XSS (Cross-Site-Script) (Stored)

Cross-Site-Script is a potentially fatal attack in which a function not considered by the developer works by inserting script code such as JavaScript into a bulletin board or webmail. Thus, it is an attack tar geting users.

When the XSS attack takes place

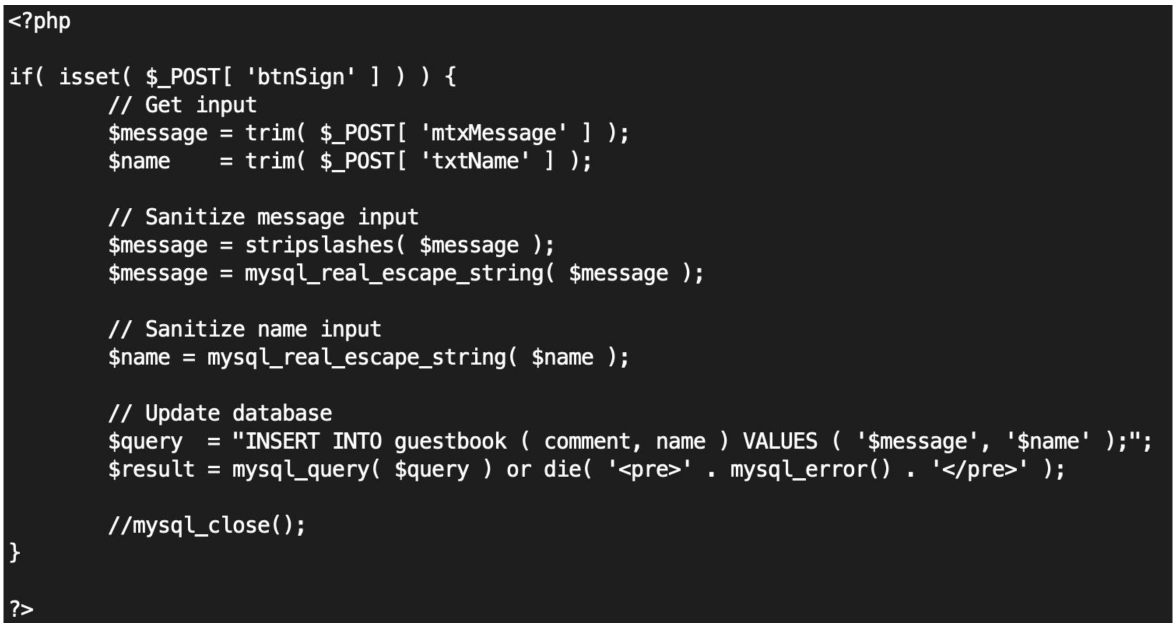
1. With frequent web requests, when data is entered into a web a pplication through untrusted source code.

2. When you have dynamic content that transmits malicious conte nt to web users whose data has not been verified

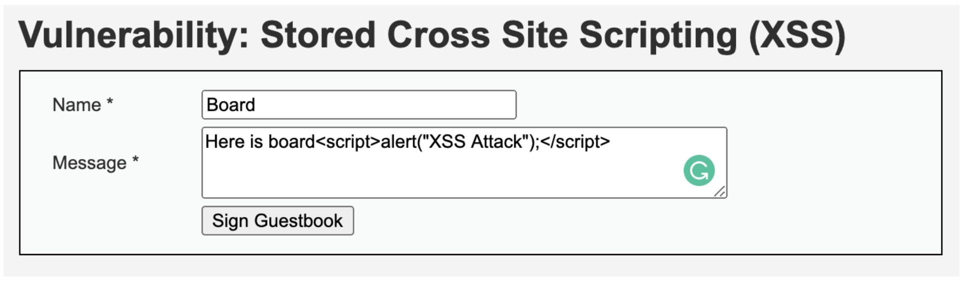
Stored XSS (AKA Persistent or Type 1)

Stored XSS generally occurs when user input is stored on the target server, such as in a database, in a message forum, visitor log, comment field, etc. And then a victim is able to retrieve the stored data from the web application without that data being made safe to render in the browser. With the advent of HTML5, and other browser technologies, we can envision the attack payload being permanently stored in the victim's browser, such as HTML5 database, and never being sent to the server at all.

