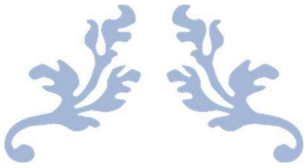


BGP Exploration and Attack

April 2024

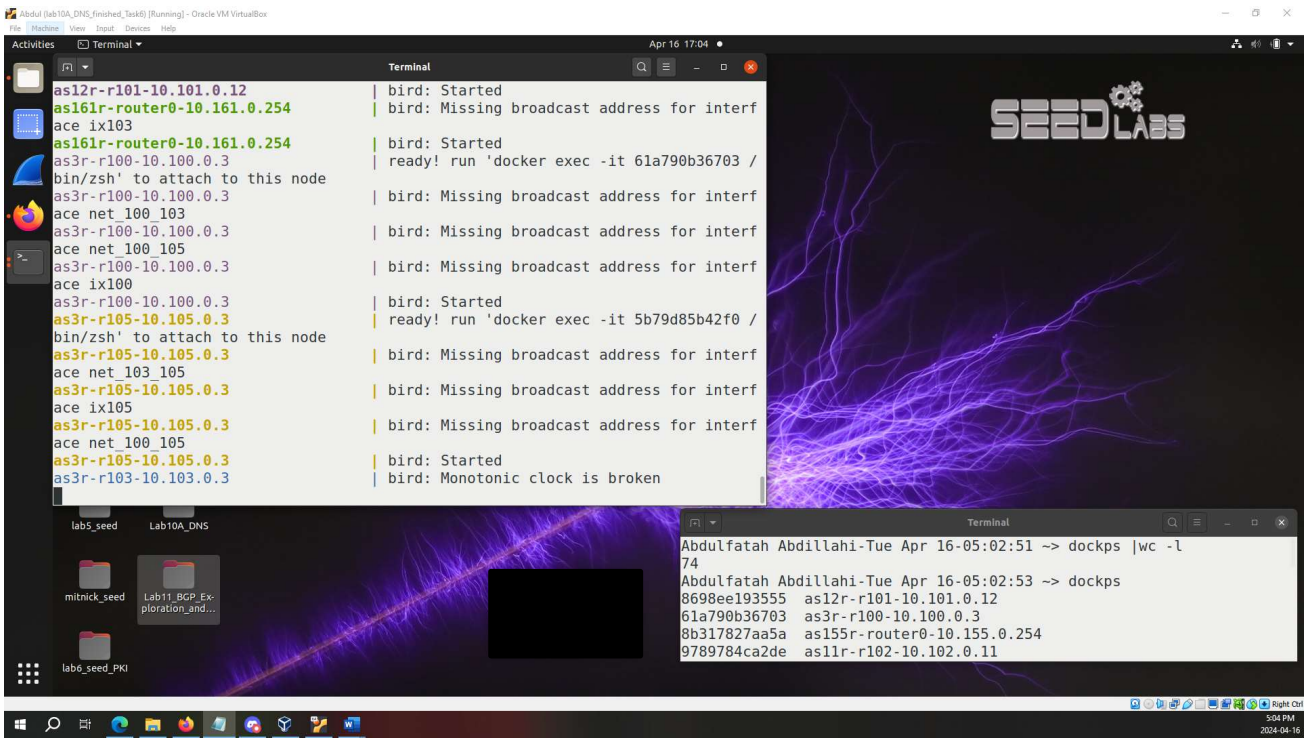
Abdulfatah Abdillahi

Contents

SEED: BGP Lab

Setting up the Environment

Creating docker containers



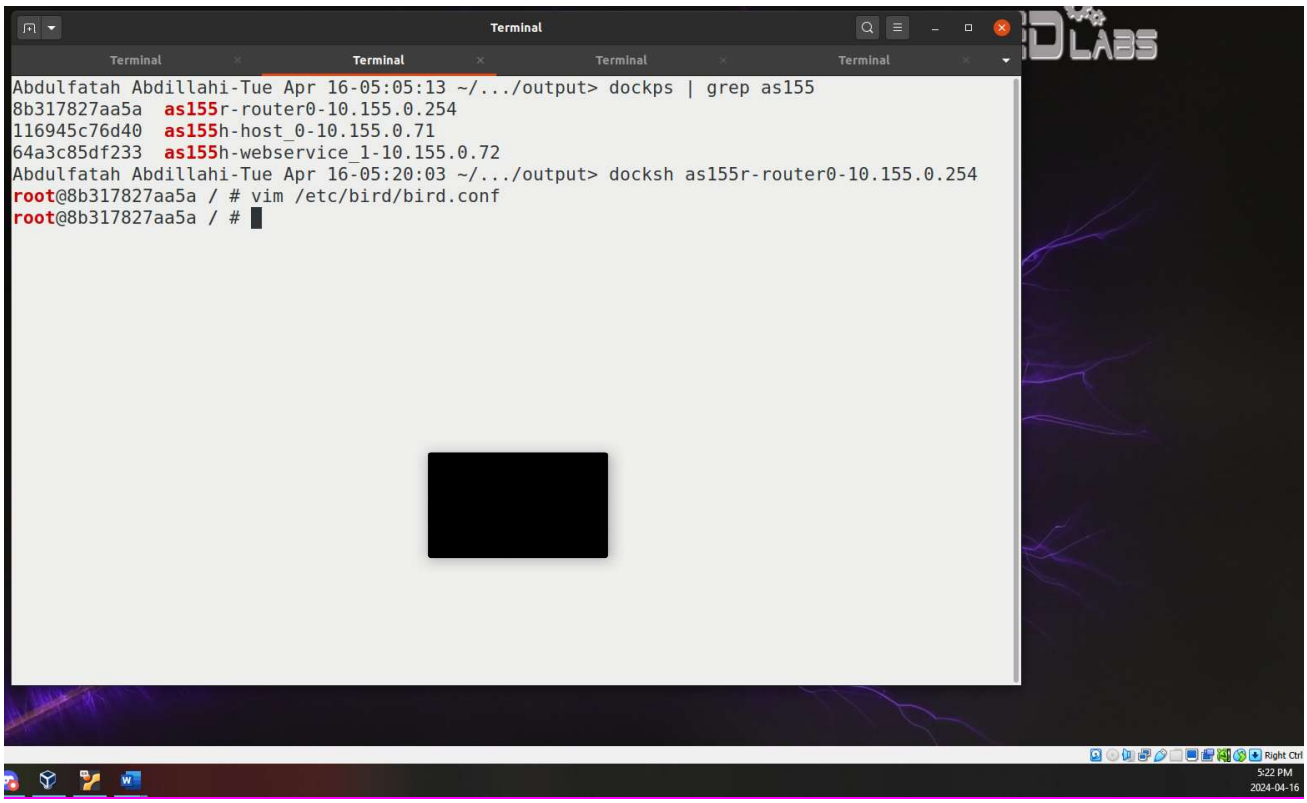
Task 1: Stub Autonomous System (AS)

In this task, we focus on stub ASes, see how it peers with others.

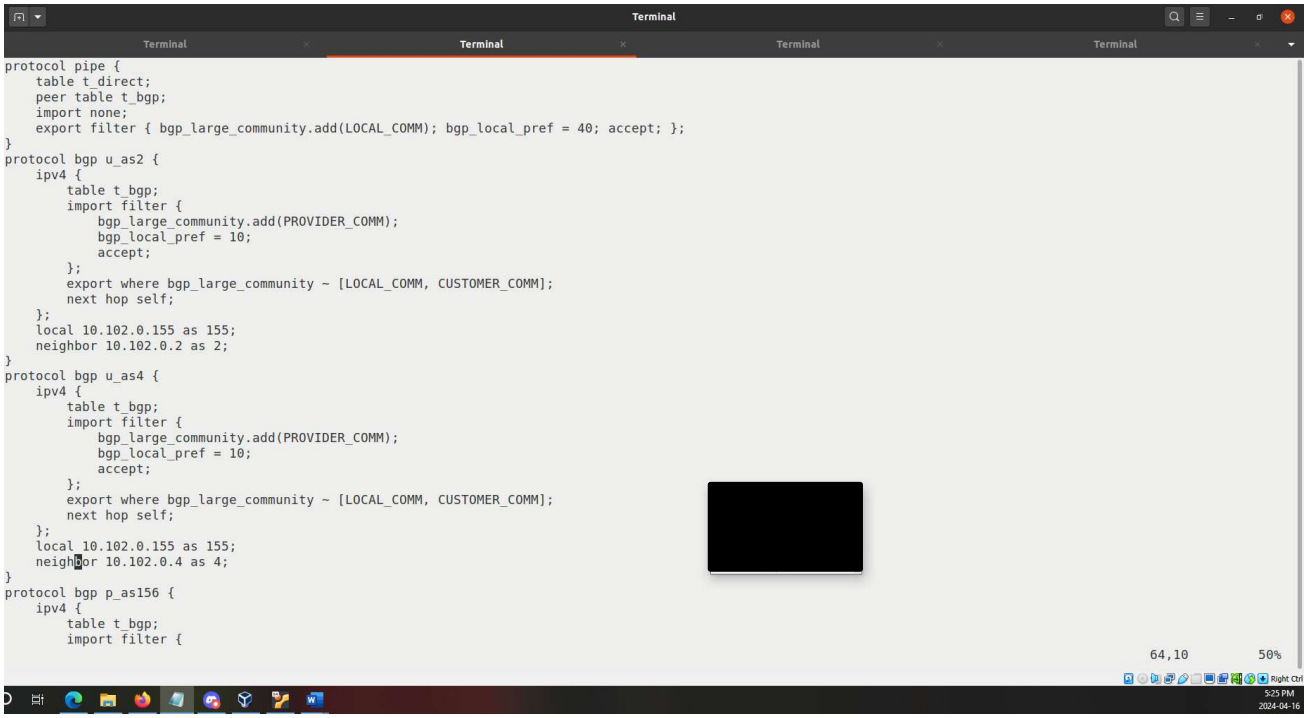
Task 1.a: Understanding AS-155's BGP Configuration

Task 1.a.1

Here, you can see me searching for AS-155 and accessing that container to view the *bird.conf* file.

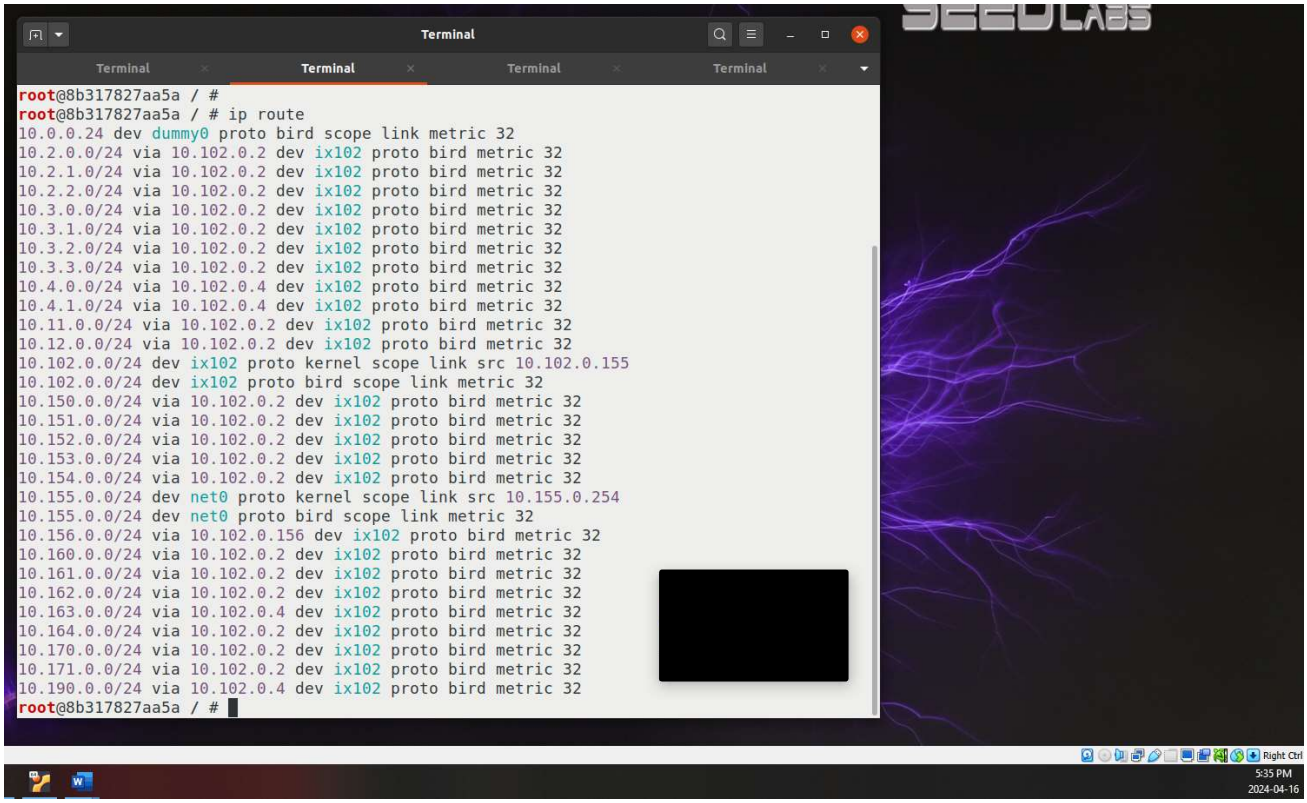


Below you can see what the *bird.conf* file looks like. After examining the file, it was concluded that AS-155 peers with AS-2 a neighbors IP "10.102.0.2", AS-4 a neighbors IP "10.102.0.4", and AS-156 a neighbors IP "10.102.0.156" and each of which have a local IP of "10.102.0.155".

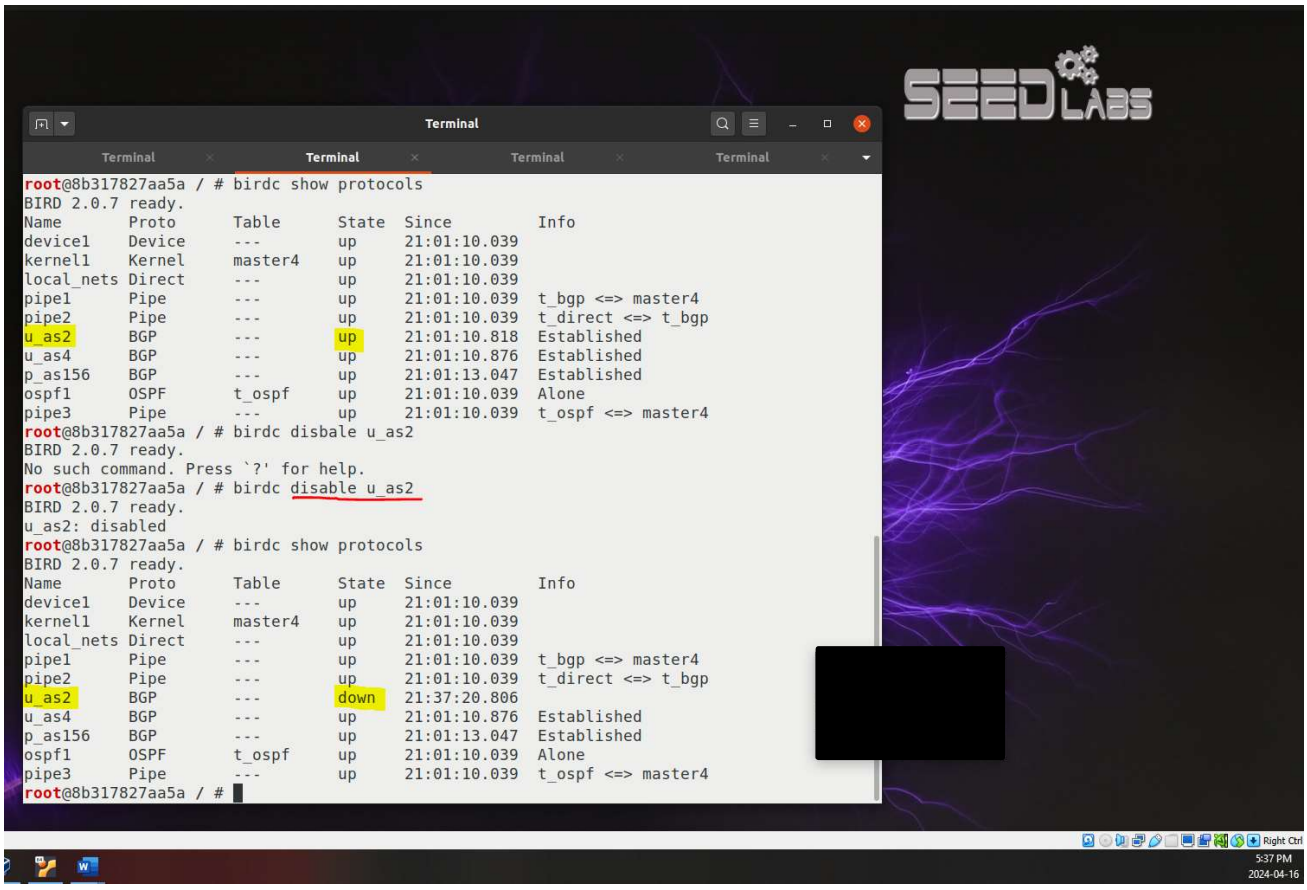


Task 1.a.2

Here, you can see the routes set on AS-155 before attempting to disable peering with AS-2 (10.102.0.2).

