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

Background

The purpose of this assignment is to familiarize the budding administrator with the syntax and implementation of firewall rules on a Linux machine. The objective is to design a set of rules that follow the criteria of Assignment 1, as well as documenting a set of test cases properly so that the routine becomes natural. Testing in this assignment will require a tertiary computer terminal located in the same local area network. In this test case, we will assume that the tertiary computer will not have anything but the default firewall in place.

Tools & Equipment

Hardware

- 8GB RAM
- Intel i5 Quad Core
- 500GB HDD
- two computer terminals

Software

- Fedora Linux 19 64-bit
- hping3
- Shell Script
- Wireshark
- iptables
- ssh
- dhclient

Testing Procedure

Test Cases Table

Case #	Test Case	Tools Used	Expected Outcome	Results
1	Verify that iptables settings are changed after script is ran	iptables -L -n -v -x	Rules are changed according to script	PASSED. See details.
2	DNS packets are allowed to and from the host	Wireshark, Google Chrome	Packets are allowed to travel to and from the machine	PASSED. See details.
3	DHCP packets are allowed to and from the host	ifconfig, dhclient	Packets are allowed to travel to and from the machine	PASSED. See details.
4	SSH traffic is allowed and accounted for	ssh, iptables -L -n -v -x	SSH traffic is permitted inbound	PASSED. See

			and outbound	details.
5	WWW(80) traffic is allowed and accounted for	hping3, iptables -L -n -v -x	WWW traffic is permitted inbound and outbound	PASSED. See details.
6	WWW(443) traffic is allowed and accounted for	hping3, iptables -L -n -v -x	WWW traffic is permitted inbound and outbound	PASSED. See details.
7	WWW(80) traffic is disallowed and accounted for when source ports are less than 1024	hping3, iptables -L -n -v -x	WWW traffic is not permitted inbound and outbound using port 0	PASSED. See details.
8	WWW(80) traffic is disallowed and accounted for when source ports are less than 1024	hping3, iptables -L -n -v -x	WWW traffic is not permitted inbound and outbound using port 1023	PASSED. See details.
9	WWW(80) traffic is disallowed and accounted for when source ports are less than 1024	hping3, iptables -L -n -v -x	WWW traffic is permitted inbound and outbound using port 1024	PASSED. See details.
10	WWW(443) traffic is disallowed and accounted for when source ports are less than 1024	hping3, iptables -L -n -v -x	WWW traffic is not permitted inbound and outbound using port 0	PASSED. See details.
11	WWW(443) traffic is disallowed and accounted for when source ports are less than 1024	hping3, iptables -L -n -v -x	WWW traffic is not permitted inbound and outbound using port 1023	PASSED. See details.
12	WWW(443) traffic is disallowed and accounted for when source ports are less than 1024	hping3, iptables -L -n -v -x	WWW traffic is permitted inbound and outbound using port 1024	PASSED. See details.
13	Incoming packets to port 0 are dropped (TCP)	hping3, iptables -L -n -v -x,	Incoming packets are dropped	PASSED. See details.
14	Incoming packets to port 0 are dropped (UDP)	hping3, iptables -L -n -v -x,	Incoming packets are dropped	PASSED. See details.
15	WWW(80) incoming traffic is accounted for	hping3, iptables -L -n -v -x	Instances of www-traffic is incremented	PASSED. See details.

16	WWW(443) incoming traffic is accounted for	hping3, iptables -L -n -v -x	Instances of www-traffic is incremented	PASSED. See details.
17	WWW(80) outgoing traffic is accounted for	hping3, iptables -L -n -v -x	Instances of www-traffic is incremented	PASSED. See details.
18	WWW(443) outgoing traffic is accounted for	hping3, iptables -L -n -v -x	Instances of www-traffic is incremented	PASSED. See details.
19	SSH(22) incoming traffic is accounted for	ssh, hping3, iptables -L -n -v -x	Instances of ssh-traffic is incremented	PASSED. See details.
20	SSH(22) outgoing traffic is accounted for	ssh, hping3, iptables -L -n -v -x	Instances of ssh-traffic is incremented	PASSED. See details.
21	All inbound SYN packets are dropped	hping3	SYN packets are dropped; exceptions are permitted	PASSED. See details.
22	All other undefined ports with inbound traffic are accounted for (TCP)	hping3, iptables -L -n -v -x	Ports 22, 80, 443 will not increment, all others will increment	PASSED. See details.
23	All other undefined ports with outbound traffic are accounted for (TCP)	hping3, iptables -L -n -v -x	Ports 22, 80, 443 will not increment, all others will increment	PASSED. See details.
24	All other undefined ports with inbound traffic are accounted for (UDP)	hping3, iptables -L -n -v -x	Ports 22, 80, 443 will not increment, all others will increment	PASSED. See details.
25	All other undefined ports with inbound traffic are accounted for (UDP)	hping3, iptables -L -n -v -x	Ports 22, 80, 443 will not increment, all others will increment	PASSED. See details.

