Vulnerability Name:	Cross Site Request Forgery
Affected Vendor:	DVWA
Affected Product Name:	http://dvwa/vulnerabilities/csrf/
Product Official Website URL	http://dvwa/login.php
Affected Components:	Affected Parameters: - change your admin password

Description: - A vulnerability that allows attackers to trick authenticated users into executing unintended actions on web applications. A CSRF attack occurs when a malicious web site, email, blog, instant message, or program tricks an authenticated user's web browser into performing an unwanted action on a trusted site. If a target user is authenticated to the site, unprotected target sites cannot distinguish between legitimate authorized requests and forged authenticated request

Root Cause: -Lack of CSRF tokens or inadequate validation of requests.

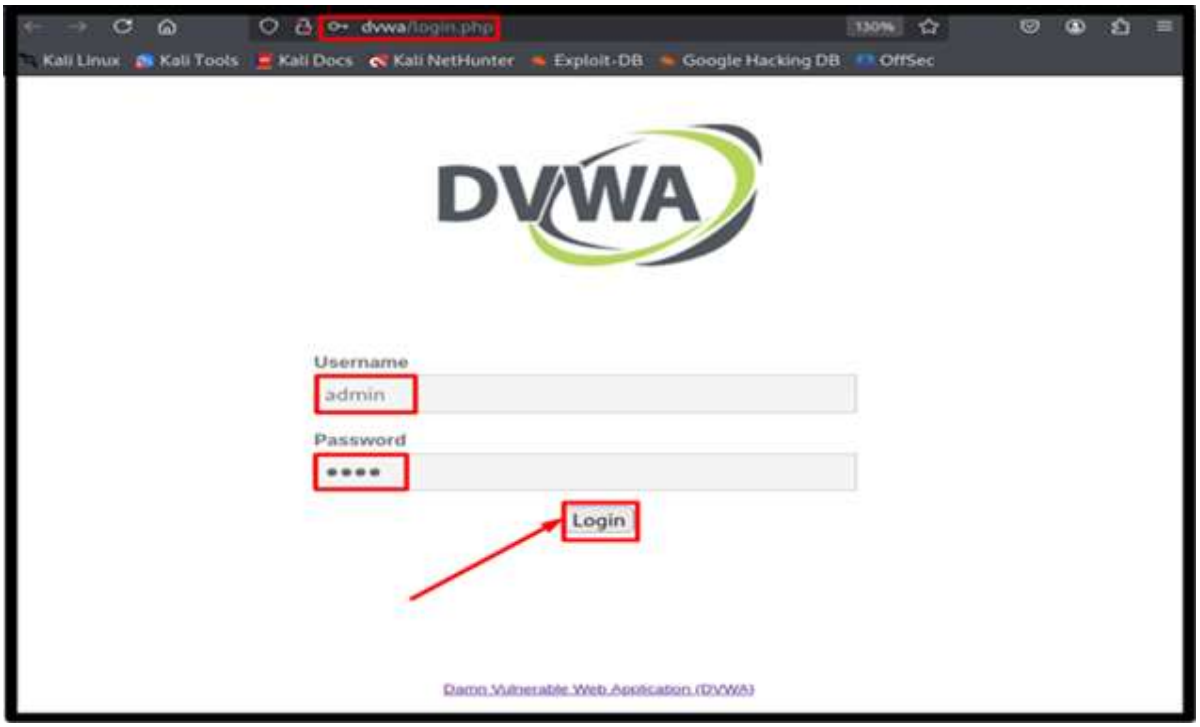
Impact: - Unauthorized actions performed by authenticated users without their consent, such as fund transfers or account deletions.

Mitigation: - Implement CSRF tokens, validate and compare request tokens, use anti-CSRF tokens in forms, and employ same-site cookie attributes to mitigate to CSFR.

Remediation: - To remediate a Cross-site Request Forgery Use anti-CSRF tokens in forms and API requests. or set the Same Site attribute to Strict or Lax in cookies. or require re-authentication or multi-factor authentication for sensitive actions. or verify Rerefer and Origin headers to ensure trusted sources. and Prefer POST, PUT, or DELETE over GET for state-changing operations. and require custom headers (e.g., X-Requested-With) in AJAX requests. or implement strict CORS policies to block unauthorized cross-origin requests. or use a separate CSRF token for logout functionality. or set Http Only and Secure flags to prevent unauthorized cookie access. and continuously test and validate CSRF protections in applications.

Proof of Concept

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



Security Level :- Low

As we Know, we will first view the source code.

