Ticket #1

Scenario: There are multiple reports of employees located in the USERNet subnet who cannot get to www.domain.edu, and they are being redirected to a suspicious site. A help desk technician states that the server team recently installed updates to DMZ_Server_3 which acts as the DNS Server for the organization.

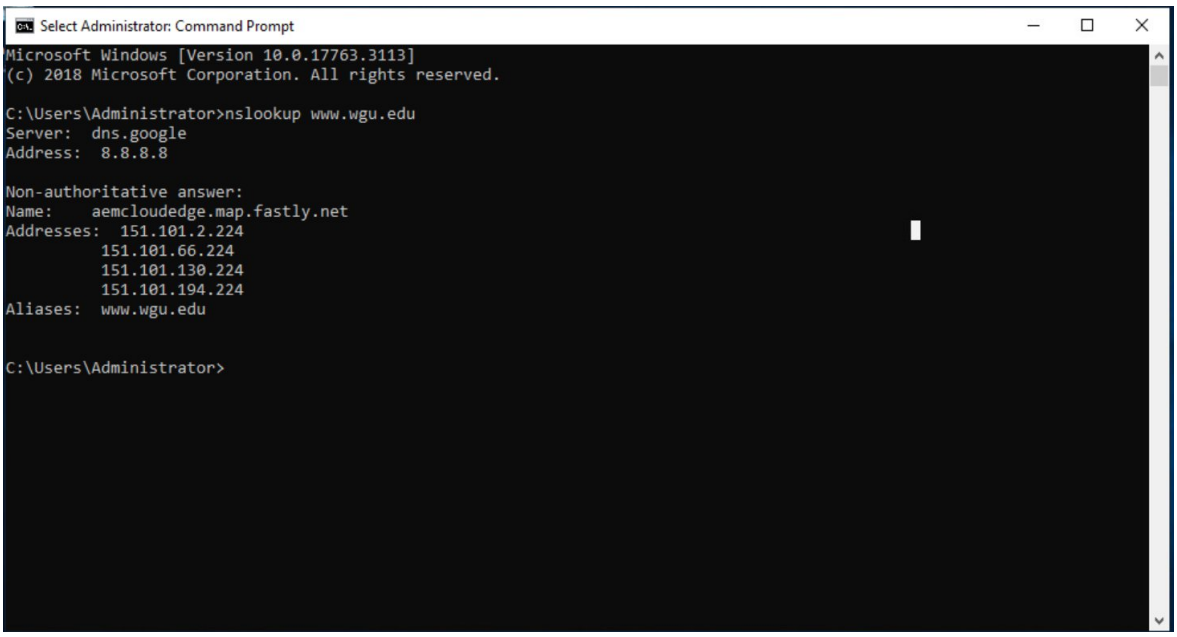
After discovering that the workstations and DNS server were displaying the wrong webpage, a detailed investigation was carried out to identify the source of the issue.

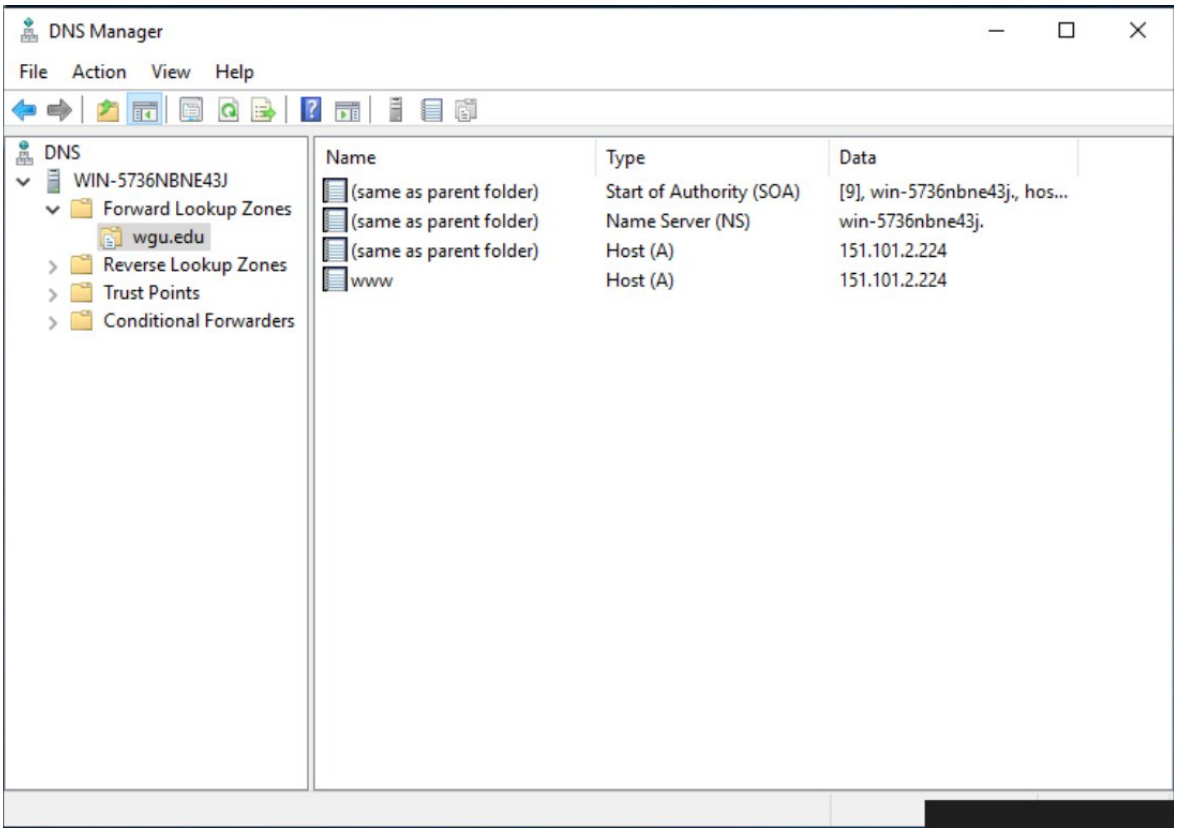
Firstly, it was noted that the DNS server was responsible for listing the wrong IP address of the website, which was the same IP address as another machine on the same subnet. This was identified as a conflict that could lead to several problems, and hence, the DNS settings were examined to identify the cause of the issue.

