Authentication

What is authentication?

Authentication is the process of verifying the identity of a user or client. Websites are potentially exposed to anyone who is connected to the internet. This makes robust authentication mechanisms integral to effective web security.

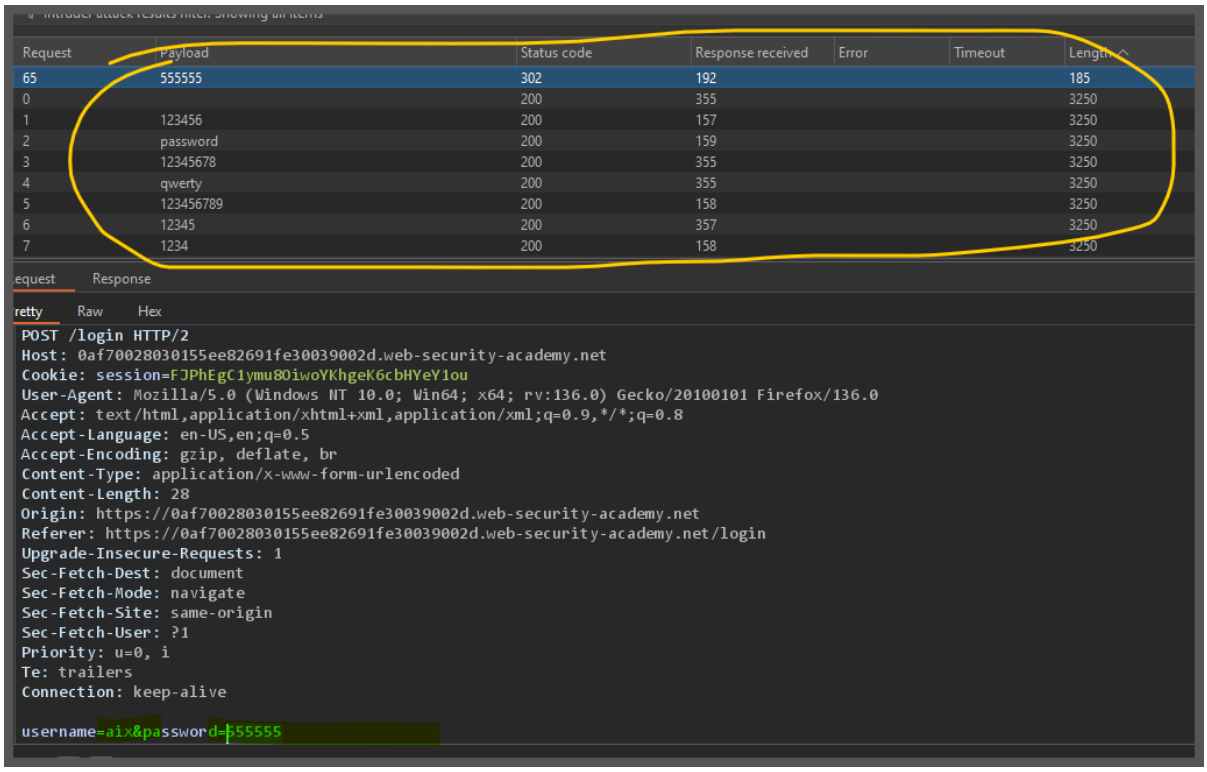
There are three main types of authentication:

- Something you **know**, such as a password or the answer to a security question. These are sometimes called "knowledge factors".
- Something you **have**, This is a physical object such as a mobile phone or security token. These are sometimes called "possession factors".
- Something you **are** or do. For example, your biometrics or patterns of behavior. These are sometimes called "inherence factors".

Authentication mechanisms rely on a range of technologies to verify one or more of these factors.

Lab: Username enumeration via different responses

> This lab is vulnerable to username enumeration and password brute-force attacks. It has an account with a predictable username and password, which can be found in the following wordlists:
>
> - Candidate usernames
> - Candidate passwords
>
> To solve the lab, enumerate a valid username, brute-force this user's password, then access their account page.



first brute force the use name

then once you got the username find it with different reponse

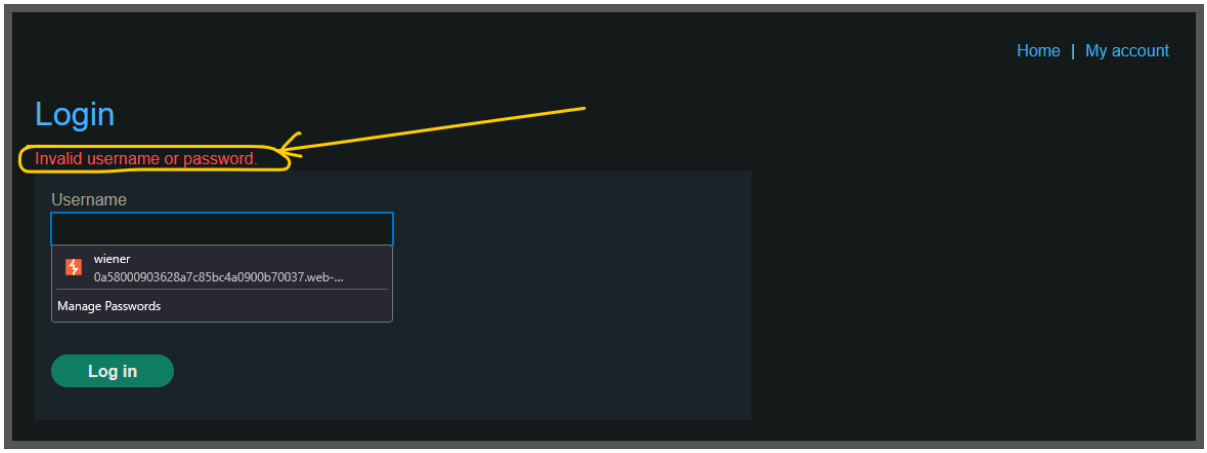
then brute force passwrod, simple.

Lab: Username enumeration via subtly different responses

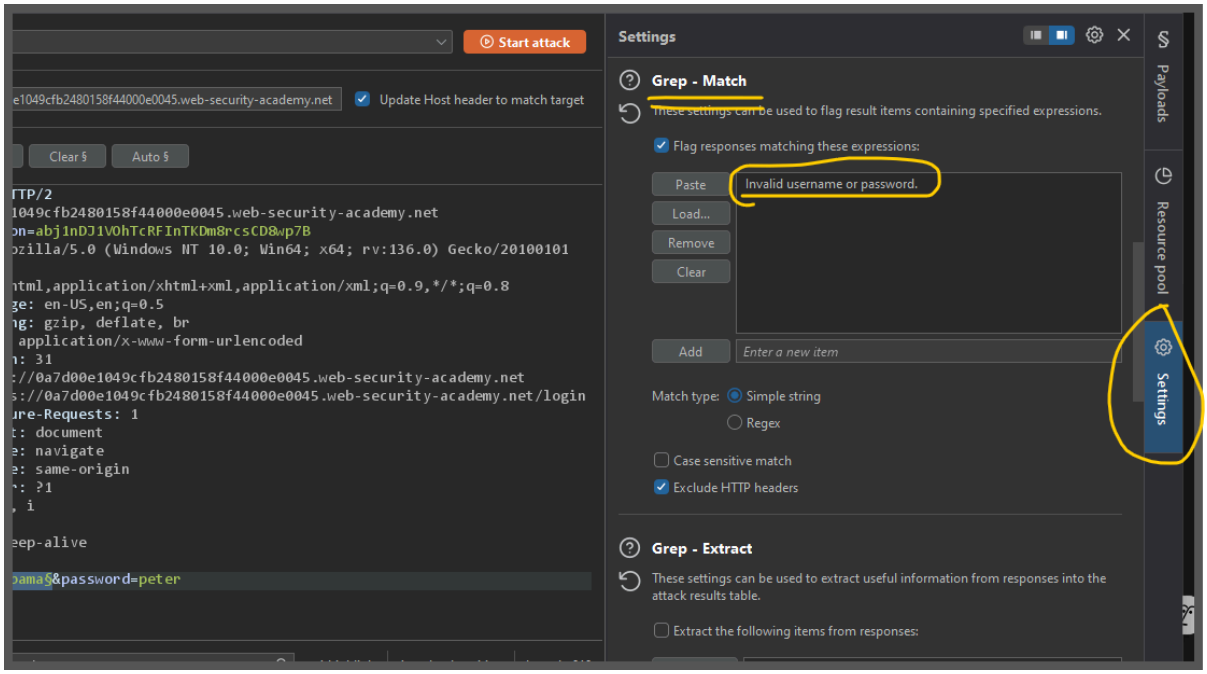
This lab is subtly vulnerable to username enumeration and password brute-force attacks. It has an account with a predictable username and password, which can be found in the following wordlists:

- Candidate usernames
- Candidate passwords

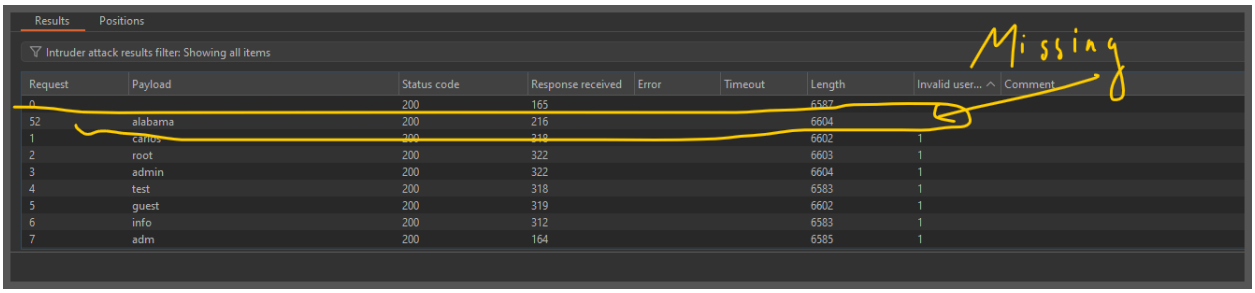
To solve the lab, enumerate a valid username, brute-force this user's password, then access their account page.



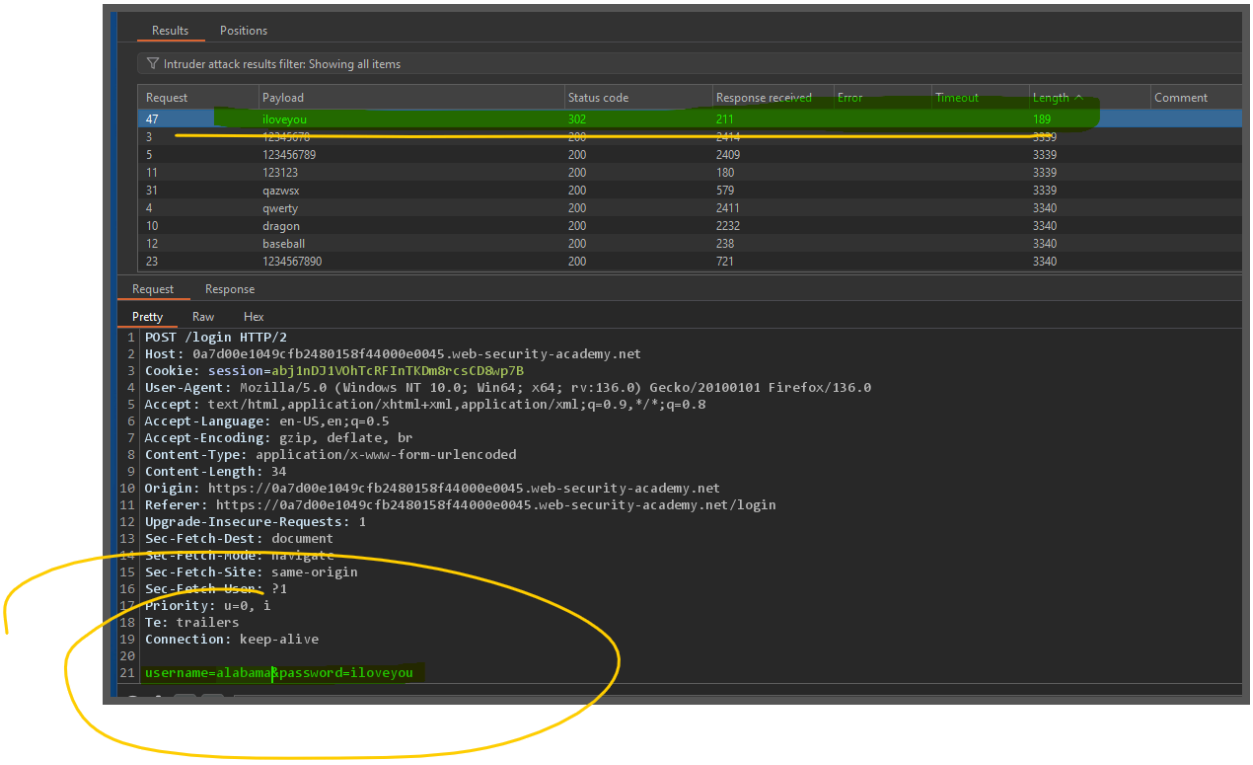
- when you enter the wrong username ans passwrod then you get this string
- lest brute froce the username first



- in this case we pasted that string see if thing string is not available in the response
- do it with grep and extract i did with grep and match “both worked”



