Web Application Penetration Testing

[SQL Injection, Cross Site Scripting, wireshark]

Submitted By

Thanusree sasikumar

Jasmeen kaur arora

Nikita singhal

Table of Contents

LIST OF FIGURES

16	Wireshark packet capture showing an HTTP POST request with login credentials	19
17	Wireshark capture showing HTTP traffic between a device and various online resources.	20
18	Wireshark packet capture showing network traffic, including HTTP requests, DNS queries, and TCP connections.	21
19	Successful SQL injection attack on a web application, resulting in unauthorized access to the database and potential data breach	22
20	Results from a web application penetration testing session, focusing on cross-site scripting (XSS) vulnerabilities	23
21	Wireshark capture showing network traffic during a web application penetration testing session, focusing on SQL injection and XSS vulnerabilities	25

CHAPTER-1

1.1 ABSTRACT:

This report details a comprehensive web application penetration test conducted on the website http://testphp.vulnweb.com, focusing on identifying and exploiting SQL Injection (SQLi) and Cross-Site Scripting (XSS) vulnerabilities. Using tools such as SQLMap, XSSer, and Wireshark, the assessment revealed critical security weaknesses within the application.

The testing methodology began with reconnaissance, followed by targeted SQLi and XSS exploitation using SQLMap and XSSer, respectively. SQLMap enabled unauthorised access to the database, extracting sensitive information, while XSSer demonstrated the potential for script injection, posing risks such as session hijacking and defacement. Wireshark was employed to capture and analyse network traffic, providing deeper insights into the attacks.

The findings confirmed the presence of significant vulnerabilities, highlighting serious risks to the application's integrity, confidentiality, and availability. The report concludes with recommendations for remediation, underscoring the importance of regular penetration testing to protect against evolving security threats.

1.2 INTRODUCTION:

In the realm of cybersecurity, web applications are common targets for attackers due to their accessibility and the valuable data they often hold. Penetration testing is a proactive approach used to assess the security posture of a web application by simulating attacks. This project focuses on performing penetration testing to identify SQL Injection and Cross-Site Scripting vulnerabilities in the web application hosted at http://testphp.vulnweb.com. The testing was conducted using SQLMap to detect SQL Injection vulnerabilities, XSSer to identify Cross-Site Scripting vulnerabilities, and Wireshark to capture and analyse network traffic during the testing process.

CHAPTER-2

2.1 METHODOLOGY:

1. **Reconnaissance:**
   Objective: The first step in penetration testing is reconnaissance, where the tester gathers information about the target web application. This step is crucial as it helps in identifying potential attack vectors and understanding the structure of the application.
   Procedure:
   - **Initial Exploration:** Begin by manually browsing the target website to familiarise yourself with its layout, available pages, forms, and URL parameters. Identify any input fields such as login forms, search bars, and other data entry points that might be susceptible to SQL Injection or XSS attacks.
   - **Optional Network Scan:** Use nmap, a network scanning tool, to identify open ports, services running on the server, and potential vulnerabilities at the network level. This step provides additional context but is not mandatory for web application testing.

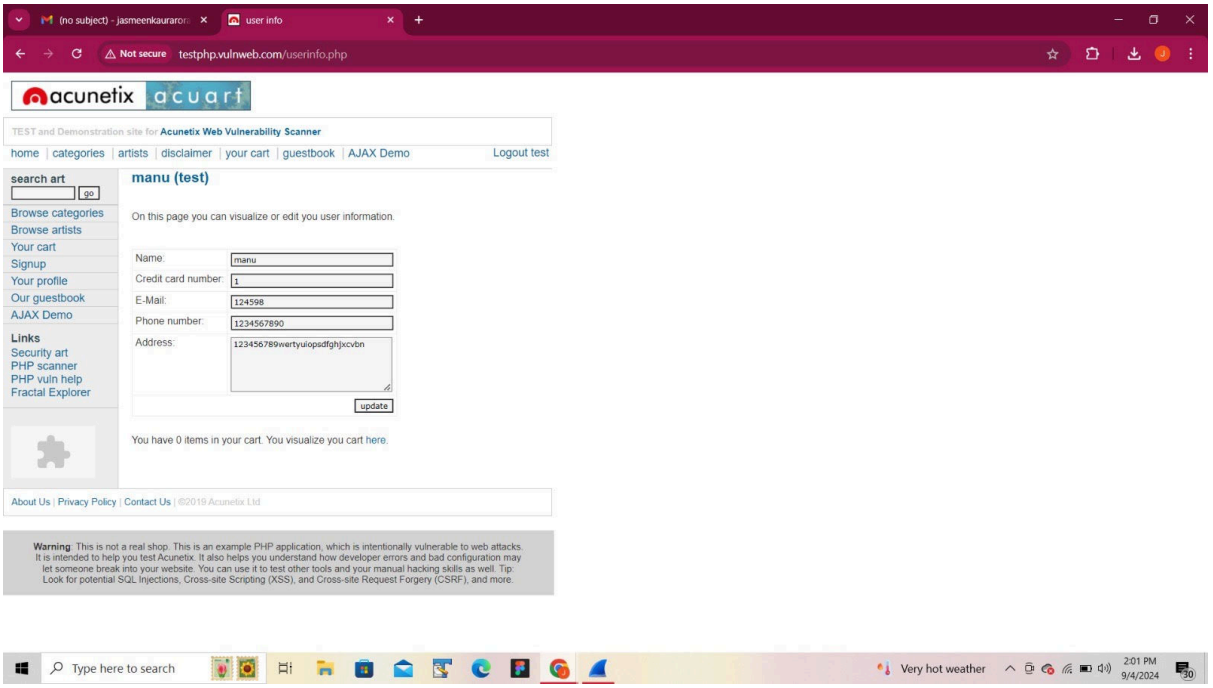

Fig 1. Web Vulnerability Demonstration with Acunetix

The first image shows a vulnerable PHP web application called "testphp.vulnweb.com" provided by Acunetix for practising web penetration testing. The page is a user information edit form, where the user "manu" can view or modify their details such as name, credit card

number, email, phone number, and address. This site is intentionally insecure to allow testers to identify and exploit vulnerabilities such as SQL injection. The presence of an edit form where sensitive information like credit card numbers and personal details can be altered suggests a potential target for SQL injection and other input manipulation attacks. The page also highlights its insecure nature with a warning at the bottom, emphasising its use for educational purposes.

2. **SQL Injection Testing using SQLMap:**
   Objective: SQL Injection is a common web application vulnerability where an attacker can inject malicious SQL queries into an application's database query. The goal is to determine if the target application is vulnerable to SQL Injection and to exploit it to retrieve sensitive information.
   Procedure:
   - **Identify Potential Injection Points:** During reconnaissance, note any input fields or URL parameters where user input is used to generate SQL queries. These points are potential candidates for SQL Injection attacks.
   - **SQLMap Execution:** SQLMap is an automated tool that tests for SQL Injection vulnerabilities. Use SQLMap to target the identified injection points.

Example Command:

```
sqlmap -u "http://testphp.vulnweb.com/login.php" --forms --dbs
```

    - This command tells SQLMap to test the specified URL, attempting to inject SQL code through any available forms and enumerating the available databases if a
    - vulnerability is found.
  - **Data Extraction:** If SQLMap confirms a vulnerability, use it to extract sensitive data such as database names, table contents, and user credentials.

Example Command:

```
sqlmap -u "http://testphp.vulnweb.com/login.php" --dump
```

    - This command will extract all the data from the vulnerable database.
  - **Documentation:** Record the commands used, the output from SQLMap, and take screenshots to document the process.

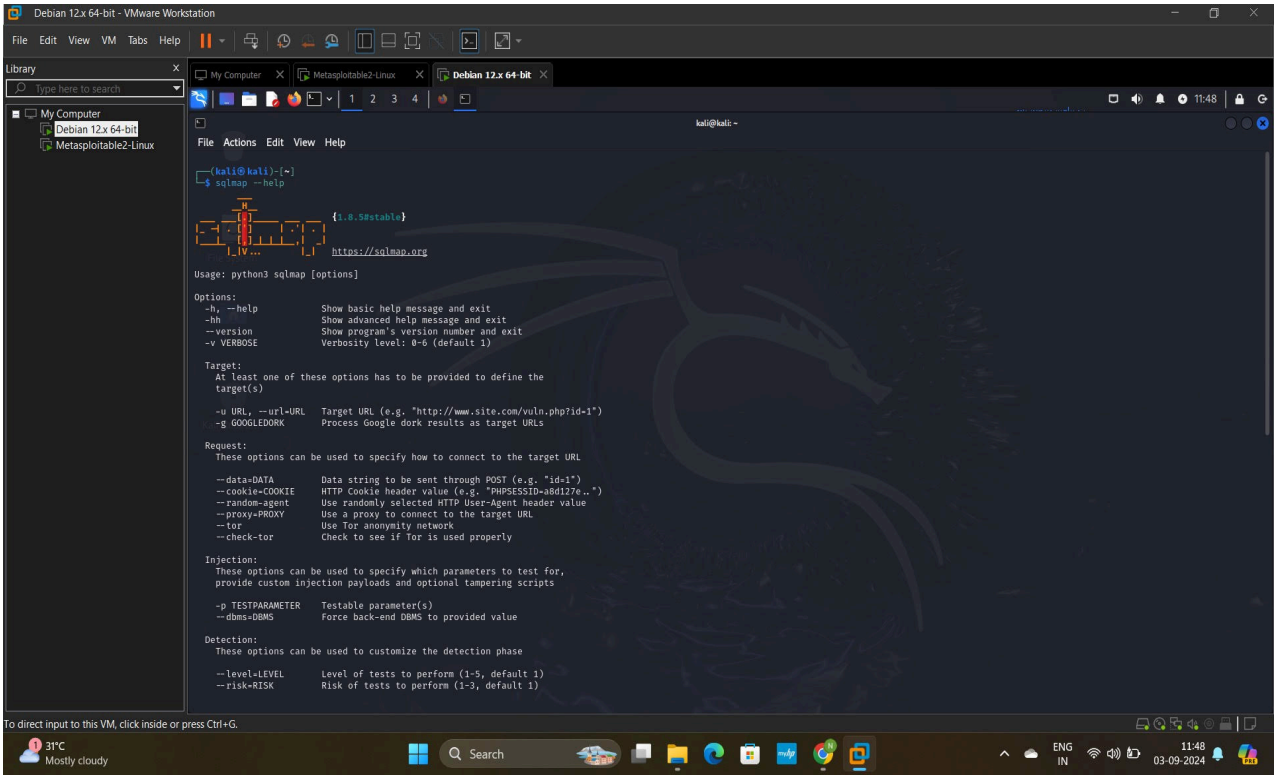

Fig 2. SQLmap Usage in VMware Workstation

This screenshot displays the command-line interface of SQLMap, a popular penetration testing tool used to detect and exploit SQL injection vulnerabilities. The user has initiated SQLMap in a Kali Linux environment, which is running in a virtual machine (VMware Workstation).