CSRF Source

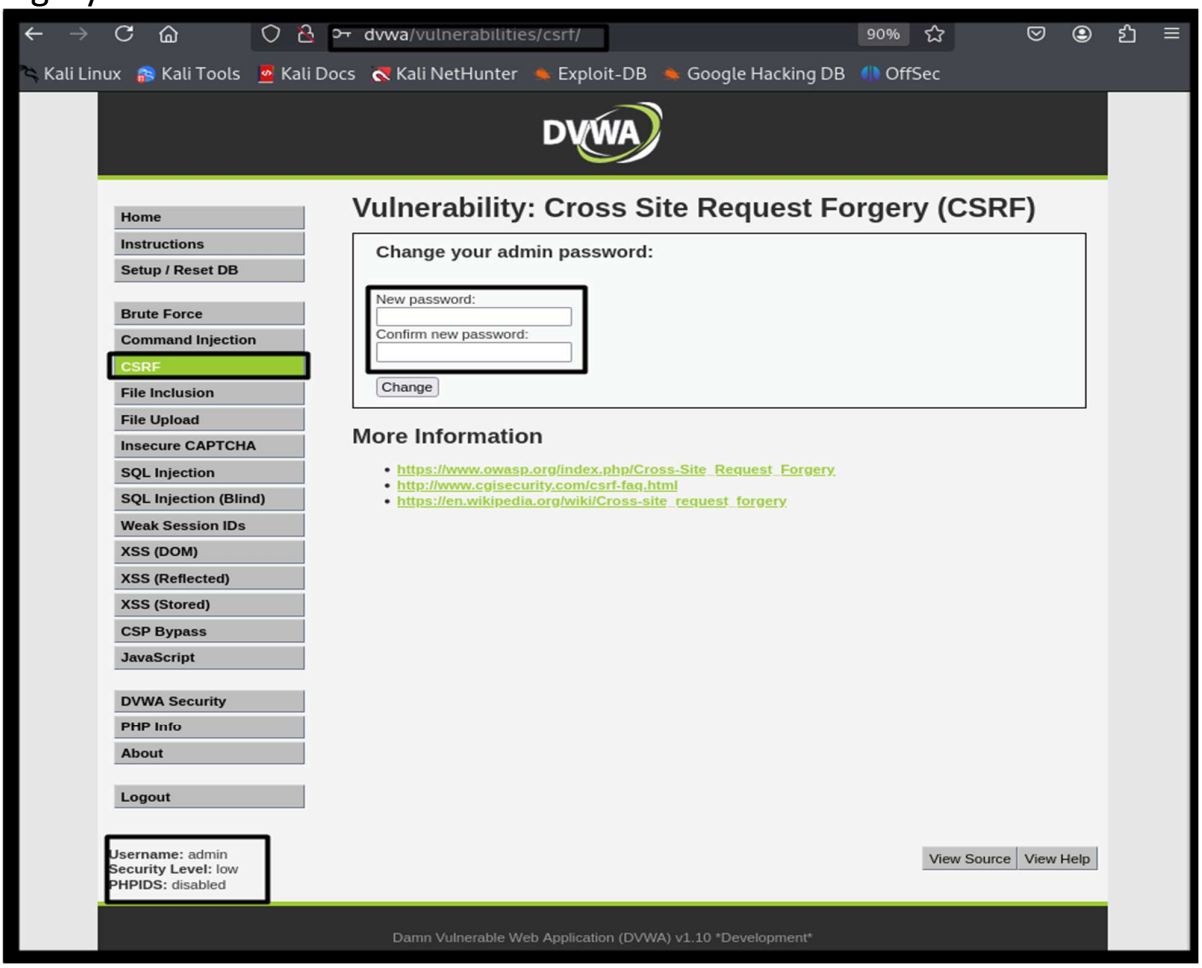
vulnerabilities/csrf/source/low.php

```php
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Get input
    $pass_new  = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // They do!
        $pass_new = (((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $pass_new ) : ((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
        $pass_new = md5( $pass_new );

        // Update the database
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
        $result = mysqli_query($GLOBALS["___mysqli_ston"],  $insert ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) . '</pre>' );

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

?>
```