- now bruete force the passwrod

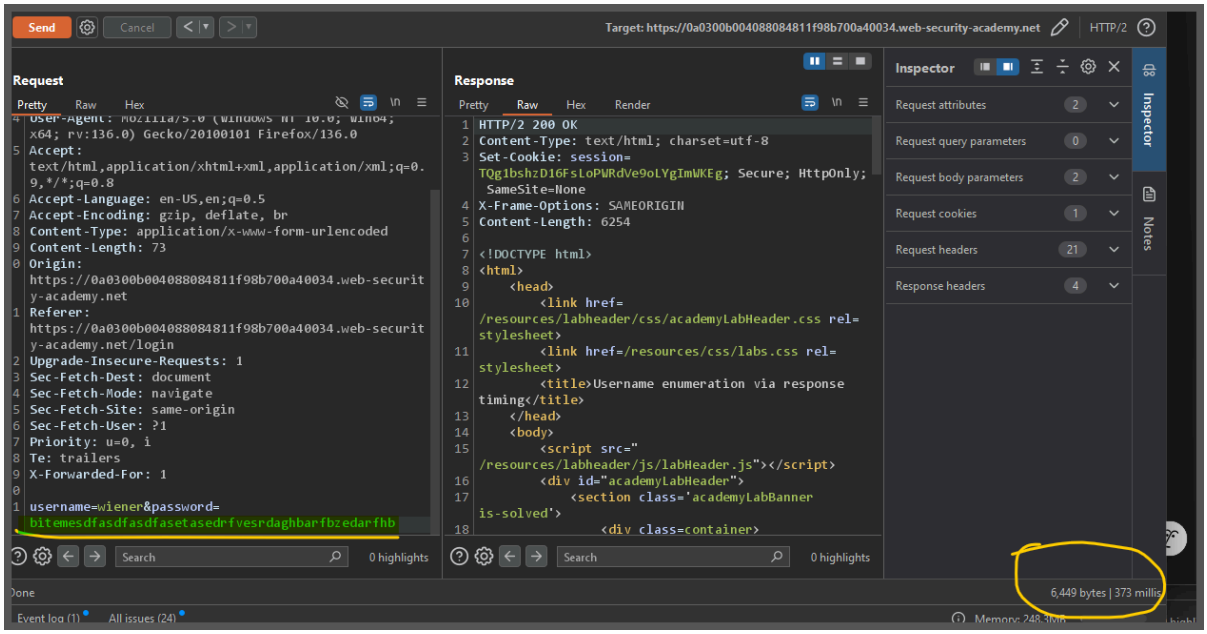


Lab: Username enumeration via response timing

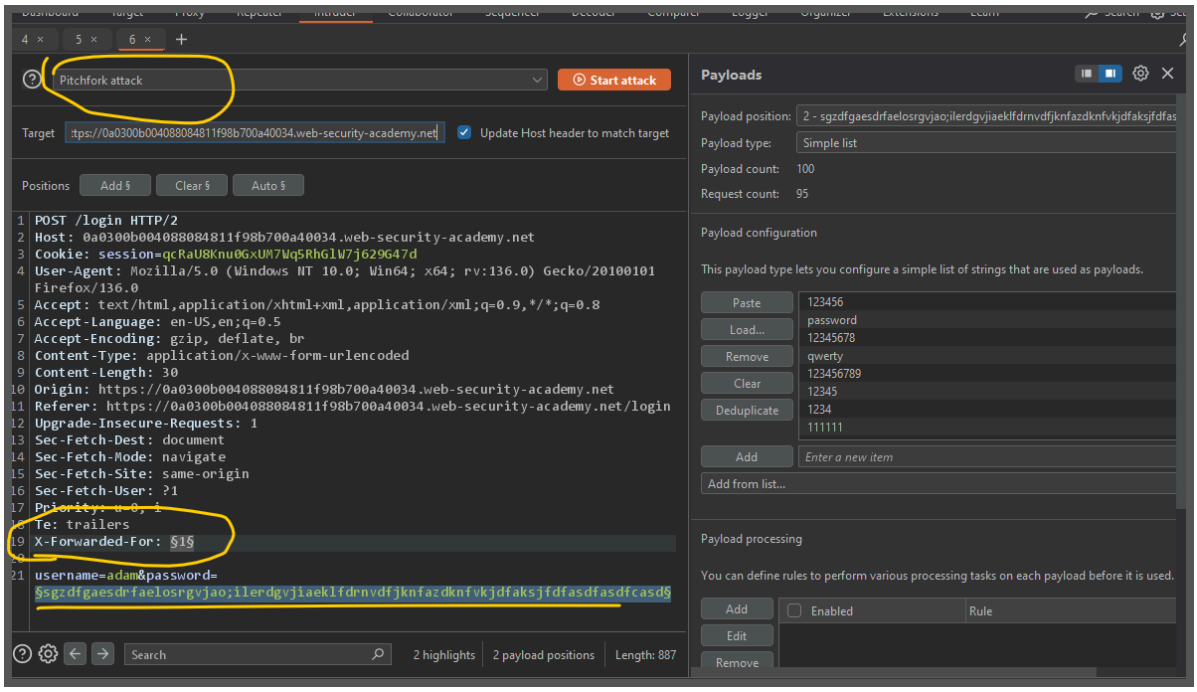
This lab is vulnerable to username enumeration using its response times. To solve the lab, enumerate a valid username, brute-force this user's password, then access their account page.

- Your credentials: `wiener:peter`
- Candidate usernames
- Candidate passwords

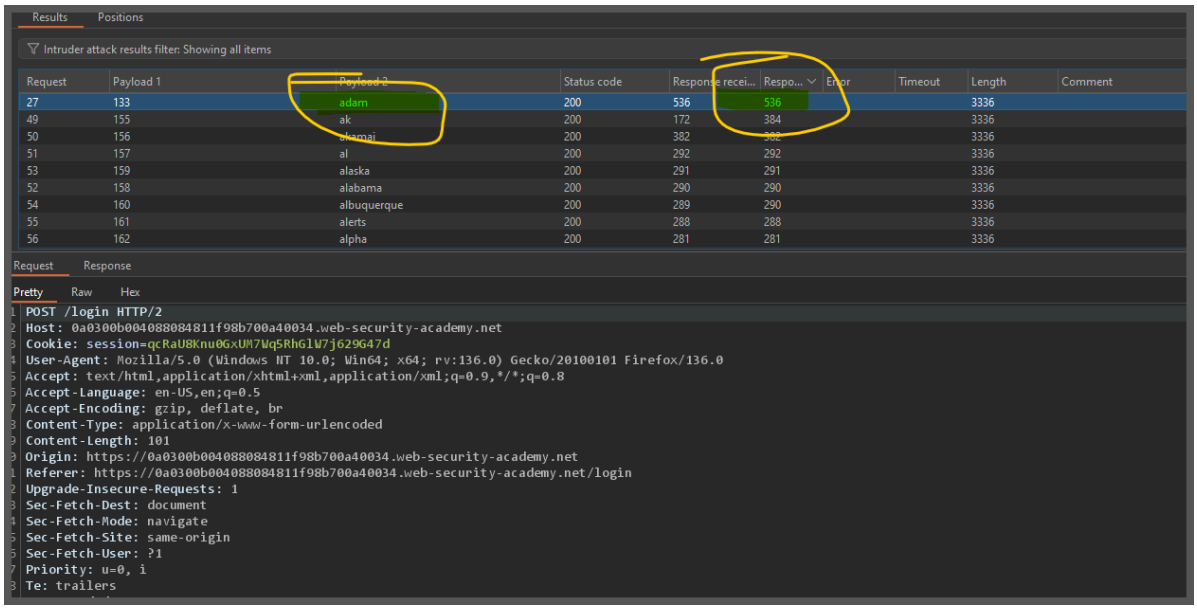
- in this lab we have to find username with the help of reponse time
- once user name is correct then it process for the password
- it means look for whether password is correct or not

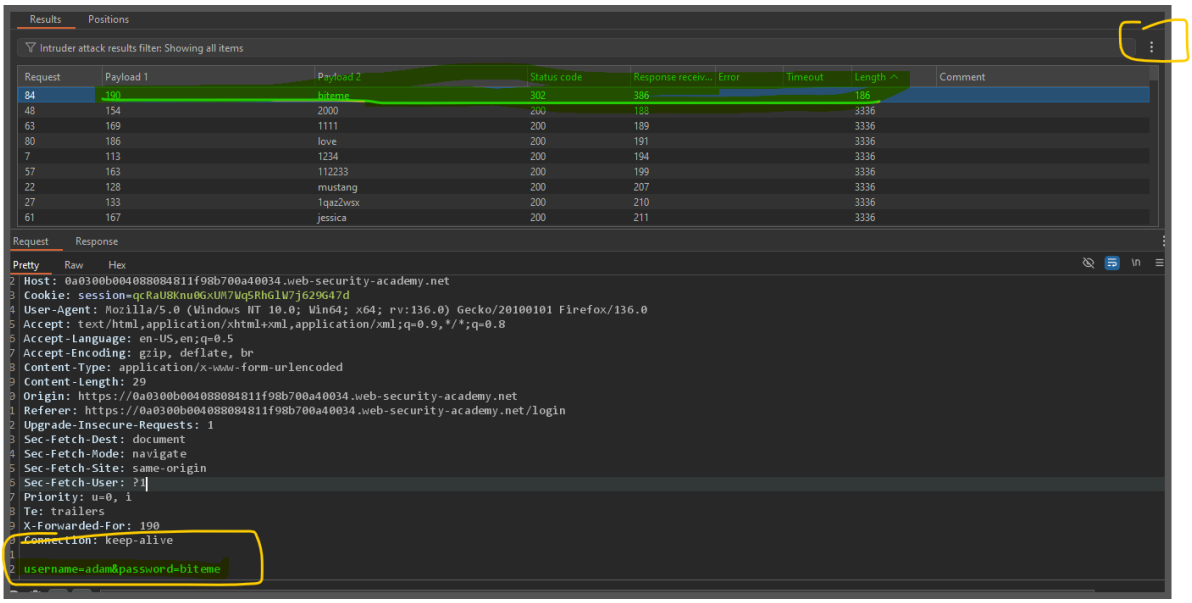


- we know username is correct wiener then it go for password and response time is much more because we have long random password which doesnt make any sense
- if user name is wrong then it does not check for the password so reponse time is low
- one more thing it also block wrong request based on ip
- so use `X-Forwarded-For: 1` and while bruteforcing change that one 1 also bureforce it



same method first find the username then password





solved

“note if your note able to see that column `respone recivied` one then get it from that three dot option “

Lab: Broken brute-force protection, IP block

This lab is vulnerable due to a logic flaw in its password brute-force protection. To solve the lab, brute-force the victim's password, then log in and access their account page.

- Your credentials: `wiener:peter`
- Victim's username: `carlos`
- Candidate passwords

Solution:

- so basically in this lab after three attamp we get block for one minute
- but what if after three attempt we tired correct one
- i mean brute forcing rright but at third time enter the correct use name and passd that we have wiener and peter and again brueteforce for carlos
- then again after two attemp enter again wiener and peter

