CMPE 279 – Assignment 4

Name	Student ID
Mayur Barge	013722488
Varun Jain	013719108

Question

Describe the attack you used. How did it work?

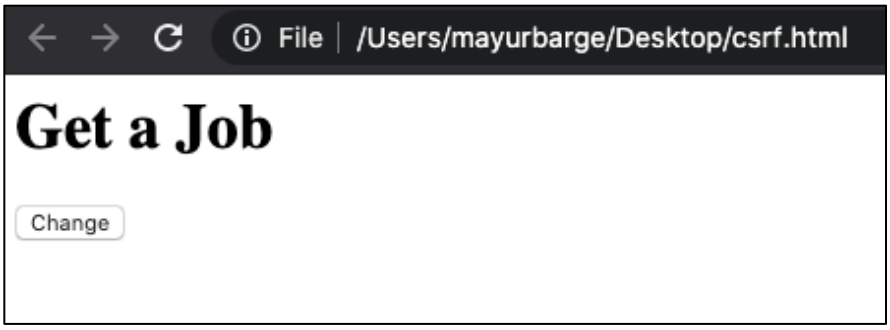
Answer

We are using CSRF (Cross Site Request Forgery) attack in this assignment. Initially we set security level at **low** for DVWA. We created a dummy page with form which will perform the HTTP GET request to original resource (http://10.0.0.220/vulnerabilities/csrf/). We hide the HTML input elements and hardcoded with our password (**cmpe279**)

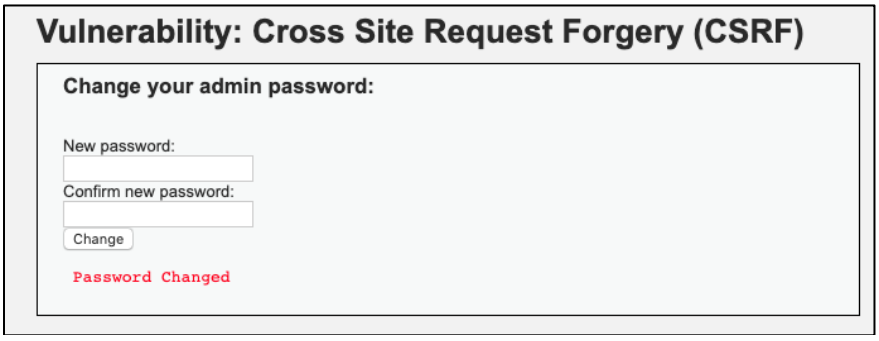
csrf.html

Users > mayurbarge > Desktop > csrf.html > form

```
1  <form action="http://10.0.0.220/vulnerabilities/csrf/?" method="GET">
2      <h1>Get a Job</h1>
3      <input type="hidden" AUTOCOMPLETE="off" name="password_new" value="cmpe279">
4      <input type="hidden" AUTOCOMPLETE="off" name="password_conf" value="cmpe279">
5      <input type="submit" value="Change" name="Change">
6  </form>
```



Once the user clicks on the **Change** button, password will change.



Question

Does your attack work in “Medium” security level?

Answer

When we set security level to **Medium** it did not work. The selected line in the code below checks for the **HTTP_REFERER** which tells you about the origin of the request.

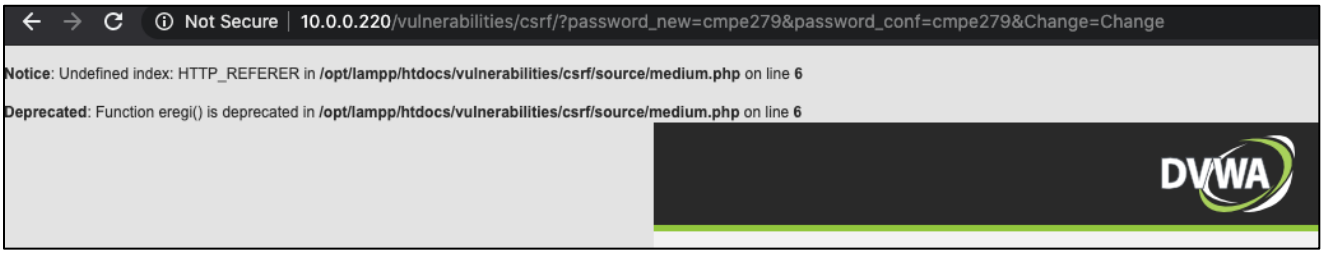
```php
<?php

    if (isset($_GET['Change'])) {

        // Checks the http referer header
        if ( eregi ( "127.0.0.1", $_SERVER['HTTP_REFERER'] ) ){

            // Turn requests into variables
            $pass_new = $_GET['password_new'];
            $pass_conf = $_GET['password_conf'];
```

Our source was different than that of DVWA application hence the attack failed.



Question

Set the security mode to “Low” and examine the code that is vulnerable, and then set the security mode to “High” and reexamine the same code. What changed? How do the changes prevent the attack from succeeding?

Answer

When we change security level to **High,** we observed that the application asks for current password also. This is an additional security check that we observed.

In the code below, it first validates the current password and then only allows to change the current password. Attacker can have access to the vulnerable page but not to an old password.

High CSRF Source

```
<?php

    if (isset($_GET['Change'])) {

        // Turn requests into variables
        $pass_curr = $_GET['password_current'];
        $pass_new = $_GET['password_new'];
        $pass_conf = $_GET['password_conf'];

        // Sanitise current password input
        $pass_curr = stripslashes( $pass_curr );
        $pass_curr = mysql_real_escape_string( $pass_curr );
        $pass_curr = md5( $pass_curr );

        // Check that the current password is correct
        $qry = "SELECT password FROM `users` WHERE user='admin' AND password='$pass_curr';";
        $result = mysql_query($qry) or die('<pre>' . mysql_error() . '</pre>' );

        if (($pass_new == $pass_conf) && ( $result && mysql_num_rows( $result ) == 1 )){
            $pass_new = mysql_real_escape_string($pass_new);
            $pass_new = md5($pass_new);

            $insert="UPDATE `users` SET password = '$pass_new' WHERE user = 'admin';";
            $result=mysql_query($insert) or die('<pre>' . mysql_error() . '</pre>' );

            echo "<pre> Password Changed </pre>";
            mysql_close();
        }

        else{
            echo "<pre> Passwords did not match or current password incorrect. </pre>";
        }

    }
?>
```