COMP 8006 – Assignment 2

Kyle Gilles & Justin Tom

A00847898 - A00852990

Table of Contents

Assignment Guidelines 3

Objective 3

Mission 3

Constraints 3

Program Summary 4

Program File Structure 4

Firewall.sh 4

FirewallConfig.sh 5

InternalHostConfig.sh 5

TestScript.sh 5

Reset.sh 5

Test Case Screen Shots 5

Test Script Packet Captures 5

Program Instructions 6

Assumptions 7

Test Cases 7

# Assignment Guidelines

## Objective

* To design, implement and test a standalone Linux firewall and packet filter.

## Mission

* Design, implement and test a firewall for Linux that will implement the following rules:
* Set the initial default policies.
* Get user specified parameters (see constraints) and create a set of rules that will implement the firewall requirements. Specifically the firewall will control:
  + Inbound/Outbound TCP packets on allowed ports.
  + Inbound/Outbound UDP packets on allowed ports.
  + Inbound/Outbound ICMP packets based on type numbers.
  + All packets that fall through to the default rule will be dropped.
  + Drop all packets destined for the firewall host from the outside.
  + Do not accept any packets with a source address from the outside matching your internal network.
  + You must ensure the you reject those connections that are coming the “wrong” way (i.e., inbound SYN packets to high ports).
  + Accept fragments.
  + Accept all TCP packets that belong to an existing connection (on allowed ports).
  + Drop all TCP packets with the SYN and FIN bit set.
  + Do not allow Telnet packets at all.
  + Block all external traffic directed to ports 32768 – 32775, 137 – 139, TCP ports 111 and 515.
  + For FTP and SSH services, set control connections to "Minimum Delay" and FTP data to "Maximum Throughput".
* Design a test procedure that will test all your firewall rules and print the results of the test to a file. Make sure that someone reading the file contents will know exactly which rule worked and which rule failed.
* The machines in the lab are equipped with two Ethernet cards. One of them is already configured and operational. You will have to enable and configure the other one for use as the gateway to your “internal” network.
* Your testbed will then have one machine operating as a firewall. It will have an “outside” connection (eth0) and it will forward datagrams to hosts on its internal hosts on the second NIC (eth1).

## Constraints

* The firewall/packet filter must be designed and implemented using Netfilter.
* Your firewall script must have two sections: a "User Configurable Section" and the "Implementation Section".
* The user configuration section will allow a user to set at least the following parameters:
  + Name and location of the utility you are using to implement the firewall.
  + Internal network address space and the network device.
  + Outside address space and the network device.
  + TCP services that will be allowed.
  + UDP services that will be allowed.
  + ICMP services that will be allowed.
* Only allow NEW and ESTABLISHED traffic to go through the firewall. In other words you are doing stateful filtering.
* You must ensure that you reject those connections that are coming the "wrong" way, meaning inbound connection requests (unless of course it is to a permitted service).
* Design test scripts to validate your firewall rules.
* You will be required to demonstrate your functional firewall in the lab on the day the assignment is due.

# Program Summary

* The bash script, when executed, will configure the host’s firewall settings through the use of the IPtables commands. The user will be able to specify the internal network address and NIC, the external network address and NIC as well as the TCP, UDP and ICMP services that will be allowed. It is a stateful filtering firewall, allowing NEW and ESTABLISHED traffic to go through the firewall. The commands will set all the default policies to DROP, allow inbound/outbound TCP, UDP and ICMP packets through the ports specified by the user, drops all packets destined for the firewall host from the outside, drops any packets with a source address from the outside matching your internal network, drops all connections coming from high ports (higher than 1023), accepts packet fragments, accepts TCP packets that belong to an existing connection (on allowed ports), drops all TCP packets with both SYN and FIN bit set, drops all Telnet packets, drops all external traffic directed to ports 32768-32775, 137-139, TCP ports 111 and 515 and sets FTP and SSH services to “Minimum Delay” as well as FTP data to “Maximum throughput”

# Program File Structure

## Firewall.sh

* This is the main file that will be executed. It contains all the IPtable commands and configurations that will design the firewall to the above specifications and restrictions. In the script, it will prompt the user for a few configuration variables such as network card names and what ports they would like open for TCP, UDP and ICMP services. There are also a few ports that are pre-defined and “default” allow. Ports 20, 21, 22, for TCP for FTP data, FTP control, SSH ports respectively. And ports 0, 3 and 8 for ICMP for regular echo-reply, port unreachable reply and echo-request ports respectively. The script sets all the default policies to DROP and runs all the DROP conditions at the top of the script since the file is read sequentially, it would be best to drop the packets that meet the specified requirements as soon as possible. Afterwards, the rest of the firewall constraints and rules are applied. For the user configuration TCP, UDP and ICMP rules, the script uses an array and a for loop to iterate through all the specified ports configured by the user and applies them to the firewall as rules. At the end of the script, the iptables is saved, restarted and then finally listed for the user to see the end result of all the rules.

## FirewallConfig.sh

* This file is required to be run before the ‘Firewall.sh’ is ran. It is to be ran on the host you wish to act as the firewall. It prompts the user for a few configuration variables such as internal firewall and external host IP addresses and the two network card interface names. It uses the user inputted information to essentially set up and run any pre-required configuration to the routing and IPtables for forward traffic to its newly created subnet network and vice versa.

## InternalHostConfig.sh

* This file is also required to be run before the ‘Firewall.sh’ is ran. It is to be ran on the host you wish to act as the internal host in the subnet. It prompts the user for a few configurations such as internal host and firewall IP addresses as well as the two network interface cards to help set-up and configure the internal host. It essentially disables the network interface card connected to the Internet and changes the second network interface card to connect to the firewall and adds the default gateway to point to the firewall.

## TestScript.sh

* This file is used to test the rules of the firewall based on the previously outlined constraints and firewall requirements. It also requests for a few configuration parameters of external firewall IP address, the internal (subnet) address of the internal host and the subnet of the network the firewall had created in order to add the proper routing rule so the host knows what gateway to connect to when looking for the specified subnet address of your internal host. It will log all the output (while keeping the standard output to the terminal) and save to a text file under the name of ‘Firewall\_Test\_log\_’ then the current time (Hour.Minutes-Month-Day-Year) in the same directory.

## Reset.sh

* This file will flush any existing rules and clear all user-defined chains from the IPtables, NAT tables and mangle tables as well as setting the default policies for Input, Output and Forward as accept.

## Test Case Screen Shots

* Directory that contains all the screenshots that were used to as figures in the test cases for each machine (firewall, internal, external)

## Test Script Packet Captures

* Directory that contains three wireshark capture files that were recorded when running the test script file - each one for the different machine (Firewall, Internal and external)

# Program Instructions

* Navigate through the directories to the folder with all the files in it.
* May have to change permissions on the files
  + chmod 777 \*
* If the user wants the firewall to allow the internal host to be able to access websites, the /etc/resolv.conf files on both firewall and internal hosts must have matching nameservers.
  + During the firewall setup, you must also allow DNS, DHCP, HTTP and HTTPS
    - UDP 53, 67, 68
    - TCP 53, 80, 443
  + nameserver 8.8.8.8 required to be added in both resolv.conf files.
* On the Internal Host execute:

./InternalHostConfig.sh

* On the standalone firewall execute:

./Reset.sh

./FirewallConfig.sh

./Firewall.sh

* On all three machines involved, enter the command: Route -n
* Ensure the results resemble the following: (If not, User Route add /del and match to below).