- now we need user name and password right like

  carlos pass123

  carlos paseqwe

  wiener peter

  carlos qwerty

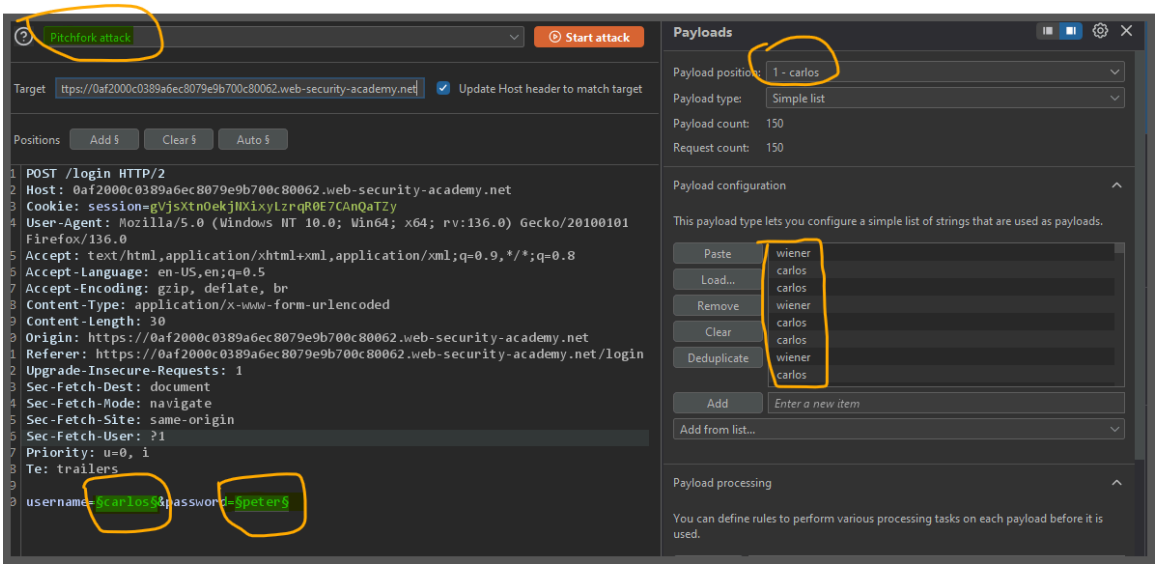
  carlos 12345

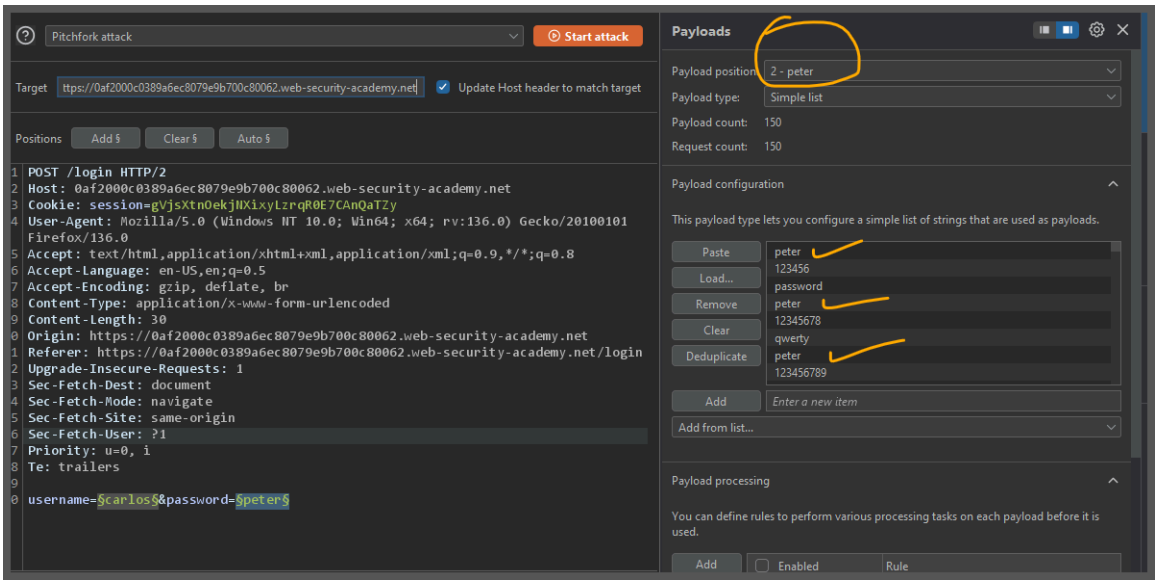
“like this so we have python”

```
print("###############the following user are:###############")
for i in range(150):
    if i % 3:
        print("carlos")
    else:
        print("wiener")


print("####################the following passwords are:####")
with open('pass.txt', 'r') as f:
    lines = f.readlines()

i = 0
for pwd in lines:
    if i % 3:
        print(pwd.strip('\n'))
    else:
        print("peter")
        print(pwd.strip('\n'))
        i = i+1
    i = i+1
```

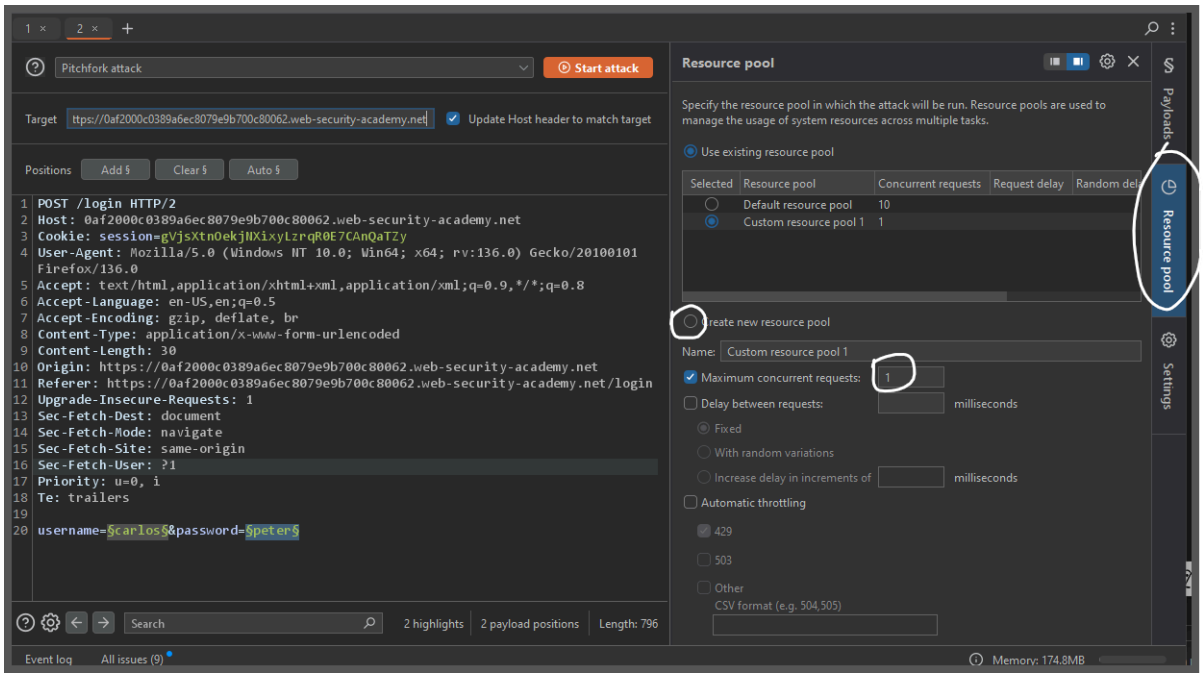
run the code in terminal and copy the output

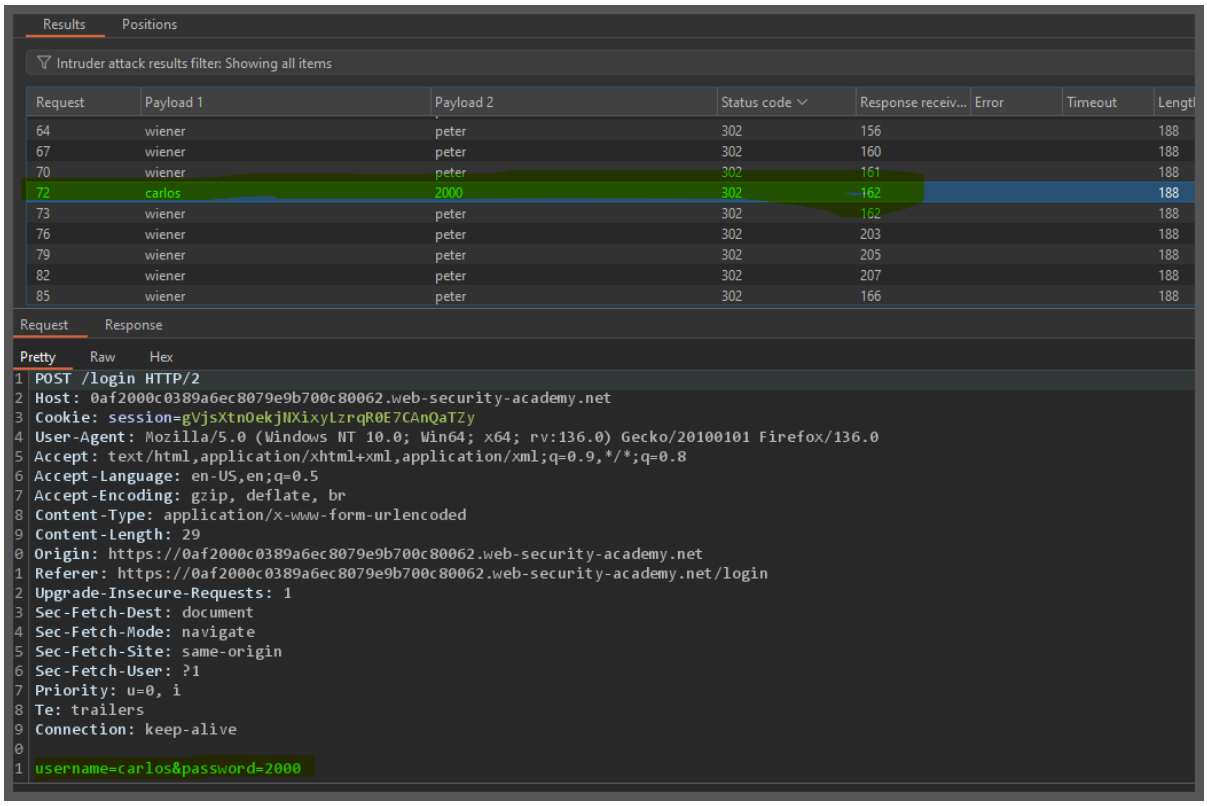
Note: copy that given password list from lab and save it in same directory were ever your going to run the srrinpt





send only one 1 request at a time make changes in burp or in default it sends 10 then if your sending ten they will block you





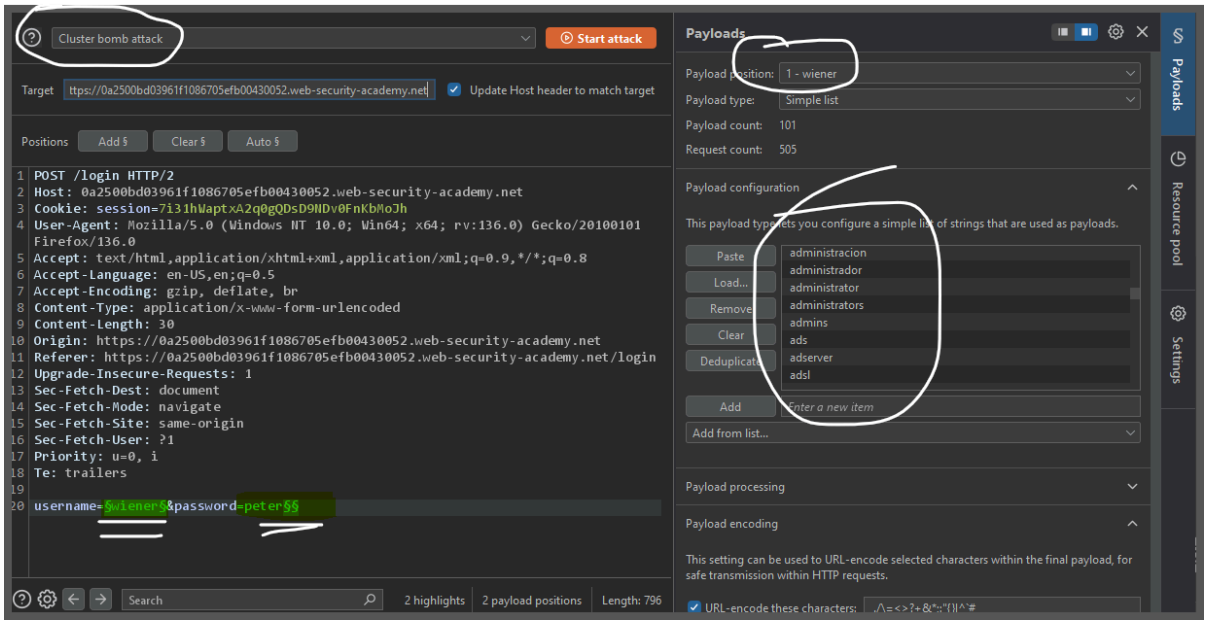
Lab: Username enumeration via account lock

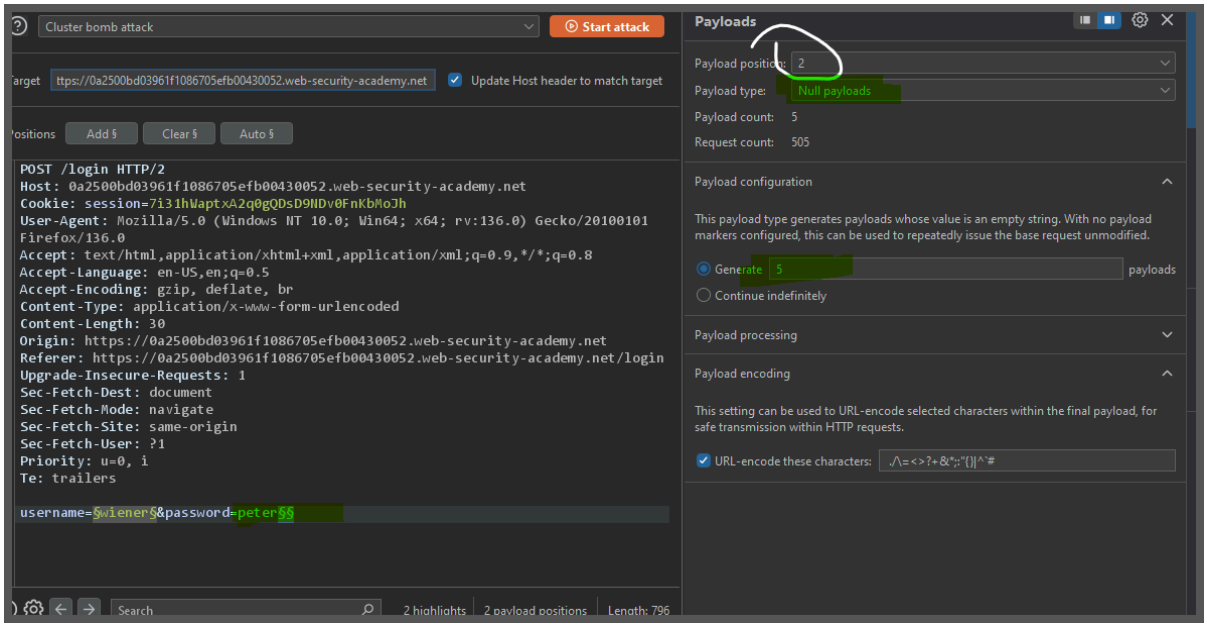
This lab is vulnerable to username enumeration. It uses account locking, but this contains a logic flaw. To solve the lab,

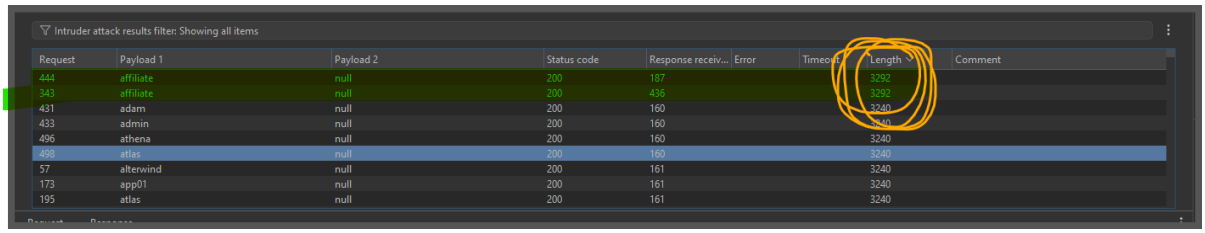
enumerate a valid username, brute-force this user's password, then access their account page.

