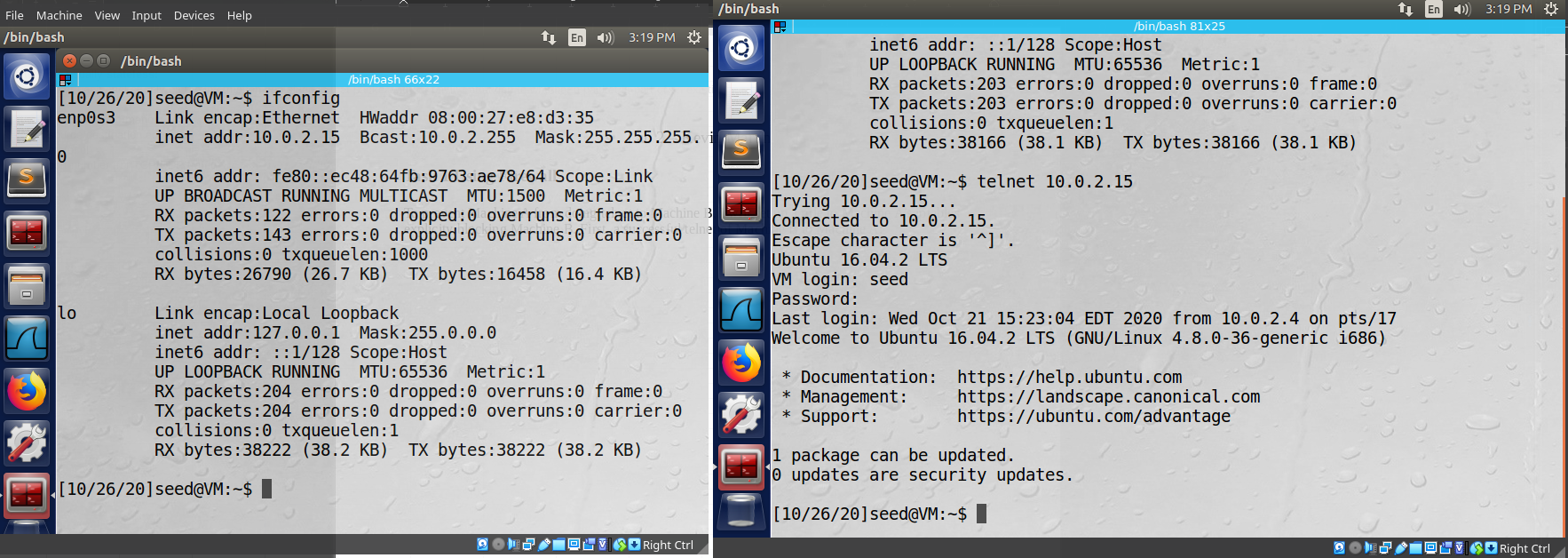
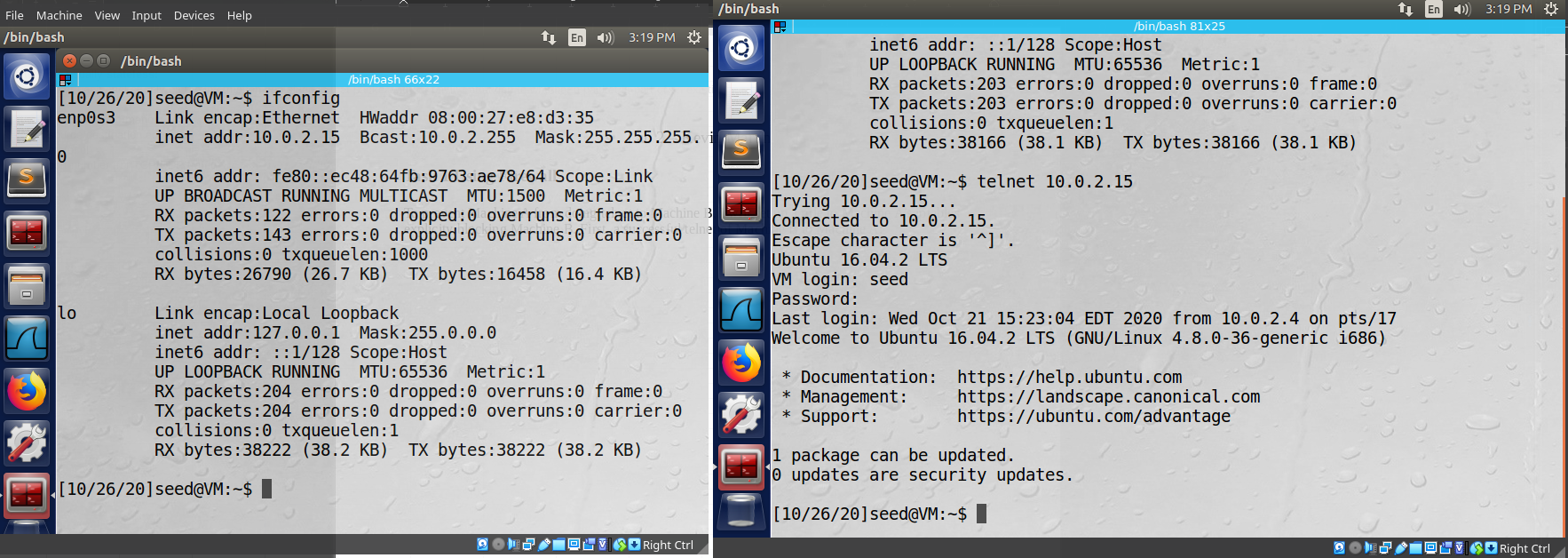
Lab 2 – Kevin Martin

