ETHICAL HACKING
LAB SERIES

Lab 14: Understanding SQL Commands & Injections

Material in this Lab Aligns to the Following Certification Domains/Objectives	
Certified Ethical Hacking (CEH) Domains	SANS GPEN Objectives
14: SQL Injection	14: Reconnaissance

Document Version: 2016-03-09

Contents

Introduction

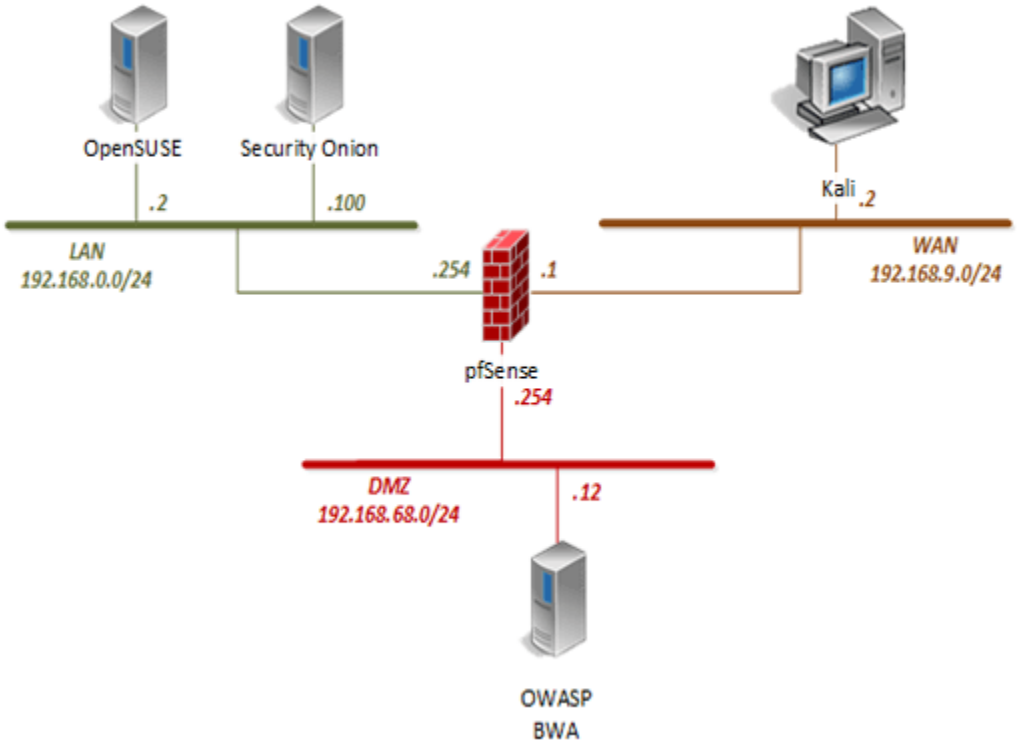
SQL (Structured Query Language) is used by many databases as a language to query, insert and delete elements. This lab demonstrates how to build, query, and delete elements in a database and how these skills can be used to attack a database.

Objective

In this lab, you will be conducting ethical hacking practices using various tools. You will be performing the following tasks:

1. Basic SQL Commands
2. Querying with SQL
3. Deleting with SQL
4. SQL Injection

Pod Topology



Lab Settings

The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

Virtual Machine	IP Address	Account (if needed)	Password (if needed)
Kali Linux	192.168.9.2	root	toor
pfSense	192.168.0.254	admin	pfsense
OWASP Broken Web App	192.168.68.12	root	owaspbwa
OpenSUSE	192.168.0.2	osboxes	osboxes.org
Security Onion	n/a	ndg	password123

1 Basic SQL Commands

1. Navigate to the *topology* page and click on the **Kali** VM icon.
2. Click anywhere within the *Kali* console window and press **Enter** to display the login prompt.
3. Enter `root` as the *username*. Click **Next**.
4. Enter `toor` as the *password*. Click **Sign In**.
5. Open the *Terminal* by clicking on the **Terminal** icon located on the left panel.

6. In the new *Terminal* window, start the *mysql* service by typing the command below followed by pressing the **Enter** key.

```
service mysql start
```

```
root@Kali2:~# service mysql start
root@Kali2:~#
```

7. Once started, enter the command below to log into the mysql database.

```
mysql -u root
```

```
root@Kali2:~# mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 5.5.46-0+deb8u1 (Debian)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```


8. Once logged in, view the available databases by entering the command below.