Internal host:

|  |  |  |  |
| --- | --- | --- | --- |
| **Destination** | **Gateway** | **Genmask** | **Interface** |
| default | 192.168.10.1 | 0.0.0.0 | p3p1 |
| 192.168.10.0 | 0.0.0.0 | 255.255.255.0 | p3p1 |

Firewall:

|  |  |  |  |
| --- | --- | --- | --- |
| **Destination** | **Gateway** | **Genmask** | **Interface** |
| 0.0.0.0 | 192.168.0.100 | 0.0.0.0 | em1 |
| 192.168.0.0 | 192.168.0.5 | 255.255.255.0 | em1 |
| 192.168.10.0 | 192.168.10.0 | 255.255.255.0 | p3p1 |

External host:

|  |  |  |  |
| --- | --- | --- | --- |
| **Destination** | **Gateway** | **Genmask** | **Interface** |
| 0.0.0.0 | 192.168.0.100 | 0.0.0.0 | em1 |
| 192.168.0.0 | 0.0.0.0 | 255.255.255.0 | em1 |
| 192.168.10.0 | 192.168.0.5 | 255.255.255.0 | em1 |

* Finally, onn the external host execute:

./TestScript.sh

## Assumptions

* No errors are made when inputting values in prompt
  + No error checking
* User knows the pre-existing/default port rules and doesn’t add duplicates for TCP (20, 21, 22) and ICMP (0, 3, 8) port rules.
  + No handling for duplicates.

# 

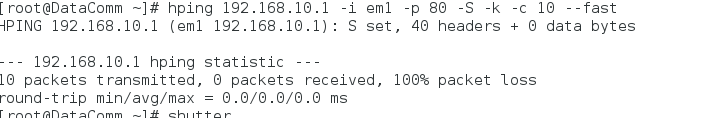
# 

# Test Cases

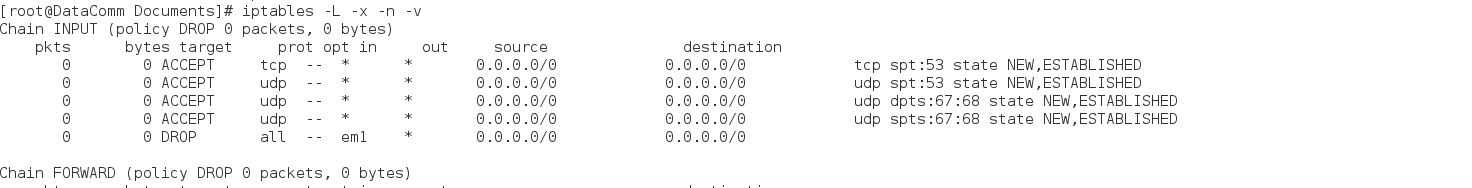
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Test Description | Tool Used | Expected Result | Pass/Fail |
| 1 | Drop all packets destined for the firewall from the outside. | hping3  Wireshark | The firewall should DROP the packet. | Pass |
| 2 | Allow Inbound / Outbound TCP Packets on allowed Ports. | hping3  Wireshark | The firewall should ACCEPT the packets. | Pass. |
| 3 | Allow Inbound / Outbound UDP Packets on allowed ports | hping3  Wireshark | The firewall should ACCEPT the packets. | Pass |
| 4 | Allow Inbound / Outbound ICMP Packets on allowed ports. | hping3  Wireshark | The firewall should ACCEPT the packets. | Pass |
| 5 | Do not accept any packets with a source address from the outside matching your internal network | hping3  Wireshark | The firewall should DROP the packets. | Pass |
| 6 | Reject those connections that are coming the 'wrong' way (incoming SYN to high ports) | hping3  Wireshark | The firewall should DROP the packets. | Pass |
| 7 | Accept fragmented packets | hping3  Wireshark | The firewall should ACCEPT the packets. | Pass |
| 8 | Drop all TCP packets with the SYN and FIN bit set | hping3  Wireshark | The firewall should DROP the packets. | Pass |
| 9 | Do not allow Telnet packets at all | telnet  Wireshark | The firewall should DROP the packets. | Pass |
| 10 | Block all external traffic directed to ports 32768-32775 | hping3  Wireshark | The firewall should DROP the packets. | Pass |
| 11 | Block all external traffic directed to ports 137-139 | hping3  Wireshark | The firewall should DROP the packets | Pass |
| 12 | Block all external traffic directed to TCP on port 111 | hping3  Wireshark | The firewall should DROP the packets | Pass |
| 13 | Block all external traffic directed to TCP on port 515 | hping3  Wireshark | The firewall should DROP the packets | Pass |
| 14 | Ensure FTP and SSH services have “Minimum Delay” (Requires a new DSCP column in wireshark. Value = ‘ip.dsfield.dscp’) | hping3  Wireshark | The firewall should ACCEPT the packets with the DSCP value of ‘4’ in Wireshark.  Source: <https://www.google.ca/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0CAQQjBw&url=https%3A%2F%2Fblogs.manageengine.com%2Fimage%2F501000001070197%2FQoS4.png&ei=SGnZVL-CMYbwoAS744DoBw&bvm=bv.85464276,d.cGU&psig=AFQjCNH9bzW-pFb10jyUrIdCl3yGy5nopg&ust=1423620747126691> | Pass |
| 15 | Ensure FTP Data service has “Maximum Throughput”  (Requires a new DSCP column in wireshark. Value = ‘ip.dsfield.dscp’) | hping3  Wireshark | The firewall should ACCEPT the packets and show DSCP values of ‘2’ in Wireshark.  Source: <https://www.google.ca/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0CAQQjBw&url=https%3A%2F%2Fblogs.manageengine.com%2Fimage%2F501000001070197%2FQoS4.png&ei=SGnZVL-CMYbwoAS744DoBw&bvm=bv.85464276,d.cGU&psig=AFQjCNH9bzW-pFb10jyUrIdCl3yGy5nopg&ust=1423620747126691> | Pass |

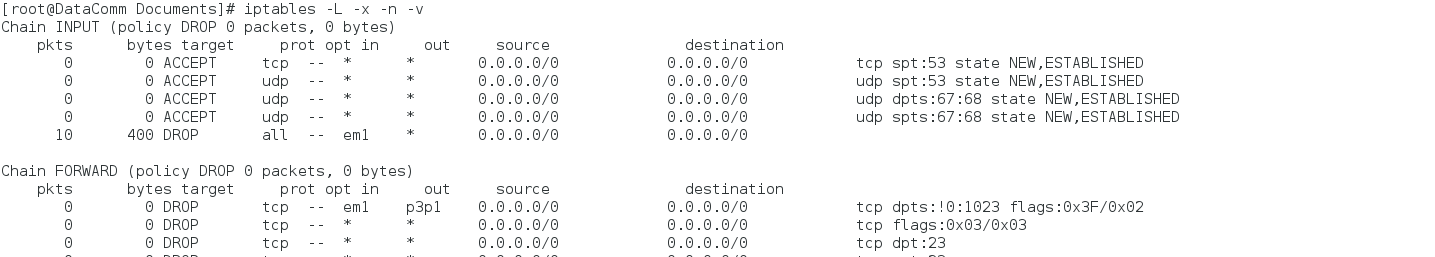
Test Case 1 Drop all packets destined for the firewall from the outside.