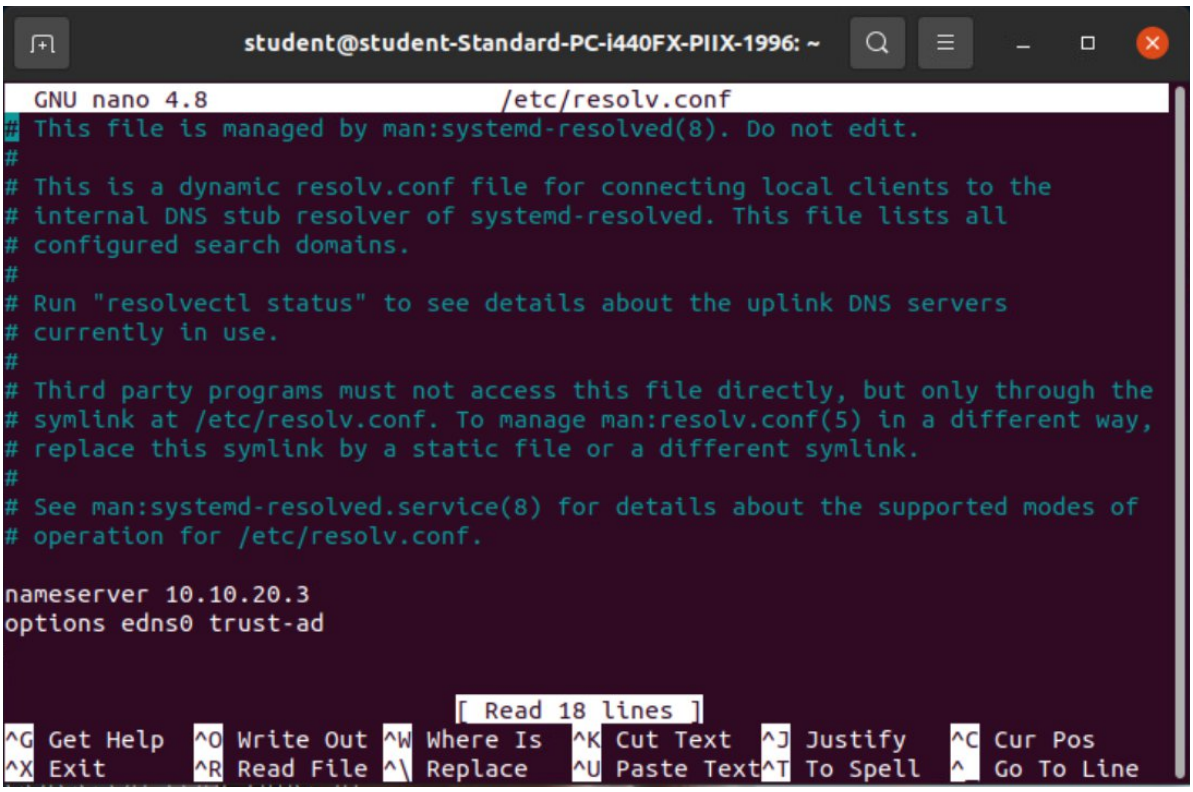
To rectify this issue, the official, public IP address of the website was found by using nslookup. Several IP addresses were identified, and it was decided to work with the primary public IP of the website, which was 151.101.2.224. This IP address was then updated in the DNS server from the incorrect IP of 10.10.20.2. This ensured that the Windows workstations were able to access the correct website.

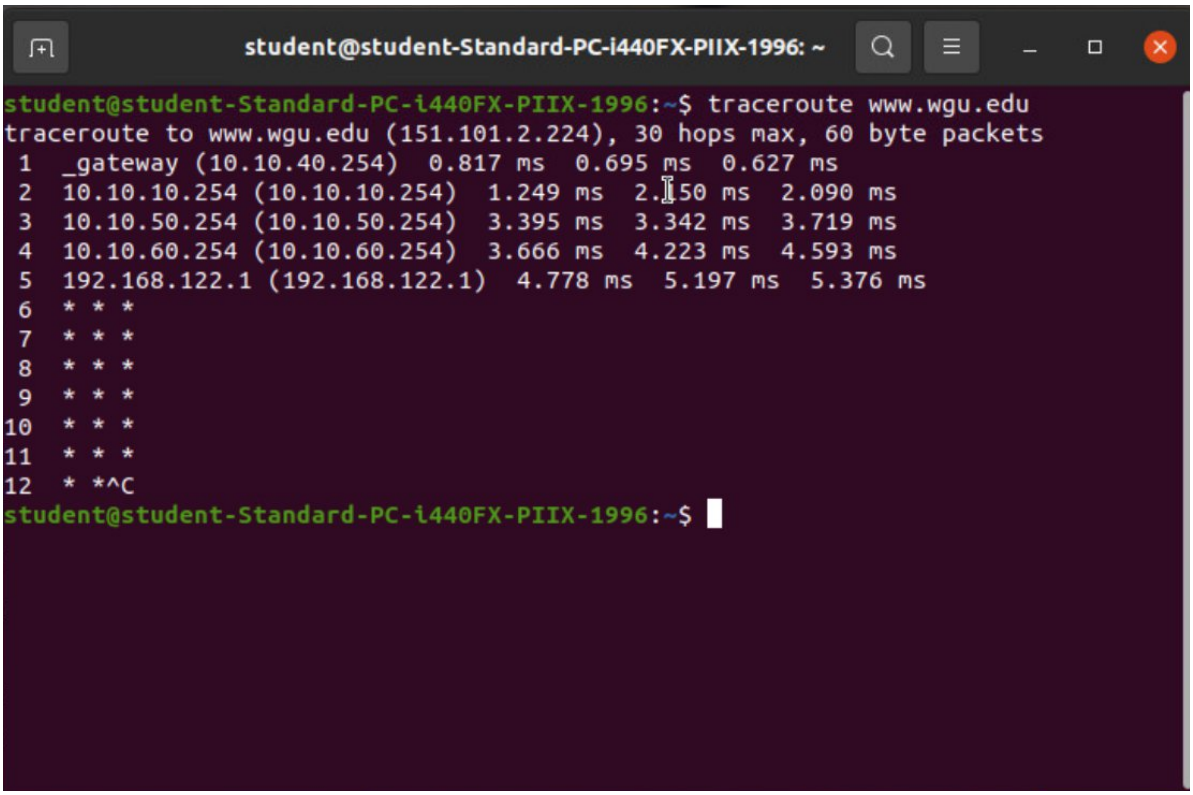
However, the issue persisted with the two Linux workstations. To resolve this, the nameserver was updated in the /etc/resolv.conf file to the IP address of DMZ server 3 (10.10.20.3). This allowed the Linux workstations to access the website successfully.

It is essential to ensure that the IP addresses used in DNS servers are correct to avoid conflicts that could cause issues with accessing websites or other online resources.