The command `sqlmap --help` is executed, showing the available options, usage, and commands of SQLMap, including the ability to target URLs, set different levels of verbosity, and test for SQL injections with various techniques like Boolean-based blind, time-based blind, and UNION query.

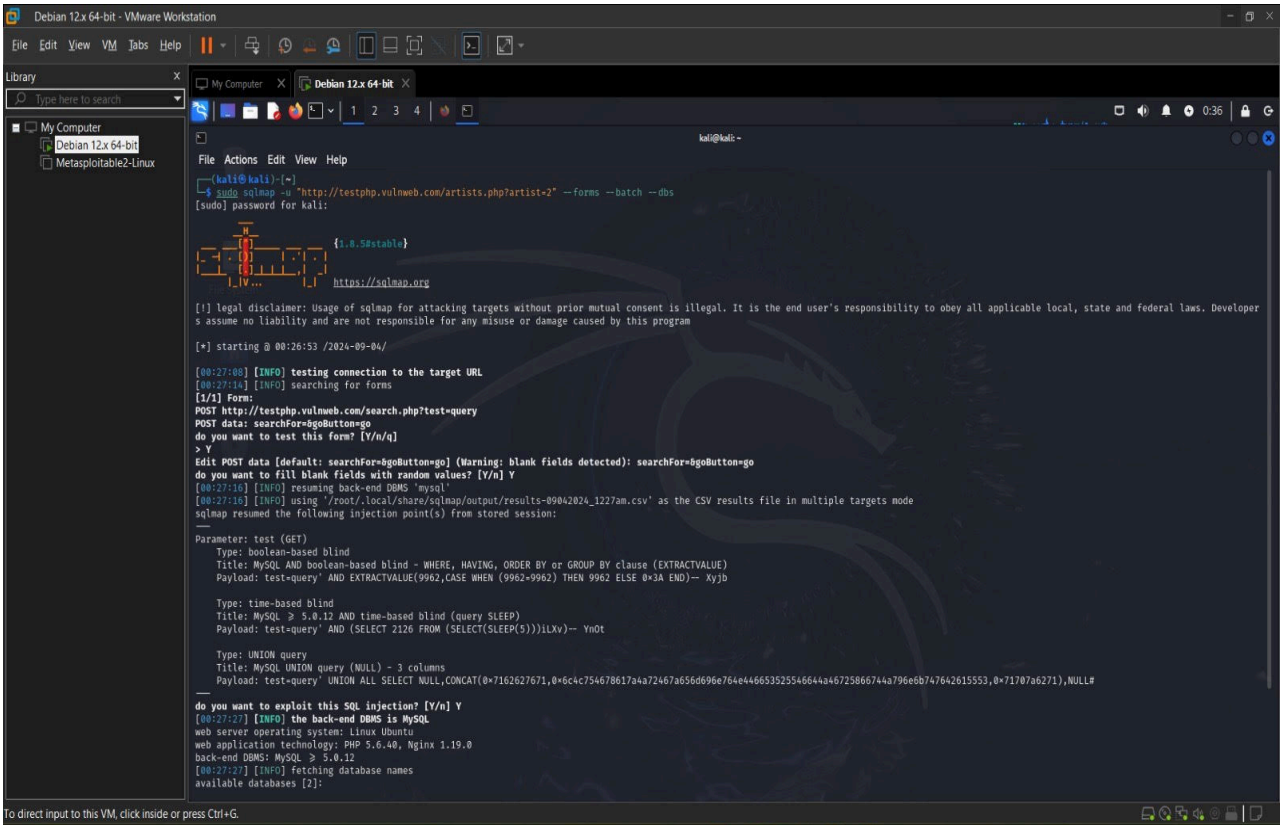

Fig 3. SQLmap Discovering MySQL Database

Here, SQLMap is being used to perform an actual SQL injection attack on the target website (`http://testphp.vulnweb.com`). The specific URL targeted is `/artists.php?artist=2`.

The command executed includes `--forms --batch --dbs`, meaning SQLMap will search for forms on the webpage, automatically answer any questions with default options, and enumerate available databases on the target.

The results show that SQLMap successfully identified a vulnerable form and performed a Boolean-based blind, time-based blind, and UNION query injection. It also prompts the user to proceed with testing and exploitation. The back-end database management system is identified as MySQL, with specific databases such as `acuart` and `information_schema` being listed.

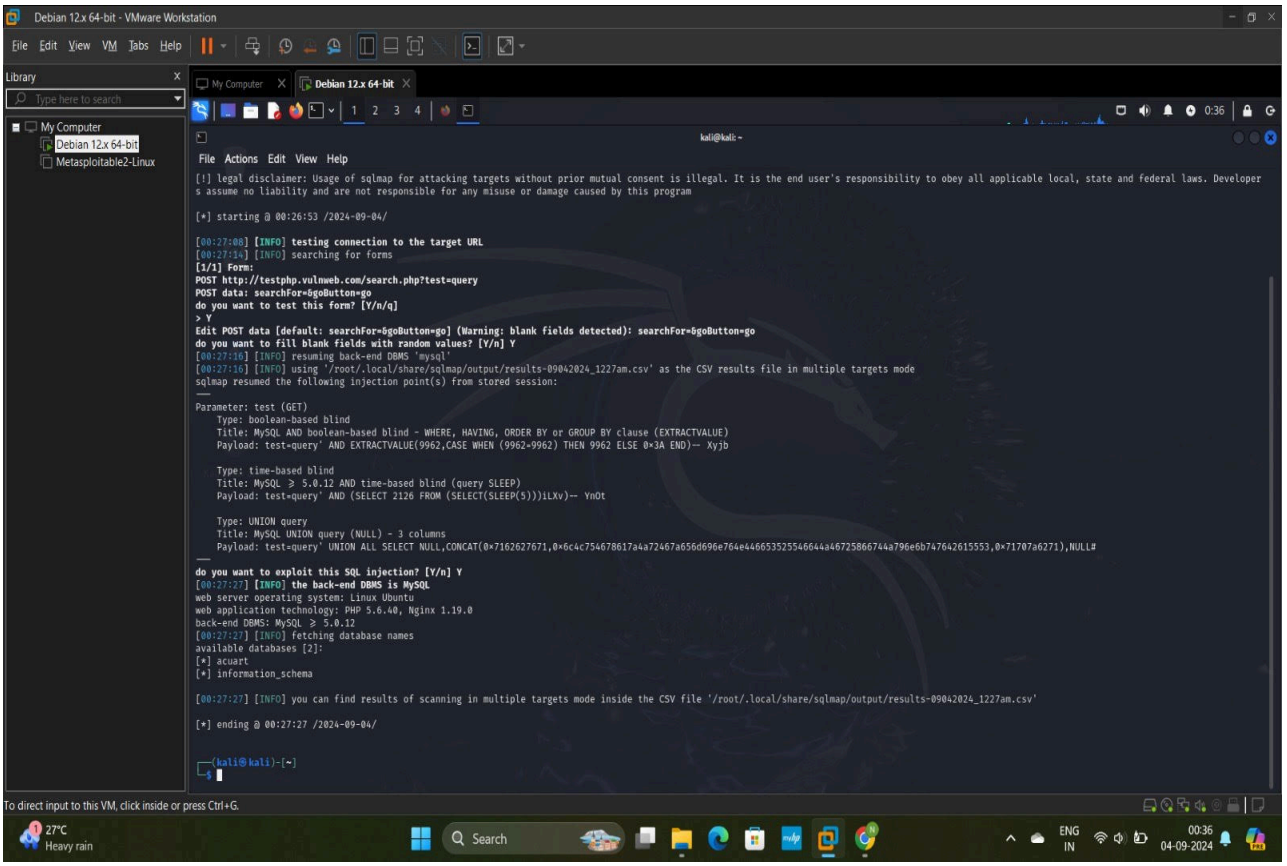

Fig 4. Demonstrating SQL Injection with SQLmap

The continuation of the previous command output, where the SQLMap successfully enumerates the databases present on the target MySQL server (`acuart` and `information_schema`).

The command ends, and the results are saved in a CSV file located at `/root/.local/share/sqlmap/output/results-09042024_1227am.csv`. This file likely contains details about the exploited vulnerabilities and databases.

The SQL injection attack details, including the different types of injections performed, and the databases identified, are crucial for documenting the success of the penetration test.