We entered the following command.

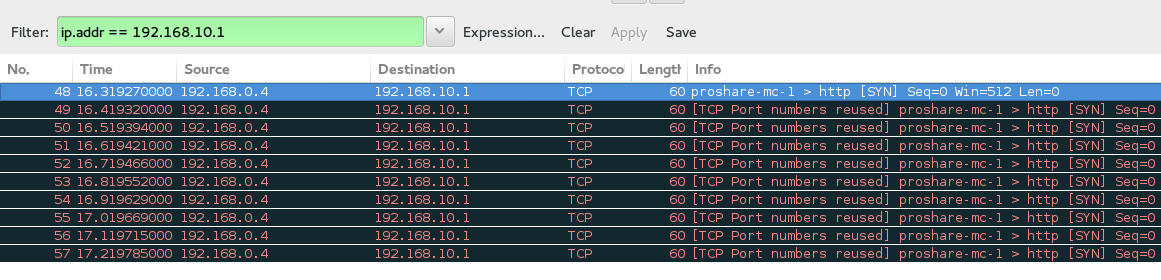


The IP Tables before we entered the command



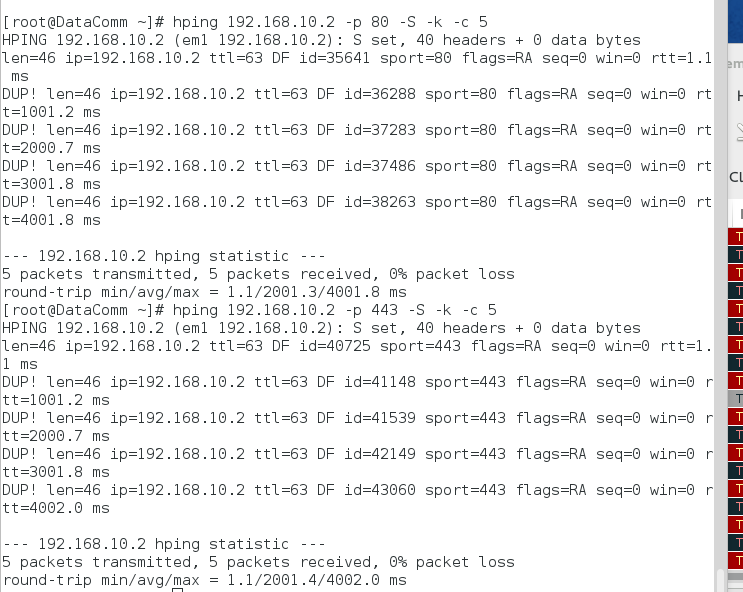
IP tables after we entered the command. We can see the Firewall dropping the 10 packets at the rule. 

Wireshark Capture of the Firewall dropping the packets.

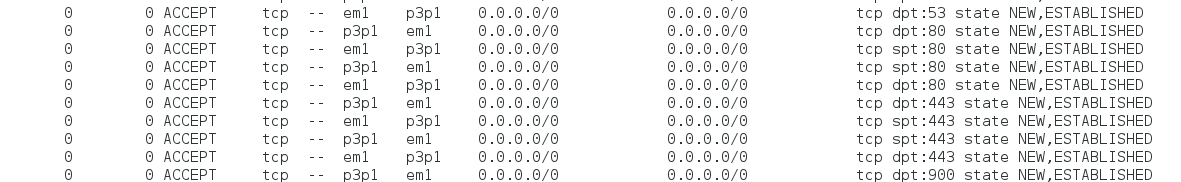


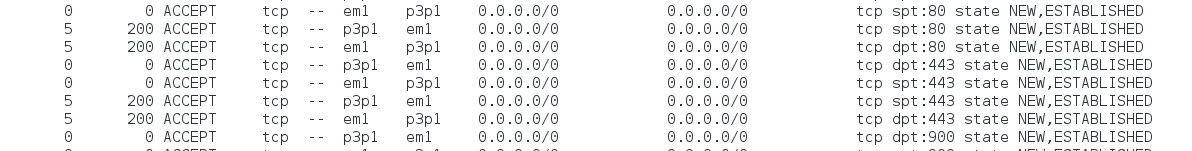
Test Case 2: Allow inbound / outbound TCP packets on allowed ports.

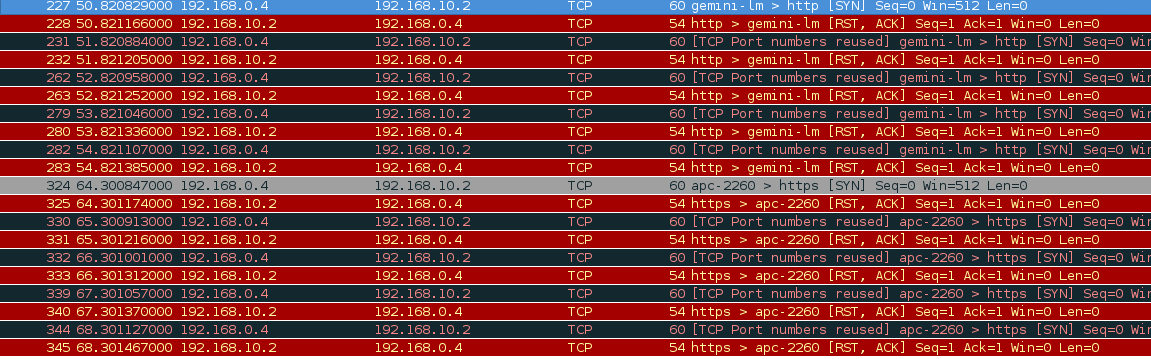
Upon initiating the firewall the user is prompted to allow traffic on specified ports. We opened port 80 and 443 for this test and entered the following commands.



IPtables before the hping.

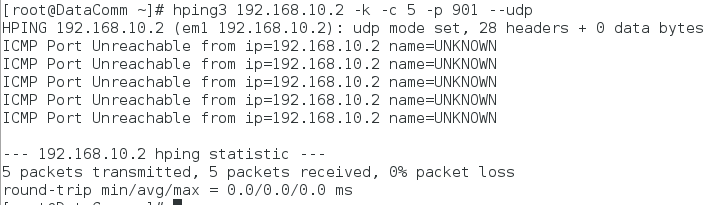


IPtables after the hping. We see the 5 packets travelling both to and from the host, on both port 80 and 443.