Select Administrator: Command Prompt
Microsoft Windows [Version 10.0.17763.3113]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\Administrator>nslookup www.wgu.edu
Server: dns.google
Address: 8.8.8.8
Non-authoritative answer:
Name: aemcloudedge.map.fastly.net
Addresses: 151.101.2.224
151.101.66.224
151.101.130.224
151.101.194.224
Aliases: www.wgu.edu
C:\Users\Administrator>


DNS Manager
File Action View Help
DNS
WIN-5736NBNE43J
Forward Lookup Zones
wgu.edu
Reverse Lookup Zones
Trust Points
Conditional Forwarders
Name
Type
Data
(same as parent folder)
Start of Authority (SOA)
[9], win-5736nbne43j., hos...
(same as parent folder)
Name Server (NS)
win-5736nbne43j.
(same as parent folder)
Host (A)
151.101.2.224
www
Host (A)
151.101.2.224


student@student-Standard-PC-i440FX-PIIX-1996: ~
GNU nano 4.8 /etc/resolv.conf
This file is managed by man:systemd-resolved(8). Do not edit.
#
This is a dynamic resolv.conf file for connecting local clients to the
internal DNS stub resolver of systemd-resolved. This file lists all
configured search domains.
#
Run "resolvectl status" to see details about the uplink DNS servers
currently in use.
#
Third party programs must not access this file directly, but only through the
symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a different way,
replace this symlink by a static file or a different symlink.
#
See man:systemd-resolved.service(8) for details about the supported modes of
operation for /etc/resolv.conf.
nameserver 10.10.20.3
options edns0 trust-ad
[Read 18 lines]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line


student@student-Standard-PC-i440FX-PIIX-1996: ~
student@student-Standard-PC-i440FX-PIIX-1996:~$ traceroute www.wgu.edu
traceroute to www.wgu.edu (151.101.2.224), 30 hops max, 60 byte packets
1 _gateway (10.10.40.254) 0.817 ms 0.695 ms 0.627 ms
2 10.10.10.254 (10.10.10.254) 1.249 ms 2.150 ms 2.090 ms
3 10.10.50.254 (10.10.50.254) 3.395 ms 3.342 ms 3.719 ms
4 10.10.60.254 (10.10.60.254) 3.666 ms 4.223 ms 4.593 ms
5 192.168.122.1 (192.168.122.1) 4.778 ms 5.197 ms 5.376 ms
6 * * *
7 * * *
8 * * *
9 * * *
10 * * *
11 * * *
12 * *^C
student@student-Standard-PC-i440FX-PIIX-1996:~$

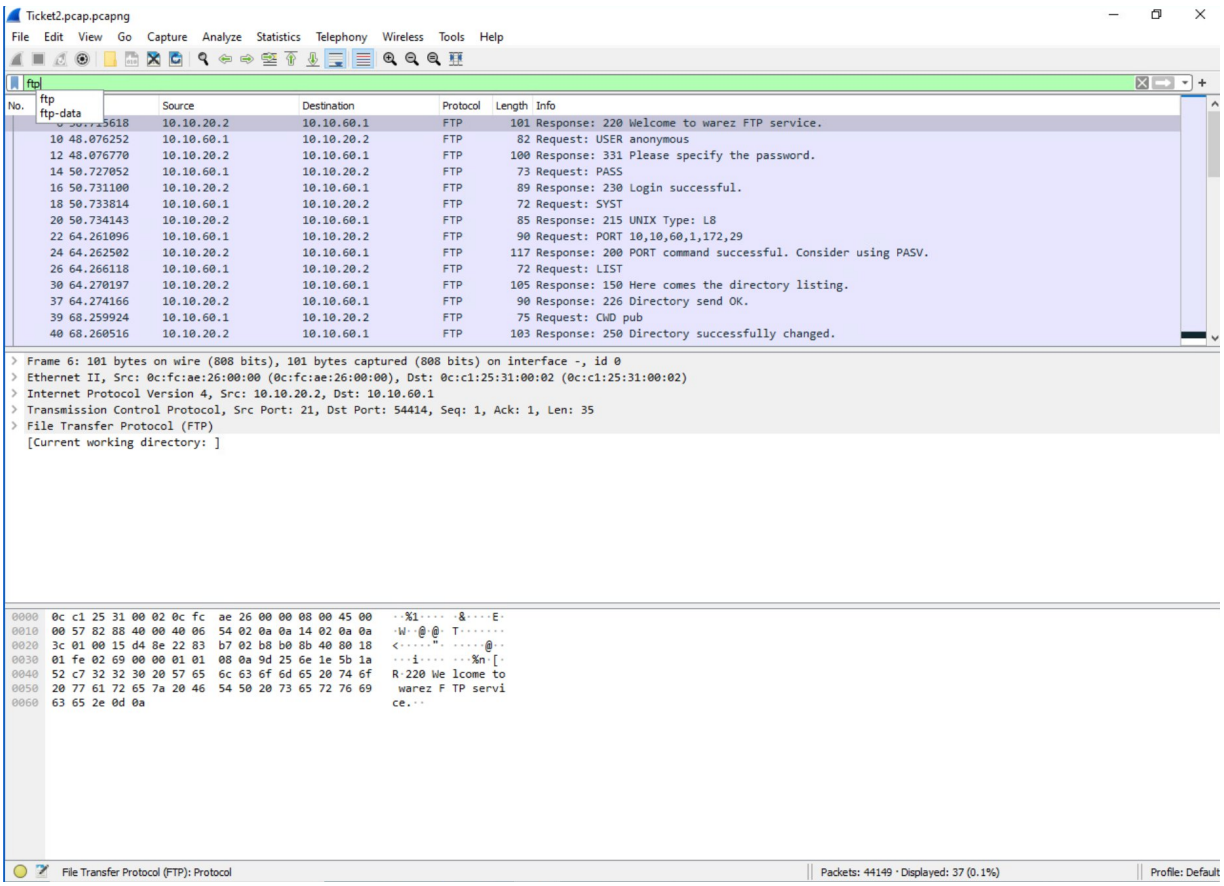
Ticket #2

Scenario: A complaint came in that a certain organization is hosting an illegal FTP site to download copyrighted software. The security team has provided a pcap file capturing all FTP traffic on the network. They've asked you to identify where the FTP site is being hosted.

Firstly, the Pcap file was opened on the machine, and the default application used to open the file was Wireshark. The Wireshark interface allowed me to filter the results by FTP, as shown in the screenshot.

Upon filtering the results, the first IP that had FTP ports open was identified as the WAREZ FTP server. This was determined from the name displayed in the interface, which was "WAREZ FTP". Further investigation in the interface revealed that the IP address of the rogue FTP server was listed as 10.10.20.2.

This detailed approach of using Wireshark to filter results by FTP ports allowed for the identification of the rogue FTP server on the network.

