

Name: Samarth Jain

USN: 4SU20CS081

Course: Cybersecurity

Trainer: Bharath Kumar

Date: 08/09/2023

Assignment Details

Assigned Date: 07/09/2023

Due Date: 08/09/2023

Topic: Multi-Point Fuzzing

Introduction

Fuzzing is a software testing technique used to uncover vulnerabilities and weaknesses in computer programs, applications, or systems. It involves the automated generation and submission of a large volume of unexpected, invalid, or random data as input to the target software. The primary goal of fuzzing is to identify security flaws, crashes, and unexpected behaviour that may be indicative of vulnerabilities that can be exploited by attackers.

During the fuzzing process, a fuzzing tool or framework generates various inputs, such as malformed data packets, unexpected command sequences, or random values, and sends them to the target software's input points, such as user interfaces, APIs, network protocols, or file parsers. The fuzzing tool monitors the target for any signs of abnormal behaviour, including crashes, hangs, excessive resource consumption, or error messages.

Fuzzing is particularly effective for uncovering memory-related vulnerabilities like buffer overflows, as well as input validation issues and boundary conditions that can lead to security vulnerabilities. It is an essential part of the security testing process, helping organizations identify and mitigate potential security risks in their software before they can be exploited by malicious actors. Fuzzing can be applied at different stages of the software development lifecycle, from early development to post-deployment security assessments, making it a valuable tool in ensuring the robustness and security of software systems.

Content

Unzip the zip file of Burpsuite pro. Extract the file onto a preferred location, in this case it is Desktop/Burp/

Command: unzip burpsuite_pro_v2023.2.2.zip

Open the extracted folder in the Terminal.

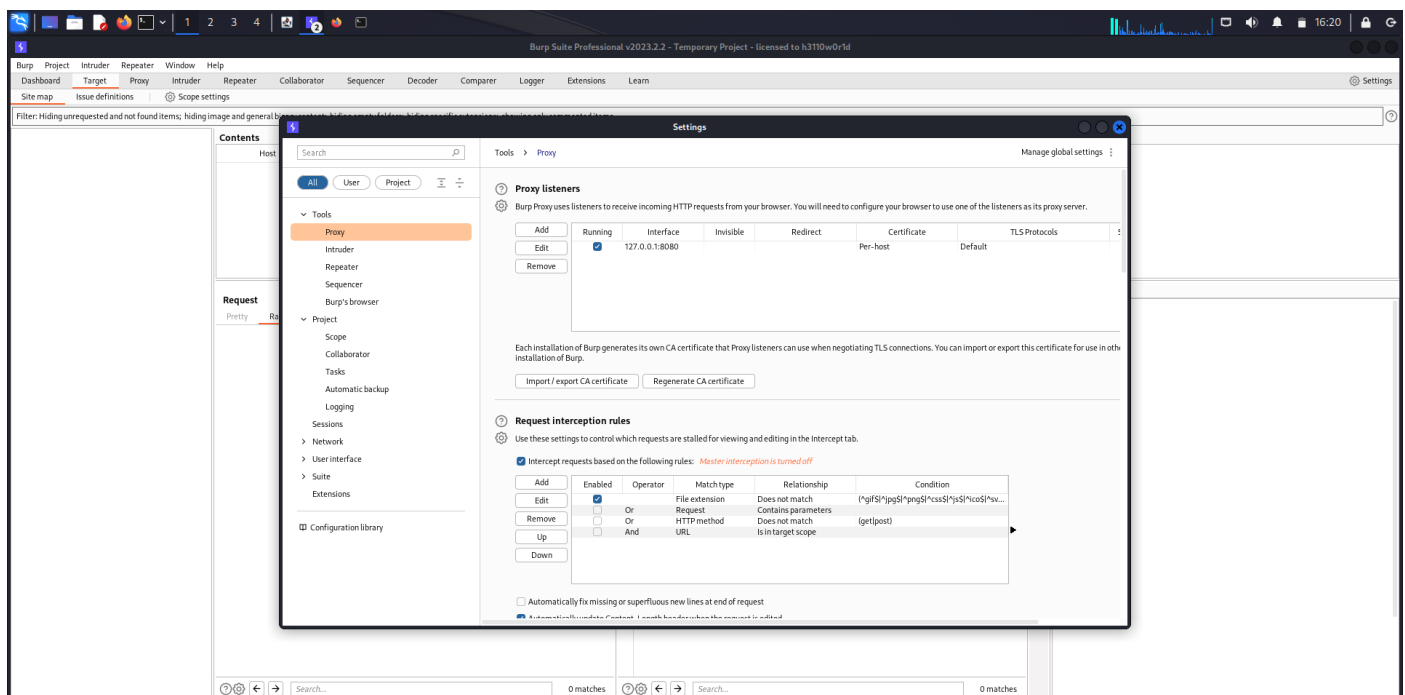
Install the Burpsuite Pro

Command: java -jar Loader.java

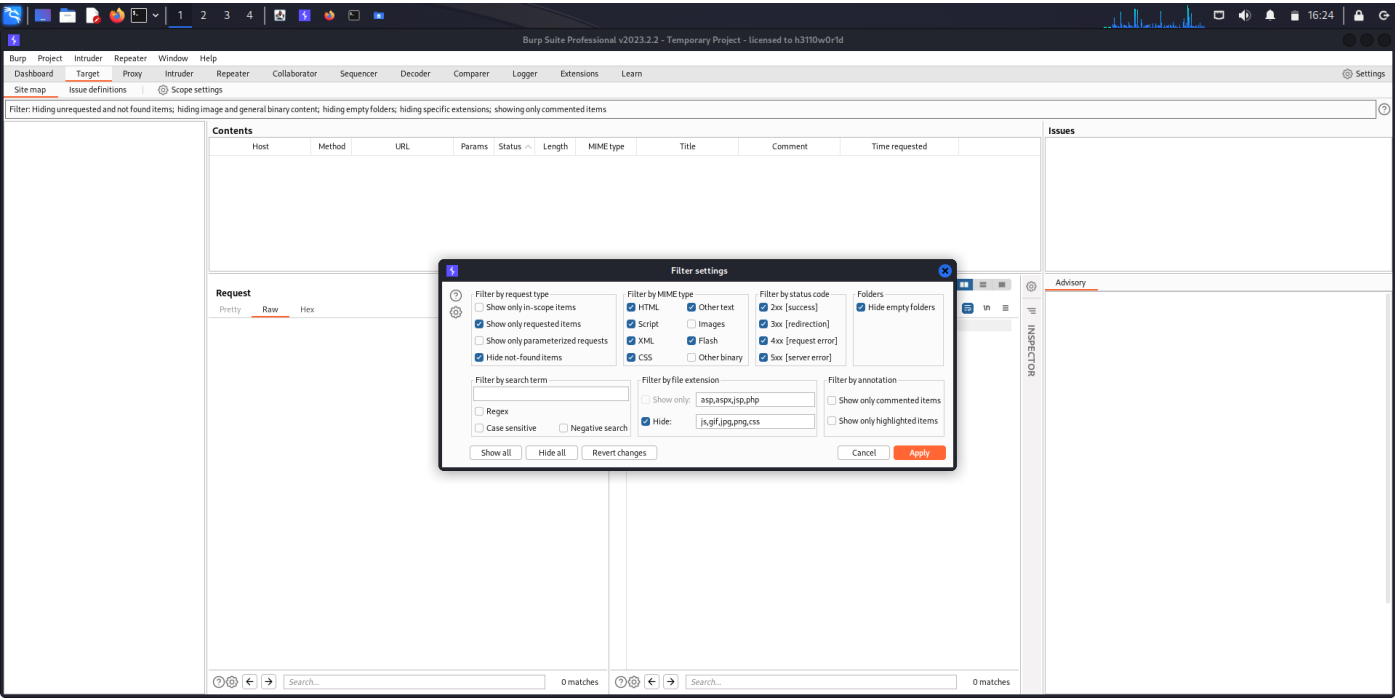


Refer the README.txt in the extracted folder for Installation(Loader) configuration.