<Low>

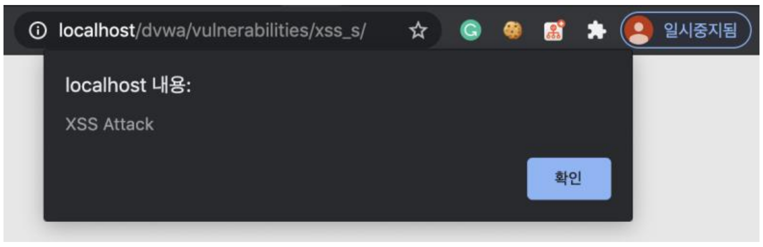


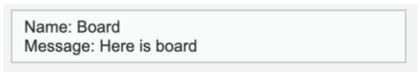
```
<?php

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

	// Sanitize message input
	$message = stripslashes( $message );
	$message = mysql_real_escape_string( $message );

	// Sanitize name input
	$name = mysql_real_escape_string( $name );

	// Update database
	$query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
	$result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

	//mysql_close();
}

?>
```


Vulnerability: Stored Cross Site Scripting (XSS)

Name * Board

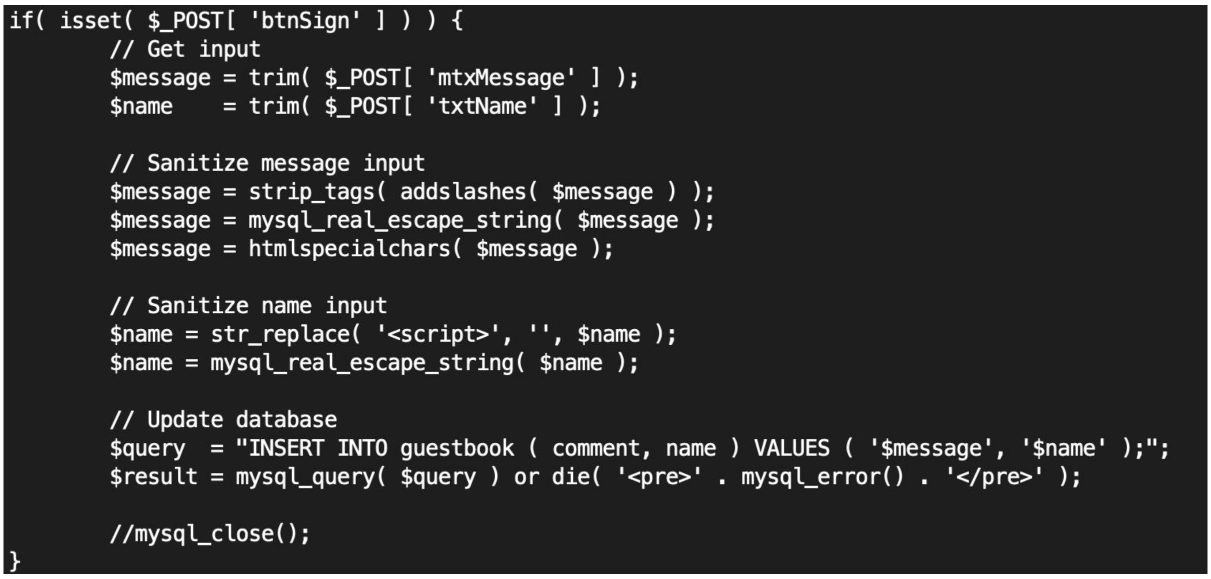
Message * Here is board<script>alert("XSS Attack");</script>




