

Vulnerability Name:	Reflected Cross Site Scripting (XSS)
Affected Vendor:	DVWA
Affected Product Name:	http://dvwa/vulnerabilities/xss_r/
Product Official Website URL:	http://dvwa/login.php
Affected Component:	Affected Parameters

Description: - Reflected Cross-Site Scripting (XSS) is a cybersecurity vulnerability where attackers inject malicious scripts into a web application. These scripts are immediately reflected and executed in the browsers of unsuspecting users. Unlike Stored XSS, where the script is saved on the server, Reflected XSS occurs when the injected payload is included in the server's response without proper validation or sanitization of user input. This typically happens in dynamically generated content, such as search results, error messages, or URL parameters.

Root Cause: - The root cause of Reflected XSS is the lack of proper input validation and output encoding in web applications, leading to the unintended execution of user-supplied scripts.

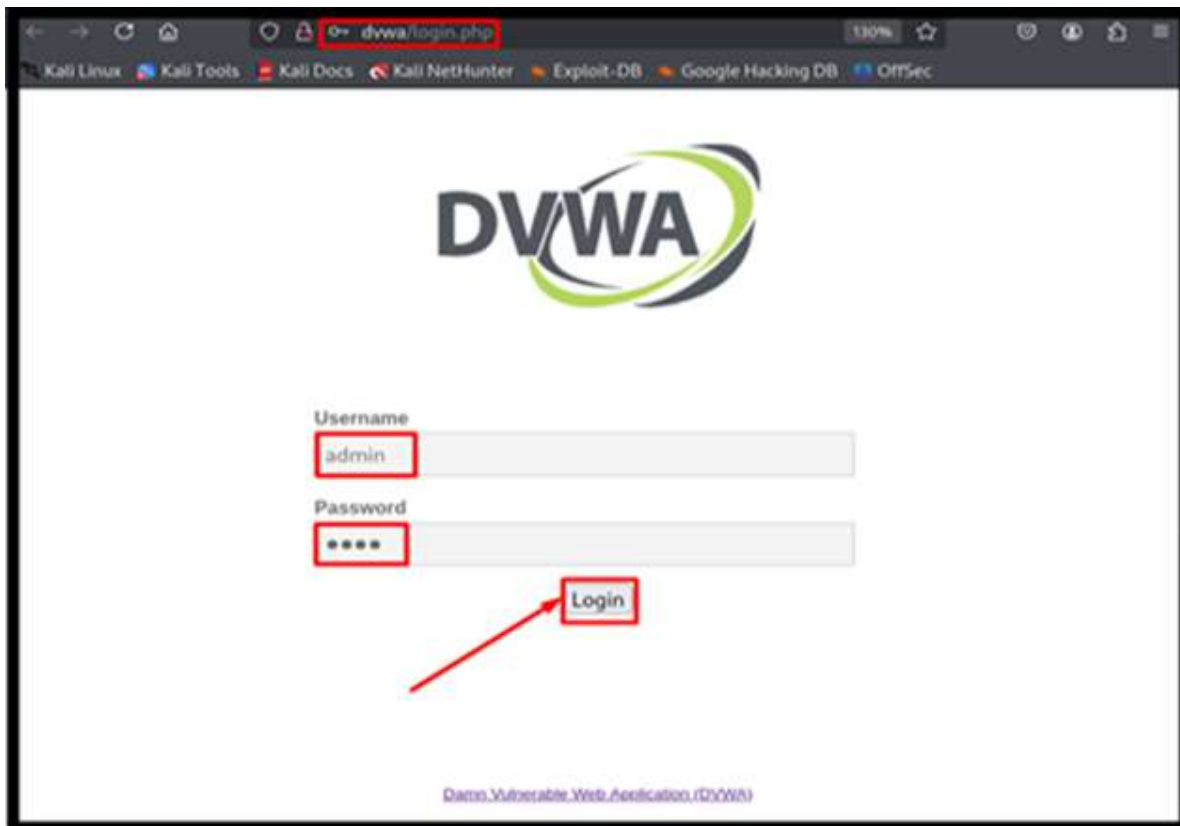
Impact: - The impact of Reflected Cross-Site Scripting can range from stealing sensitive information, such as login credentials or personal data, to performing actions on behalf of the user without their consent. Additionally, attackers can use Cross-Site Scripting to deliver malware, deface web pages, or conduct phishing attacks, potentially causing reputational damage to the affected organization.