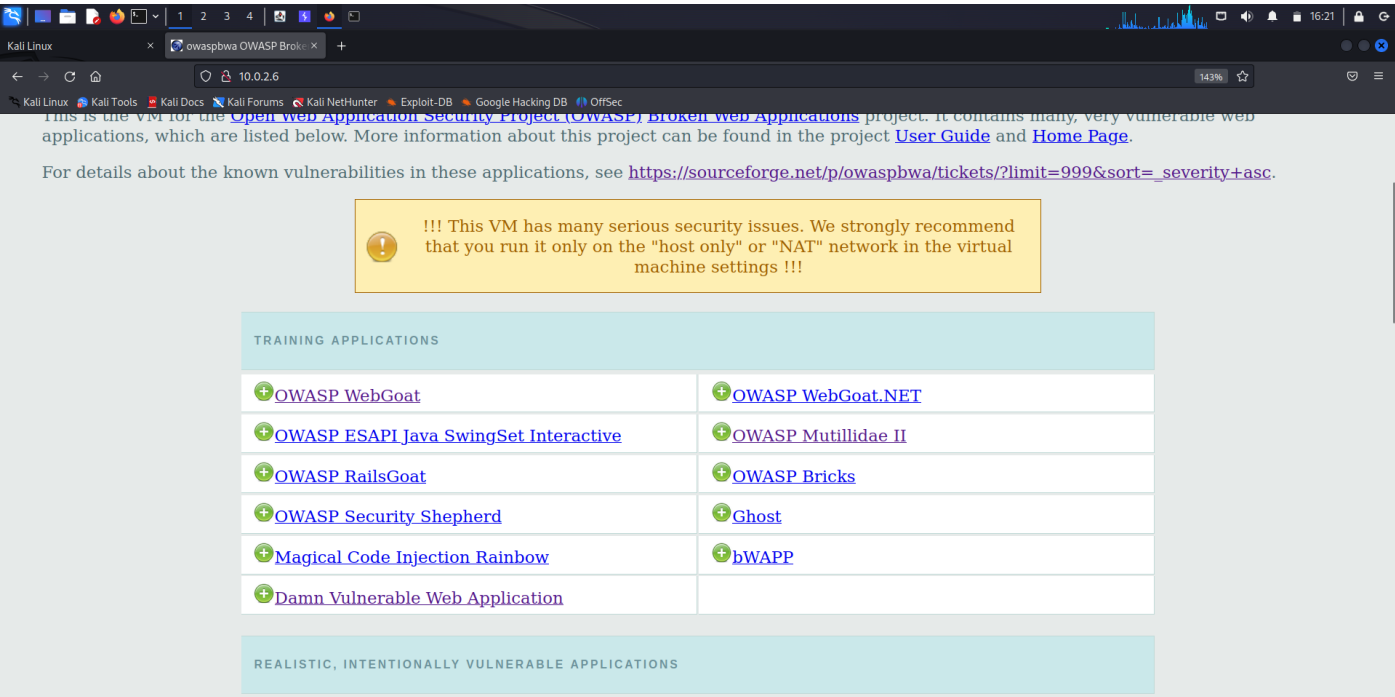
Start the Burp and ensure it is running on localhost and port 8080.



Filter the Target according to your preferences.

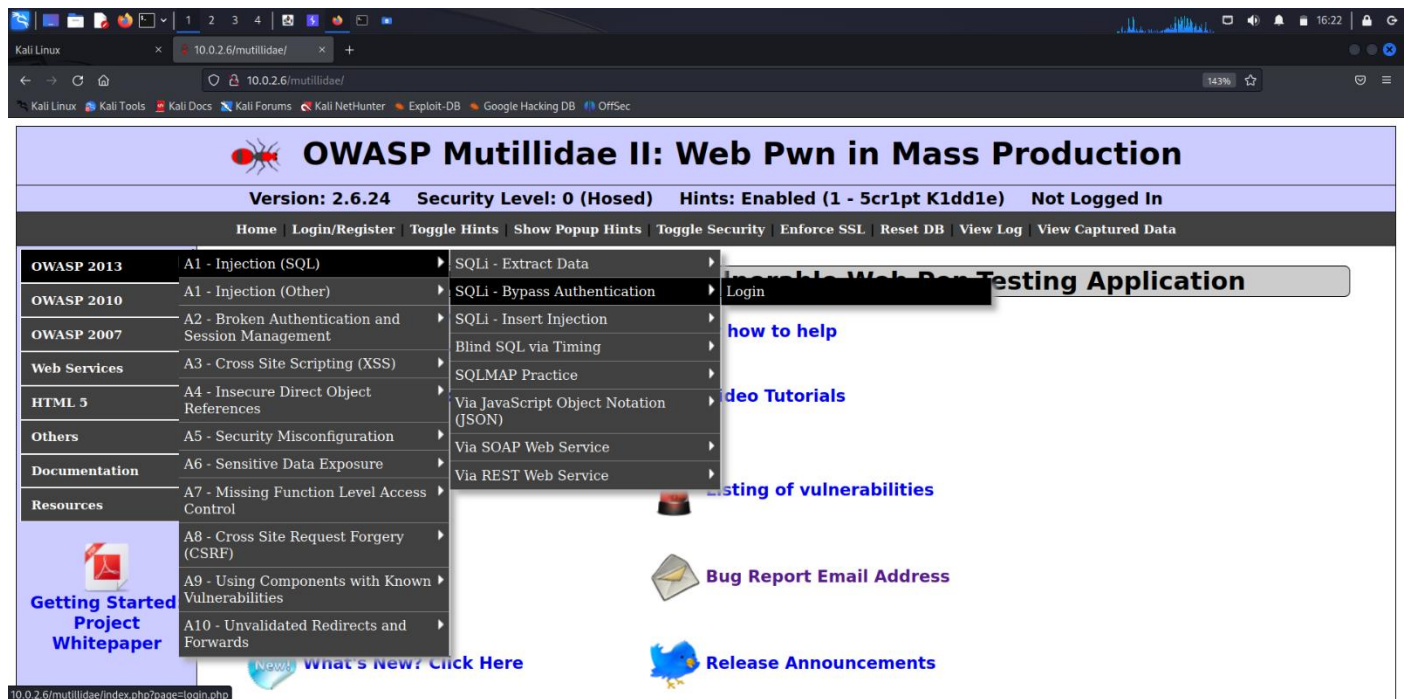


Open the OWASP Broken Web Application

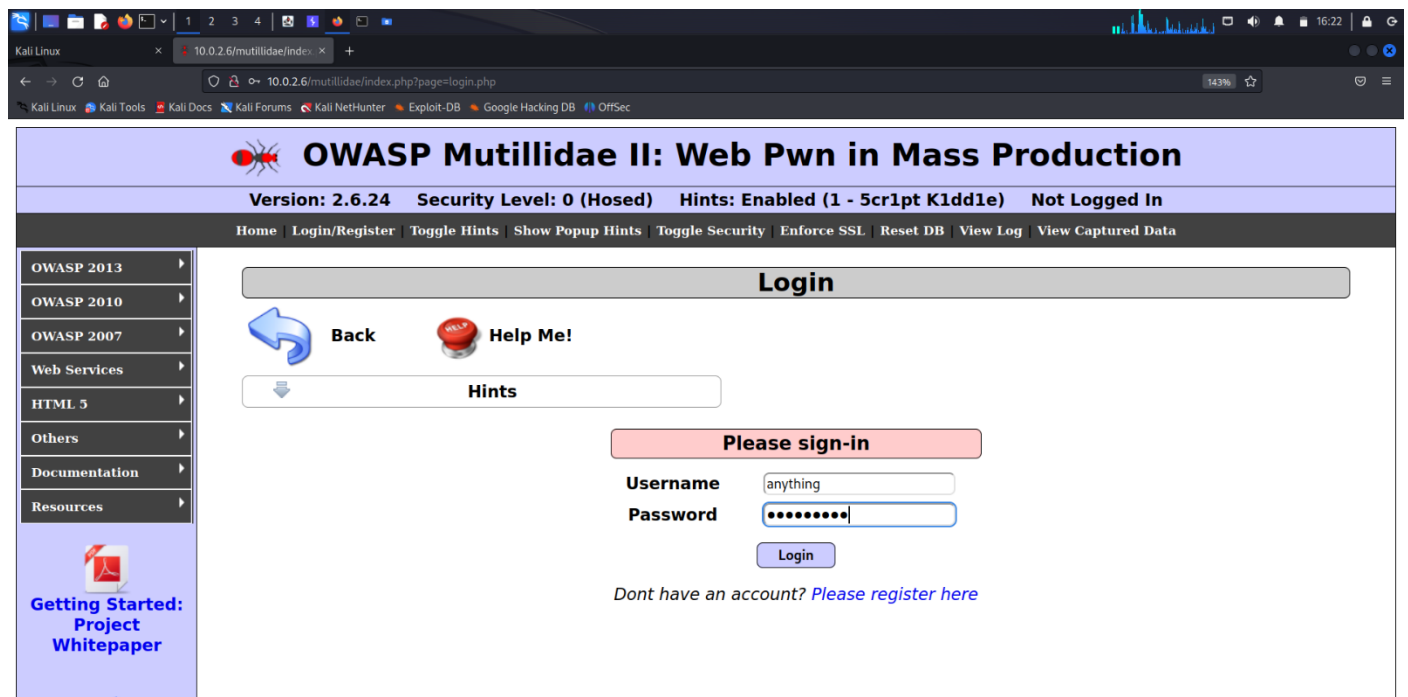


Open the Mutillidae II broken application in the OWASP BWA.

Open the Login page. The path of the Login page is displayed below.



Enter the fake credentials in the Login page.