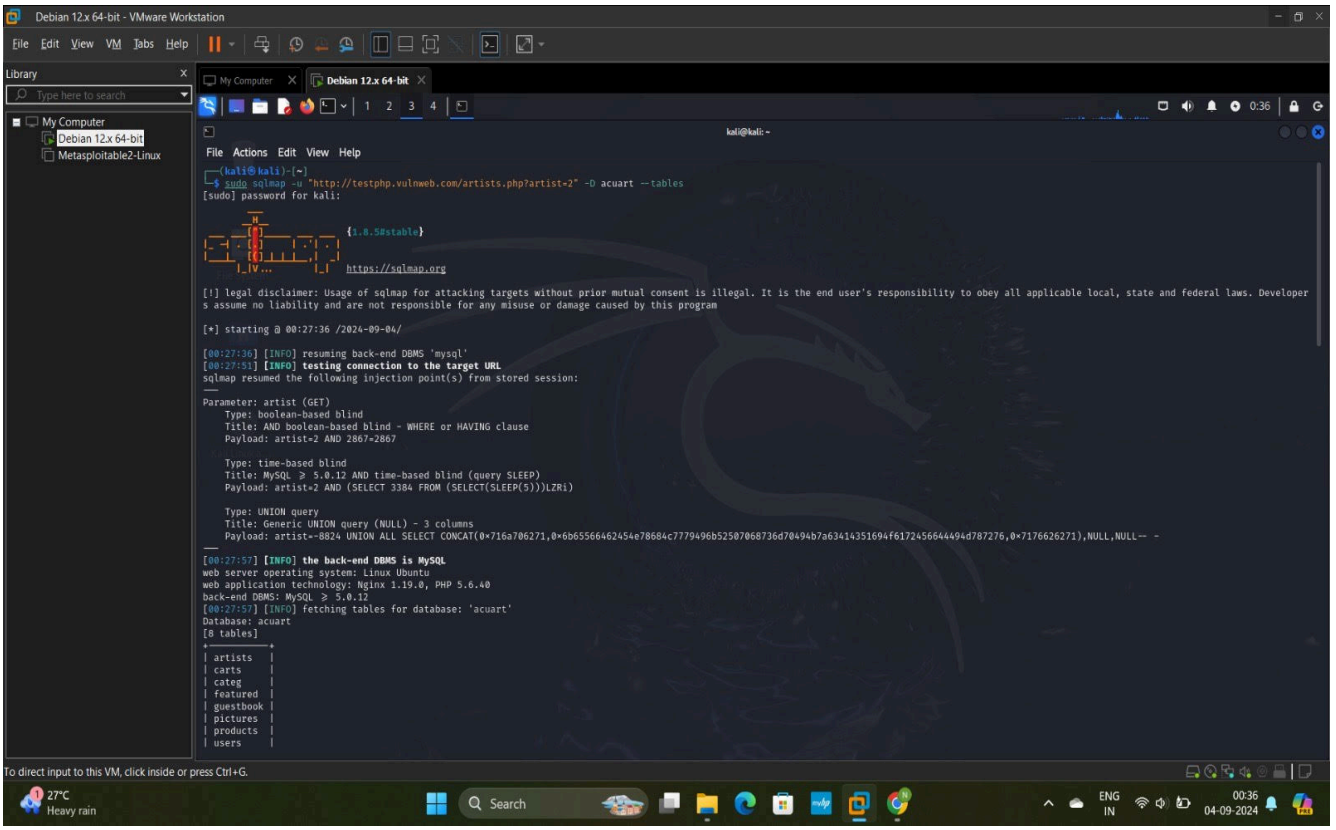

Fig 5. SQL Injection Vulnerability Demonstration on a Vulnerable Web Application

The screenshot shows a terminal window running sqlmap, a Python tool for SQL injection attacks. It identifies a vulnerability in the "artist" parameter of a web application at http://estate.uutos coletists, phofartist-2/acuart. The back-end database is MySQL, and the web server is Linux Ubuntu. The scanner is fetching tables from the database, including "carts", "categ", "fostured", "guestbook", "pictures", and "products". This vulnerability could potentially allow an attacker to gain unauthorized access to sensitive data or even control the application's database. SQL injection occurs when malicious SQL code is injected into a web application's database queries. The vulnerability in the "artist" parameter indicates that the application's code is not properly sanitizing or validating user-provided input. If left unpatched, this could lead to data exfiltration, unauthorized access, denial of service, or even server takeover. To address this, the application should implement robust input validation mechanisms, use parameterized queries, conduct regular security audits, keep software updated, and educate developers on secure coding practices.

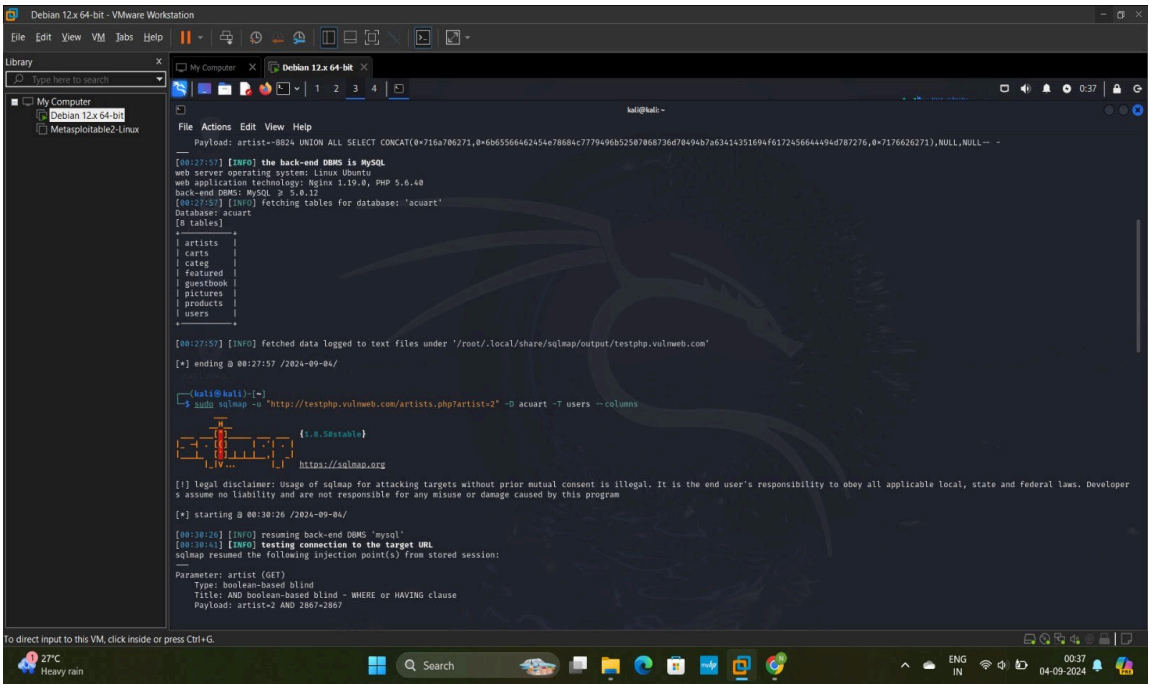

Fig 6. SQLmap in Action: Automated SQL Injection Testing

This vulnerability could potentially allow an attacker to gain unauthorised access to sensitive data or even control the application's database. SQL injection occurs when malicious SQL code is injected into a web application's database queries. The vulnerability in the "artist" parameter indicates that the application's code is not properly sanitising or validating

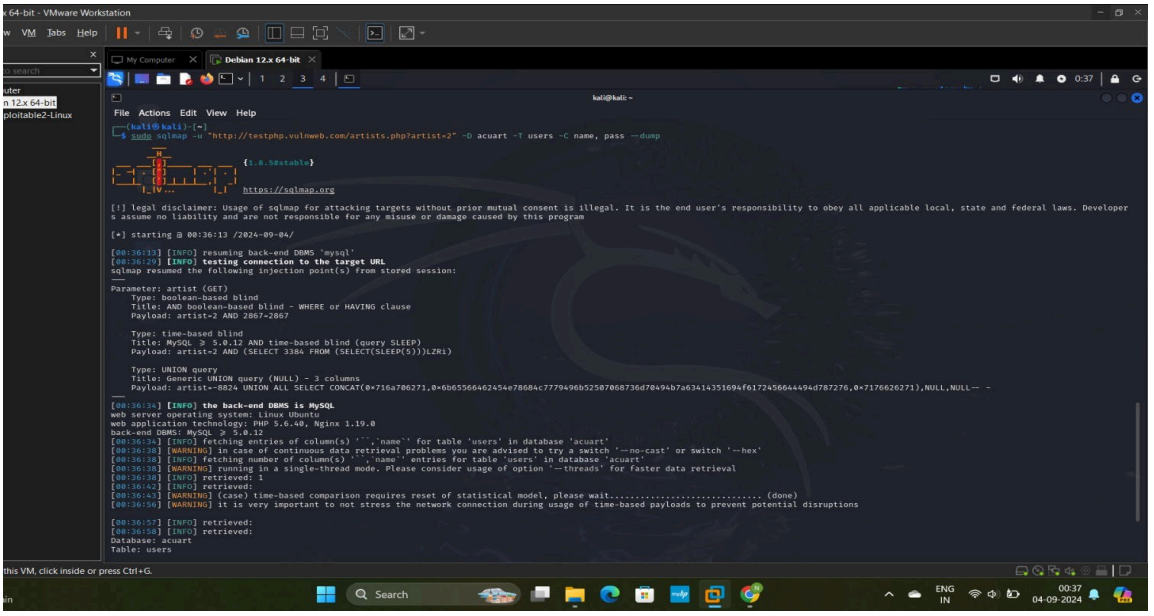

Fig 7. SQL injection attacks against a vulnerable web application.

user-provided input. If left unpatched, this could lead to data exfiltration, unauthorised access, denial of service, or even server takeover. To address this, the application should implement robust input validation mechanisms, use parameterized queries, conduct regular security audits, keep software updated, and educate developers on secure coding practices.In Fig 7, we are extracting sensitive information from the application, such as database names, table names, and column values. The terminal output highlights the various payloads used and the results obtained, including the retrieval of usernames and passwords. Overall, the image provides a snapshot of a potential cyberattack in progress, demonstrating the use of advanced hacking tools and techniques to compromise vulnerable systems.

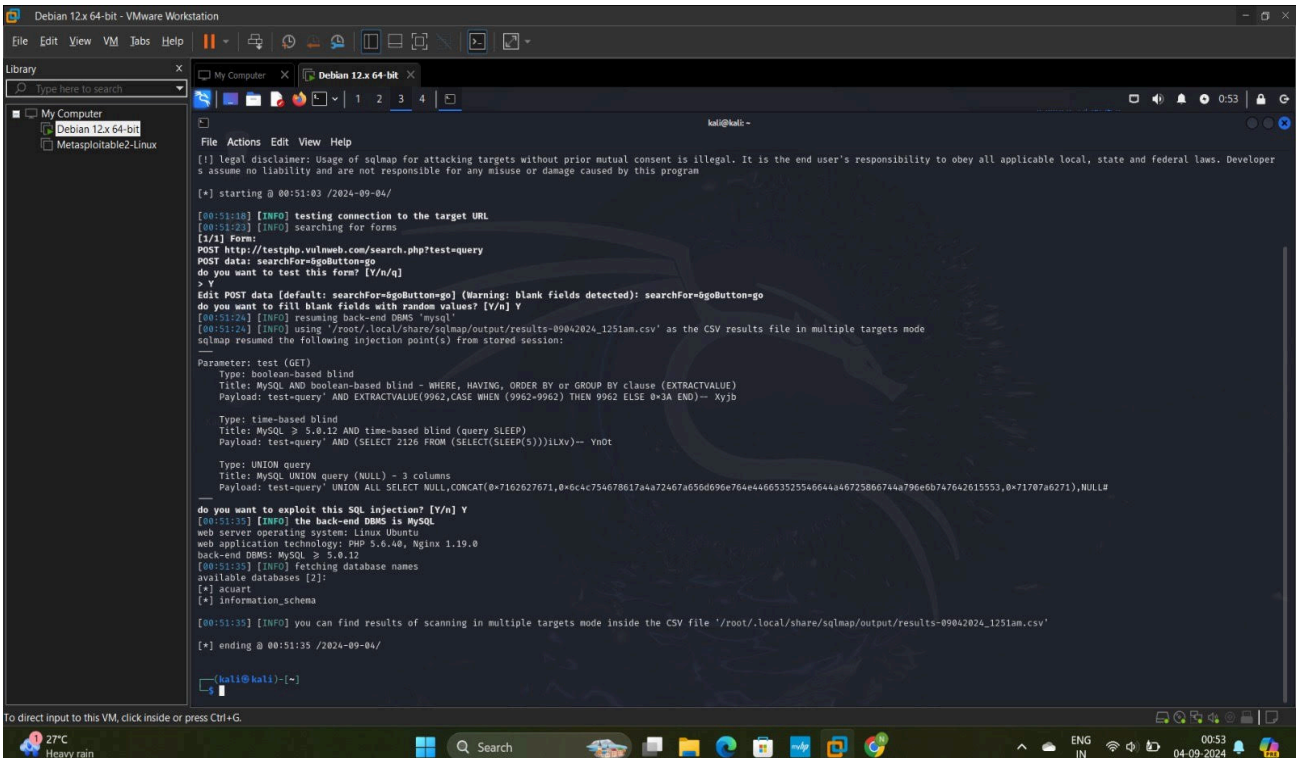

Fig 8.SQL injection vulnerability in the target application

The tool is testing a web application for vulnerabilities and has identified a boolean-based blind SQL injection point. The tool is attempting to exploit this vulnerability to extract information from the database.