Name: Board
Message: Here is board

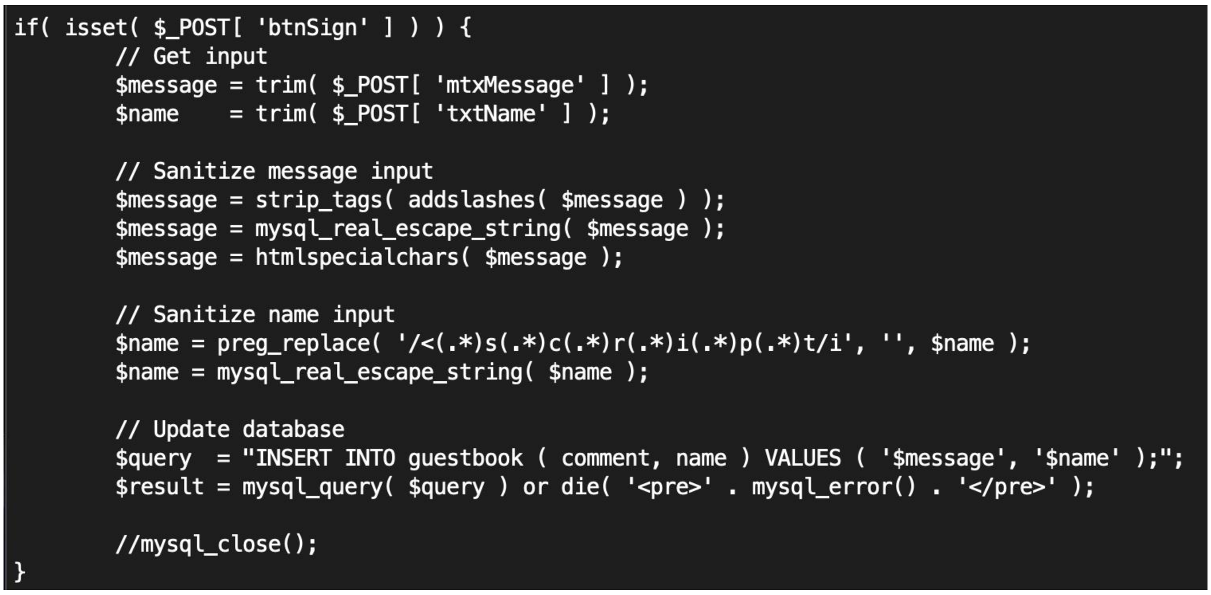
Users can’t show the script contents on the board. Thus, if specific user confirms the board, the script code is automatically executed.

<Medium>


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

        // Sanitize message input
        $message = strip_tags( addslashes( $message ) );
        $message = mysql_real_escape_string( $message );
        $message = htmlspecialchars( $message );

        // Sanitize name input
        $name = str_replace( '<script>', '', $name );
        $name = mysql_real_escape_string( $name );

        // Update database
        $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
        $result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

        //mysql_close();
}
```
medium.php

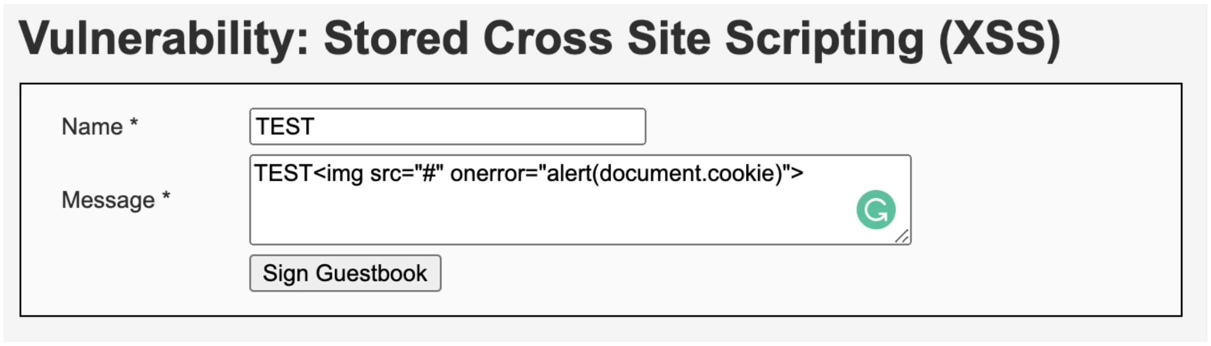
This code can’t use the script tag. Because there is a function where script tag is replaced with ‘’.

<High>



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

        // Sanitize message input
        $message = strip_tags( addslashes( $message ) );
        $message = mysql_real_escape_string( $message );
        $message = htmlspecialchars( $message );

        // Sanitize name input
        $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $name );
        $name = mysql_real_escape_string( $name );

        // Update database
        $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
        $result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

        //mysql_close();
}
```

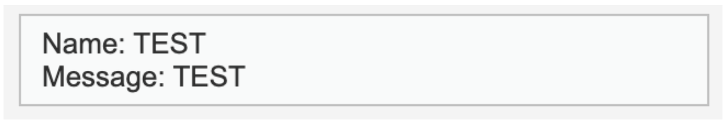
This code can’t use the script tag. Because there is a function where script tag is replaced with ‘’.