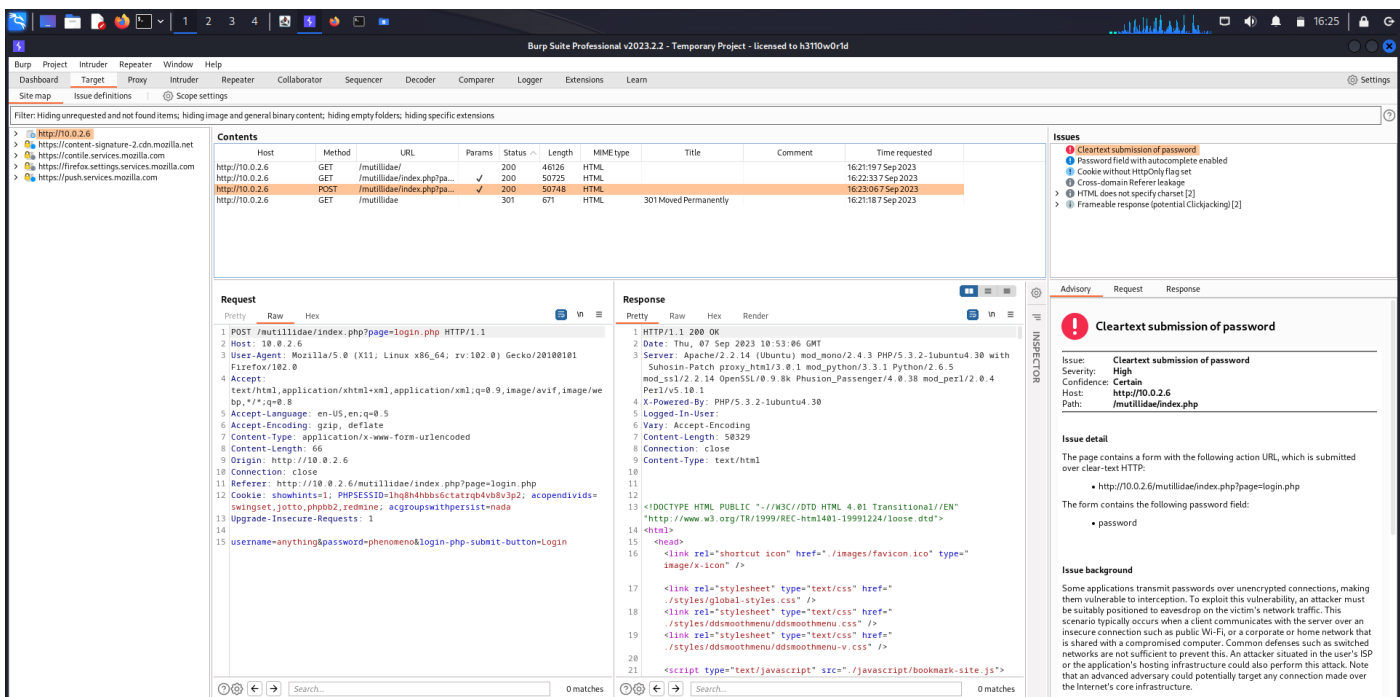


The entered username and password is obviously denied. The account does not exist.



Catch the entered username and password into the Target tab of the burpsuite.

The Request makes use of the POST method.



Right click the HTTP Request Message in the burpsuite and click on ‘Send to Intruder’.

When the HTTP Request Message is sent to the Intruder tab, it is highlighted in red color indicating that the message has been successfully sent to the Intruder tab.

The screenshot shows the Burp Suite Professional v2023.2.2 interface. The 'Intruder' tab is active, displaying a list of HTTP requests. The first request, a POST to `/mutillidae/index.php?page=login.php`, is selected and highlighted in red. The 'Request' and 'Response' panels are expanded, showing the raw HTTP data. The request is a POST with a body containing a username and password. The response is a 200 OK status with an HTML page. The 'Issues' panel on the right shows a 'Cleartext submission of password' issue, indicating that the password is transmitted over an unencrypted connection.

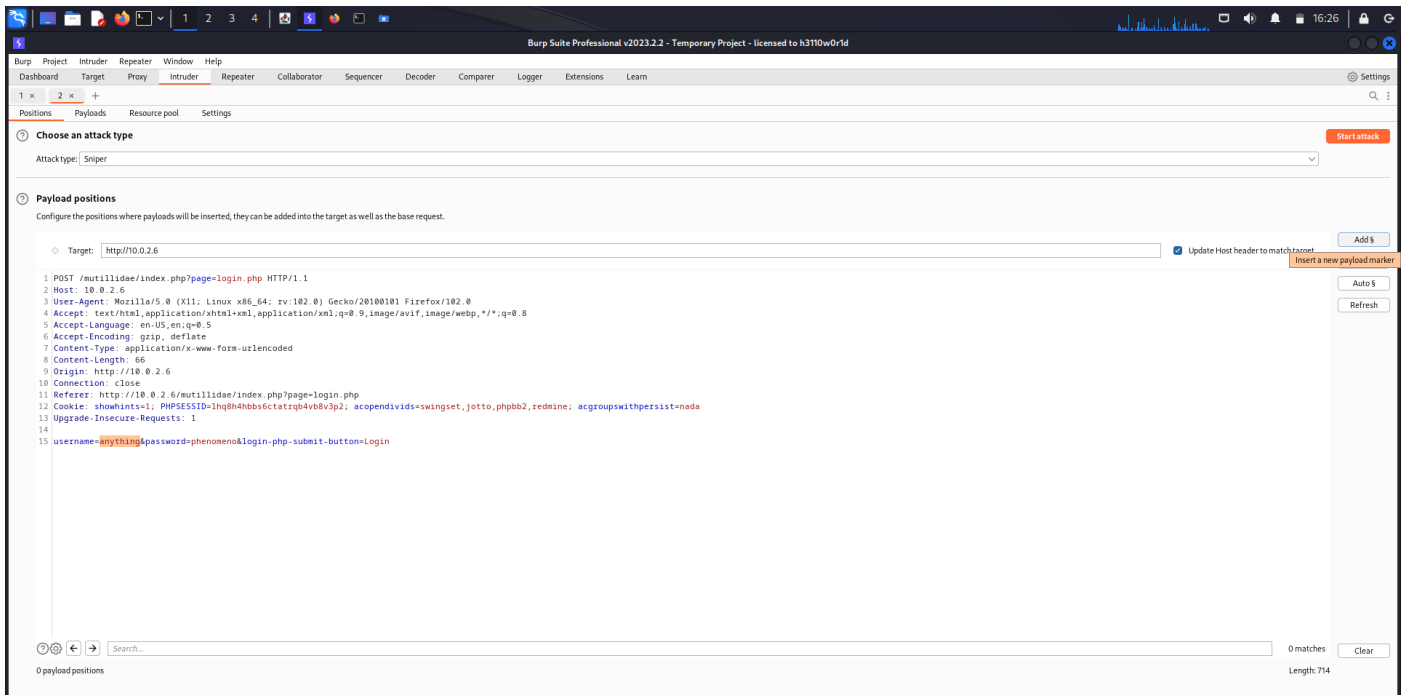
Open the Positions section in the Intruder tab.

First, clear all the payload markers by clicking Clear button on the right edge of Burpsuite.

The screenshot shows the 'Positions' section in the Burp Suite Professional v2023.2.2 interface. The 'Intruder' tab is active, and the 'Positions' section is expanded. It shows a list of payload positions, including the target URL `http://10.0.2.6` and the request body. The 'Clear all payload markers' button is visible on the right side of the interface.

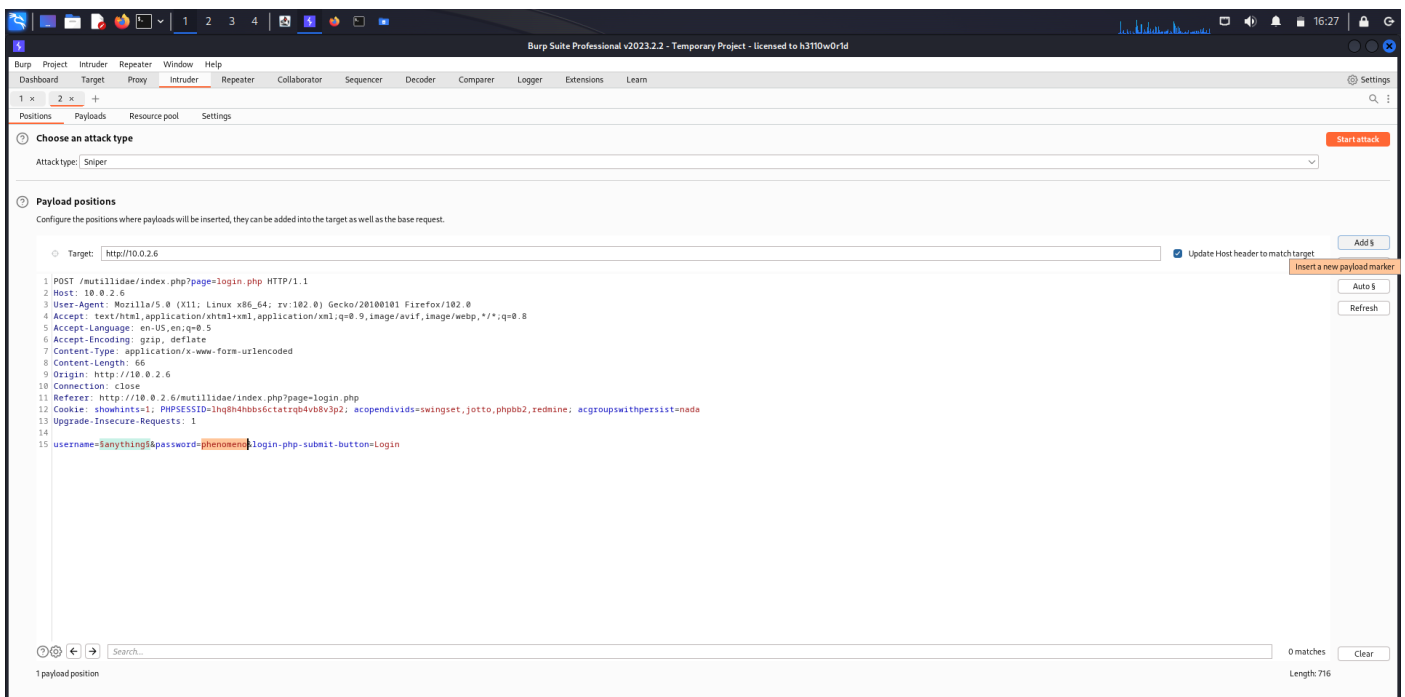
After the clearing the default markers, select the fake username which was entered and click on Add.