Test Case Evidence & Details

(1) Verify that iptables settings are changed after script is ran

Before: iptables -L -n -v -x

```
[root@DataComm ~]# iptables -L -n -v -x
Chain INPUT (policy DROP 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source               destination
     796   104249 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy DROP 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source               destination
     718   237901 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
[root@DataComm ~]#
```

After: ./firewall_1.sh

```
[root@DataComm ~]# ./firewall_1.sh
iptables: Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
Redirecting to /bin/systemctl restart  iptables.service
Chain INPUT (policy DROP 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 tcp_in_chain  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0
       0        0 udp_in_chain  udp  --  em1    *       0.0.0.0/0            0.0.0.0/0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2          tcp dpt:0
       0        0 DROP       udp  --  em1    *       0.0.0.0/0            192.168.0.2          udp dpt:0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0            tcp flags:!0x17/0x02 state NEW

Chain FORWARD (policy DROP 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 tcp_out_chain  tcp  --  *      em1     0.0.0.0/0            0.0.0.0/0
       0        0 udp_out_chain  udp  --  *      em1     0.0.0.0/0            0.0.0.0/0
       0        0 DROP       tcp  --  *      em1     192.168.0.2          0.0.0.0/0            tcp spt:0
       0        0 DROP       udp  --  *      em1     192.168.0.2          0.0.0.0/0            udp spt:0

Chain noness-traffic (10 references)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
       0        0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            udp dpt:53
       0        0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            udp spt:53
       0        0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:53
       0        0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp spt:53
       0        0 ACCEPT     udp  --  *      em1     0.0.0.0/0            0.0.0.0/0            udp dpts:68:67
       0        0 ACCEPT     udp  --  em1    *       0.0.0.0/0            0.0.0.0/0            udp spts:67:68

Chain ssh-traffic (5 references)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
       0        0 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          tcp dpt:22 state NEW,ESTABLISHED
       0        0 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            tcp spt:22 state ESTABLISHED
       0        0 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            tcp dpt:22 state NEW,ESTABLISHED
       0        0 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          tcp spt:22 state ESTABLISHED

Chain tcp_in_chain (1 references)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
       0        0 ssh-traffic  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp spt:22
       0        0 www-traffic  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            multiport sports 80,443
       0        0 noness-traffic  all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

(2) DNS Packets are allowed to and from the host

To begin, we opened a browser and navigated to Google.ca (142.232.191.38)

Inbound DNS Packets:

Filter: dns && ip.dst eq 192.168.0.2 Expression... Clear Apply Save

No.	Time	Source	Destination	Protocc	Lengtl	Info
80	30.775388000	142.232.191.38	192.168.0.2	DNS	121	Standard query response 0x22b2
183	32.199753000	142.232.191.38	192.168.0.2	DNS	117	Standard query response 0x9441
193	32.221983000	142.232.191.38	192.168.0.2	DNS	135	Standard query response 0xab1b
208	32.240683000	142.232.191.38	192.168.0.2	DNS	272	Standard query response 0xfd37
209	32.240694000	142.232.191.38	192.168.0.2	DNS	149	Standard query response 0xeb65
211	32.240784000	142.232.191.38	192.168.0.2	DNS	162	Standard query response 0xd6be
214	32.240871000	142.232.191.38	192.168.0.2	DNS	251	Standard query response 0xfd59
215	32.240881000	142.232.191.38	192.168.0.2	DNS	91	Standard query response 0x13a9
217	32.240943000	142.232.191.38	192.168.0.2	DNS	154	Standard query response 0x950e
219	32.242548000	142.232.191.38	192.168.0.2	DNS	91	Standard query response 0x169c

Outbound DNS Packets:

Filter: dns && ip.src eq 192.168.0.2 Expression... Clear Apply Save

No.	Time	Source	Destination	Protocc	Lengtl	Info
79	30.774443000	192.168.0.2	142.232.191.38	DNS	73	Standard query 0x22b2 A www.google.ca
173	32.181664000	192.168.0.2	142.232.191.38	DNS	69	Standard query 0x9441 A google.ca
191	32.221147000	192.168.0.2	142.232.191.38	DNS	78	Standard query 0xab1b A accounts.google.ca
201	32.238527000	192.168.0.2	142.232.191.38	DNS	75	Standard query 0xfd37 A apis.google.com
202	32.238579000	192.168.0.2	142.232.191.38	DNS	72	Standard query 0xeb65 A id.google.ca
203	32.238660000	192.168.0.2	142.232.191.38	DNS	85	Standard query 0xd6be A lh4.googleusercontent.com
204	32.238739000	192.168.0.2	142.232.191.38	DNS	75	Standard query 0xfd59 A plus.google.com
205	32.238785000	192.168.0.2	142.232.191.38	DNS	75	Standard query 0x13a9 A ssl.gstatic.com
206	32.238876000	192.168.0.2	142.232.191.38	DNS	74	Standard query 0x950e A www.google.com
210	32.240752000	192.168.0.2	142.232.191.38	DNS	75	Standard query 0x169c A www.gstatic.com

(3) DHCP Packets are allowed to and from the host

Firstly, in Terminal, we deactivated our network card and then brought it back up again using: ipconfig em1 down; ipconfig em1 up;

```
[root@DataComm ~]# ifconfig em1 down
[root@DataComm ~]# ifconfig em1 up
[root@DataComm ~]# ifconfig
em1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::7a2b:cbff:fea3:4325  prefixlen 64  scopeid 0x20<link>
        ether 78:2b:cb:a3:43:25  txqueuelen 1000  (Ethernet)
        RX packets 152599  bytes 71349241 (68.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 62561  bytes 14364078 (13.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 20  memory 0xe1a00000-e1a20000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 284  bytes 22708 (22.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 284  bytes 22708 (22.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Afterwards, we ran this command:

dhclient em1

```
[root@DataComm ~]# dhclient em1
[root@DataComm ~]# ifconfig
em1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.2  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::7a2b:cbff:fea3:4325  prefixlen 64  scopeid 0x20<link>
        ether 78:2b:cb:a3:43:25  txqueuelen 1000  (Ethernet)
        RX packets 152806  bytes 71443815 (68.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 62757  bytes 14394570 (13.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 20  memory 0xe1a00000-e1a20000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 284  bytes 22708 (22.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 284  bytes 22708 (22.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

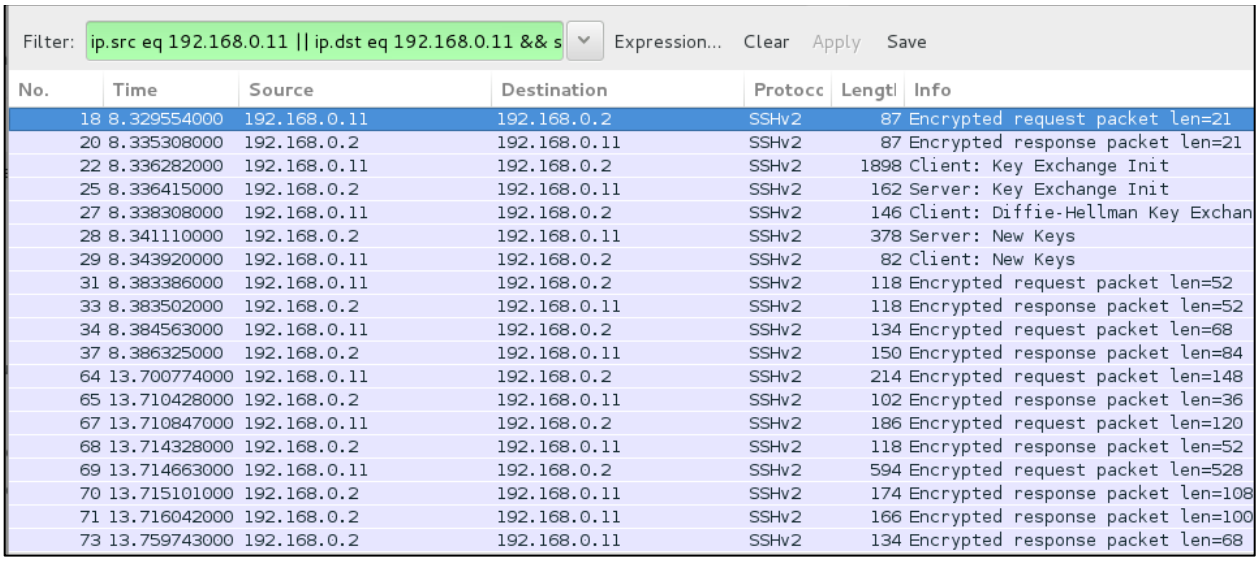
We were then able to see that we were dedicated the same IPv4 Address again: 192.168.0.2

(4) SSH traffic is allowed to and from the host

In a second computer terminal, assume that this terminal is the client. We entered the following command into Terminal:

ssh 192.168.0.2

Here's the Wireshark capture of inbound and outbound traffic for SSHv2



No.	Time	Source	Destination	Protocc	Lengtl	Info
18	8.329554000	192.168.0.11	192.168.0.2	SSHv2	87	Encrypted request packet len=21
20	8.335308000	192.168.0.2	192.168.0.11	SSHv2	87	Encrypted response packet len=21
22	8.336282000	192.168.0.11	192.168.0.2	SSHv2	1898	Client: Key Exchange Init
25	8.336415000	192.168.0.2	192.168.0.11	SSHv2	162	Server: Key Exchange Init
27	8.338308000	192.168.0.11	192.168.0.2	SSHv2	146	Client: Diffie-Hellman Key Exchan
28	8.341110000	192.168.0.2	192.168.0.11	SSHv2	378	Server: New Keys
29	8.343920000	192.168.0.11	192.168.0.2	SSHv2	82	Client: New Keys
31	8.383386000	192.168.0.11	192.168.0.2	SSHv2	118	Encrypted request packet len=52
33	8.383502000	192.168.0.2	192.168.0.11	SSHv2	118	Encrypted response packet len=52
34	8.384563000	192.168.0.11	192.168.0.2	SSHv2	134	Encrypted request packet len=68
37	8.386325000	192.168.0.2	192.168.0.11	SSHv2	150	Encrypted response packet len=84
64	13.700774000	192.168.0.11	192.168.0.2	SSHv2	214	Encrypted request packet len=148
65	13.710428000	192.168.0.2	192.168.0.11	SSHv2	102	Encrypted response packet len=36
67	13.710847000	192.168.0.11	192.168.0.2	SSHv2	186	Encrypted request packet len=120
68	13.714328000	192.168.0.2	192.168.0.11	SSHv2	118	Encrypted response packet len=52
69	13.714663000	192.168.0.11	192.168.0.2	SSHv2	594	Encrypted request packet len=528
70	13.715101000	192.168.0.2	192.168.0.11	SSHv2	174	Encrypted response packet len=108
71	13.716042000	192.168.0.2	192.168.0.11	SSHv2	166	Encrypted response packet len=100
73	13.759743000	192.168.0.2	192.168.0.11	SSHv2	134	Encrypted response packet len=68

Here is the accounting of SSH traffic:

```
[root@DataComm ~]# iptables -L ssh-traffic -v -n -x
Chain ssh-traffic (5 references)
    pkts      bytes target     prot opt in     out     source               destination
      37     6093 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0
       0        0 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          tcp dpt:22 state NEW,ESTABLISHED
       0        0 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            tcp spt:22 state ESTABLISHED
       0        0 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            tcp dpt:22 state NEW,ESTABLISHED
       0        0 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          tcp spt:22 state ESTABLISHED
[root@DataComm ~]#
```

(5) WWW(80) traffic is allowed and accounted for

From the testing terminal, here are the results of the hping command to our personal firewall:

```
[root@DataComm ~]# hping3 192.168.0.2 -p 80 -c 5
HPING 192.168.0.2 (em1 192.168.0.2): NO FLAGS are set, 40 headers + 0 data bytes
len=46 ip=192.168.0.2 ttl=64 DF id=48618 sport=80 flags=RA seq=0 win=0 rtt=0.3 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48619 sport=80 flags=RA seq=1 win=0 rtt=0.3 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48620 sport=80 flags=RA seq=2 win=0 rtt=0.3 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48621 sport=80 flags=RA seq=3 win=0 rtt=0.4 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48622 sport=80 flags=RA seq=4 win=0 rtt=0.4 ms