```
show databases;
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)

mysql>
```

9. Notice the predefined databases. Create a new database named **test**.

```
create database test;
```

```
mysql> create database test;
Query OK, 1 row affected (0.00 sec)

mysql>
```

10. Confirm the new *test* database appears.

```
show databases;
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.00 sec)

mysql>
```

11. Use the new database by entering the command below.

```
use test;
```

```
mysql> use test;
Database changed
mysql>
```


12. View if there are any tables in the *test* database.

```
show tables;
```

```
mysql> show tables;
Empty set (0.00 sec)

mysql>
```

13. Create a new table within the *test* database for users and populate it.

```
create table users (name varchar (30), account integer, balance decimal
(10,2));
```

```
mysql> create table users (name varchar (30), account integer, balance decimal (
10,2));
Query OK, 0 rows affected (0.02 sec)

mysql>
```

14. Show the tables and confirm a new *users* table appears.

```
show tables;
```

```
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| users          |
+----------------+
1 row in set (0.00 sec)

mysql>
```

15. Add some data into the *users* table.

```
insert into users values ('John', 123, 10.00);
```

```
mysql> insert into users values ('John', 123, 10.00);
Query OK, 1 row affected (0.00 sec)

mysql>
```


16. View the data in the *users* table.

```
select * from users;
```

```
mysql> select * from users;
+------+---------+---------+
| name | account | balance |
+------+---------+---------+
| John |     123 |   10.00 |
+------+---------+---------+
1 row in set (0.00 sec)

mysql>
```

17. Populate the *users* table once more with a different customer.

```
insert into users values ('Joe', 456, 20.00);
```

```
mysql> insert into users values ('Joe', 456, 20.00);
Query OK, 1 row affected (0.00 sec)

mysql>
```

18. View the data in the *users* table.

```
select * from users;
```

19. Create another table name **personal** and populate it.

```
create table personal (name varchar(30), address varchar(30), city
varchar(20), telephone integer);
```

```
mysql> create table personal (name varchar(30), address varchar(30), city varcha
r(20), telephone integer);
Query OK, 0 rows affected (0.00 sec)

mysql>
```

20. Verify the new table exists.

```
show tables;
```

```
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| personal       |
| users          |
+----------------+
2 rows in set (0.00 sec)

mysql>
```


21. Add some data into the *personal* table.

```
insert into personal values('John', '1313 Mockingbird Lane', 'Mockingbird
Heights', 3105552368);
```

```
mysql> insert into personal values('John', '1313 Mockingbird Lane', 'Mockingbird
 Heights', 3105552368);
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql>
```

22. Insert additional data into the personal table.

```
insert into personal values('Joe', '1313 Cemetery Lane', 'Greenbrier',
1313131313);
```

```
mysql> insert into personal values('Joe', '1313 Cemetery Lane', 'Greenbrier', 13
13131313);
Query OK, 1 row affected (0.00 sec)

mysql>
```

23. Analyze the data from the personal data.

```
select * from personal;
```

```
mysql> select * from personal;
+------+----------------------+--------------------+------------+
| name | address              | city               | telephone  |
+------+----------------------+--------------------+------------+
| John | 1313 Mockingbird Lane | Mockingbird Heights | 2147483647 |
| Joe  | 1313 Cemetery Lane    | Greenbrier         | 1313131313 |
+------+----------------------+--------------------+------------+
2 rows in set (0.00 sec)

mysql>
```


2 Querying with SQL

1. Using the test database, query the names of the users and balance from the *users* table.

```
select name, balance from users;
```

```
mysql> select name, balance from users;
+------+---------+
| name | balance |
+------+---------+
| John |   10.00 |
| Joe  |   20.00 |
+------+---------+
2 rows in set (0.01 sec)

mysql>
```

2. Query the names of the users and telephone numbers from the *personal* table.

```
select name, telephone from personal;
```

```
mysql> select name, telephone from personal;
+------+------------+
| name | telephone  |
+------+------------+
| John | 2147483647 |
| Joe  | 1313131313 |
+------+------------+
2 rows in set (0.00 sec)

mysql>
```

3. Retrieve data across both tables: *users* and *personal*.