After adding the username, it is marked.



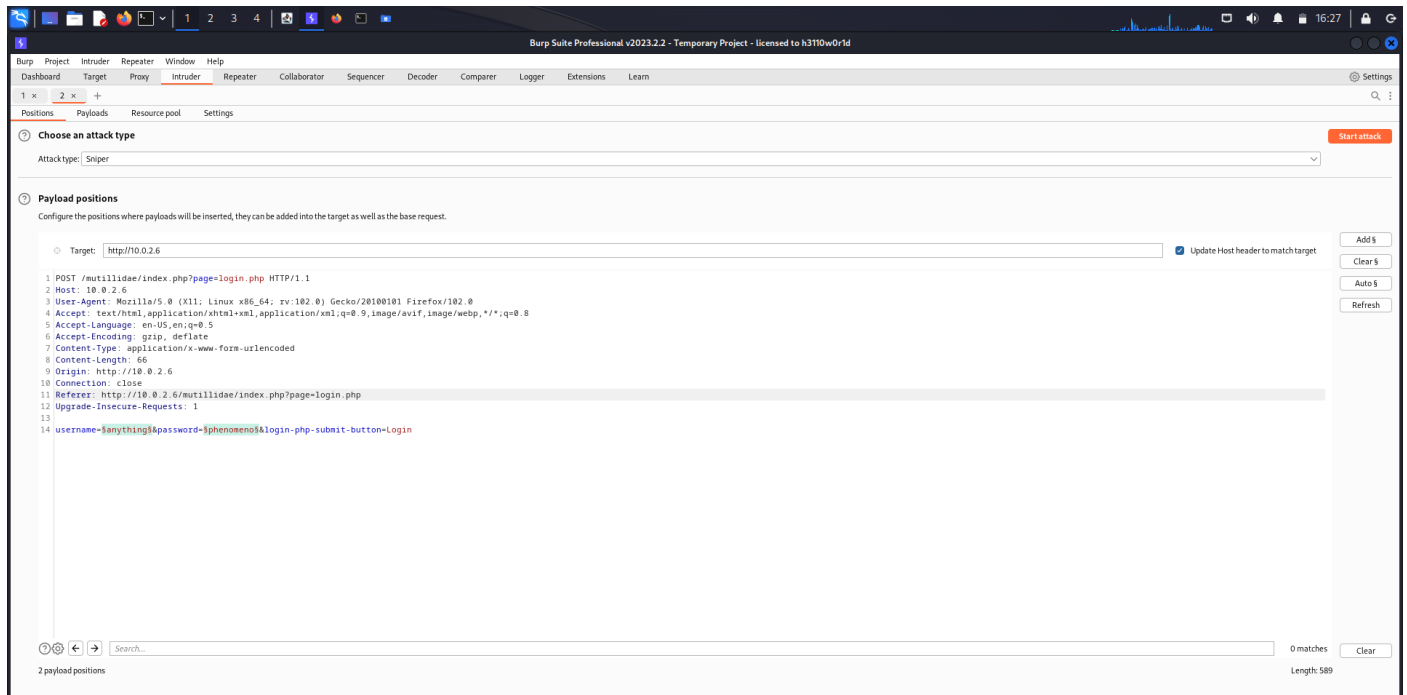
Select the fake password which was entered during the login process and click on Add on the right edge of Burpsuite.

After adding the password, it is marked.

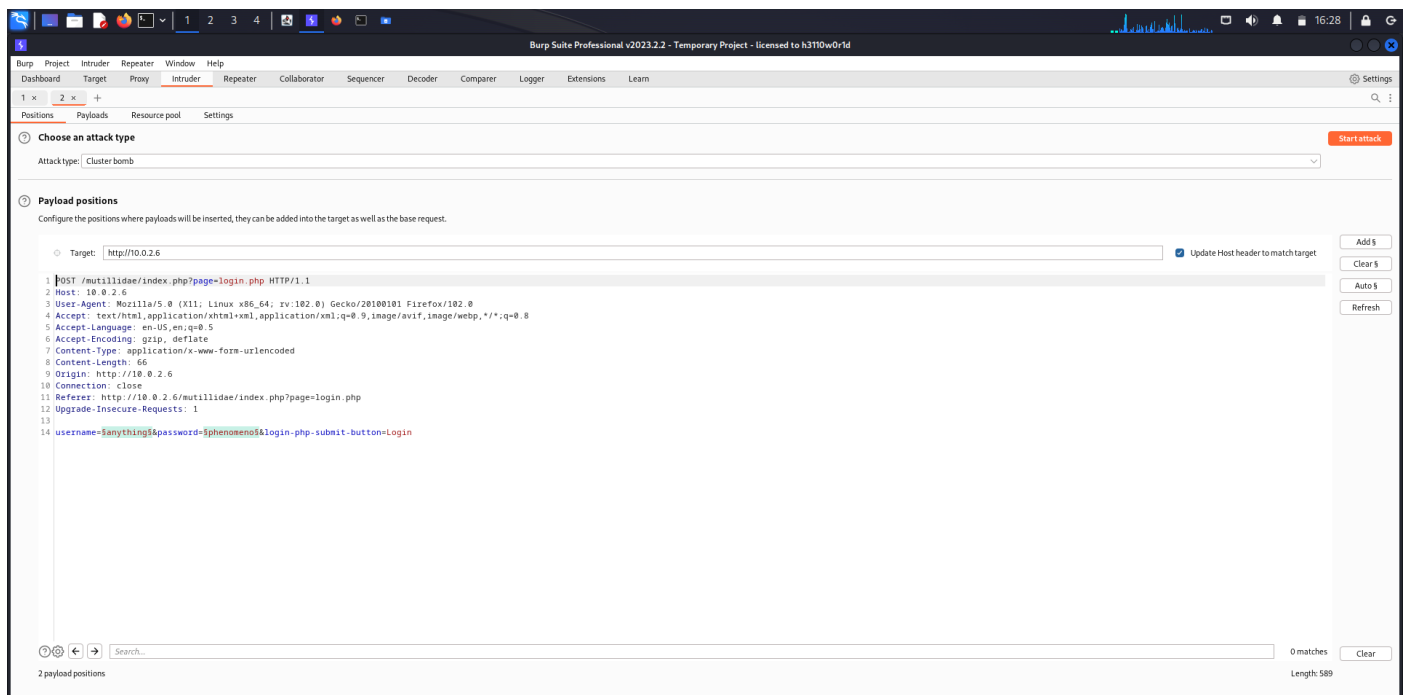


Remove the Cookie field from the HTTP Request message because it is irrelevant to the fuzzing attack.

We can see that both username and password have been marked.



Alter the attack type from Sniper(Only for one fuzzing point) to Cluster bomb(For multiple fuzzing point) since we are fuzzing for both username and password.



A file by the name `unix_users.txt` contains all the common usernames which are used in the web.

