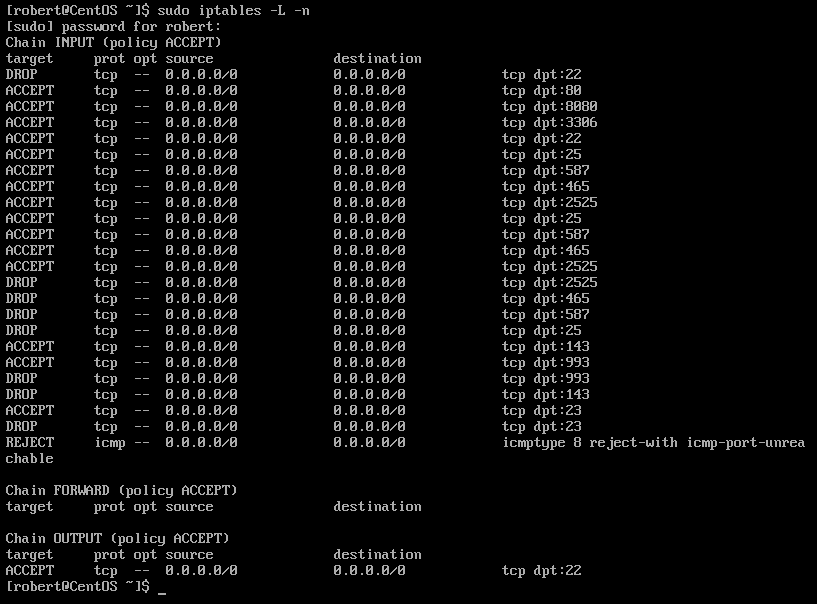
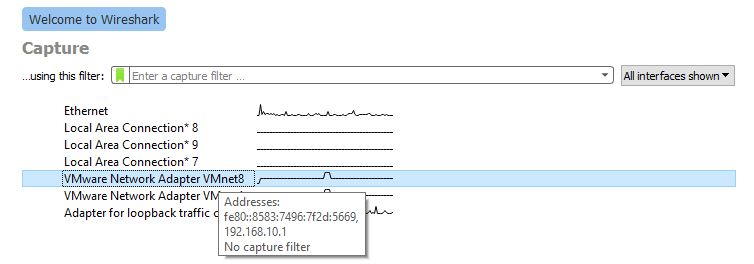
# Use Wireshark to show that your statements/scripts are working properly.

My current Firewall Rule



1. Open Wireshark in the local host

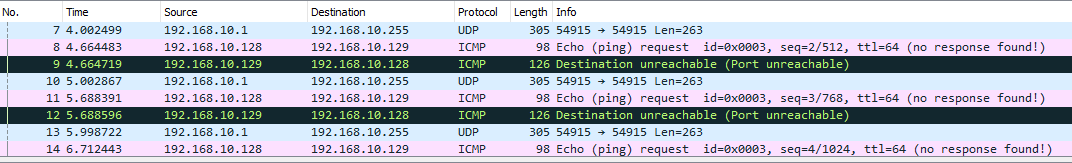
In our local host locate the capturing filter adapter where our server is running.

****The IP of my virtual server is 192.168.10.129. The VMware Network Adapter VMnet8 is capturing package from 192.168.10.1. Therefore, VMnet8 must be selected to capture the traffic of the server.

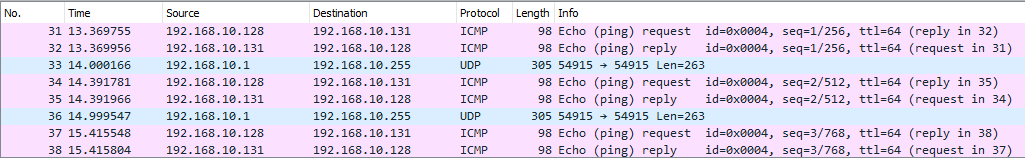
One of my Firewall rules is to disable Incoming ping request

**sudo iptables -A INPUT -p --icmp-type echo-request -j REJECT**

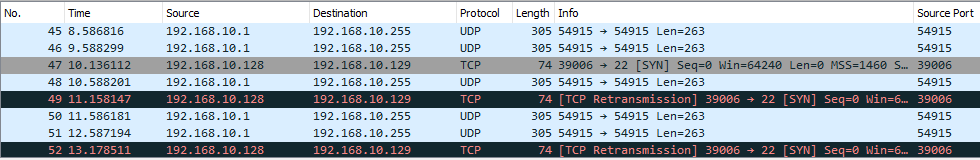
When I tried to ping my CentOS server, the destination was unreachable. In other words, I could not connect to the CentOS server. Therefore, the disable ping request is working.



192.168.10.129 It is the CentOS server’s IP

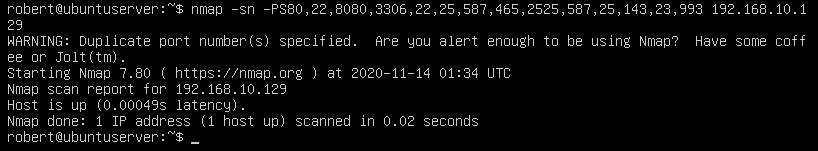
 Then I tried the 192.168.10.131 which is Ubuntu server that does not have any firewall rules and the ping request worked as expected.

I tried to send a file using SCP to the CentOS server but failed. Reason it failed because the port 22 is set to DROP in my iptables rules. Therefore, the rule is working as it should.