Vulnerability: Stored Cross Site Scripting (XSS)

Name * TEST

Message * TEST<img src="#" onerror="alert(document.cookie)">

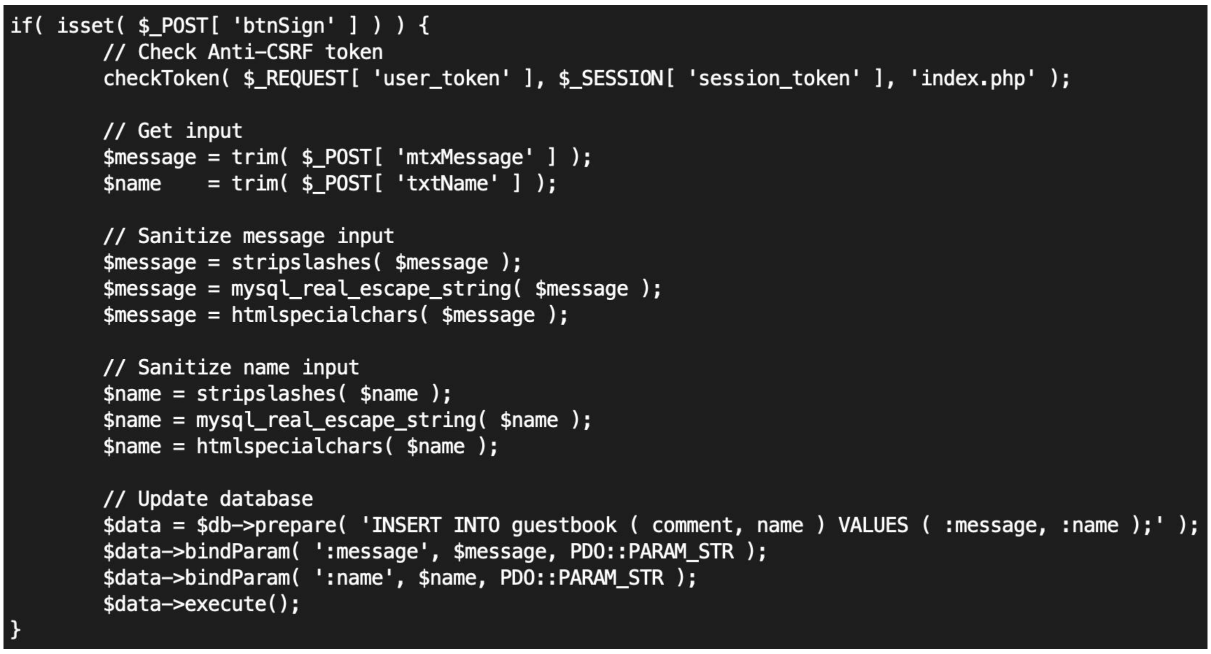


Name: TEST
Message: TEST

As you see, you can attack through the img tag. If you use the various script code like this, the attacker can gain the cookie information of specific user.

Summary
After an attacker stores malicious code on the server using a board or pop-up window, it is to attack of data interception or malicious code installation without the user knowing by working whenever users open it.

<Security>



```
if( isset( $_POST[ 'btnSign' ] ) ) {
	// Check Anti-CSRF token
	checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

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

	// Sanitize message input
	$message = stripslashes( $message );
	$message = mysql_real_escape_string( $message );
	$message = htmlspecialchars( $message );

	// Sanitize name input
	$name = stripslashes( $name );
	$name = mysql_real_escape_string( $name );
	$name = htmlspecialchars( $name );

	// Update database
	$data = $db->prepare( 'INSERT INTO guestbook ( comment, name ) VALUES ( :message, :name );' );
	$data->bindParam( ':message', $message, PDO::PARAM_STR );
	$data->bindParam( ':name', $name, PDO::PARAM_STR );
	$data->execute();
}
```

If a programmer use the token, XSS Attack is impossible like this.