--- 192.168.0.2 hping statistic ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.3/0.4 ms
[root@DataComm ~]#
```

Here are the results of the machine containing our firewall sending outbound packets:

```
[root@DataComm ~]# hping3 192.168.0.11 -S -p 80 -c 5
HPING 192.168.0.11 (em1 192.168.0.11): S set, 40 headers + 0 data bytes
len=46 ip=192.168.0.11 ttl=64 DF id=0 sport=80 flags=SA seq=0 win=29200 rtt=0.3 ms
len=46 ip=192.168.0.11 ttl=64 DF id=0 sport=80 flags=SA seq=1 win=29200 rtt=0.3 ms
len=46 ip=192.168.0.11 ttl=64 DF id=0 sport=80 flags=SA seq=2 win=29200 rtt=0.4 ms
len=46 ip=192.168.0.11 ttl=64 DF id=0 sport=80 flags=SA seq=3 win=29200 rtt=0.4 ms
len=46 ip=192.168.0.11 ttl=64 DF id=0 sport=80 flags=SA seq=4 win=29200 rtt=0.4 ms

--- 192.168.0.11 hping statistic ---
5 packets tramitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.4/0.4 ms
[root@DataComm ~]#
```

And here are the accounting information regarding WWW(80) traffic using iptables:

```
[root@DataComm ~]# iptables -L www-traffic -v -x -n
Chain www-traffic (5 references)
    pkts      bytes target     prot opt in     out     source               destination
      58     7347            all  --  *      *       0.0.0.0/0            0.0.0.0/0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2          tcp spts:0:1023 dpt:80
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2          tcp spts:0:1023 dpt:443
       5      200 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          multiport dports 80,443 state NEW,ESTABLISHED
       5      200 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            multiport sports 80,443 state ESTABLISHED
      24     5172 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            multiport dports 80,443 state NEW,ESTABLISHED
      24     1775 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          multiport sports 80,443 state ESTABLISHED
[root@DataComm ~]#
```

(6) WWW(443) traffic is allowed and accounted for

From the testing terminal, here are the results of the hping command to our personal firewall:

```
[root@DataComm ~]# hping3 192.168.0.2 -S -p 443 -c 5
HPING 192.168.0.2 (em1 192.168.0.2): S set, 40 headers + 0 data bytes
len=46 ip=192.168.0.2 ttl=64 DF id=48638 sport=443 flags=RA seq=0 win=0 rtt=0.4 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48639 sport=443 flags=RA seq=1 win=0 rtt=0.4 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48640 sport=443 flags=RA seq=2 win=0 rtt=0.3 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48641 sport=443 flags=RA seq=3 win=0 rtt=0.3 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48642 sport=443 flags=RA seq=4 win=0 rtt=0.3 ms

--- 192.168.0.2 hping statistic ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.4/0.4 ms
[root@DataComm ~]#
```

Here are the results of the machine containing our firewall sending outbound packets:

```
[root@DataComm ~]# hping3 192.168.0.11 -S -p 443 -c 5
HPING 192.168.0.11 (em1 192.168.0.11): S set, 40 headers + 0 data bytes
len=46 ip=192.168.0.11 ttl=64 DF id=18612 sport=443 flags=RA seq=0 win=0 rtt=0.3 ms
len=46 ip=192.168.0.11 ttl=64 DF id=18613 sport=443 flags=RA seq=1 win=0 rtt=0.3 ms
len=46 ip=192.168.0.11 ttl=64 DF id=18614 sport=443 flags=RA seq=2 win=0 rtt=0.3 ms
len=46 ip=192.168.0.11 ttl=64 DF id=18615 sport=443 flags=RA seq=3 win=0 rtt=0.4 ms
len=46 ip=192.168.0.11 ttl=64 DF id=18616 sport=443 flags=RA seq=4 win=0 rtt=0.3 ms

--- 192.168.0.11 hping statistic ---
5 packets tramitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.3/0.4 ms
[root@DataComm ~]#
```

And here are the accounting information regarding WWW(80) traffic using iptables:

```
[root@DataComm ~]# iptables -L www-traffic -v -x -n
Chain www-traffic (5 references)
    pkts      bytes target     prot opt in     out     source               destination
     268    88768            all  --  *      *       0.0.0.0/0            0.0.0.0/0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2          tcp spts:0:1023 dpt:80
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2          tcp spts:0:1023 dpt:443
      10      400 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          multiport dports 80,443 state NEW,ESTABLISHED
      10      400 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            multiport sports 80,443 state ESTABLISHED
     113    21322 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            multiport dports 80,443 state NEW,ESTABLISHED
     135    66646 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          multiport sports 80,443 state ESTABLISHED
[root@DataComm ~]#
```

Note the increase of our packet count from 5 to 10.

(7) WWW(80) traffic is disallowed and accounted for when source ports are less than 1024

Here, we set our source port to the left-most extreme of our constraint (port 0). We will execute this on our non-firewalled terminal:

```
[root@DataComm ~]# hping3 192.168.0.2 -S -p 80 -c 5 -s 0
HPING 192.168.0.2 (em1 192.168.0.2): S set, 40 headers + 0 data bytes

--- 192.168.0.2 hping statistic ---
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@DataComm ~]#
```

Our packets are dropped here. Let's take a look at our packet auditing on our firewalled terminal:

```
[root@DataComm ~]# iptables -L www-traffic -n -v -x
Chain www-traffic (5 references)
    pkts      bytes target     prot opt in     out     source               destination
    1599   395621            all  --  *      *       0.0.0.0/0            0.0.0.0/0
       5      200 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2          tcp spts:0:1023 dpt:80
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2          tcp spts:0:1023 dpt:443
      10      400 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          multiport dports 80,443 state
```

(8) WWW(80) traffic is disallowed and accounted for when source ports are less than 1024
We will redo the test, except we will attempt our right-most extreme (port 1023).
NOTE: we have to specify the --keep switch, otherwise, our ports will increment