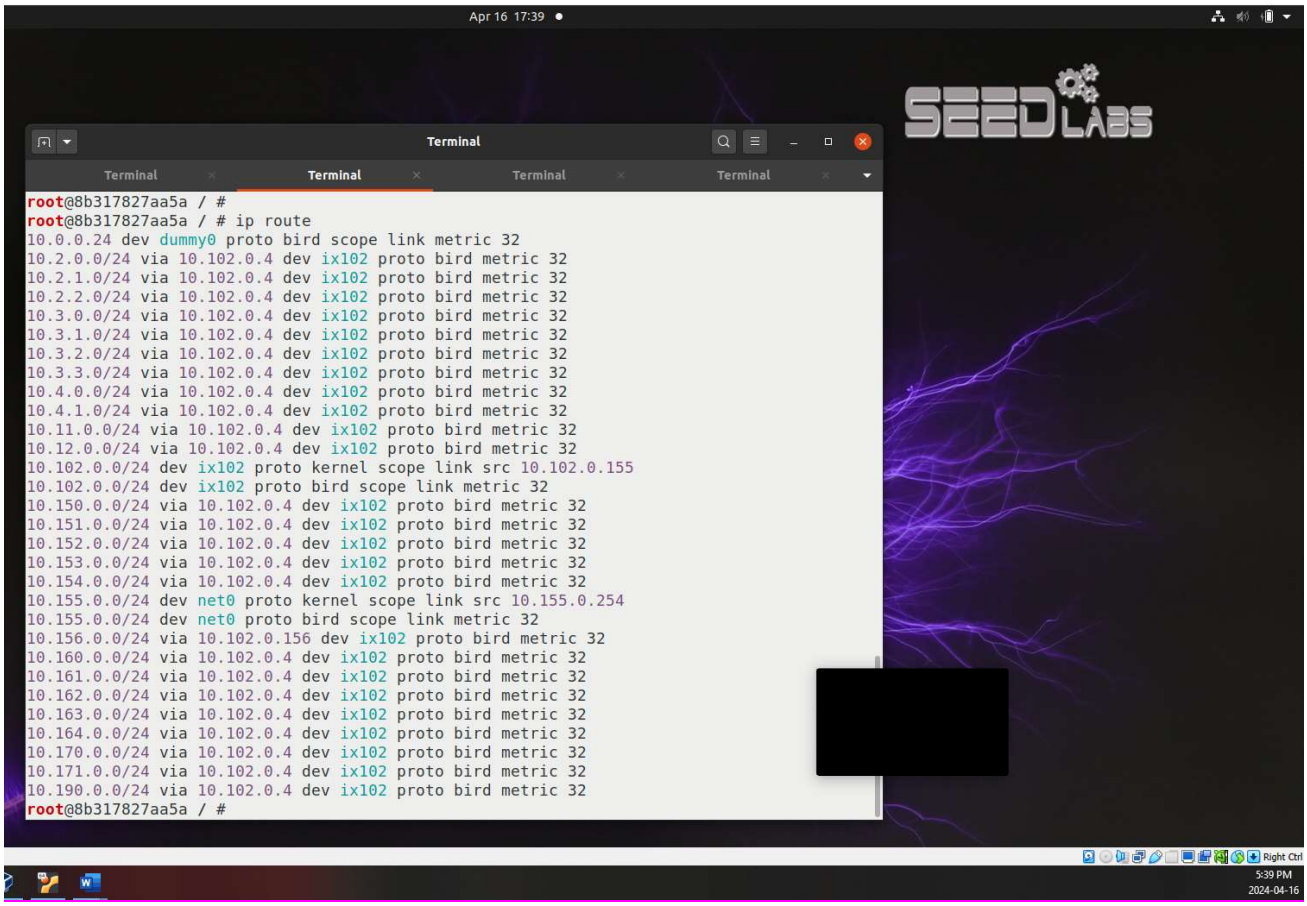


The below screenshot shows a comparison of the status of the BGP session before and after disabling AS-2 as a peer for AS-155 router.

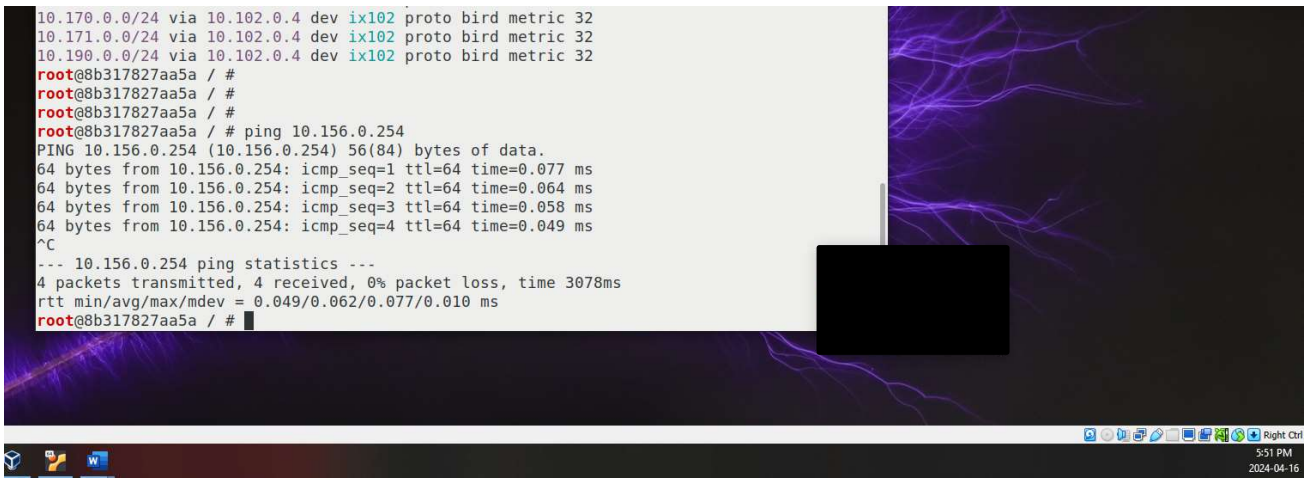


Here, you can see that as a result of disabling AS-2 as a peer, the gateway is now set to AS-4

(10.102.0.4) instead of AS-2.

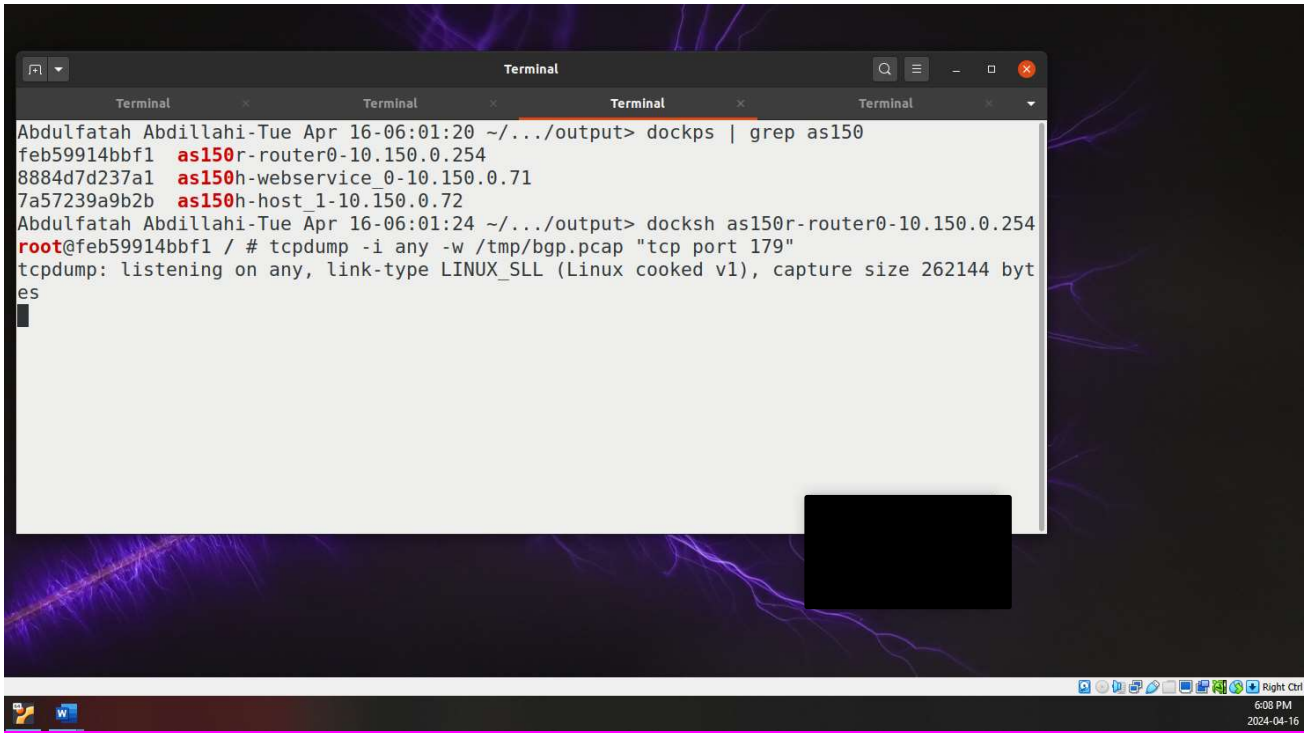


To prove that AS-155 indeed does still have access to the internet, we can ping another router. This still works because of the dynamic nature of BGP as AS-155 currently relies on AS-4 instead of AS-2.



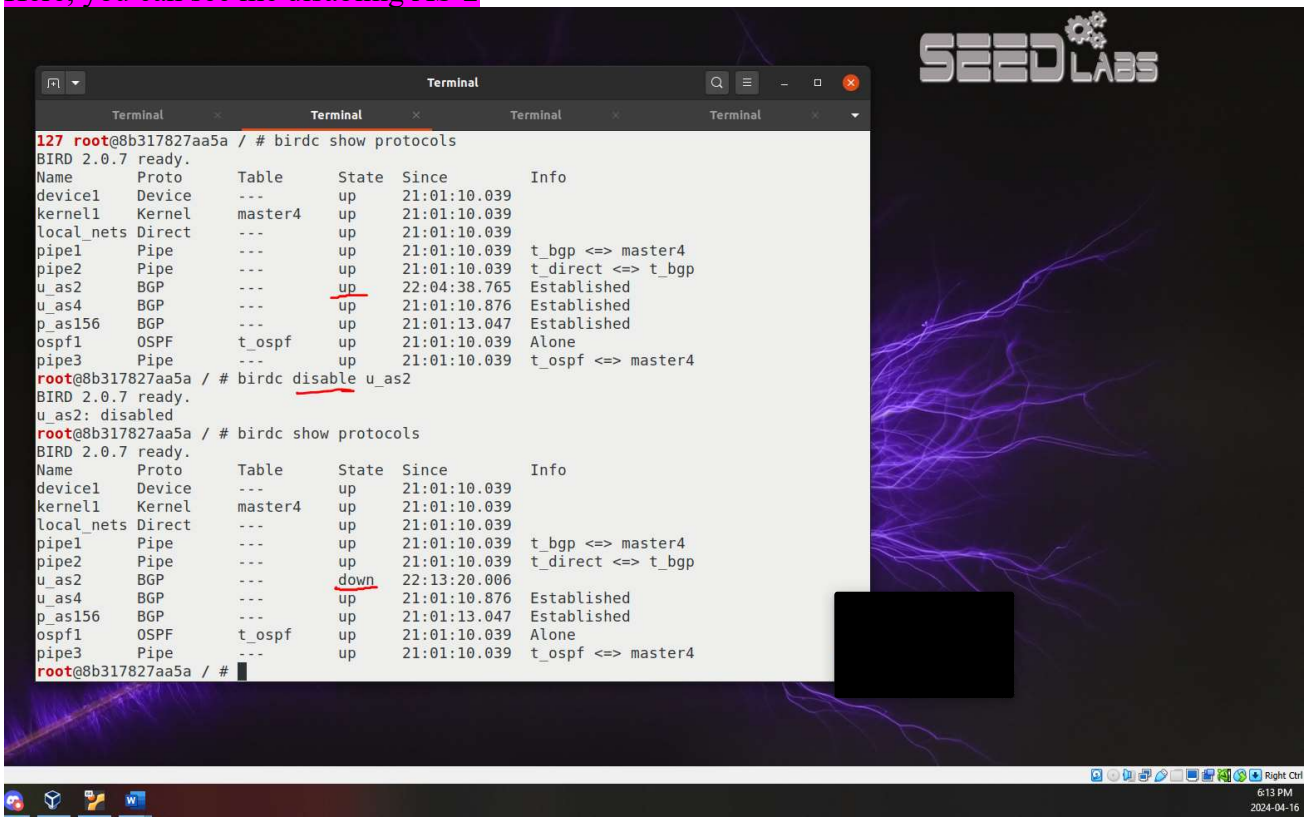
Task 1.b: Observing BGP UPDATE Messages

Here, you can see me searching for AS-155 and accessing that container to run the tcpdump command provided in the instructions (tcpdump -i any -w /tmp/bgp.pcap "tcp port 179").

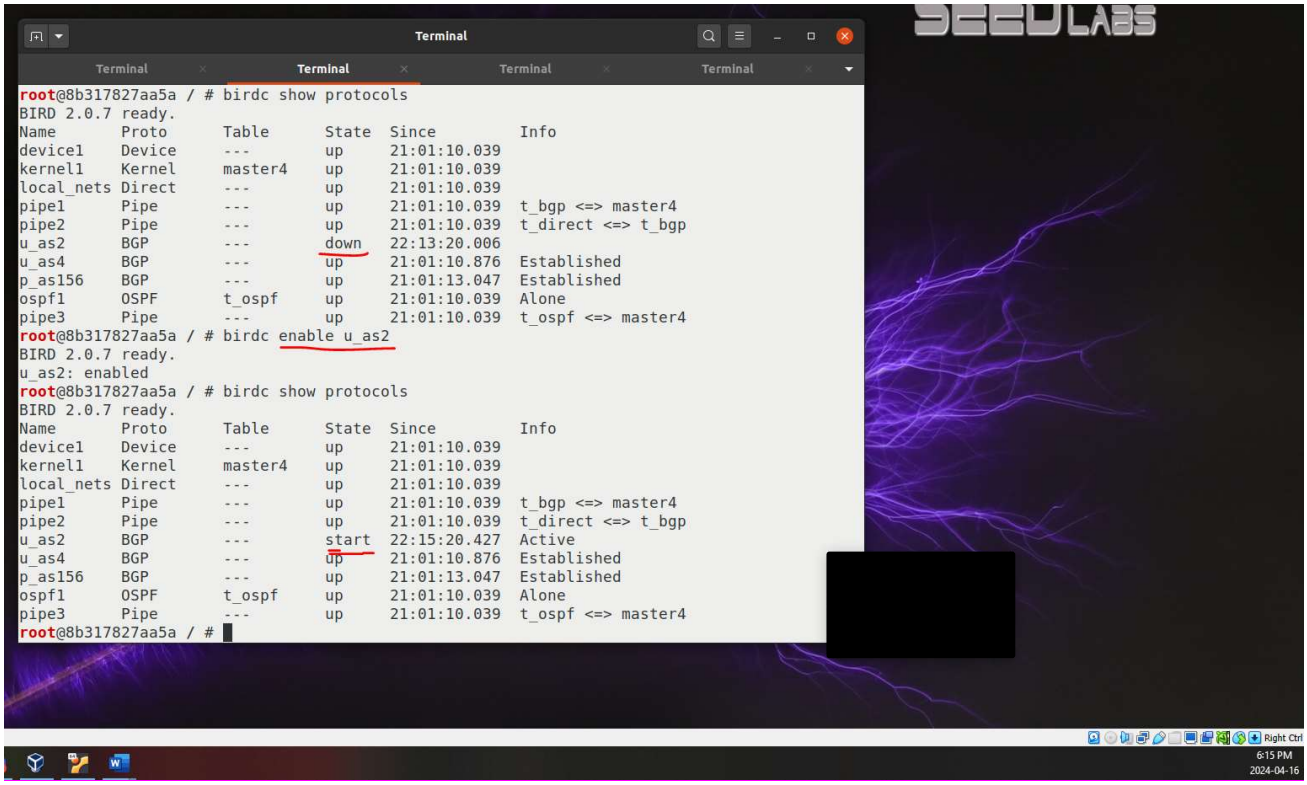


Now, I will be attempting to generate route withdrawal and route advertisement messages by going back to AS-155 and disabling and enabling AS-2 as a peer, respectively.