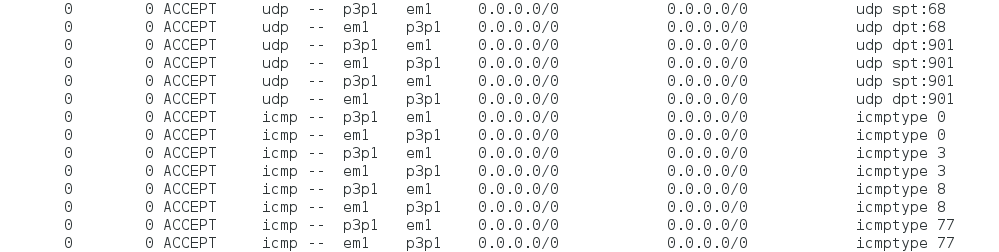
Wireshark capturing the accepted packets. 

Test Case 3: Allow Inbound / Outbound UDP Packets on allowed ports

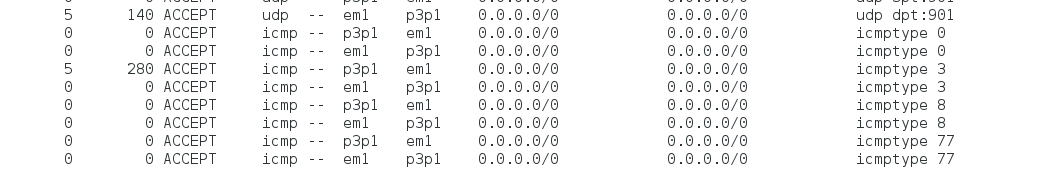
To test, we opened port 901 for UDP outgoing packets. We saw in Wireshark that “Port Unreachable” packets were being created on the internal host and sent back as ACKS, so we made a rule to accept ICMP type 3 packets to accept these.



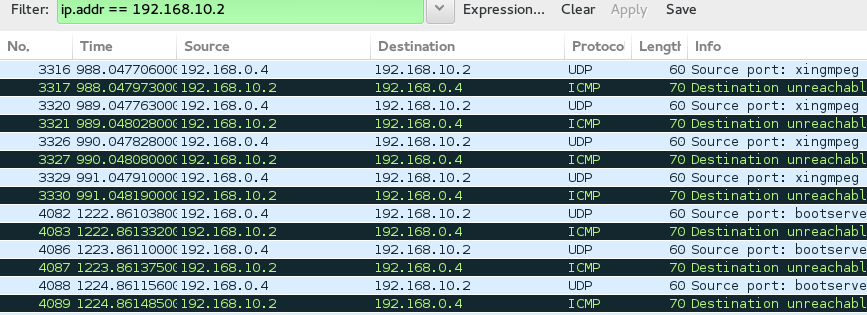
IPtables before the command.



IPtables after the command. We see the accepted UDP packets and ICMP packets (our UDP ACKS).

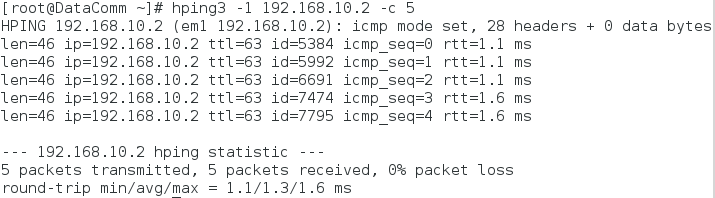


Wireshark capture at the firewall of above. .



Test Case 4: Allow Inbound / Outbound ICMP Packets on allowed ports.

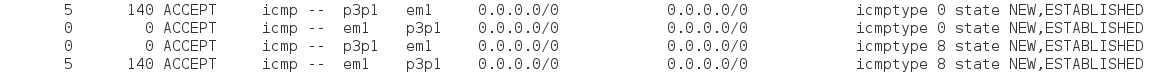
We allowed ICMP Type 0 and 8 for the test and entered the following command from the external host.



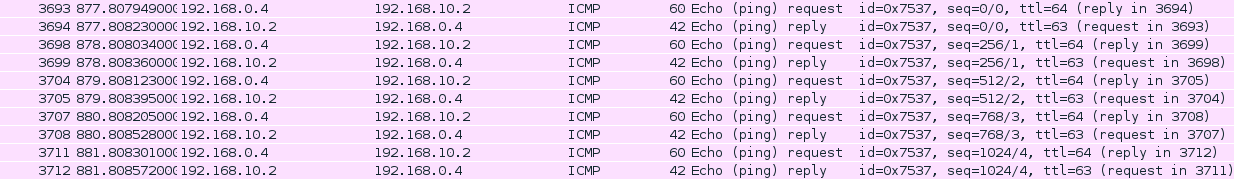
IPtables before entering the command.



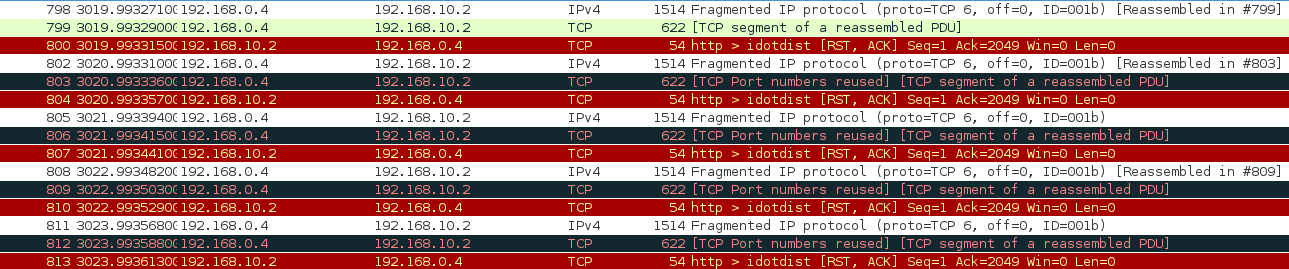
IPtables after entering the command. We can see the accepted ICMP type 0 and type 8 packets.



Firewall Wireshark capture of above.

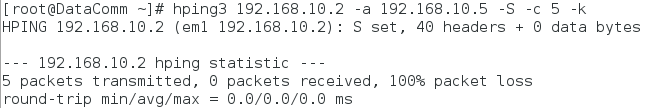


Internal Host Wireshark capture of above.

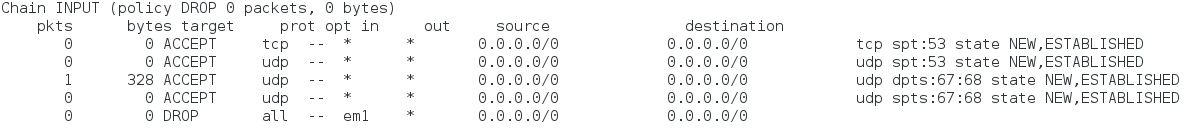


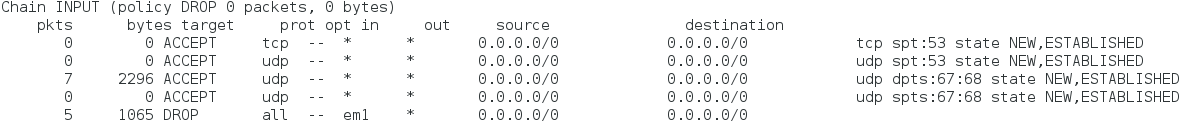
Test Case 5: Do not accept any packets with a source address from the outside matching your internal network.