```
[root@DataComm ~]# hping3 192.168.0.2 -S -p 80 -c 5 -s 1023 --keep
HPING 192.168.0.2 (em1 192.168.0.2): S set, 40 headers + 0 data bytes

--- 192.168.0.2 hping statistic ---
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@DataComm ~]#
```

(9) WWW(80) traffic is disallowed and accounted for when source ports are less than 1024
We will redo our previous test, except this time, we will try a valid port (port 1024+)
NOTE: we won't have to specify --keep switch, because anything greater than 1024 is permitted

```
[root@DataComm ~]# hping3 192.168.0.2 -S -p 80 -c 5 -s 1024
HPING 192.168.0.2 (em1 192.168.0.2): S set, 40 headers + 0 data bytes
len=46 ip=192.168.0.2 ttl=64 DF id=48711 sport=80 flags=RA seq=0 win=0 rtt=0.3 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48712 sport=80 flags=RA seq=1 win=0 rtt=0.3 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48713 sport=80 flags=RA seq=2 win=0 rtt=0.3 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48714 sport=80 flags=RA seq=3 win=0 rtt=0.3 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48715 sport=80 flags=RA seq=4 win=0 rtt=0.4 ms

--- 192.168.0.2 hping statistic ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.3/0.4 ms
[root@DataComm ~]#
```

(10) WWW(443) traffic is disallowed and accounted for when source ports are less than 1024
We will flush our accounting traffic and restart the last 3 tests again for port 443 (https)
Starting with our left-most extreme (port 0):

```
[root@DataComm ~]# hping3 192.168.0.2 -S -p 443 -c 5 -s 0
HPING 192.168.0.2 (em1 192.168.0.2): S set, 40 headers + 0 data bytes

--- 192.168.0.2 hping statistic ---
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@DataComm ~]#
```

And then checking our accounting tables:

```
[root@DataComm ~]# iptables -L www-traffic -n -v -x
Chain www-traffic (5 references)
    pkts      bytes target     prot opt in     out     source               destination
     306    77018            all  --  *      *       0.0.0.0/0            0.0.0.0/0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2          tcp spts:0:1023 dpt:80
       5      200 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2          tcp spts:0:1023 dpt:443
       0        0 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          multiport dports 80,443
```

(11) WWW(443) traffic is disallowed and accounted for when source ports are less than 1024
NOTE: we have to specify --keep switch, otherwise our source port will increment
Starting with our right-most extreme (port 1023):

```
[root@DataComm ~]# hping3 192.168.0.2 -S -p 443 -c 5 -s 1023 --keep
HPING 192.168.0.2 (em1 192.168.0.2): S set, 40 headers + 0 data bytes

--- 192.168.0.2 hping statistic ---
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@DataComm ~]#
```

(12) WWW(443) traffic is disallowed and accounted for when source ports are less than 1024
NOTE: we won't have to specify --keep switch, because anything greater than 1024 is permitted
Using port 1024:

```
[root@DataComm ~]# hping3 192.168.0.2 -S -p 443 -c 5 -s 1024
HPING 192.168.0.2 (em1 192.168.0.2): S set, 40 headers + 0 data bytes
len=46 ip=192.168.0.2 ttl=64 DF id=48716 sport=443 flags=RA seq=0 win=0 rtt=0.4 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48717 sport=443 flags=RA seq=1 win=0 rtt=0.3 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48718 sport=443 flags=RA seq=2 win=0 rtt=0.4 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48719 sport=443 flags=RA seq=3 win=0 rtt=0.3 ms
len=46 ip=192.168.0.2 ttl=64 DF id=48720 sport=443 flags=RA seq=4 win=0 rtt=0.3 ms

--- 192.168.0.2 hping statistic ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.4/0.4 ms
[root@DataComm ~]#
```

(13) Incoming packets to port 0 are dropped (TCP)
Here is our command:

```
[root@DataComm ~]# hping3 192.168.0.2 -S -p 0 -c 5
HPING 192.168.0.2 (em1 192.168.0.2): S set, 40 headers + 0 data bytes

--- 192.168.0.2 hping statistic ---
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@DataComm ~]#
```

As you can see, no responses. Here is our accounting table to prove it:

```
[root@DataComm ~]# iptables -L -n -v -x
Chain INPUT (policy DROP 18 packets, 1867 bytes)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 ssh-traffic  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp spt:22
      12      966 www-traffic  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           multiport sp
      84     9198 noness-traffic  all  --  *      *       0.0.0.0/0            0.0.0.0/0
       5      200 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2         tcp dpt:0
       0        0 DROP       udp  --  em1    *       0.0.0.0/0            192.168.0.2         udp dpt:0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           tcp flags:!0x1
       0        0 www-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multiport dp
       0        0 www-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multiport sp
      59     6697 ssh-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22
       0        0 ssh-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           tcp spt:22
       0        0 noness-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multiport
       0        0 noness-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multiport
      18     1867 noness-traffic  udp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multiport
      18     1867 noness-traffic  udp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multiport
```

(14) Incoming packets to port 0 are dropped (UDP)
Here is our command; note the --udp switch:

```
[root@DataComm ~]# hping3 192.168.0.2 --udp -p 0 -c 5
HPING 192.168.0.2 (em1 192.168.0.2): udp mode set, 28 headers + 0 data bytes

--- 192.168.0.2 hping statistic ---
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@DataComm ~]#
```

As you can see, no responses. Here is our accounting table to prove it; note the difference between the TCP and UDP count:

```
[root@DataComm ~]# iptables -L -n -v -x
Chain INPUT (policy DROP 69 packets, 8629 bytes)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 ssh-traffic  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp spt:22
     430   176693 www-traffic  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           multiport s
     242    27184 noness-traffic  all  --  *      *       0.0.0.0/0            0.0.0.0/0
       5      200 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2         tcp dpt:0
       5      140 DROP       udp  --  em1    *       0.0.0.0/0            192.168.0.2         udp dpt:0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           tcp flags:!0x
       0        0 www-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multiport d
       0        0 www-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multiport s
     153    15830 ssh-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22
       0        0 ssh-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           tcp spt:22
       0        0 noness-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multipor
       0        0 noness-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multipor
      69     8629 noness-traffic  udp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multipor
      69     8629 noness-traffic  udp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multipor
```

(15) WWW(80) incoming traffic is accounted for
In our non-firewall terminal, we ran this command:
hping3 192.168.0.2 -S -p 80 -c 5
Here is our accounting table:

```
[root@DataComm ~]# iptables -L www-traffic -n -v -x
Chain www-traffic (5 references)
    pkts      bytes target     prot opt in     out     source               destination
      83    14624            all  --  *      *       0.0.0.0/0            0.0.0.0/0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2         tcp spts:0:1023 dpt:80
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2         tcp spts:0:1023 dpt:443
       5      200 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2         multiport dports 80,443 state NEW,ESTABLISHED
       5      200 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0           multiport sports 80,443 state ESTABLISHED
      39     7078 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0           multiport dports 80,443 state NEW,ESTABLISHED
      34     7146 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2         multiport sports 80,443 state ESTABLISHED
[root@DataComm ~]#
```

(16) WWW(80) outgoing traffic is accounted for
In our firewalled terminal, we ran this command:
hping3 192.168.0.11 -S -p 80 -c 5
Here is our accounting table:

```
[root@DataComm ~]# iptables -L www-traffic -n -v -x
Chain www-traffic (5 references)
    pkts      bytes target     prot opt in     out     source               destination
      83    14624            all  --  *      *       0.0.0.0/0            0.0.0.0/0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2         tcp spts:0:1023 dpt:80
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2         tcp spts:0:1023 dpt:443
       5      200 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2         multiport dports 80,443 state NEW,ESTABLISHED
       5      200 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0           multiport sports 80,443 state ESTABLISHED
      39     7078 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0           multiport dports 80,443 state NEW,ESTABLISHED
      34     7146 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2         multiport sports 80,443 state ESTABLISHED
[root@DataComm ~]#
```

NOTE the two different highlighted lines: the former depicts incoming while the latter depicts outgoing.

(17) WWW(443) incoming traffic is accounted for
In our non-firewall terminal, we ran this command:
hping3 192.168.0.2 -S -p 443 -c 5
Here is our accounting table:

```
[root@DataComm ~]# iptables -L www-traffic -n -v -x
Chain www-traffic (5 references)
    pkts      bytes target     prot opt in     out     source               destination
    1781   439307             all  --  *      *       0.0.0.0/0            0.0.0.0/0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2          tcp spts:0:1023 dpt:80
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2          tcp spts:0:1023 dpt:443
      10      400 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          multiport dports 80,443 state NEW,ESTABLISHED
      10      400 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            multiport sports 80,443 state ESTABLISHED
     855    83242 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            multiport dports 80,443 state NEW,ESTABLISHED
     906   355265 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          multiport sports 80,443 state ESTABLISHED
[root@DataComm ~]#
```

(18) WWW(443) outgoing traffic is accounted for
In our firewalled terminal, we ran this command:
hping3 192.168.0.11 -S -p 443 -c 5
Here is our accounting table:

```
[root@DataComm ~]# iptables -L www-traffic -n -v -x
Chain www-traffic (5 references)
    pkts      bytes target     prot opt in     out     source               destination
    1781   439307             all  --  *      *       0.0.0.0/0            0.0.0.0/0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2          tcp spts:0:1023 dpt:80
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2          tcp spts:0:1023 dpt:443
      10      400 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          multiport dports 80,443 state NEW,ESTABLISHED
      10      400 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            multiport sports 80,443 state ESTABLISHED
     855    83242 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            multiport dports 80,443 state NEW,ESTABLISHED
     906   355265 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          multiport sports 80,443 state ESTABLISHED
[root@DataComm ~]#
```

NOTE the two different highlighted lines: the former depicts incoming while the latter depicts outgoing. Also note the increase of packets from 5 to 10.

(19) SSH(22) incoming traffic is accounted for
In our non-firewalled terminal, we ran this command:
hping3 192.168.0.2 -S -p 22 -c 5
Here is our accounting table:

```
[root@DataComm ~]# iptables -L ssh-traffic -n -v -x
Chain ssh-traffic (5 references)
    pkts      bytes target     prot opt in     out     source               destination
      30     1240             all  --  *      *       0.0.0.0/0            0.0.0.0/0
      10      400 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          tcp dpt:22 state NEW,ESTABLISHED
       5      220 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            tcp spt:22 state ESTABLISHED
      10      400 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            tcp dpt:22 state NEW,ESTABLISHED
       5      220 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          tcp spt:22 state ESTABLISHED
[root@DataComm ~]#
```

(20) SSH(22) outgoing traffic is accounted for
In our firewalled terminal, we ran this command:
hping3 192.168.0.11 -S -p 22 -c 5
Here is our accounting table:

```
[root@DataComm ~]# iptables -L ssh-traffic -n -v -x
Chain ssh-traffic (5 references)
    pkts      bytes target     prot opt in     out     source               destination
      30     1240            all  --  *      *       0.0.0.0/0            0.0.0.0/0
      10      400 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          tcp dpt:22 state NEW,ESTABLISHED
       5      220 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            tcp spt:22 state ESTABLISHED
      10      400 ACCEPT     tcp  --  *      *       192.168.0.2          0.0.0.0/0            tcp dpt:22 state NEW,ESTABLISHED
       5      220 ACCEPT     tcp  --  *      *       0.0.0.0/0            192.168.0.2          tcp spt:22 state ESTABLISHED
[root@DataComm ~]#
```

NOTE the two different highlighted lines: the former depicts incoming while the latter depicts outgoing.

(21) All incoming SYN packets are dropped unless permitted
In our non-firewalled terminal, we ran this command with the following results:

```
[root@DataComm ~]# hping3 192.168.0.2 -S -c 5
HPING 192.168.0.2 (em1 192.168.0.2): S set, 40 headers + 0 data bytes

--- 192.168.0.2 hping statistic ---
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@DataComm ~]#
```

NOTE: There is no -p switch. We assume that SYN packets that are allowed are specified in previous test cases. Any other unspecified TCP packets coming in are therefore dropped because they are not explicitly permitted.

(22) All other undefined ports with inbound traffic are accounted for (TCP)
We ran hping3 with an undefined port of 8080:

```
[root@DataComm ~]# hping3 192.168.0.2 -S -p 8080 -c 5
HPING 192.168.0.2 (em1 192.168.0.2): S set, 40 headers + 0 data bytes

--- 192.168.0.2 hping statistic ---
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

We expect that it will drop these packets but we will see that it is accounted for in our tables:

```
[root@DataComm ~]# iptables -L INPUT -n -v -x
Chain INPUT (policy DROP 11 packets, 1131 bytes)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 ssh-traffic  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp spt:22
      31     6745 www-traffic  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           multiport sports 80,443
      12     1368 noness-traffic  all  --  *      *       0.0.0.0/0            0.0.0.0/0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2         tcp dpt:0
       0        0 DROP       udp  --  em1    *       0.0.0.0/0            192.168.0.2         udp dpt:0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           tcp flags:!0x17/0x02 state NEW
       0        0 www-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multiport dports 80,443
       0        0 www-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multiport sports 80,443
       0        0 ssh-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22
       0        0 ssh-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           tcp spt:22
       0        0 noness-traffic  tcp  --  em1    *       192.168.0.2          0.0.0.0/0           multiport dports  !80,443,22
       5      200 noness-traffic  tcp  --  em1    *       0.0.0.0/0            192.168.0.2         multiport sports  !80,443,22
       0        0 noness-traffic  udp  --  em1    *       192.168.0.2          0.0.0.0/0           multiport dports  !80,443,22
       0        0 noness-traffic  udp  --  em1    *       0.0.0.0/0            192.168.0.2         multiport sports  !80,443,22
[root@DataComm ~]#
```

(23) All other undefined ports with outbound traffic are accounted for (TCP)
We ran hping3 with an undefined port of 8080 from our firewalled terminal:

```
[root@DataComm ~]# hping3 192.168.0.11 -S -p 8080 -c 5
HPING 192.168.0.11 (em1 192.168.0.11): S set, 40 headers + 0 data bytes
[send_ip] sendto: Operation not permitted
```

We expect that it will drop these packets because we should be a “good Internet neighbour”. Therefore, our firewall restricts us from being a bad neighbour. However, at least one instance of our “probing” must go through. It is recorded in our tables here:

```
[root@DataComm ~]# iptables -L OUTPUT -n -v -x
Chain OUTPUT (policy DROP 23 packets, 7256 bytes)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 DROP       tcp  --  *      em1     192.168.0.2          0.0.0.0/0           tcp spt:0
       0        0 DROP       udp  --  *      em1     192.168.0.2          0.0.0.0/0           udp spt:0
       0        0 www-traffic  tcp  --  *      em1     0.0.0.0/0            0.0.0.0/0           multiport sports 80,443
     318   173941 www-traffic  tcp  --  *      em1     0.0.0.0/0            0.0.0.0/0           multiport dports 80,443
       0        0 ssh-traffic  tcp  --  *      em1     0.0.0.0/0            0.0.0.0/0           tcp spt:22
       0        0 ssh-traffic  tcp  --  *      em1     0.0.0.0/0            0.0.0.0/0           tcp dpt:22
       1       40 noness-traffic  tcp  --  *      em1     192.168.0.2          0.0.0.0/0           multiport sports  !80,443,22
       0        0 noness-traffic  tcp  --  *      em1     0.0.0.0/0            192.168.0.2         multiport dports  !80,443,22
      26     7474 noness-traffic  udp  --  *      em1     192.168.0.2          0.0.0.0/0           multiport sports  !80,443,22
       0        0 noness-traffic  udp  --  *      em1     0.0.0.0/0            192.168.0.2         multiport dports  !80,443,22
[root@DataComm ~]#
```

Note the one instance of outbound traffic.

(24) All other undefined ports with inbound traffic are accounted for (UDP)
We ran hping3 with an undefined port of 8080 with the --udp switch from our non-firewalled terminal:

```
[root@DataComm ~]# hping3 192.168.0.2 --udp -p 8080 -c 5
HPING 192.168.0.2 (em1 192.168.0.2): udp mode set, 28 headers + 0 data bytes

--- 192.168.0.2 hping statistic ---
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@DataComm ~]#
```

We expect that it will drop these packets but we will see that it is accounted for in our tables:

```
[root@DataComm ~]# iptables -L INPUT -n -v -x
Chain INPUT (policy DROP 143 packets, 18807 bytes)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 ssh-traffic  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp spt:22
    2681   471749 www-traffic  tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           multiport sports 80,443
     158    22691 noness-traffic  all  --  *      *       0.0.0.0/0            0.0.0.0/0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            192.168.0.2         tcp dpt:0
       0        0 DROP       udp  --  em1    *       0.0.0.0/0            192.168.0.2         udp dpt:0
       0        0 DROP       tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           tcp flags:!0x17/0x02 state NEW
       0        0 www-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multiport dports 80,443
       1       64 www-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           multiport sports 80,443
       0        0 ssh-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22
       0        0 ssh-traffic  tcp  --  em1    *       0.0.0.0/0            0.0.0.0/0           tcp spt:22
       0        0 noness-traffic  tcp  --  em1    *       192.168.0.2          0.0.0.0/0           multiport dports  !80,443,22
       5      200 noness-traffic  tcp  --  em1    *       0.0.0.0/0            192.168.0.2         multiport sports  !80,443,22
       0        0 noness-traffic  udp  --  em1    *       192.168.0.2          0.0.0.0/0           multiport dports  !80,443,22
       5      140 noness-traffic  udp  --  em1    *       0.0.0.0/0            192.168.0.2         multiport sports  !80,443,22
[root@DataComm ~]#
```

