CONESTOGA

Connect Life and Learning

PROJECT: SQL MAP

COMPUTER SECURITY APPLICATION

Applied Computer Science & IT

Onyeka Mmesooma Ofojetu-9005016

Course Instructor: Baljeet S. Bilkhu

TABLE OF CONTENTS

INTRODUCTION

SQLMap is an offensive security application to replicate the attacks to the web application. It is created to discover and exploit SQL injection vulnerability, which is a critical vulnerability that might present an attacker with the ability to read or control the database of a web site. However, even during the time when SQLMap is launched in attacks, it is broadly utilized by white hat hackers and penetration testers that help to discover security-related vulnerabilities before the actual attackers can discover reported vulnerabilities and exploit them. It is vital to make it a powerful tool when enhancing the security of web application by performing responsible and moderated testing.

1.1 DESCRIPTION OF SQLMAP AS A TOOL

SQLMap is a free, open-source penetration testing tool used to detect and exploit SQL injection vulnerabilities in web applications. Written in Python, it is cross-platform and runs on Linux, Windows, and macOS, with a pre-installed version in Kali Linux making it a go-to resource for cybersecurity professionals. Operating entirely from the command line, SQLMap can automate the complete SQL injection process, from identifying vulnerabilities to extracting database information, which is valuable for penetration testers, red teams, and ethical hackers conducting realistic attack simulations.

The tool supports multiple database management systems, including MySQL, Oracle, Microsoft SQL Server, PostgreSQL, and SQLite, and can connect directly to databases if valid credentials and IP details are provided. The latest release, **v1.9.8-1**, is available on its official GitHub repository, with ongoing updates introducing new attack methods and enhancements to keep pace with evolving security threats.

1.2 WHAT SQL MAP IS USED FOR IN CYBERSECURITY

SQLMap is a penetration testing tool for detecting and exploiting SQL injection vulnerabilities in web applications. It supports multiple attack techniques, including error-based, time-based, boolean-based blind, and out-of-band methods. Once a vulnerability is found, it can extract database contents such as usernames, passwords, and sensitive records, and if database privileges allow read/write server files or execute system commands.

Although designed as an offensive tool, SQLMap is widely used by ethical hackers and penetration testers to identify vulnerabilities before malicious actors exploit them. Its applications include demonstrating how attackers could compromise or destroy data, helping organizations remediate flaws, and training students and professionals in safe, legal hacking practices (SQLMap Developers, 2025).

1.3 THE KEY FEATURES OF SQL MAP

SQLMap is a powerful open-source penetration testing tool that automates the detection and exploitation of SQL injection vulnerabilities in web applications (Kali Linux, 2024). It supports a variety of injection techniques, including Boolean-based blind, time-based blind, error-based, UNION query-based, and out-of-band methods (SQLMap Developers, 2025). The tool can fingerprint backend database management systems (DBMS) such as MySQL, Oracle, PostgreSQL, Microsoft SQL Server, and SQLite, enabling precise and tailored exploitation (SQLMap Developers, 2025).

With appropriate privileges, SQLMap allows full database takeover—reading and writing data, accessing the underlying file system, and executing arbitrary system commands. It can crack extracted password hashes using tools like John the Ripper or Hashcat (Checkpoint, 2012) and enumerate users, roles, privileges, and schema details for reconnaissance and post-exploitation (Bilkhu, 2025).

Additional features include:

- **Tamper script support** to obfuscate payloads and bypass Web Application Firewalls (WAFs) or IDS/IPS.
- **Integration with other tools** like Burp Suite for advanced testing workflows.
- **Support for multiple injection points** in URLs, POST data, and HTTP headers (e.g., cookies, User-Agent).
- **Batch mode** for automated testing of multiple URLs or parameters.
- **Cross-platform compatibility**, running on Linux, Windows, and macOS (SQLMap Developers, 2025).
- **Customizable verbosity and output options**, allowing testers to fine-tune scanning and reporting.
- **Proxy and Tor support** to route traffic anonymously during engagements.

These capabilities make SQLMap a staple in penetration testing and ethical hacking for simulating real-world attacks and identifying vulnerabilities before malicious actors can exploit them.

1.4 WHAT TYPES OF ORGANIZATIONS/COMPANIES WOULD USE SQLMAP?

SQLMap is used across multiple industries to assess and improve the security of web applications against SQL injection attacks. **Cybersecurity consulting firms** integrate it into penetration testing engagements to find and exploit vulnerabilities for clients (SQLMap Developers, 2025). **Financial institutions** such as banks and fintech companies rely on SQLMap to protect sensitive customer data and ensure compliance with standards like PCI-DSS. **E-commerce and online service providers** use it in security audits to identify insecure input handling in forms, search fields, and checkout processes.

Government agencies and defense contractors apply SQLMap in red team exercises and vulnerability assessments to secure mission-critical web systems and meet regulations such as FISMA and NIST. **Managed Security Service Providers (MSSPs)** employ it within broader vulnerability management offerings, often pairing it with reporting and remediation services. **Universities and research institutions** leverage SQLMap for academic research, ethical hacking training, and simulated real-world attack labs (Kali Linux, 2024).

Additional uses include:

- **Internal security teams** performing routine web application security checks.
- **Compliance audits** where injection testing is required for certification or regulatory purposes.
- **Bug bounty programs**, where researchers use SQLMap to responsibly disclose vulnerabilities.

1.5 ROLES THAT USE SQL MAP IN AN ORGANIZATION

- **Penetration Testers (Red Team)** – Use SQLMap to automate SQL injection testing, extract sensitive data, and combine with tools like Burp Suite or Nmap for full-stack exploitation. It saves significant time compared to manual testing (Vaadata, 2022; Hacker Target, 2021; Kali Linux, 2024).

- **Security Researchers** – Employ SQLMap for rapid vulnerability testing, proof-of-concept demonstrations, and automating repetitive SQL injection checks across multiple projects (CyberTalents, 2022).
- **Blue Team / SOC Analysts** – Simulate attacker SQLMap usage to study attack patterns, fine-tune WAF/IDS configurations, and adjust logging/alert systems for better defense (Springboard, 2023).
- **DevSecOps** – Integrate SQLMap into CI/CD pipelines for automated security testing at each deployment, ensuring early vulnerability detection and maintaining CIA principles (Red Hat, 2023).
- **IT Auditors / Compliance Officers** – Include SQL injection testing results in compliance reports, verify systems are free from SQLi vulnerabilities, and ensure adherence to industry regulations.
- **Cybersecurity Trainers** – Use SQLMap in lab sessions to teach students real-world SQL injection methods, demonstrate exploits, and provide prevention techniques.

1.6 COMPETITOR TOOLS

- **Havij** – GUI-based automated SQL injection tool, beginner-friendly, and good for quick, basic tests. Limited DBMS support, fewer customization options, and lacks WAF bypass features (WebAsha, 2021).
- **JSQL Injection** – Open-source Java-based tool with GUI, supports GET, POST, and header injections, and offers multiple payload templates. Less effective for advanced exploitation, chained queries, and WAF evasion (LinuxSecurity, 2022).
- **NoSQL Map** – Targets NoSQL databases like MongoDB, useful for testing REST APIs and NoSQL endpoints, but not applicable to relational SQL databases (SecOps Group, 2023).