**Task 1 Using Firewall**

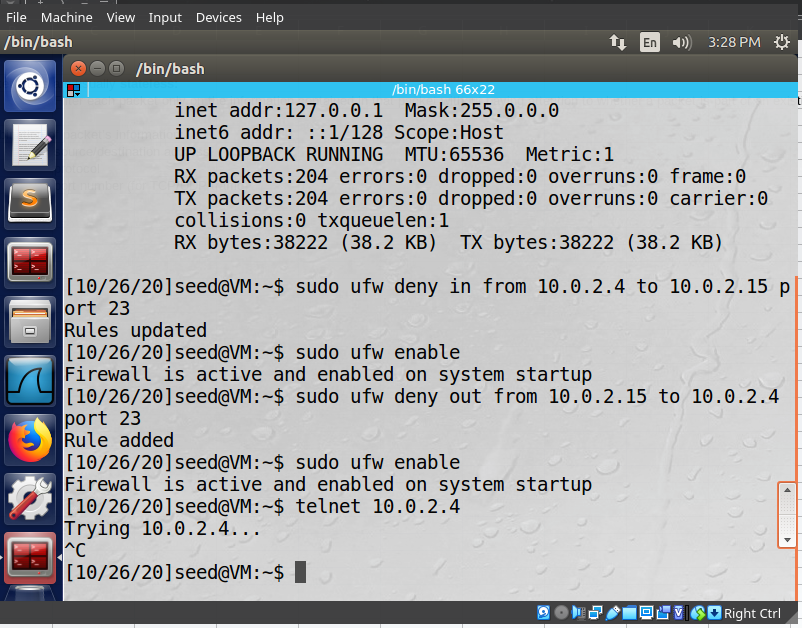
To prevent Machine A from doing telnet to Machine B, I will set up a rule to Machine A’s iptables explicity blocking Machine B. First, a successful telnet of Machine B to Machine A:



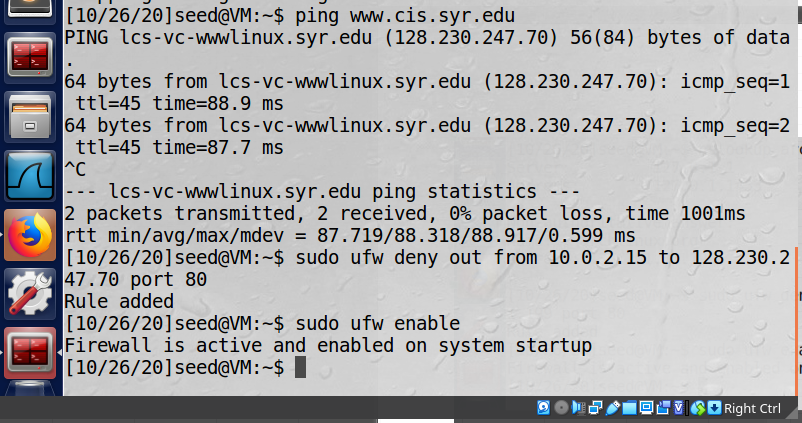
Note how Machine A is 10.2.15 (left VM), and machine B (10.0.2.4, right VM) was able to successfully telnet in. Now, the rule will be added to Machine A, and Machine B will try to establish a connection:

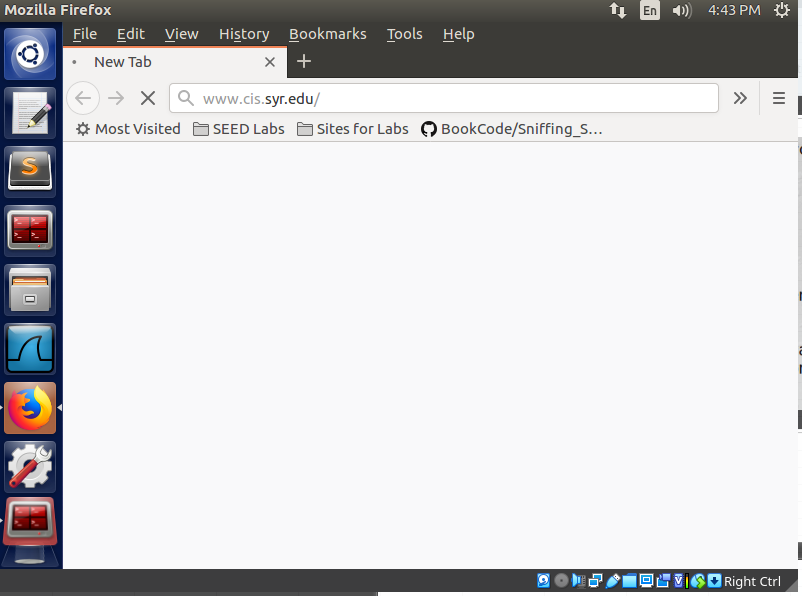


The rule from Machine A successfully prevented the same connection from Machine B. Now Machine A will prevent itself from being able to telnet into Machine B:



Finally, I will prevent A from visiting an external website (Syracuse server). First, a successful ping to the page to get the IP address, then a new rule added, and finally the successful block:

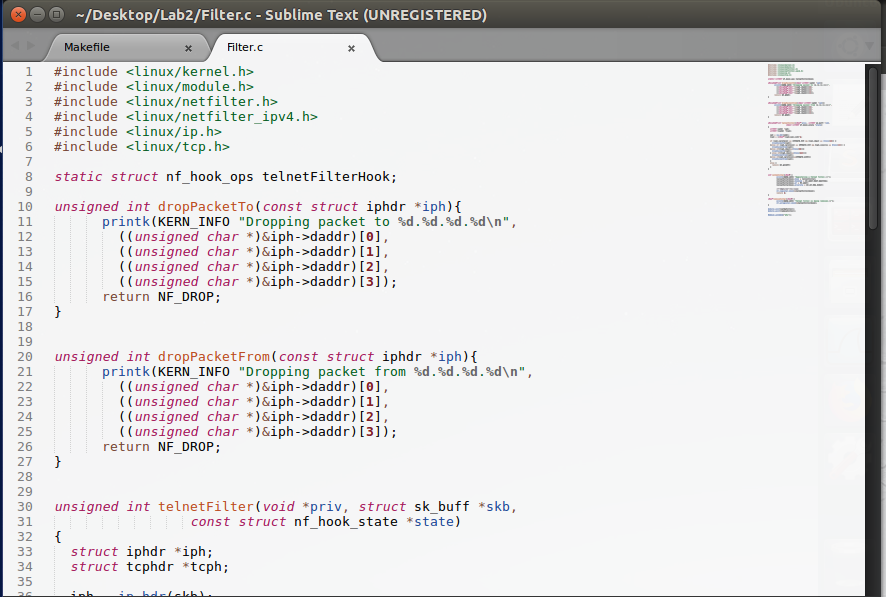


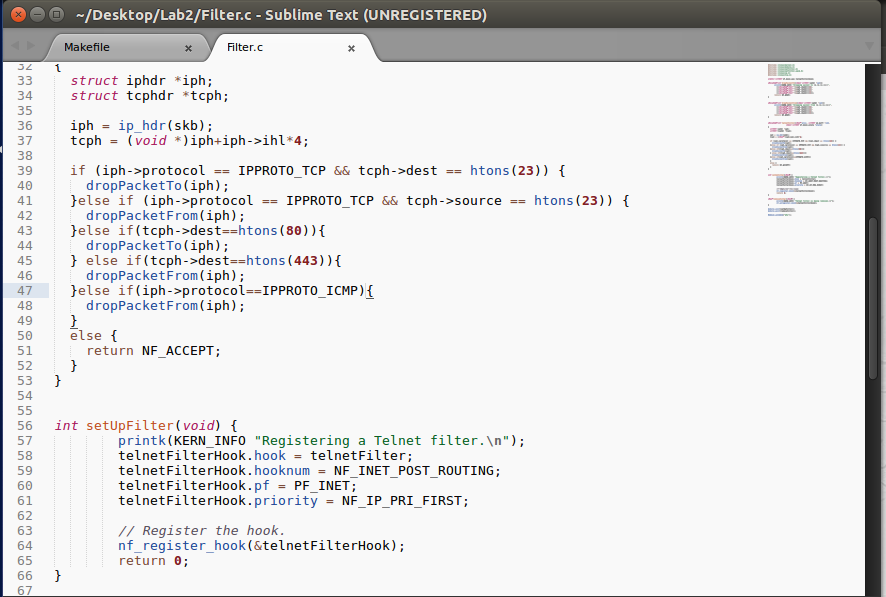


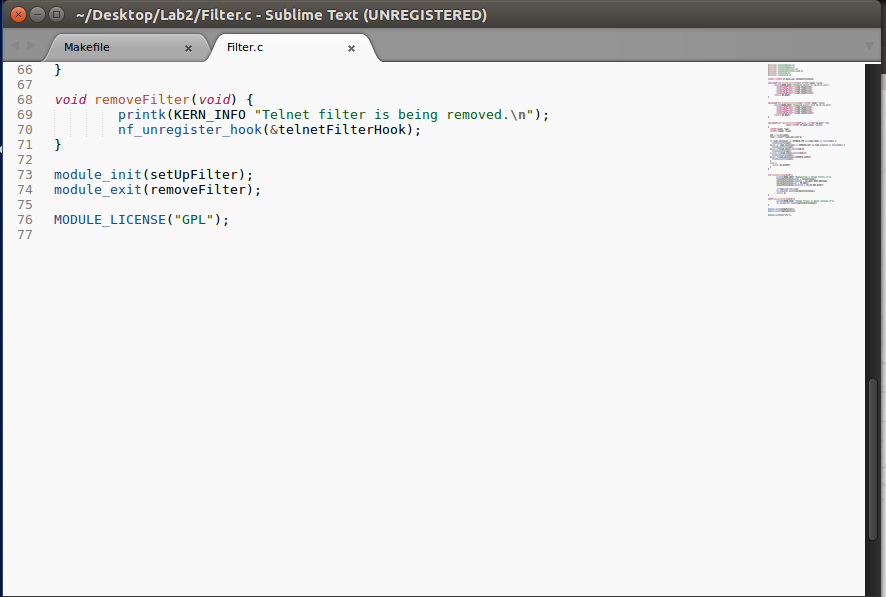
The site will never load now, it just sits here. Note that after this, I ran “sudo ufw reset” to remove the previously set firewalls. I will most likely forget they are there otherwise.

**Task 2: Implementing a Simple Firewall**

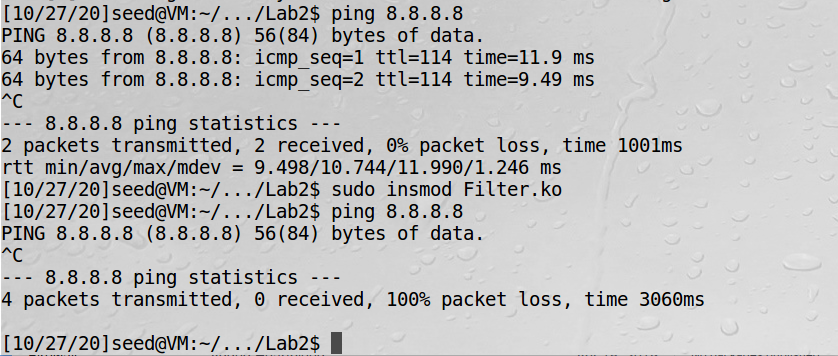
To implement a firewall, I’m using code from the book and modifying to capture more scenarios. It is compiled with “make” and I’m using the standard Makefile provided. The code:







To run, the code is invoked with “insmod” and stopped with “rmmod”. I first ping 8.8.8.8 and note no issues. Then run the code, and now the ping requests do not go through:

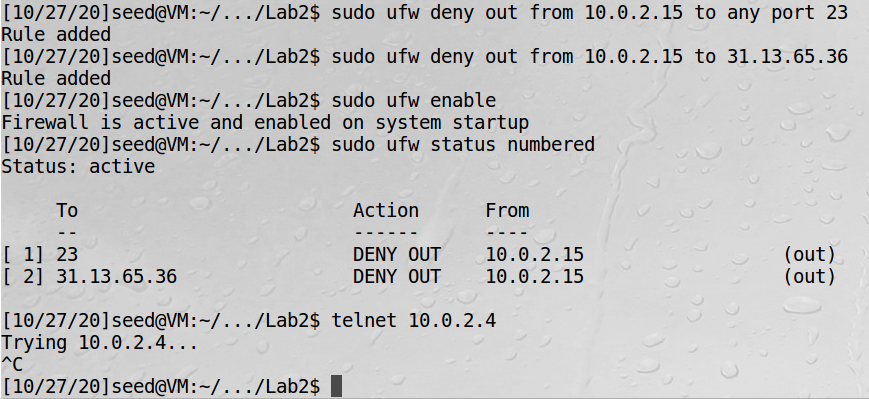


Finally, the dmesg output showing the packets being dropped. Note that after pinging 8.8.8.8 I tried to load google.com, and you can see the packets there were also dropped (the webpage timed out in the browser):

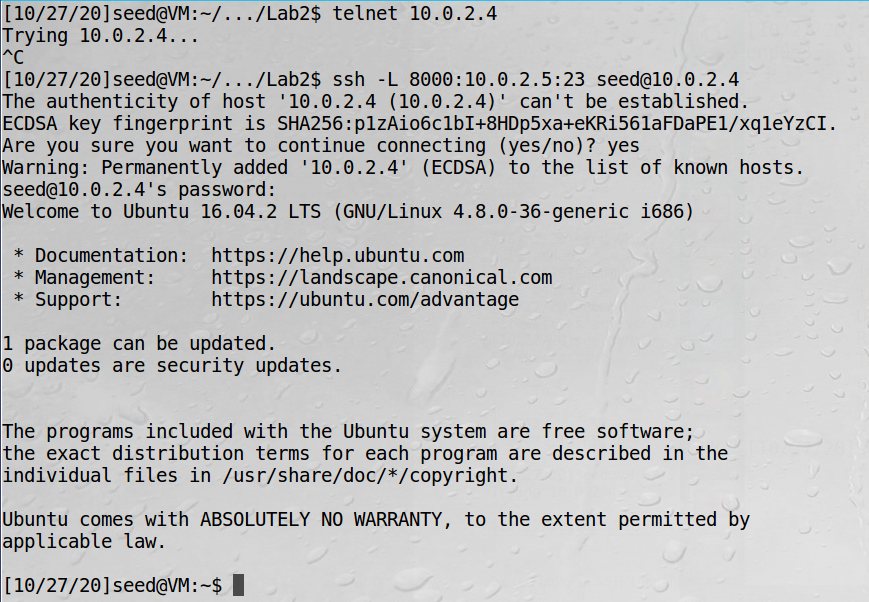


**Task 3a: Evading Egress Filtering**

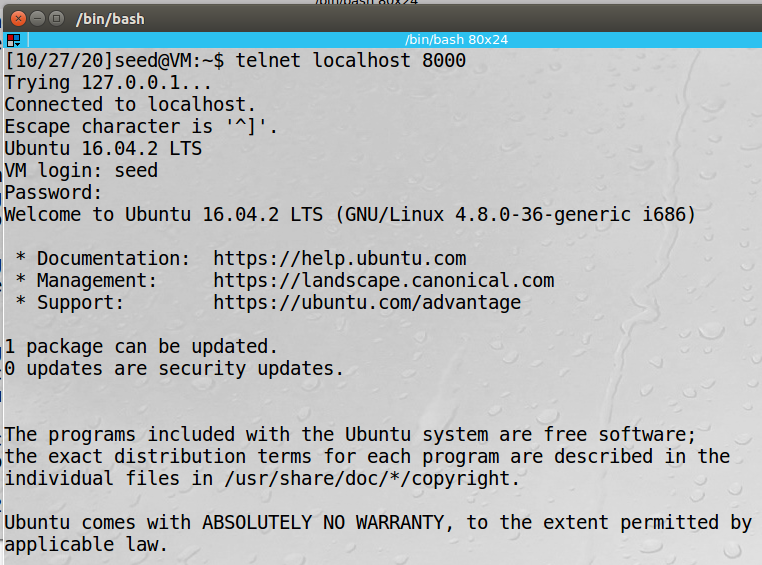
I am going to add the firewall on VM A. Note that I removed the rules at the end of Task 1. First, I reinstate them, then show that they are active (VM A is 10.0.2.15) and that VM A cannot telnet to VM B (10.0.2.4):