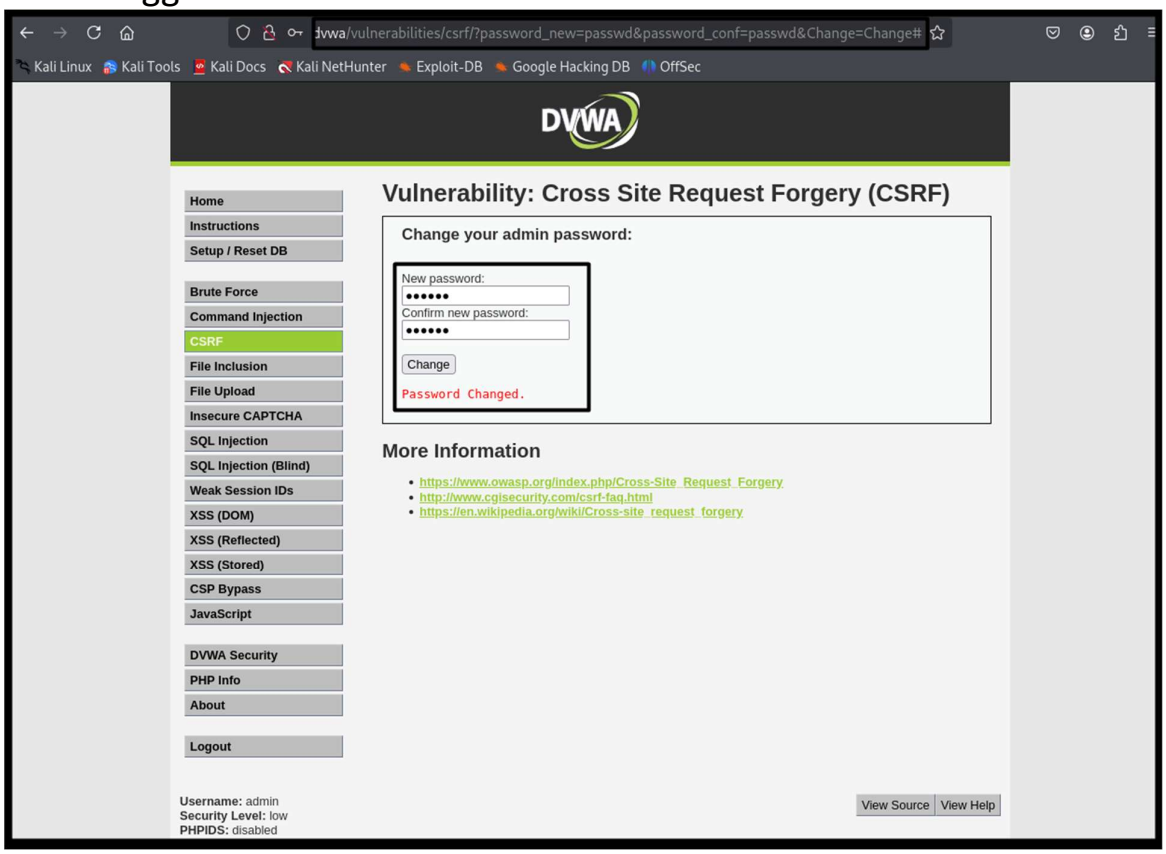
This code is vulnerable due to the lack of proper CSRF protection, allowing an attacker to craft a malicious URL and trick a logged-in user into unknowingly executing unintended actions.

The issue stems from the absence of request origin verification. As a result, an attacker can generate a URL containing the necessary parameters (password_new and password_conf) and send it to a victim. If the victim clicks on the malicious link while authenticated on the vulnerable website, the password change will occur without any additional authentication or user approval.

Step: -2 log in the home page of DVWA then click to the Cross-site Request Forgery Section.



Step:-3 In Step I, I changed the password, and you can see in the URL that it lacks the necessary CSRF token. Due to the absence of CSRF protection, an attacker can exploit this vulnerability by tricking the victim into clicking on the URL while logged into the vulnerable website.



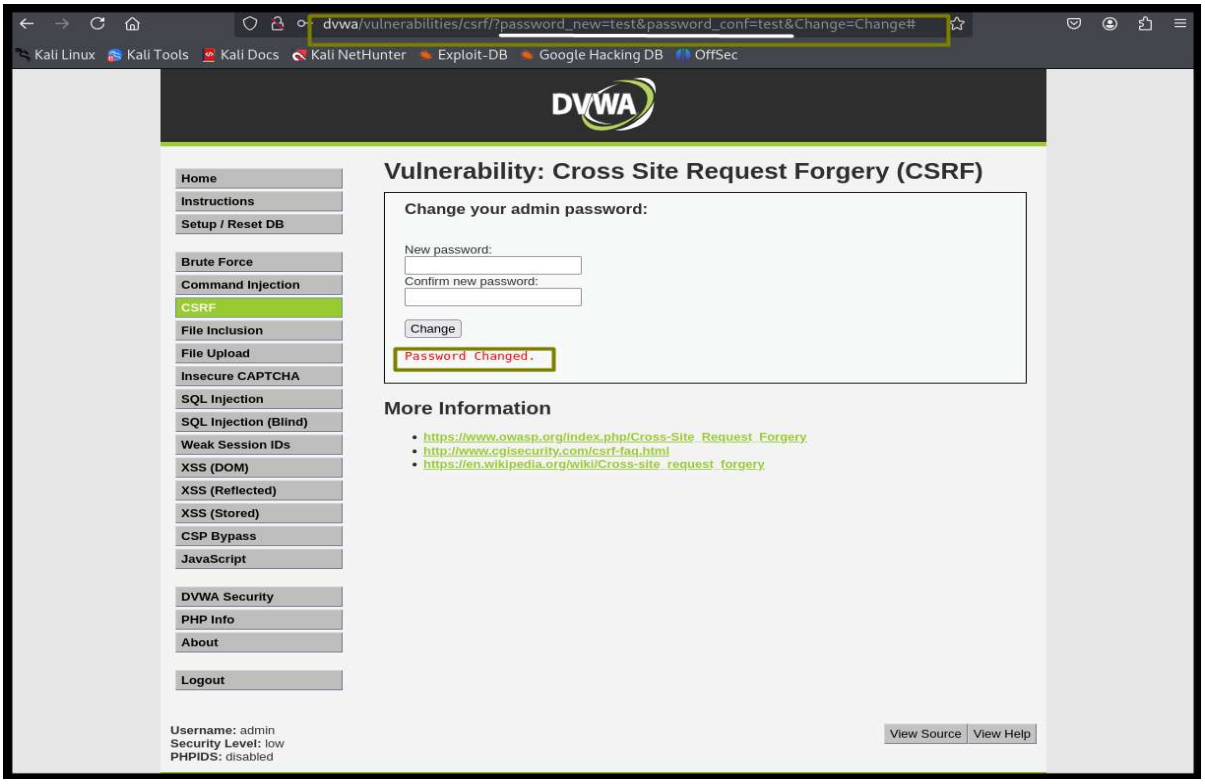
Step: -4 In this step Now we will Display The HTML code for the page, and password has changed by attacker . If attacker send link to victim, the password will be changed.

```
<html>
    <body>
    <script>
    document.location="http://dvwa/vulnerabilities/csrf/?password_new=test&password_conf=test&Change=Change#";
    </script>
    </body>
</html>
```

Step:-5 If the victim tries to open the html page. It will looks like this…

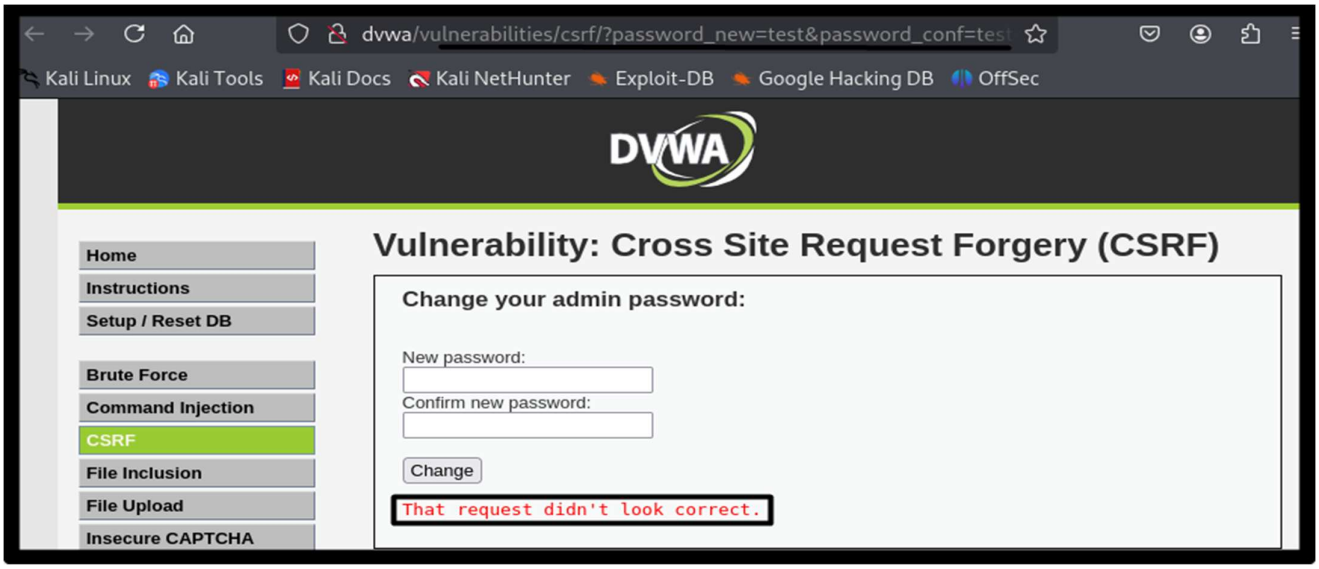
The password “test” will be changed automatically

Step:-6 In this Step We can see that password has changed.



SECURITY LEVEL: - MEDIUM

If we attempt to use the low-security method, it will no longer work.



As we Know, we will first view the source code.

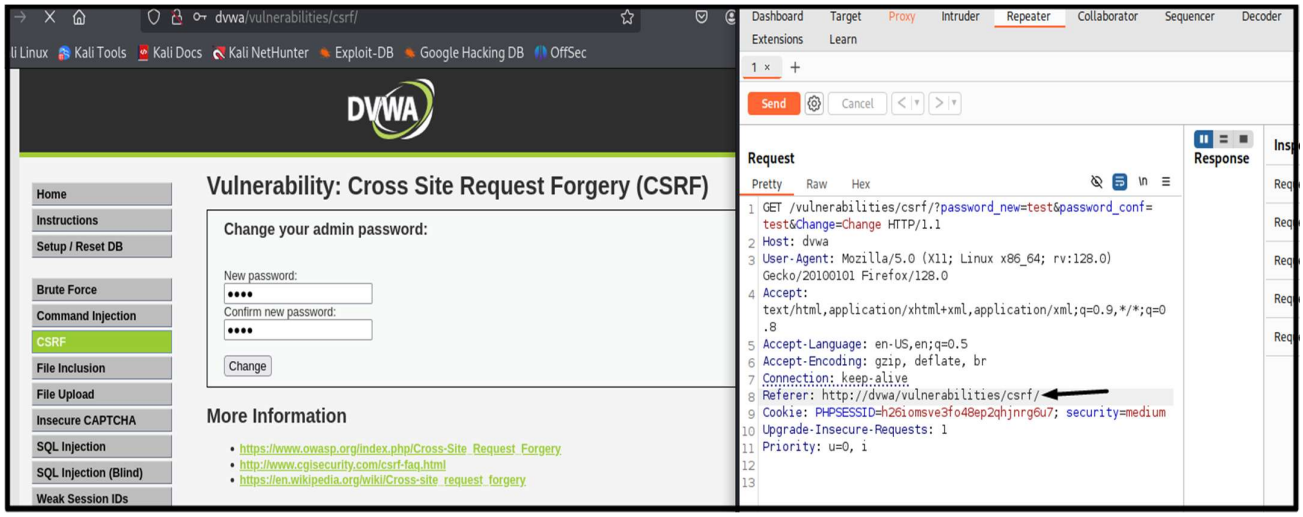
CSRF Source

/vulnerabilities/csrf/source/medium.php

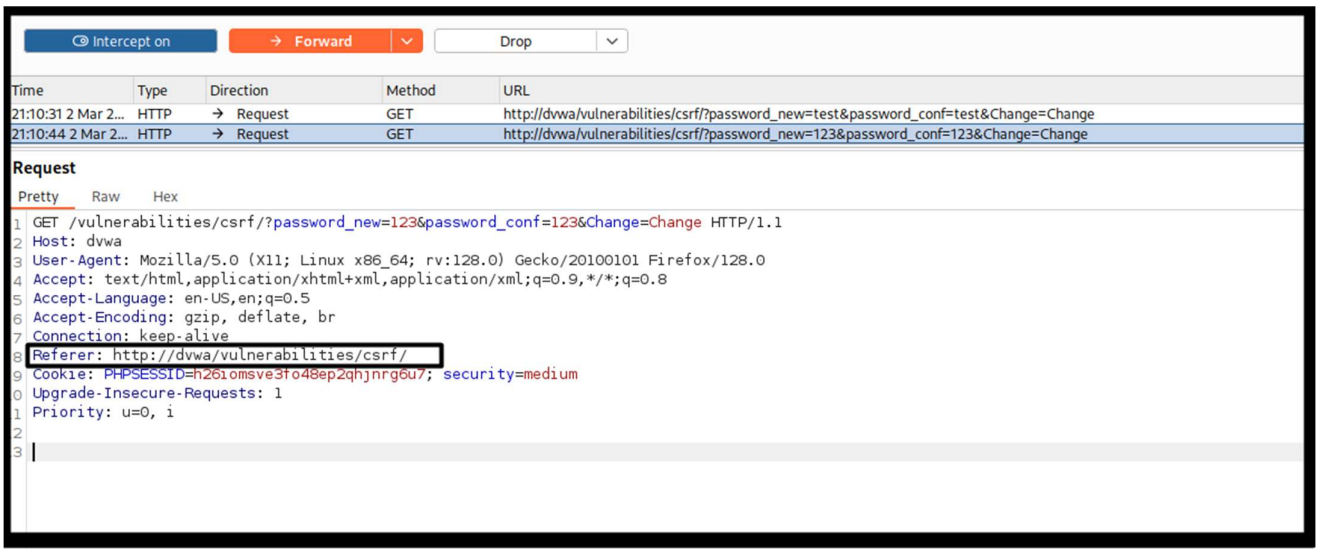
```php
<?php

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

        // Do the passwords match?
        if( $pass_new == $pass_conf ) {
            // They do!
            $pass_new = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $pass_new ) : ((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""))
            $pass_new = md5( $pass_new );

            // Update the database
            $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
            $result = mysqli_query($GLOBALS["___mysqli_ston"],  $insert ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) . '</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>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

?>
```

The flaw in this code is a cross-Site Request Forgery (CSRF) vulnerability. The code uses the HTTP Refer header to check if the request came from the same server, assuming it’s a trusted source. However, the Referrer header can be easily manipulated by an attacker. This allows an attacker to create a malicious website or craft or URL that makes a request to this script, tricking the user’s browser into performing an unwanted action o their behalf, such as changing their password without their knowledge or consent.

Step:-1 Can you see the difference? Within the legitimate request we see there is a Referer, where the request came from. That matches up so the request goes ahead.

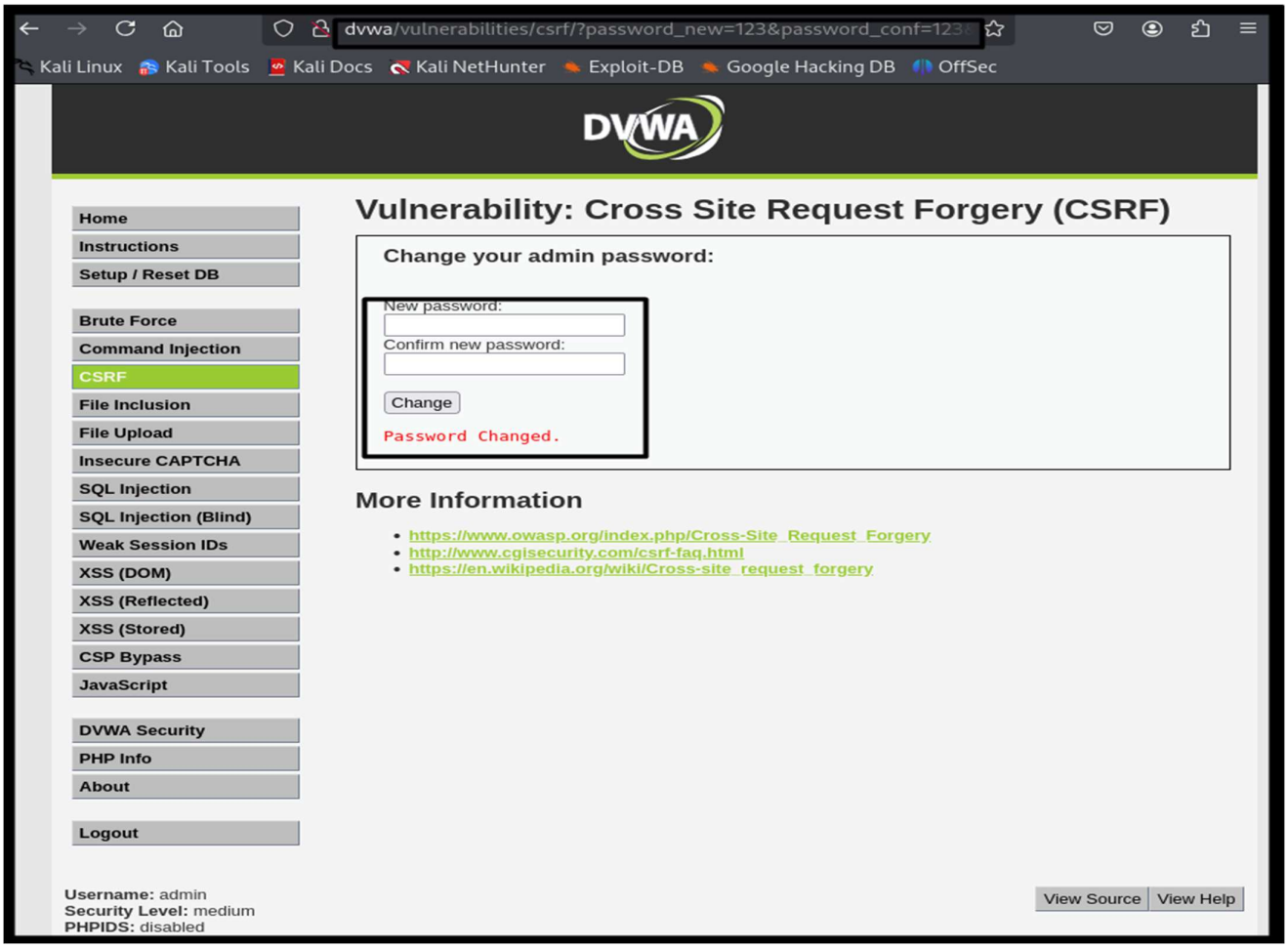


Step:-2 So what if we intercept the illegitimate request with Burp and add the HTTP Referer. Like so.



Step: -3 Password changed successfully

Now we will try to intercept the website and add legitimate Referrer using burp suite



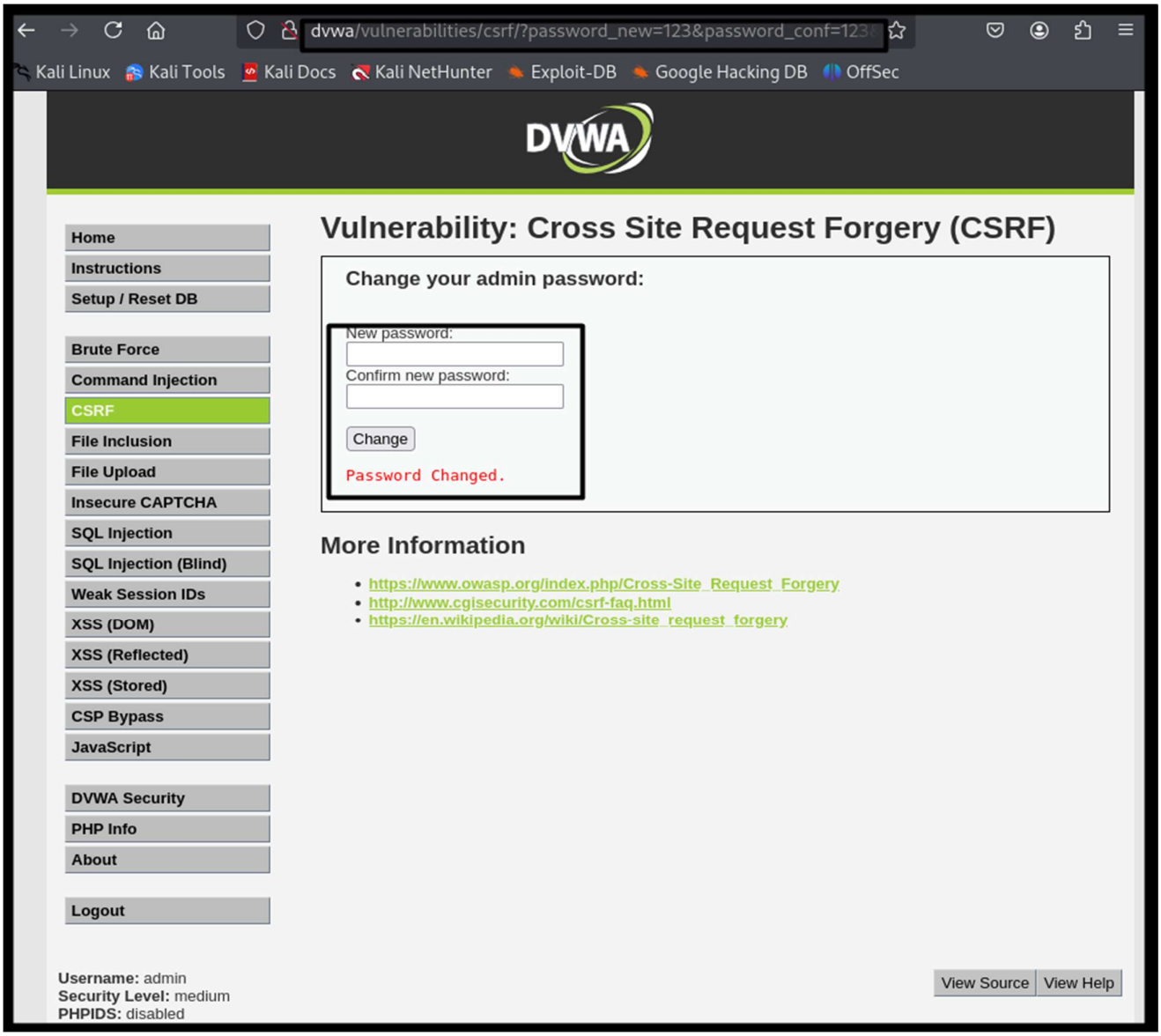
SECURITY LEVEL: -HIGH

CSRF Source

vulnerabilities/csrf/source/high.php

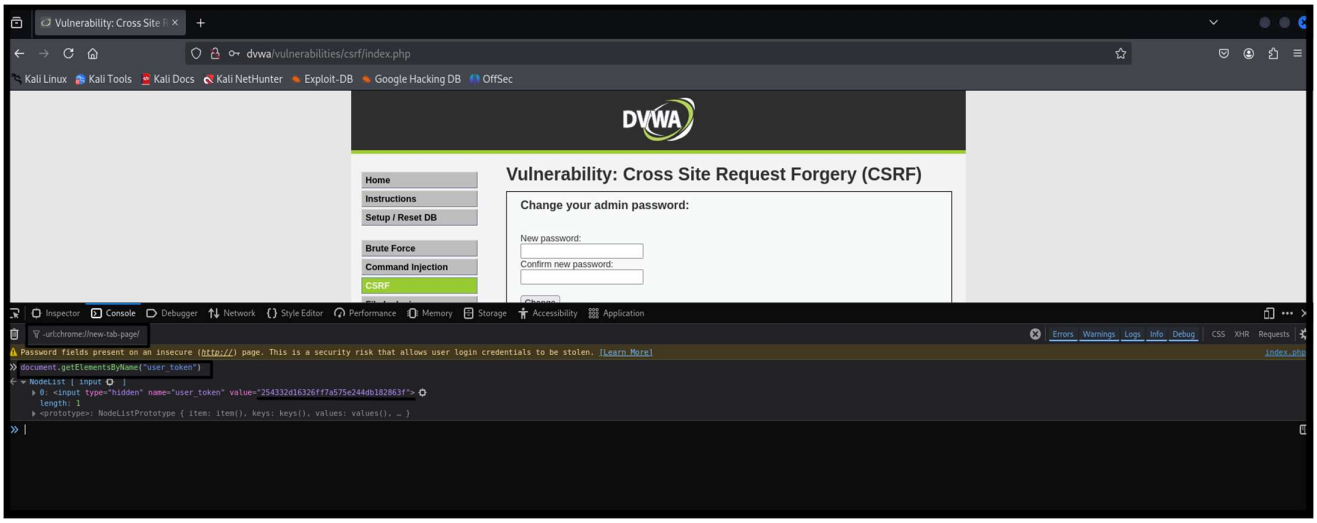
```php
<?php

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

    // Get input
    $pass_new  = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // They do!
        $pass_new = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $pass_new ) : ((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
        $pass_new = md5( $pass_new );

        // Update the database
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
        $result = mysqli_query($GLOBALS["___mysqli_ston"],  $insert ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) . '</pre>' );

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

Step:-1 For this level, we cannot use the previous method first try to understand the request with the help of burp you can notice it has a csrf token**.**So first, we will change the password to admin and after changing the password copy the URL



Step:-2 We can see the password has been changed now draft the URL with a different token and refresh the page after refreshing inspect the page go to console and type document.getElementsByName("user_token") and press enter we will get the output in the below way.

Expand the default value



Step:-3 Analysis the previous URL and create a new URL to add the new user token. Then url open in the browser.

Step:-4 In this Step Password changed successfully

Now we will try to intercept the website and add legitimate Referrer using burp suite