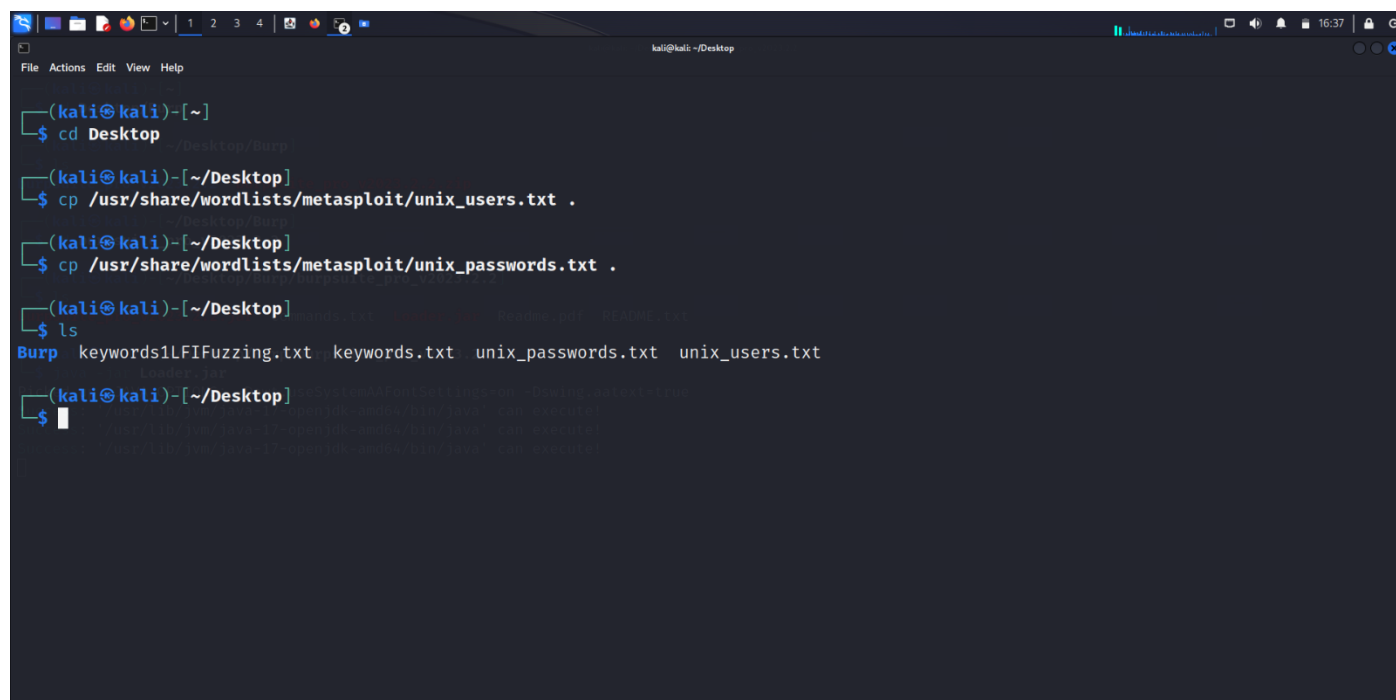
The file `unix_passwords.txt` contains all the common passwords which are used in web.

Copy the `unix_file.txt` onto a desired location.

Syntax: `cp source destination` //Copy command from source to destination

Command: `cp /usr/share/wordlists/metasploit/unix_users.txt .` // . represents the pwd

`cp /usr/share/wordlists/metasploit/unix_passwords.txt .`

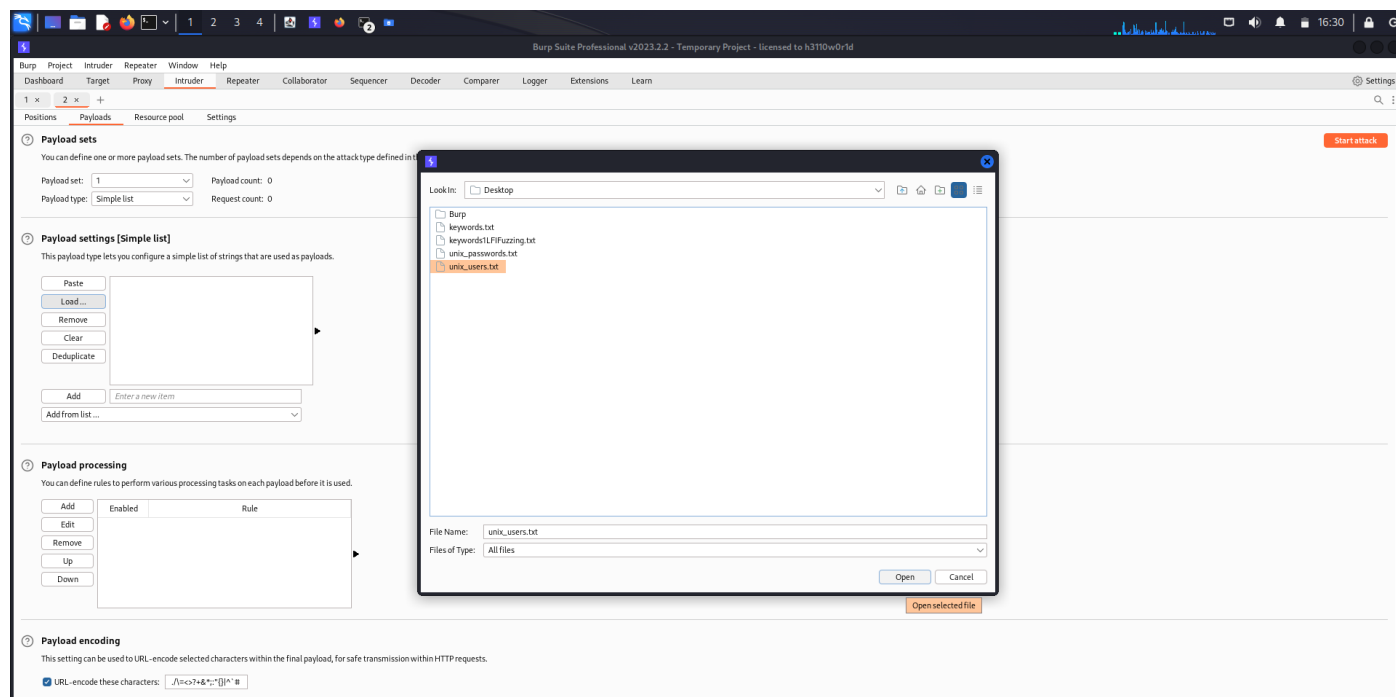


```
(kali㉿kali)-[~]
└─$ cd Desktop
      /Desktop/Burp
(kali㉿kali)-[~/Desktop]
└─$ cp /usr/share/wordlists/metasploit/unix_users.txt .
      /Desktop/Burp
(kali㉿kali)-[~/Desktop]
└─$ cp /usr/share/wordlists/metasploit/unix_passwords.txt .
      /Desktop/Burp
(kali㉿kali)-[~/Desktop]
└─$ ls
Burp  keywords1LFIffuzzing.txt  keywords.txt  unix_passwords.txt  unix_users.txt
└─$
```

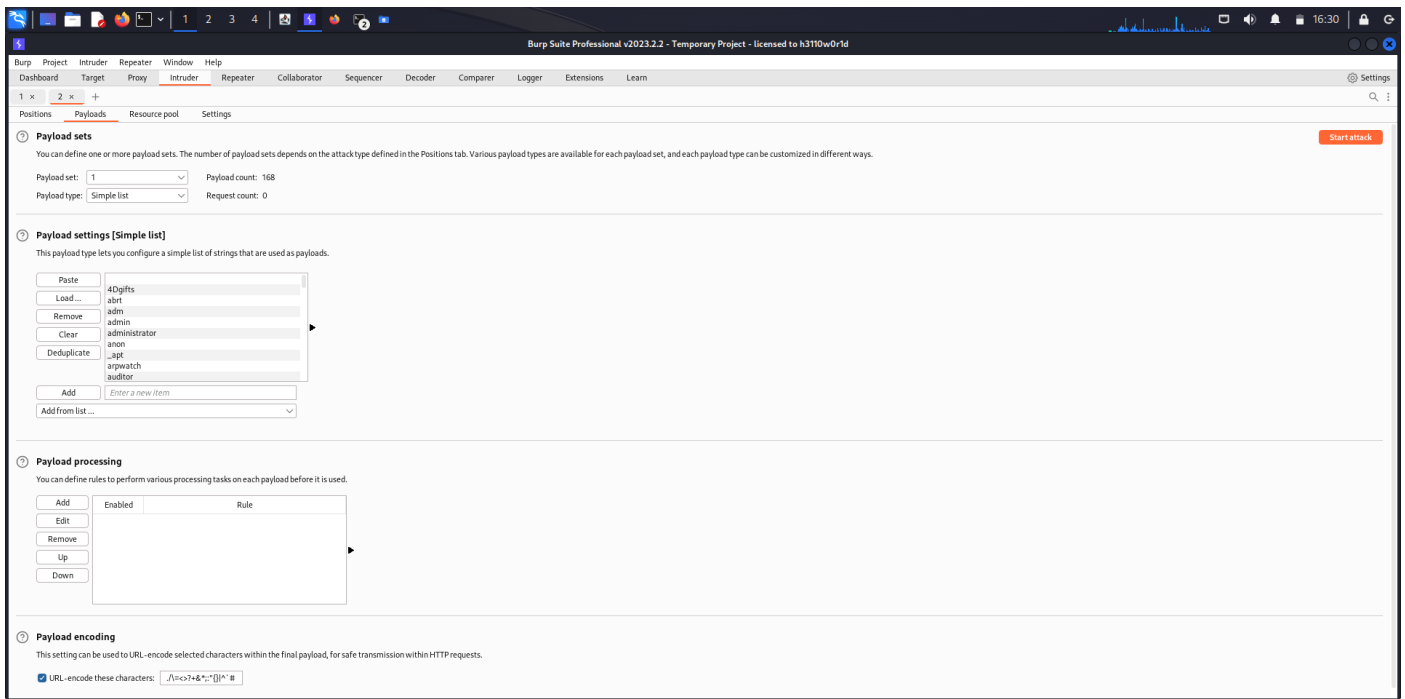
Open the Payloads section in the Intruder tab.