Mitigation: - Implement strict input validation, sanitize user input, and use output encoding to prevent XSS attacks. Use Content Security Policy (CSP) headers to restrict the execution of scripts.

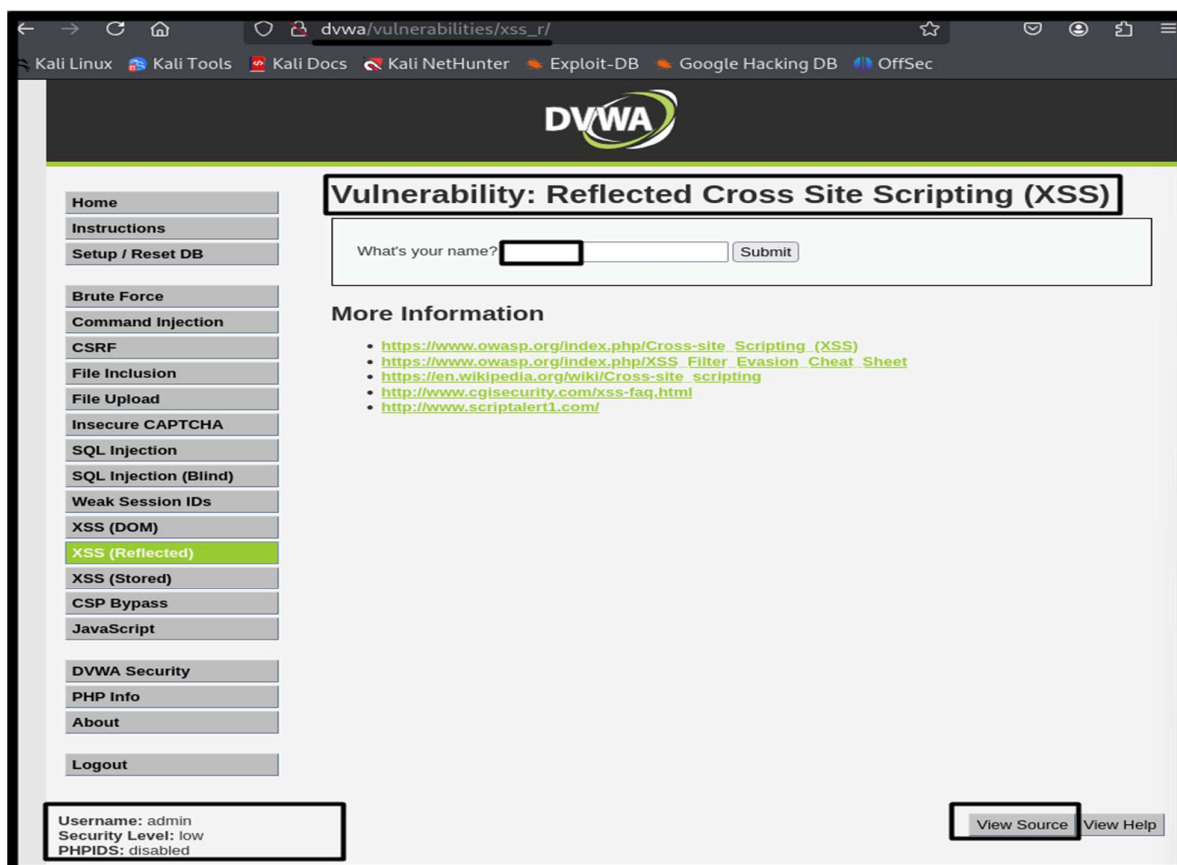
Remediation: - Developers must adopt secure coding practices, including input validation, output encoding, and Content Security Policy (CSP) implementation. Regular security audits and the utilization of web application firewalls are also essential for detecting and mitigating XSS vulnerabilities.

Proof Of Concept

Step: -1 First navigate to <http://dvwa/login.php> and login with username and Password.



Step: -2 log in the home page of DVWA then click to the Reflected Cross Site Scripting (XSS) Section.



SECURITY LEVEL(LOW)

As we know we will first view the source code. There is no filtration at all. We are directly able to apply javascript.