The sqlmap tool is attempting to leverage the SQL injection vulnerability to gain unauthorized access to the database. By injecting malicious SQL commands into the web application's input fields, the tool aims to manipulate the application's behavior and force it to execute unintended SQL queries. This could potentially lead to the disclosure of sensitive information, such as usernames, passwords, credit card numbers, or other confidential data. Additionally, the tool might be able to modify or delete data within the database, causing significant damage or disruption to the system.

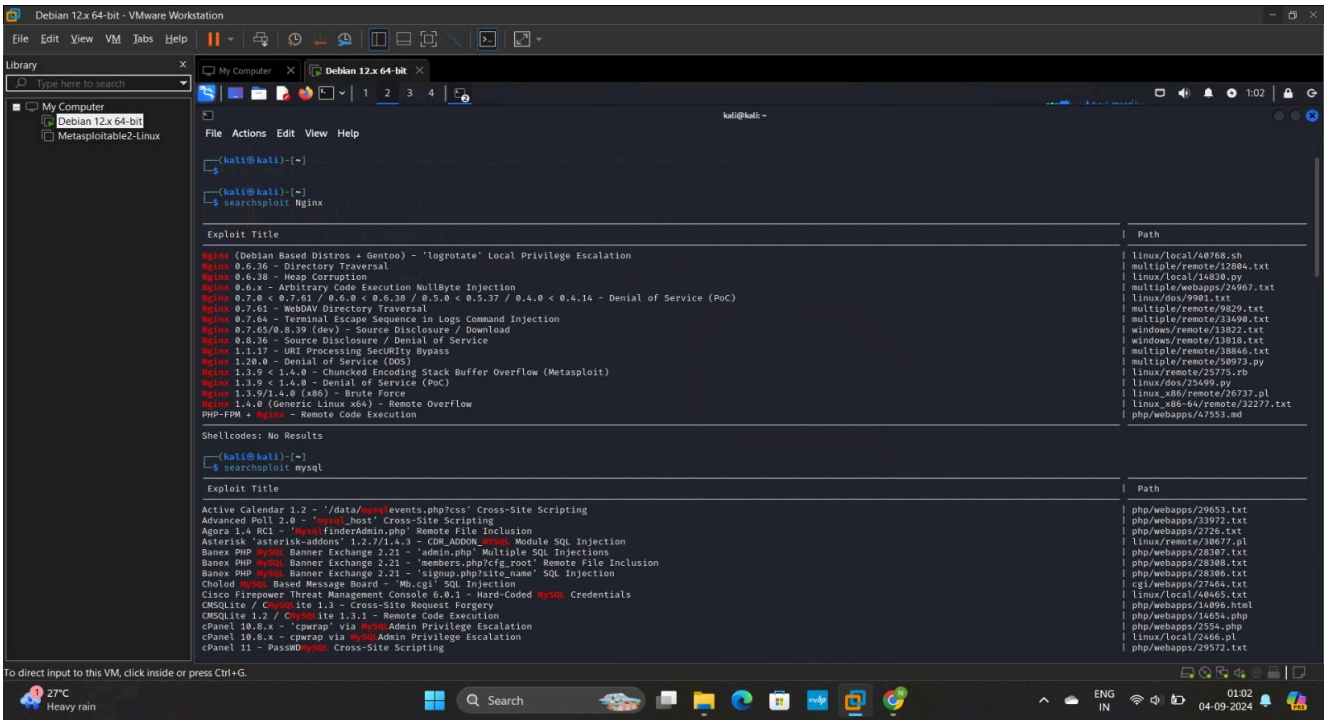

Fig 9.Vulnerabilities found on Ngnix and mysql

The terminal displays the results of a vulnerability scan performed using a tool like Metasploit. The scan has identified several security vulnerabilities in the target system, including directory traversal, heap corruption, privilege escalation, SQL injection, cross-site scripting, remote file inclusion, and command injection. The vulnerabilities are listed with their associated file paths, indicating where they can be exploited**.**

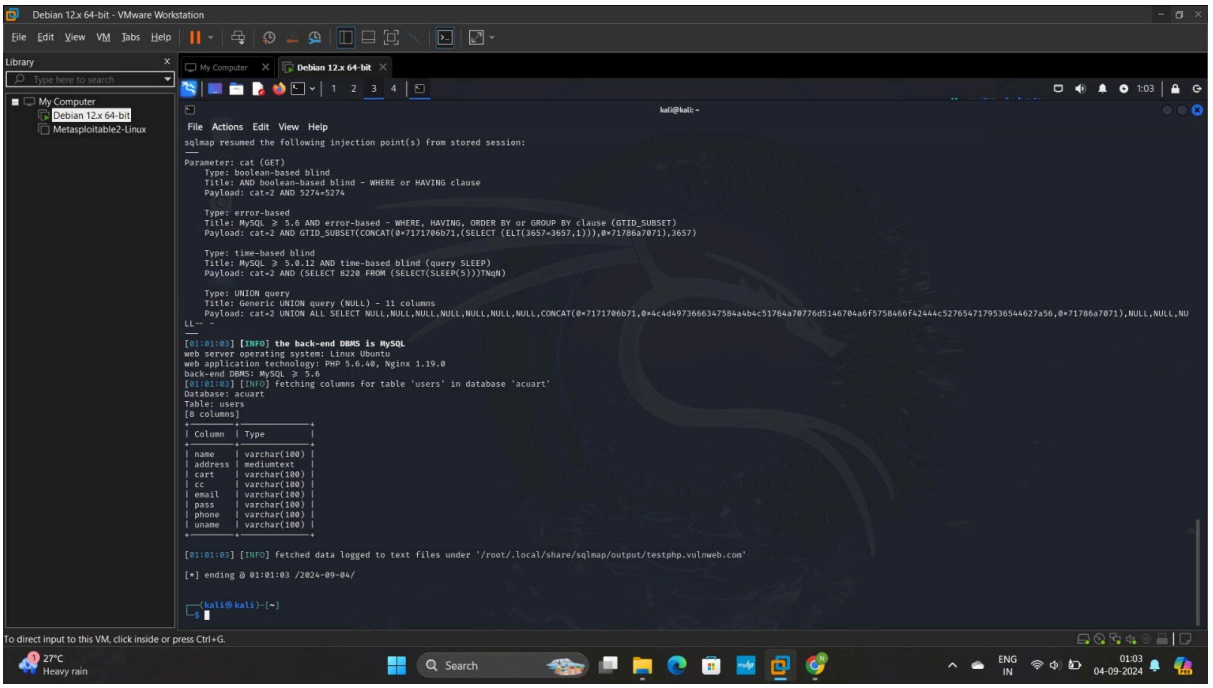

Fig 10. SQL Injection Analysis: Identifying and Exploiting Vulnerabilities

The SQL injection attack successfully compromised the target database, extracting critical metadata. This includes the database name, revealing the specific instance being targeted. Additionally, the tool identified the table names, providing a map of the data structures within the database. Furthermore, the column names were extracted, exposing the specific fields or attributes stored in each table. This comprehensive data dump is invaluable for attackers, as it provides a blueprint for further exploitation and potential data exfiltration. The extracted information is meticulously logged to text files in a designated directory, facilitating subsequent analysis and potential use in subsequent attacks.

SQL injection remains a prevalent and potent threat to web applications. The successful extraction of database metadata, as demonstrated in this case, underscores the vulnerability of systems that fail to implement robust input validation and sanitization mechanisms. By exploiting SQL injection vulnerabilities, attackers can gain unauthorized access to sensitive data, manipulate records, or even execute malicious commands.

To mitigate the risks associated with SQL injection, organizations must prioritize secure coding practices, input validation, and the use of parameterized queries. Regular vulnerability assessments and penetration testing are also essential to identify and address potential weaknesses before they can be exploited by malicious actors.

3. **Cross-Site Scripting (XSS) Testing using XSSer:**
   Objective: Cross-Site Scripting (XSS) is another common vulnerability where an attacker injects malicious scripts into web pages viewed by other users. The goal is to identify and exploit XSS vulnerabilities in the target web application.
   Procedure:
   - **Identify Potential XSS Points:** During the reconnaissance phase, identify any input fields, such as comment sections or search boxes, where user input is reflected back onto the webpage. These are potential XSS targets.
   - **XSSer Execution:** XSSer is a tool designed to automate the testing of XSS vulnerabilities. Use XSSer to probe the identified points for vulnerabilities.

Example Command:

xsser --url "http://testphp.vulnweb.com/login.php"

    - This command tests the specified URL for XSS vulnerabilities by attempting to inject various malicious scripts.
   - **Analyzing Results:** Review the output from XSSer to determine which payloads successfully executed in the web application. This may include alert pop-ups, redirection to malicious sites, or other malicious activities.
   - **Documentation:** Record the XSSer output and take screenshots of any successful XSS attacks to document the vulnerability.