Next, I build an SSH Tunnel from VM A to VM C (10.0.2.5):

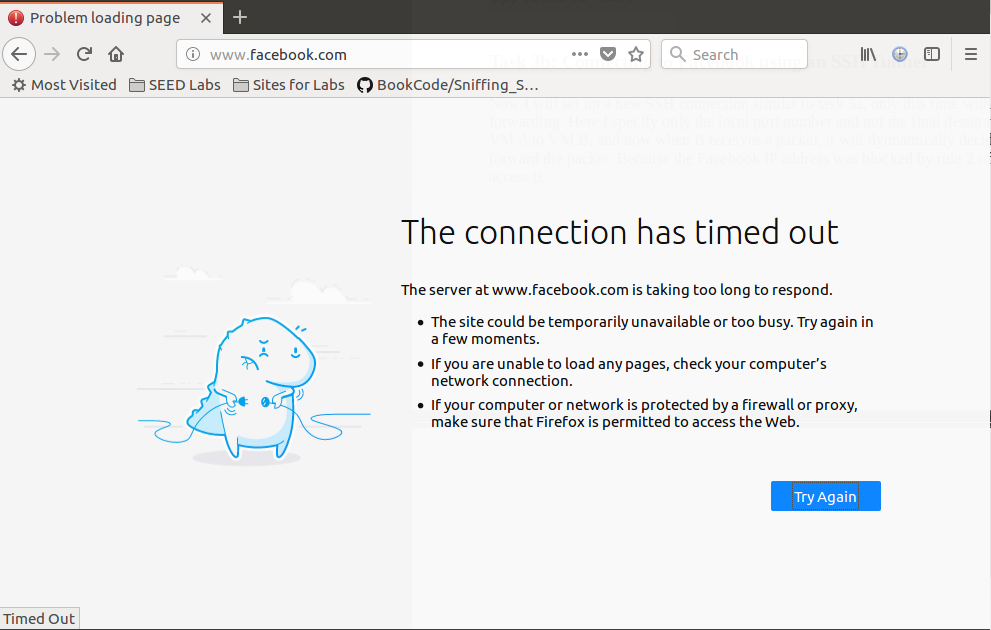


Finally, I open a second terminal window on VM A and telnet into machine B (via the localhost, provided by C):