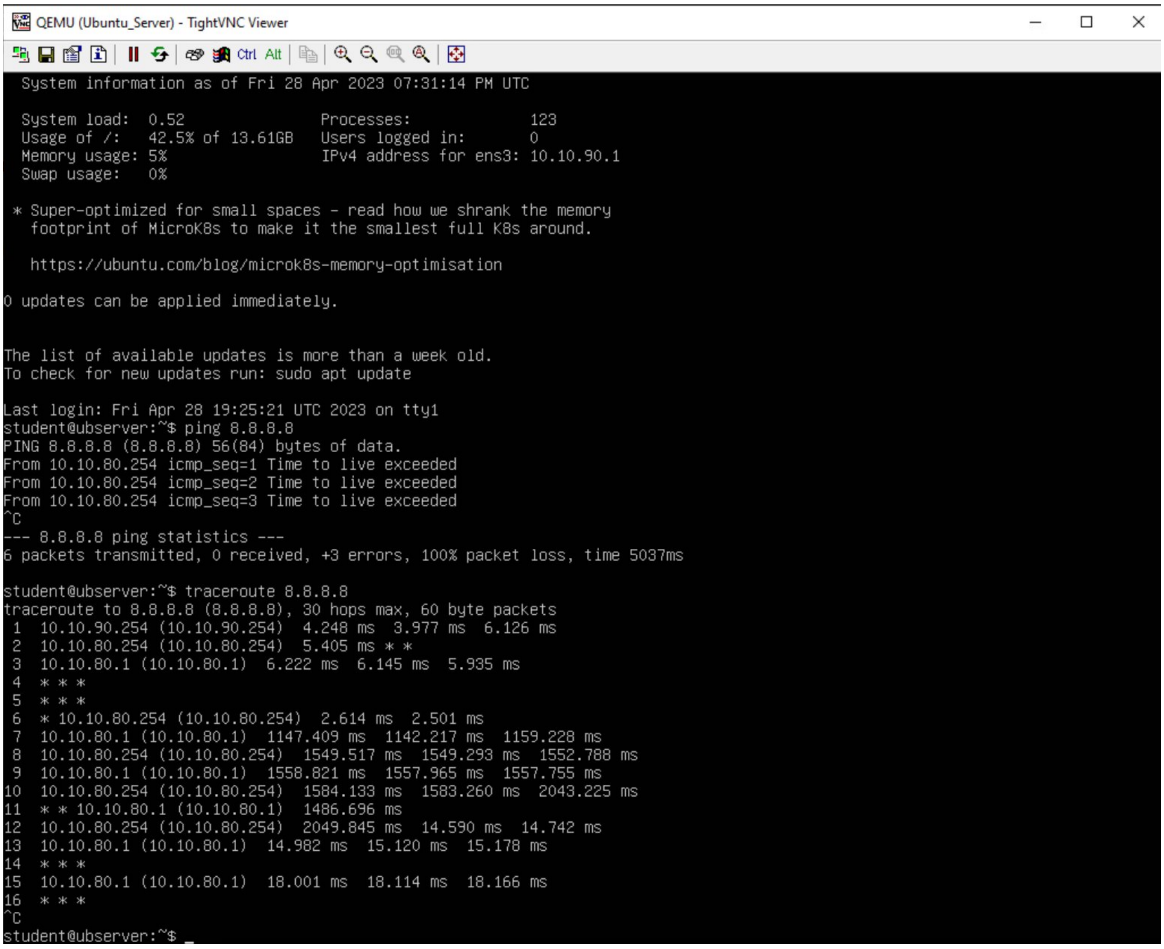


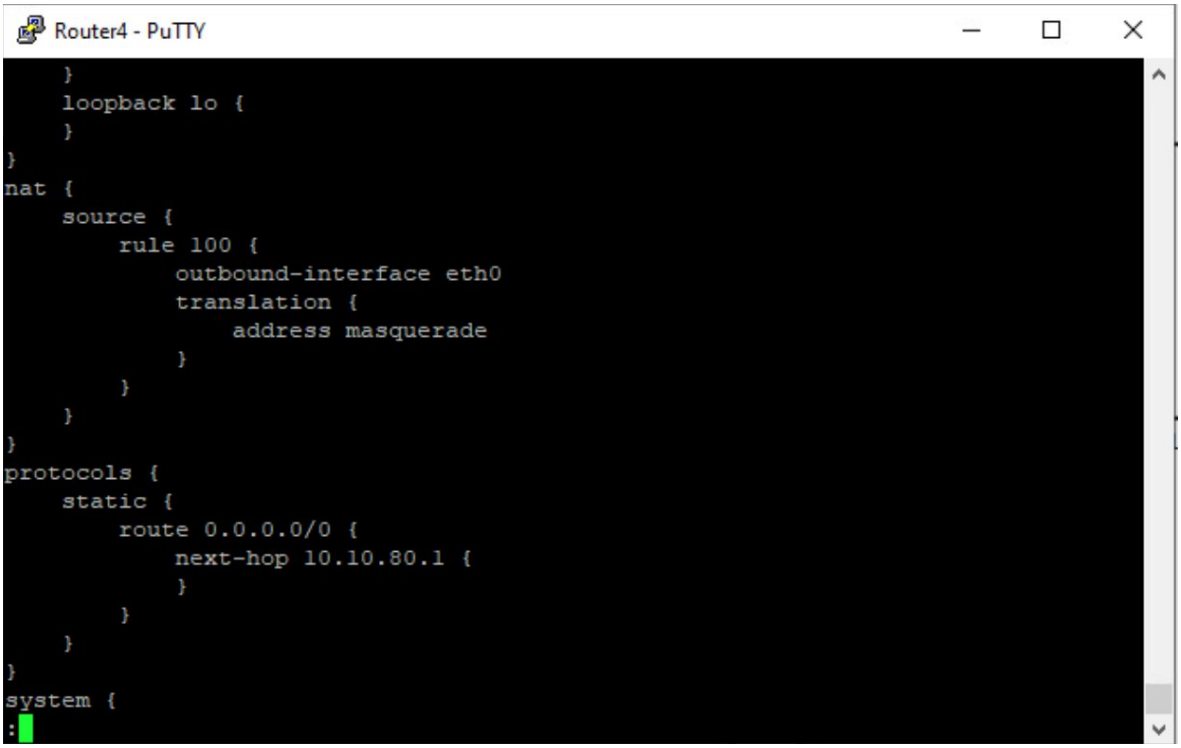
Ticket #3

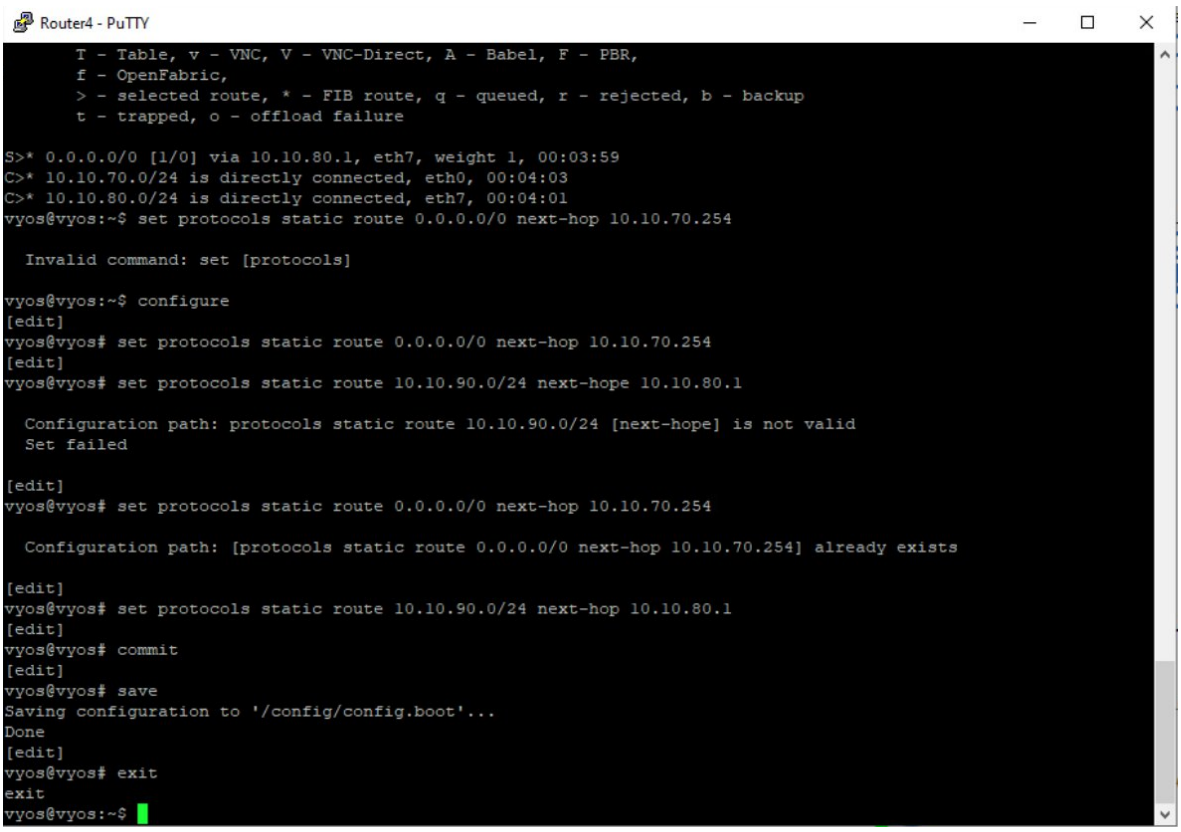
Scenario: The host "Ubuntu_Server" can't get to any of the assigned networks or the internet, which is preventing the server from pulling the required security patches.

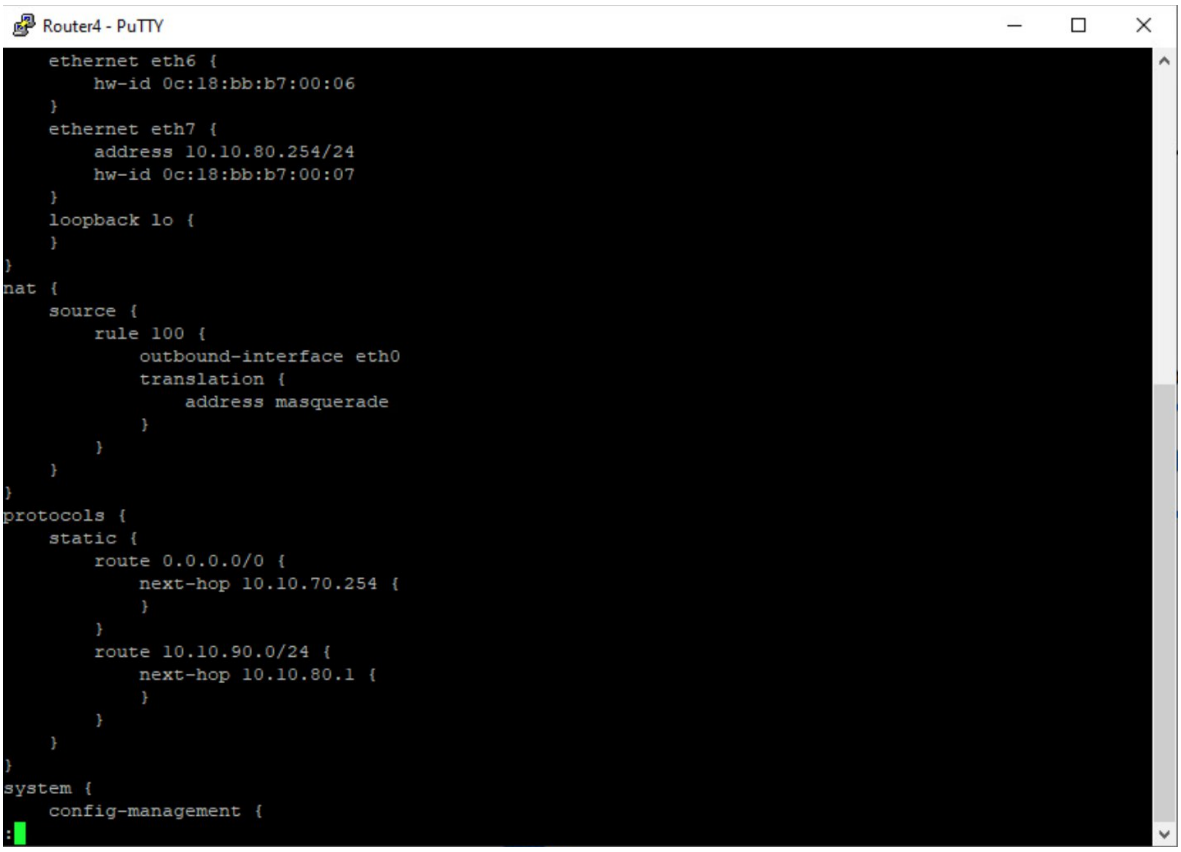
I tested the Ubuntu Server's internet connection by trying to run sudo apt update and running a ping command to 8.8.8.8. The goal was to test for any connectivity to the internet or another device on the network. Ping to the internet, so I decided to use a traceroute command to see where in the network there was a connectivity issue. Traceroute showed that routers were sending data to each other in a loop, from 10.10.80.1 and 10.10.80.254. This means that data was connecting to router 4, and then it was being sent back to its next destination, which was the previous step in the loop. I went into router 4 to see which settings were connected with which interfaces. I entered entered *show ip route* to show the route table of the device to make sure all devices were properly listed. I decided to enter the configuration of Router 4, using 'configure' to see if the next-hop was properly configured. It was not, and that was the reason for the traceroute loop. There needed to be a next hop to the internet via 10.10.70.254 and also a next hop to the Ubuntu server via 10.10.80.1