What SQLMap Can Do That Competitors Cannot

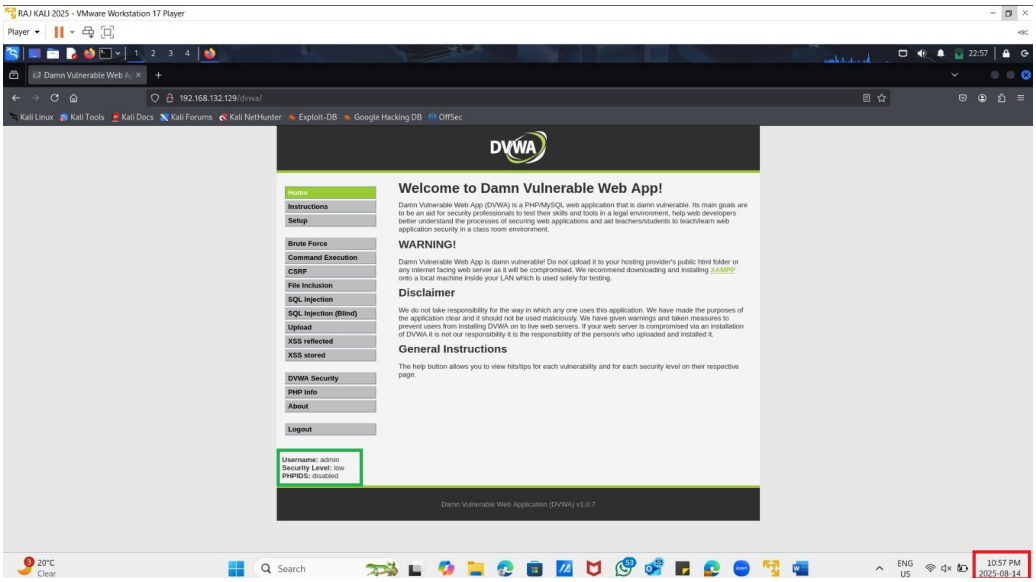
- **Comprehensive DBMS Support** – Works with nearly all major and minor relational databases, including MySQL, SQL Server, Oracle, PostgreSQL, SQLite, Sybase, Informix, SAP MaxDB, HSQLDB, and Firebird.
- **Advanced Exploitation Automation** – Fully automates detection, fingerprinting, enumeration, file system access, privilege escalation, and shell spawning.
- **Tamper Scripts for WAF Evasion** – Customizable scripts to obfuscate payloads, bypass WAFs like Cloudflare or AWS WAF, and adapt to specific targets.
- **CLI and Automation Integration** – Easily integrates into CI/CD pipelines for automated SQLi testing, unlike most GUI-focused tools.
- **Custom Exploitation Modes** – Allows selection of injection points (GET, POST, headers, cookies) and injection types (error-based, time-based, etc.) for precise testing.

2.1 ATTACK DEMONSTRATION USING SQLMAP

This was intended to demonstrate (with help of Kali VM and SQLMap) SQL injection into a DVWA (Damn Vulnerable Web App) running on Metasploitable 2, and then demonstrate an actual compromise by extracting user credentials and logging in with one of the stolen accounts. All this was performed on a host only network and traffic never left the laboratory.

2.1.1 STEP 1 — REACHING THE TARGET AND LOGGING IN TO DVWA

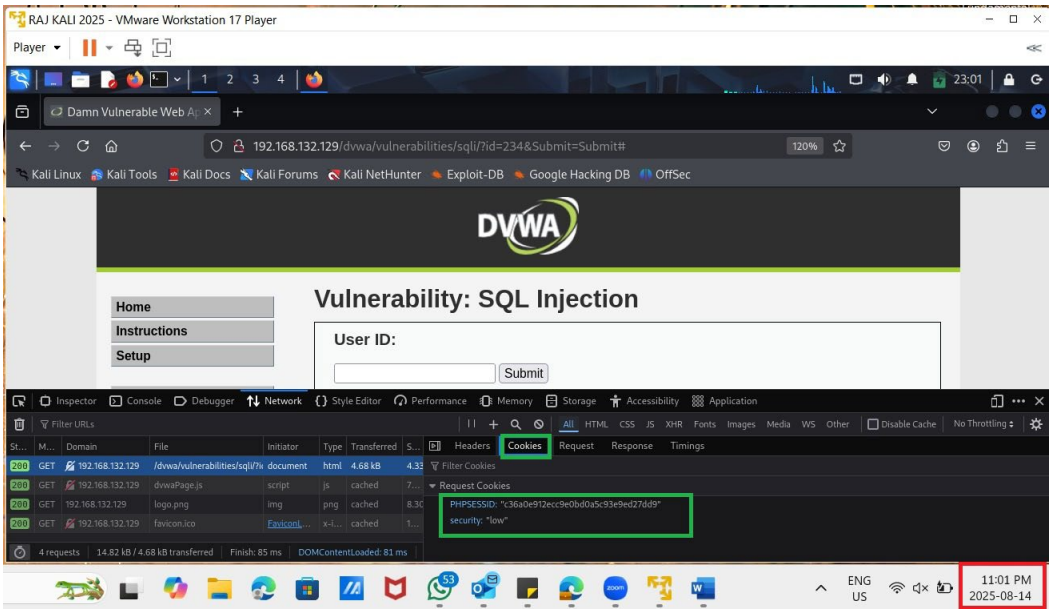
Both VMs were booted in host-only mode, and connectivity to the DVWA web app on Metasploitable 2 was confirmed from Kali. Using Firefox, the page http://192.168.132.129/dvwa/ was accessed and logged into with default credentials (admin / password), establishing an authenticated session required to access the vulnerable page.



Screenshot 2.1: (DVWA Home – admin logged in): DVWA dashboard confirming successful authentication on the target.

2.1.2 STEP 2 — OPENING THE VULNERABLE PAGE AND CAPTURING OUR SESSION COOKIE

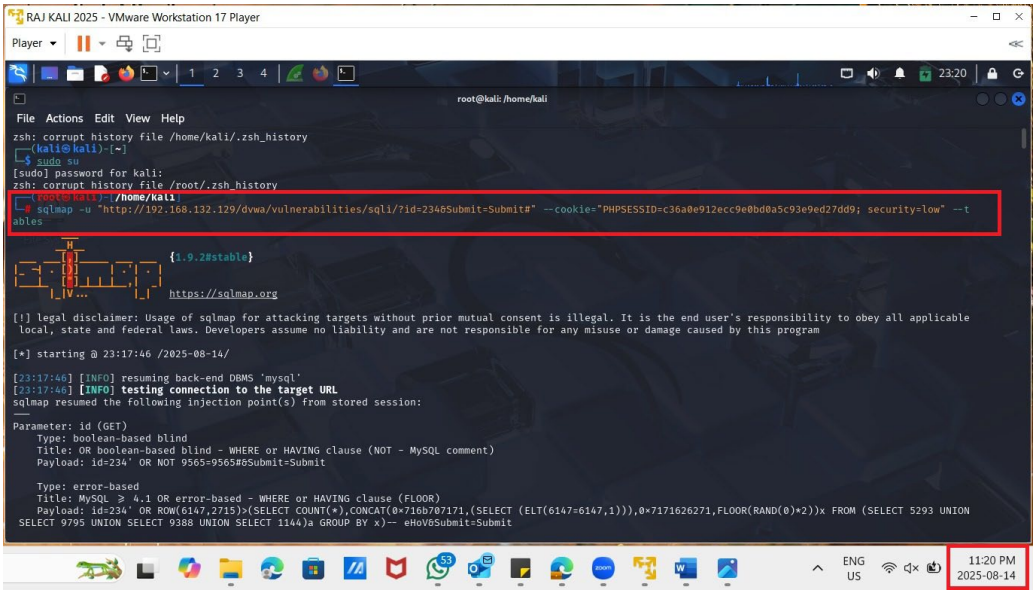
In DVWA, the SQL Injection page was accessed, and a test User ID (e.g., 234) was submitted while monitoring the request in Developer Tools. The PHPSESSID cookie was copied, and the DVWA security level confirmed as low, providing the session details needed for SQLMap to mimic an authenticated browser session.