Set Payload set as 1 and Payload type as Simple List.

Load the `unix_users.txt` file in the Payload settings.

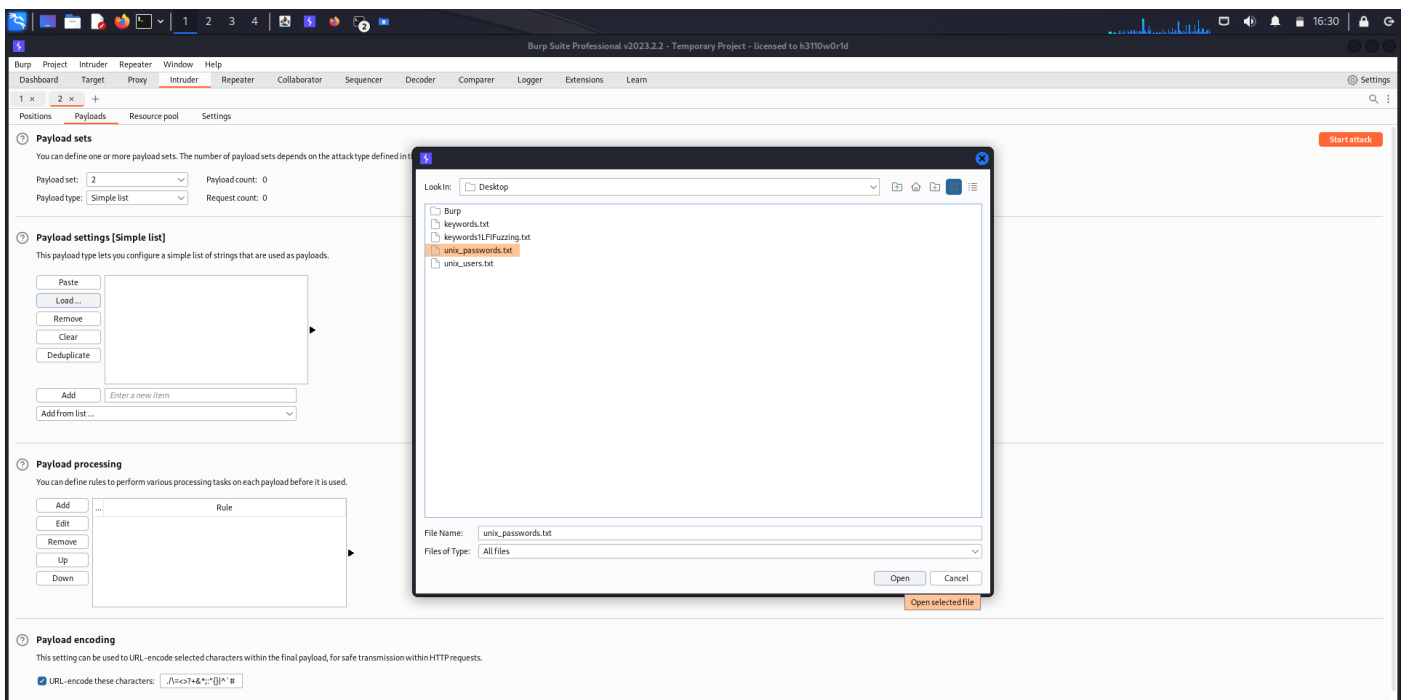


We can see that unix_users.txt has been loaded onto the burpsuite.

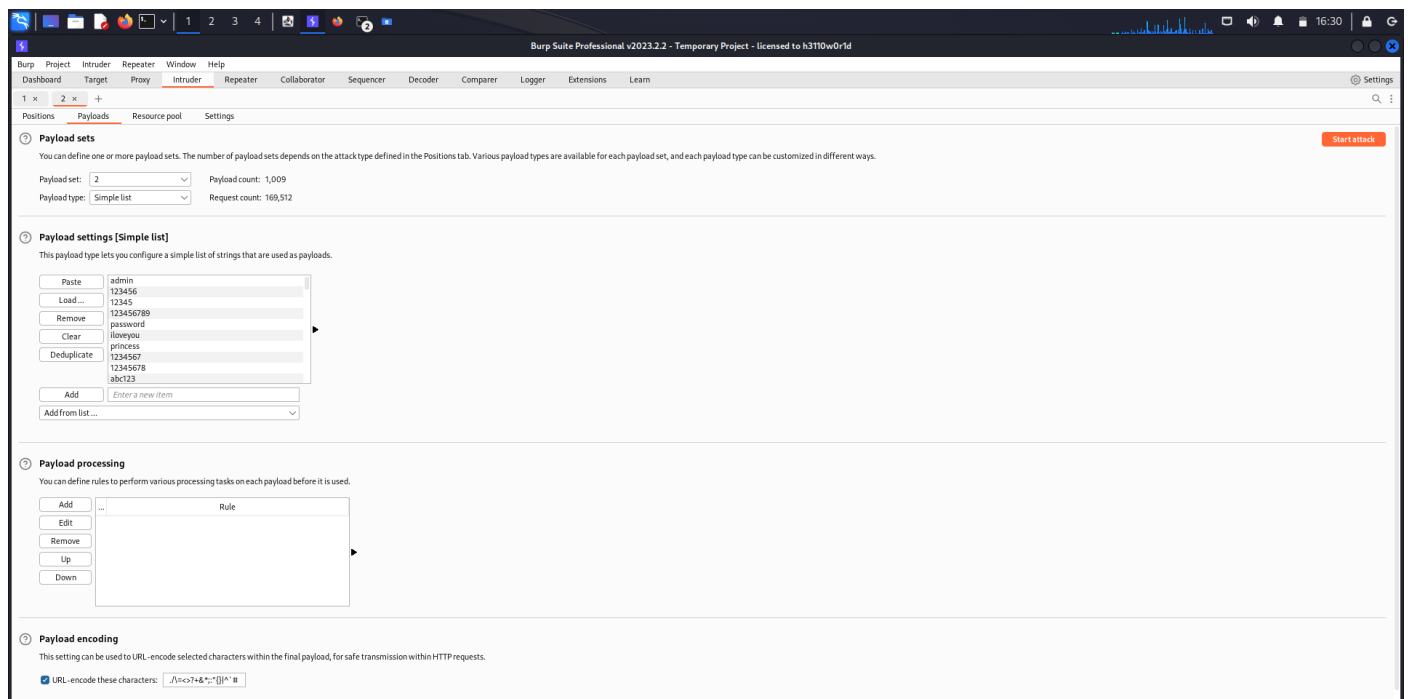


Set Payload set as 2 and Payload type as Simple List.

Load the unix_passwords.txt file in the Payload settings.