```
select users.name, users.balance, personal.telephone from users join
personal where users.name=personal.name;
```

```
mysql> select users.name, users.balance, personal.telephone from users join pers
onal where users.name=personal.name;
+------+---------+------------+
| name | balance | telephone  |
+------+---------+------------+
| John |   10.00 | 2147483647 |
| Joe  |   20.00 | 1313131313 |
+------+---------+------------+
2 rows in set (0.00 sec)

mysql>
```


3 Deleting with SQL

1. Enter the command below to delete a row of data from the *personal* table.

```
delete from personal where name='Joe';
```

```
mysql> delete from personal where name='Joe';
Query OK, 1 row affected (0.00 sec)

mysql>
```

2. View the deleted changes.

```
select * from personal;
```

```
mysql> select * from personal;
+------+----------------------+--------------------+------------+
| name | address              | city               | telephone  |
+------+----------------------+--------------------+------------+
| John | 1313 Mockingbird Lane | Mockingbird Heights | 2147483647 |
+------+----------------------+--------------------+------------+
1 row in set (0.00 sec)

mysql>
```

3. Delete the entire **personal** table.

```
drop table personal;
```

```
mysql> drop table personal;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

4. View all the available tables.

```
show tables;
```

5. Delete the entire **test** database.

```
drop database test;
```

```
mysql> drop database test;
Query OK, 1 row affected (0.01 sec)

mysql>
```

6. Show all databases.

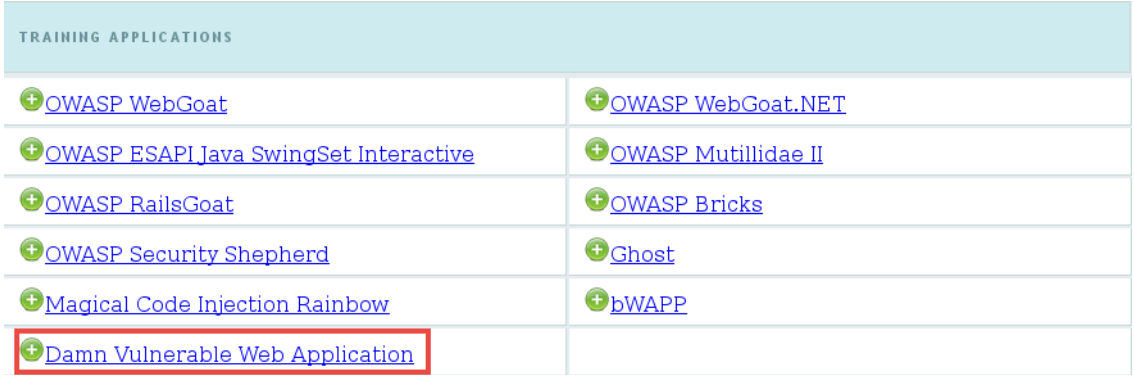
```
show databases;
```


4 SQL Injection

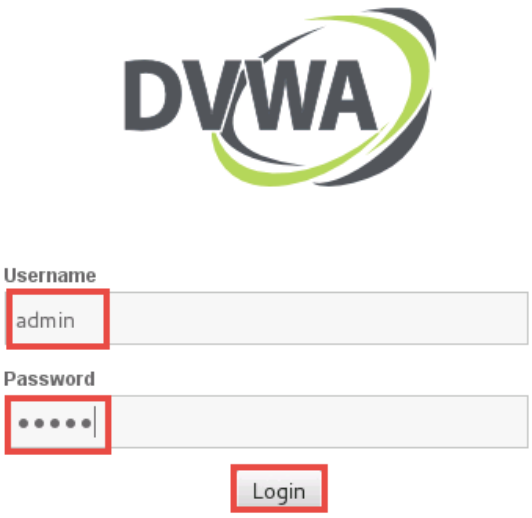
1. Open the *Iceweasel* browser by clicking on the **Iceweasel** icon located on the left panel.

