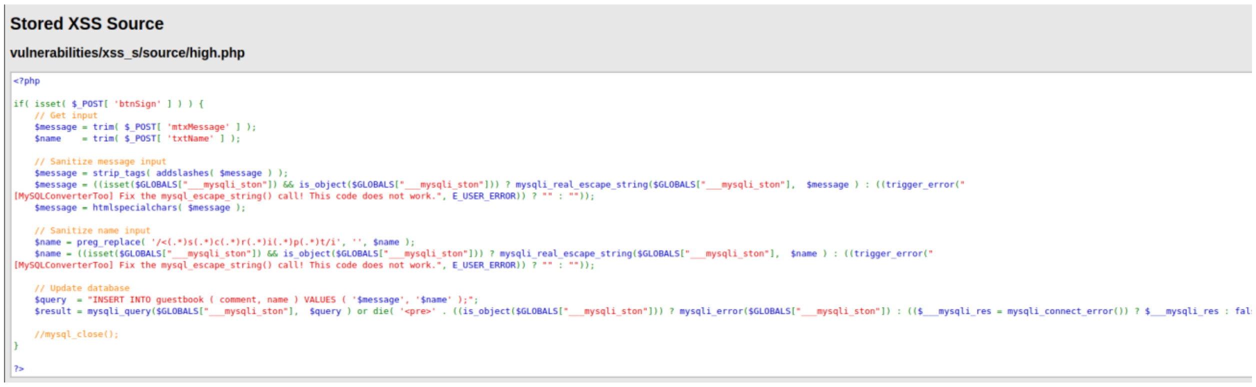


Stored XSS Source

vulnerabilities/xss_s/source/low.php

```
<?php

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

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

    // Sanitize name input
    $name = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $name ) : ((trigger_error("
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));

    // 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($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : fal

    //mysql_close();
}

?>
```

High security level code examine



Stored XSS Source

vulnerabilities/xss_s/source/high.php

```
<?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 ) : ((trigger_error("
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $name );
    $name = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $name ) : ((trigger_error("
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));

    // 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($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : fal

    //mysql_close();
}

?>
```

Inside message field, high level code uses strip_tags() function to strip the html tags, and htmlspecialchars() function to convert to html character. So that the the html tags will be removed, and code inside html tag will be output as characters.