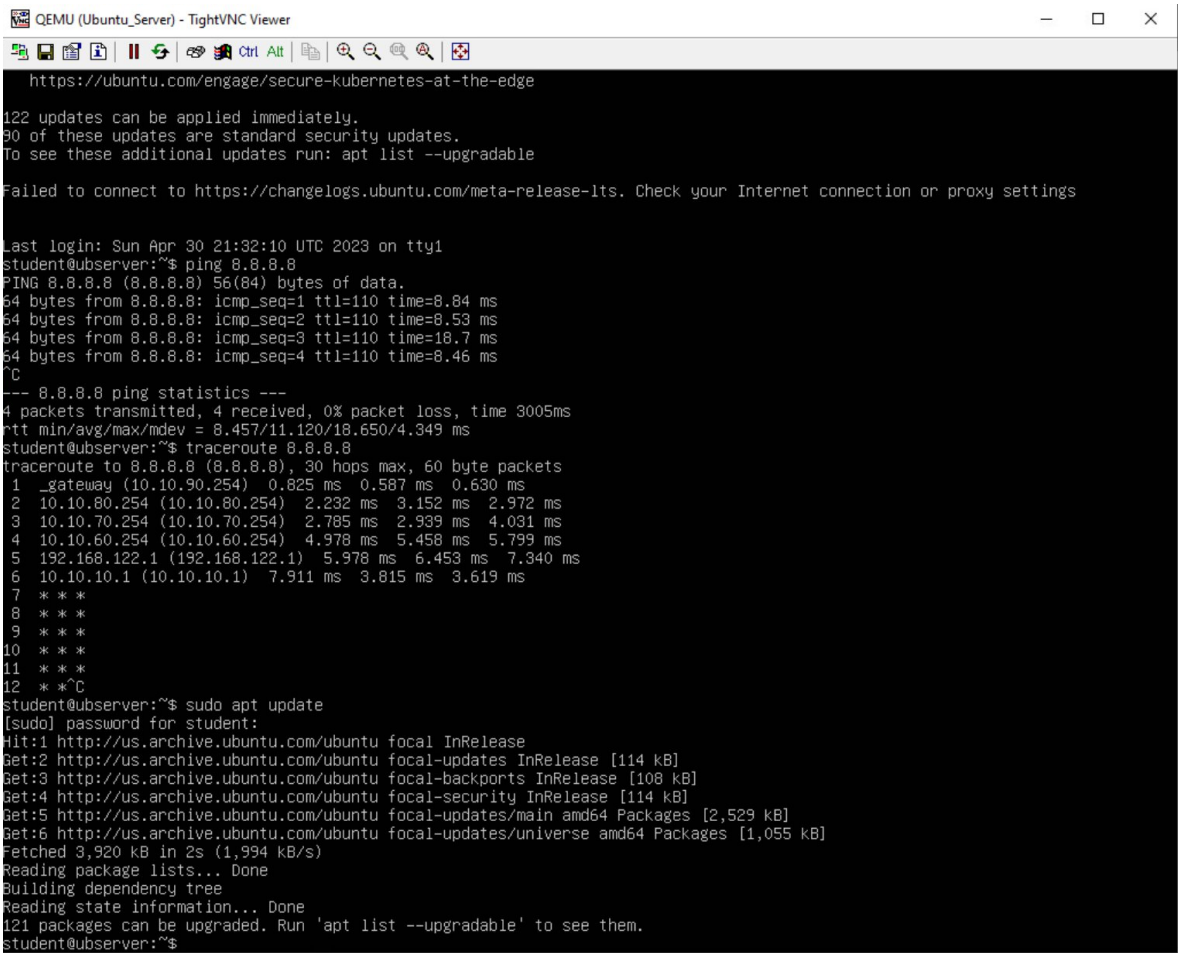
There was only one static route, and it was pointed to 10.10.80.1 as the next hop. I changed the internet facing route with: set protocols static route 0.0.0.0/24 next-hop 10.10.70.254. I then added a static route facing the Ubuntu server with: set protocols static route 10.10.90.0/24 next-hop 10.10.80.1 so that data could be sent to the server. The steps were to use: *configure, <command>, commit, save, exit*. I then tested connectivity to the internet from the Ubuntu server, using ping and traceroute commands to 8.8.8.8. The commands worked successfully, and I was able to run sudo apt update to start updating the Ubuntu Server with internet connectivity.


QEMU (Ubuntu_Server) - TightVNC Viewer
Ctrl Alt
System information as of Fri 28 Apr 2023 07:31:14 PM UTC
System load: 0.52 Processes: 123
Usage of /: 42.5% of 13.61GB Users logged in: 0
Memory usage: 5% IPv4 address for ens3: 10.10.90.1
Swap usage: 0%
* Super-optimized for small spaces - read how we shrank the memory
footprint of MicroK8s to make it the smallest full K8s around.
https://ubuntu.com/blog/microk8s-memory-optimisation
0 updates can be applied immediately.
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Fri Apr 28 19:25:21 UTC 2023 on tty1
student@ubserver:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
From 10.10.80.254 icmp_seq=1 Time to live exceeded
From 10.10.80.254 icmp_seq=2 Time to live exceeded
From 10.10.80.254 icmp_seq=3 Time to live exceeded
^C
--- 8.8.8.8 ping statistics ---
6 packets transmitted, 0 received, +3 errors, 100% packet loss, time 5037ms
student@ubserver:~$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
1 10.10.90.254 (10.10.90.254) 4.248 ms 3.977 ms 6.126 ms
2 10.10.80.254 (10.10.80.254) 5.405 ms * *
3 10.10.80.1 (10.10.80.1) 6.222 ms 6.145 ms 5.935 ms
4 * * *
5 * * *
6 * 10.10.80.254 (10.10.80.254) 2.614 ms 2.501 ms
7 10.10.80.1 (10.10.80.1) 1147.409 ms 1142.217 ms 1159.228 ms
8 10.10.80.254 (10.10.80.254) 1549.517 ms 1549.293 ms 1552.788 ms
9 10.10.80.1 (10.10.80.1) 1558.821 ms 1557.965 ms 1557.755 ms
10 10.10.80.254 (10.10.80.254) 1584.133 ms 1583.260 ms 2043.225 ms
11 * * 10.10.80.1 (10.10.80.1) 1486.696 ms
12 10.10.80.254 (10.10.80.254) 2049.845 ms 14.590 ms 14.742 ms
13 10.10.80.1 (10.10.80.1) 14.982 ms 15.120 ms 15.178 ms
14 * * *
15 10.10.80.1 (10.10.80.1) 18.001 ms 18.114 ms 18.166 ms
16 * * *
^C
student@ubserver:~$


Router4 - PuTTY
}
loopback lo {
}
}
nat {
source {
rule 100 {
outbound-interface eth0
translation {
address masquerade
}
}
}
}
protocols {
static {
route 0.0.0.0/0 {
next-hop 10.10.80.1 {
}
}
}
}
system {
:


Router4 - PuTTY
T - Table, v - VNC, V - VNC-Direct, A - Babel, F - PBR,
f - OpenFabric,
> - selected route, * - FIB route, q - queued, r - rejected, b - backup
t - trapped, o - offload failure
S>* 0.0.0.0/0 [1/0] via 10.10.80.1, eth7, weight 1, 00:03:59
C>* 10.10.70.0/24 is directly connected, eth0, 00:04:03
C>* 10.10.80.0/24 is directly connected, eth7, 00:04:01
vyos@vyos:~$ set protocols static route 0.0.0.0/0 next-hop 10.10.70.254
Invalid command: set [protocols]
vyos@vyos:~$ configure
[edit]
vyos@vyos# set protocols static route 0.0.0.0/0 next-hop 10.10.70.254
[edit]
vyos@vyos# set protocols static route 10.10.90.0/24 next-hope 10.10.80.1
Configuration path: protocols static route 10.10.90.0/24 [next-hope] is not valid
Set failed
[edit]
vyos@vyos# set protocols static route 0.0.0.0/0 next-hop 10.10.70.254
Configuration path: [protocols static route 0.0.0.0/0 next-hop 10.10.70.254] already exists
[edit]
vyos@vyos# set protocols static route 10.10.90.0/24 next-hop 10.10.80.1
[edit]
vyos@vyos# commit
[edit]
vyos@vyos# save
Saving configuration to '/config/config.boot'...
Done
[edit]
vyos@vyos# exit
exit
vyos@vyos:~$


Router4 - PuTTY
ethernet eth6 {
hw-id 0c:18:bb:b7:00:06
}
ethernet eth7 {
address 10.10.80.254/24
hw-id 0c:18:bb:b7:00:07
}
loopback lo {
}
}
nat {
source {
rule 100 {
outbound-interface eth0
translation {
address masquerade
}
}
}
}
protocols {
static {
route 0.0.0.0/0 {
next-hop 10.10.70.254 {
}
}
route 10.10.90.0/24 {
next-hop 10.10.80.1 {
}
}
}
}
system {
config-management {



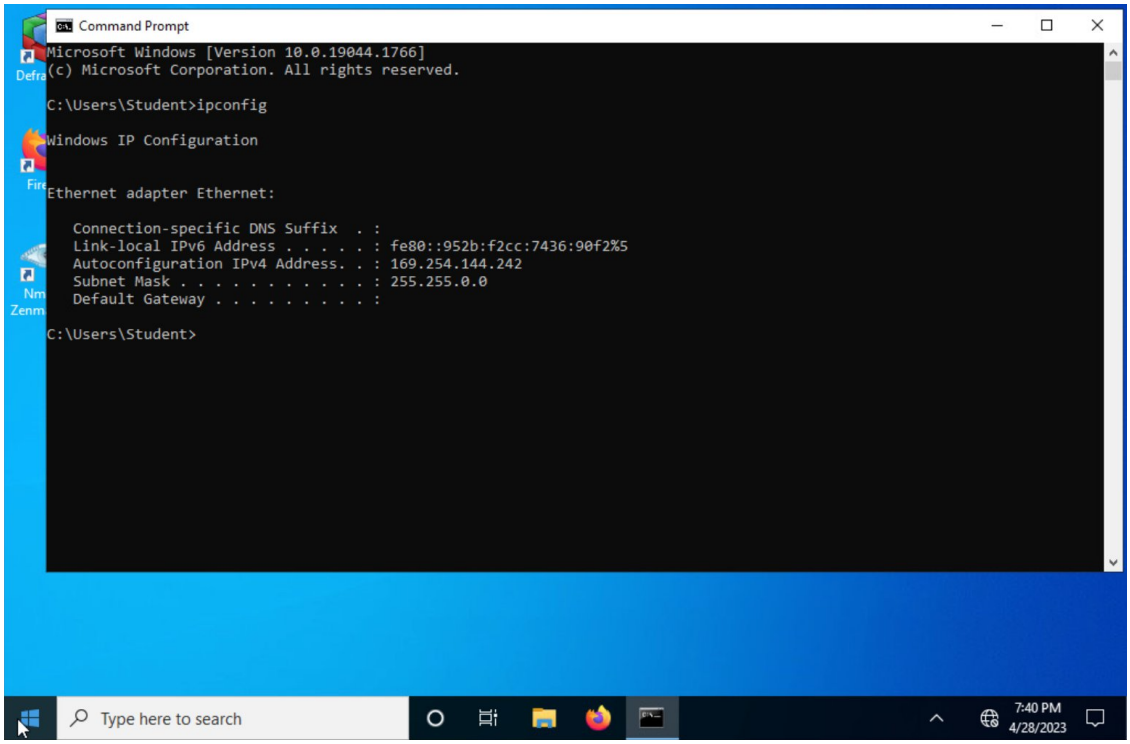
Ticket #4

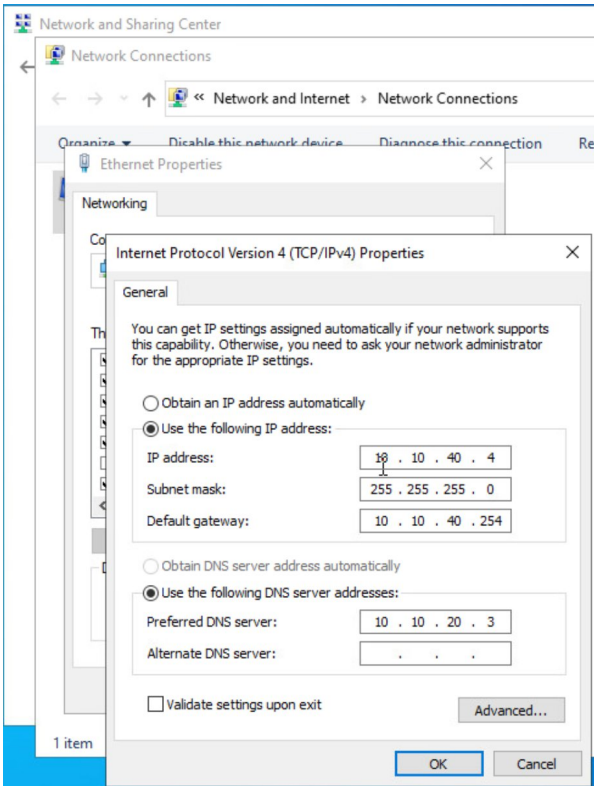
Scenario: A user complains that he cannot access the internet or network resources on his company laptop (Windows_Laptop_1) when it is connected via an ethernet cable to the office network.

This workstation had no connection to the internet, so I decided to check IP configuration with the command *ipconfig*. It was found that the machine was assigned an Automatic Private IP Address (APIPA) which indicated that there was no DHCP server available to assign an IP address. To resolve this, I manually assigned a static IP address of 10.10.40.4, in accordance with the network diagram on GNS3.

In addition, I updated the default gateway to the DNS of DMZ Server 3, which provided the necessary routing to connect to the internet. This was done to ensure that the workstation was able to communicate with other devices on the network and access the internet.

To confirm that the issue was resolved, I performed a ping test and successfully loaded websites on the workstation. This ensured that the workstation was connected to the network and had internet connectivity.