(25) All other undefined ports with outbound traffic are accounted for (UDP)
We ran hping3 with an undefined port of 8080 with the --udp switch from our firewalled terminal:

```
[root@DataComm ~]# hping3 192.168.0.11 --udp -p 8080 -c 5
HPING 192.168.0.11 (em1 192.168.0.11): udp mode set, 28 headers + 0 data bytes
[send_ip] sendto: Operation not permitted
[root@DataComm ~]#
```

We expect this to happen because we are not being a “good Internet neighbour”. Thus, our firewall blocks our attempt to probe. However, because our UDP rules are universal, we will have to check a before and after screenshot of our tables:

```
[root@DataComm ~]# iptables -L OUTPUT -n -v -x
Chain OUTPUT (policy DROP 51 packets, 15336 bytes)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 DROP       tcp  --  *      em1     192.168.0.2          0.0.0.0/0           tcp spt:0
       0        0 DROP       udp  --  *      em1     192.168.0.2          0.0.0.0/0           udp spt:0
       0        0 www-traffic  tcp  --  *      em1     0.0.0.0/0            0.0.0.0/0           multiport sports 80,443
    2655  1162079 www-traffic  tcp  --  *      em1     0.0.0.0/0            0.0.0.0/0           multiport dports 80,443
       0        0 ssh-traffic  tcp  --  *      em1     0.0.0.0/0            0.0.0.0/0           tcp spt:22
       0        0 ssh-traffic  tcp  --  *      em1     0.0.0.0/0            0.0.0.0/0           tcp dpt:22
       1       40 noness-traffic  tcp  --  *      em1     192.168.0.2          0.0.0.0/0           multiport sports  !80,443,22
       0        0 noness-traffic  tcp  --  *      em1     0.0.0.0/0            192.168.0.2         multiport dports  !80,443,22
      60    15972 noness-traffic  udp  --  *      em1     192.168.0.2          0.0.0.0/0           multiport sports  !80,443,22
       0        0 noness-traffic  udp  --  *      em1     0.0.0.0/0            192.168.0.2         multiport dports  !80,443,22
```

Note the 60…
After running another instance of our hping3 command with the same switches, and then re-running our command to display our accounting tables, note the increment in outbound UDP packets:

```
[root@DataComm ~]# hping3 192.168.0.11 --udp -p 8080 -c 5
HPING 192.168.0.11 (em1 192.168.0.11): udp mode set, 28 headers + 0 data bytes
[send_ip] sendto: Operation not permitted
[root@DataComm ~]# iptables -L OUTPUT -n -v -x
Chain OUTPUT (policy DROP 52 packets, 15364 bytes)
    pkts      bytes target     prot opt in     out     source               destination
       0        0 DROP       tcp  --  *      em1     192.168.0.2          0.0.0.0/0           tcp spt:0
       0        0 DROP       udp  --  *      em1     192.168.0.2          0.0.0.0/0           udp spt:0
       0        0 www-traffic  tcp  --  *      em1     0.0.0.0/0            0.0.0.0/0           multiport sports 80,443
    2662  1164668 www-traffic  tcp  --  *      em1     0.0.0.0/0            0.0.0.0/0           multiport dports 80,443
       0        0 ssh-traffic  tcp  --  *      em1     0.0.0.0/0            0.0.0.0/0           tcp spt:22
       0        0 ssh-traffic  tcp  --  *      em1     0.0.0.0/0            0.0.0.0/0           tcp dpt:22
       1       40 noness-traffic  tcp  --  *      em1     192.168.0.2          0.0.0.0/0           multiport sports  !80,443,22
       0        0 noness-traffic  tcp  --  *      em1     0.0.0.0/0            192.168.0.2         multiport dports  !80,443,22
      61    16000 noness-traffic  udp  --  *      em1     192.168.0.2          0.0.0.0/0           multiport sports  !80,443,22
       0        0 noness-traffic  udp  --  *      em1     0.0.0.0/0            192.168.0.2         multiport dports  !80,443,22
[root@DataComm ~]#
```

Similar to our TCP instance, the firewall must capture at least one before hping3 gets terminated after one instance of probing.

Conclusion

Through the extensive test cases of this assignment, it is sufficient to say that our firewall implementation has covered the relevant criteria of Assignment 1. They are the following:

- Set the default policies to DROP.
- Permit inbound and outbound ssh packets.
- Permit inbound and outbound www packets.
- Drop inbound traffic to port 80 (http) from source ports less than 1024.
- Drop all incoming packets from reserved port 0 as well as outbound traffic to port 0.
- Create a set of user-defined chains that will implement accounting rules to keep track of
- www, ssh traffic, versus the rest of the traffic on your system.
- Use Netfilter for your firewall implementation.
- Ensure the the firewall drops all inbound SYN packets, unless there is a rule that permits inbound traffic.
- Remember to allow DNS and DHCP traffic

We have also included the functionality to check WWW port 443 (https) in our rule set as well.

Appendix

Located on disk are the following:

- Personal Firewalls - An Introduction to Firewall Administration (.pdf)
- Personal Firewalls - Design and Preliminary Testing (.pdf)
- firewall_1.sh
- README.txt