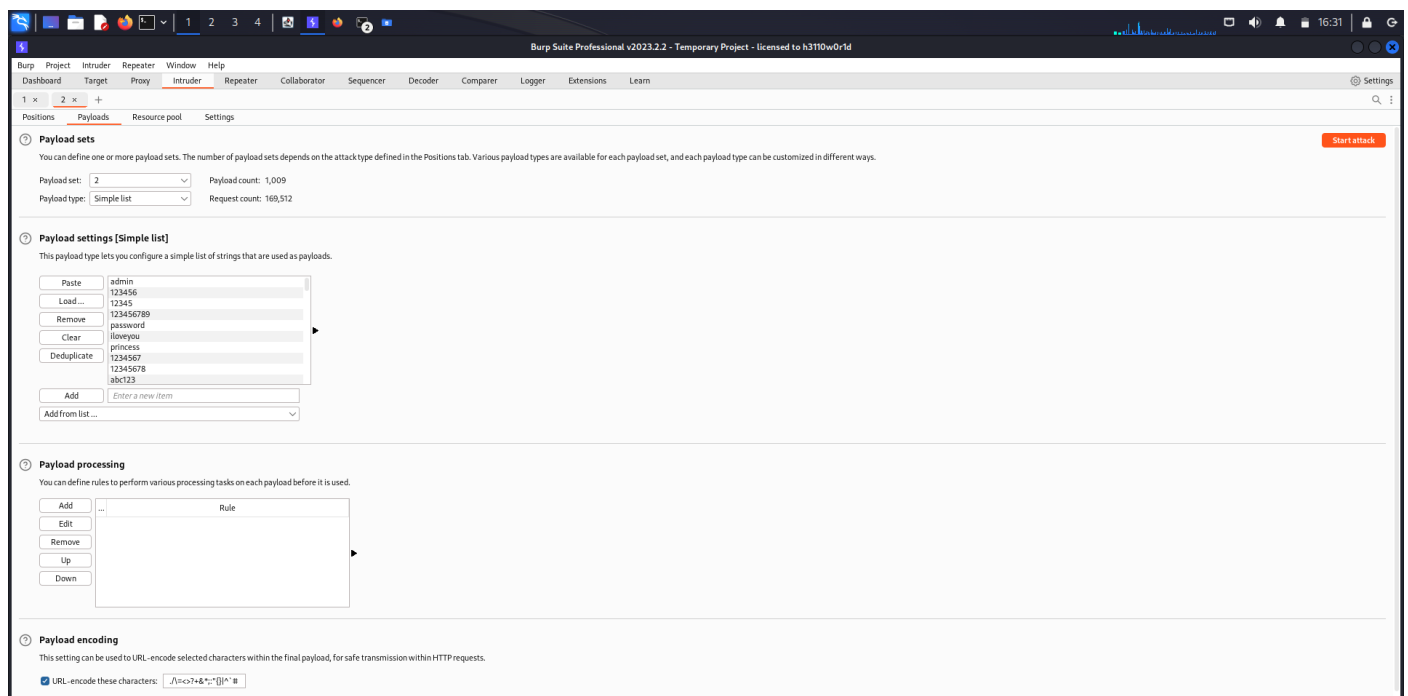


We can see that unix_passwords.txt has been loaded onto the burpsuite.



Note: If the password is selected and added first, then password list(unix_password.txt) should be loaded first in Payload set 1.

After configuring the payload, start the attack by clicking on the Start Attack button on the top-right corner of burpsuite.



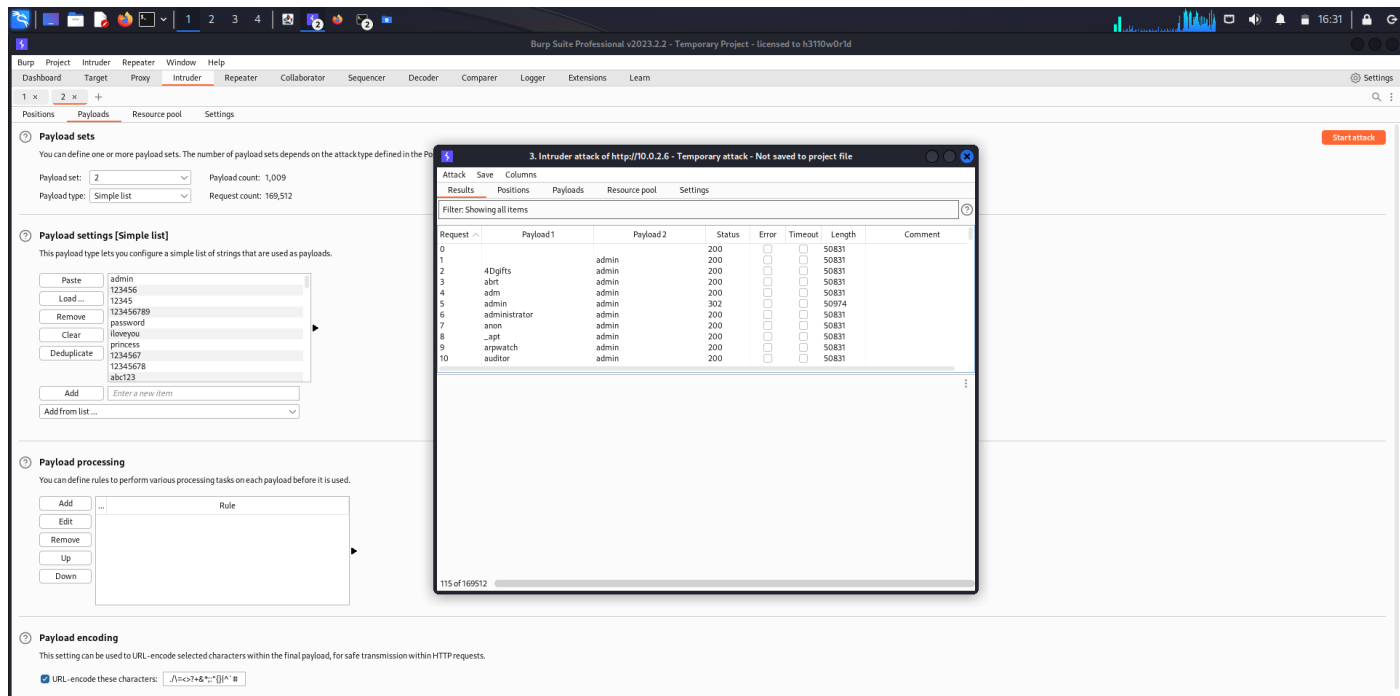
The attack has been started.

The file unix_users.txt contains 168 usernames.

The file unix_passwords.txt contains 1009 passwords.

Burpsuite will perform combinations of all usernames and passwords(168*1009=169512).

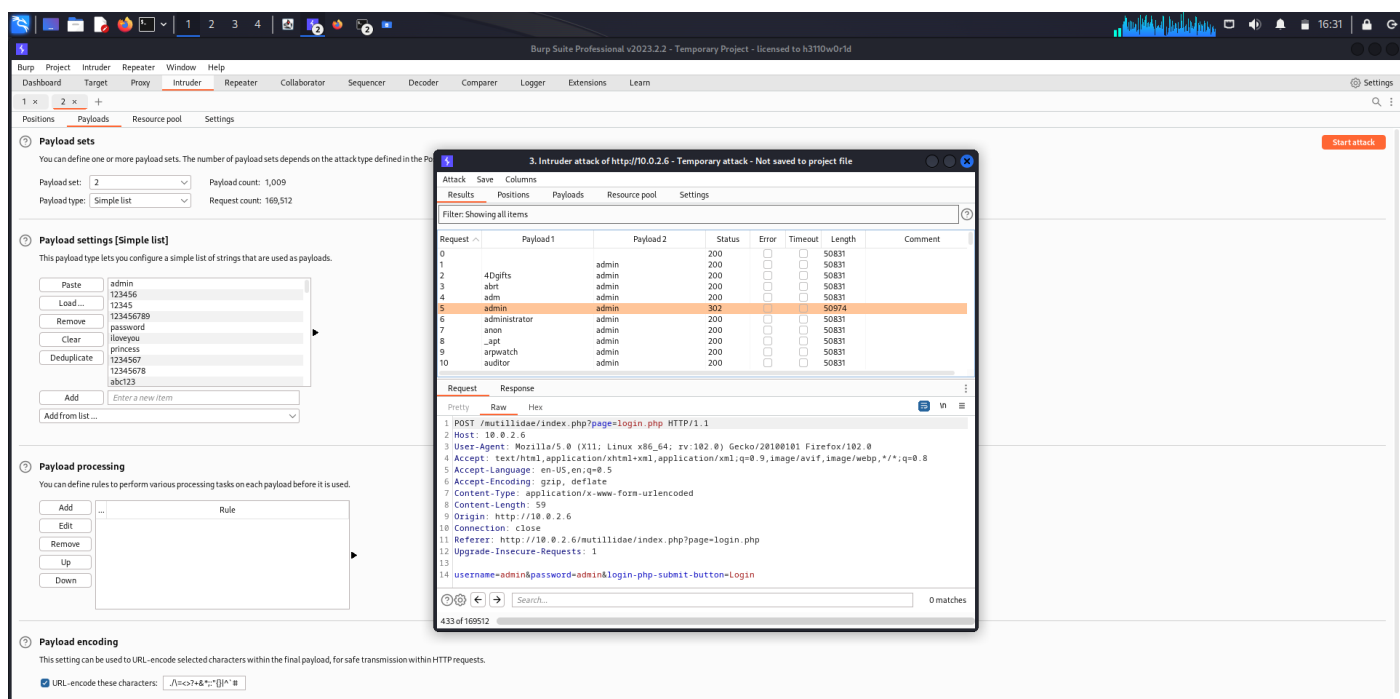
Hence the burpsuite will perform 169512 number of requests on the Login page.



Click on the cracked username and password.

It displays the HTTP Request message sent to that Login page.

In this case, the username turned out to be admin and also the password turned out to be the same.



We can also see the HTTP Response message sent by the webpage to the request which was made.

Burp Suite Professional v2023.2.2 - Temporary Project - licensed to h3T10w0r1d

3. Intruder attack of http://10.0.2.6 - Temporary attack - Not saved to project file

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
0			200			50831	
1	admin		200			50831	
2	4Dg1ts	admin	200			50831	
3	abrt	admin	200			50831	
4	adm	admin	200			50831	
5	admin	admin	302			50974	
6	administrator		200			50831	
7	anon	admin	200			50831	
8	_apt	admin	200			50831	
9	arpwatch	admin	200			50831	
10	auditor	admin	200			50831	

Request Response

1 HTTP/1.1 302 Found
2 Date: Thu, 07 Sep 2023 11:01:20 GMT
3 Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.38 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
4 X-Powered-By: PHP/5.3.2-1ubuntu4.38
5 Set-Cookie: PHPSESSID=mat7h3aot6rvsu0259qh1111; path=/
6 Set-Cookie: showInt=1
7 Set-Cookie: username=admin
8 Set-Cookie: uid=1
9 Location: index.php?popUpNotificationCode=AUI
10 Logged-In-User: admin
11 Vary: Accept-Encoding
12 Content-Length: 58370
13 Connection: close
14 Content-Type: text/html

We can end the attack as we have attained the username and password of the login page in just the fifth request.

Burp Suite Professional v2023.2.2 - Temporary Project - licensed to h3T10w0r1d

3. Intruder attack of http://10.0.2.6 - Temporary attack - Not saved to project file

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
0			200			50831	
1	admin		200			50831	
2	4Dg1ts	admin	200			50831	
3	abrt	admin	200			50831	
4	adm	admin	200			50831	
5	admin	admin	302			50974	
6	administrator		200			50831	
7	anon	admin	200			50831	
8	_apt	admin	200			50831	
9	arpwatch	admin	200			50831	
10	auditor	admin	200			50831	

Request Response

1 POST /mutillidae/index.php?page=login.php
2 Host: 10.0.2.6
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safari/537.36
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 59
9 Origin: http://10.0.2.6
10 Connection: close
11 Referer: http://10.0.2.6/mutillidae/index.php?page=login.php
12 Upgrade-Insecure-Requests: 1
13
14 username=admin&password=admin&login.php-submit-button=Login

Do you want Intruder to continue running this attack in the background?

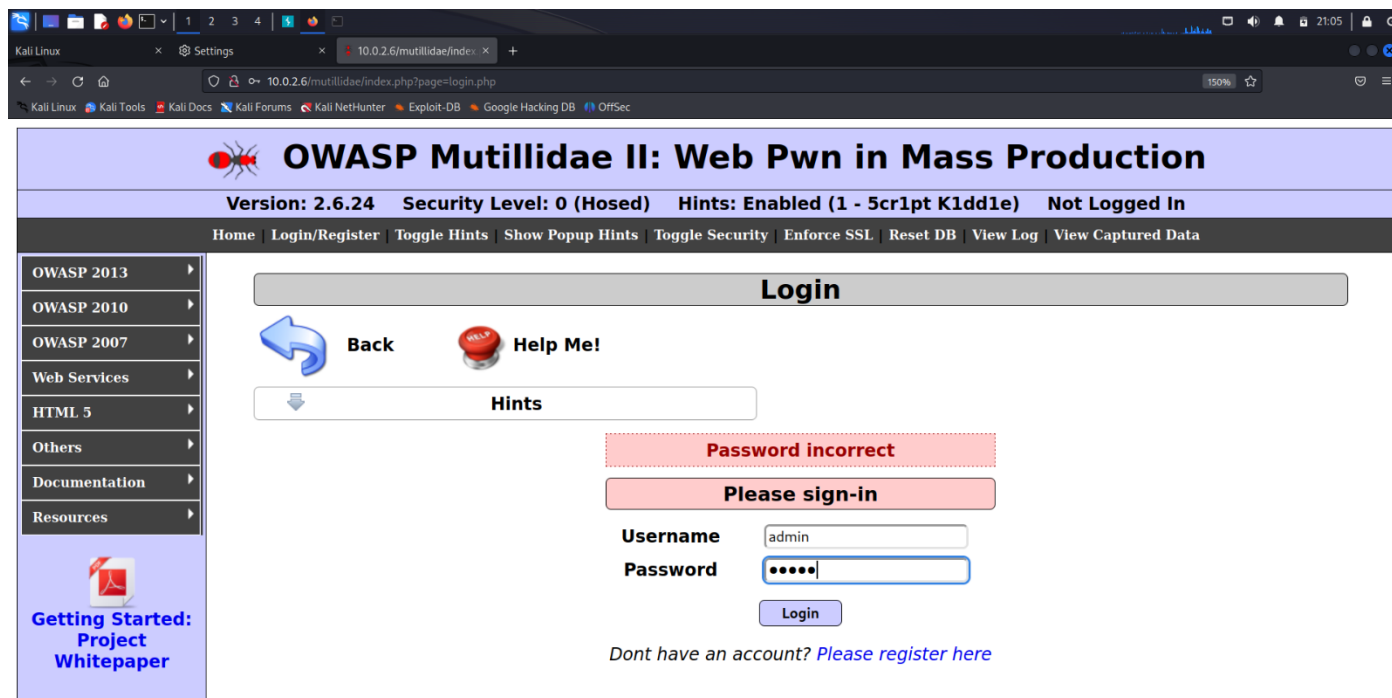
You can monitor its progress and access the results from the dashboard.

☐ Remember my choice when closing attacks in future
You can change this setting in the Intruder menu.

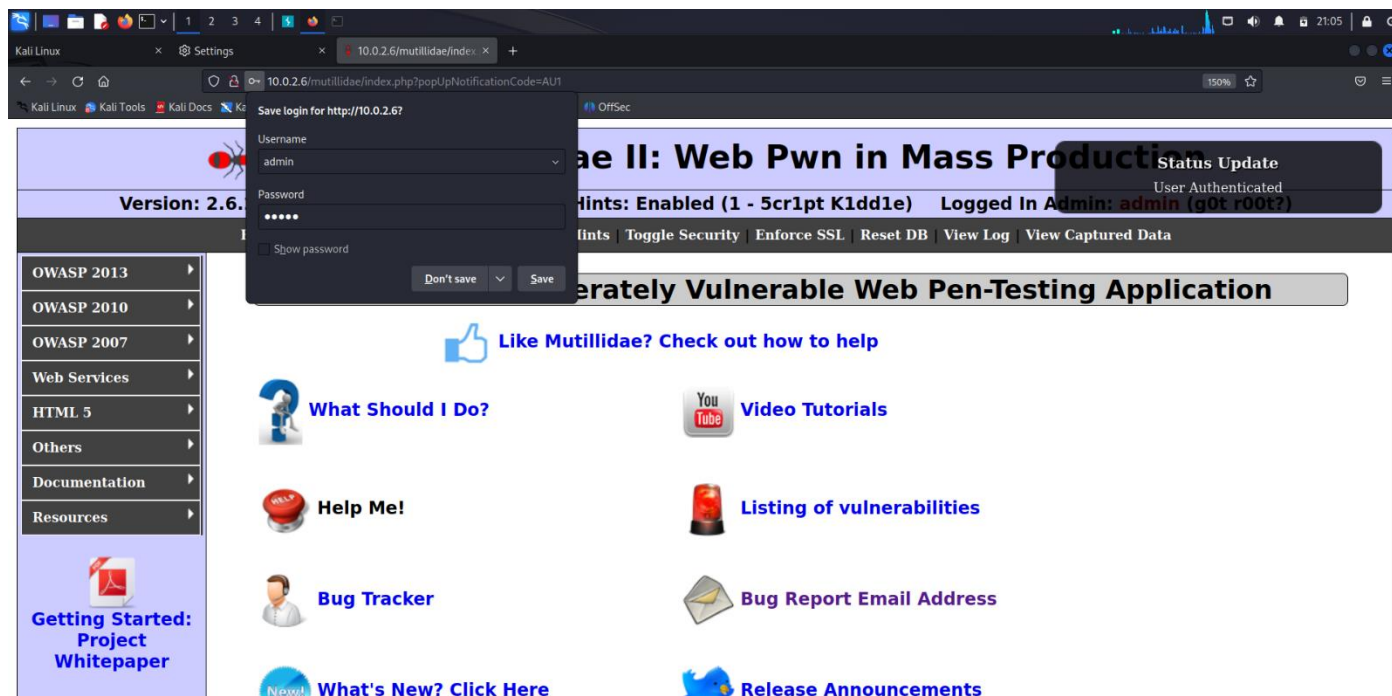
We enter the cracked username and password in the login page.

Username: admin

Password: admin



We have successfully logged in with the given username and password.



Analysis

In this report, we provide a comprehensive analysis of the fuzzing technique as a pivotal component of modern software security testing. Fuzzing, a systematic method of sending unexpected and malformed data inputs to software applications, plays a critical role in identifying vulnerabilities and defects that may otherwise go unnoticed. Our analysis highlights the effectiveness of fuzzing in uncovering security weaknesses, including buffer overflows, injection attacks, and denial-of-service vulnerabilities. We also discuss the importance of integrating fuzzing into software development and quality assurance processes to proactively enhance software security and reduce the risk of exploitation.

Conclusion

In conclusion, this report underscores the significant value of fuzzing as a vital tool in the realm of software security testing. Fuzzing has proven to be a highly effective technique for systematically identifying vulnerabilities and defects within software applications and systems. Its ability to provoke unexpected behaviour, crashes, and security issues enables early detection and mitigation of potential threats. As software security continues to be a paramount concern in our interconnected world, the integration of fuzzing into development and testing processes remains a critical step toward enhancing software resilience and reducing the risk of security breaches.