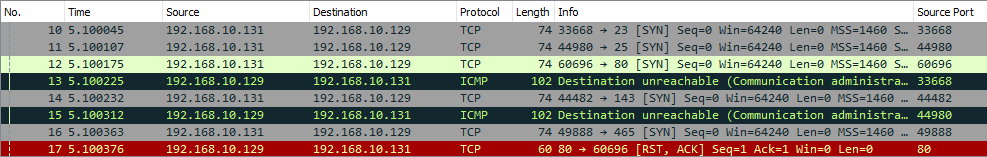
It will take a lot time to test if all my firewall rules will work. For this case I am going to use a command called **nmap**. Nmap is a network mapper that allow the user to generate packets. I will use nmap to generate packets and send it to all my ports at once.

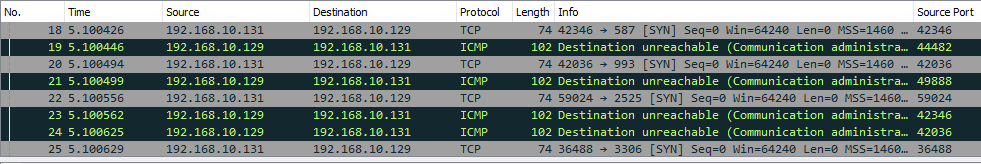
**nmap -sn -PS <target>**

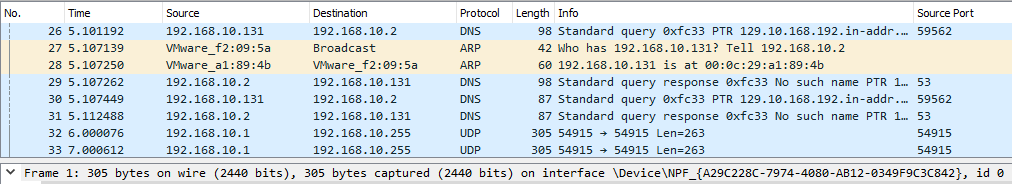


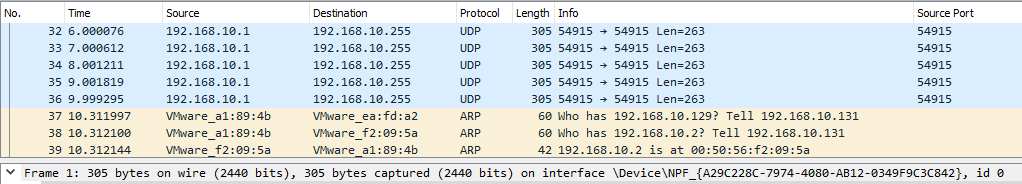
Using nmap in the other server I generated and sent packet to 19.168.10.1 which is the server with the custom firewall rules. See picture for references.

The results are the following:









If we compared my current Firewall Rules and the results from Wireshark, we could see that my statements are working properly.

# Write the specs for how you think you could prevent a DDOS attack using IPtables.  Can you write a script for this?

**#!/bin/bash**

I would drop all ICMP packets. ICMP is primarily used to ping a host to find out if it is still alive.

**iptables -t mangle -A PREROUTING -p icmp -j DROP**

I would block fragmented packets .This could mitigate UDP flooding but most of the time they use a large amount of bandwidth that depletes the capacity of the network card, thus making this rule optional.

**iptables -t mangle -A PREROUTING -f -j DROP**

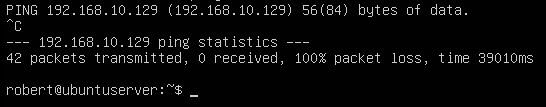
This rule is the one that allows the host to block incoming connection that exceed a certain threshold for the number of established connections. For example, if a host on the Internet establishes 90 connections against port 80 of our web server, it is surely a type of attack, and we can block it with something like this:

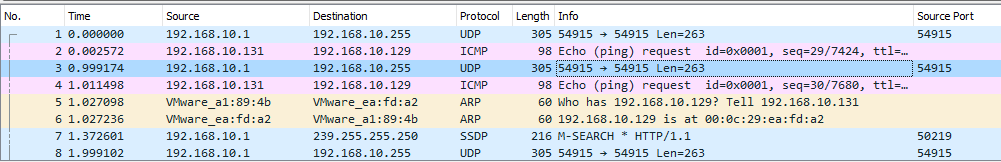
**iptables -A INPUT -p tcp -m connlimit --connlimit-above 90 -j REJECT --reject-with tcp-reset**

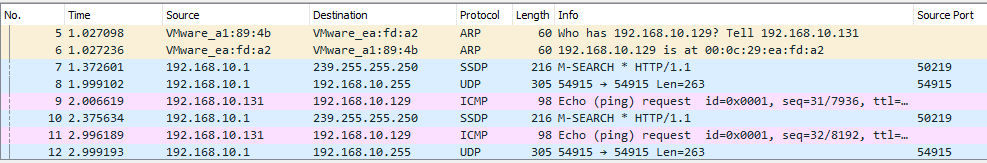
I will use the **ping of death** to test if my Firewall rules work to prevent a DDOS attack.

Ping of death : A ping of death is a type of attack on a computer system that involves sending a malformed or malicious ping to a computer. A properly formed ping packet is typically 56 bytes in size, or 64 bytes when the ICMP header is considered, and 84 including the Internet Protocol version 4 header. However, any IPv4 packet (including pings) can have a size up to 65,535 bytes. Ping of death.

Note: I will delete my existing firewall rules and write the new firewall rules that will help my server mitigate a DDOS attack. I will also monitor the server with Wireshark.







The server with the firewall rules to prevent DDOS attack was rejecting IMCP packets as well as rejecting connection that exceed the set bandwidth.