- Candidate usernames
- Candidate passwords

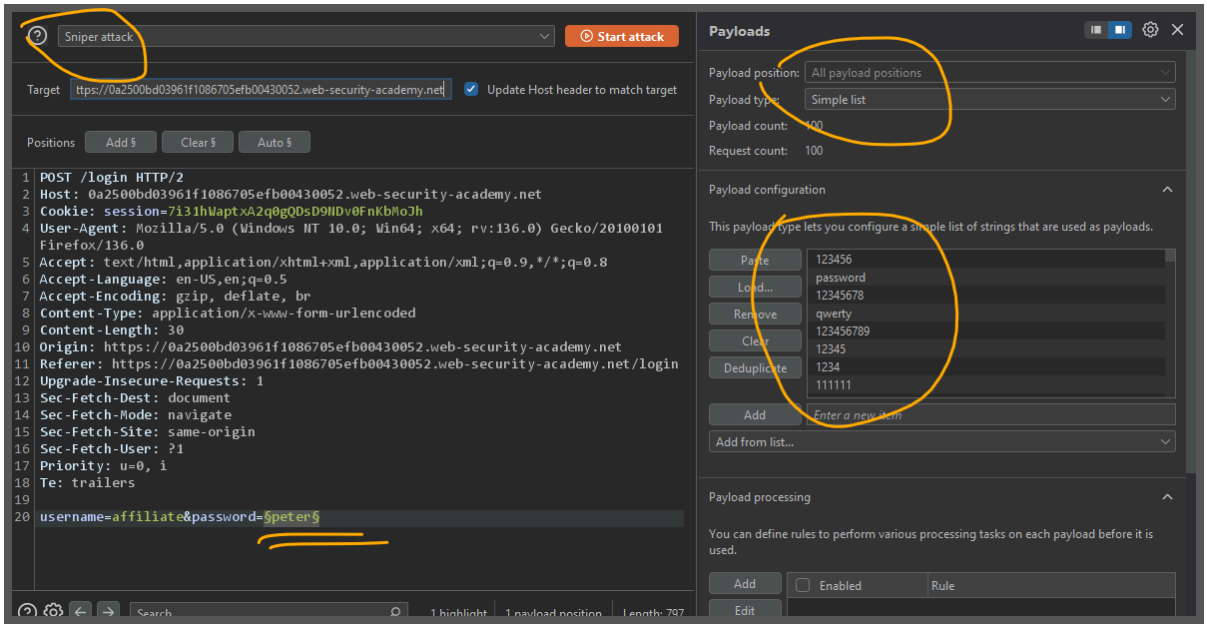
- in this lab if you enter the wrong username and brute forcing then no one is going to block
- but if you have valid username and brutefrcing then it is going to block
- so we have to
- make like this
- single username and five null or any password
- next username and finve nll or any
- it means for one user we are tring five password attemp
- the thing is if we got correct user name then we enter five wrong pasword they will block and with that we will come to know this is the username

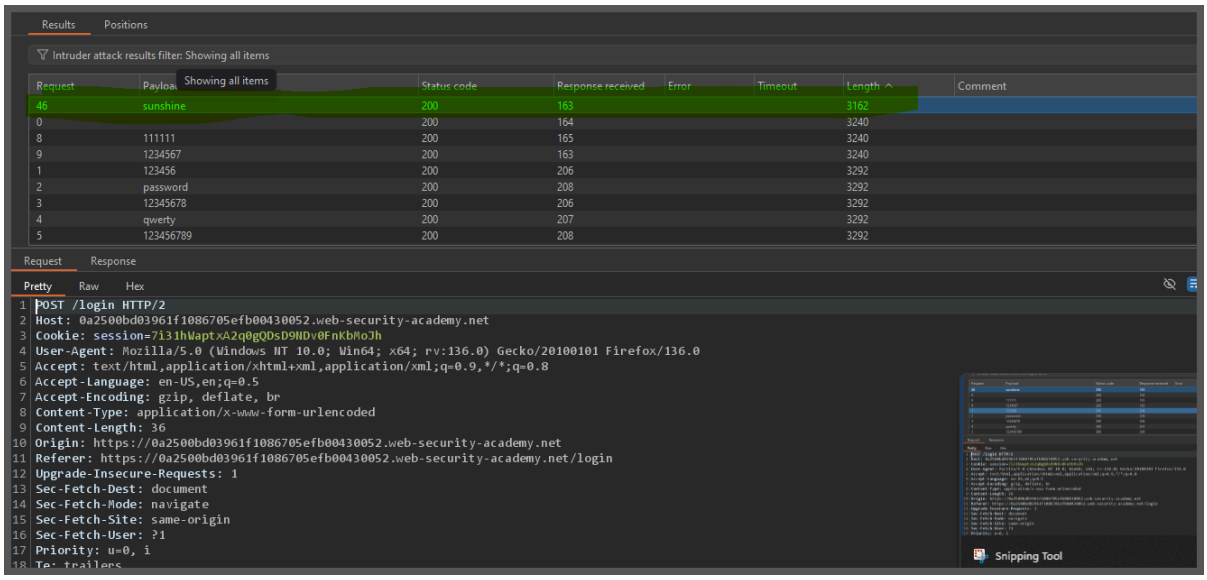






“now we got the username now lets find the password”



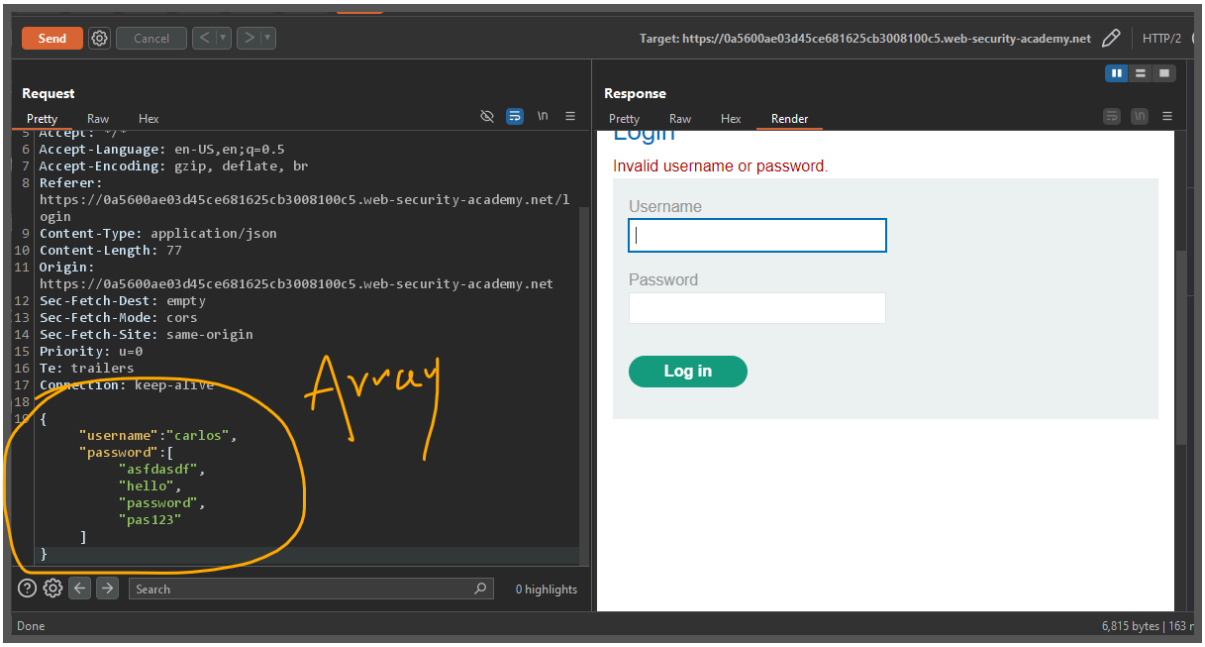


Lab: Broken brute-force protection, multiple credentials per request

This lab is vulnerable due to a logic flaw in its brute-force protection. To solve the lab, brute-force Carlos's password, then access his account page.

- Victim's username: `carlos`
- Candidate passwords

- in this lab if you sends the multiple request your going to block
- but seen in request credential are passed in differenct format
- let try array some multiple passwords in arryay

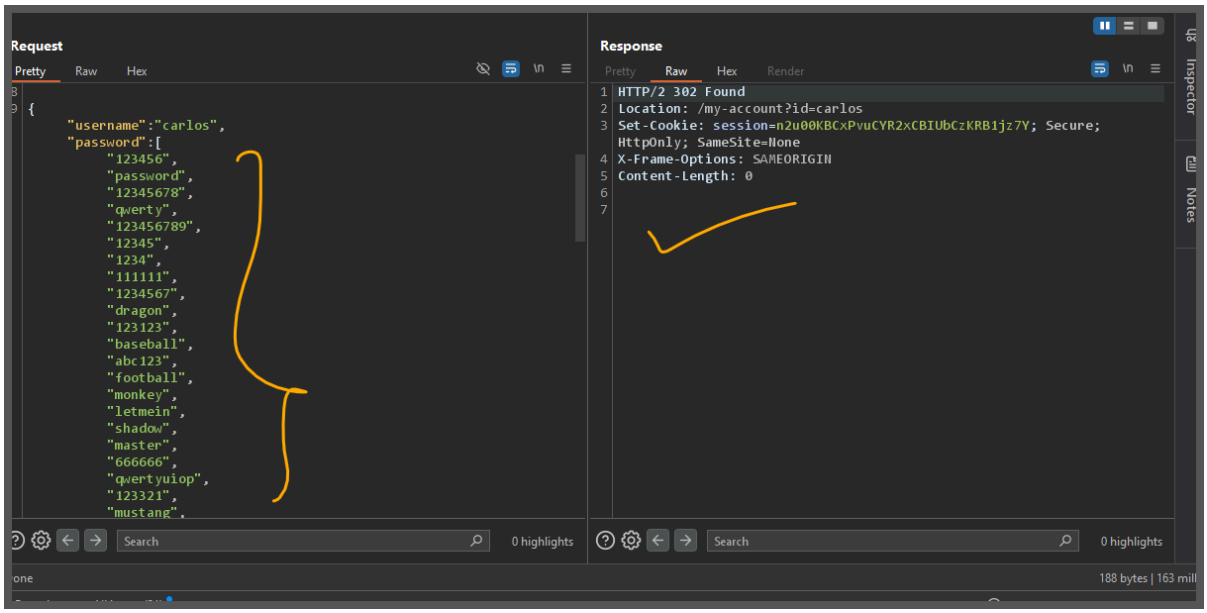


“we are able to store the multiple passwords in array so lets use the all passwrod “

“but wait do we have to do it maually adding that double queate “” and and quama

“no we have python”

```
print("[", end='')

with open('pass.txt', 'r') as f:
    lines = f.readlines()

for pwd in lines:
    print('"' + pwd.rstrip("\n") + '",',end='')

print('"random"]', end='')

#save that password file in same directory and name it pass.txt
```



“one of the passwrods woked but we dont know which one so get the reposne in browser of copyt the session id and paste it in browser”

Lab: 2FA simple bypass

This lab's two-factor authentication can be bypassed. You have already obtained a valid username and password, but do not have access to the user's 2FA verification code. To solve the lab, access Carlos's account page.

- Your credentials: `wiener:peter`
- Victim's credentials `carlos:montoya`

“

At times, the implementation of two-factor authentication is flawed to the point where it can be bypassed entirely.

If the user is first prompted to enter a password, and then prompted to enter a verification code on a separate page, the user is effectively in a "logged in" state before they have entered the verification code. In this case, it is worth testing to see if you can directly skip to "logged-in only" pages after completing the first authentication step. Occasionally, you will find that a

website doesn't actually check whether or not you completed the second step before loading the page.