2. In the *Iceweasel* browser, type `192.168.68.12` into the address field and press the **Enter** key.

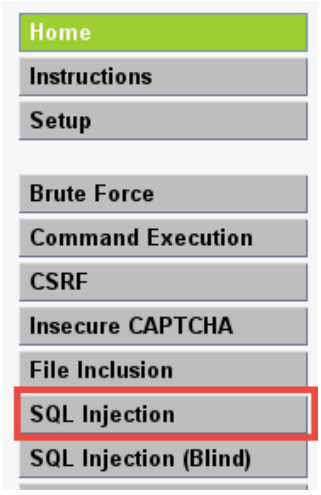
3. Scroll down to the *Training Applications* pane and click on the **Damn Vulnerable Web Application** link.



4. On the *DVWA* login page, login using **`admin`** as the *username* and **`admin`** as the *password*. Click **Login**.

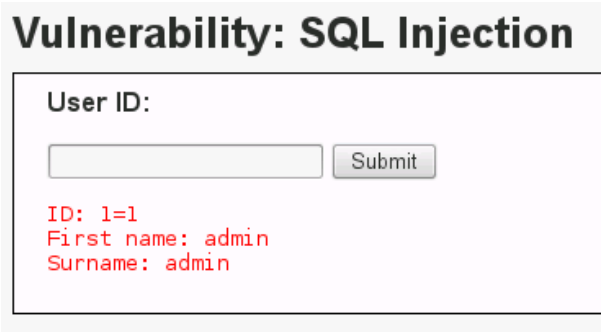


5. On the *DVWA* homepage, click on **SQL Injection** button located in the left pane.



6. Test if the application is vulnerable to *SQL* injection by trying a simple test using a true statement. Type the command below into the *User ID* text field followed by clicking the **Submit** button.

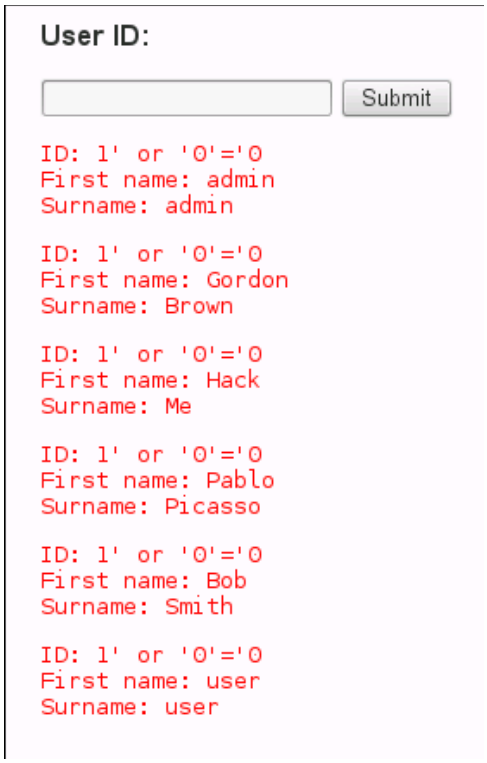
```
1=1
```



Notice what happened was that a query was sent to the database that executed the following: *select first_name,surname from "some table" where user_id=1*

7. Display all records that are false (empty) and all records that are true (not empty). Enter the command below into the *User ID* field followed by clicking **Submit**.

