

Lab 19: Cross-Site Request Forgery

Explain what is CSRF?

CSRF, or Cross-Site Request Forgery, is a type of web attack where an attacker exploits the trust of a website in a user's browser to execute unauthorized actions on behalf of the user. This is done by tricking a user into clicking a link or visiting a website controlled by the attacker while logged into a targeted website. The attacker then sends a request from the user's browser to the targeted website, which is treated as a legitimate request from the user. This can result in the attacker changing the user's settings, accessing sensitive information, or even making purchases on the user's behalf.

LOW

The script takes the user input and checks if the two passwords match, if they do the password is updated, if not the password is not updated. What we can see here is there is no protection against CSRF, such as Anti-CSRF token.

CSRF Source

[vulnerabilities/csrf/source/low.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(
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? $pass_new :
        $pass_new = md5( $pass_new );

        // Update the database
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dwwaCurrentUser() . "'";
        $result = mysqli_query($GLOBALS["__mysqli_ston"], $insert ) or die( '<pre>' . ((is_object($
        // 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);
}

?>
```

What we can see is it is a GET request and you can see the value of the new password has been changed to 123. You can also see the session ID of the user in the cookies.

The screenshot shows a Kali Linux VM with two windows. The left window is the DVWA (Damn Vulnerable Web Application) interface, specifically the 'Vulnerability: Cross Site Request Forgery (CSRF)' page. It displays a form to 'Change your admin password:' with fields for 'New password:' and 'Confirm new password:', both containing '123'. The right window is Burp Suite Community Edition v2022.12.4. It shows an intercepted HTTP GET request to 'http://127.0.0.1:80/DVWA/vulnerabilities/csrf/?password_new=123&password_conf=123&Change=Change#'. The 'Inspector' tab on the right shows the request details, including the 'Cookie' field: 'PHPSESSID=ed79a795j1fabuu5v4fqng4; security=low'.

http://127.0.0.1/DVWA/vulnerabilities/csrf/?password_new=1234&password_conf=1234&Change=Change#

The screenshot shows a Kali Linux VM with two windows. The left window is the Burp Suite interface, displaying a landing page with the text 'Keep up with the latest vulnerabilities', 'WebSecurity Academy', and 'Upgrade to Burp Suite Professional'. The right window is Burp Suite Community Edition v2022.12.4. It shows an intercepted HTTP GET request to 'http://127.0.0.1:80/DVWA/vulnerabilities/csrf/?password_new=1234&password_conf=1234&Change=Change#'. The 'Inspector' tab on the right shows the request details, including the 'Cookie' field: 'PHPSESSID=ed79a795j1fabuu5v4fqng4; security=low'.

So just trick the user into clicking on the link above to be able to change the password to the password we want

MEDIUM

It is checking if the HTTP referer is in the server name and vice-versa. If yes the request goes ahead. A HTTP Referrer is a HTTP header field that identifies the address of the webpage (i.e. the URI or IRI) that linked to the resource being requested. By checking the referrer, the new webpage can see where the request originated.

CSRF Source

vulnerabilities/csrf/source/medium.php

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Checks to see where the request came from
    if( strpos( $_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"])
[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_ob

            // 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>";
}
```

When you send the request, you will get more information about the HTTP Referrer

The screenshot shows a Kali Linux virtual machine with two windows open. The left window displays the DVWA (Damn Vulnerable Web Application) interface, specifically the 'Vulnerability: Cross Site Request Forgery (CSRF)' page. The page has a sidebar with various attack categories like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF (highlighted), File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The main content area shows the 'Change your admin password:' form with fields for 'New password:', 'Confirm new password:', and a 'Change' button. Below the form, there are announcements, more information links, and a 'View Source' button. The right window shows Burp Suite Community Edition v2022.12.4 intercepting an HTTP request. The request is a GET request to 'http://127.0.0.1:80/DVWA/vulnerabilities/csrf/?password_new=1234&password_conf=1234&Change=Change#'. The 'Inspector' tab on the right shows the request details, including the 'Referer' header: 'http://127.0.0.1/DVWA/vulnerabilities/csrf/'.

http://127.0.0.1/DVWA/vulnerabilities/csrf/?password_new=1234&password_conf=1234&Change=Change#

So when before it is sent to the server, we just need to intercept the request and add the HTTP Referrer information and it has passed successfully.

The screenshot displays a Kali Linux virtual machine environment. On the left, a web application titled "Vulnerability: Cross Site Request Forgery (CSRF)" is open in a browser. The page shows a "Change your admin password:" form with fields for "New password:" and "Confirm new password:", a "Test Credentials" button, and a "Change" button. Below the form, a message states: "That request didn't look correct." The page also includes a sidebar with navigation links like "Home", "Instructions", "Setup / Reset DB", "Brute Force", "Command Injection", "CSRF", "File Inclusion", "File Upload", "Insecure CAPTCHA", "SQL Injection", "SQL Injection (Blind)", "Weak Session IDs", "XSS (DOM)", "XSS (Reflected)", "XSS (Stored)", "CSP Bypass", "JavaScript", "DVWA Security", "PHP Info", "About", and "Logout".

On the right, Burp Suite Community Edition v2022.12.4 is running. The "Intercept" tab is active, showing a request to `http://127.0.0.1:80`. The request details are visible in the "Inspector" pane, showing the "Request Headers" section. The "Referer" header is highlighted, showing its value: `http://127.0.0.1/DVWA/vulnerabilities/csrf/`. The "Request Body" section is also visible, showing the form data: `password_new=1234&password_conf=1234&Change=Change`.