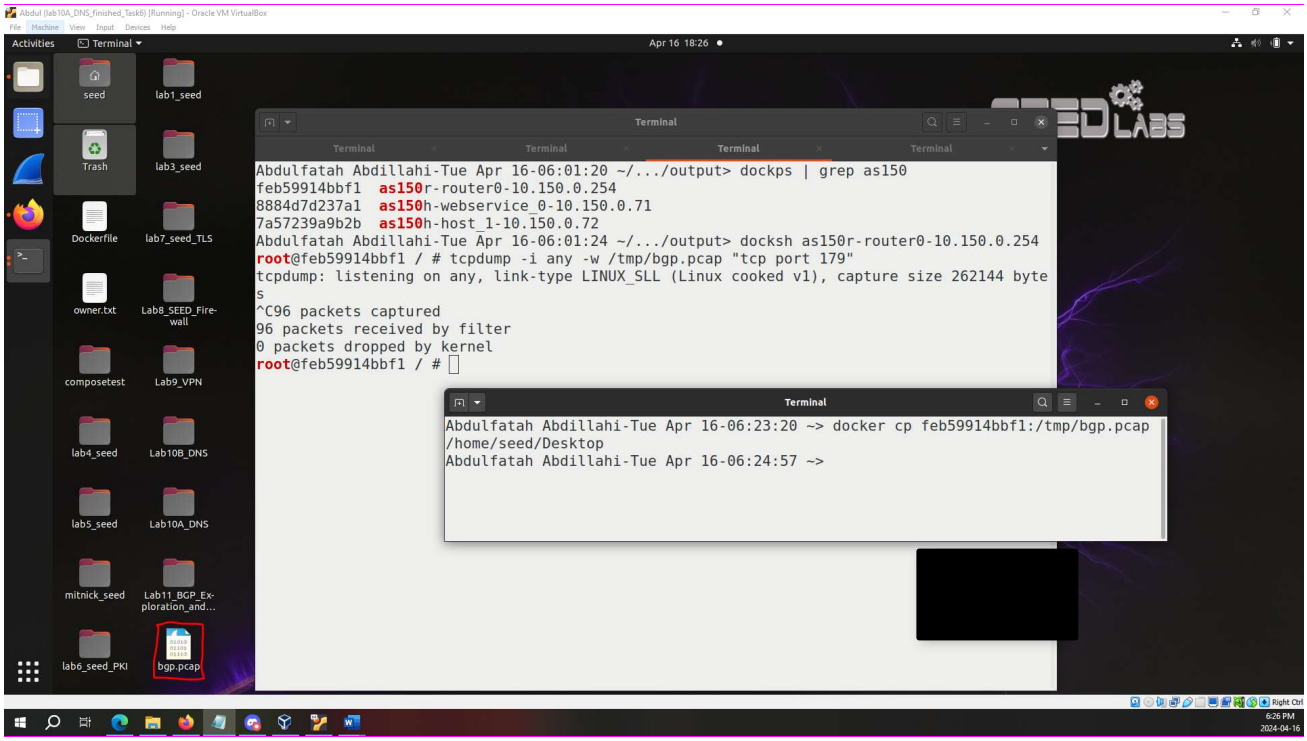
Here, you can see me disabling AS-2



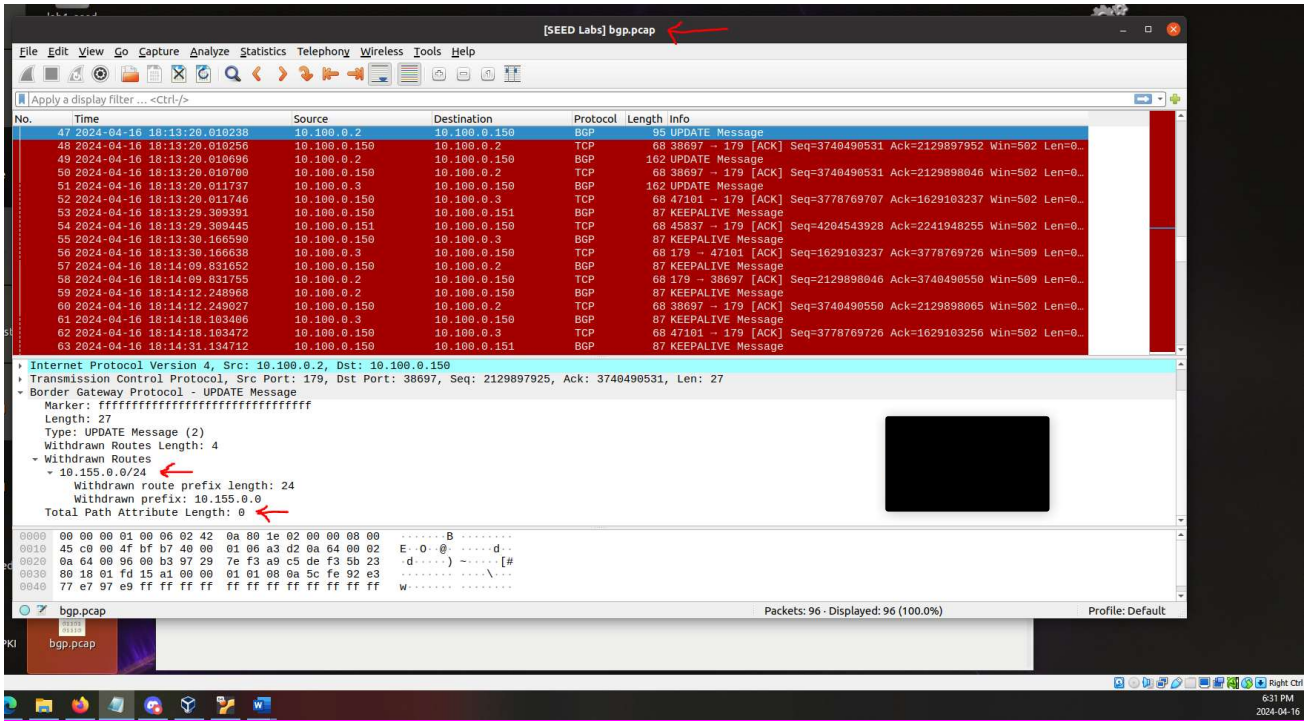
Here, you can see me enabling AS-2 once again.



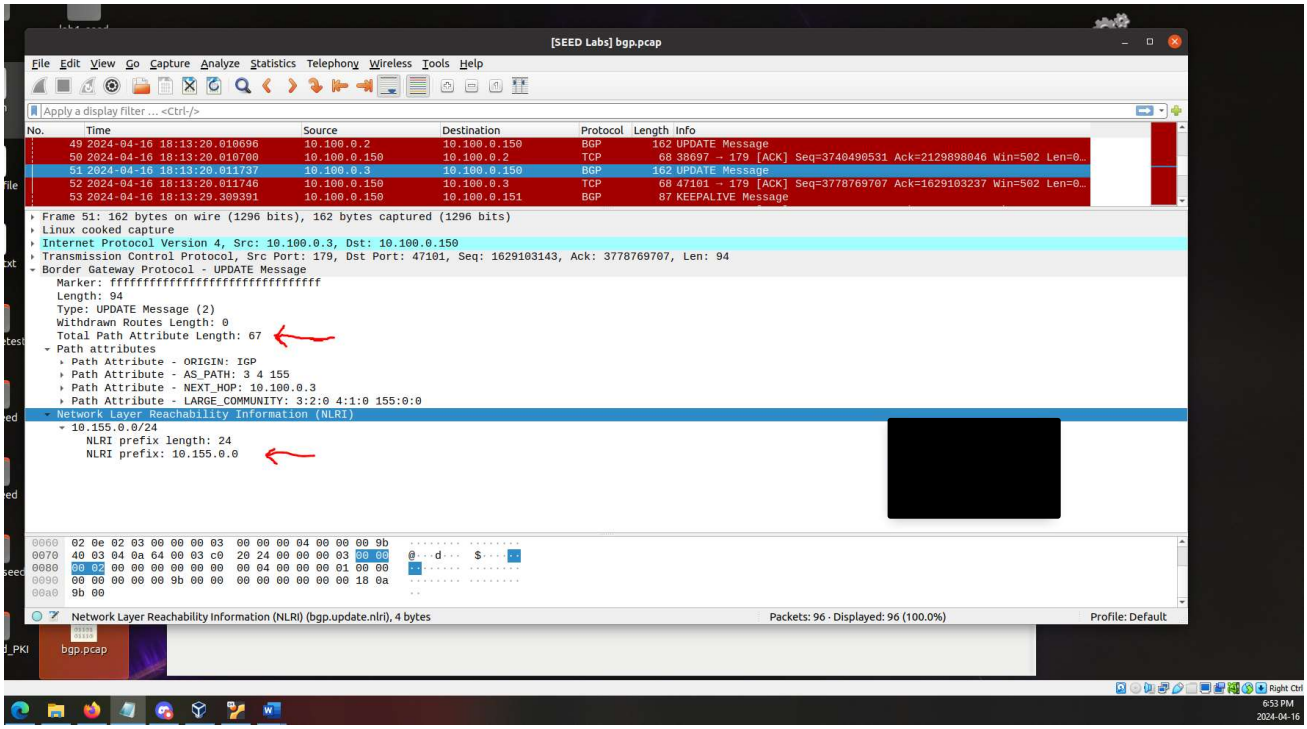
Then I stop the capture on AS-150 and copy the resulting pcap to the host computer using the "docker cp" command (from the host), to load it into Wireshark.



Here, you can see that I was successfully able to open the resulting pcap file. The first packet at the top displaying the UPDATE Message. If we expand below, we can also see that a route Withdrawal message was created specifically pointing to10.155.0.0/24 as the withdrawn route. We can also see that no advertisements we made as the total path attribute length is set to 0.



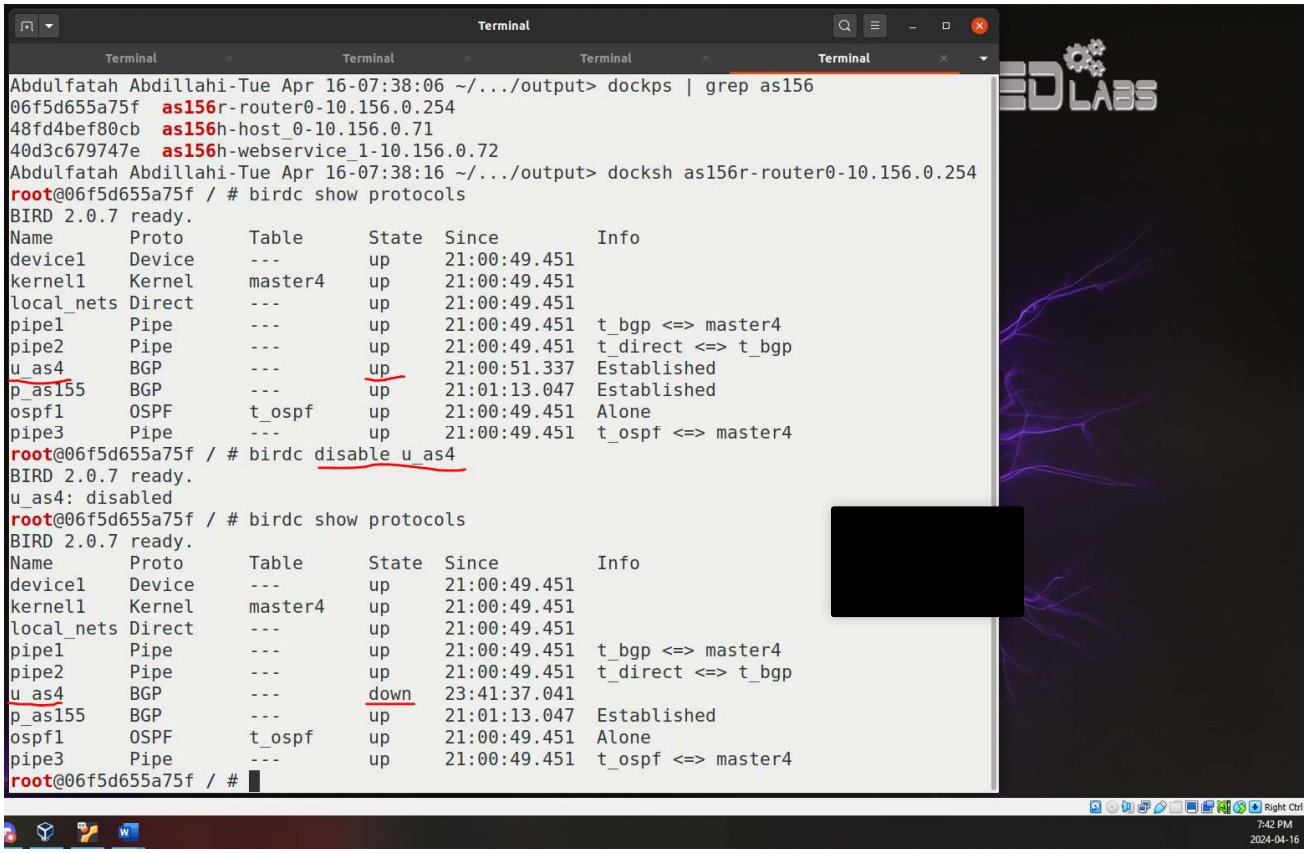
Here, we can see the next packet which is also another type of UPDATE Message. This time however, if we expand below, we will see that it displays route advertisement message specifically for the same network prefix in the previous screenshot (10.155.0.0/24). We can also see that unlike in the previous screenshot, this message is a is a route advertisement update as the total path attribute length is set to 67.



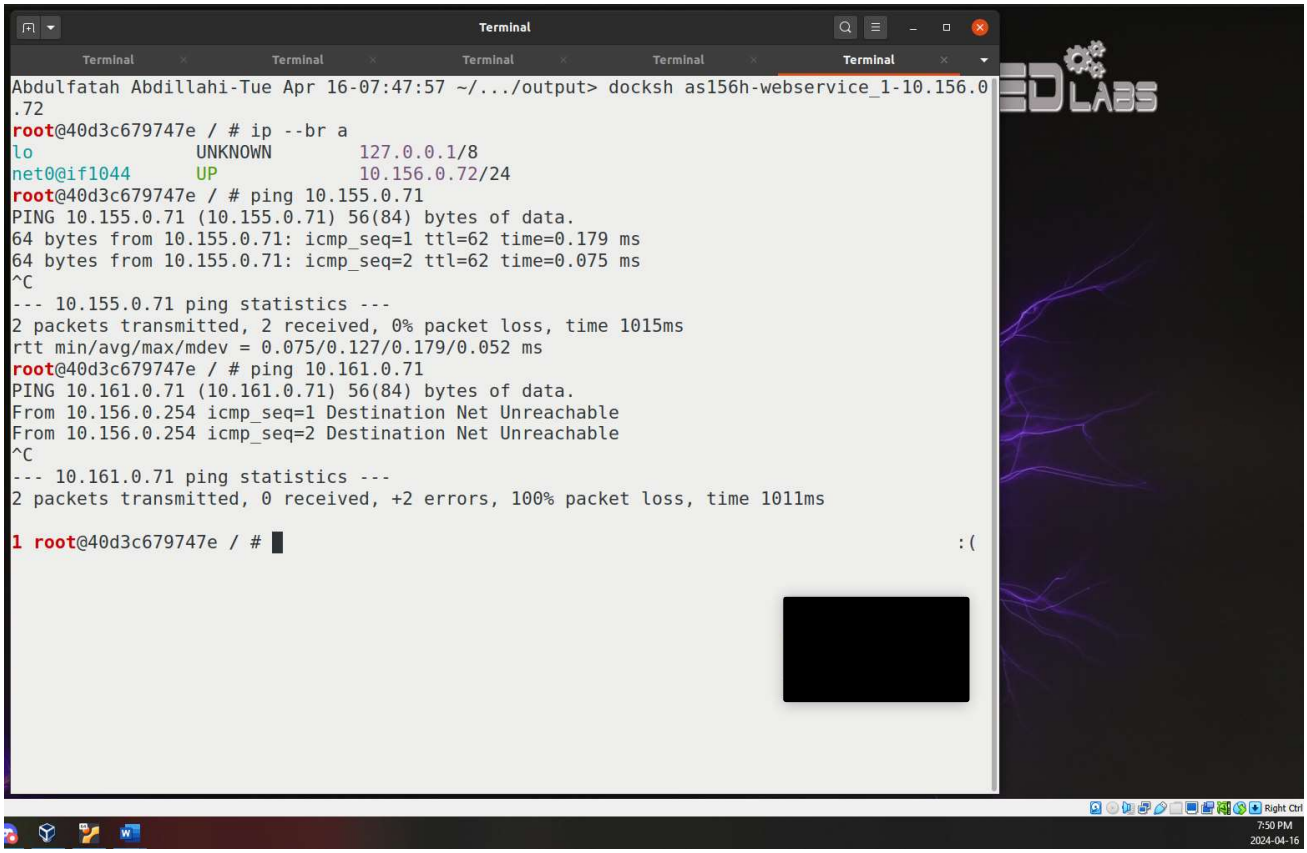
Task 1.c: Experimenting with Large Communities

Here, you can see me searching for AS-156 and accessing that container to disable the peering between AS-4 and AS-156. Since AS-4 is the only service provider for AS-156, this essentially

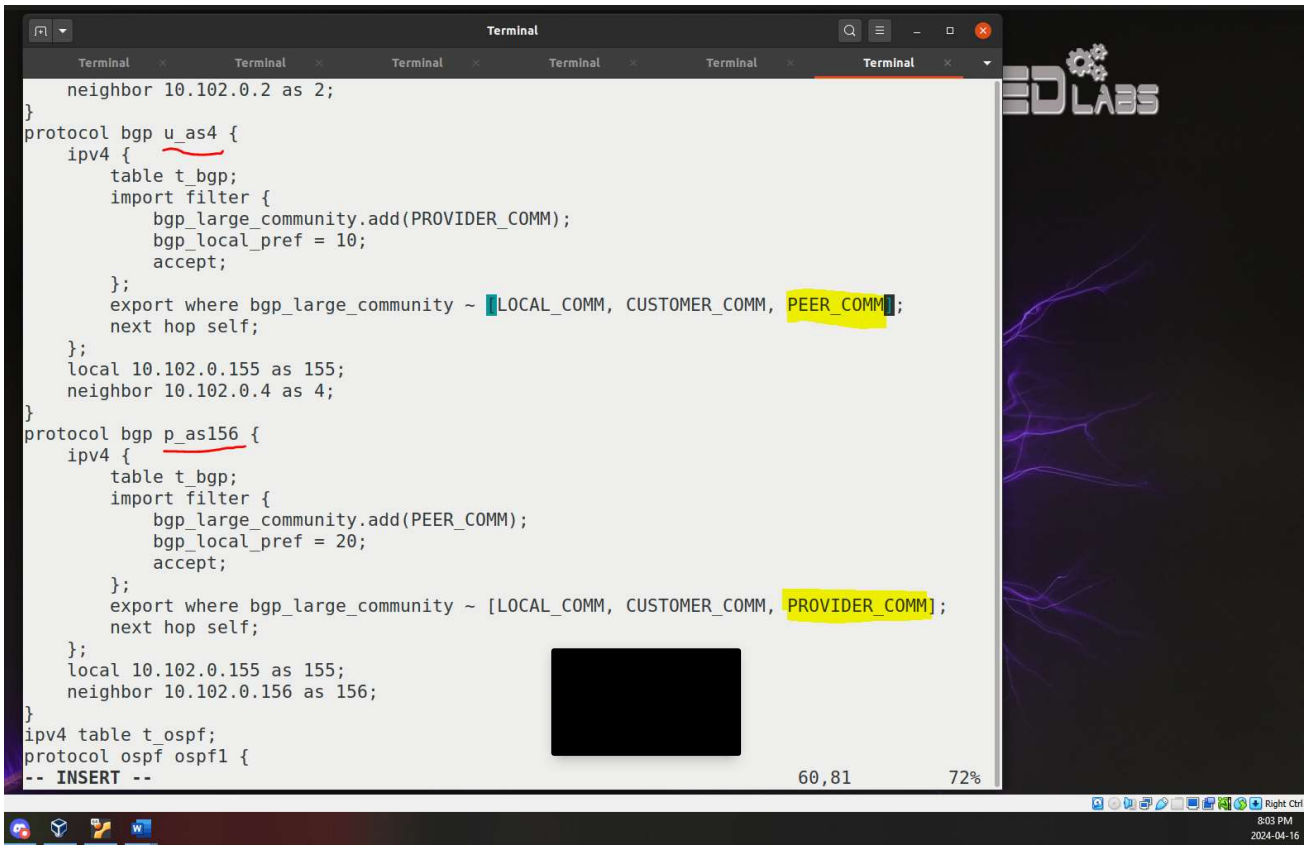
disconnects AS-156 from the Internet.



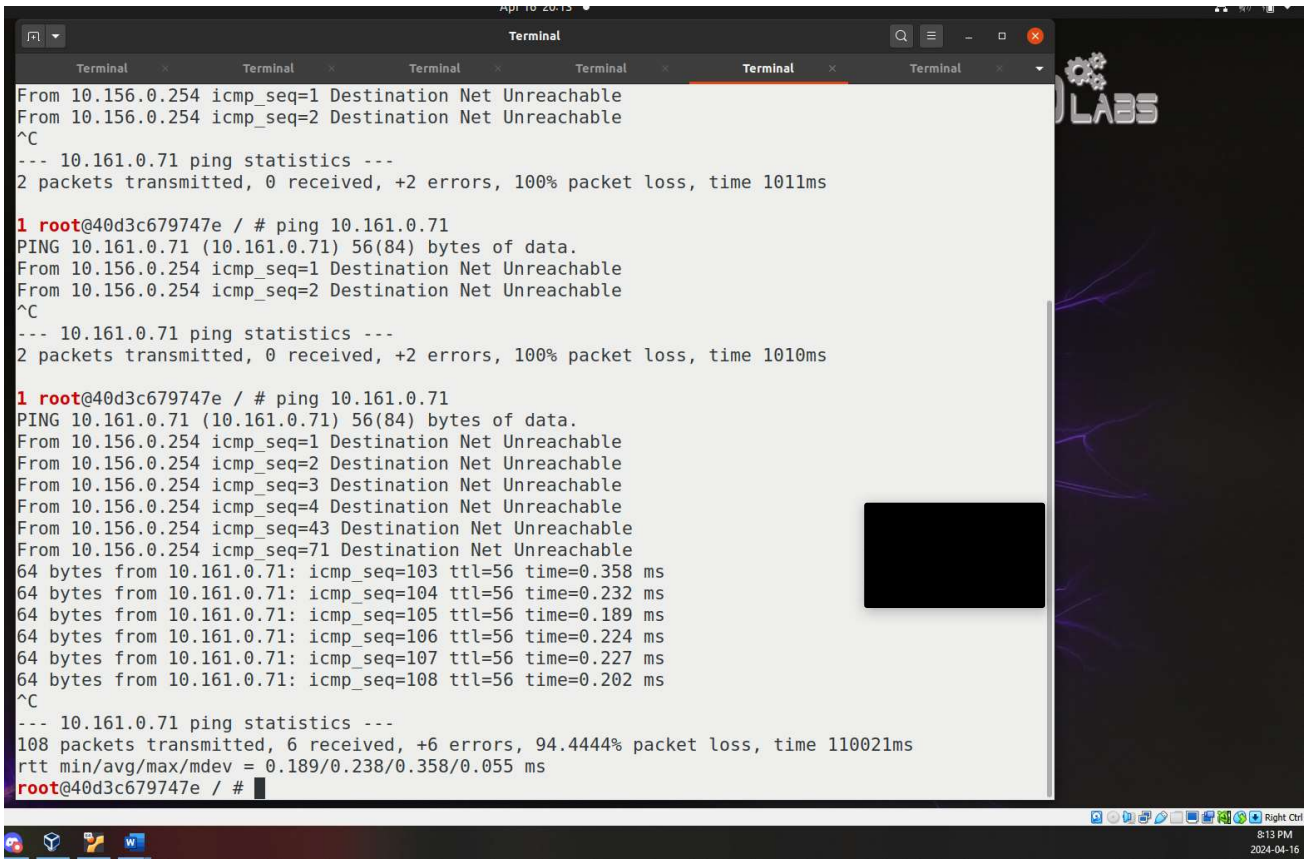
If we ping another host from one of the hosts in AS-156, we can see the following results:



We can see that 10.155.0.71 is still reachable, because it belongs to AS-155, which is still peered with AS-156. However, 10.161.0.71 (belonging to AS-161) cannot be reached, because nobody will route the packet for AS-156.
To fix this AS-156 needs to pay AS-155 so it can temporarily allow traffic to go through AS-155 to reach the Internet. We can do this by editing the **bird.conf** file on AS-155 and adding "PEER_COMM" under the *u_as4* section and adding the "PROVIDER_COMM" under the *p_as156* section.

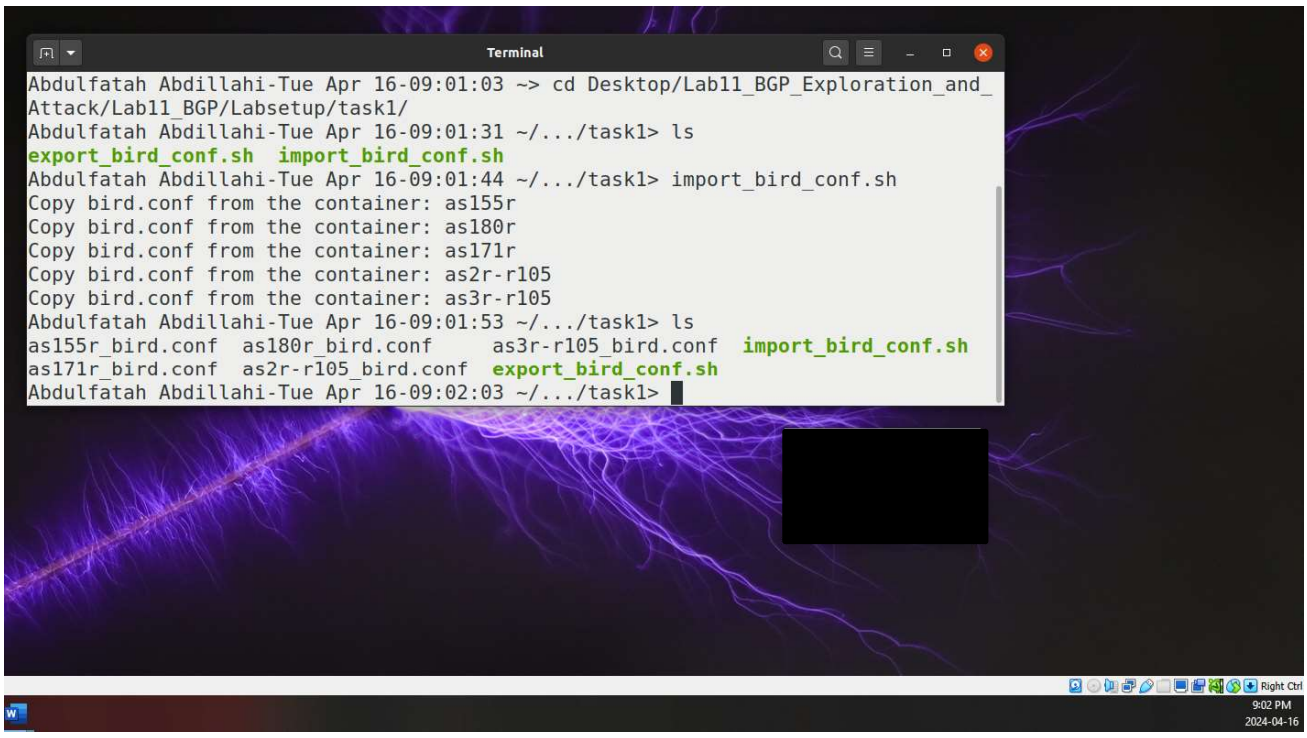


After saving the changes I attempted the ping one more time and as you can see this worked.



Task 1.d: Configuring AS-180

Here, you can see me importing the ***bird.conf*** files from the containers to the host.



Below you can see the piece of code I have appended to the bottom of the ***as180r_bird.conf*** file.

```
define LOCAL_COMM = (180, 0, 0);
```

```
define CUSTOMER_COMM = (180, 1, 0);
define PEER_COMM = (180, 2, 0);
define PROVIDER_COMM = (180, 3, 0);

ipv4 table t_bgp;

protocol pipe {
   table t_bgp;
   peer table master4;
   import none;
   export all;
}

protocol pipe {
   table t_direct;
   peer table t_bgp;
   import none;
   export filter {
      bgp_large_community.add(LOCAL_COMM);
      bgp_local_pref = 40;
      accept;
   };
}

protocol bgp b_as2 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PROVIDER_COMM);
         bgp_local_pref = 20;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.180.0.171 as 180;
   neighbor 10.105.0.2 as 105;
}

protocol bgp b_as3 {
   ipv4 {
      table t_bgp;
      import filter {
         bgp_large_community.add(PROVIDER_COMM);
         bgp_local_pref = 20;
         accept;
      };
      export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
      next hop self;
   };
   local 10.180.0.171 as 180;
```

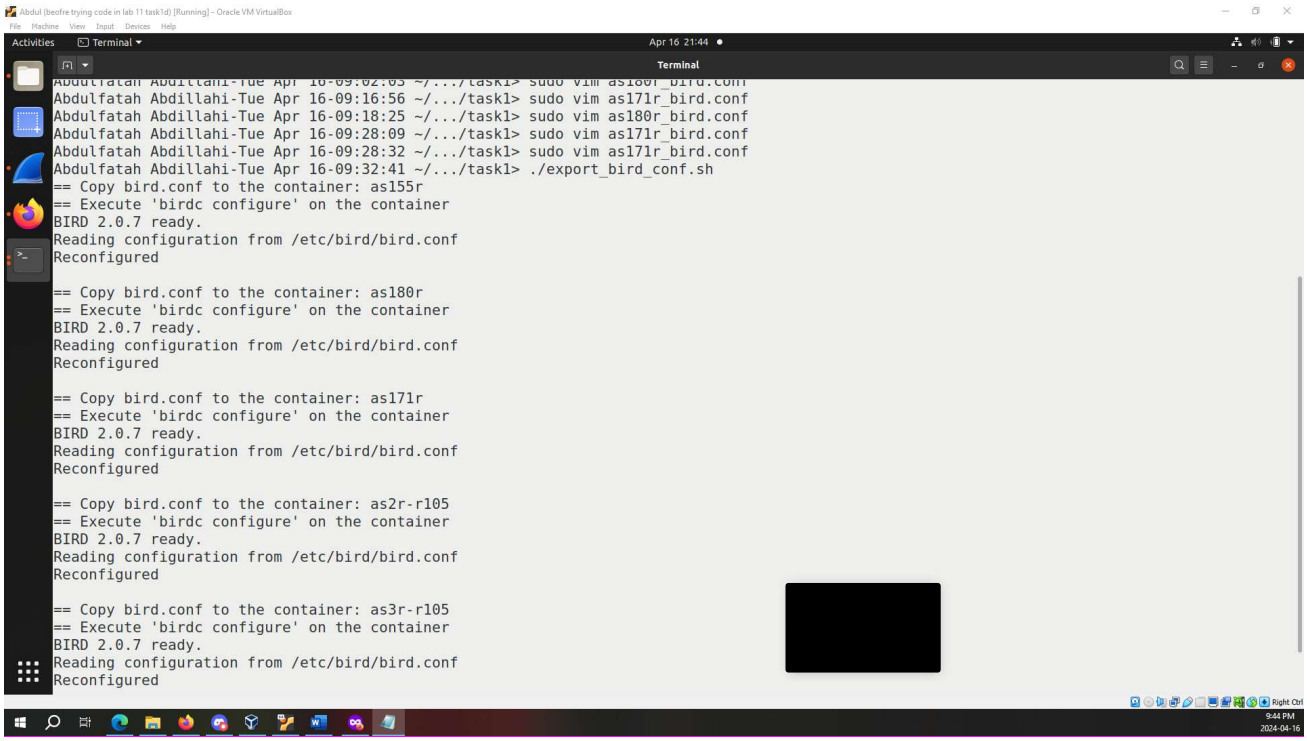
```
    neighbor 10.105.0.3 as 105;
}

protocol bgp b_as171 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(PROVIDER_COMM);
            bgp_local_pref = 20;
            accept;
        };
        export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
        next hop self;
    };
    local 10.105.0.180 as 180;
    neighbor 10.105.0.171 as 171;
}
```

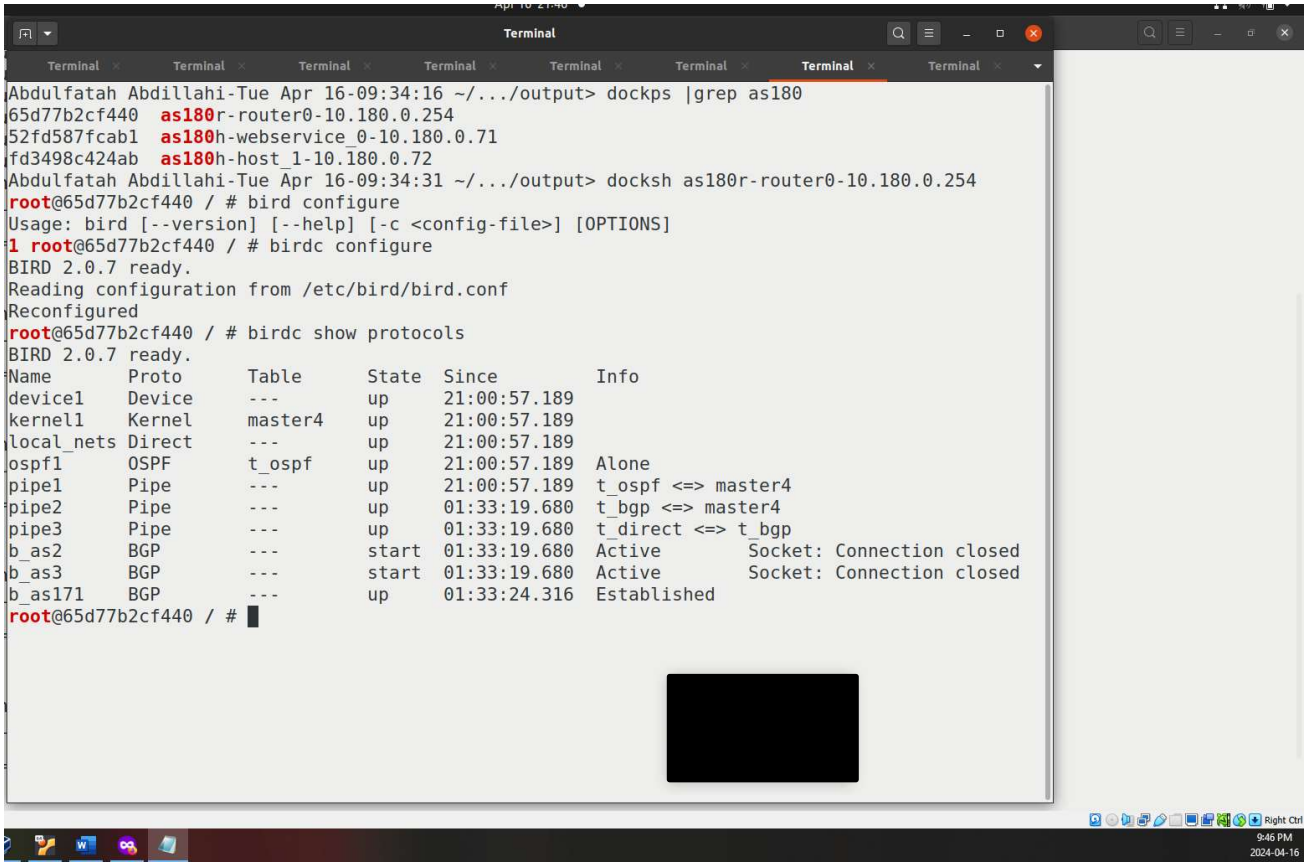
Below you can see the piece of code I have appended to the bottom of the ***as171r_bird.conf*** file.

```
protocol bgp b_as180 {
    ipv4 {
        table t_bgp;
        import filter {
            bgp_large_community.add(PROVIDER_COMM);
            bgp_local_pref = 20;
            accept;
        };
        export where bgp_large_community ~ [LOCAL_COMM, CUSTOMER_COMM];
        next hop self;
    };
    local 10.105.0.171 as 171;
    neighbor 10.105.0.180 as 180;
}
```

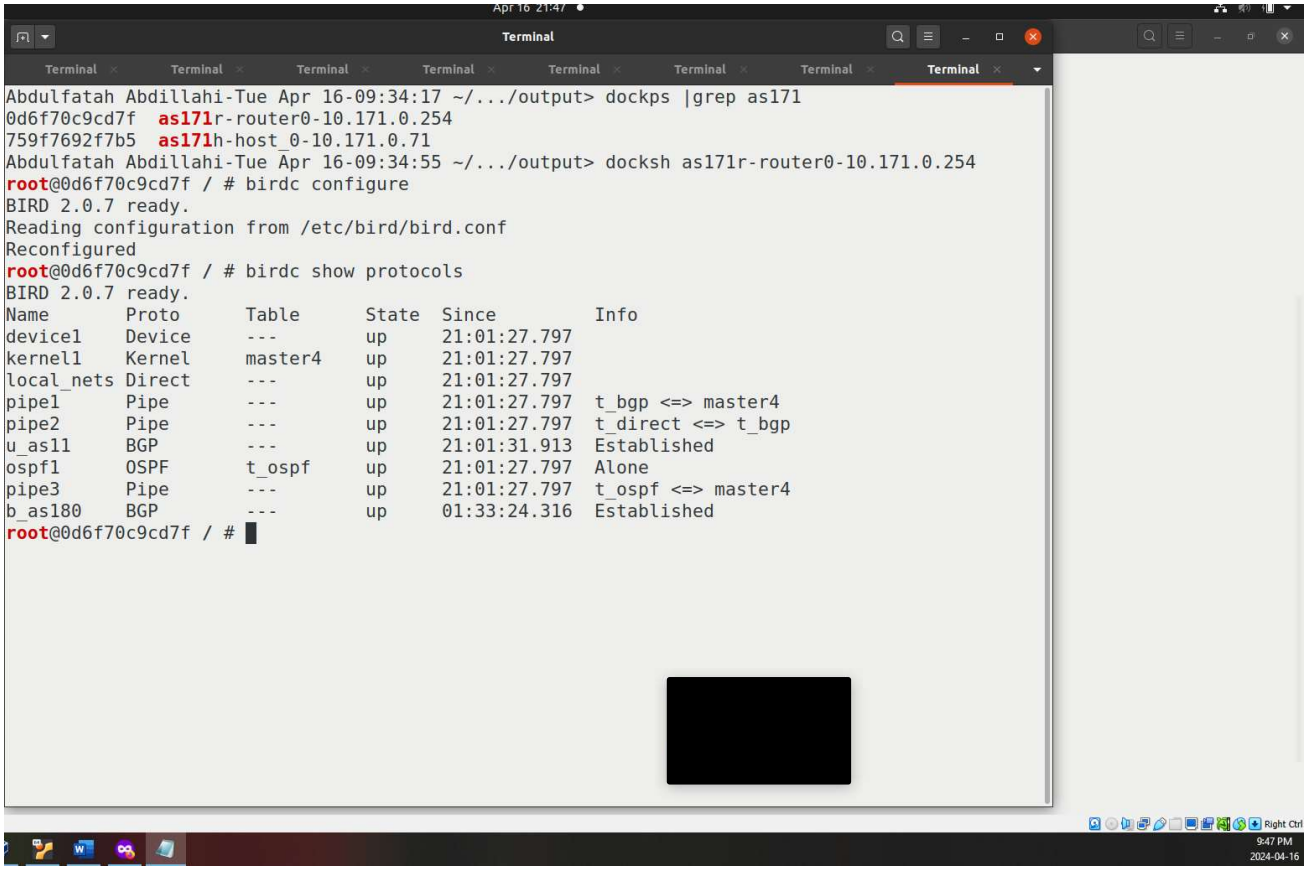
Here, you can see me exporting the ***bird.conf*** files from the host to the containers.



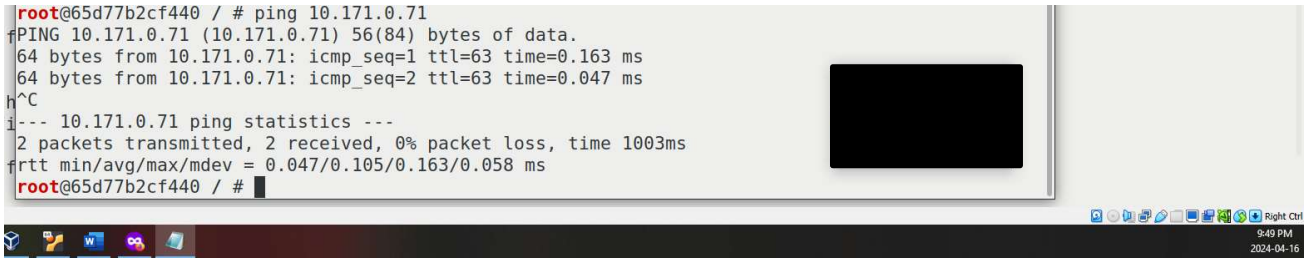
Here, you can see that the new peers were successfully created and added. By first looking at AS-180 we can see the new AS being reflected in the output.

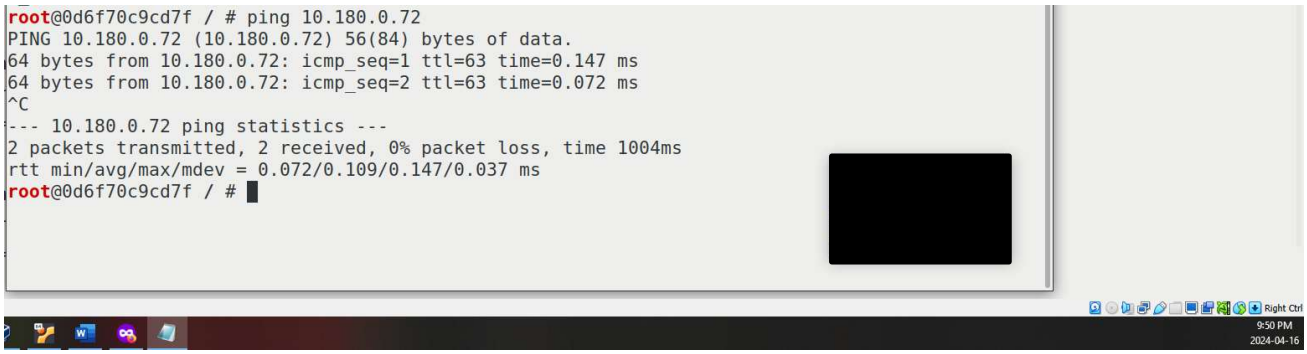


The same can also be said about AS-171.



Then to prove that this works I made sure to ping a host from the AS-171 network from AS-180 and vice versa. As you can see this was successful.



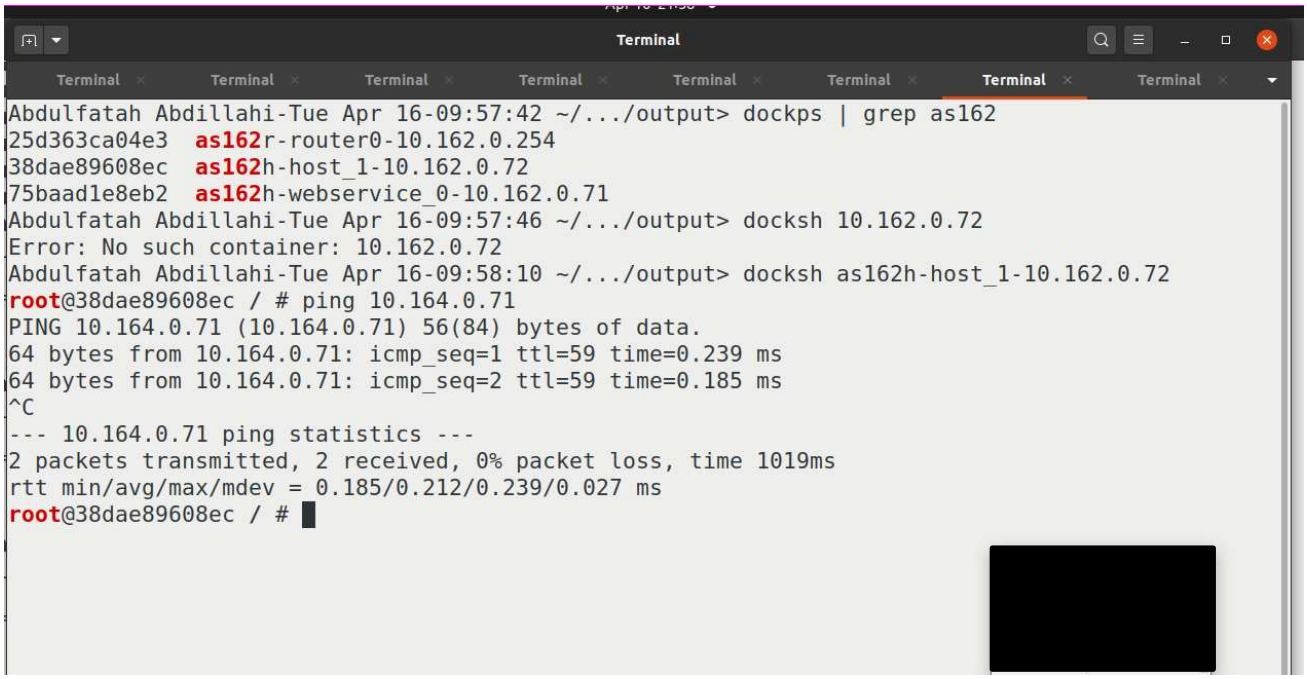


Then instructions also request a screenshot of traceroute, but this was not possible as traceroute was not installed on the containers before hand.

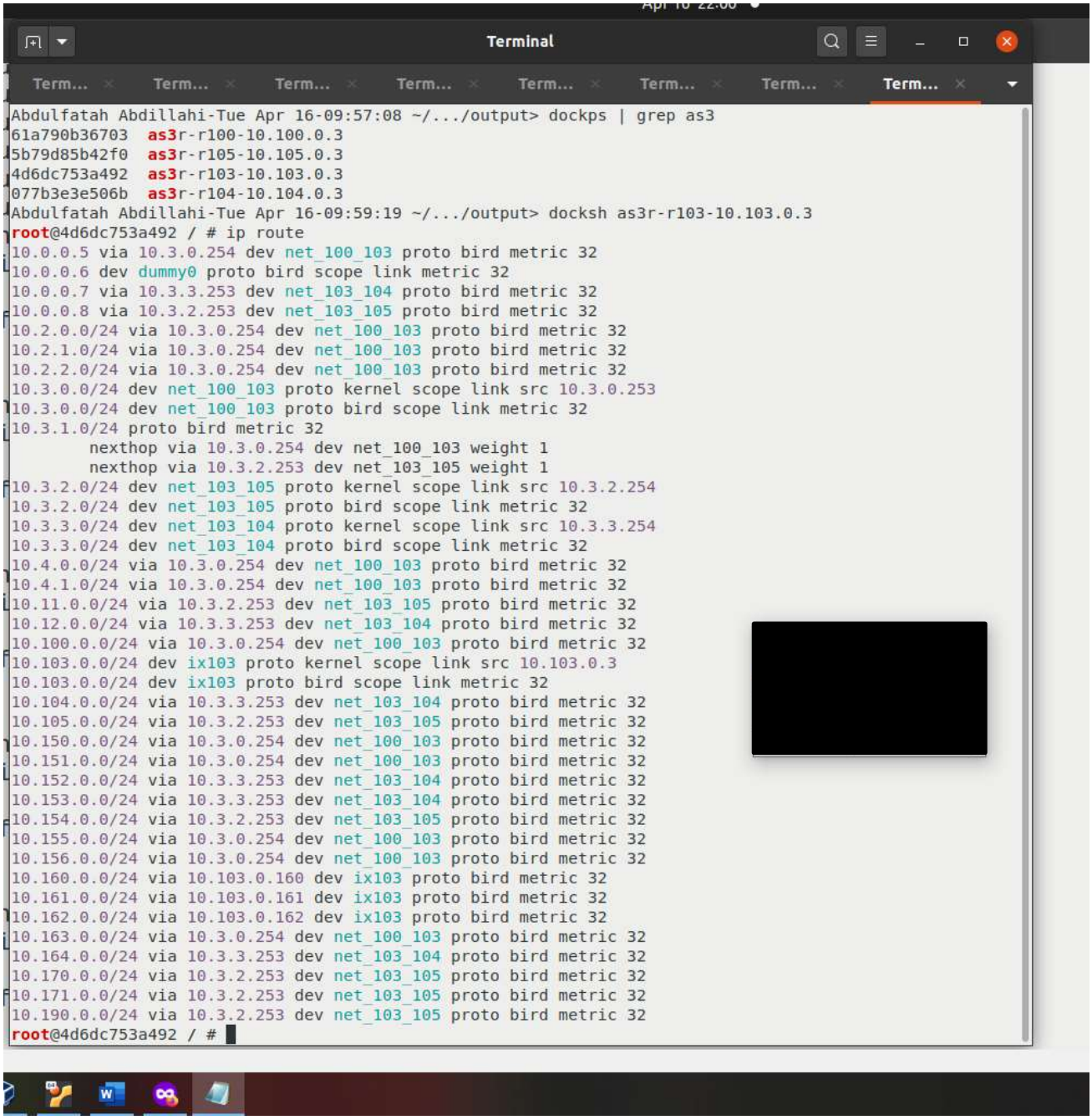
Task 2: Transit Autonomous System

Task 2.a: Experimenting with IBGP

Here, you can see me searching for AS-162 and accessing that container to ping 10.164.0.71 from a host within AS-162.



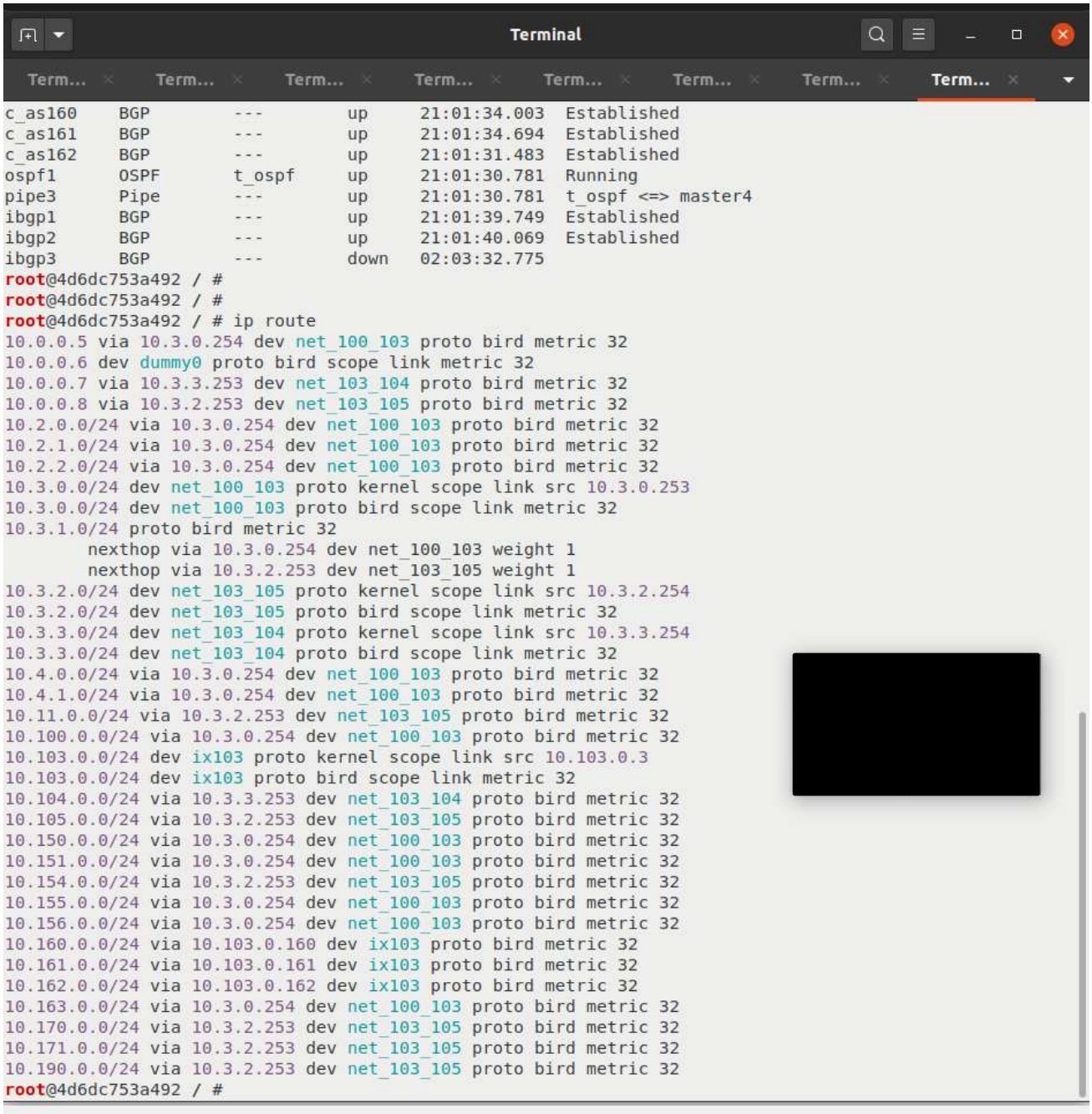
Then moving on to AS-3, this screenshot shows the routes before disabling IBGP.



Then I disabled the IBGP.

```
root@4d6dc753a492 / # birdc show protocols
BIRD 2.0.7 ready.
Name       Proto      Table      State  Since         Info
device1    Device     ---        up     21:01:30.781
kernel1    Kernel     master4    up     21:01:30.781
local_nets Direct     ---        up     21:01:30.781
pipe1      Pipe       ---        up     21:01:30.781  t_bgp <=> master4
pipe2      Pipe       ---        up     21:01:30.781  t_direct <=> t_bgp
c_as160    BGP        ---        up     21:01:34.004  Established
c_as161    BGP        ---        up     21:01:34.695  Established
c_as162    BGP        ---        up     21:01:31.483  Established
ospf1      OSPF       t_ospf     up     21:01:30.781  Running
pipe3      Pipe       ---        up     21:01:30.781  t_ospf <=> master4
ibgp1      BGP        ---        up     21:01:39.749  Established
ibgp2      BGP        ---        up     21:01:40.069  Established
ibgp3      BGP        ---        up     21:01:38.228  Established
root@4d6dc753a492 / # birdc disable ibgp3
BIRD 2.0.7 ready.
ibgp3: disabled
root@4d6dc753a492 / # birdc show protocols
BIRD 2.0.7 ready.
Name       Proto      Table      State  Since         Info
device1    Device     ---        up     21:01:30.781
kernel1    Kernel     master4    up     21:01:30.781
local_nets Direct     ---        up     21:01:30.781
pipe1      Pipe       ---        up     21:01:30.781  t_bgp <=> master4
pipe2      Pipe       ---        up     21:01:30.781  t_direct <=> t_bgp
c_as160    BGP        ---        up     21:01:34.003  Established
c_as161    BGP        ---        up     21:01:34.694  Established
c_as162    BGP        ---        up     21:01:31.483  Established
ospf1      OSPF       t_ospf     up     21:01:30.781  Running
pipe3      Pipe       ---        up     21:01:30.781  t_ospf <=> master4
ibgp1      BGP        ---        up     21:01:39.749  Established
ibgp2      BGP        ---        up     21:01:40.069  Established
ibgp3      BGP        ---        down   02:03:32.775
root@4d6dc753a492 / #
```

Now I checked the routes again after disabling the IBGP. I noticed that the routes to several different networks were removed (namely 10.152.0.0/24, 10.153.0.0/24, and 10.164.0.0/24). This demonstrates the direct impact of IBGP3 on routing decisions within the network. Disabling this protocol changed the available routes, which could hypothetically be exploited by attackers to manipulate routing paths.



Task 2.b: Experimenting with IGP

The above screenshot from the previous task shows what the current route already looks like. As you can see, I will now disable the OSPF1 protocol as per the instructions.

```
root@4d6dc753a492 / #
root@4d6dc753a492 / #
root@4d6dc753a492 / # birdc disable ospf1
BIRD 2.0.7 ready.
ospf1: disabled
root@4d6dc753a492 / # birdc show protocols
BIRD 2.0.7 ready.
Name       Proto      Table      State  Since         Info
device1    Device     ---        up     21:01:30.781
kernel1    Kernel     master4    up     21:01:30.781
local_nets Direct     ---        up     21:01:30.781
pipe1      Pipe       ---        up     21:01:30.781  t_bgp <=> master4
pipe2      Pipe       ---        up     21:01:30.781  t_direct <=> t_bgp
c_as160    BGP        ---        up     21:01:34.004  Established
c_as161    BGP        ---        up     21:01:34.695  Established
c_as162    BGP        ---        up     21:01:31.484  Established
ospf1      OSPF       t_ospf     down   02:14:29.620
pipe3      Pipe       ---        up     21:01:30.781  t_ospf <=> master4
ibgp1      BGP        ---        up     21:01:39.750  Established
ibgp2      BGP        ---        up     21:01:40.069  Established
ibgp3      BGP        ---        down   02:03:32.776
root@4d6dc753a492 / #
```

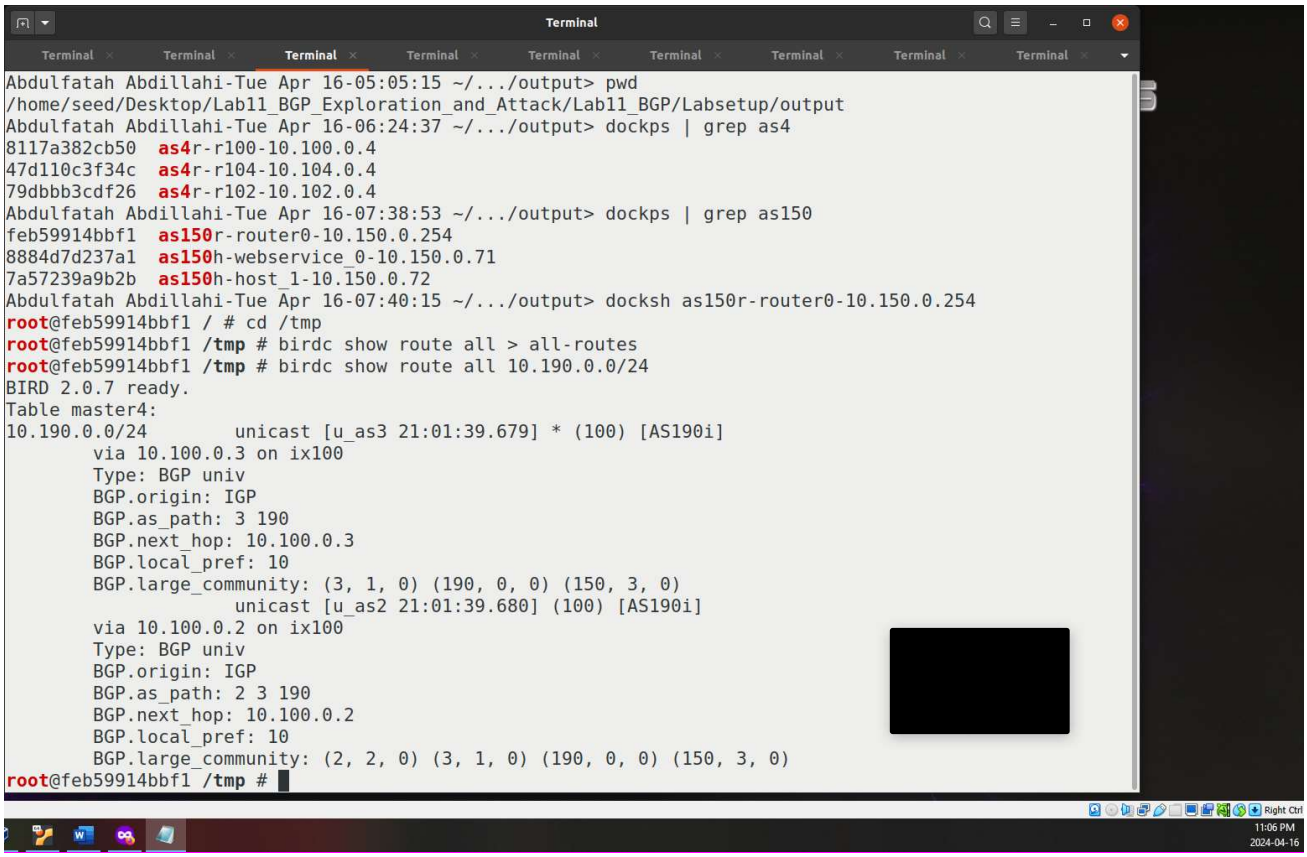
Then I checked the routes again after disabling ospf. I noticed that the routes to directly connected networks via Bird were only present. The fact that the screenshot provided lacks OSPF routes emphasizes how important Interior Gateway Protocols (IGPs), are for transit autonomous systems (ASes). Within an AS, IGPs are in charge of dynamically sharing routing data to guarantee dependable and effective packet forwarding. The network's adaptability, scalability, and fault tolerance would be severely limited if it had to rely on manual configurations in the absence of a functional IGP. The efficient operation of transit ASes, which are in charge of forwarding traffic between various networks and autonomous systems on the internet, depends on maintaining a strong IGP.

```
root@4d6dc753a492 / # ip route
unreachable 10.2.0.0/24 proto bird metric 32
unreachable 10.2.1.0/24 proto bird metric 32
unreachable 10.2.2.0/24 proto bird metric 32
10.3.0.0/24 dev net_100_103 proto kernel scope link src 10.3.0.253
10.3.0.0/24 dev net_100_103 proto bird scope link metric 32
unreachable 10.3.1.0/24 proto bird metric 32
10.3.2.0/24 dev net_103_105 proto kernel scope link src 10.3.2.254
10.3.2.0/24 dev net_103_105 proto bird scope link metric 32
10.3.3.0/24 dev net_103_104 proto kernel scope link src 10.3.3.254
10.3.3.0/24 dev net_103_104 proto bird scope link metric 32
unreachable 10.4.0.0/24 proto bird metric 32
unreachable 10.4.1.0/24 proto bird metric 32
unreachable 10.11.0.0/24 proto bird metric 32
10.103.0.0/24 dev ix103 proto kernel scope link src 10.103.0.3
unreachable 10.150.0.0/24 proto bird metric 32
unreachable 10.151.0.0/24 proto bird metric 32
unreachable 10.154.0.0/24 proto bird metric 32
unreachable 10.155.0.0/24 proto bird metric 32
unreachable 10.156.0.0/24 proto bird metric 32
10.160.0.0/24 via 10.103.0.160 dev ix103 proto bird metric 32
10.161.0.0/24 via 10.103.0.161 dev ix103 proto bird metric 32
10.162.0.0/24 via 10.103.0.162 dev ix103 proto bird metric 32
unreachable 10.163.0.0/24 proto bird metric 32
unreachable 10.170.0.0/24 proto bird metric 32
unreachable 10.171.0.0/24 proto bird metric 32
unreachable 10.190.0.0/24 proto bird metric 32
root@4d6dc753a492 / #
```

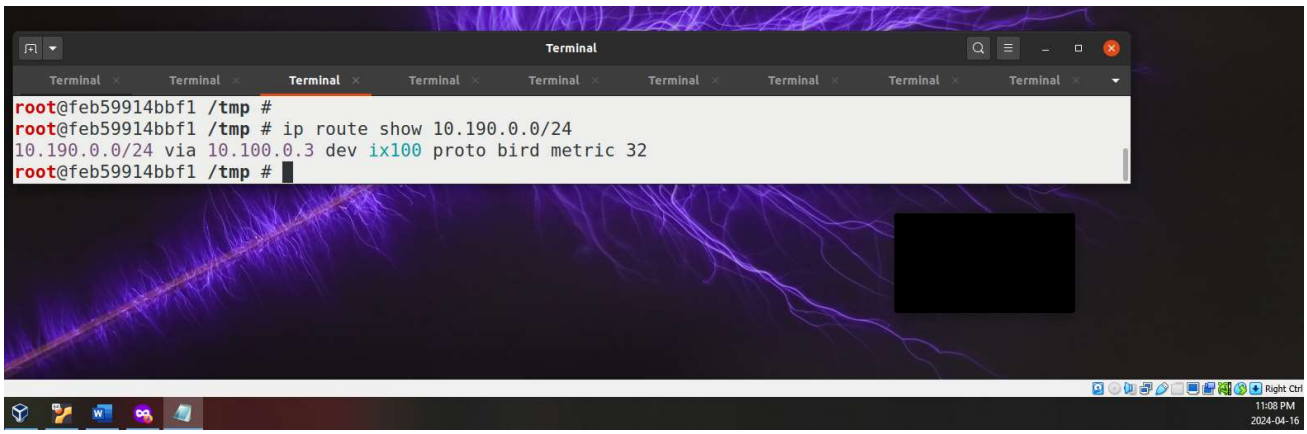
Task 3: Path Selection

Task 3.a.

Here, you can see me redirecting all the routes to a file called “all-routes”. I've chosen to utilize the network prefix "10.190.0.0/24" after looking over the data because it may be reached via two different routes that were obtained from autonomous systems AS-3 and AS-2. Every route starts with an Interior Gateway Protocol (IGP) and has a unique next hop. AS-2's path is longer, with an AS path of 2 3 190, than AS-3's, which has an AS path of 3 190. Both routes have the same local preference rating of 10, notwithstanding their differences. In this situation, the shorter AS path of AS3's route makes it the recommended way to get to the destination.

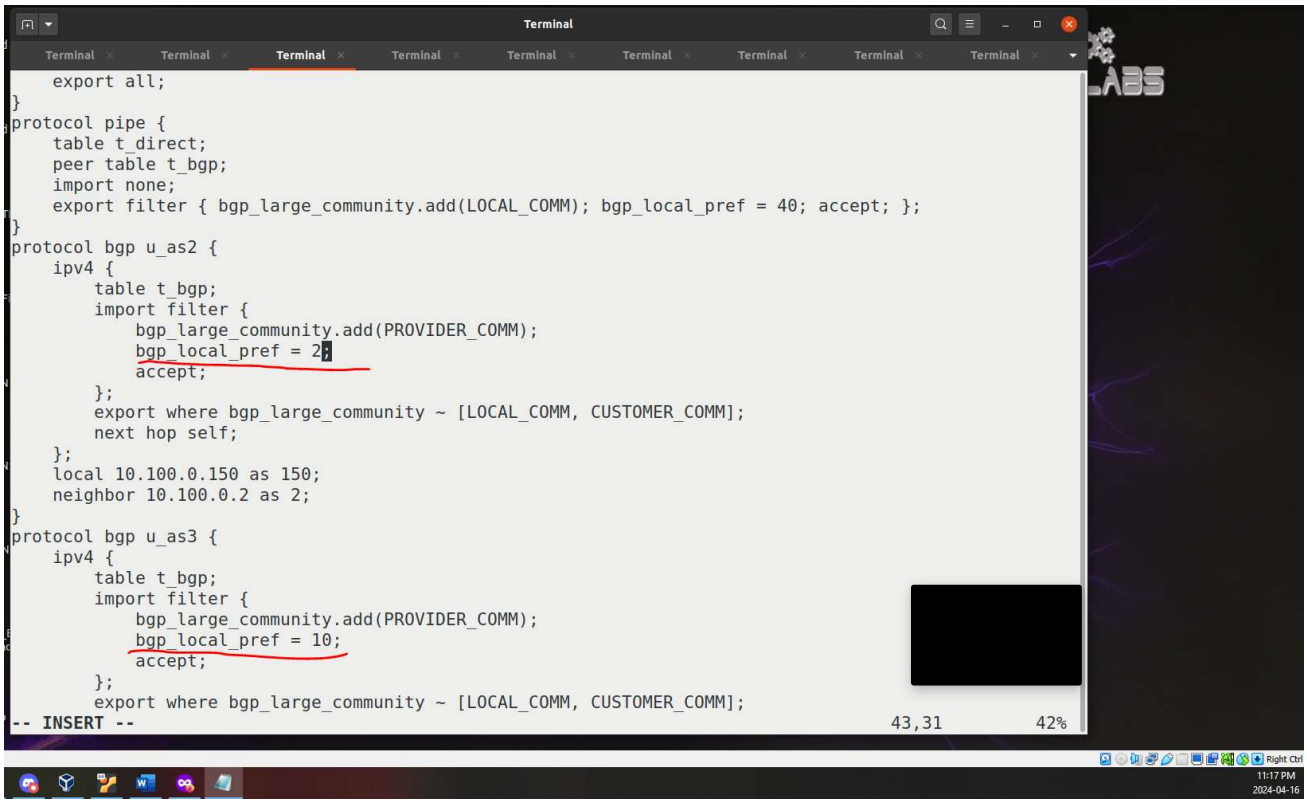


Here, you can see the routes used for my desired network prefix. You can clearly see that the route chosen by the kernel aligns with the one selected by the BGP router.



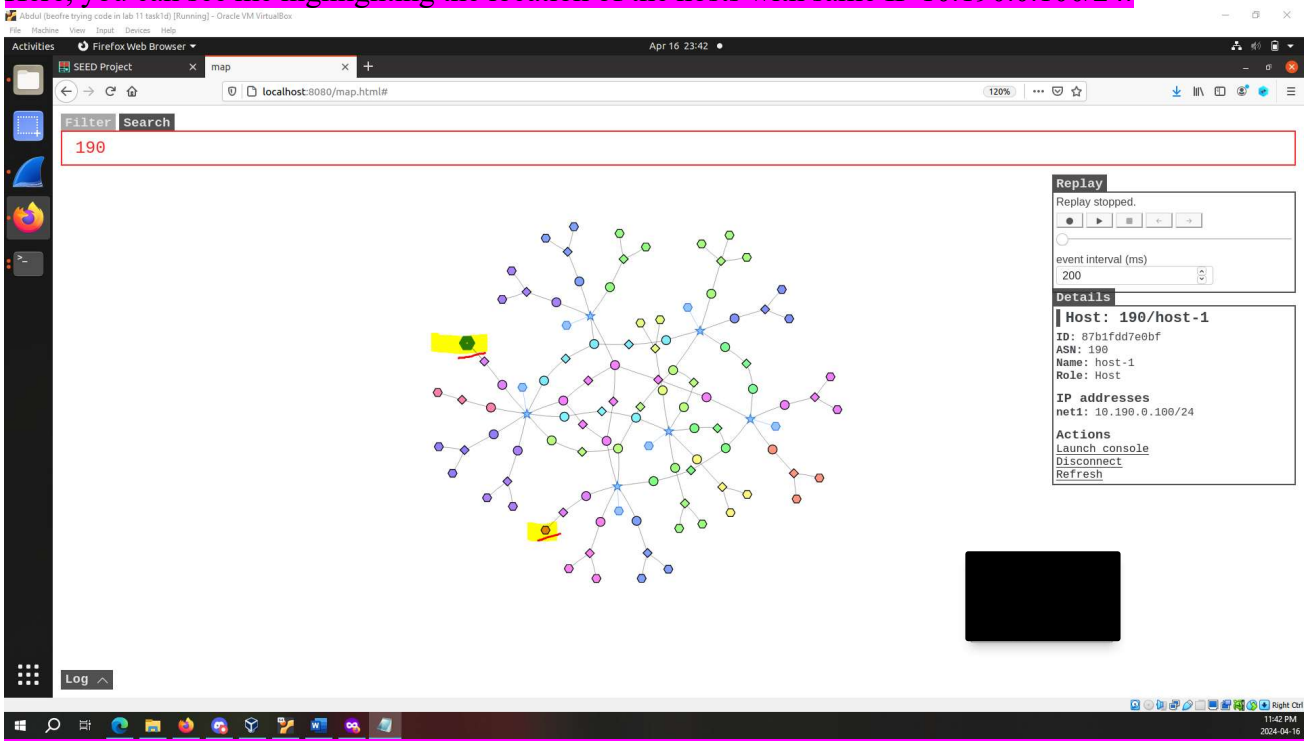
Task 3.b.

As the main upstream link, AS-3 is intended to be prioritized in the BGP configuration for AS-150. To do this, routes received from AS-3 are given a higher local preference value of 10. As the backup connection, AS-2, on the other hand, is configured with a lower local preference value of 2. With this setup, AS-3 will be the primary connection used by AS-150 for all inbound and outbound traffic, with AS-2 serving as a backup link only in case the AS-3 link fails.

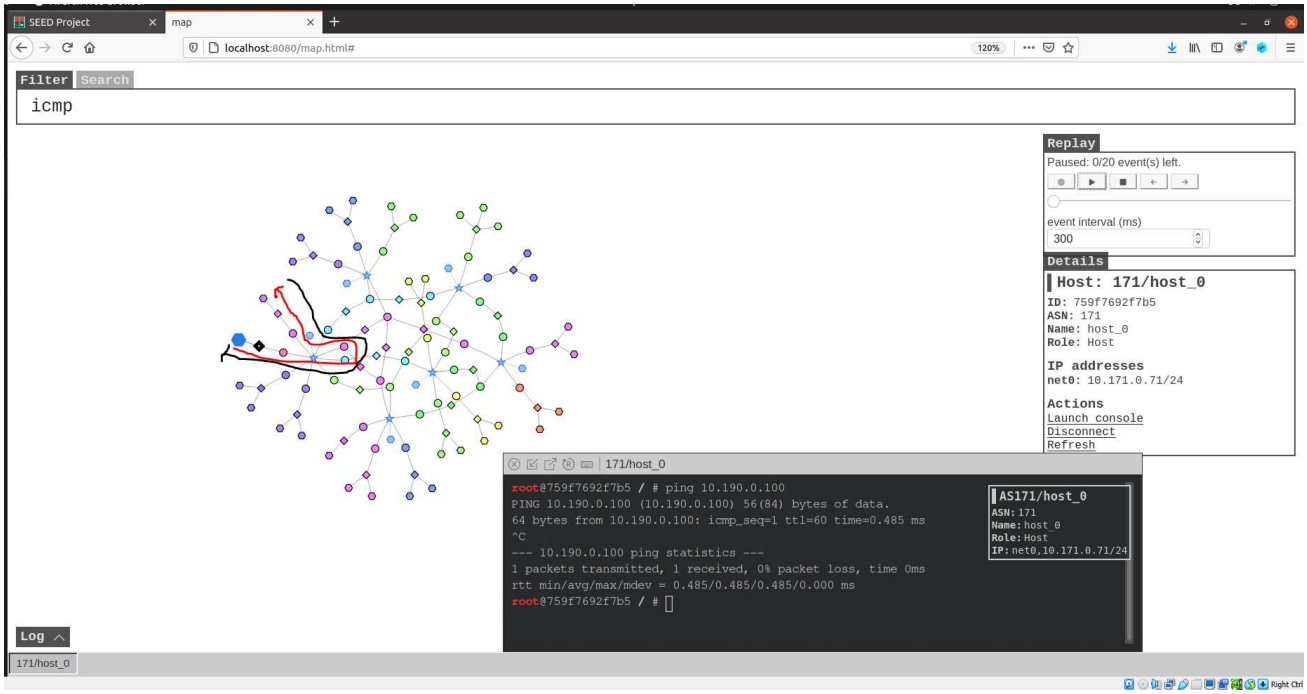


Task 4: IP Anycast

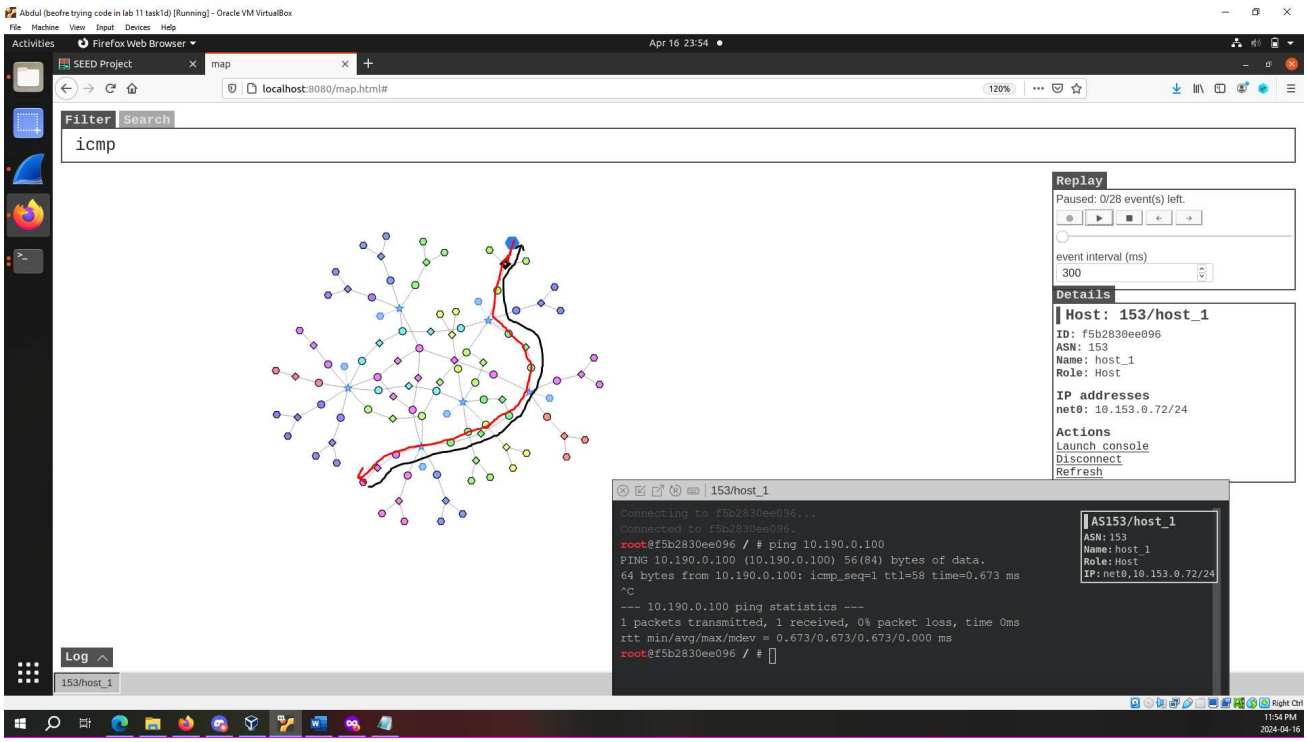
Here, you can see me highlighting the location of the hosts with same IP 10.190.0.100/24.



In order to determine the path taken, the screenshot shows my ping activity from a chosen host with the IP address "10.171.0.71/24" to the destination "10.190.0.100". The ICMP ping reply path is shown in black, while the path used by the ICMP ping request to reach "10.190.0.100" is highlighted in red.



The same thing is done in the below screenshot from a different machine. In order to determine the path taken, the screenshot shows my ping activity from a host with the IP address "10.153.0.72/24" to the destination "10.190.0.100". The ICMP ping reply path is shown in black, while the path used by the ICMP ping request to reach "10.190.0.100" is highlighted in red.

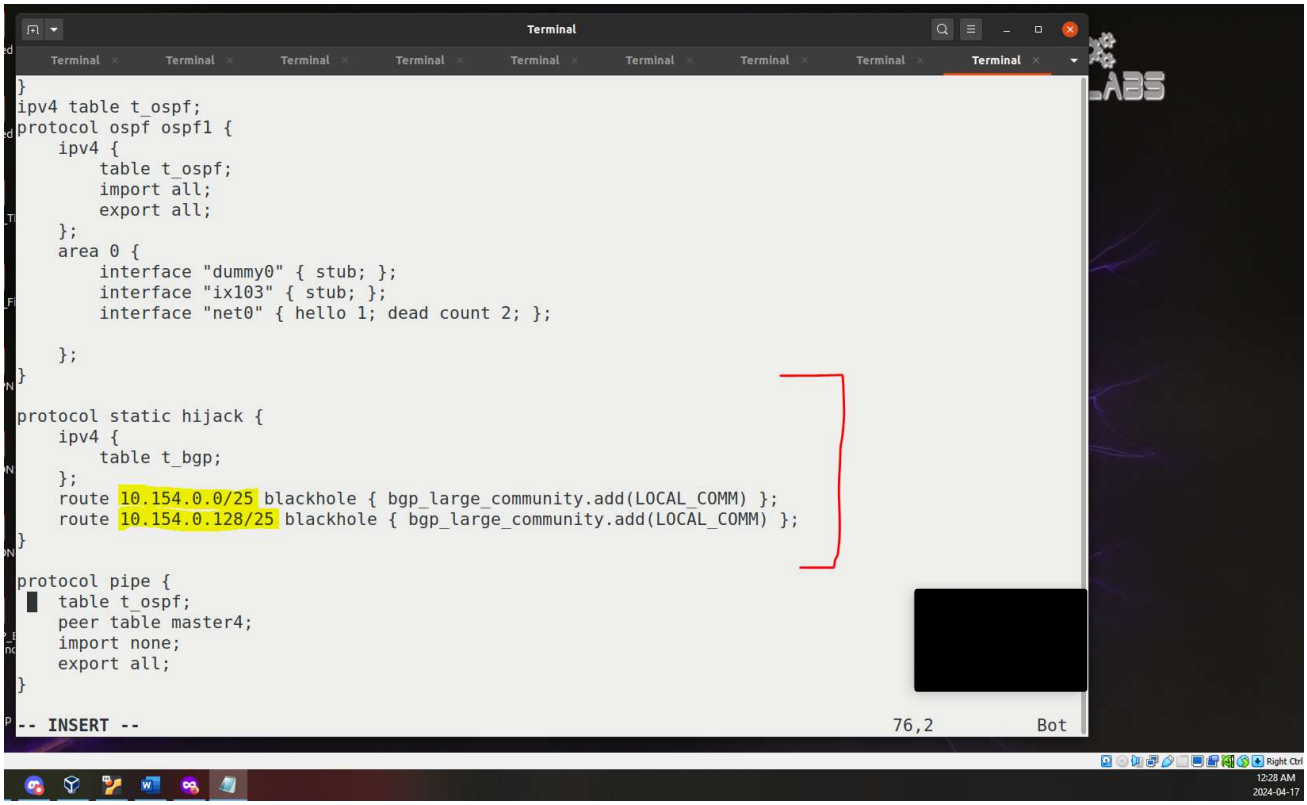


When IP anycast is used, several workstations scattered over different areas are assigned the same IP address. Using entries from the routing table, routing protocols such as BGP specify which machine is to receive incoming packets. In the scenario outlined, AS-190 oversees two disconnected networks, both connected to separate interfaces (IX-100 and IX-105) and having the same network prefix (10.190.0.0/24). 10.190.0.100 is the IP address shared by both networks. Pings from different hosts (10.171.0.71/24 and 10.153.0.72/24) to 10.190.0.100 follow different routes as a result of the routing choices made by the BGP routers. Packets with the same IP address may take different paths since each host chooses the best or shortest route to get to the destination on its own.

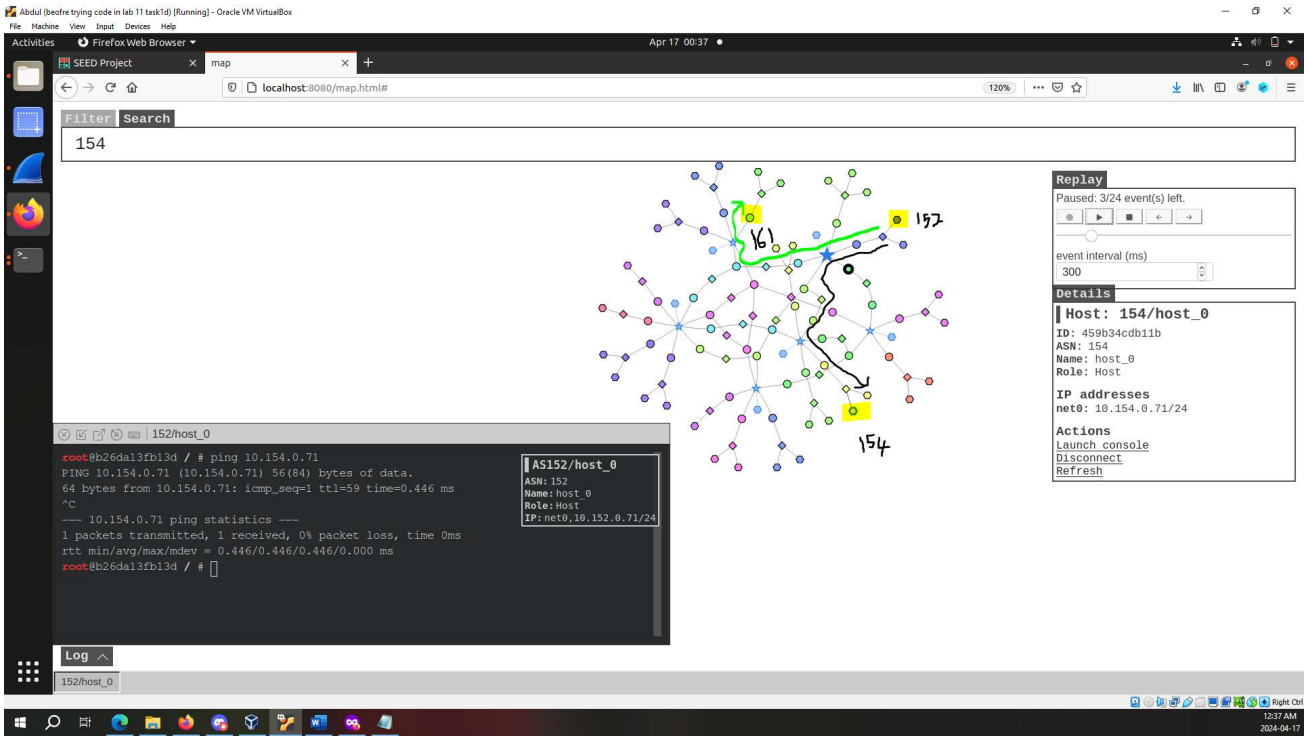
Task 5: BGP Prefix Attacks

Task 5.a: Launching the Prefix Hijacking Attack from AS-161

Here, you can see me modifying the ***bird.conf*** file in AS-161 such that AS-161 can receive traffic from AS-154.

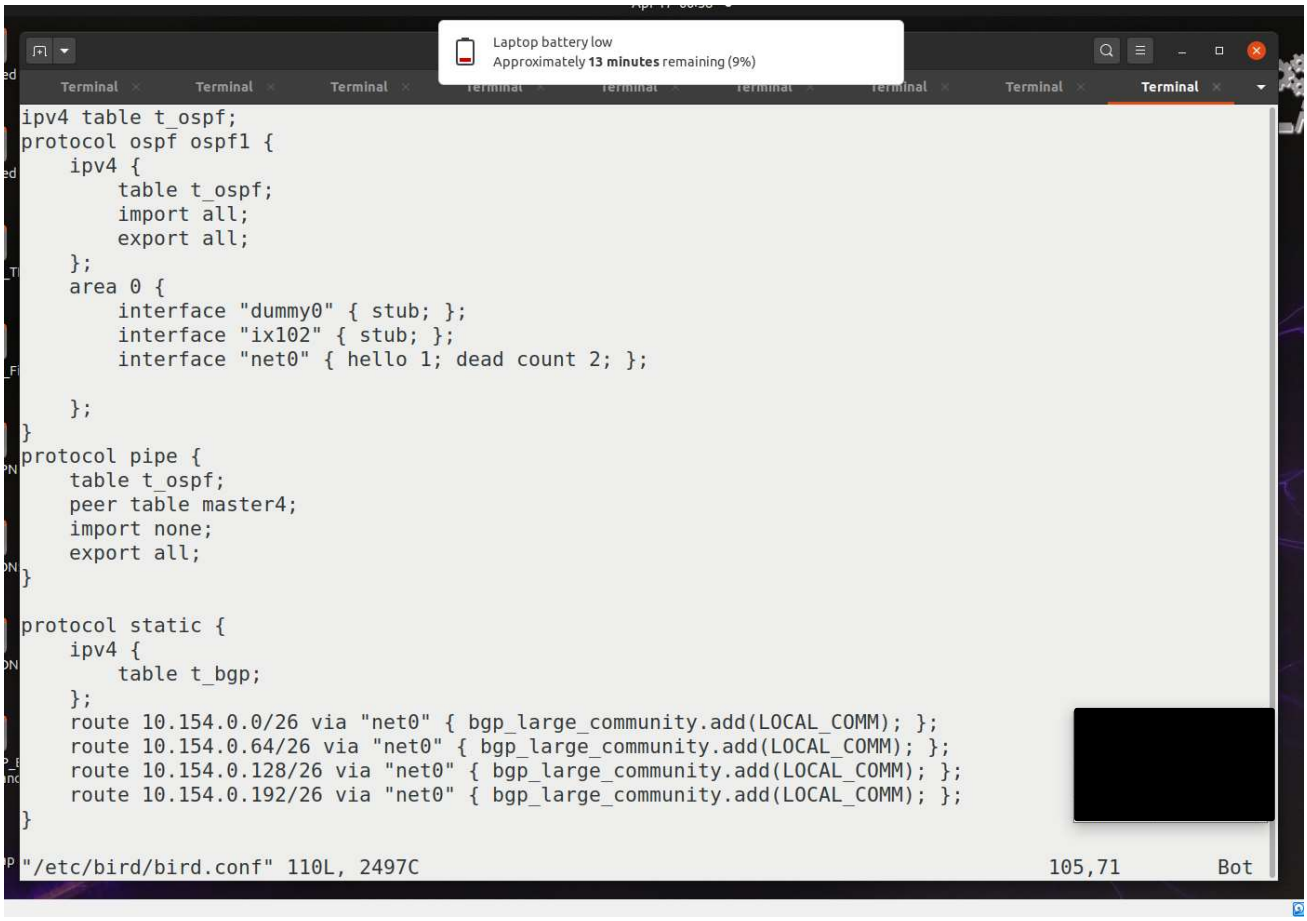


Here, you can see me attempting to ping 10.154.0.71 form host 152. The line highlighted in black shows the expected path to be taken, however the line highlighted in green is the path taken instead, towards the attacker.

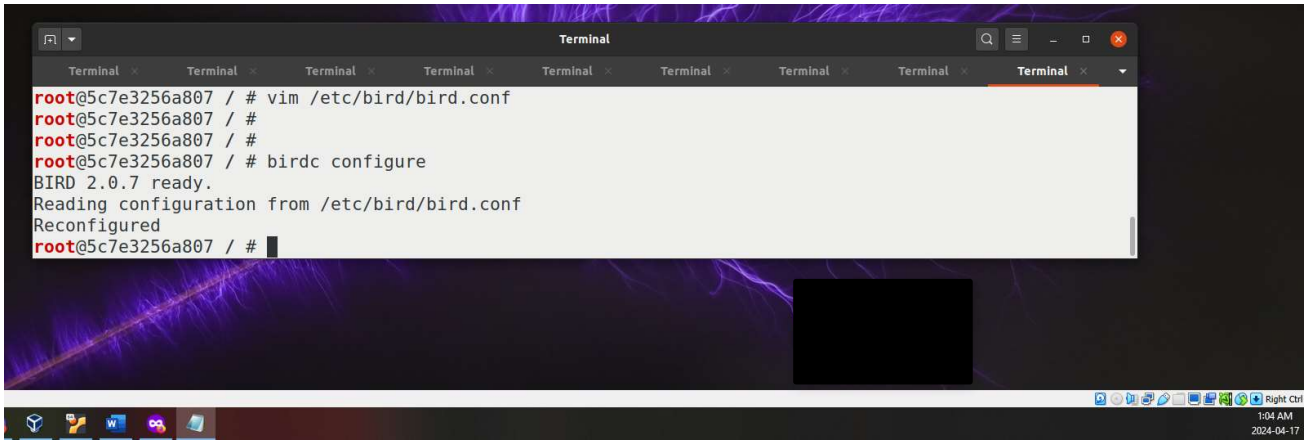


Task 5.b: Fighting back from AS-154

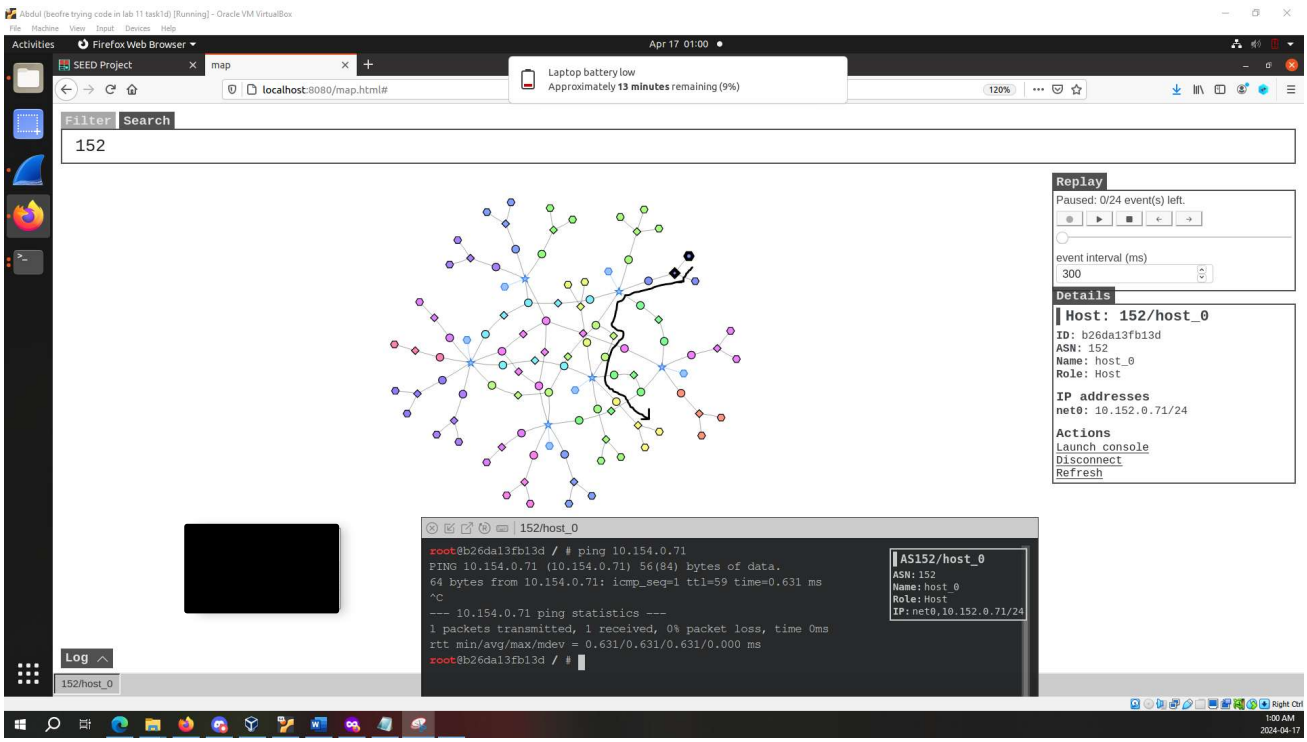
According to the book one method of fighting back is to add more prefixes. As you can see in the below screenshot, I have attempted to do so by adding a few more prefixes in the ***birds.conf*** file of AS-154.



I ensured these changes were applied by running birdc configure.

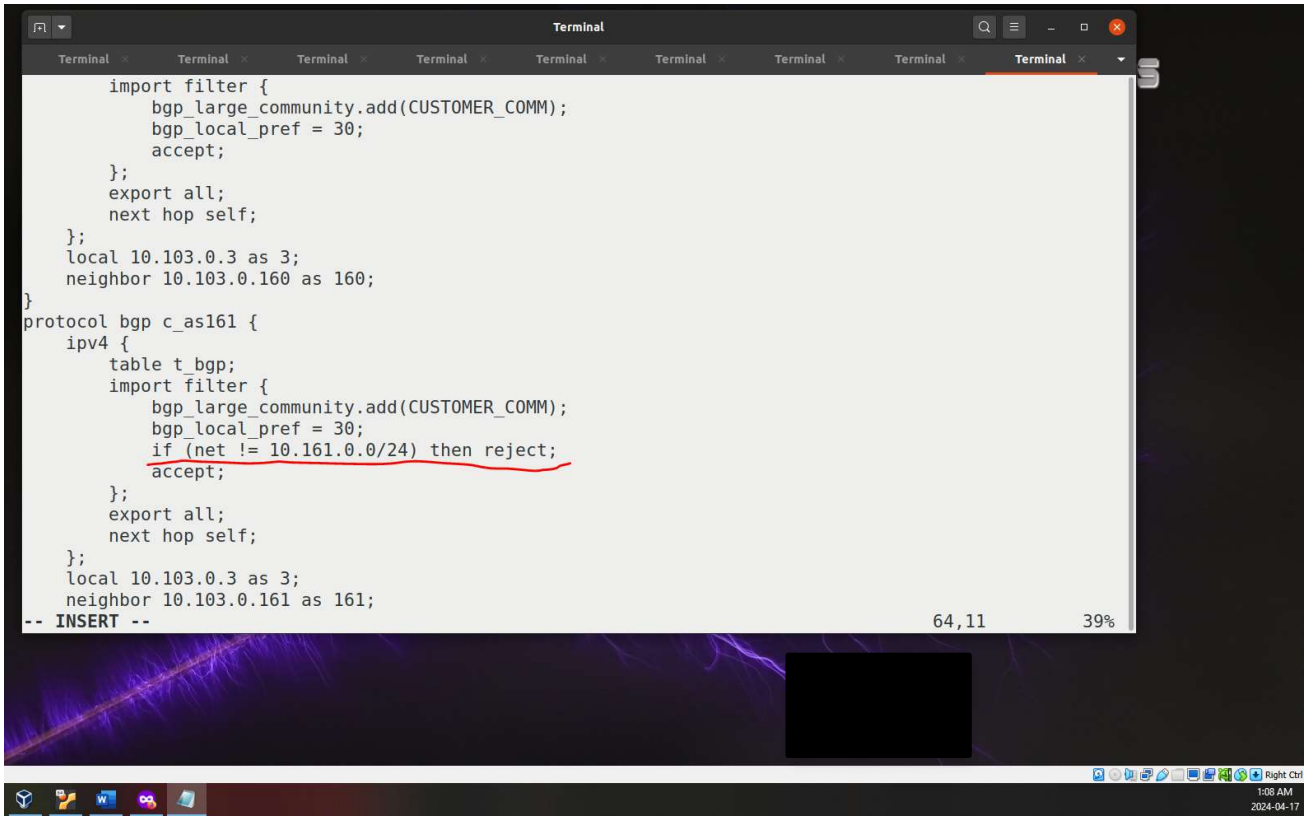


After completing the above changes, and pinging once again, we can see that this time the expected path was taken (as highlighted in black).

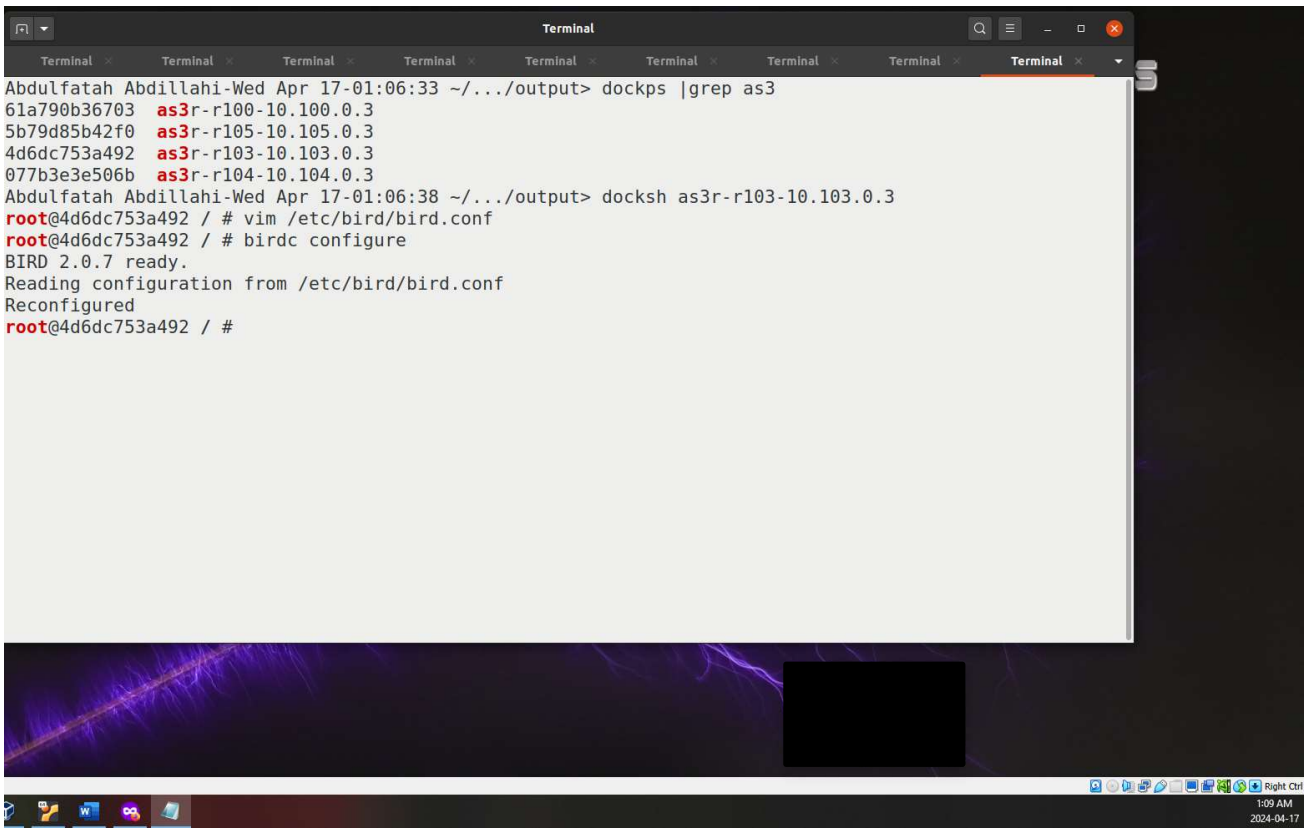


Task 5.c: Fixing the problem at AS-3

Here, you can see me modifying the configuration by adding a single filter line in the ***birds.conf*** file of AS-3. This rule checks to see if the network is different from if that is the case it is rejected. This modification is intended to inhibit the spread of fake alerts, protecting the adoption of only legitimate routes. Strict filtering like this is essential to enhancing network security and stability.



Finally, you can see that this modification was applied successfully.



References

https://seedsecuritylabs.org/Labs_20.04/Networking/BGP/BGP_Exploration_Attack/

https://seedsecuritylabs.org/Labs_20.04/Files/BGP_Exploration_Attack/BGP_Exploration_Attack.pdf

Du, W. (2022). Internet Security: A Hands-on Approach (Third edition). Independent.