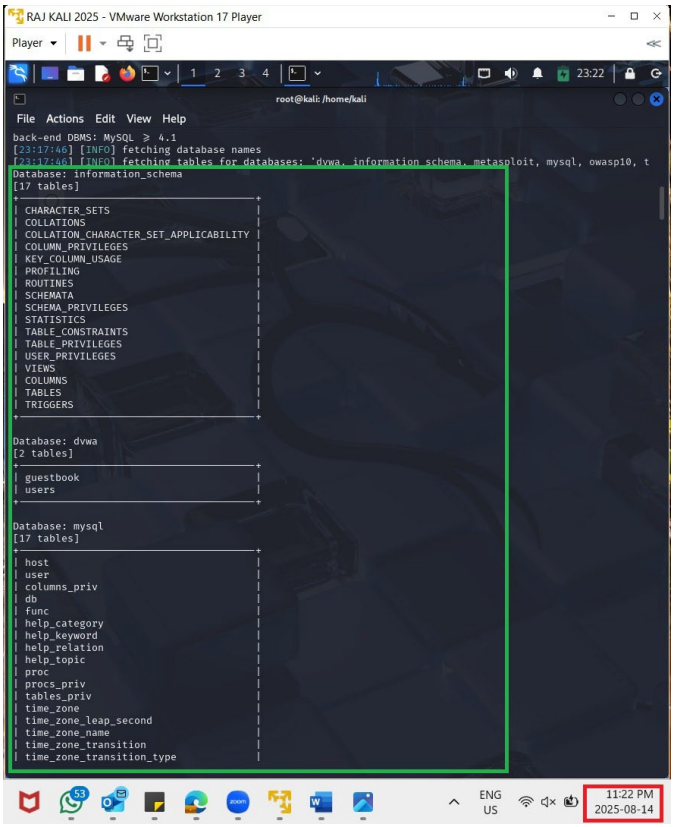
Screenshot 2.2: (SQL Injection page + Cookies): The vulnerable page is visible and DevTools shows PHPSESSID= c36a0e912ecc9e0bd0a5c93e9ed27dd9; security=low.

2.1.3 STEP 3 — CONFIRMING INJECTION AND ENUMERATING TABLES WITH SQLMAP

Using the captured cookie, SQLMap was run from the Kali root shell against the DVWA SQL Injection URL and id parameter. The tool quickly confirmed the parameter was injectable through boolean, blind, error, and time-based techniques, proving that untrusted input was directly appended to SQL queries on the server. Following verification, SQLMap automatically enumerated accessible database structures, revealing how easily sensitive information could be mapped and targeted.



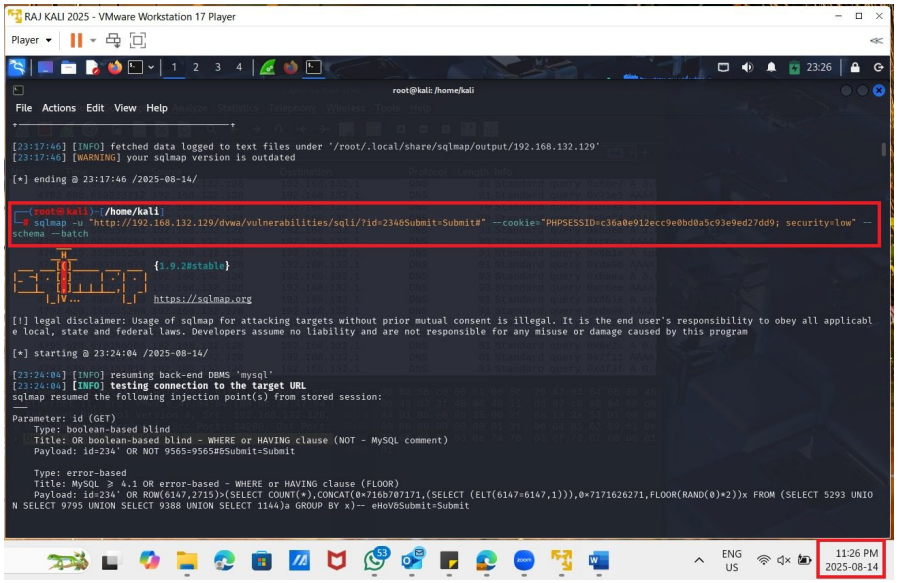
Screenshot 2.3: To enumerate all tables on the vulnerable app, we ran “sqlmap -u "http://192.168.132.129/dvwa/vulnerabilities/sqli/?id=234&Submit=Submit#" --cookie="PHPSESSID= c36a0e912ecc9e0bd0a5c93e9ed27dd9; security=low" --tables".



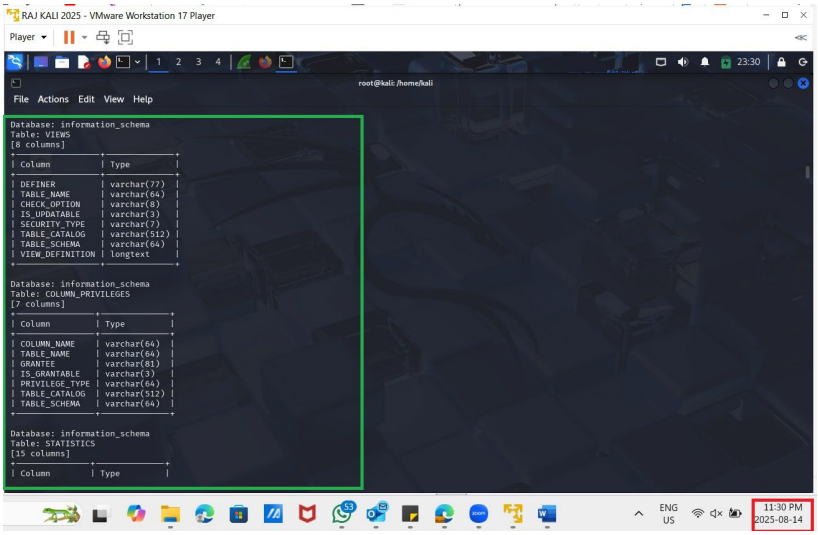
Screenshot 2.4: SQLMap lists databases such as dvwa, information_schema, and mysql, showing real enumeration of the backend.

2.1.4 STEP 4 — PULLING THE SCHEMA TO SEE THE BIG PICTURE

To view the database structure, SQLMap was run with the --schema and --batch options, using the authenticated session cookie. The scan confirmed the cookie was valid and allowed access to information_schema, enabling SQLMap to list databases, tables, and columns. This provided a clear overview of the data organization and helped identify sensitive tables for further analysis.