To test, we sent the command below.

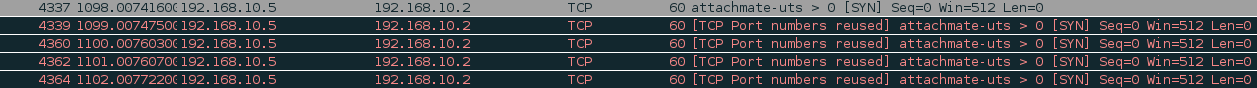


IPtables before entering the command.



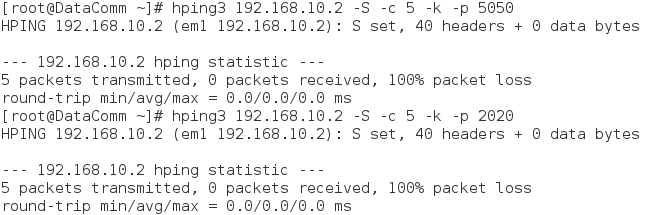
IPtables after entering the command.

Wireshark capturing the dropped packets because the source IP matched the internal network.



Test Case 6: Reject those connections that are coming the 'wrong' way (incoming SYN to high ports).

To test. we entered the below commands:



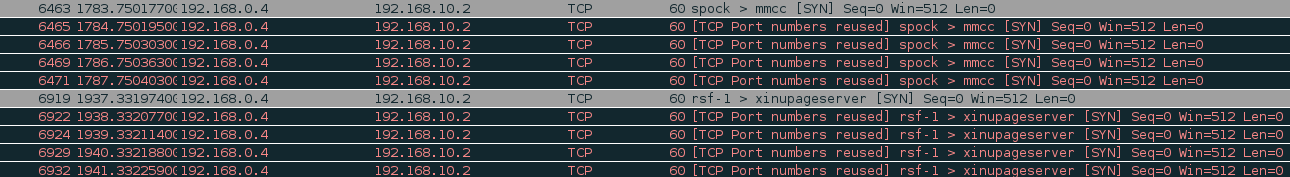
IPtables before the command



IPtables after the command. We see our rule dropping the 10 packets.

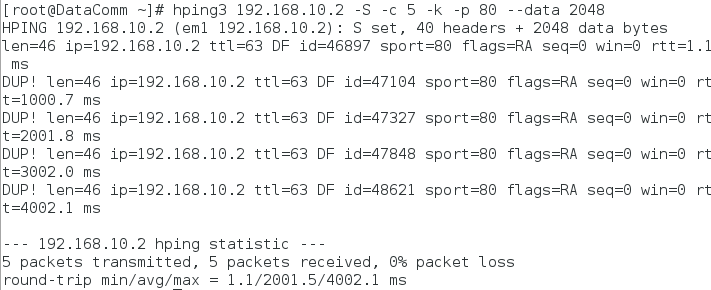


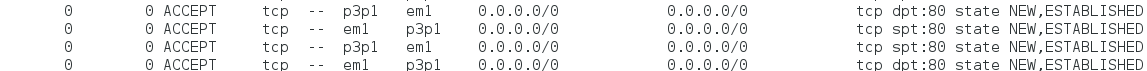
Firewall Wireshark capture of above.



Test Case 7: Accept fragmented packets

We sent fragments using the following command:

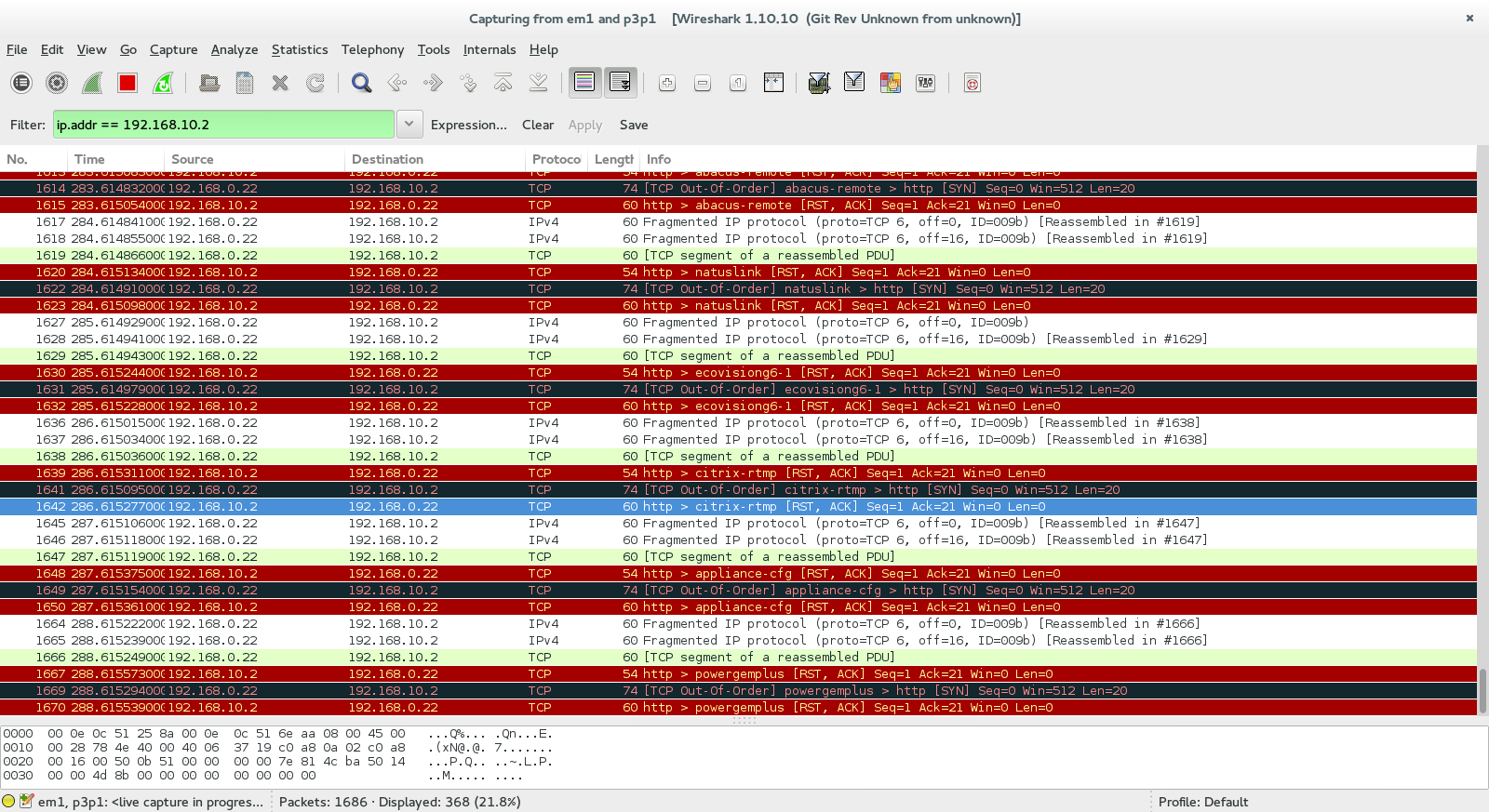


IPtables before the command.

