

CSE-343 | Firewall Exploration Lab

Marc Soda Jr

March 30, 2022

Contents

Task 1.A:

- I followed the instructions as specified in the PDF and was successful. The only difference was that I added 'MODULE_LICENSE("GPL");' to hello.c because I got a make error because the license was not specified.
- Code:

```
#include <linux/module.h>
#include <linux/kernel.h>

MODULE_LICENSE("GPL");

int initialization(void) {
    printk(KERN_INFO "Hello World!\n");
    return 0;
}

void cleanup(void) {
    printk(KERN_INFO "Bye-bye World!.\n");
}

module_init(initialization);
module_exit(cleanup);
```

- dmesg output:

```
[1370717.558395] hello: loading out-of-tree module taints kernel.
[1370717.558458] hello: module verification failed: signature and/or required key missing - tainting kernel
[1370717.558730] Hello World!
[1370770.346669] Bye-bye World!.
```

Task 1.B:

Task 1:

- Before adding the module, I am able to dig google.com with no issue. After adding the module, the request is blocked.
- Proof:

```
seed@VM:~/.../packet_filter$ dig @8.8.8.8 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22600
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:: udp: 512
;; QUESTION SECTION:
;www.example.com.                IN      A

;; ANSWER SECTION:
www.example.com.                20525   IN      A      93.184.216.34

;; Query time: 12 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Mar 21 14:21:38 EDT 2022
;; MSG SIZE rcvd: 60

seed@VM:~/.../packet_filter$ sudo insmod seedFilter.
seedFilter.ko      seedFilter.mod.o  seedFilter.o
seed@VM:~/.../packet_filter$ sudo insmod seedFilter.ko
seed@VM:~/.../packet_filter$ lsmod | grep seedF
seedFilter          16384  0
seed@VM:~/.../packet_filter$ !dig
dig @8.8.8.8 www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 www.example.com
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached

seed@VM:~/.../packet_filter$
```

Task 2:

- Code

```

int registerFilter(void) {
    printk(KERN_INFO "PRINT: Registering filters.\n");
    /* NF_INET_PRE_ROUTING */
    hook1.hook = printInfo;
    hook1.hooknum = NF_INET_PRE_ROUTING;
    hook1.pf = PF_INET;
    hook1.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook1);
    /* NF_INET_LOCAL_IN */
    hook2.hook = printInfo;
    hook2.hooknum = NF_INET_LOCAL_IN;
    hook2.pf = PF_INET;
    hook2.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook2);
    /* NF_INET_FORWARD */
    hook3.hook = printInfo;
    hook3.hooknum = NF_INET_FORWARD;
    hook3.pf = PF_INET;
    hook3.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook3);
    /* NF_INET_LOCAL_OUT */
    hook4.hook = printInfo;
    hook4.hooknum = NF_INET_LOCAL_OUT;
    hook4.pf = PF_INET;
    hook4.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook4);
    /* NF_INET_POST_ROUTING */
    hook5.hook = printInfo;
    hook5.hooknum = NF_INET_POST_ROUTING;
    hook5.pf = PF_INET;
    hook5.priority = NF_IP_PRI_FIRST;
    nf_register_net_hook(&init_net, &hook5);
    return 0;
}

void removeFilter(void) {
    printk(KERN_INFO "PRINT: The filters are being removed.\n");
    nf_unregister_net_hook(&init_net, &hook1);
    nf_unregister_net_hook(&init_net, &hook2);
    nf_unregister_net_hook(&init_net, &hook3);
    nf_unregister_net_hook(&init_net, &hook4);
    nf_unregister_net_hook(&init_net, &hook5);
}