“

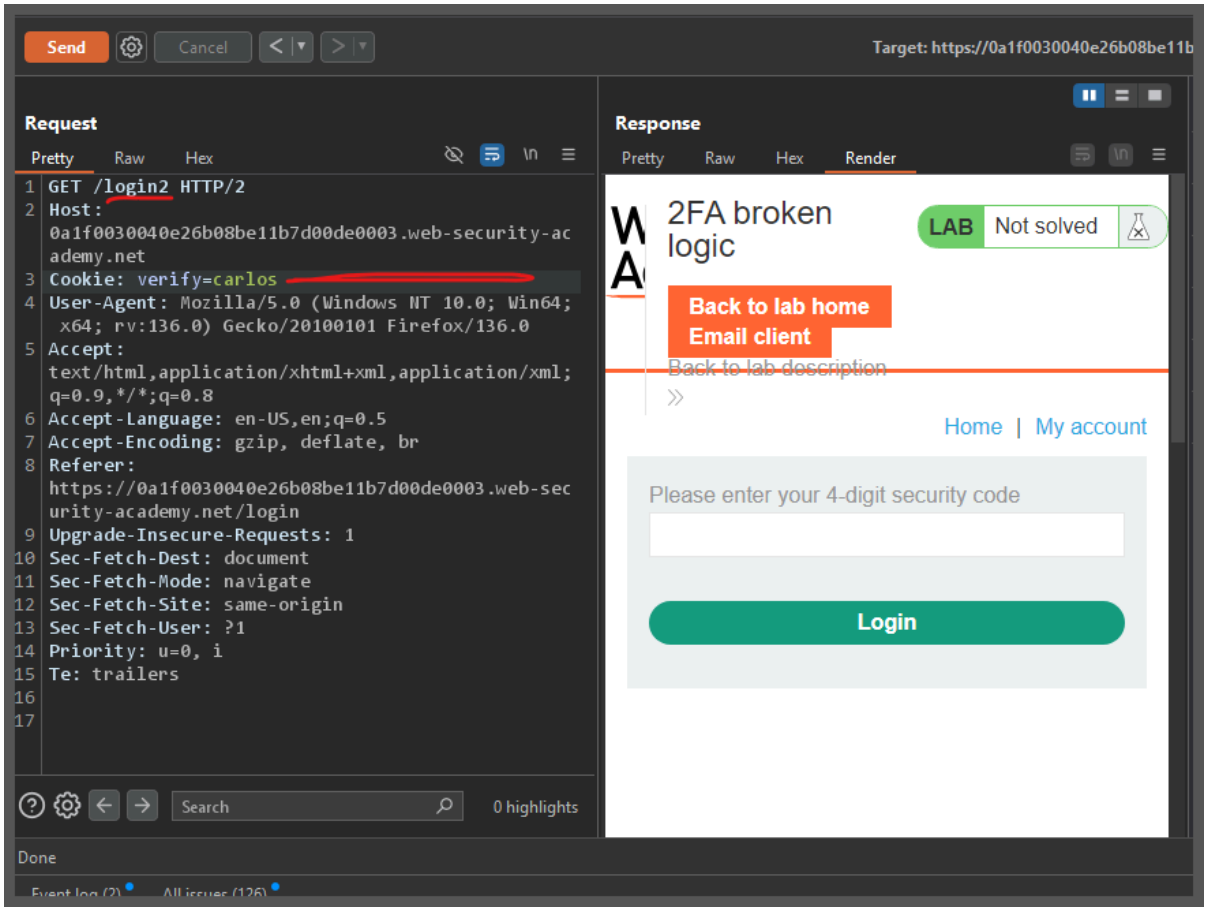
Simple hai bhai just carlos ka username aur password dal aur vo tere ko login2 pe leke jayga url me se login2 hata de and got to my account your logedin succesfully”

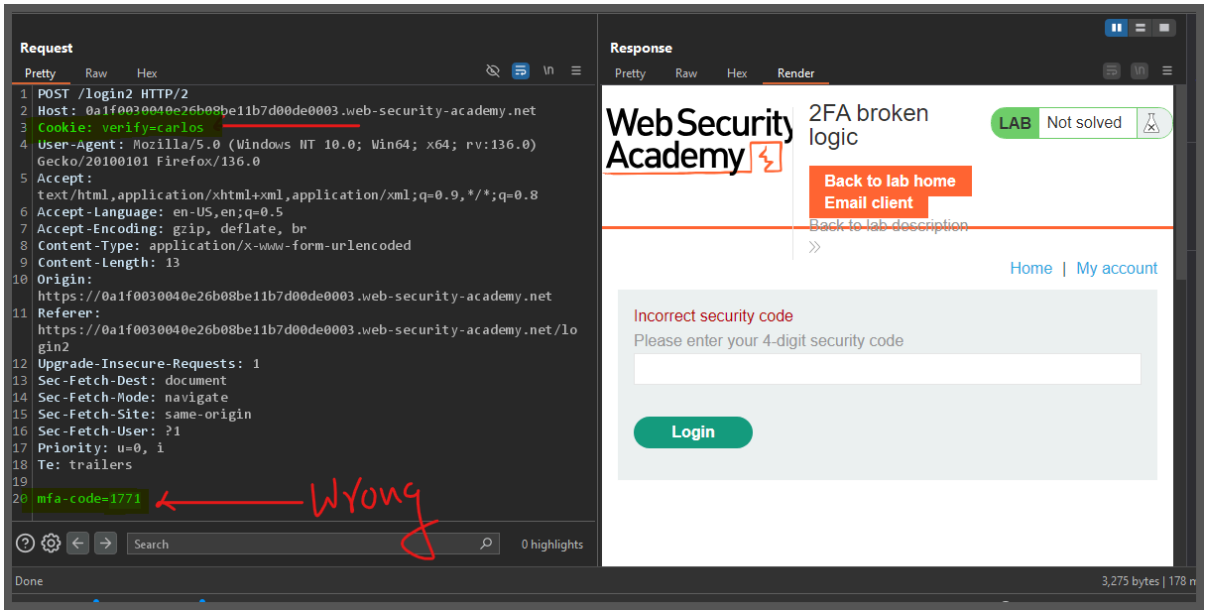
Lab: 2FA broken logic

This lab's two-factor authentication is vulnerable due to its flawed logic. To solve the lab, access Carlos's account page.

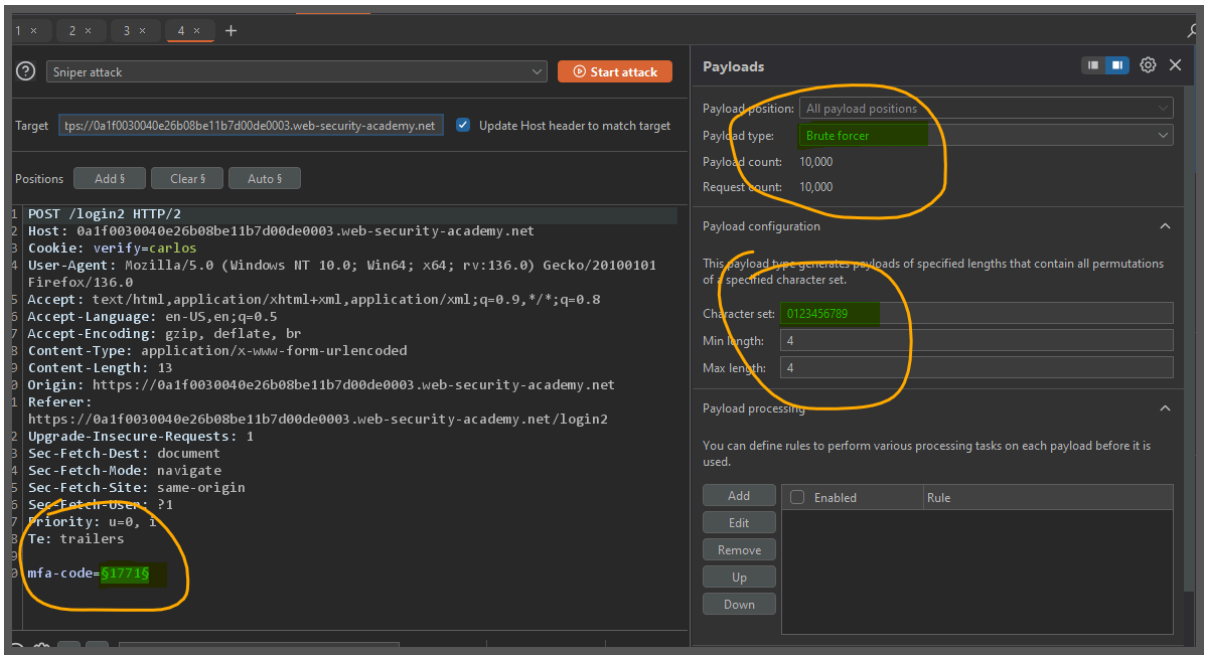
- Your credentials: `wiener:peter`
- Victim's username: `carlos`

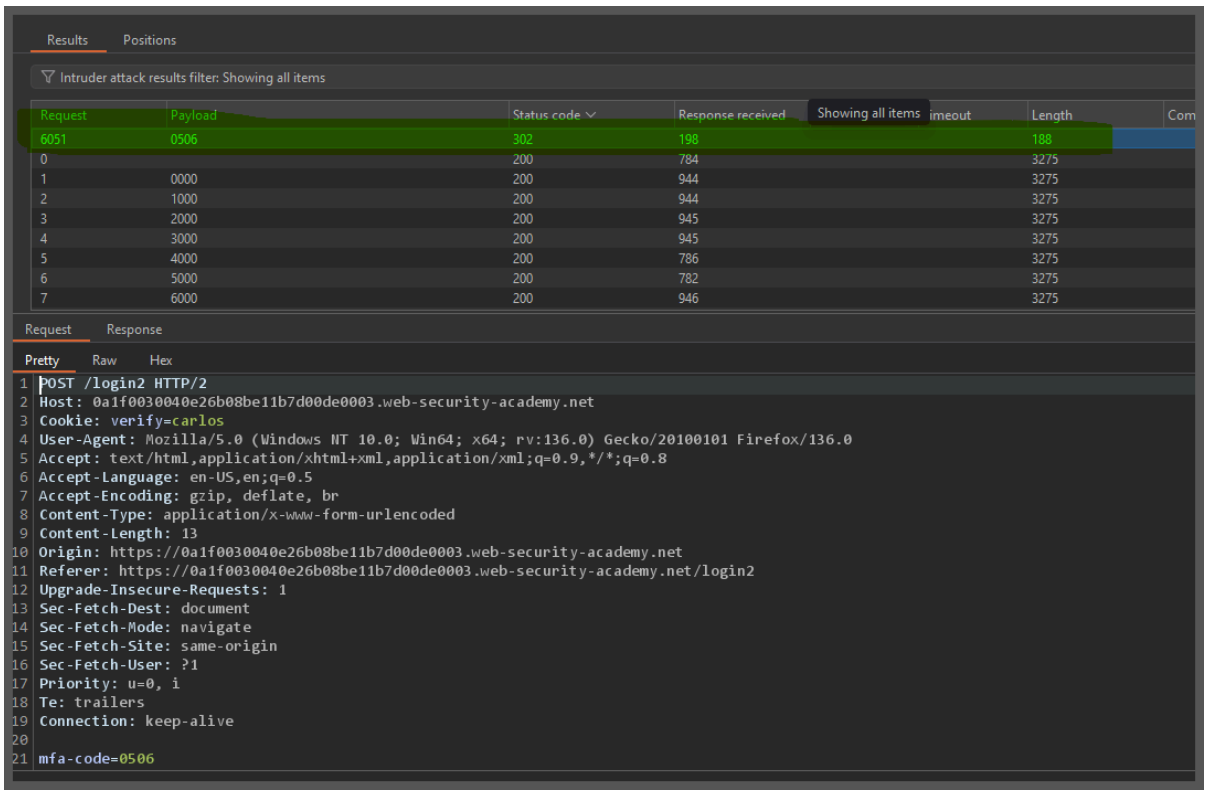
You also have access to the email server to receive your 2FA verification code.