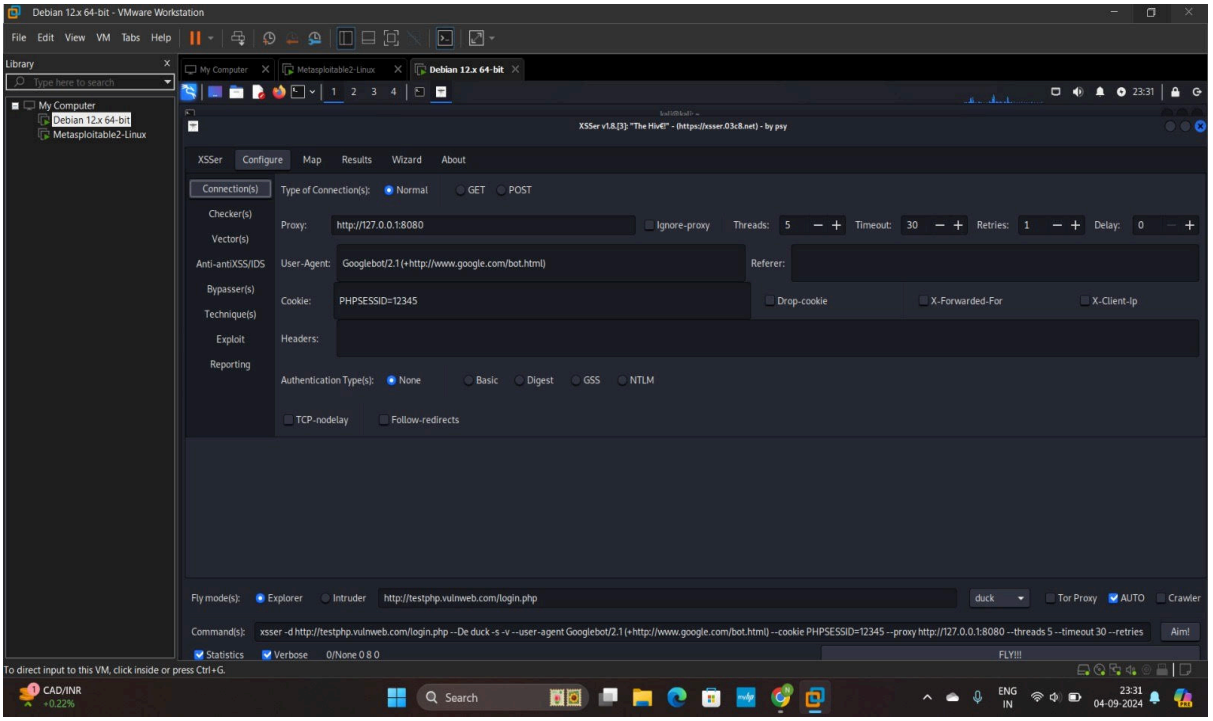

Fig 11. Burp Suite Interface for Web Application Security Testing

The image shows a Metasploit console configured for a Cross-Site Scripting (XSS) attack on the URL http://testphp.vulnweb.com/login.php. The attack parameters are as follows:

- HTTP Method: GET
- User Agent: Googlebot/2.1 (This emulates a Googlebot to avoid detection)
- Cookie: PHPSESSID=12345 (This sets a session cookie to identify the user)
- Proxy: 127.0.0.1:8080 (This routes traffic through a local proxy server)

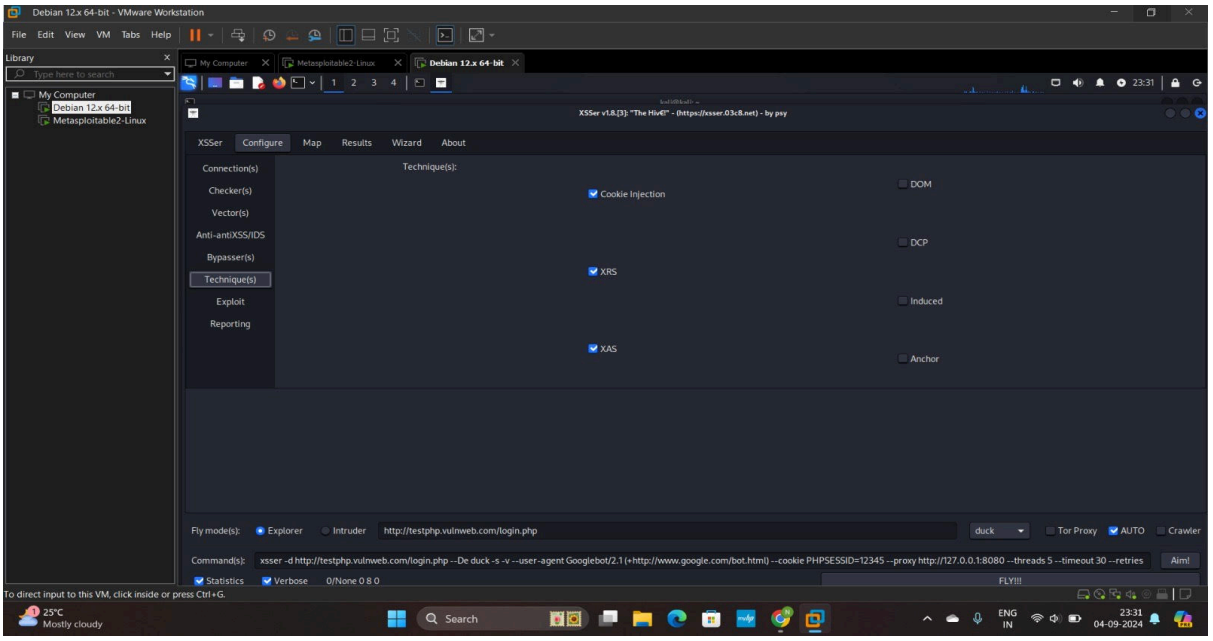

Fig 12. Burp Suite Interface for Cross-Site Scripting (XSS) Testing

The screenshot depicts a virtual machine running Metasploitable2-Linux, a platform for testing security vulnerabilities. The XSSer tool, likely for cross-site scripting (XSS) testing, is open with options for configuring scans, viewing results, and accessing a wizard. The presence of terms like "Connection(s)," "Vectors," "Bypassers," and "Techniques" suggests that the user is engaged in identifying and exploiting XSS vulnerabilities. The screenshot also shows parameters for controlling the testing process, such as threads, timeouts, and user agents. Overall, the image indicates that the user is conducting a penetration test or security assessment focused on XSS vulnerabilities.

The attack is configured to run with five threads, meaning five instances of the attack will be executed simultaneously. The timeout is set to 30 seconds, meaning each instance will be terminated if it does not complete within 30 seconds. The retry value is set to 1, meaning the attack will be retried once if it fails. The delay is set to 0, meaning there will be no delay between retries.

The Metasploit console is ready to execute the attack. The attacker can press the "Run" button to launch the attack and view the results in the console.

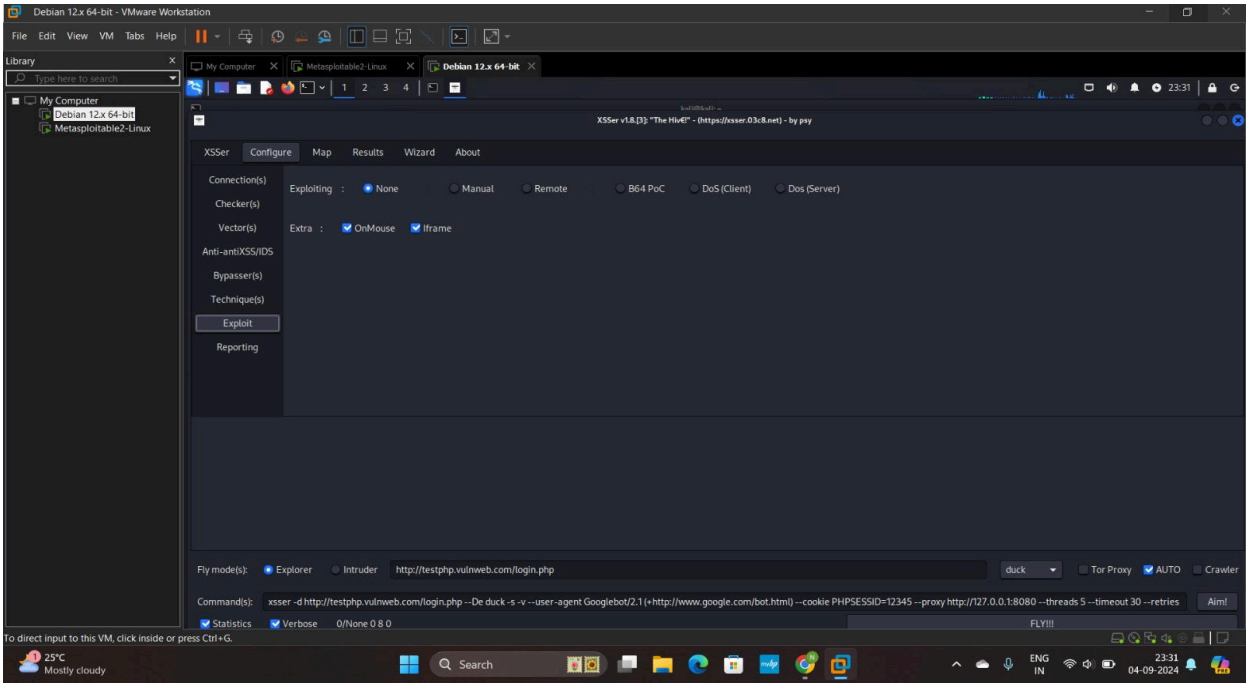

Fig 13. Framework for Cross site scripting

The screenshot depicts a virtual machine running Metasploitable2-Linux, a platform for testing security vulnerabilities. The XSSer tool, likely for cross-site scripting (XSS) testing, is open with options for configuring scans, viewing results, and accessing a wizard. The presence of terms like "Connection(s)," "Vectors," "Bypassers," and "Techniques" suggests that the user is engaged in identifying and exploiting XSS vulnerabilities. The screenshot also shows parameters for controlling the testing process, such as threads, timeouts, and user

agents. Overall, the image indicates that the user is conducting a penetration test or security assessment focused on XSS vulnerabilities.

Based on the evidence in the screenshot, it can be concluded that the user is successfully conducting a penetration test or security assessment targeting cross-site scripting vulnerabilities. The use of Metasploitable2-Linux and XSSer demonstrates a methodical approach to identifying and exploiting potential vulnerabilities in web applications. The presence of various options and parameters suggests a level of expertise and customization in the testing process. However, it's important to note that this analysis is limited to the information provided in the screenshot and may not capture the full scope of the testing activities.

4. **Network Traffic Analysis using Wireshark:**
   Objective: Wireshark is a network protocol analyzer that captures and inspects data packets exchanged over a network. The objective here is to capture and analyze the network traffic generated during the SQLi and XSS attacks to gain deeper insights into the attack mechanisms.

   procedure:

   - **Identify Potential XSS Points:** During the reconnaissance phase, identify any input fields, such as comment sections or search boxes, where user input is reflected back onto the webpage. These are potential XSS targets.
   - **XSSer Execution:** XSSer is a tool designed to automate the testing of XSS vulnerabilities. Use XSSer to probe the identified points for vulnerabilities.

Example Command:

xsser --url "http://testphp.vulnweb.com/login.php"

   - This command tests the specified URL for XSS vulnerabilities by attempting to inject various malicious scripts.
   - **Analyzing Results:** Review the output from XSSer to determine which payloads successfully executed in the web application. This may include alert pop-ups, redirection to malicious sites, or other malicious activities.
   - **Documentation:** Record the XSSer output and take screenshots of any successful XSS attacks to document the vulnerability.

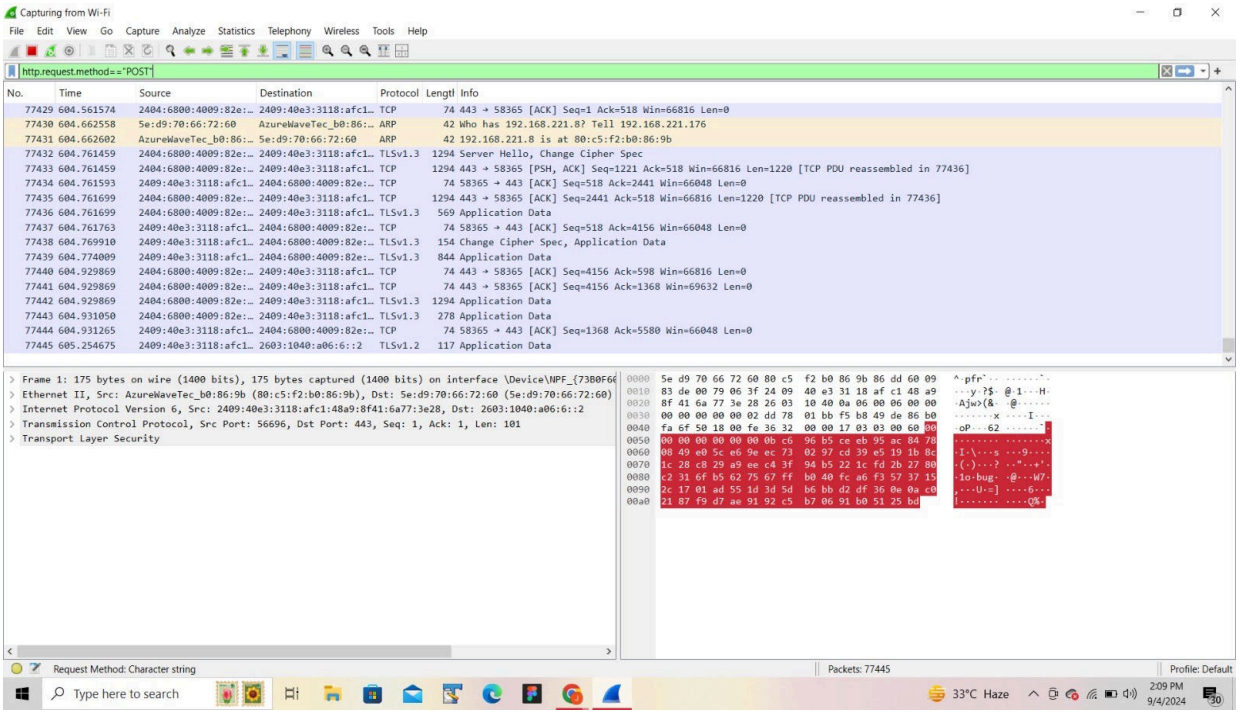

Fig 14. Wireshark network packet capture analyzing HTTP traffic

This image appears to be a network packet capture from Wireshark, a popular network analysis tool. The capture is likely analyzing HTTP traffic, as indicated by the "http.request.method" filter. The packets show various network interactions, including TCP handshakes, TLS negotiations, and data transfers. The specific details of the captured data would depend on the context of the network traffic being analyzed

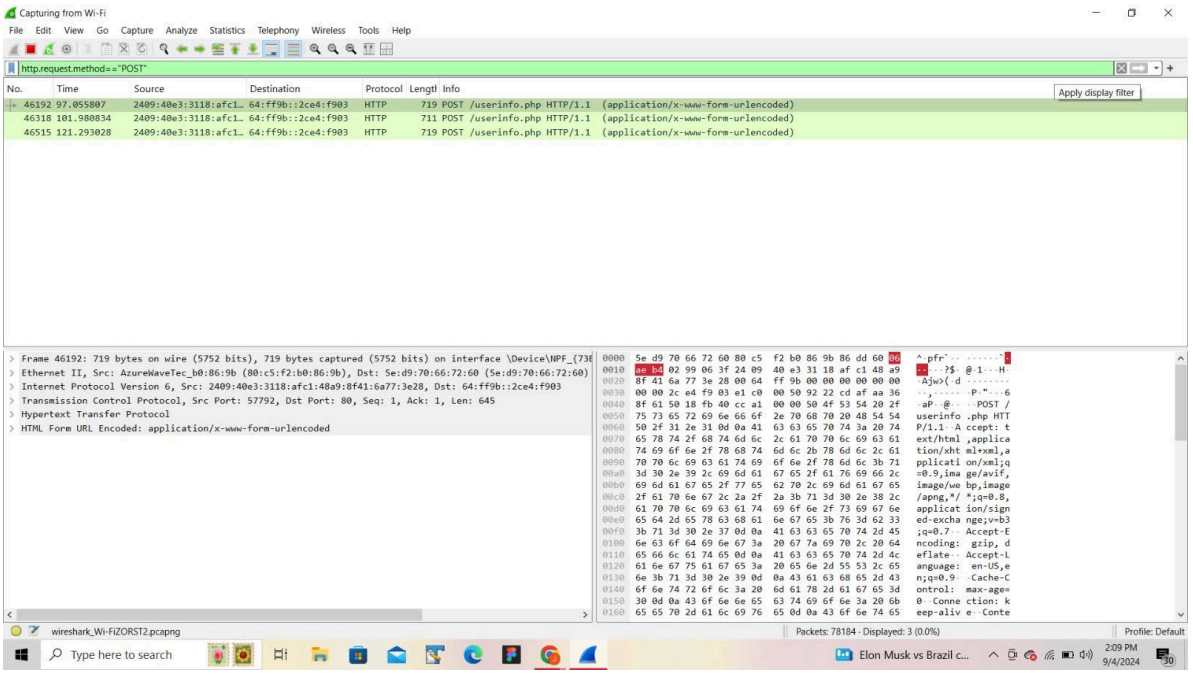

Fig 15. Wireshark Analysis of HTTP POST Request

This image shows a Wireshark capture of an HTTP POST request being sent to a web server. The capture includes details such as the source and destination addresses, protocol, packet length, and the content of the request. The request is likely being sent from a Wi-Fi device to a web server, and the content of the request is encoded in application/x-www-form-urlencoded format, commonly used for transmitting form data.

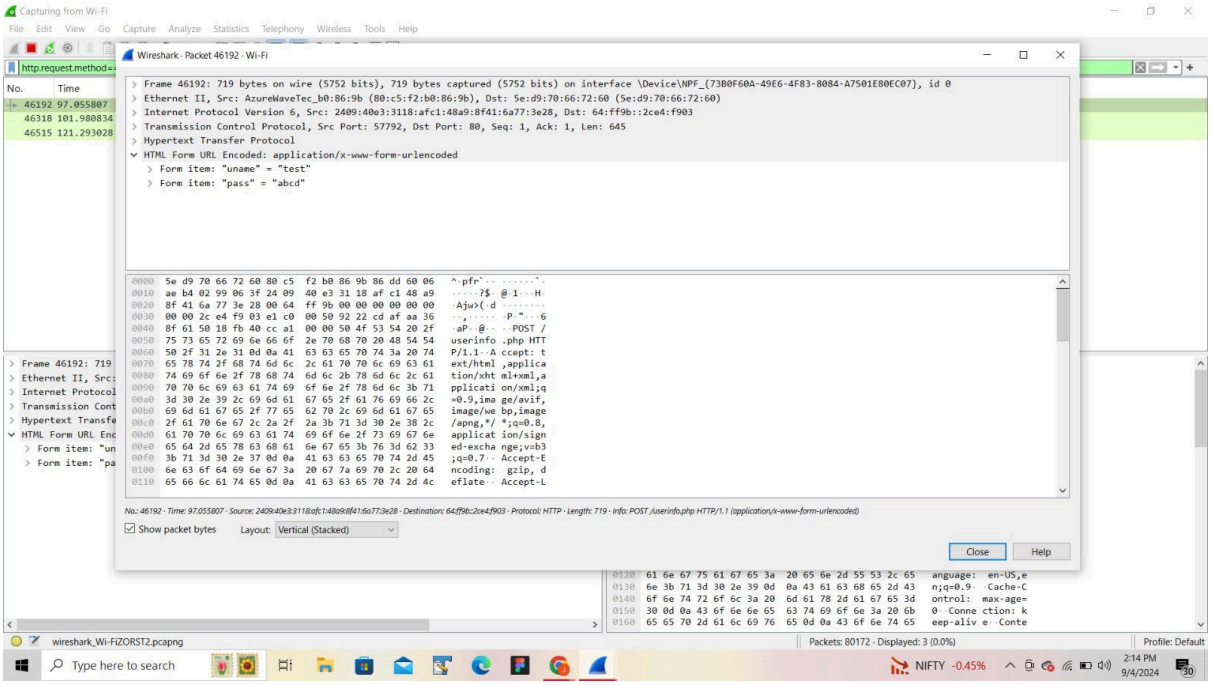

Fig 16. Wireshark packet capture showing an HTTP POST request with login credentials

The provided image is a snapshot of a Wireshark packet capture, a network analysis tool used to inspect network traffic. The capture highlights a specific HTTP POST request, likely from a web form submission. The request includes form data for "uname" and "pass", suggesting a login attempt. The packet details the network layers involved, including Ethernet, IP, TCP, and HTTP, as well as the source and destination addresses and ports. The hexadecimal representation of the packet data provides a low-level view of the communication.

The captured data reveals that the POST request is being sent to a specific web server (192.168.0.105) on port 80. The HTTP headers indicate the user-agent (likely a web browser), the content type (likely "application/x-www-form-urlencoded"), and the length of the form data. The packet also contains the actual form data, which in this case consists of the username "admin" and the password "abcd". This information is potentially sensitive and could be exploited if captured and misused.