```

<N> seedPrint/seedPrint.c 108 58%

VM 1: bash- 2: emacs* 3: docker-compose 4: bash

- After adding the module, I can the information being printed (through dmesg) corresponding to each hook.


```

[2090753.856324] *** LOCAL_OUT
[2090753.856326] 128.180.123.88 --> 128.180.6.102 (TCP)
[2090753.856328] *** LOCAL_OUT
[2090753.856328] 128.180.123.88 --> 128.180.6.102 (TCP)
[2090753.856331] *** POST_ROUTING
[2090753.856332] 128.180.123.88 --> 128.180.6.102 (TCP)
[2090753.856797] *** PRE_ROUTING
[2090753.856799] 128.180.6.102 --> 128.180.123.88 (TCP)
[2090753.856803] *** LOCAL_IN
[2090753.856803] 128.180.6.102 --> 128.180.123.88 (TCP)
[2090753.860402] *** LOCAL_OUT
[2090753.860404] 128.180.123.88 --> 128.180.6.102 (TCP)
[2090753.860405] *** LOCAL_OUT
[2090753.860406] 128.180.123.88 --> 128.180.6.102 (TCP)
[2090753.860408] *** POST_ROUTING
[2090753.860409] 128.180.123.88 --> 128.180.6.102 (TCP)
[2090753.860827] *** PRE_ROUTING
[2090753.860842] 128.180.6.102 --> 128.180.123.88 (TCP)

```

Task 3:

- blockPing and blockTelnet code:

```

unsigned int blockPing(void *priv, struct sk_buff *skb,
                      const struct nf_hook_state *state)
{
    struct iphdr *iph;
    struct icmphdr *icmph;

    char ip[16] = "10.9.0.1";
    u32 ip_addr;

    if (!skb) return NF_ACCEPT;

    iph = ip_hdr(skb);
    // Convert the IPv4 address from dotted decimal to 32-bit binary
    in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);

    if (iph->protocol == IPPROTO_ICMP) {
        icmph = icmp_hdr(skb);
        if (iph->daddr == ip_addr && icmph->type == ICMP_ECHO){
            printk(KERN_WARNING "*** Dropping %pI4 (ICMP)\n", &(iph->daddr));
            return NF_DROP;
        }
    }
    return NF_ACCEPT;
}

unsigned int blockTel(void *priv, struct sk_buff *skb,
                     const struct nf_hook_state *state)
{
    struct iphdr *iph;
    struct tcphdr *tcph;

    u16 port = 23;
    char ip[16] = "10.9.0.1";
    u32 ip_addr;

    if (!skb) return NF_ACCEPT;

    iph = ip_hdr(skb);
    // Convert the IPv4 address from dotted decimal to 32-bit binary
    in4_pton(ip, -1, (u8 *)&ip_addr, '\0', NULL);

    if (iph->protocol == IPPROTO_TCP) {
        tcph = tcp_hdr(skb);
        if (iph->daddr == ip_addr && ntohs(tcph->dest) == port){
            printk(KERN_WARNING "*** Dropping %pI4 (TCP), port %d\n", &(iph->daddr), port);
            return NF_DROP;
        }
    }
    return NF_ACCEPT;
}

unsigned int printInfo(void *priv, struct sk_buff *skb,

```

- The following screenshot shows two shells. One shows HostA running telnet and ping. The other is a filtered cat of syslog showing which packets were dropped as a result of this task. As you can see, the new kernel module is blocking telnet and icmp packets. The reason why I could not show dmesg output is because when I do 'dmesg -k -w', the output scrolls much too fast to read. I believe this is because I am connected to my seed server over ssh.

```

root@lec4f67a6c7b:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
--- 10.9.0.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2030ms

^Croot@lec4f67a6c7b:/# telnet 10.9.0.1
Trying 10.9.0.1...
^C
root@lec4f67a6c7b:/#

Mar 29 22:41:43 VM kernel: [2093178.532783] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:41:46 VM kernel: [2093182.059924] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:41:47 VM kernel: [2093183.076603] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:41:48 VM kernel: [2093184.100617] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:41:49 VM kernel: [2093185.124595] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:41:50 VM kernel: [2093186.148478] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:41:51 VM kernel: [2093187.172630] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:41:52 VM kernel: [2093188.196487] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:41:53 VM kernel: [2093189.220399] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:41:54 VM kernel: [2093190.248512] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:41:55 VM kernel: [2093191.268338] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:41:56 VM kernel: [2093192.292297] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:41:57 VM kernel: [2093193.316422] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:41:58 VM kernel: [2093194.340288] *** Dropping 10.9.0.1 (UDP)
Mar 29 22:42:24 VM kernel: [2093220.095972] *** Dropping 10.9.0.1 (UDP), port 23
Mar 29 22:42:25 VM kernel: [2093221.123647] *** Dropping 10.9.0.1 (UDP), port 23
Mar 29 22:45:16 VM kernel: [2093392.048936] *** Dropping 10.9.0.1 (ICMP)
Mar 29 22:45:17 VM kernel: [2093393.055189] *** Dropping 10.9.0.1 (ICMP)
Mar 29 22:45:18 VM kernel: [2093394.079038] *** Dropping 10.9.0.1 (ICMP)
Mar 29 22:45:22 VM kernel: [2093397.356639] *** Dropping 10.9.0.1 (TCP), port 23
Mar 29 22:45:23 VM kernel: [2093398.366934] *** Dropping 10.9.0.1 (TCP), port 23
Mar 29 22:45:25 VM kernel: [2093400.382860] *** Dropping 10.9.0.1 (TCP), port 23

seed@VM:~$
VM 1:bash* 2:emacs- 3:docker-compose