now we can brute force the otp



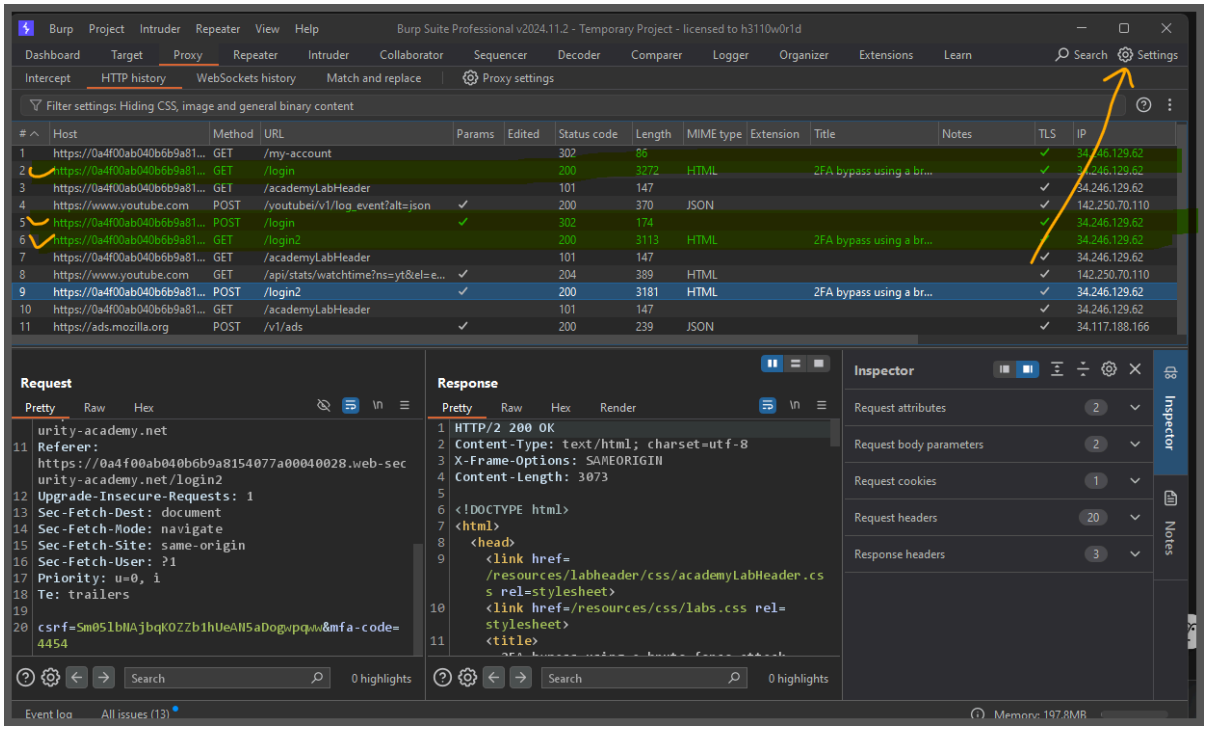


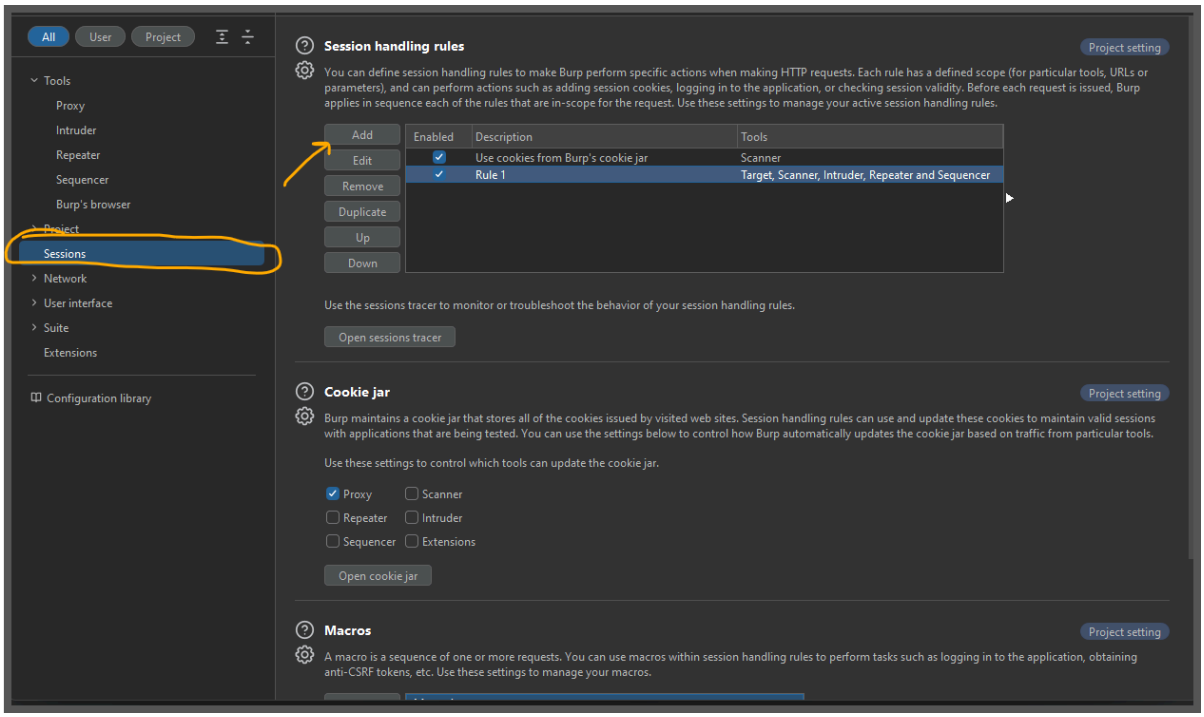
Lab: 2FA bypass using a brute-force attack

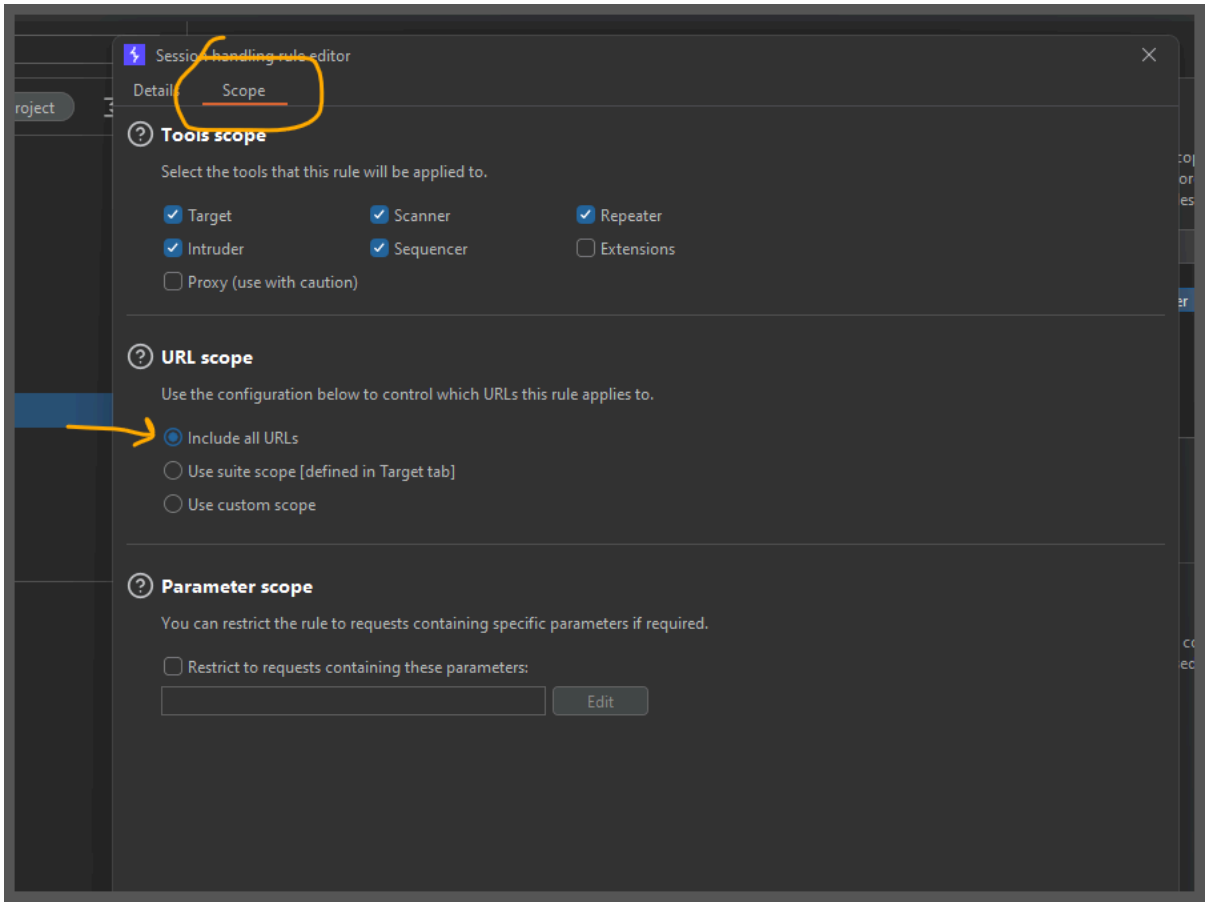
This lab's two-factor authentication is vulnerable to brute-forcing. You have already obtained a valid username and password,

but do not have access to the user's 2FA verification code. To solve the lab, brute-force the 2FA code and access Carlos's account page.

Victim's credentials: `carlos:montoya`




All
User
Project
Tools
Proxy
Intruder
Repeater
Sequencer
Burp's browser
Project
Sessions
Network
User interface
Suite
Extensions
Configuration library
Session handling rules
Project setting
You can define session handling rules to make Burp perform specific actions when making HTTP requests. Each rule has a defined scope (for particular tools, URLs or parameters), and can perform actions such as adding session cookies, logging in to the application, or checking session validity. Before each request is issued, Burp applies in sequence each of the rules that are in-scope for the request. Use these settings to manage your active session handling rules.
Add
Edit
Remove
Duplicate
Up
Down
Enabled
Description
Tools
Use cookies from Burp's cookie jar
Scanner
Rule 1
Target, Scanner, Intruder, Repeater and Sequencer
Use the sessions tracer to monitor or troubleshoot the behavior of your session handling rules.
Open sessions tracer
Cookie jar
Project setting
Burp maintains a cookie jar that stores all of the cookies issued by visited web sites. Session handling rules can use and update these cookies to maintain valid sessions with applications that are being tested. You can use the settings below to control how Burp automatically updates the cookie jar based on traffic from particular tools.
Use these settings to control which tools can update the cookie jar.
Proxy
Scanner
Repeater
Intruder
Sequencer
Extensions
Open cookie jar
Macros
Project setting
A macro is a sequence of one or more requests. You can use macros within session handling rules to perform tasks such as logging in to the application, obtaining anti-CSRF tokens, etc. Use these settings to manage your macros.


Session handling rule editor
Details
Scope
Tools scope
Select the tools that this rule will be applied to.
Target
Scanner
Repeater
Intruder
Sequencer
Extensions
Proxy (use with caution)
URL scope
Use the configuration below to control which URLs this rule applies to.
Include all URLs
Use suite scope [defined in Target tab]
Use custom scope
Parameter scope
You can restrict the rule to requests containing specific parameters if required.
Restrict to requests containing these parameters:
Edit

pending.....

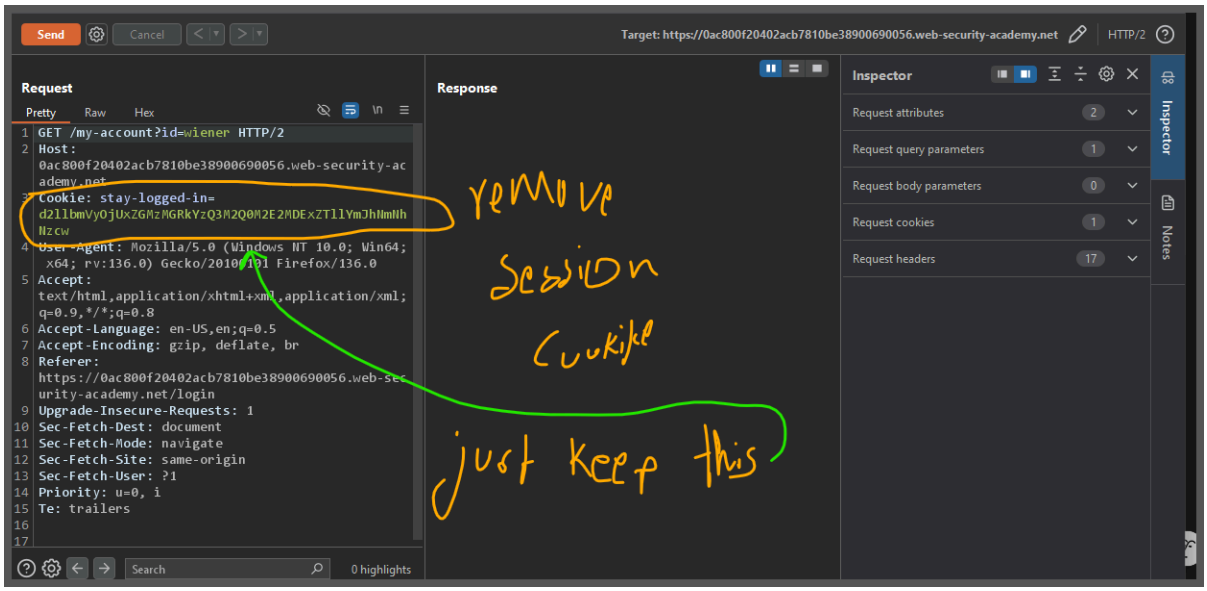
Lab: Brute-forcing a stay-logged-in cookie

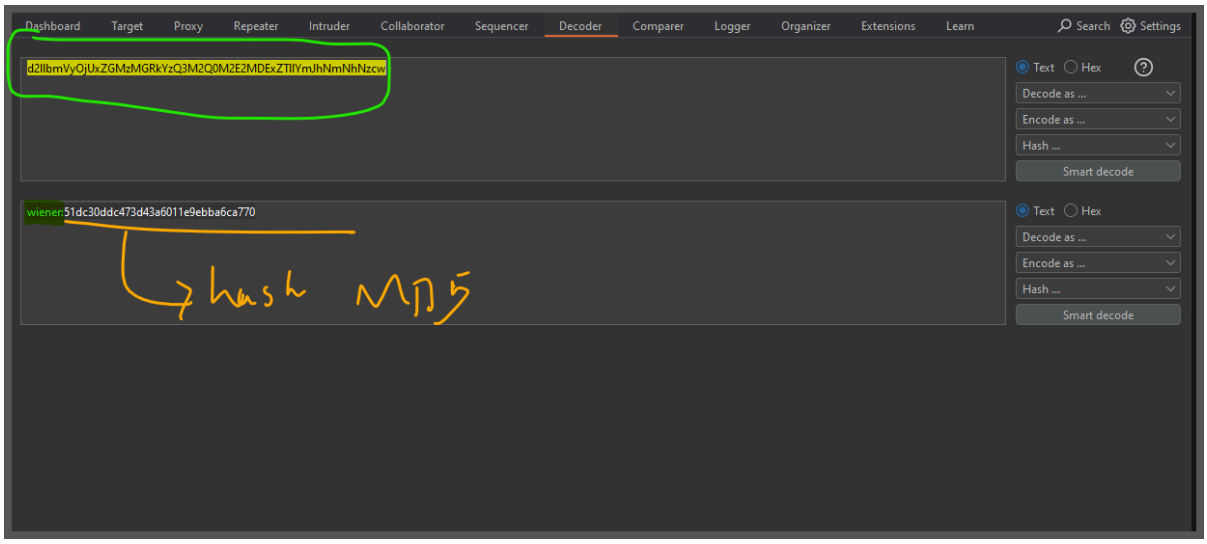
This lab allows users to stay logged in even after they close their browser session. The cookie used to provide this functionality is vulnerable to brute-forcing.