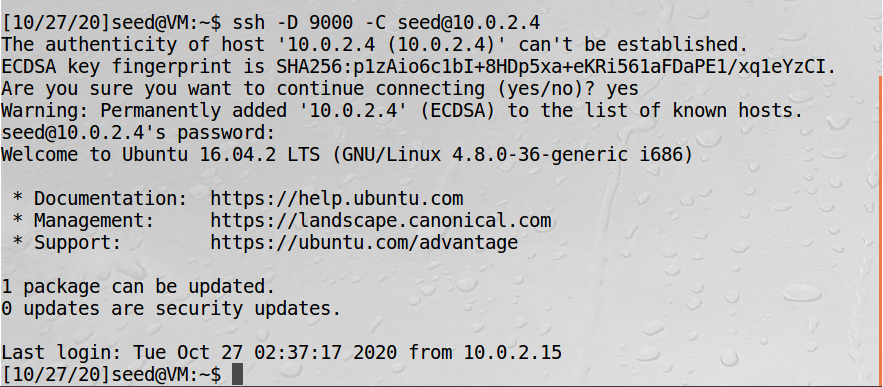


**Task 3b: Connecting to Facebook using an SSH Tunnel**

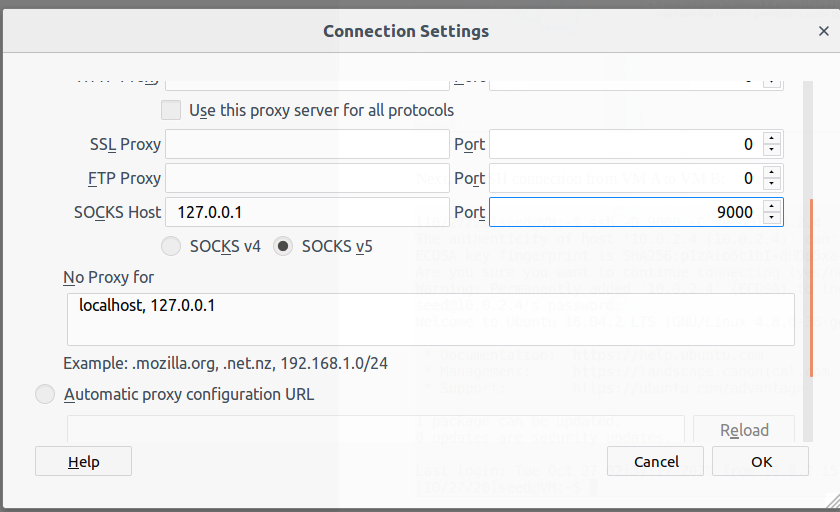
Now I will set up a new SSH connection similar to task 3a, only this time with dynmaic port forwarding. Here I specify only the local port number and not the final destination. I will SSH from VM A to VM B, and now when B receives a packet, it will dynnamically decide where it should forward the packet. Because the Facebook IP address was blocked by rule 2 of my firewall, we cannot access it:



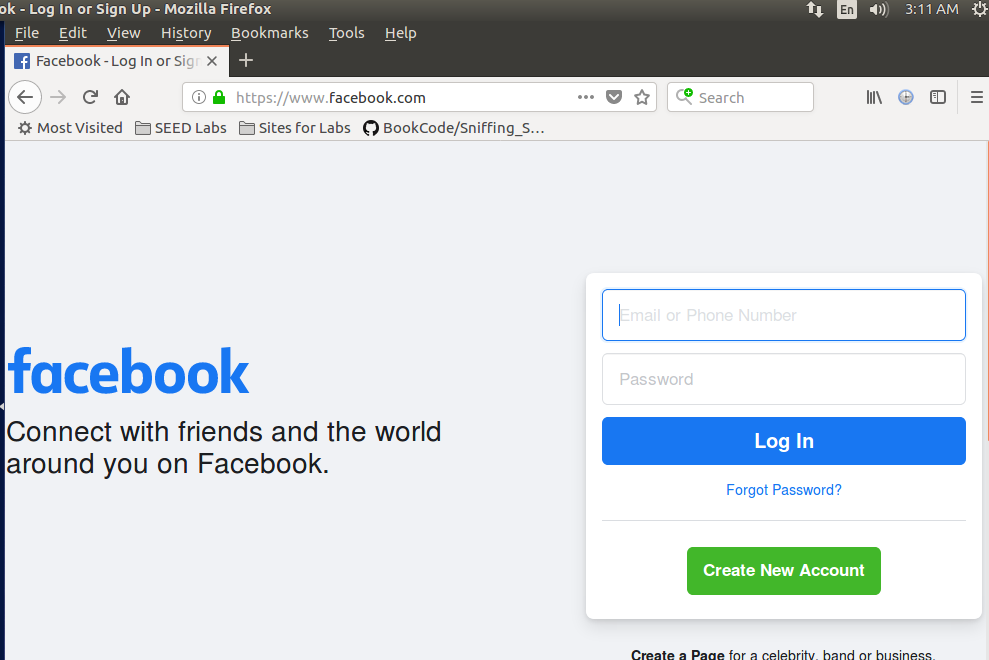
Next, the SSH connection from VM A to VM B:



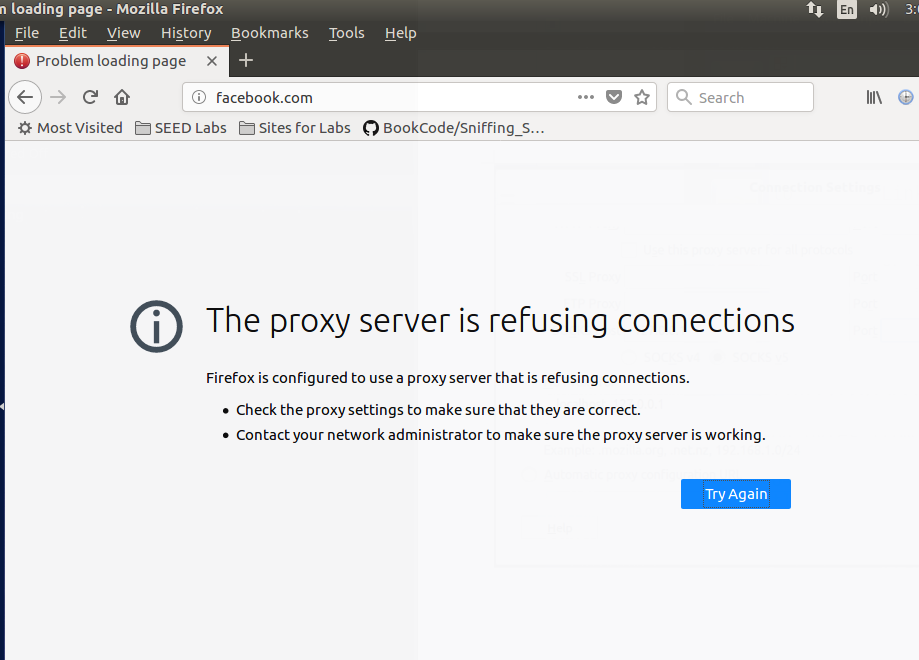
Once the connection is established, we force Firefox to connec to localhost:9000, which allows traffic from VM A to be transferred through the SSH tunnel to VM B. Firefox proxy:



After establishing the tunnel:



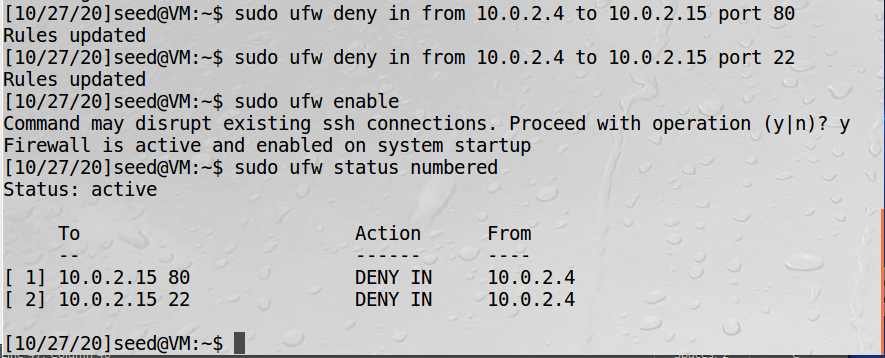
And after breaking the tunnel, we lose access again:



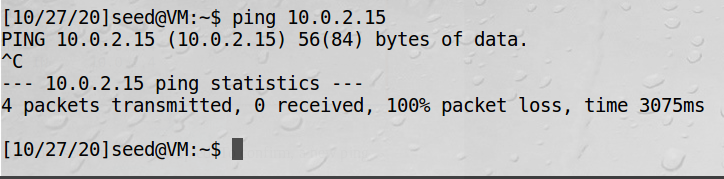
To recap, after adding the firewall to VM A, it could not longer telnet or access Facebook (as identified by a specific IP address). Creating an SSH tunnel to a third VM, which did not have a firewall restriction, allowed access to the target second VM. As for Facebook, dynamically forwarding the ports to the second VM (which again, did not have any restrictions) allowed access to the server via an SSH tunnel.

**Task 4: Evading Ingress Filtering**

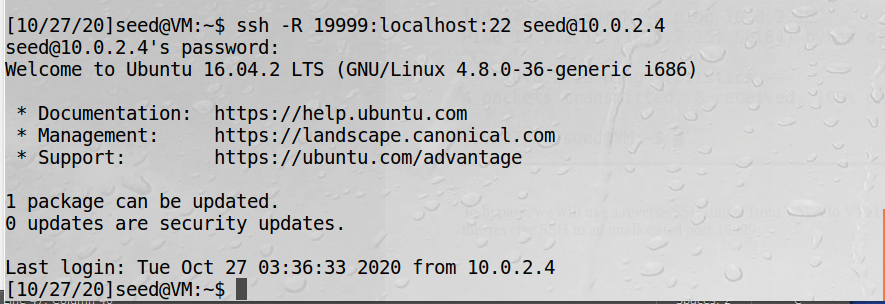
Now I will set up a firewall blocking VM B from accessing VM A through port 80 and 22. See firewall from VM A:



Note the existing SSH connection from the previous section was interrupted. To confirm, a new (failed) ping attempt from VM B to VM A:



To bypass, we will use a reverse SSH tunnel from VM A to VM B, and then try the SSH again. First, the reverse SSH to an unallocated port 19999:



And finally, from VM B, a successful SSH back into VM A using the local host and specified port:

