Personal Firewalls - An Introduction to Firewall Administration

David Tran - A00801942

COMP 6D

COMP 8006

British Columbia Institute of Technology

Aman Abdulla

Thursday, February 6 2014

# Table of Contents

# Design Work

// variables to set ie…

// WWW\_PORT=’80,443’

// SSH\_PORT=’22’

// and so on...

# Flush the tables

// flush the rule sets

// flush any existing chains

# Set the default policies

// set default input policies to drop

// set default output policies to drop

// set default forward policies to drop

# User-Defined Chains

// create chains: ssh-traffic, www-traffic, noness-traffic

// activate the three chains made

// any input with protocol tcp and its source port is 22, send it to ssh-traffic chain

// any input with protocol tcp and its source port is 80 or 443, send it to www-traffic chain

// any input otherwise will be sent through noness-traffic chain

# Allow DNS traffic

// allow any input through noness-traffic with protocol udp and ports 53 to be accepted

// allow any output through noness-traffic with protocol udp and ports 53 to be accepted

# Allow DHCP traffic

// allow any input through noness-traffic with protocol udp and ports 67 to 68 to be accepted

// allow any output through noness-traffic with protocol udp and ports 67 to 68 to be accepted

# Drop all inbound traffic to HTTP from source ports less than 1024

// drop any traffic coming through www-traffic with protocol tcp and source port 0 to 1023, and destination port 80 and 443

# Allow inbound and outbound HTTP packets

// allow any traffic coming through www-traffic with protocol tcp and destination port 80 or 443 that has a new or established state

// allow any traffic going out www-traffic with protocol tcp and source ports 80 or 443 with states that are established

# Allow inbound and outbound SSH packets

// allow any traffic coming through ssh-traffic with protocol tcp and destination port 22 that has a new or established state

// allow any traffic going out ssh-traffic with protocol tcp and source ports 22 with states that are established

# Drop all incoming and outgoing packets to and from port 0

// drop all incoming packets with protocol tcp to destination port 0

// drop all incoming packets with protocol udp to destination port 0

// drop all outgoing packets with protocol tcp from source port 0

// drop all outgoing packets with protocol udp from source port 0

# Drop all inbound SYN packets

// drop anything coming in with protocol tcp and flagged as SYN with a new state

# Traffic Accounting Rules

// anything coming to port 80 or 443, send through chain www-traffic

// anything going from port 80 or 443, send it through chain www-traffic

// anything coming to port 22, send through chain ssh-traffic

// anything going from port 22, send it through chain ssh-traffic

// anything coming to neither port 80, 443 and 22, and protocol is tcp, send through chain noness-traffic

// anything coming to neither port 80, 443 and 22, and protocol is udp, send through chain noness-traffic

// anything going from neither port 80, 443 and 22, and protocol is tcp, send through chain noness-traffic

// anything going from neither port 80, 443 and 22, and protocol is udp, send through chain noness-traffic

# Save, Restart and List the IP tables

// save the tables

// restart the daemon

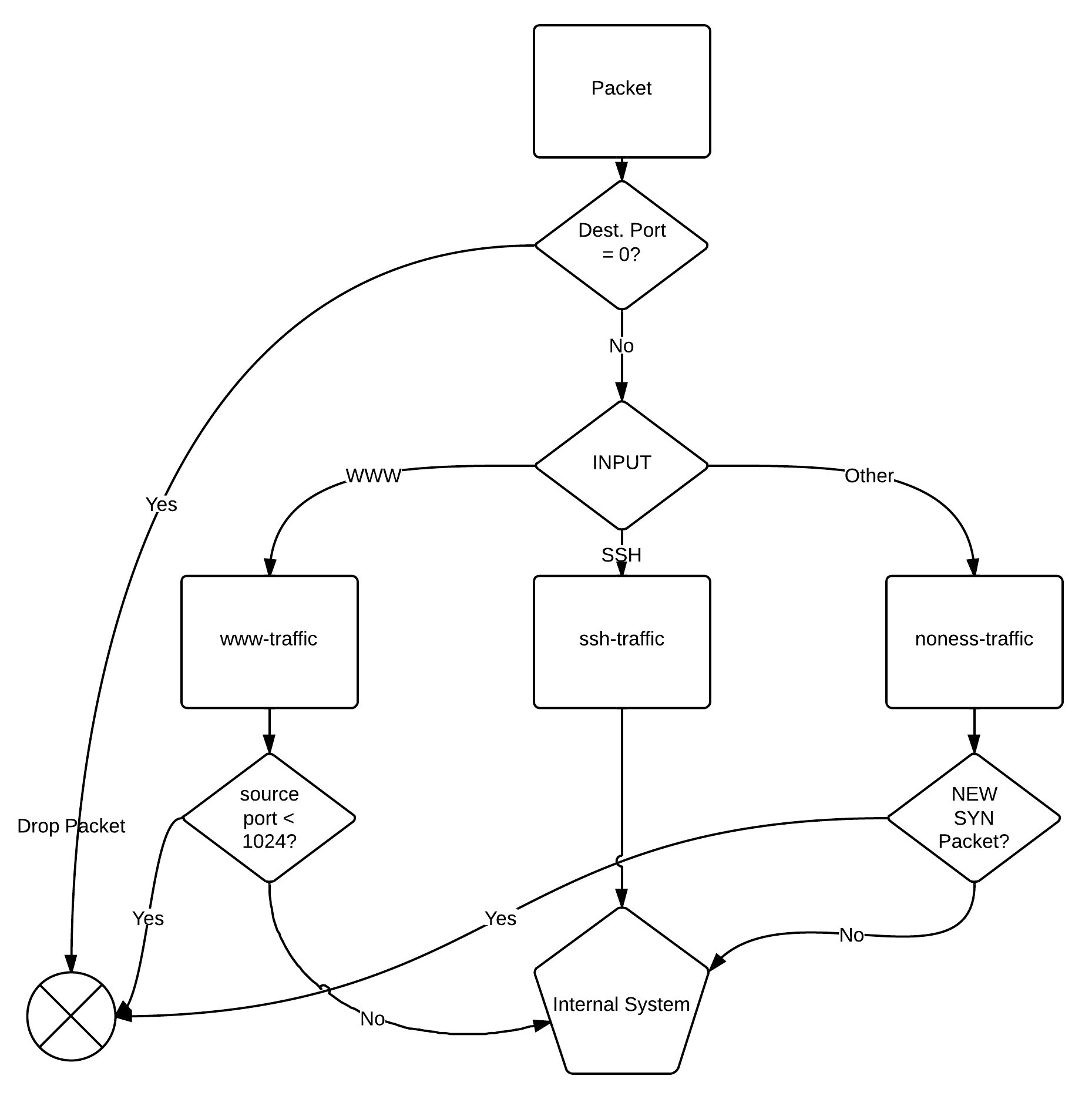
// list them; they should be as we just described above

# 

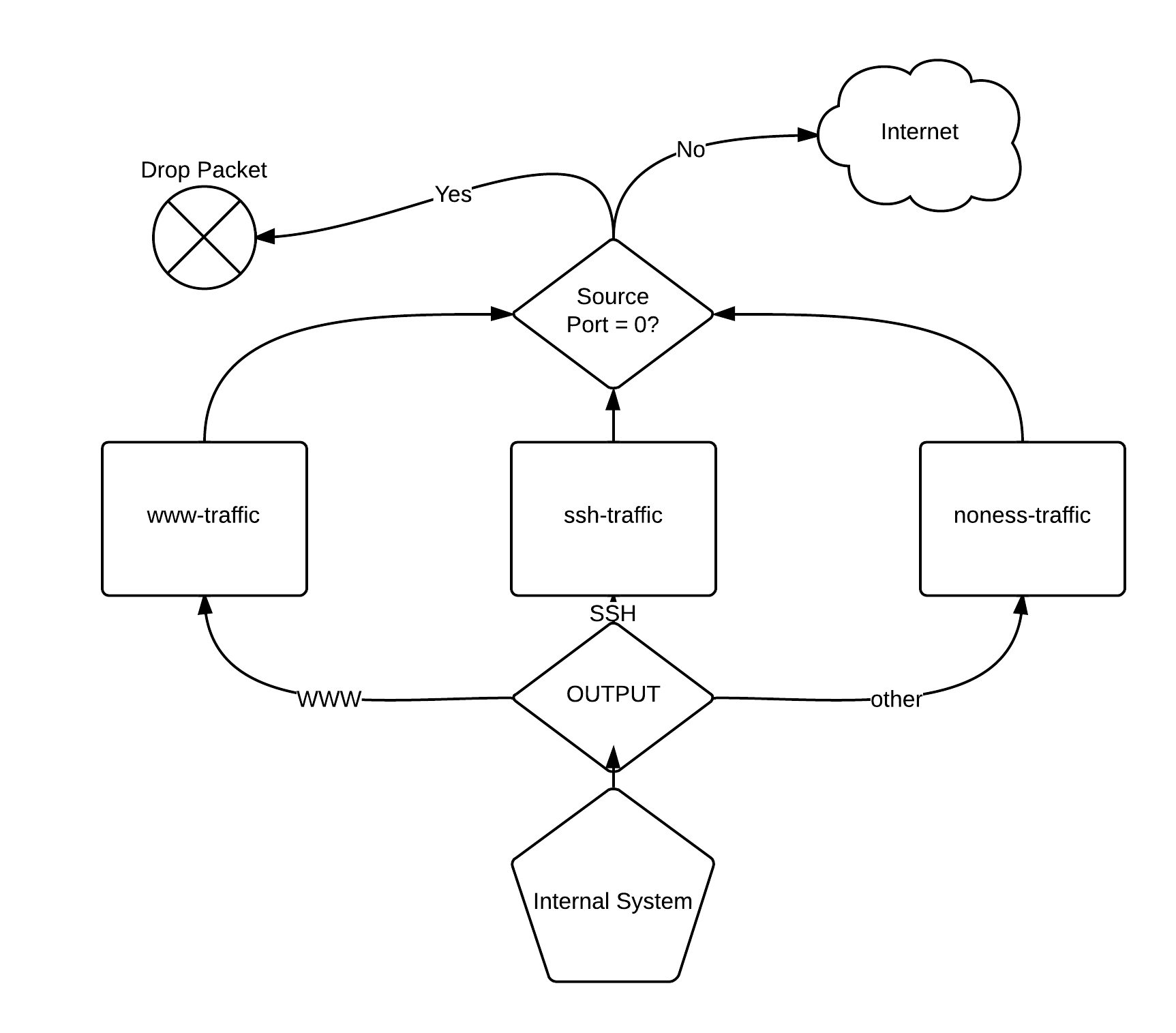
# 

# Flow-Chart Diagram

## Input



## Output

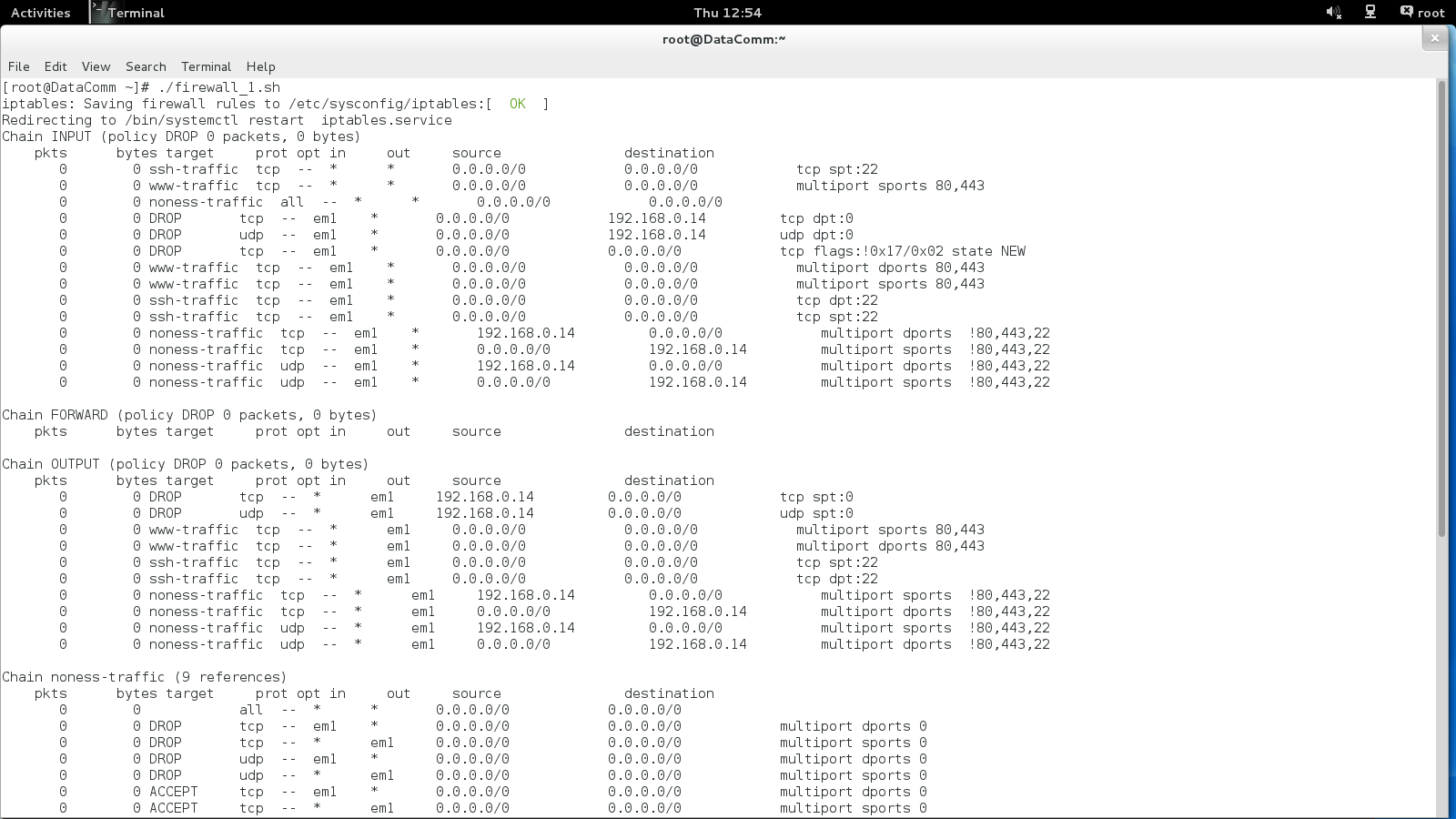


# Preliminary Testing

Running the firewall script takes no arguments. Any arguments given to the execution of the script will be ignored.

## Test Case 1: Normal Input

Here is what happens when the file is executed with no arguments:



There are no errors or warnings.

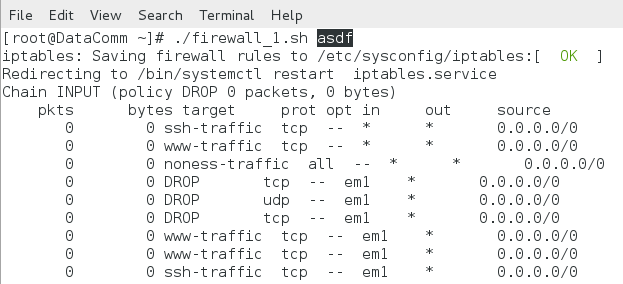
RESULT: PASSED

## 

## 

## Test Case 2: Input followed by (an unnecessary) argument

Here is what happens when the file is executed with an unnecessary argument:



Note the highlighted “argument”. As you can see, no errors are displayed and the highlighted argument is simply ignored.

RESULT: PASSED