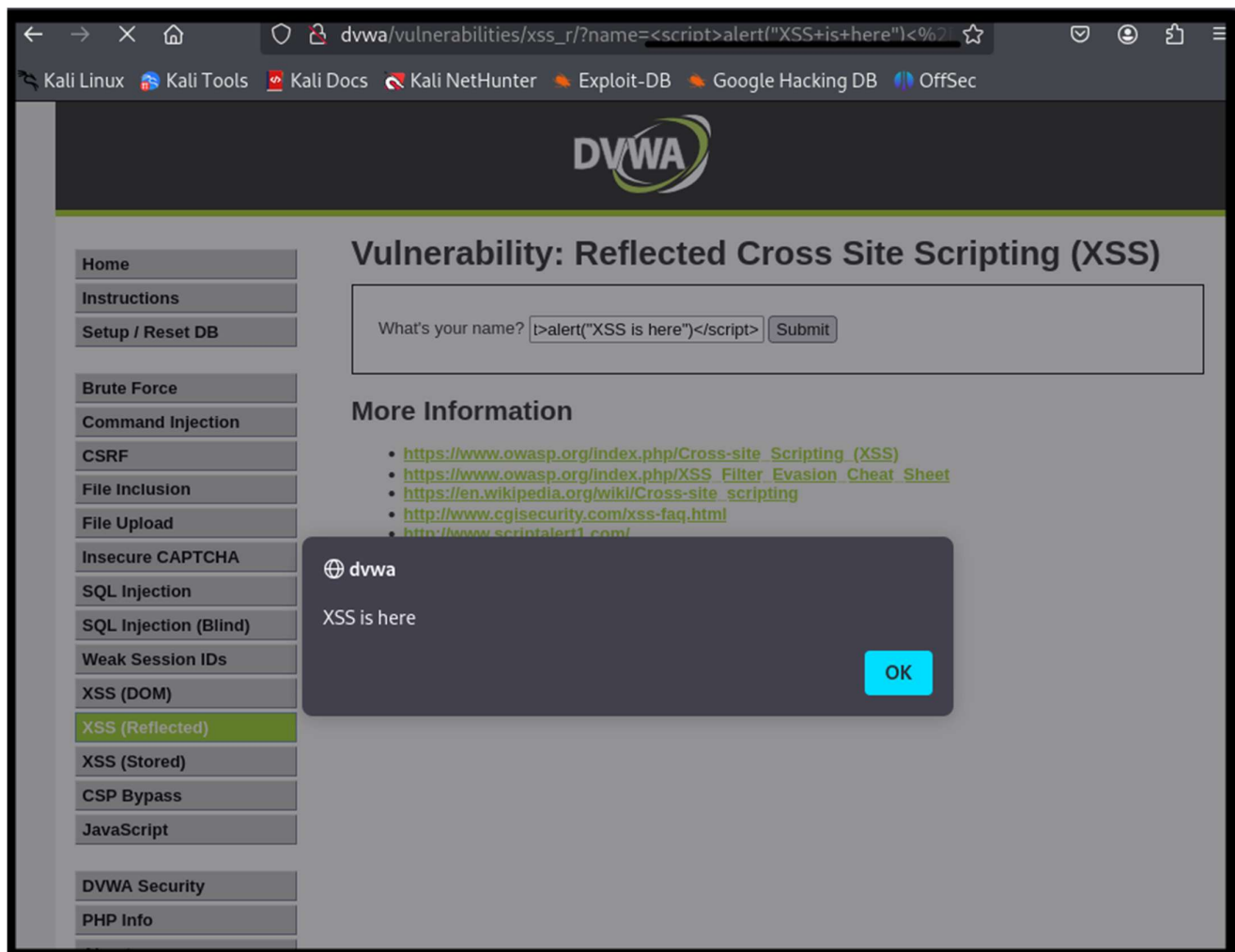
Reflected XSS Source
vulnerabilities/xss_r/source/low.php

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

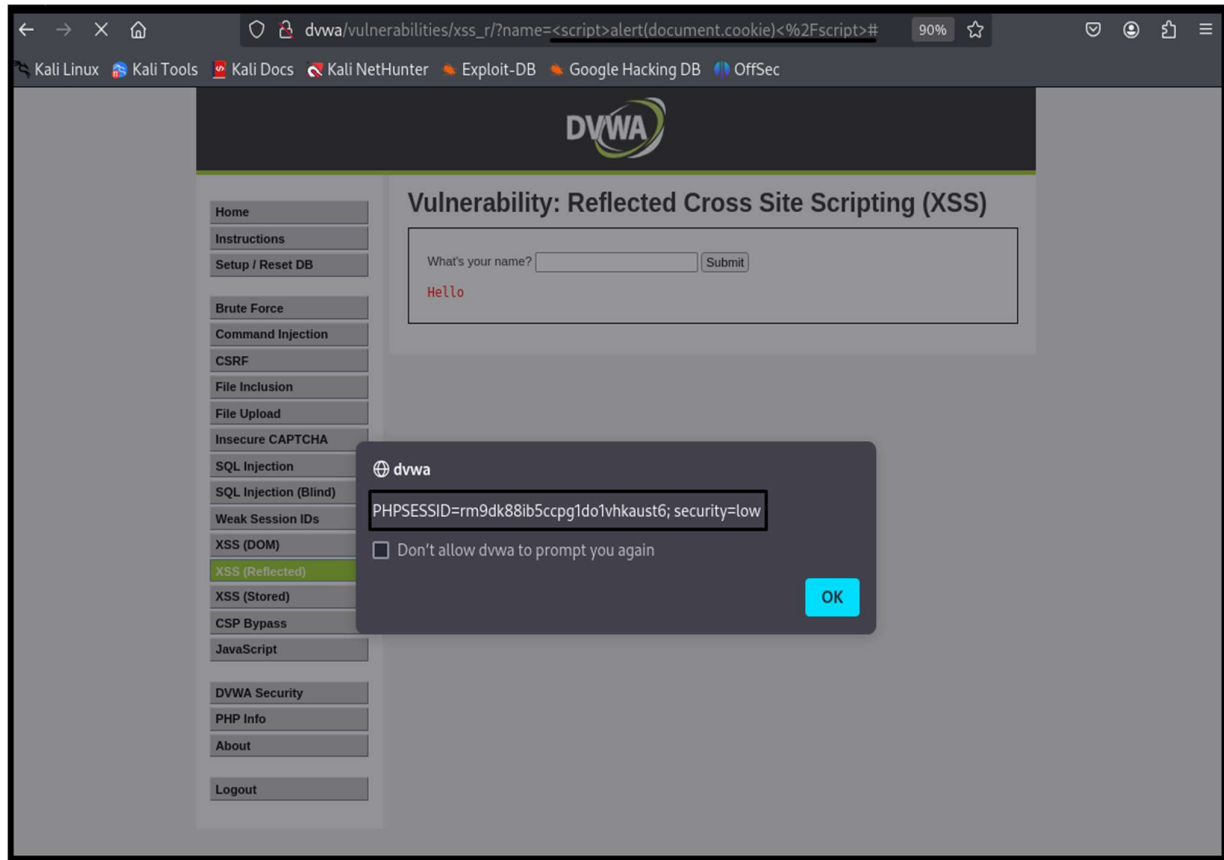
// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Feedback for end user
    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}