HIGH

This part of the site is secured in such a way that each version will have its own token so first we need to get the victim's token

```

<?php

$change = false;
$request_type = "html";
$return_message = "Request Failed";

if ($_SERVER['REQUEST_METHOD'] == "POST" && array_key_exists ("CONTENT_TYPE", $_SERVER) && $_SERVER['CONTENT_TYPE'] == "application/json") {
    $data = json_decode(file_get_contents('php://input'), true);
    $request_type = "json";
    if (array_key_exists("HTTP_USER_TOKEN", $_SERVER) &&
        array_key_exists("password_new", $data) &&
        array_key_exists("password_conf", $data) &&
        array_key_exists("Change", $data)) {
        $token = $_SERVER['HTTP_USER_TOKEN'];
        $pass_new = $data["password_new"];
        $pass_conf = $data["password_conf"];
        $change = true;
    }
} else {
    if (array_key_exists("user_token", $_REQUEST) &&
        array_key_exists("password_new", $_REQUEST) &&
        array_key_exists("password_conf", $_REQUEST) &&
        array_key_exists("Change", $_REQUEST)) {
        $token = $_REQUEST["user_token"];
        $pass_new = $_REQUEST["password_new"];
        $pass_conf = $_REQUEST["password_conf"];
        $change = true;
    }
}

if ($change) {
    // Check Anti-CSRF token
    checkToken( $token, $_SESSION[ 'session_token' ], 'index.php' );

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // They do!
        $pass_new = mysqli_real_escape_string ($GLOBALS["__mysqli_ston"], $pass_new);
        $pass_new = md5( $pass_new );

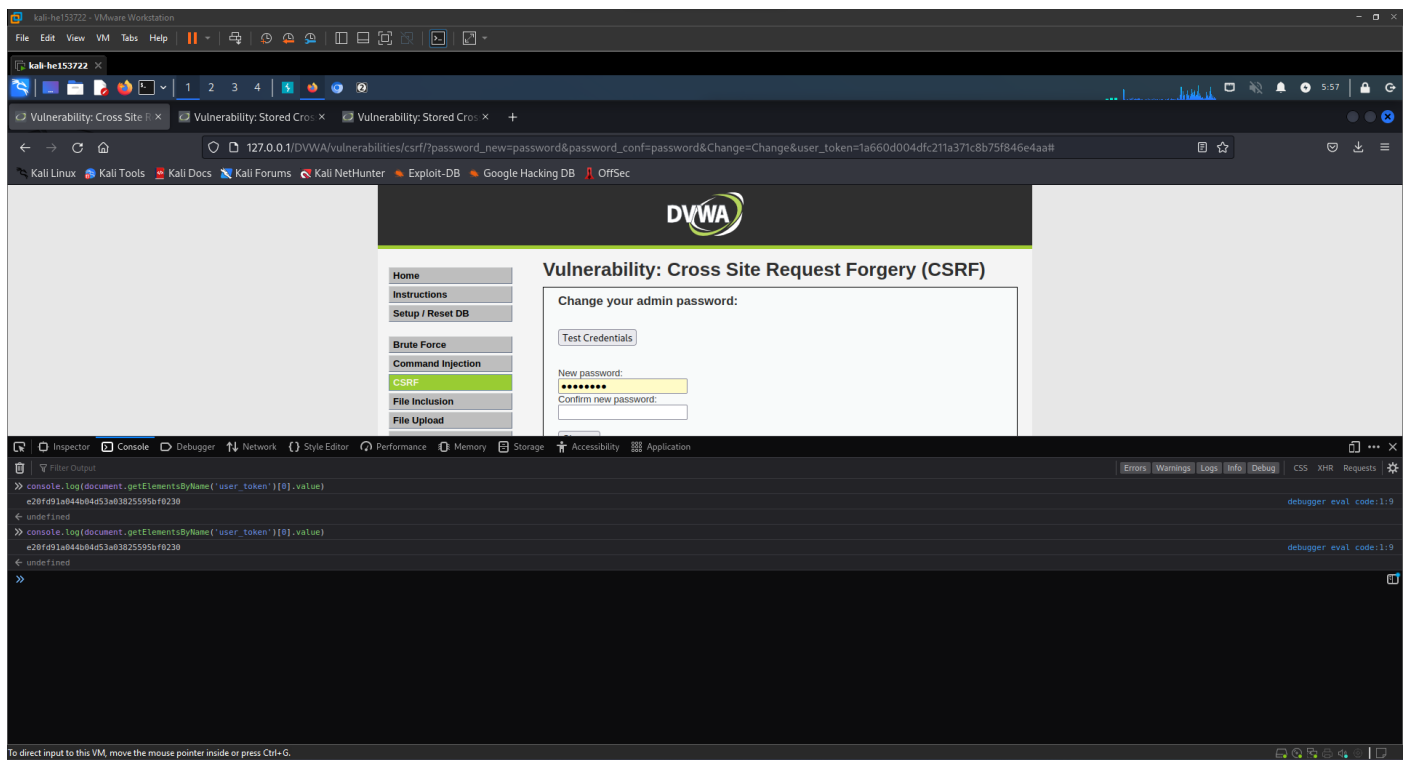
        // Update the database
        $insert = "UPDATE `users` SET password = '" . $pass_new . "' WHERE user = '" . dwwaCurrentUser . "'";
        $result = mysqli_query($GLOBALS["__mysqli_ston"], $insert );

        // Feedback for the user
        $return_message = "Password Changed.";
    }
    else {
        // Issue with passwords matching
        $return_message = "Passwords did not match.";
    }
}

mysqli_close($GLOBALS["__mysqli_ston"]);

```

To get token we write the following command in console: `console.log(document.getElementsByName("user_token")[0].value)`



http://127.0.0.1/DVWA/vulnerabilities/csrf/?password_new=password&password_conf=password&Change=Change&user_token=e20fd91a044b04d53a03825595bf0230#