IPtables after the command.

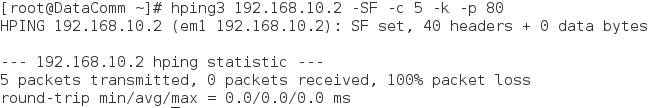


Wireshark Capture of above. We see the packets being fragmented by the Firewall, and then accepted by the internal host.

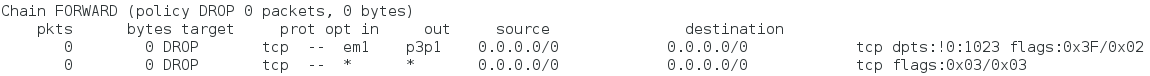


Test Case 8: Drop all TCP packets with the SYN and FIN bit set

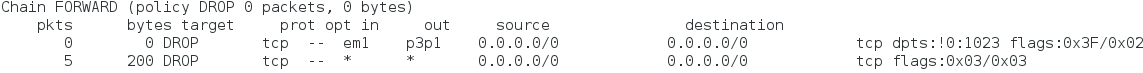
To test, we sent a packet on an open TCP port (80) with the S and F set. The below command was used.



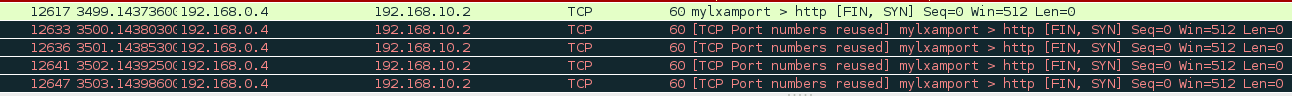
IPtables before the command:



IPtables after:

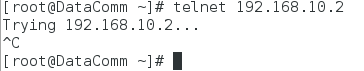


Firewall Wireshark Capture of the dropped packets.

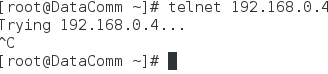


Test Case 9: Do not allow Telnet packets

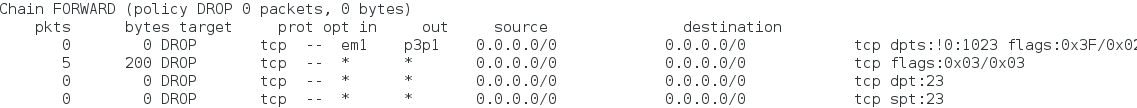
To test, we tried to Telnet the internal host:



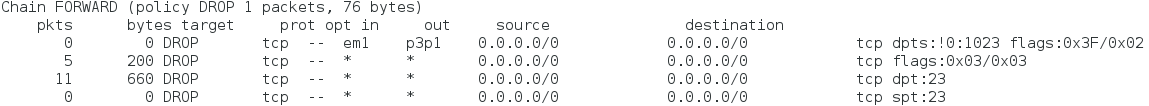
We also tried to telnet the external host from the internal host.



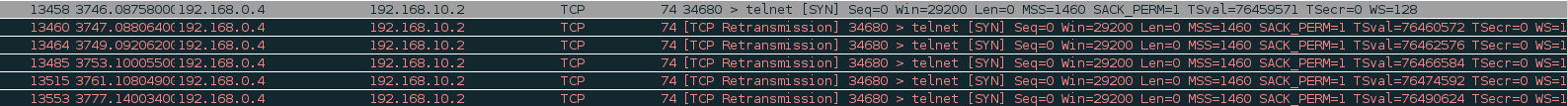
IPtables before the attempted Telnet connections:



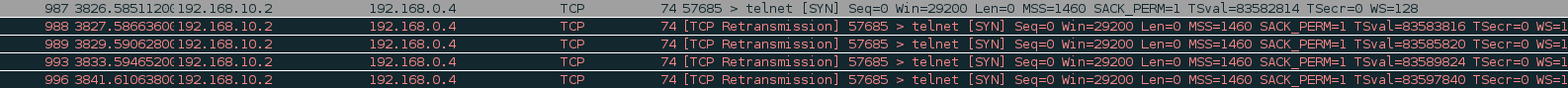
IPtables after the attempted Telnet connection:



Firewall connection Wireshark capture of the Telnet attempt:

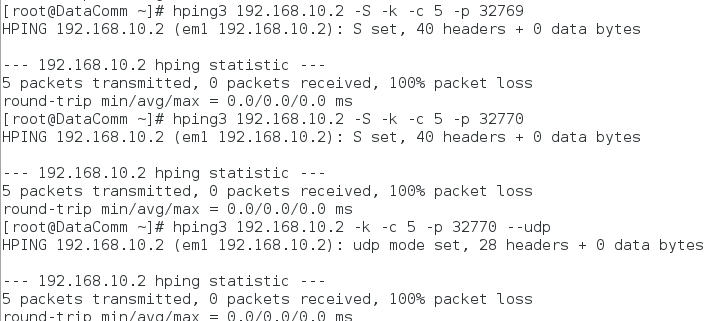


Internal Host Wireshark capture of the Telnet connection:



Test Case 10: Block all external traffic directed to ports 32768-32775

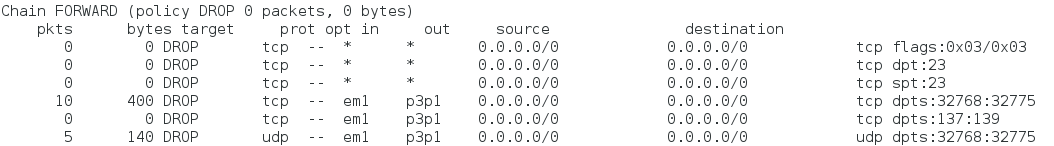
To test we entered the commands below:



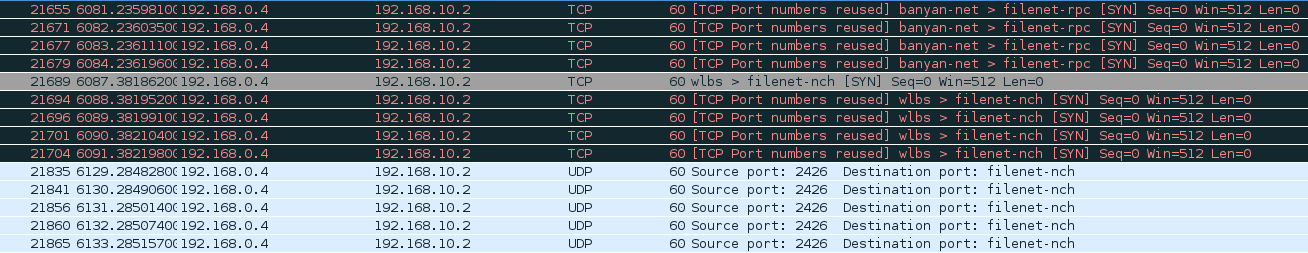
IPtables before the command:



IPtables after:

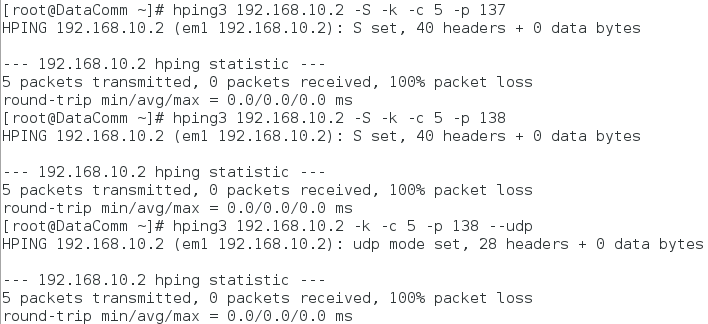


Firewall Wireshark capture of above:

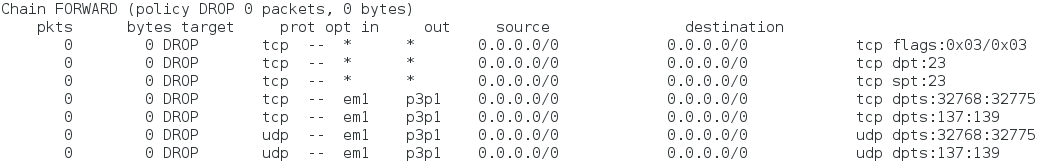


Test Case 11: Block all external traffic directed to ports 137-139

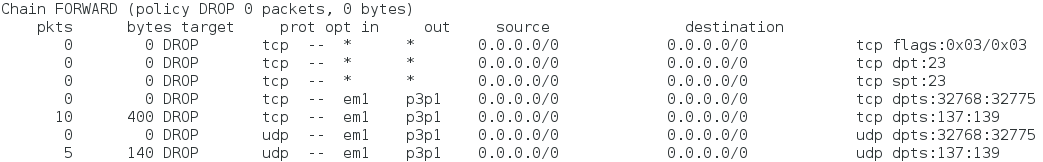
We tested all 3 ports by entereing the following commands:



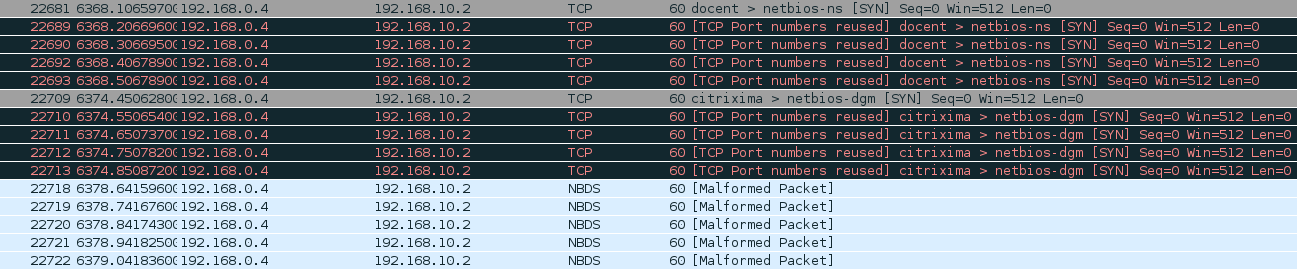
IPtables before:



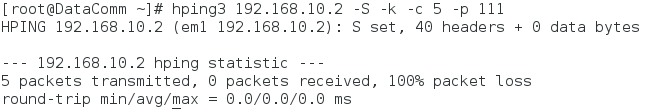
IPtables after:



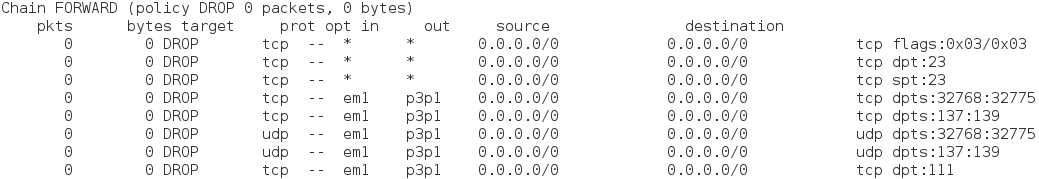
Firewall Wireshark of the dropped packets.



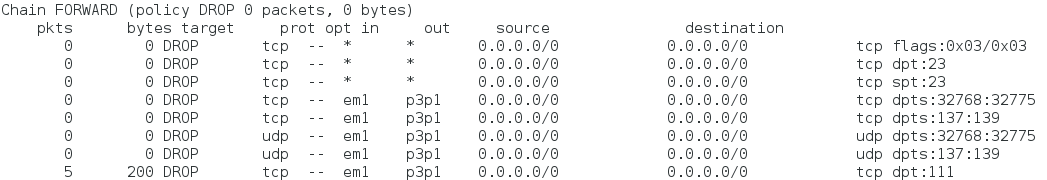
Test Case 12: Block all external traffic directed to TCP on port 111

We entered the following command:

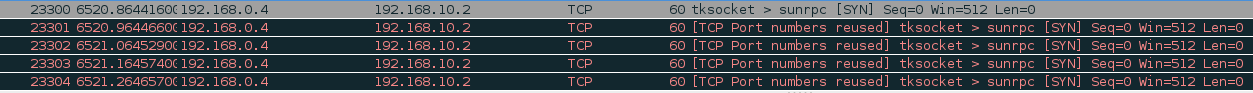
IPtables before:



IPtables after:

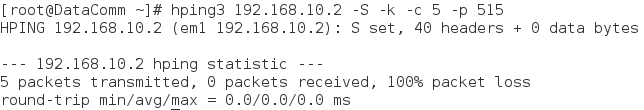


Firewall Wireshark capture of above:

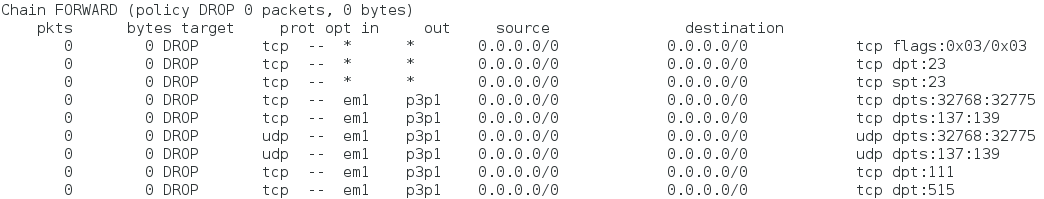


Test Case 13: Block all external traffic directed to TCP on port 515

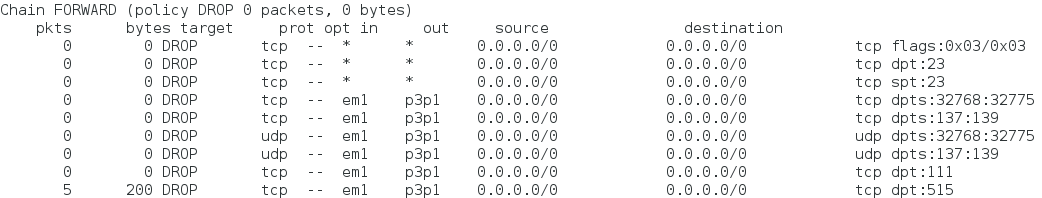
To test we entered the below command:



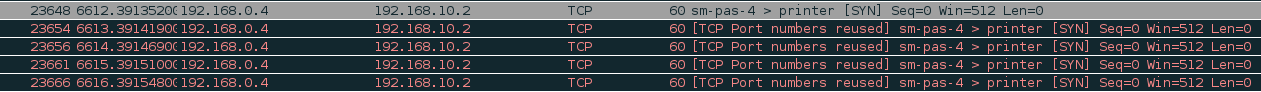
IPtables before:



IPtables after:

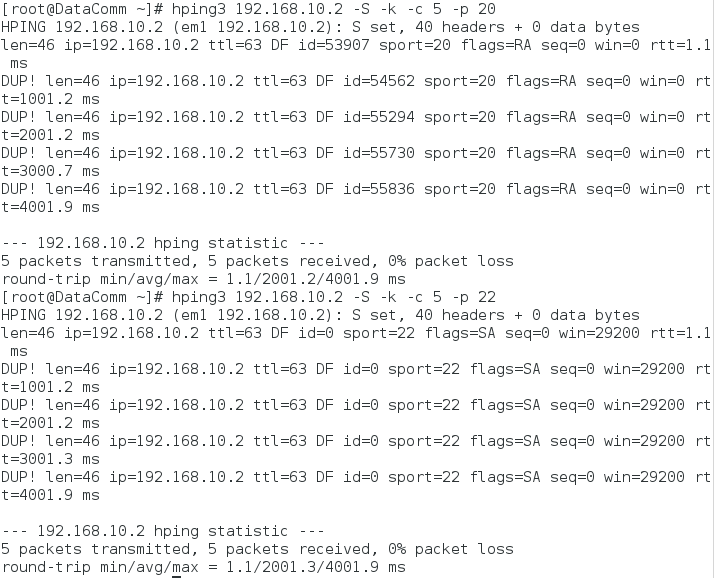


Wireshark Capture of above:

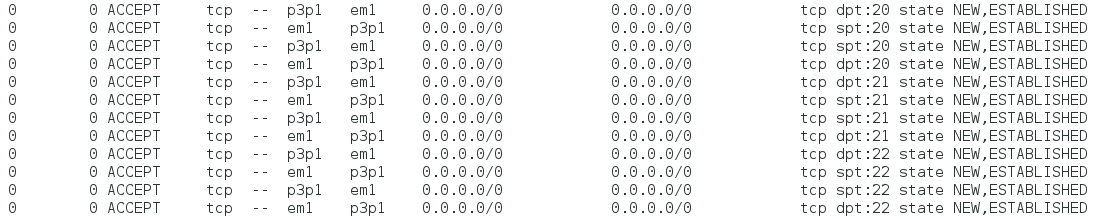


Test Case 14: Ensure FTP and SSH services have “Minimum Delay” (Requires a new DSCP column in wireshark. Value = ‘ip.dsfield.dscp’)

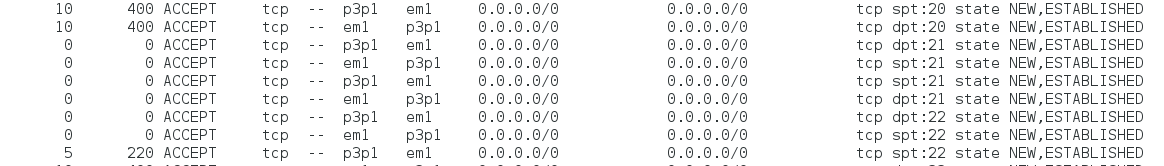
We entered the below command to test FTP followed by SSH:



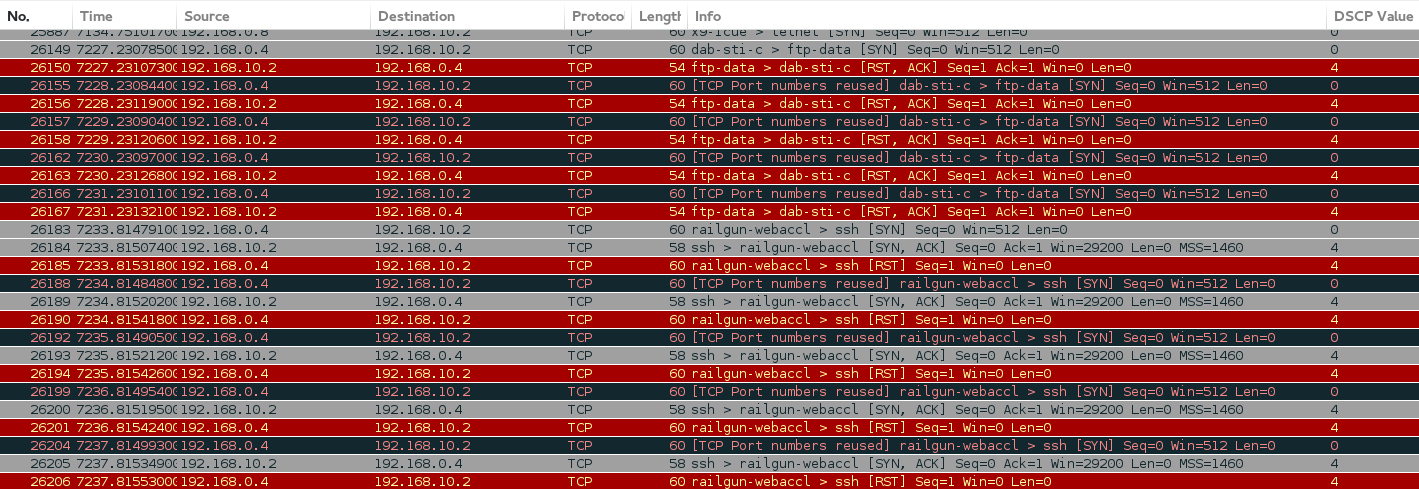
IPtables before:



IPtables after:



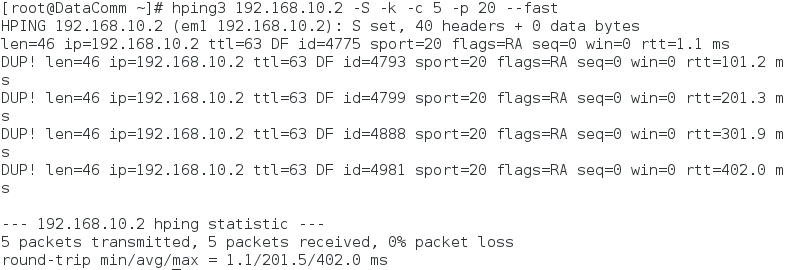
Wireshark capture of above. We see the DSCP column show a DCSP value of 4, which signifies the packet has minimum delay.



Test Case 15: Ensure FTP Data service has “Maximum Throughput”

(Requires a new DSCP column in wireshark. Value = ‘ip.dsfield.dscp’)

To test we sent FTP packets at port 20 with the following command:



IPTables before:



IPtables after:



Wireshark Capture of above. We see the DSCP column show a value of 2, which signifies maximum throughput.