?>
```

Step: -3 In this Step I am applying a javascript payloads `<script>alert("XSS is here")</script>`



Step: -2 In this step, I am finding the cookie, and the payload is `<script>alert(document.cookie)</script>`



SECURITY LEVEL (MEDIUM)

As we know we will first view the source code.

Here, they are replacing the `<script>` tag with an empty value. We can easily bypass this by using the `` tag. Since the filter only checks for the lowercase `<script>` tag, it can also be bypassed by using a mixed-case variation like `<scRipT>`.

Reflected XSS Source

vulnerabilities/xss_r/source/medium.php

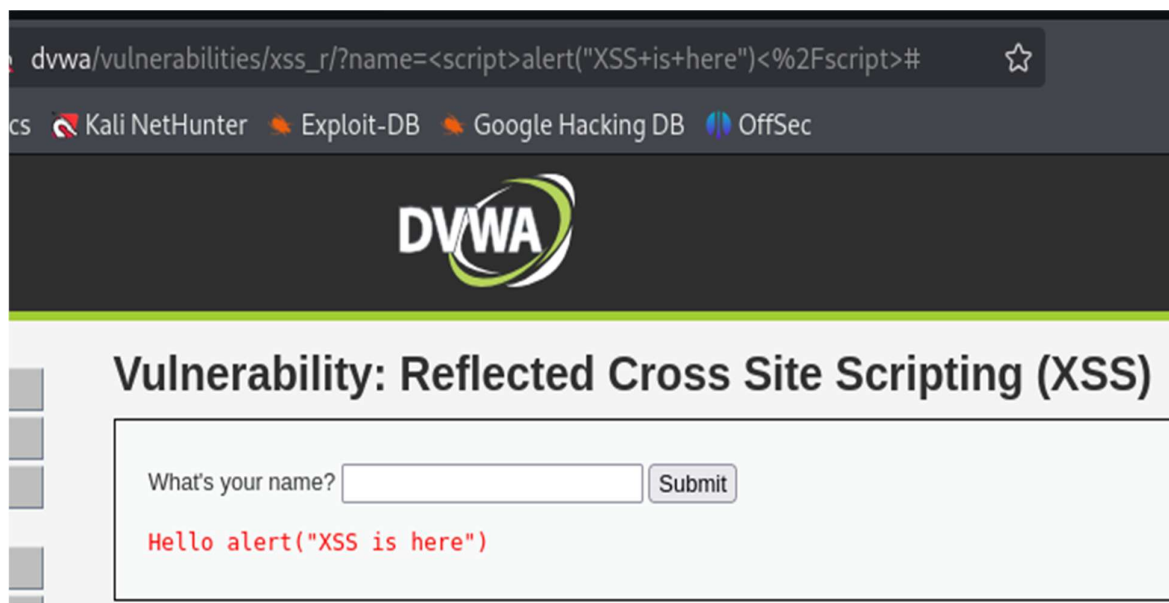
```
<?php
header ("X-XSS-Protection: 0");