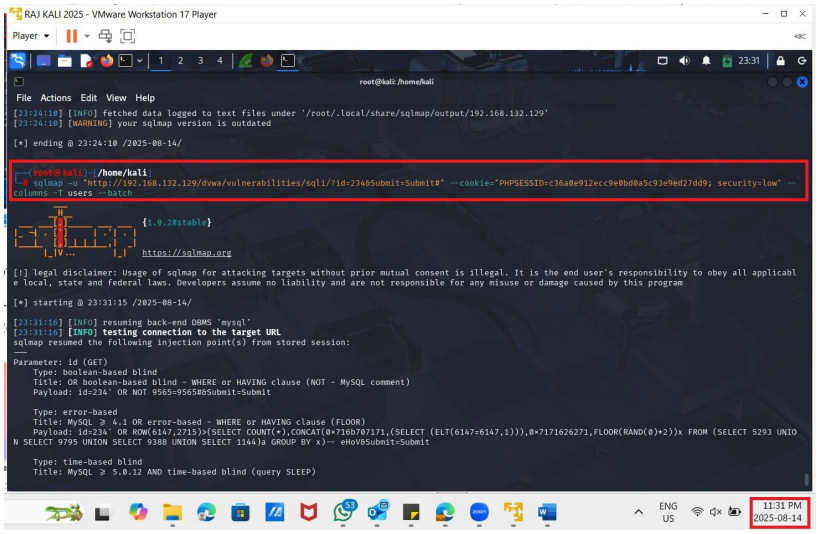
Screenshot 2.5: To pull the full database schema while accepting defaults, we ran "sqlmap -u "http://192.168.132.129/dvwa/vulnerabilities/sqli/?id=234&Submit=Submit#" --cookie="PHPSESSID=c36a0e912ecc9e0bd0a5c93e9ed27dd9; security=low" --schema --batch".



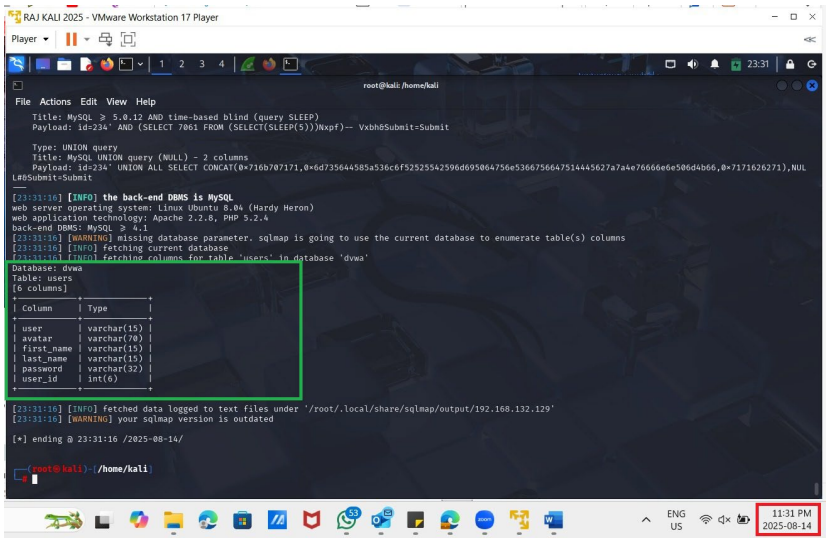
Screenshot 2.6: Output from information_schema showing table/column metadata that our app user can read.

2.1.5 STEP 5 — TARGETING THE SENSITIVE TABLE: DVWA.USERS (COLUMN DISCOVERY)

Focusing on the dvwa.users table, SQLMap was run with the --columns option to list its structure using the authenticated session cookie. The tool returned columns including user, first_name, last_name, password, user_id, and avatar, confirming that credential-related data was stored in this table and accessible via the injectable parameter.



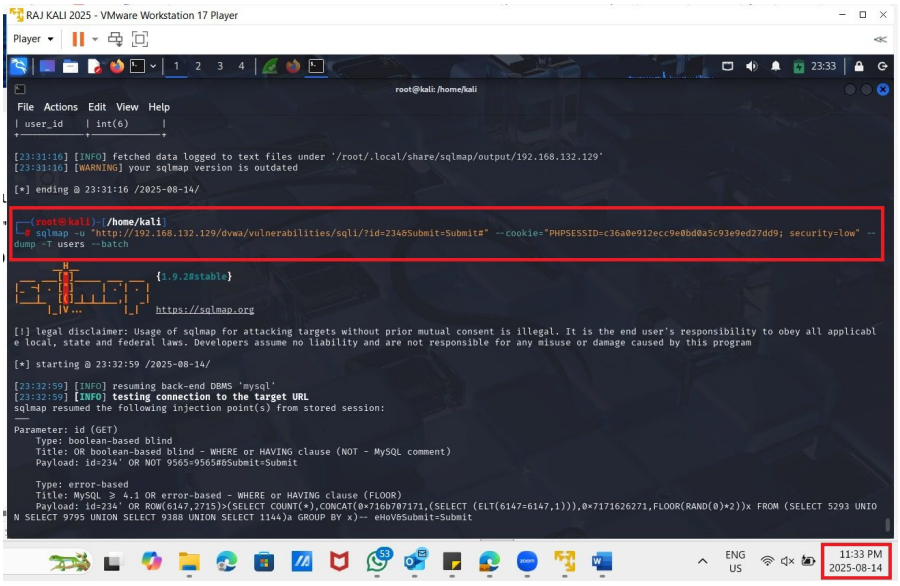
Screenshot 2.7: Out To list the columns in the dvwa.users table, we ran "sqlmap -u "http://192.168.132.129/dvwa/vulnerabilities/sqli/?id=234&Submit=Submit#" --cookie="PHPSESSID=c36a0e912ecc9e0bd0a5c93e9ed27dd9; security=low" --columns -T users --batch".



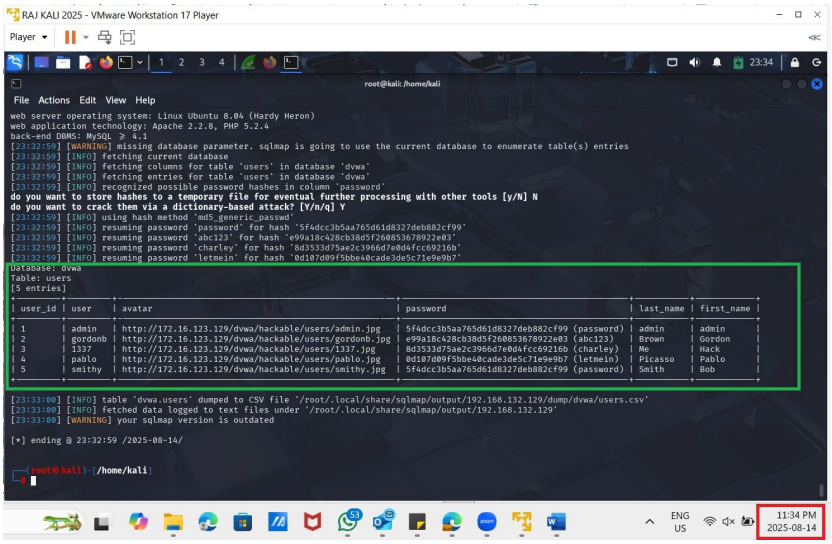
Screenshot 2.8: Shows the column names and data types, including the password column.

2.1.6 STEP 6 — DUMPING DVWA.USERS AND CRACKING COMMON PASSWORDS

Using the --dump option on the dvwa.users table, SQLMap identified the backend as MySQL, retrieved all rows, and detected password hashes. It successfully matched several hashes to plaintext credentials, recovering accounts such as admin:password, gordonb:abc123, 1337:charley, pablo:letmein, and smithy:password. The results were saved in SQLMap's default output directory for record-keeping.



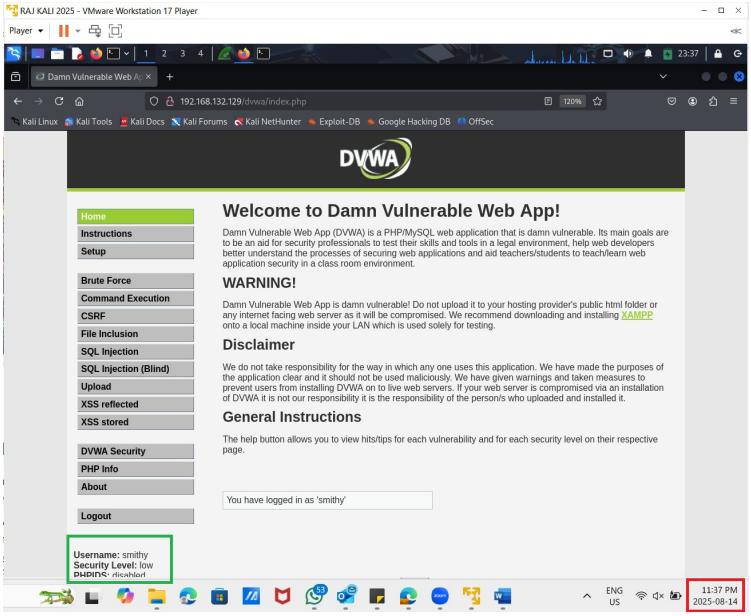
Screenshot 2.9: To dump the contents of the dvwa.users table (and let SQLMap try simple cracking), we ran "sqlmap -u "http://192.168.132.129/dvwa/vulnerabilities/sqli/?id=234&Submit=Submit#" --cookie="PHPSESSID= c36a0e912ecc9e0bd0a5c93e9ed27dd9; security=low" --dump -T users --batch".



Screenshot 2.10: Full dvwa.users dump with hashes and matched plaintexts beside them.

2.1.7 STEP 7 — PROVING IMPACT BY LOGGING IN WITH A STOLEN USER

To display actual damage other than merely as a method of reading data, we entered one captured credential through the typical web interface of DVWA. We could find smithy / password. The attempt to log in worked and DVWA gave an output of “You have logged in as smitty.” That serves to demonstrate that it was no mere database read with that injection--it facilitated complete account access within the application.

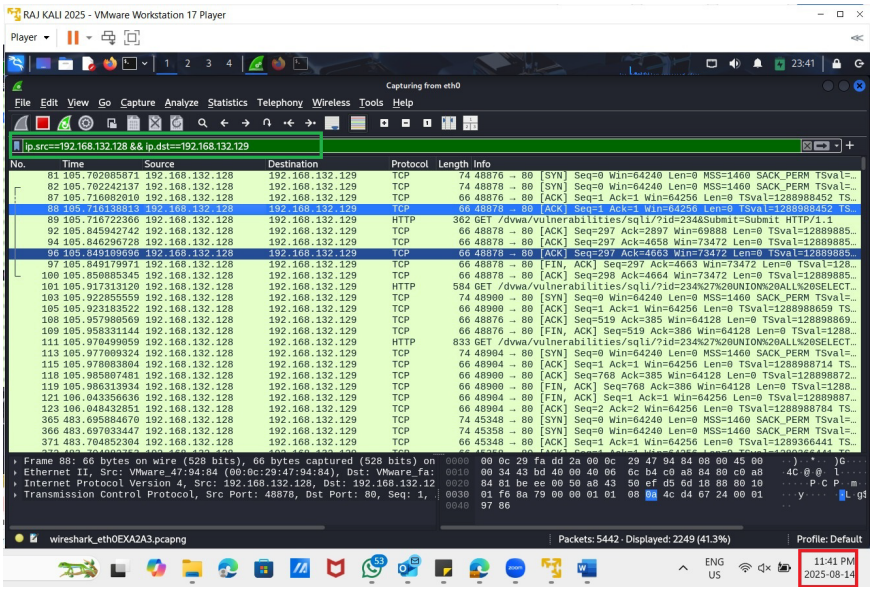


Screenshot 2.11: DVWA home shows the status line confirming we’re logged in as smithy.

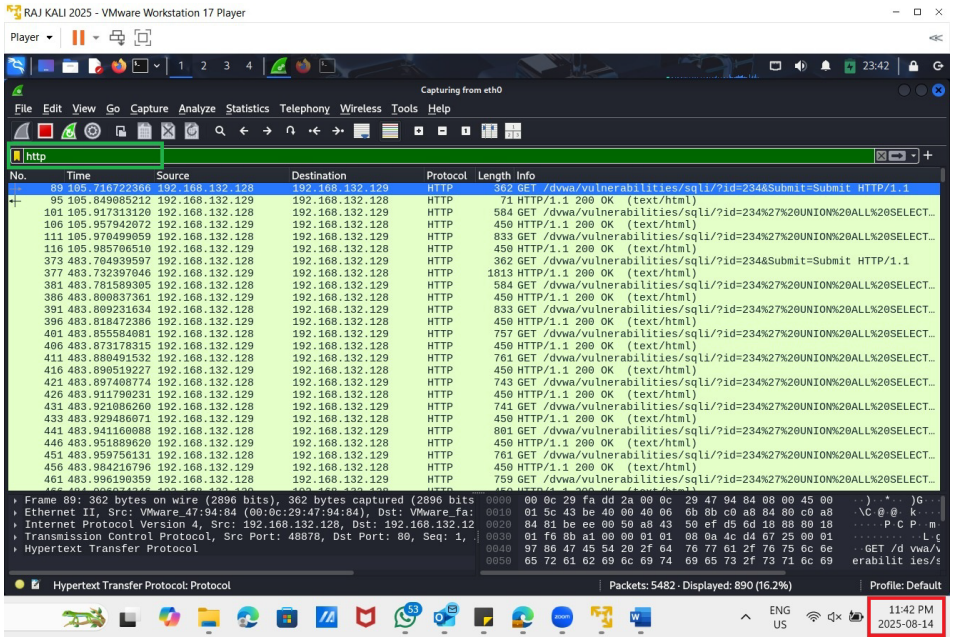
2.2 DEFENCE DEMONSTRATION AGAINST SQLMAP

The task showcases layered defense by using Wireshark to detect suspicious traffic to DVWA and UFW to block the attacker’s IP and restrict vulnerable ports. This simulates a real-world process of detecting threats and responding quickly with firewall rules to prevent further attacks.

2.2.1 WIRESHARK DEFENCE DEMONSTRATION



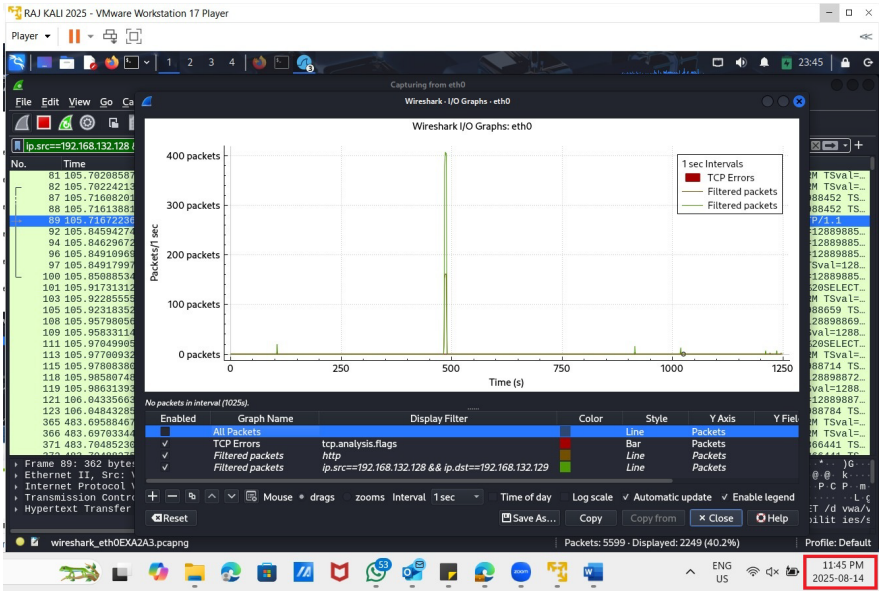
Screenshot 2.12: *Wireshark capture showing HTTP traffic from 192.168.132.128 (attacker) to 192.168.132.129 (target) involving multiple GET requests to DVWA resources over TCP port 80. This confirms the enumeration or attack attempts over an unencrypted channel.*



Screenshot 2.13: *Wireshark packet capture showing HTTP traffic from 192.168.132.128 to 192.168.132.129, interacting with the DVWA application. HTTP requests include resource fetches, login attempts, and form submissions, all transmitted in plaintext.*

The Wireshark capture filters traffic from **192.168.132.128** to **192.168.132.129**, revealing unencrypted HTTP requests to the Damn Vulnerable Web Application (DVWA) over TCP port 80, including /dvwa and /dvwa/login.php. This exposes the traffic to sniffing or MITM attacks. A selected packet confirms an attacker's GET request, and the source's ephemeral port (58882) reflects standard client-side outbound communication.

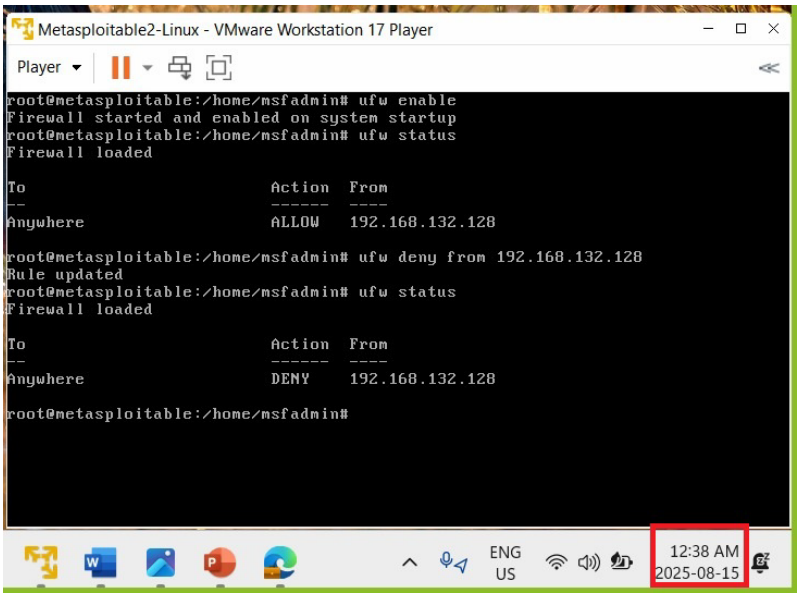
The Wireshark capture shows unencrypted HTTP traffic from **192.168.132.128** to **192.168.132.129**, targeting the Damn Vulnerable Web Application (DVWA) with requests to /dvwa/, /login.php, and other resources. This exposes risks such as credential leakage, interception of clear-text data, and unauthenticated access to insecure endpoints—indicating possible SQL injection or brute-force attack attempts.



Screenshot 2.14: Wireshark I/O graph showing filtered packet activity from 192.168.132.128 to 192.168.132.129, with a high concentration of HTTP packets around the 300-second mark, and TCP errors indicating possible abnormal behavior. Filters used include HTTP traffic and specific IP communication, supporting targeted traffic analysis during incident response or threat identification.

The Wireshark I/O graph shows filtered network activity for TCP errors (red), traffic from **192.168.132.128** to **192.168.132.129** (green), and HTTP packets (brown). A notable spike between 280–320 seconds indicates a burst of HTTP traffic with anomalies such as retransmissions, suggesting possible automated scanning or attacks on the /dvwa application. TCP errors point to failed handshakes or rejected packets, consistent with targeted reconnaissance or exploitation attempts.

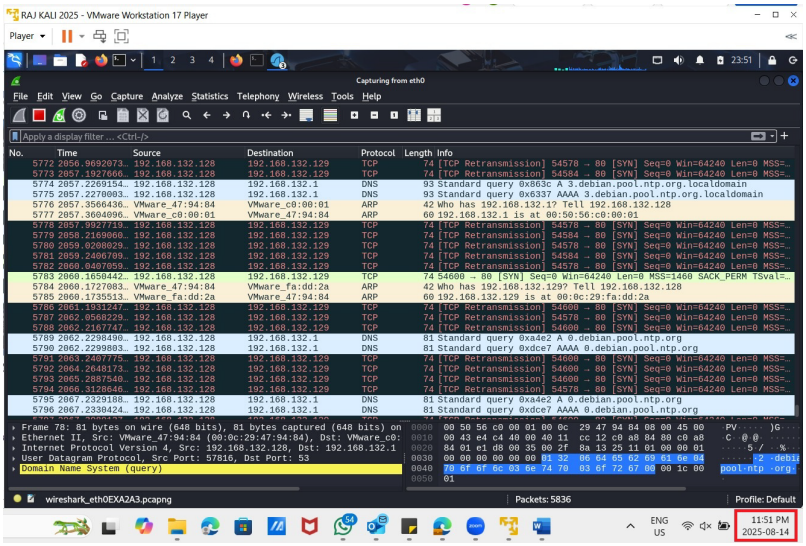
2.2.2 UFW FIREWALL DEFENCE DEMONSTRATION



Screenshot 2.15: Firewall rule implementation on Metasploitable2 VM using ufw to block incoming traffic from suspicious IP 192.168.132.128, identified through Wireshark traffic analysis.

On a Metasploitable2 VM, UFW was enabled and configured to deny all incoming connections from IP **192.168.132.128** after it was flagged in Wireshark as a potential attack source. The ufw status output confirmed the rule was active, effectively blocking further malicious activity and strengthening the system's security.

2.2.3 WIRESHARK SHOWING DENY OF TRAFFIC FROM ATTACKER



Screenshot 2.16: Wireshark capture showing blocked TCP connection attempts and multiple retransmissions from attacker IP 192.168.132.128 to target IP 192.168.132.129, indicating that the UFW rule is actively denying access.

The Wireshark capture shows numerous TCP retransmissions and repeated SYN packets from attacker IP **192.168.132.128** to victim IP **192.168.132.129**, caused by UFW firewall rules blocking the traffic. This indicates the victim is dropping connection attempts. While DNS queries from internal and external sources are present, no successful HTTP communication is detected.

2.3 LIMITATIONS OF SQLMAP

1. **Focuses only on SQL Injection**

SQLMap works only for SQL Injection (SQLi). It will not detect:

- Cross-Site Scripting (XSS)
- Remote File Inclusion (RFI)
- Cross-Site Request Forgery (CSRF)
- Authentication flaws
- Server misconfigurations

The entire code base of SQLMap is made to construct SQL payloads, detect injection points, and automate the attack against databases — rather than searching other vulnerability types.

2. **Noisy and Easy to Detect**

SQLMap can execute various SQL injection types, including boolean-based, time-based, and error-based attacks. During scans, it sends a large volume of requests to the database server to test different payloads, often reaching hundreds of requests per second compared to just a few in manual testing. This high-intensity activity generates significant network noise, making it easily detectable in monitoring tools like Wireshark. As a result, security analysts can quickly spot the abnormal traffic patterns and block the attacker's IP address to halt the SQL injection attempt.

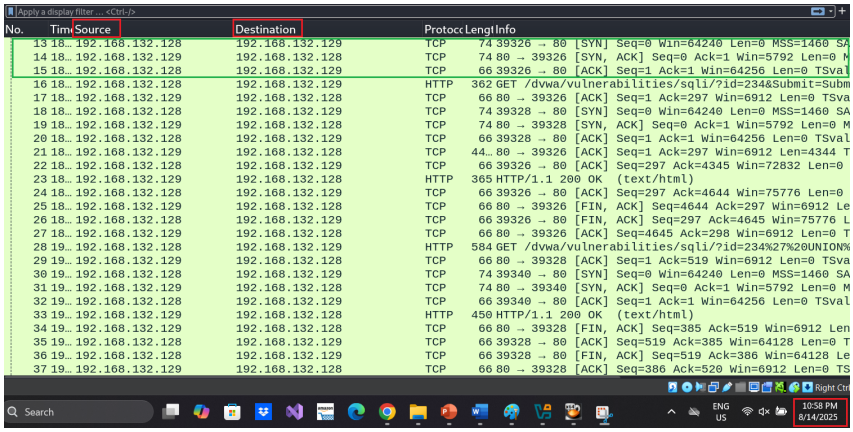


Fig 2.17: Packet capture in Wireshark of the network traffic between Kali Linux and Metasploitable

The packet capture shows repeated TCP handshakes between Kali Linux (**192.168.132.128**) and Metasploitable (**192.168.132.129**) over HTTP port 80, starting with a standard SYN → SYN-ACK → ACK sequence. The high volume of successful handshakes indicates SQLMap activity against DVWA, which could alert security analysts to suspicious traffic and lead them to block the source IP in the firewall, stopping the SQL injection attempt.

3. **Dependent on Database User Privileges**

SQLMap works by exploiting SQL injection to act as the database user used by the web application. Its capabilities are limited to the privileges of that account. If the account has restricted rights such as access to only one table or read-only permissions SQLMap cannot list all databases, dump unrelated tables, read system files, run OS commands, or escalate privileges. It can only perform actions the current database user is authorized to do, making privilege level a key factor in the extent of the exploitation.

Fig 2.19 shows the output of the SQL injection using the SQLMap tool before any changes in the privileges in the dvwa database. It can be seen that two tables (guestbook and users) are listed for the dvwa data

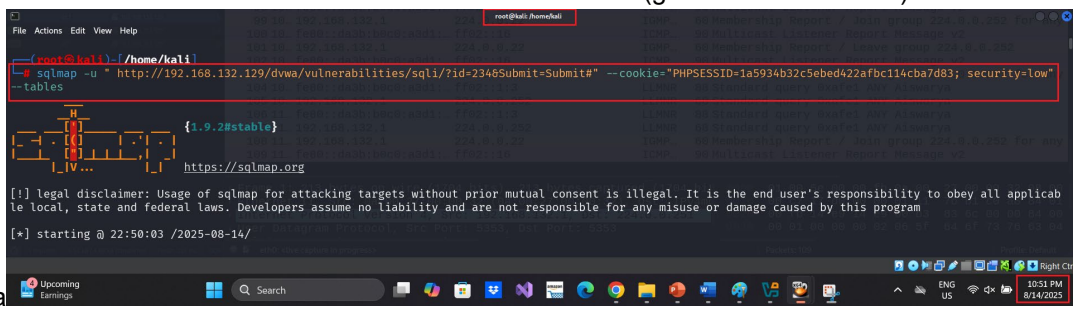


Fig 2.18: SQLMap command to see the tables in DVWA Database tables

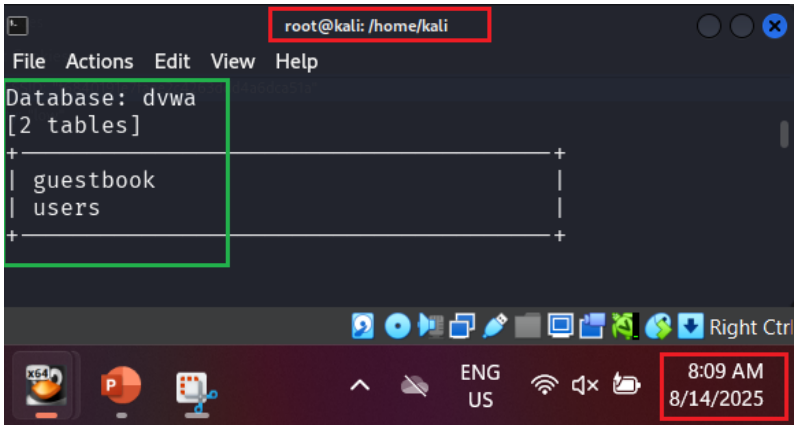


Fig 2.19: DVWA Database tables shown on low security level before privilege escalation on the database.

Then, tried to change the privileges in the dvwa database. The change was that only ‘users’ table can be selected in the dvwa database. (Fig 2.20)

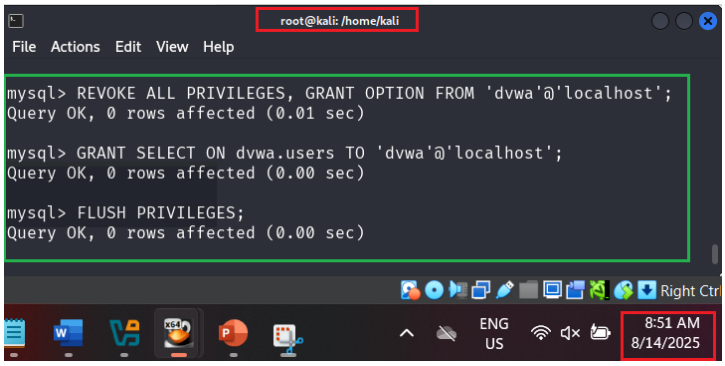


Fig 2.20: Privilege escalation on the DVWA database

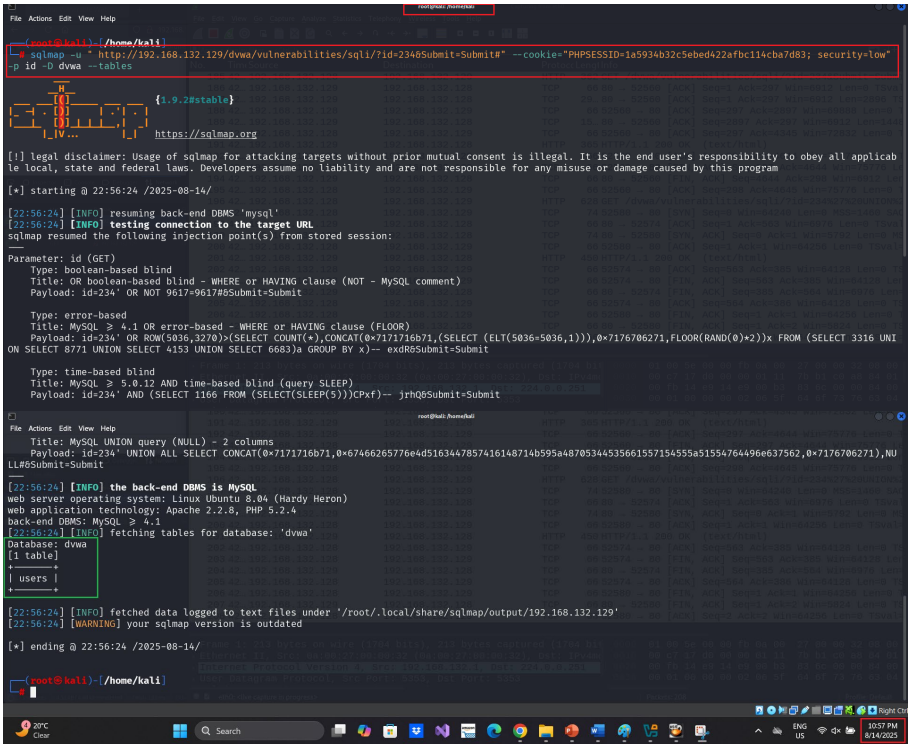


Fig 2.21: Only ‘users’ table can be seen after the privilege changes

From Fig 2.21 it can be understood that after changing the privileges in the dvwa database in such a way that only ‘users’ table can be selected, then trying the SQL injection only ‘users’ tables are being displayed. So, as said above, the SQL injection attack using the SQLMap tool highly depends on the privileges the database user has open the database. kali

3.1 EMERGING TOOL

AI-WAFs are advanced web application firewalls enhanced with Artificial Intelligence (AI) and Machine Learning (ML) to detect and block attacks in real time. Unlike traditional WAFs that rely solely on fixed rules and known attack signatures, AI-WAFs analyze live traffic behavior, learn from normal and malicious patterns, and adapt to block unknown or zero-day attacks automatically (Prophaze, 2024; Akamai, 2023).

3.2 IMPACT ON SQLMAP

SQLMap, a widely used automated SQL injection tool, faces increasing difficulty against AI-WAFs. These intelligent firewalls detect behaviors typical of SQLMap—such as high-frequency parameter tampering, suspicious user agents, or rapid request bursts—through behavioral modeling and anomaly detection (Cloudflare, 2023). Solutions from vendors like Prophaze emphasize zero-day protection by predicting threats without relying on static rules (Prophaze, 2024), Akamai applies adaptive modeling to detect subtle SQL injection attempts (Akamai, 2023), and Imperva integrates AI-driven bot detection to block automated exploitation tools based on patterns and intent (Imperva, 2024). This significantly reduces the success rate of common payloads and automated scans.

3.3 OUTLOOK

While AI-WAFs enhance security posture, they are not foolproof. Skilled attackers may adapt by slowing requests, rotating IPs/user-agents, or using tamper scripts to disguise payloads. The most effective defense remains layered secure coding practices such as parameterized queries, strict input validation, least-privilege database accounts, and continuous monitoring through SIEMs should complement AI-WAF deployment (Cloudflare, 2023). Over the next 1–3 years, AI-WAFs will likely integrate with API gateways, in-app protections, and database firewalls, while offensive tools evolve to use AI-generated payloads and API-focused attacks (Prophaze, 2024; Akamai, 2023).

This dynamic between AI-WAFs and automated exploitation tools like SQLMap represents an ongoing cybersecurity arms race requiring adaptive, multi-layered defense strategies to stay ahead.

CONCLUSION

This project covered the full cycle of web application security testing using SQLMap to exploit SQL injection in DVWA, applying UFW to block malicious traffic, and exploring AI-based Web Application Firewalls (AI WAFs) as an advanced defense. The **attack phase** showed SQLMap's efficiency in automating SQLi exploitation, while also revealing its noisy traffic and reduced effectiveness against hardened sites. The **defense phase** demonstrated how UFW limits attacker access by blocking specific IPs, reinforcing the role of network controls alongside secure coding. Research on **AI WAFs** highlighted their ability to learn from traffic patterns, detect anomalies, and adapt to evolving threats, potentially stopping SQLMap probes in real time though they still face challenges like false positives and the need for continual updates.

REFERENCES

Beaver, K. (2018, September 13). *Understanding the risk SQL injection vulnerabilities pose*. Retrieved from TechTarget: https://www.techtarget.com/searchsecurity/feature/Understanding-the-risk-SQL-injection-vulnerabilities-pose

Bilkhu, B. (2025). *SECU8020_Lesson04_SQLMap*. Retrieved from eConestoga: https://conestoga.desire2learn.com/d2l/le/content/1426757/viewContent/29847513/View

Bugcrowd. (n.d.). *Security researcher*. Retrieved from Bugcrowd: https://www.bugcrowd.com/glossary/security-researcher/

Checkpoint. (2012). *Analysis: Havij SQL injection tool*. Retrieved from Checkpoint: https://blog.checkpoint.com/security/analysis-havij-sql-injection-tool/

Cloudflare. (2023). *What is penetration testing?* Retrieved from Cloudflare: https://www.cloudflare.com/learning/security/threats/penetration-testing/

CyberTalents. (2022). *What is a security researcher? How can I become one?* Retrieved from CyberTalents: https://cybertalents.com/blog/what-is-a-security-researcher-how-can-i-become-one

Dark Reading. (2012). *Cybercrime's love affair with Havij spells SQL injection trouble*. Retrieved from Dark Reading: https://www.darkreading.com/application-security/cybercrime-s-love-affair-with-havij-spells-sql-injection-trouble

GeeksforGeeks. (2023). *Database assessment tools for Kali Linux*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/linux-unix/database-assessment-tools-for-kali-linux/

Hacker Target. (2021). *SQLMap tutorial*. Retrieved from Hacker Target: https://hackertarget.com/sqlmap-tutorial/

Kali Linux. (2024). *SQLMap*. Retrieved from Kali Linux: https://www.kali.org/tools/sqlmap/

LinuxSecurity. (2022). *jSQL Injection*. Retrieved from LinuxSecurity: https://linuxsecurity.expert/tools/jsql-injection/

OWASP Foundation. (2021). *OWASP top 10: 2021 top 10 web application security risks*. Retrieved from OWASP Foundation: https://owasp.org/Top10/

Red Hat. (2023). *What is DevSecOps?* Retrieved from Red Hat: https://www.redhat.com/en/topics/devops/what-is-devsecops

SecOps Group. (2023). *A pentester's guide to NoSQL injection*. Retrieved from SecOps Group: https://secops.group/a-pentesters-guide-to-nosql-injection/

Springboard. (2023). *SOC analyst: Job description, skills required & career path*. Retrieved from Springboard: https://www.springboard.com/blog/cybersecurity/soc-analyst-guide/

SQLMap Developers. (2025). *sqlmapproject/sqlmap [GitHub repository]*. Retrieved from GitHub: https://github.com/sqlmapproject/sqlmap

Vaadata. (2022). *SQLMap: The tool for detecting and exploiting SQL injections*. Retrieved from Vaadata: https://www.vaadata.com/blog/sqlmap-the-tool-for-detecting-and-exploiting-sql-injections/

WebAsha. (2021). *Havij overview, features, and why ethical hackers should use it*. Retrieved from WebAsha: https://www.webasha.com/blog/havij-overview-features-and-why-ethical-hackers-should-use-it

Akamai. (2023). *Harnessing artificial intelligence for superior web application firewall*. Retrieved from Akamai Technologies: https://www.akamai.com/blog/security/harnessing-artificial-intelligence-for-superior-web-application-firewall

Cloudflare. (2023). *What Is a Web Application Firewall (WAF)?* Retrieved from Cloudflare: https://www.cloudflare.com/learning/ddos/glossary/web-application-firewall-waf/

Imperva. (2024). *Navigating the new era of AI traffic: How to identify and block AI scrapers*. Retrieved from Imperva Inc.: https://www.imperva.com/blog/navigating-the-new-era-of-ai-traffic-how-to-identify-and-block-ai-scrapers/

Prophaze. (2024). *What is an AI powered WAF?* Retrieved from Prophaze Technologies: https://prophaze.com/learn/waf/what-is-an-ai-powered-waf/

Prophaze. (n.d.). *What is an AI-powered WAF?* Prophaze. https://prophaze.com/learn/waf/what-is-an-ai-powered-waf/

GitGuardian. (2023, February 22). *Dynamic application security testing: Benefits, pitfalls, and top open-source solutions*. GitGuardian Blog. https://blog.gitguardian.com/dynamic-application-security-testing-benefits-pitfalls-and-top-open-source-solutions/