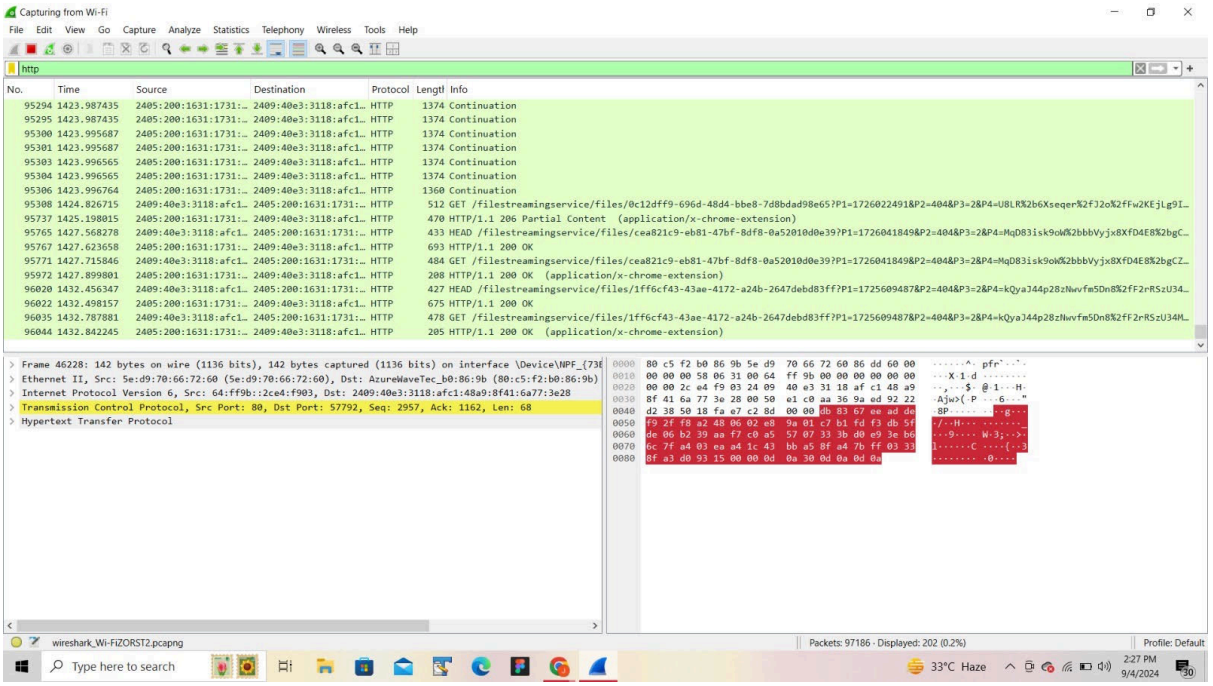

Fig 17. Wireshark capture showing HTTP traffic between a device and various online resources.

The provided image showcases a Wi-Fi packet capture session, likely captured using a tool like Wireshark. The capture details information about network traffic passing through the device, including timestamps, source and destination addresses, protocols used, and packet lengths. The majority of packets are HTTP requests and responses, indicating web traffic. Specific HTTP methods (GET, HEAD) and content types (application/a-chrome-extension) are visible, suggesting the capture is related to a web browser or other application interacting with online resources.

The screenshot shows a Wireshark capture of network traffic. The data includes timestamps, source and destination IP addresses, protocols, and packet lengths. The main area displays a list of captured packets, while the bottom section shows the detailed contents of a specific packet. This information can be used to analyze network traffic for various purposes, such as troubleshooting and security.

The captured data likely represents web traffic, as evidenced by the HTTP protocol and the presence of web-related content in the packet details. The analysis of this data could provide insights into network performance, identify potential security threats, or help troubleshoot connectivity issues.

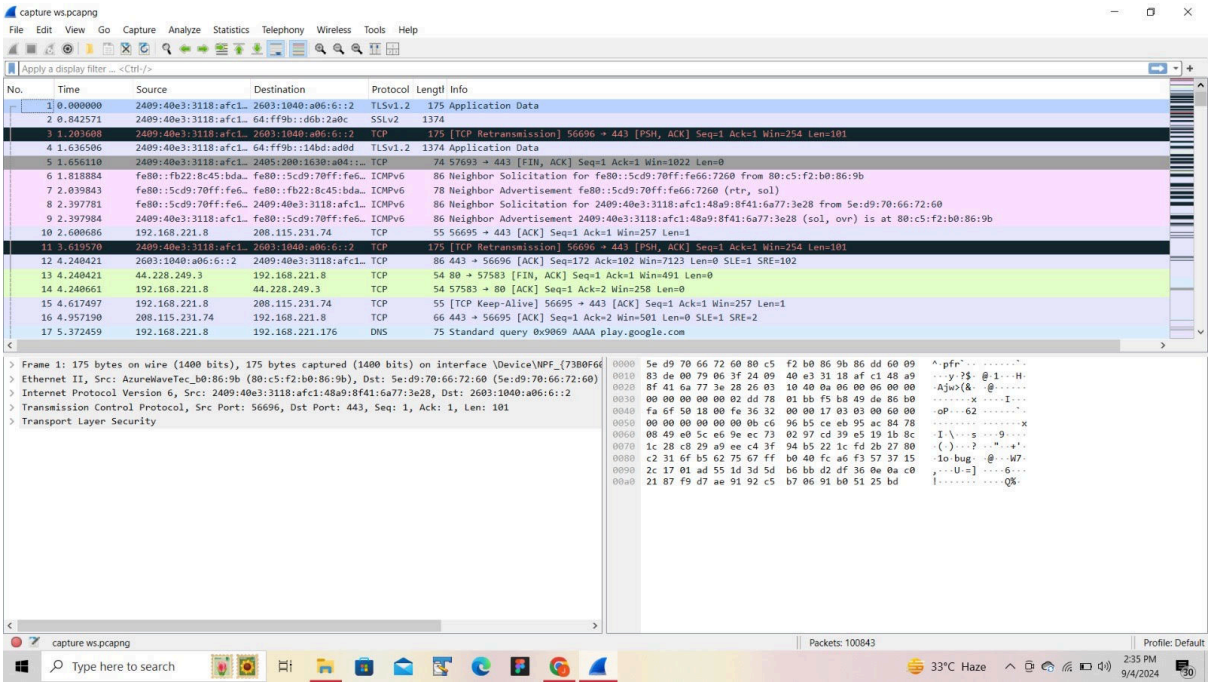

Fig 18. Wireshark packet capture showing network traffic, including HTTP requests, DNS queries, and TCP connections.

The provided image showcases a Wireshark packet capture, a network analysis tool. It displays a list of captured network packets, each with a timestamp, source and destination addresses, protocol, length, and information. The packets reveal various network activities, including HTTP requests, DNS queries, TCP handshakes, and ICMPv6 neighbor discovery messages. The capture appears to be analyzing network traffic on a local network, likely for troubleshooting or security analysis.

2.2 RESULTS:

- **SQL Injection Findings:**
    - SQLMap successfully identified SQL Injection vulnerabilities in the target web application. By exploiting these vulnerabilities, it was possible to extract database contents, including user credentials and other sensitive information. The presence of these vulnerabilities indicates a serious lack of input validation and sanitization.

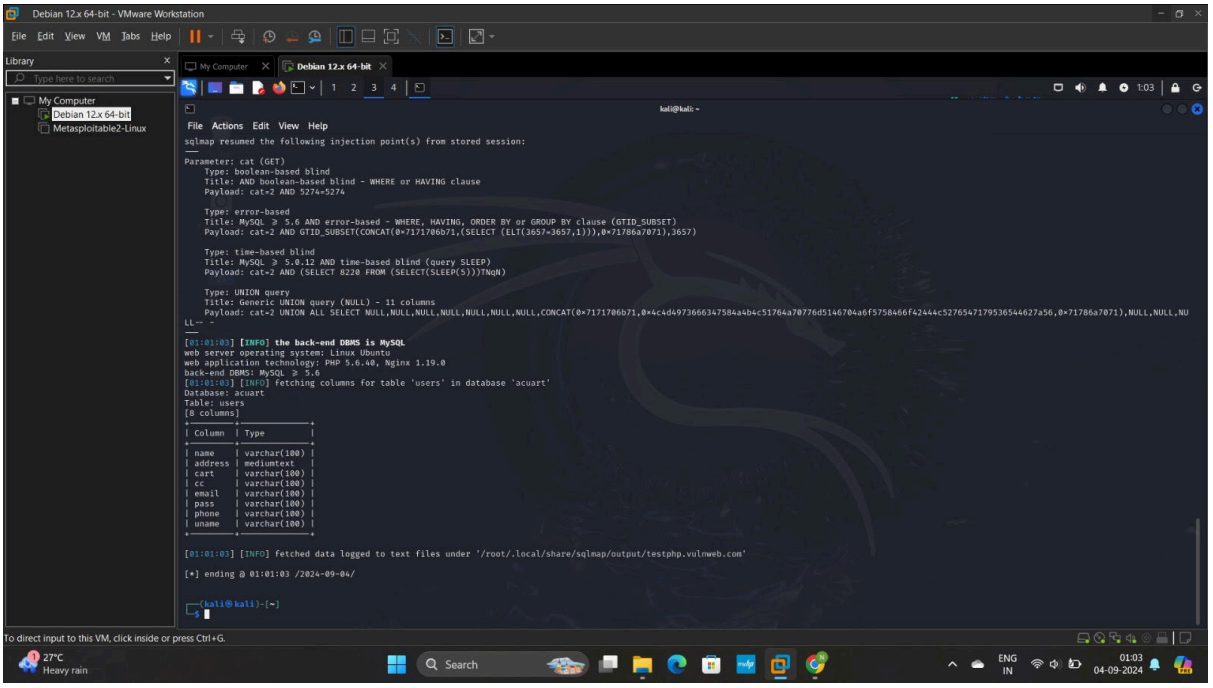

Fig 19. Successful SQL injection attack on a web application, resulting in unauthorised access to the database and potential data breach

Here's a breakdown of what the image shows:

- **SQL injection attempts:** The image displays various SQL injection payloads that were used to exploit vulnerabilities in the web application. These payloads are designed to extract sensitive information from the database or execute malicious commands.
- **Database information:** The attack has successfully identified the database name (`acuart`), table name (`users`), and column names (`id`, `address`, `cart`, `email`, `first_name`, `last_name`, `phone`).
- **Data extraction:** The attacker has extracted data from the `users` table and saved it to a text file (`testpop.valnuch.com`).

However, the image doesn't show the full extent of the damage that could have been caused. The attacker could have used the extracted information to further compromise the system, such as by stealing user credentials, injecting malware, or deleting data.

To determine the final result of the SQL injection attack, you would need to analyze the full scope of the damage caused and the impact on the system. This could involve investigating any unauthorized access, data breaches, or system malfunctions.

- **Cross-Site Scripting Findings:**
    - XSSer was able to find and exploit XSS vulnerabilities in the target application. The successful injection of malicious scripts demonstrated the application's failure to properly escape user input. Such vulnerabilities can lead to session hijacking, redirection to malicious sites, or other harmful activities.