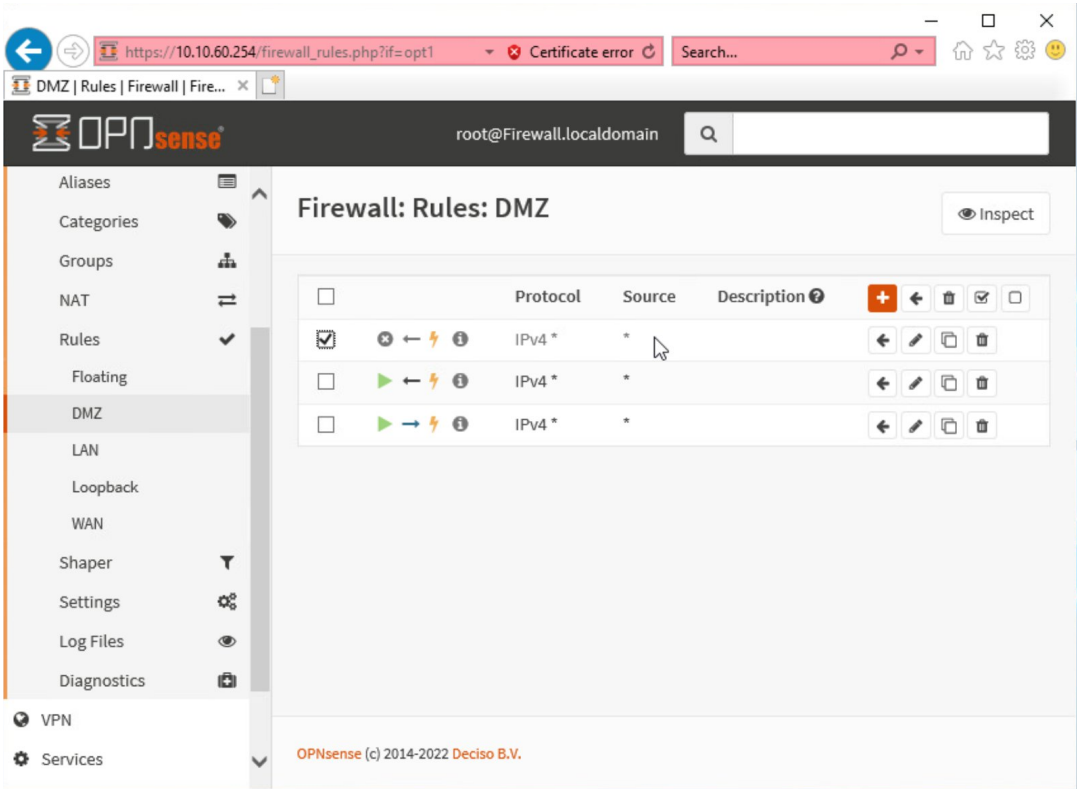
Ticket #5

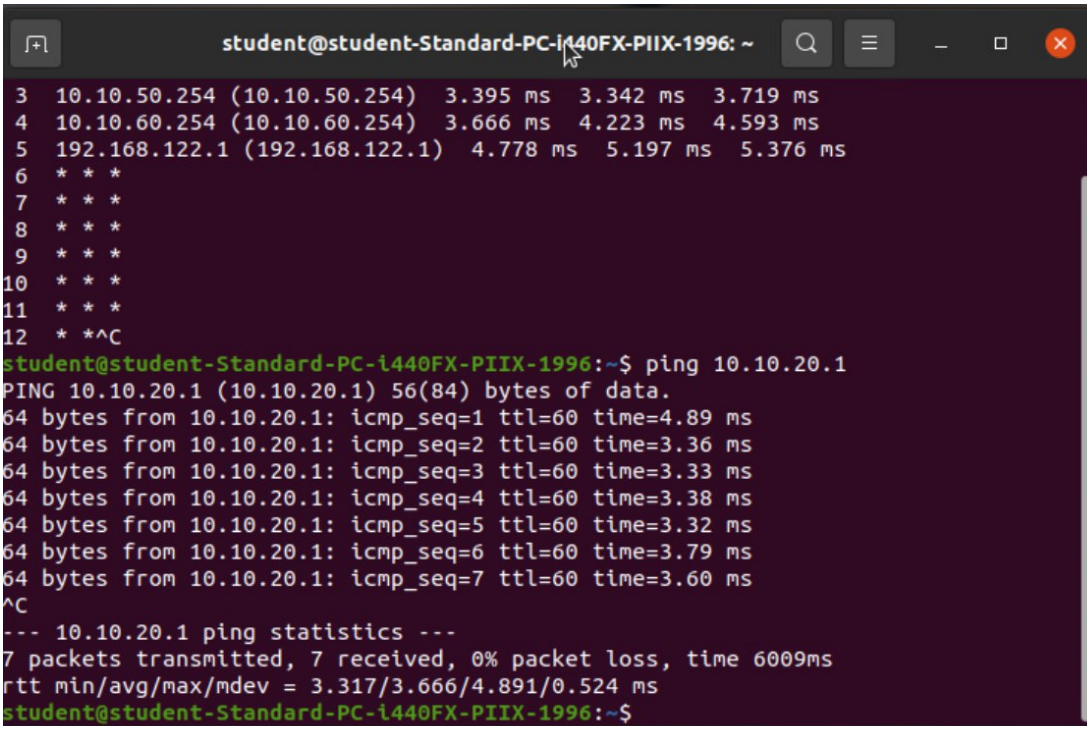
Scenario: A coworker states that she worked on a ticket to allow access through the firewall to DMZ_Server_1. There is now no access to the server from any device outside its network.

Since there was limited connectivity to DMZ Server 1 (10.10.20.1), I decided to test which servers were able to connect. I used the *ping* command, and was able to obtain a successful response from the device. I tried to ping 10.10.20.1 from the workstations, and was unsuccessful. Since there was a previous ticket regarding connectivity through the firewall, I decided to investigate the firewall.

I opened the OPNSense Firewall to check settings using the URL given (10.10.60.254) on a device on the same subnet, DMZ Server 3. I used the address 10.10.60.254 in the browser, and was able to access the rules list in Firewall > Rules > DMZ. There was a rule that explicitly denied all traffic to DMZ Server 1. Someone must have placed this rule here, and this is the reason why devices could not access the device through the firewall!

To resolve the issue, I removed the rule that was restricting access, as shown in the screenshot. This restored access to 10.10.20.1, and I tested connectivity by using the ping command from an external workstation, Linux 1. The ping was successful, indicating that the issue had been resolved.





Ticket #6

Scenario: Your local cyber security team is requesting to know what ports are open on DMZ_Server2 to identify services that may be running outside of the permitted services. Permitted services are 22/ssh, 135/msrpc, 3389/ms-wbt-server, and 8080/http-proxy.

I needed to check which ports were open on another device on the same subnet, so I decided to use Zenmap. I installed Zenmap on DMZ Server 3 and used it to run an nmap scan of DMZ Server 2 (10.10.20.2) with a quick scan to see which ports were open. It reported the following ports as open, 21, 22, 23, 80, 135, 139, 443, 3389, 8080, and 9999. Permitted services are 22/ssh, 135/msrpc, 3389/ms-wbt-server, and 8080/http-proxy, so we would need to block the other open ports for added security. Zenmap was a great tool where I could easily compare the ports that were open and closed.