// 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>";
}

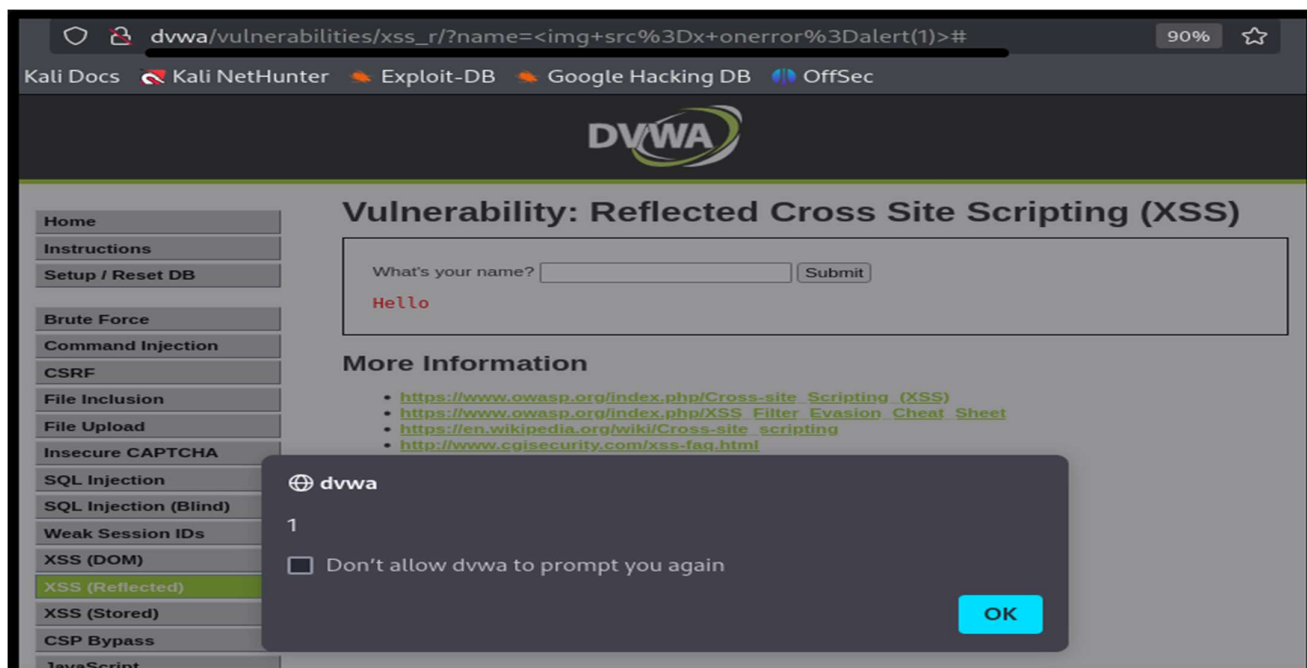
?>
```

first try can be just by inserting the same input as the previous step and seeing the result.

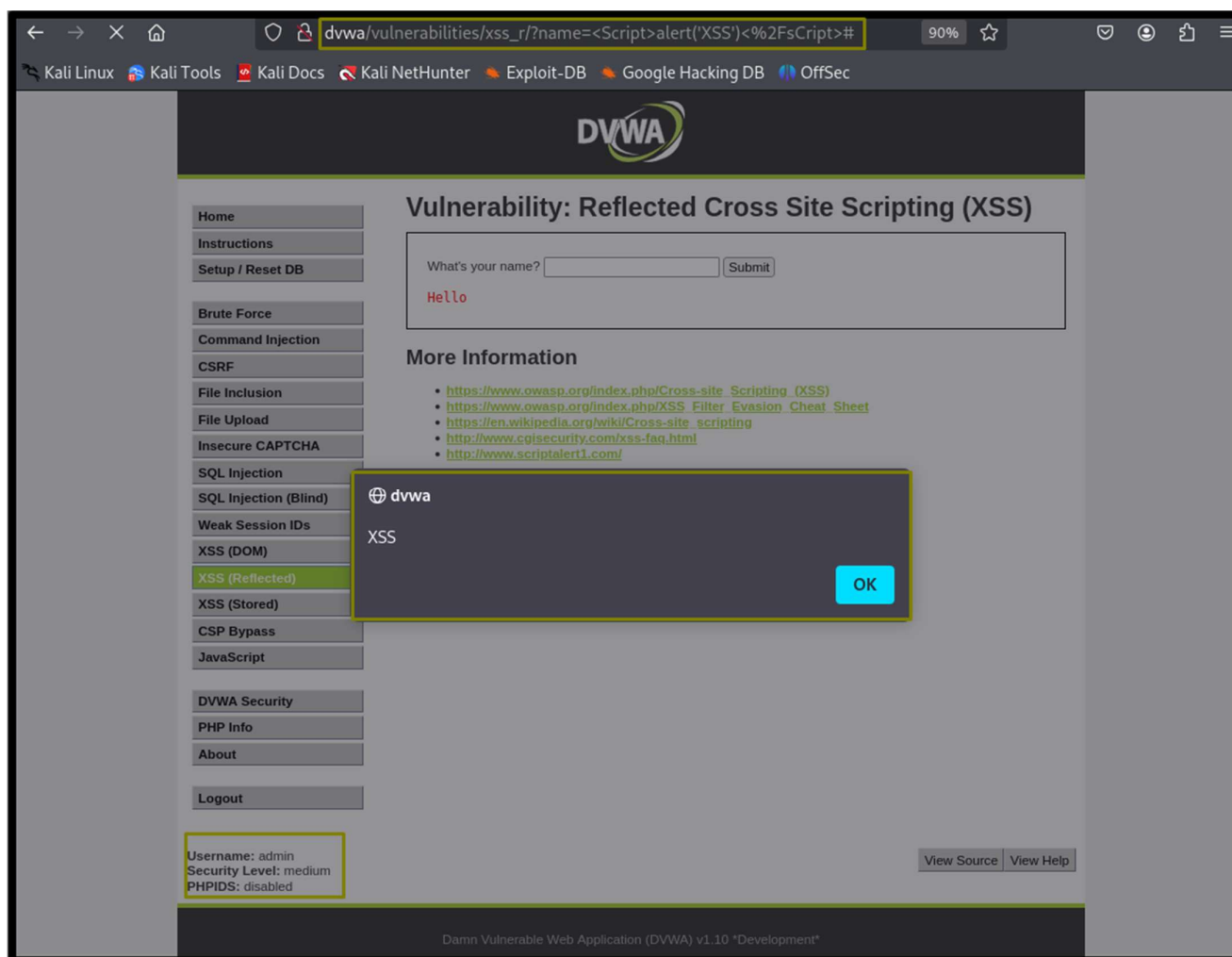


The screenshot shows a web browser window with the URL `dvwa/vulnerabilities/xss_r/?name=<script>alert("XSS+is+here")<%2Fscript>#`. The browser's address bar also shows search engines like Kali NetHunter, Exploit-DB, Google Hacking DB, and OffSec. The DVWA logo is visible. The page title is "Vulnerability: Reflected Cross Site Scripting (XSS)". Below the title, there is a form with the label "What's your name?" and a "Submit" button. The output of the script is displayed in red text: `Hello alert("XSS is here")`.

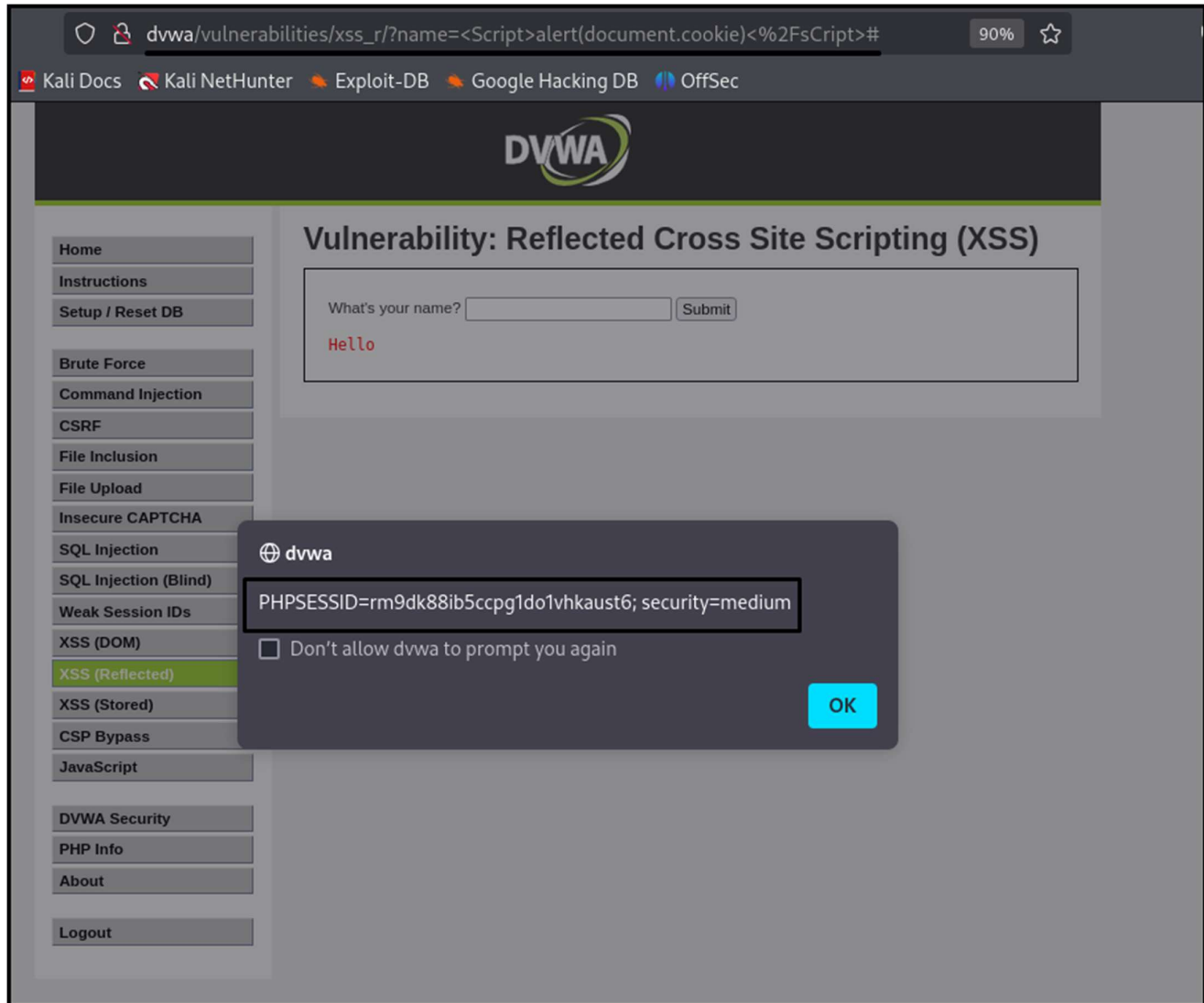
Step: -1 In this Step I am applying a javascript payloads



The next payloads apply: - <Script>alert('XSS')</script>



Step: - 2 In this step, I am finding the cookie, and the payload is
<Script>alert(document.cookie)</sCript



SECURITY LEVEL(HIGH)

As we know we will first view the source code.