```

Task 2:

Task 2A:

- Before manipulating the IP tables I can ping and telnet into the router (10.9.0.11) just fine.
- After manipulating the IP tables with the specified command, I can ping the router, but I can't telnet into it.
- These rules drop ICMP packets except for PING.

Task 2B:

- Commands (see screenshot for iptables rules output):
 - iptables -A FORWARD -i eth0 -p icmp –icmp-type echo-request -j DROP
 - iptables -A FORWARD -i eth1 -p icmp –icmp-type echo-request -j ACCEPT
 - iptables -A FORWARD -i eth0 -p icmp –icmp-type echo-reply -j ACCEPT
 - iptables -P FORWARD DROP
- Observations (see screenshot for proof):
 - HostA (outside) cannot ping Host1 (inside).
 - HostA (outside) can ping the router.
 - HostA (outside) cannot telnet Host1 (inside).
 - Host1 (inside) can ping HostA (outside)
 - Host1 (inside) cannot telnet HostA (outside)
- Screenshot
 - Top right is Host1. Bottom left is router. Bottom right is HostA. Hostnames reflected in prompt.

```

seedVM:~/.../seedBlock$

ROUTER > iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out    source            destination

Chain FORWARD (policy DROP 7 packets, 420 bytes)
pkts bytes target      prot opt in     out    source            destination
  3   252 DROP        icmp -- eth0   *      0.0.0.0/0         0.0.0.0/0
      icmp type 8
  6   504 ACCEPT     icmp -- eth1   *      0.0.0.0/0         0.0.0.0/0
      icmp type 8
  6   504 ACCEPT     icmp -- eth0   *      0.0.0.0/0         0.0.0.0/0
      icmp type 0

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

ROUTER >

HOST-1 > ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data:
64 bytes from 10.9.0.5: icmp_seq=1 ttl=63 time=0.147 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=63 time=0.060 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=63 time=0.059 ms
^C
--- 10.9.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2038ms
rtt min/avg/max/mdev = 0.059/0.088/0.147/0.041 ms
HOST-1 > telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
HOST-1 >
HOST-1 >

HOST-A > ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data:
--- 192.168.60.5 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5098ms
HOST-A > telnet 192.168.60.5
Trying 192.168.60.5...
^C
HOST-A >
HOST-A > ping 192.168.60.11
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data:
64 bytes from 192.168.60.11: icmp_seq=1 ttl=64 time=0.085 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=64 time=0.074 ms
64 bytes from 192.168.60.11: icmp_seq=3 ttl=64 time=0.050 ms
64 bytes from 192.168.60.11: icmp_seq=4 ttl=64 time=0.063 ms
^C
--- 192.168.60.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3050ms
rtt min/avg/max/mdev = 0.050/0.068/0.085/0.013 ms
HOST-A >

```

Task 2C:

- Commands (see screenshot for iptables rules output)
 - iptables -A FORWARD -i eth0 -p tcp -d 192.160.60.5 –dport 23 -j ACCEPT
 - iptables -A FORWARD -i eth1 -p tcp -s 192.160.60.5 –sport 23 -j ACCEPT
 - iptables -P FORWARD DROP
 - iptables -A FORWARD -i eth0 -p tcp -sport 5000 -j ACCEPT
- Observations (see screenshot for proof)
 - Host1 (inside) is able to telnet into other hosts on the internal network.
 - HostA (outside) is able to telnet to Host1 (inside), but not to any other internal hosts.
 - Host1 (inside) is unable to telnet to HostA (outside).
 - Note that other internal and external hosts were tested and the specifications were met. See screenshot.


```

seed@VM:~/../seedBlock$
HOST-1 > telnet 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
d44104f23e88 login: ^CConnection closed by foreign host.
HOST-1 > telnet 10.9.0.5
Trying 10.9.0.5...
^C
HOST-1 >

ROUTER > iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source            destination

Chain FORWARD (policy DROP 15 packets, 900 bytes)
pkts bytes target      prot opt in     out     source            destination
19  1073 ACCEPT     tcp  --  eth0   *      0.0.0.0/0         192.168.60.5      tcp dpt:23
19  1091 ACCEPT     tcp  --  eth1   *      192.168.60.5      0.0.0.0/0         tcp spt:23

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

ROUTER >
HOST-A > telnet 192.168.60.6
Trying 192.168.60.6...
^C
HOST-A > telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
51c62151c92e login: ^CConnection closed by foreign host.
HOST-A >

```

Task 3:

Task 3A:

ICMP Experiment:

- After pinging Host1 from the router, conntrack -L shows details about the connection that was made.
- The ICMP connection state was kept for 30 seconds.

UDP Experiment:

- After communicating to Host1 from the router, conntrack -L shows details about the connection that was made.
- The UDP connection state was kept for 30 seconds.

TCP Experiment:

- After communicating from Host1 from the router, conntrack -L shows details about the connection that was made.
- The TCP connection state was kept for 120 seconds.

Task 3B:

Commands:

- iptables -A FORWARD -p tcp -i eth0 -d 192.168.60.5 -dport 23 -syn -m conntrack -ctstate NEW -j ACCEPT
- iptables -A FORWARD -p tcp -i eth1 -syn -m conntrack -ctstate NEW -j ACCEPT
- iptables -A FORWARD -p tcp -m conntrack -ctstate ESTABLISHED,RELATED -j ACCEPT

- iptables -A FORWARD -p tcp -j DROP
- iptables -A FORWARD ACCEPT

Observations (see screenshot):

- HostA (outside) can telnet to Host1 (inside)
- Host1 (inside) can telnet to HostA (outside)
- These patterns are maintained for other internal and external hosts.

```
seed@VM:~/seedBlock$
ROUTER > iptables -A FORWARD -p tcp -i eth0 -d 192.168.60.5 --dport 23 --syn -m conntrack --ctstate NEW -j ACCEPT
ROUTER > iptables -A FORWARD -p tcp -i eth1 --syn -m conntrack --ctstate NEW -j ACCEPT
ROUTER > iptables -A FORWARD -p tcp -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
ROUTER > iptables -A FORWARD -p tcp -j DROP
ROUTER > iptables -A FORWARD -j ACCEPT
ROUTER > iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source            destination
0      0 ACCEPT     tcp  --  eth0   *      0.0.0.0/0         192.168.60.5      tcp dpt:23 flags:0x17/0x02 ctstate NEW
0      0 ACCEPT     tcp  --  eth1   *      0.0.0.0/0         0.0.0.0/0         tcp flags:0x17/0x02 ctstate NEW
0      0 ACCEPT     tcp  --  *      *      0.0.0.0/0         0.0.0.0/0         ctstate RELATED,ESTABLISHED
0      0 DROP       tcp  --  *      *      0.0.0.0/0         0.0.0.0/0
0      0 ACCEPT     tcp  --  eth0   *      0.0.0.0/0         192.168.60.5      tcp dpt:23 flags:0x17/0x02 ctstate NEW
0      0 ACCEPT     tcp  --  eth1   *      0.0.0.0/0         0.0.0.0/0         tcp flags:0x17/0x02 ctstate NEW
0      0 ACCEPT     tcp  --  *      *      0.0.0.0/0         0.0.0.0/0         ctstate RELATED,ESTABLISHED
0      0 DROP       tcp  --  *      *      0.0.0.0/0         0.0.0.0/0
0      0 ACCEPT     all  --  *      *      0.0.0.0/0         0.0.0.0/0
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source            destination
ROUTER >
HOST-1 > telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
1ec4f67a6c7b login: Connection closed by foreign host.
HOST-1 >
HOST-A > telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
51c62151c92e login: ^CConnection closed by foreign host.
HOST-A >
```

Conntrack advantages:

- Consumes less CPU because caching

Conntrack disadvantages:

- Consumes more memory because connection states need to be saved for a certain amount of time.
- Can be poor at handling a high volume of connections per second.