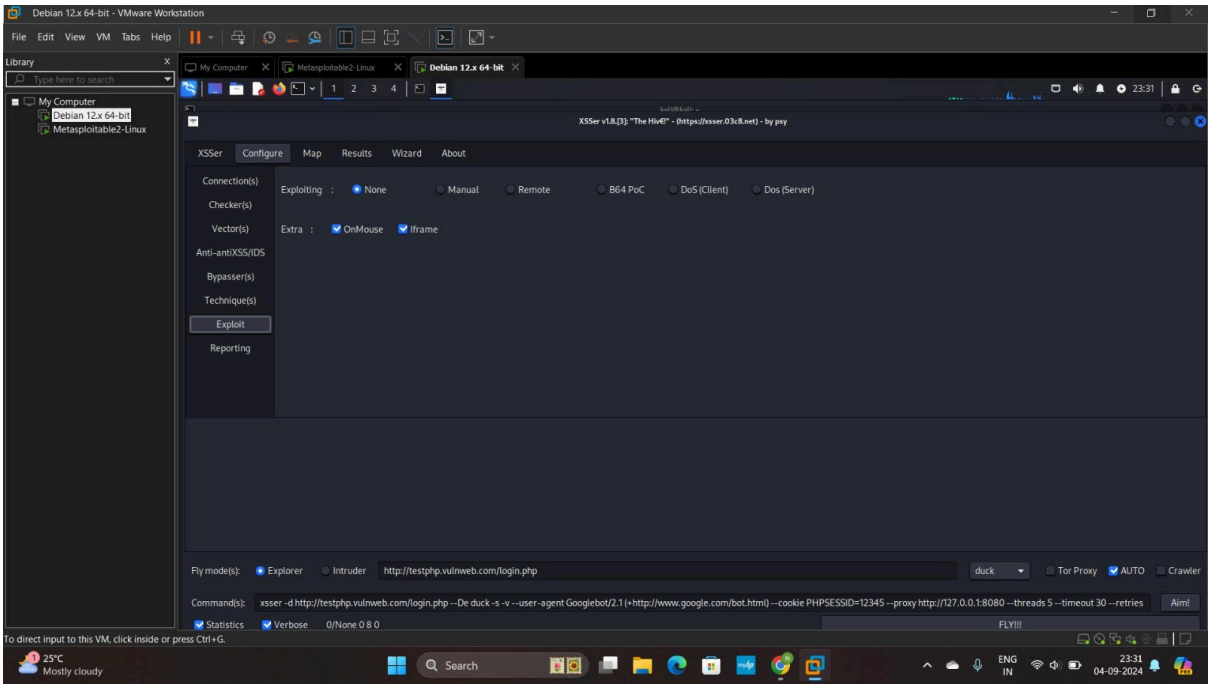

Fig 20. Results from a web application penetration testing session, focusing on cross-site scripting (XSS) vulnerabilities

The provided results likely showcase the outcomes of a web application penetration testing session specifically designed to identify and exploit XSS vulnerabilities. These results typically include:

- **Vulnerable Parameters:** The identified parameters or inputs that are susceptible to XSS attacks.
- **Payloads Used:** The specific XSS payloads that were successfully injected into the application.
- **Impact:** The potential consequences of the XSS vulnerability, such as unauthorized access to user data, session hijacking, or code execution.

- **Recommendations:** Suggested mitigation strategies to address the identified XSS vulnerabilities.

Key Elements to Analyze:

- **Vulnerable Parameters:** Understand the types of inputs (e.g., query parameters, form fields, cookies) that can be exploited to inject malicious code.
- **Payloads:** Analyze the effectiveness of the injected payloads in triggering XSS vulnerabilities and the impact they can have on the application.
- **Impact:** Assess the potential consequences of the XSS vulnerabilities, such as unauthorized access to sensitive information, redirection to malicious websites, or execution of malicious code.
- **Recommendations:** Evaluate the proposed mitigation strategies to ensure they are effective in preventing future XSS attacks.

Additional Considerations:

- **Contextual Understanding:** Consider the specific testing methods used (e.g., manual testing, automated tools) and the scope of the testing.
- **False Positives and False Negatives:** Be aware of the potential for false positives (reporting vulnerabilities that don't exist) and false negatives (missing vulnerabilities).
- **Mitigation Implementation:** Ensure that the recommended mitigation strategies are implemented promptly and effectively to address the identified vulnerabilities.

By carefully analyzing the XSS results and considering these factors, you can gain a comprehensive understanding of the security posture of the web application and take appropriate measures to mitigate XSS risks.

The provided results likely showcase the outcomes of a web application penetration testing session specifically designed to identify and exploit cross-site scripting (XSS) vulnerabilities. These results typically include identified vulnerable parameters, injected payloads, potential impact, and recommended mitigation strategies. By carefully analyzing these elements and considering factors such as testing methods, false positives, and false negatives, you can gain a comprehensive understanding of the web application's security posture and take appropriate measures to mitigate XSS risks.

- **Wireshark Analysis:**
    - The network traffic captured during the penetration test provided valuable insights into the attack process. SQL injection traffic showed the injection of SQL payloads and the corresponding database responses. XSS attack traffic revealed the injection and execution of malicious scripts. The Wireshark analysis confirmed the success of the attacks and highlighted the types of data exchanged between the attacker and the server.

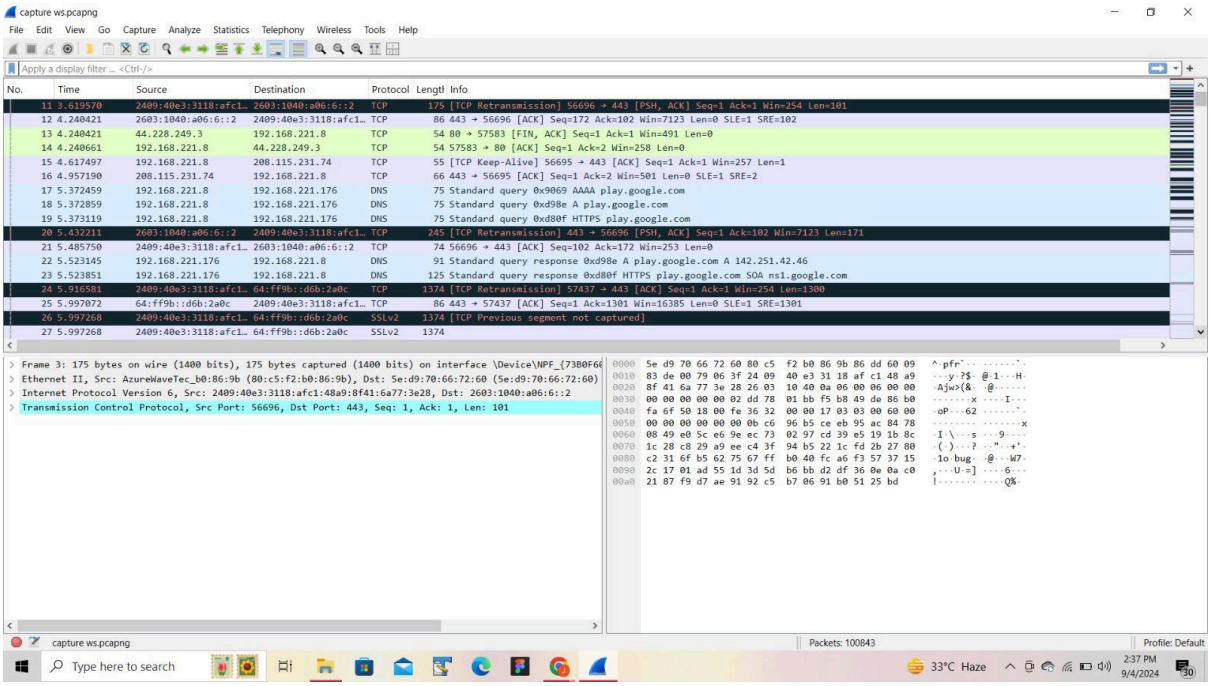

Fig 21. Wireshark capture showing network traffic during a web application penetration testing session, focusing on SQL injection and XSS vulnerabilities

The Wireshark capture reveals network traffic from a web application penetration testing session focusing on SQL injection and XSS vulnerabilities. By analyzing malicious input, error messages, and injected payloads, potential vulnerabilities can be identified and exploited. The capture can also be used for attack reconstruction and security awareness training.

- **Malicious Input:** The capture may show crafted input that exploits SQL injection vulnerabilities, such as injecting SQL commands into query parameters.
- **Error-Based SQL Injection:** Look for error messages that reveal sensitive information or database structure, indicating successful SQL injection attacks.
- **Blind SQL Injection:** If error-based SQL injection is not possible, blind SQL injection techniques might be used, which can be identified by analyzing timing differences or other subtle indicators.
- **XSS Payload Injection:** The capture might show XSS payloads being injected into web pages, such as JavaScript code or HTML tags.

- **Reflected XSS:** If the injected XSS payload is reflected back to the user's browser, it can be identified in the HTTP response.
- **Stored XSS:** If the XSS payload is stored on the server and later executed when a user visits the affected page, it might be more difficult to detect in the capture.

Additional Analysis Techniques:

- **HTTP Request and Response Headers:** Examine the HTTP headers for information such as the content type, cookies, and server-side scripting language. These can provide clues about potential vulnerabilities.
- **URL Encoding:** Analyze how URL-encoded parameters are handled by the application. Incorrect decoding or validation can lead to vulnerabilities.
- **Cookie Handling:** Check how cookies are set, transmitted, and stored. Improper cookie handling can expose sensitive information or enable session hijacking.
- **Session Management:** Examine how session IDs are generated, transmitted, and stored. Weak session management can make the application vulnerable to session fixation attacks.

Mitigation Strategies:

- **Input Validation:** Implement strict input validation to prevent malicious input from being processed by the application.
- **Parameterized Queries:** Use parameterized queries to prevent SQL injection by separating SQL code from user-provided data.
- **Output Encoding:** Encode output to prevent XSS attacks by rendering special characters as HTML entities.
- **Secure Session Management:** Use strong session IDs and implement appropriate session timeout and invalidation mechanisms.
- **Regular Security Testing:** Conduct regular penetration testing and vulnerability assessments to identify and address potential vulnerabilities.

By carefully analyzing the Wireshark capture and employing these additional techniques, you can gain a deeper understanding of the network communication related to web application security and take effective measures to mitigate vulnerabilities.

CHAPTER - 3

3.1CONCLUSION:

The penetration testing of the web application hosted at http://testphp.vulnweb.com revealed significant security vulnerabilities, including SQL Injection and Cross-Site Scripting. The use of SQLMap and XSSer tools effectively identified and exploited these vulnerabilities, leading to unauthorised access to sensitive data and the ability to execute malicious scripts within users' browsers. Wireshark analysis provided a detailed view of the attack traffic, confirming the nature and extent of the vulnerabilities. It is strongly recommended that the web application implement robust input validation, output encoding, and other security best practices to mitigate these risks.

3.2REFERENCES:

1. Web Application Penetration Testing Using SQL Injection Attack, Alde Alandaa,*, Deni Satriaa, M.Isthofa Ardhanaa, Andi Ahmad Dahlanb, Hanriyawan Adnan Moodutoa, september 2021.
2. Deep Learning-Based Detection Technology for SQL Injection Research and Implementation,by Hao Sun 1,Yuejin Du 2 andQi Li 1,**Appl. Sci.* 2023.
3. A Survey on Web Application Penetration Testing, by Esra Abdullatif Altulaihan *, Abrar Alismail and Mounir Frikha, *Electronics* 2023.