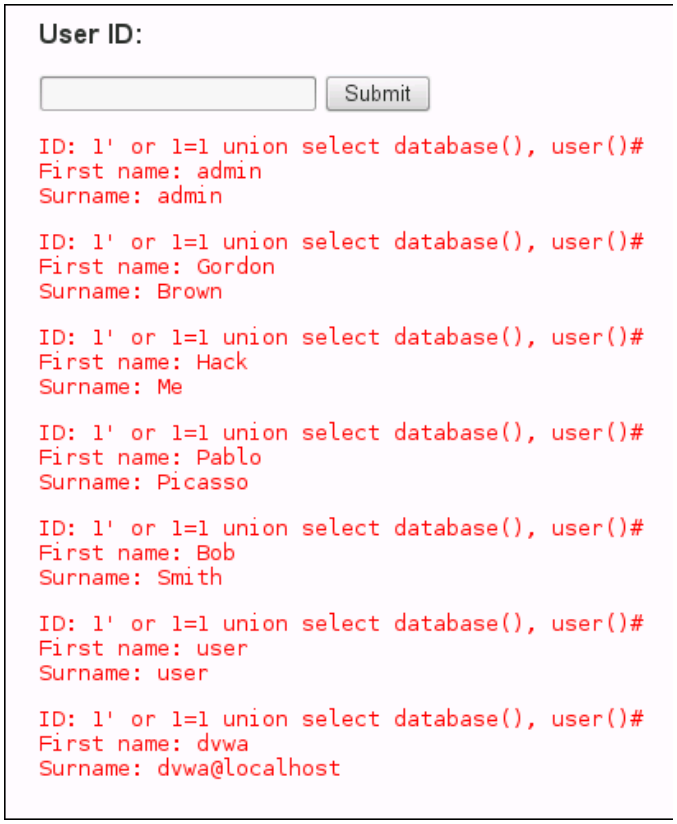
```
1' or '0'='0
```



Users have now been dumped into the database. The following query was executed: *select first_name,surname from "some table" where user_id = 1' or '0'='0';*

8. Attempt to pull database information and the user of the database. Enter the command below into the *User ID* field, click **Submit**.

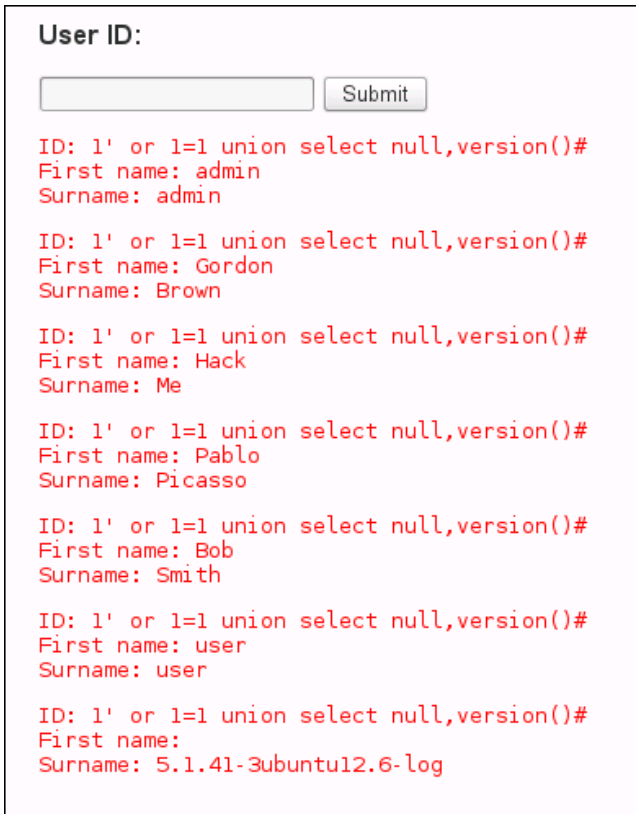
```
1' or 1=1 union select database(), user()#
```



Notice the *database()* command returns the database name of *dvwa* and its user *dvwa@localhost*. The union statement is similar to "join" except that it links 2 select statements together and the # character ends the statement.

9. Try to pull the database version by entering the command below into the *User ID* field, click **Submit**.

```
1’ or 1=1 union select null,version()#
```

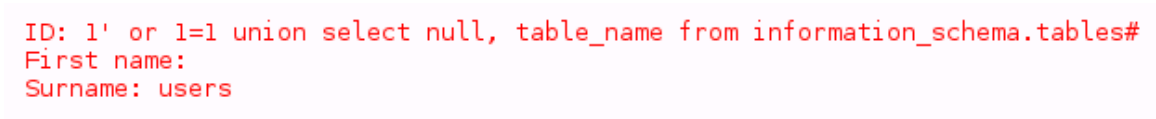


Notice null was used as a placeholder and issued the *version()* command. Given the output, it appears the OS is running on *5.1.41-3ubuntu12.6*.

10. Enter the command below into the *User ID* field to identify the tables in the database.

```
1’ or 1=1 union select null, table_name from information_schema.tables#
```

11. Scroll down and identify the table being a *users* table.



In the query, *null* was a placeholder again and the *table_name* is something that exists in the main part of the database build called the information schema.

12. Attempt to see if any password fields are associated with the *users* table. Enter the command below into the *User ID* field and click **Submit**.

```
1’ or 1=1 union select user, password from users#
```

Notice towards the bottom, hashes are given out from the query.

13. Close the **Kali** PC viewer.