Task 4:

- After running the first command only, pinging 192.168.60.5 is unaffected.
- After adding the second rule, the following behavior is observed:
 - The first 5 pings go through as normal.
 - Pings then begin being blocked due to the 5 connection burst limit and the 10/minute limit. I ended up getting a 67% packet loss after 10 successful pings.

Task 5:

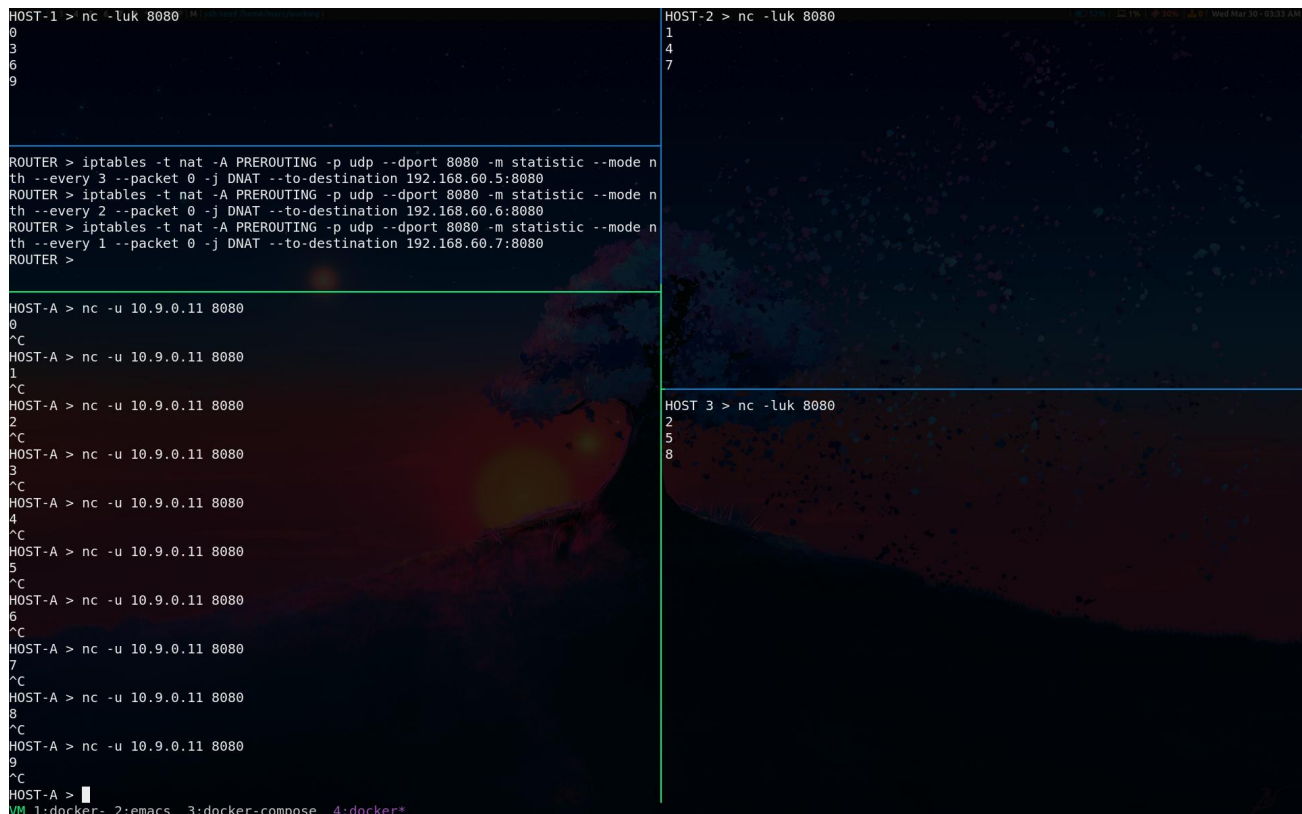
Round Robin Mode

Commands:

- `iptables -t nat -A PREROUTING -p udp -dport 8080 -m statistic --mode nth --every 3 --packet 0 -j DNAT --to-destination 192.168.60.5:8080`
- `iptables -t nat -A PREROUTING -p udp -dport 8080 -m statistic --mode nth --every 2 --packet 0 -j DNAT --to-destination 192.168.60.6:8080`
- `iptables -t nat -A PREROUTING -p udp -dport 8080 -m statistic --mode nth --every 1 --packet 0 -j DNAT --to-destination 192.168.60.7:8080`
- Each rule chooses which number packet (out of three) to send to each different server.

Observations

- After adding the rules, I can make a connection and send a message from HostA to the router. Each time a connection is made, it is forwarded to a different server on the network. First Host1, then Host2, then Host3, then repeat. The load balancing task was successful. See screenshot.



```
HOST-1 > nc -luk 8080
0
3
6
9

ROUTER > iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 3 --packet 0 -j DNAT --to-destination 192.168.60.5:8080
ROUTER > iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 2 --packet 0 -j DNAT --to-destination 192.168.60.6:8080
ROUTER > iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode nth --every 1 --packet 0 -j DNAT --to-destination 192.168.60.7:8080
ROUTER >

HOST-A > nc -u 10.9.0.11 8080
0
^C
HOST-A > nc -u 10.9.0.11 8080
1
^C
HOST-A > nc -u 10.9.0.11 8080
2
^C
HOST-A > nc -u 10.9.0.11 8080
3
^C
HOST-A > nc -u 10.9.0.11 8080
4
^C
HOST-A > nc -u 10.9.0.11 8080
5
^C
HOST-A > nc -u 10.9.0.11 8080
6
^C
HOST-A > nc -u 10.9.0.11 8080
7
^C
HOST-A > nc -u 10.9.0.11 8080
8
^C
HOST-A > nc -u 10.9.0.11 8080
9
^C
HOST-A >
VM 1:docker- 2:emacs 3:docker-compose 4:docker*
```

Random Mode

Commands

- `iptables -t nat -A PREROUTING -p udp -dport 8080 -m statistic --mode random --probability .3333 -j DNAT --to-destination 192.168.60.5:8080`
- `iptables -t nat -A PREROUTING -p udp -dport 8080 -m statistic --mode random --probability .5 -j DNAT --to-destination 192.168.60.6:8080`

- `iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode random --probability 1 -j DNAT --to-destination 192.168.60.7:8080`
- Apparently making the probability for each rule .3333 is wrong. When I did that I found that the connection would get dropped sometimes. I had to look it up and apparently you are supposed to make the first one .3333, the second .5, and the third 1. This has something to do with the fact that the rules are executed sequentially. With this setup, each host has a 33% chance of being selected.

Observations

```

HOST-1 > nc -luk 8080
0
1
2
3
4
5
6
7
8
9

ROUTER > iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode r
andom --probability .3333 -j DNAT --to-destination 192.168.60.5:8080
ROUTER > iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode r
andom --probability .5 -j DNAT --to-destination 192.168.60.6:8080
ROUTER > iptables -t nat -A PREROUTING -p udp --dport 8080 -m statistic --mode r
andom --probability 1 -j DNAT --to-destination 192.168.60.7:8080
ROUTER >

HOST-A > nc -u 10.9.0.11 8080
0
^C
HOST-A > nc -u 10.9.0.11 8080
1
^C
HOST-A > nc -u 10.9.0.11 8080
2
^C
HOST-A > nc -u 10.9.0.11 8080
3
^C
HOST-A > nc -u 10.9.0.11 8080
4
^C
HOST-A > nc -u 10.9.0.11 8080
5
^C
HOST-A > nc -u 10.9.0.11 8080
6
^C

VM 1:docker- 2:emacs 3:docker-compose 4:docker*

HOST-2 > nc -luk 8080
3
1
5
6
9
8
9
1
2
4
8
7

HOST 3 > nc -luk 8080
1
5
7
8
9
3
4
8
0
1
4
5
7
0
5

```