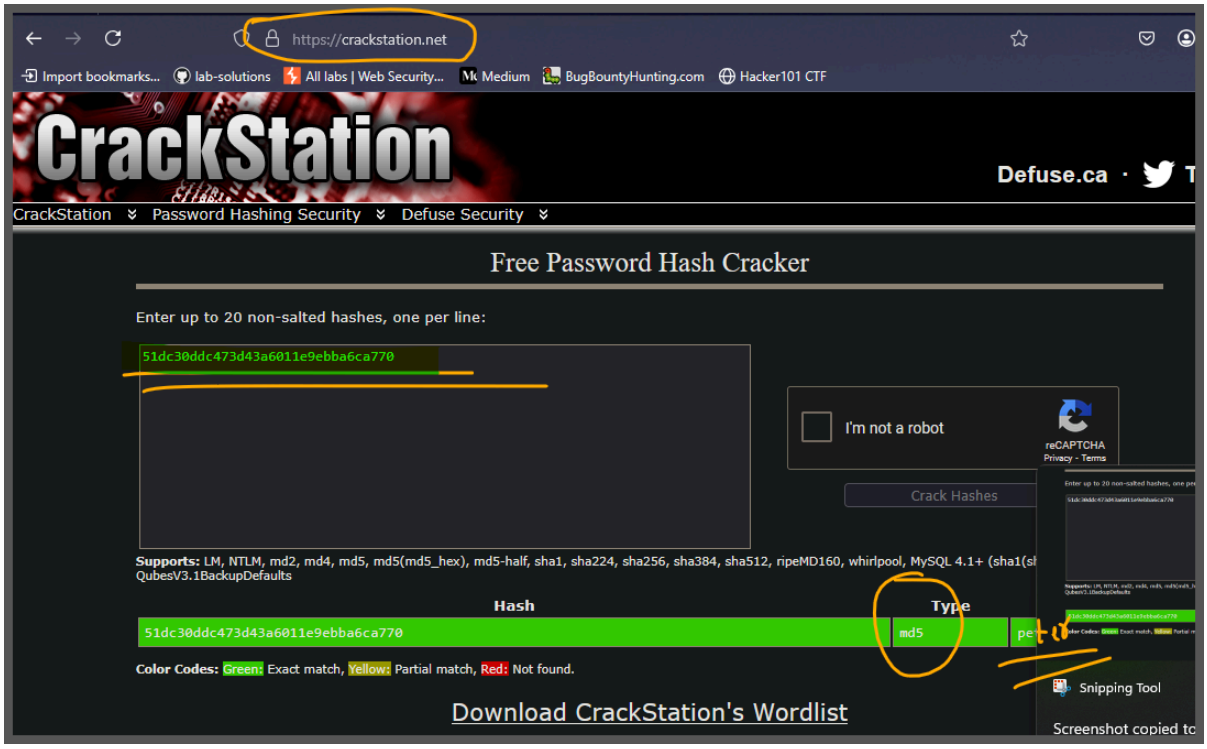
To solve the lab, brute-force Carlos's cookie to gain access to his **My account** page.

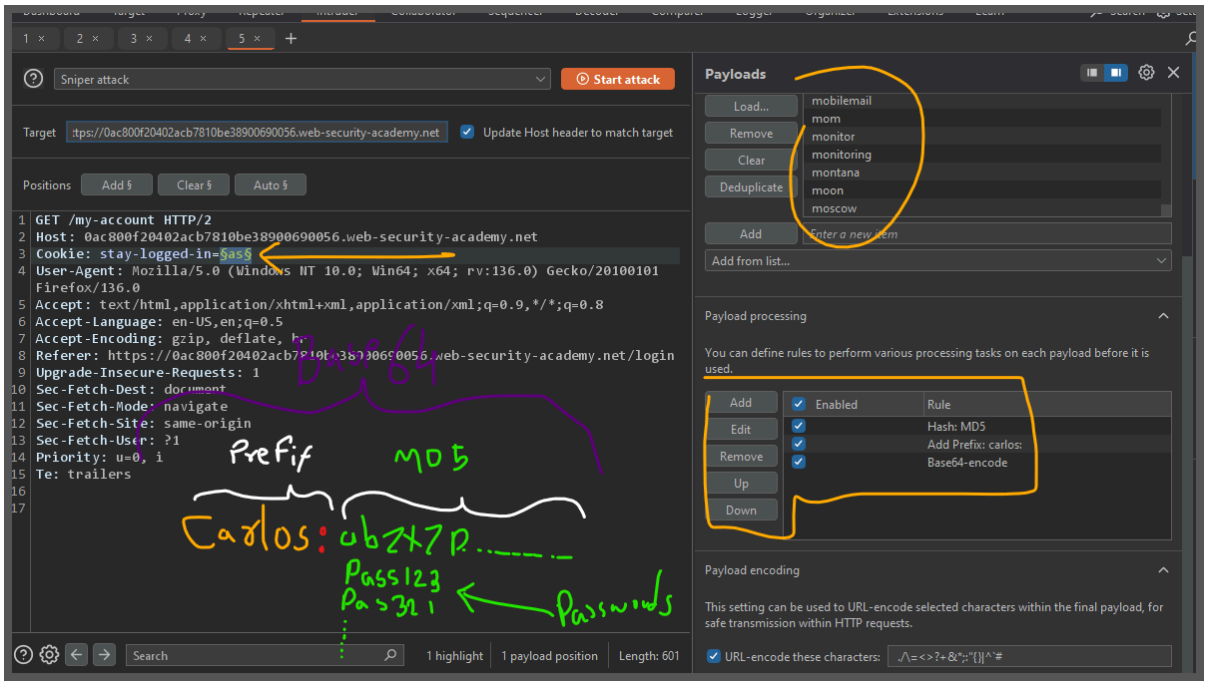
- Your credentials: `wiener:peter`
- Victim's username: `carlos`
- Candidate passwords

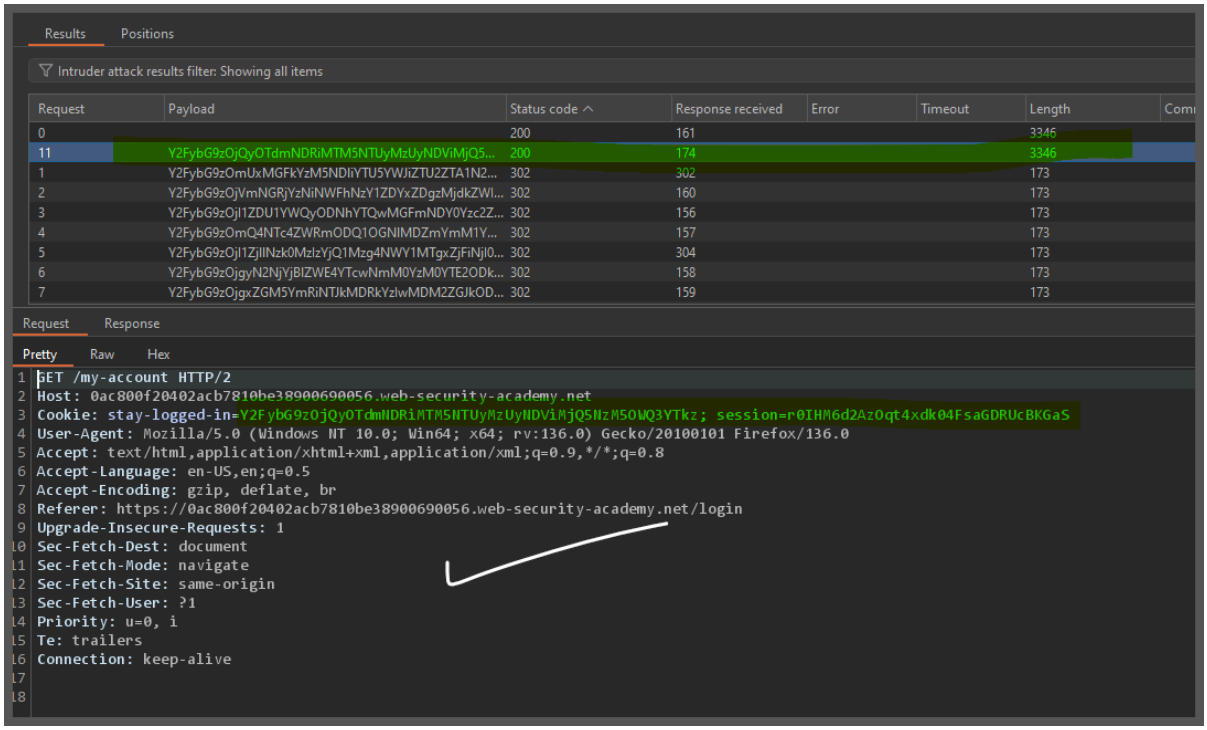
if you have stay logged in cookie then it is not going to chech session cookie so we have to brute furce it that staylogged in cookie




Dashboard
Target
Proxy
Repeater
Intruder
Collaborator
Sequencer
Decoder
Comparer
Logger
Organizer
Extensions
Learn
Search
Settings
d2lIbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw
Text
Hex
Decode as ...
Encode as ...
Hash ...
Smart decode
wiener:51dc30ddc473d43a6011e9ebba6ca770
Text
Hex
Decode as ...
Encode as ...
Hash ...
Smart decode
hash MD5


https://crackstation.net
Import bookmarks...
lab-solutions
All labs | Web Security...
Medium
BugBountyHunting.com
Hacker101 CTF
CrackStation
Defuse.ca
CrackStation
Password Hashing Security
Defuse Security
Free Password Hash Cracker
Enter up to 20 non-salted hashes, one per line:
51dc30ddc473d43a6011e9ebba6ca770
I'm not a robot
reCAPTCHA
Privacy - Terms
Crack Hashes
Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sh
QubesV3.1BackupDefaults
Hash
Type
51dc30ddc473d43a6011e9ebba6ca770
md5
Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.
Download CrackStation's Wordlist
Snipping Tool
Screenshot copied to

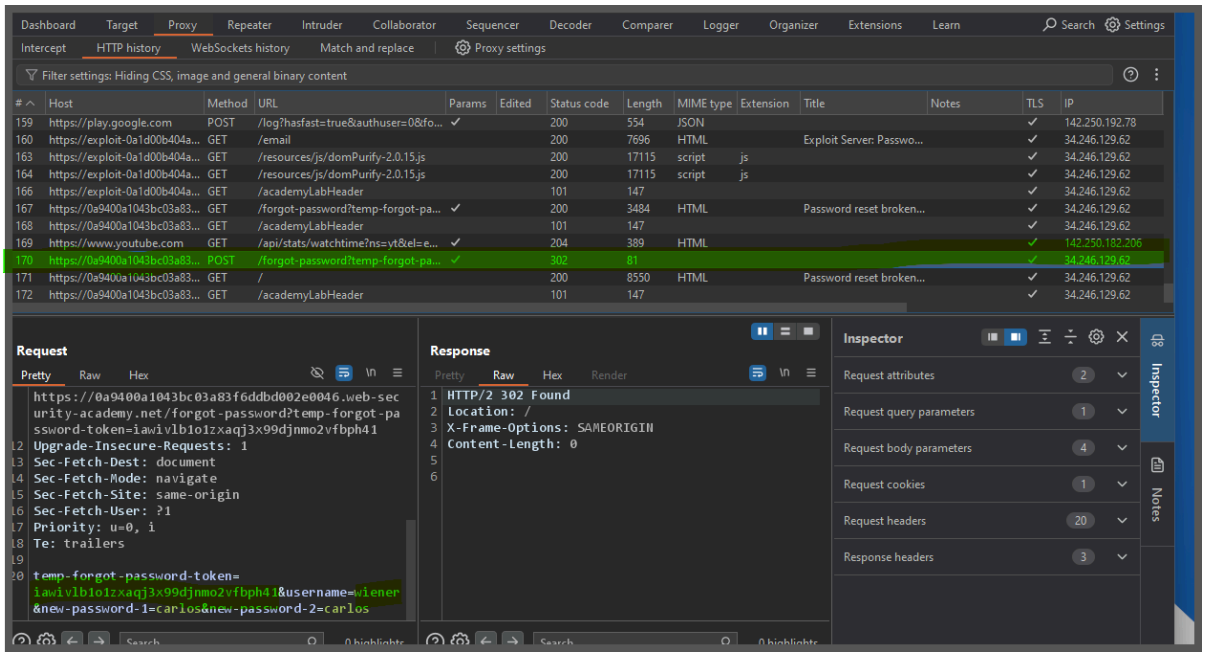


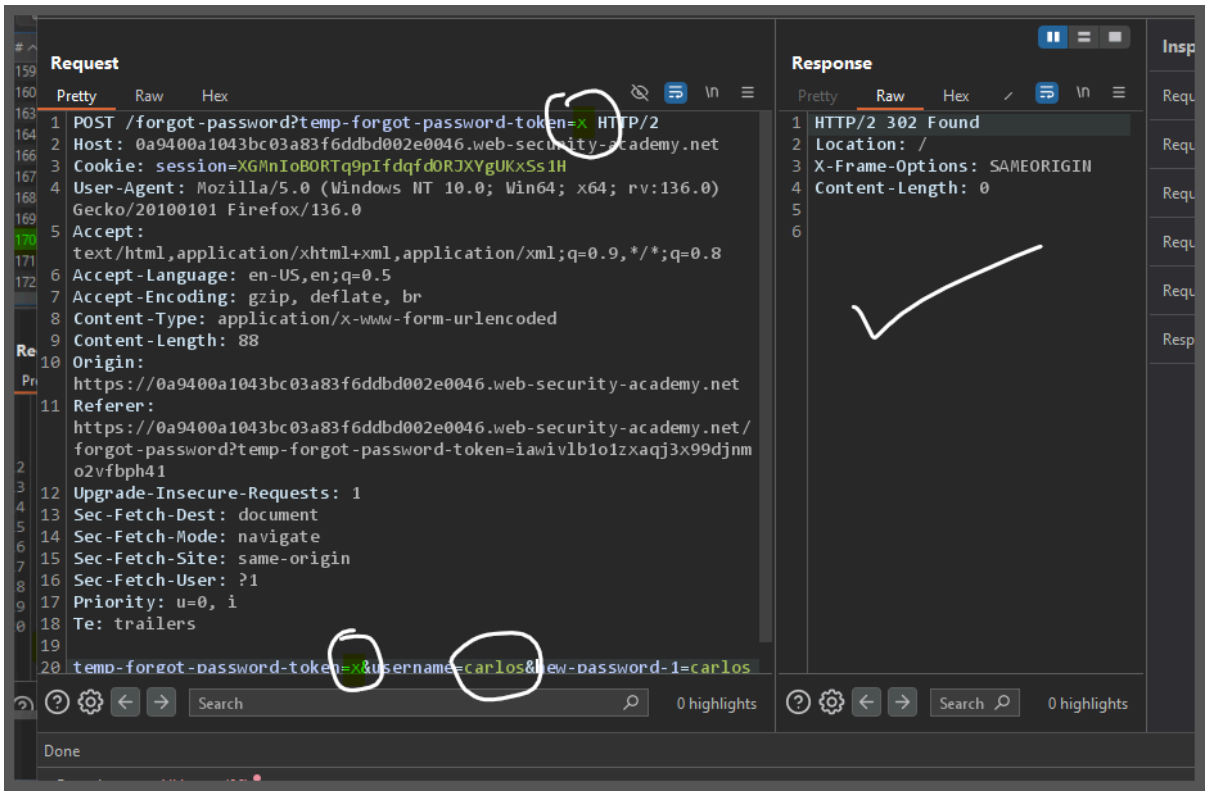


Lab: Password reset broken logic

This lab's password reset functionality is vulnerable. To solve the lab, reset Carlos's password then log in and access his "My account" page.

- Your credentials: `wiener:peter`
- Victim's username: `carlos`





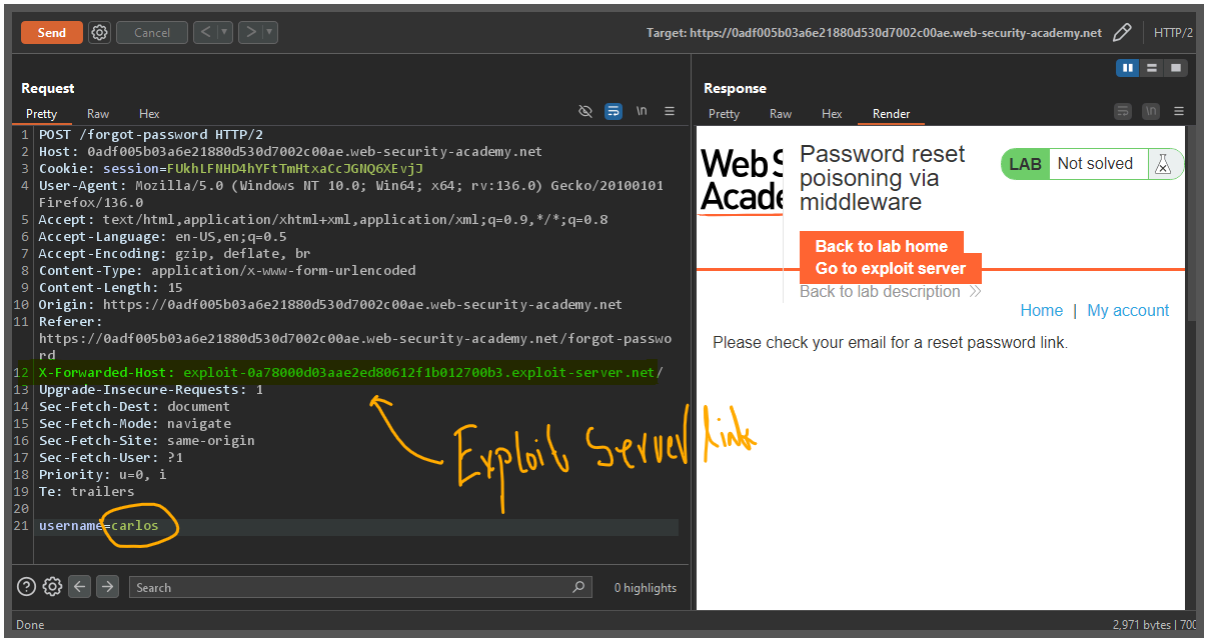
“its just checking the both the values as same or not

so we keept “X” in that both value and just cahnge the name as carlos”

Lab: Password reset poisoning via middleware

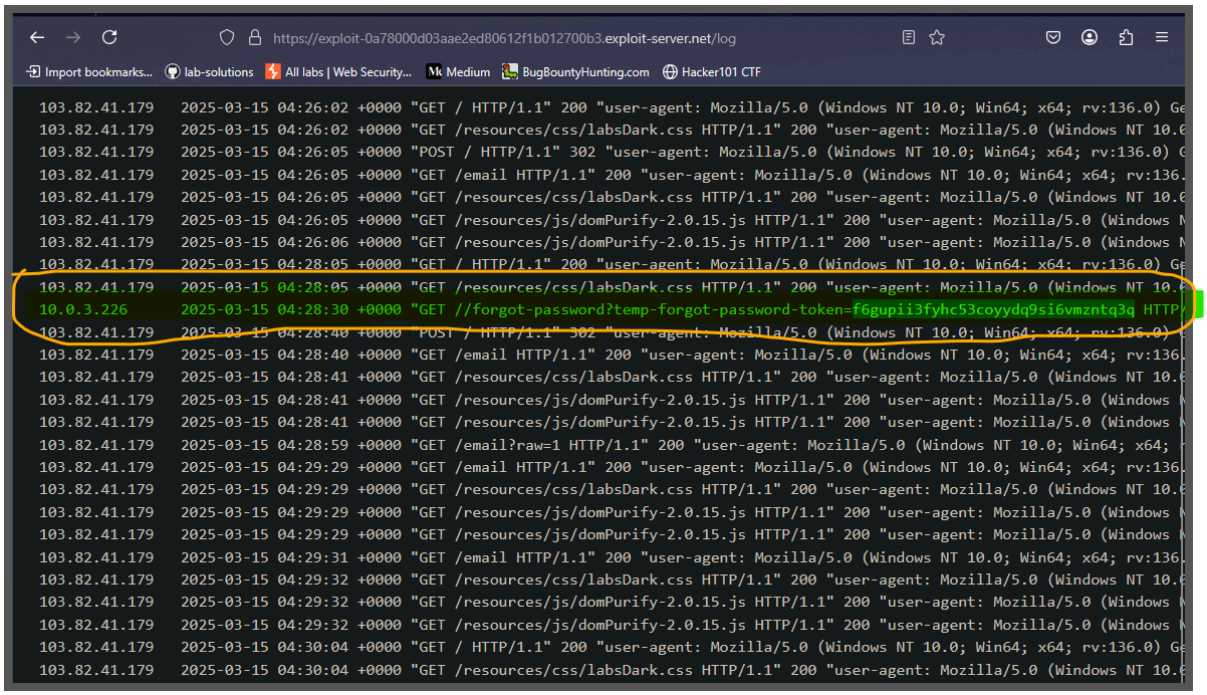
This lab is vulnerable to password reset poisoning. The user `carlos` will carelessly click on any links in emails that he receives. To solve the lab, log in to Carlos's account.

You can log in to your own account using the following credentials: `wiener:peter`. Any emails sent to this account can be read via the email client on the exploit server.



it will send the reset link on carlos and he will click

as he click we will get the clikc link on our server exploit server



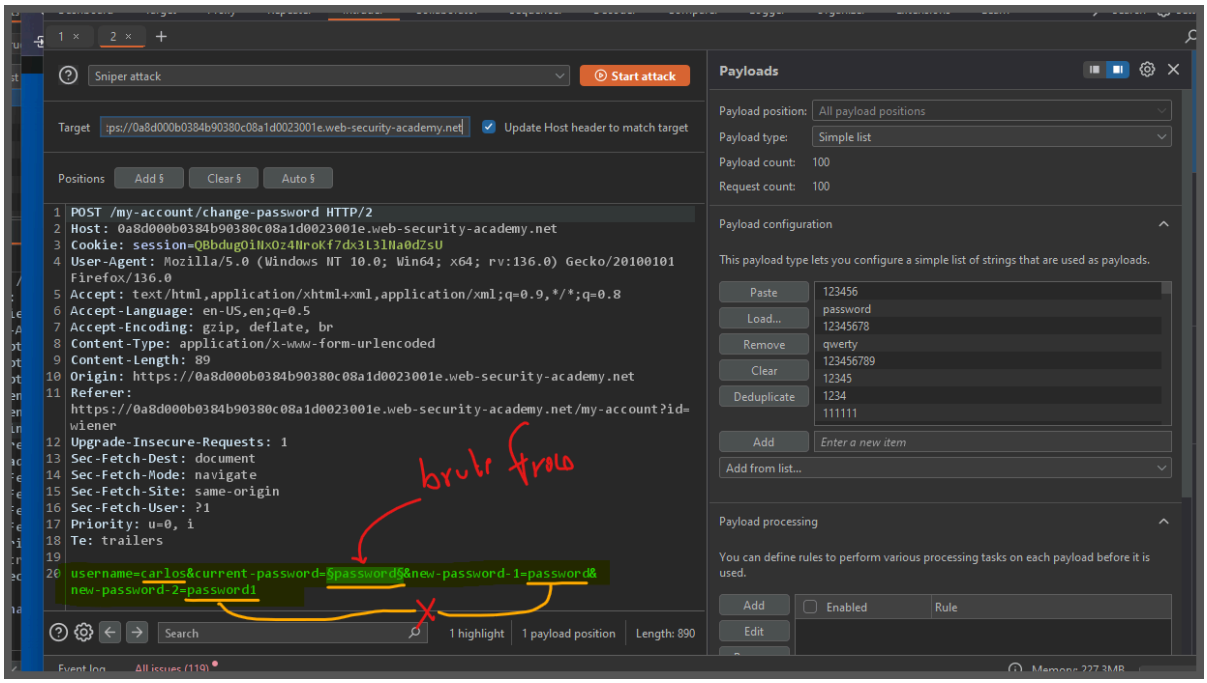
Lab: Password brute-force via password change

This lab's password change functionality makes it vulnerable to brute-force attacks. To solve the lab, use the list of candidate passwords to brute-force Carlos's account and access his "My account" page.

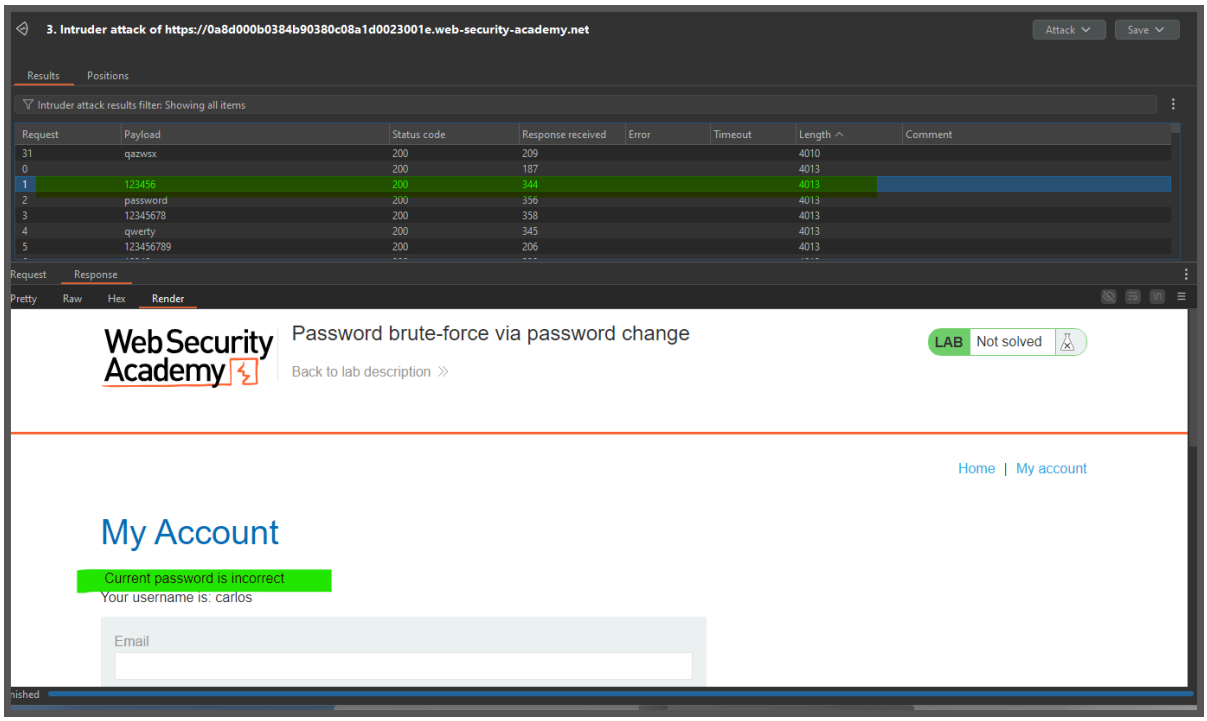
- Your credentials: `wiener:peter`
- Victim's username: `carlos`
- Candidate passwords

- if we trying with wrong password it will logout and it will lock use for one minut
- but what if we enter mismatch password in both filed
- like we have to enter twice an new passwod
- if current password is currect && twice password are mismatch == o/p == password do not match

- if current password is wrong && twice passwod are mismatch == o/p == wrong password
- now we can brute force the current passwrod



if password is wrong and that both password doesnt match



if password is correct and that both password do not match