```
Reflected XSS Source
vulnerabilities/xss_r/source/high.php

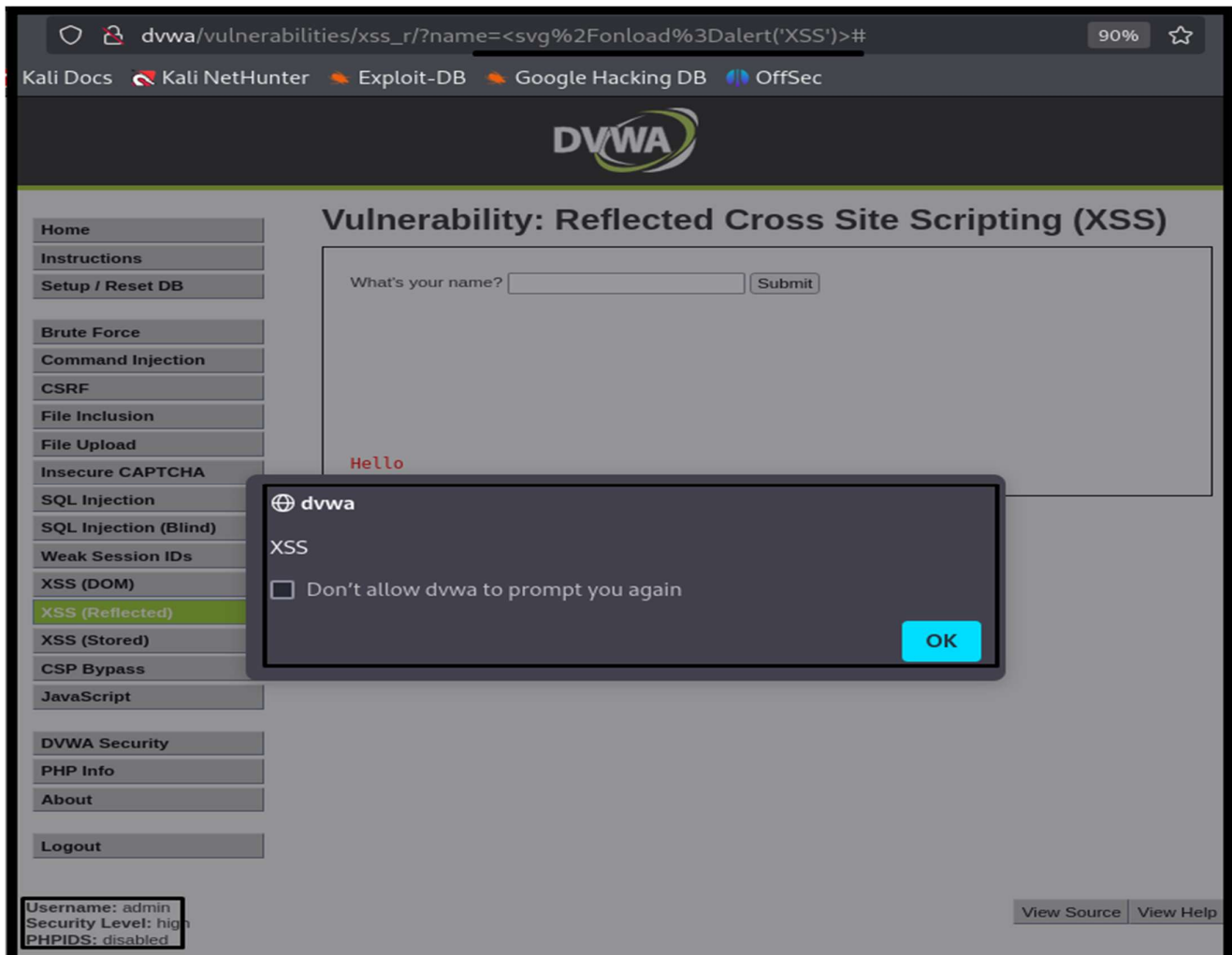
<?php
header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = preg_replace( '/<(.*?)s(.*?)c(.*?)r(.*?)i(.*?)p(.*?)t/1', '', $_GET[ 'name' ] );

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

?>
```

Step: -1 In this Step I am applying a javascript payloads <svg/onload alert('XSS')>



Step: -2 In this step, I am finding the cookie, and the payload is:

`<svg/onload=alert(document.cookie)>`

