Networks and Systems Security
Exercise 1

Name: Harsh Kumar Roll Number: 2019043

1.

a. The boolean flag, to enable forwarding is written in */proc/sys/net/ipv4/ip_forward* file. We can enable forwarding by changing the value of the flag.

```
[artix2 ~]# echo 1 >> /proc/sys/net/ipv4/ip_forward
[artix2 ~]# cat /proc/sys/net/ipv4/ip_forward
1
[artix2 ~]# _
```

When we send a request to 20.0.0.2 (eth0 on vm3) from 10.0.0.1 (eth0 on vm1), the request is first send to 10.0.0.1, which is redirected to 10.0.0.2. Thus, we see two hops when we run traceroute on vm1.

```
[artix1 ~]# traceroute 20.0.0.2
traceroute to 20.0.0.2 (20.0.0.2), 30 hops max, 60 byte packets
 1  10.0.0.2 (10.0.0.2)  4.247 ms  3.890 ms  3.724 ms
 2  20.0.0.2 (20.0.0.2)  3.942 ms  3.606 ms  3.325 ms
[artix1 ~]#
```

b.

```
[artix2 ~]# iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

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

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
[artix2 ~]# iptables --table nat --append POSTROUTING --out-interface eth0 -j MASQUERADE
[artix2 ~]# iptables --append FORWARD --in-interface eth1 -j ACCEPT
[artix2 ~]# iptables --table nat --append POSTROUTING --out-interface eth1 -j MASQUERADE
[artix2 ~]# iptables --append FORWARD --in-interface eth0 -j ACCEPT
[artix2 ~]# iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  eth1   *       0.0.0.0/0            0.0.0.0/0
    0     0 ACCEPT     all  --  eth0   *       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
[artix2 ~]# _
```

```
[artix2 ~]# iptables -L -n -v -t nat
Chain PREROUTING (policy ACCEPT 21 packets, 1380 bytes)
 pkts bytes target     prot opt in     out     source               destination

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

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

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  all  --  *      eth0    0.0.0.0/0            0.0.0.0/0
   18  1200 MASQUERADE  all  --  *      eth1    0.0.0.0/0            0.0.0.0/0
```

Right now, we are forwarding all packets from vm1 to vm3, and vice versa. Now, we would like to filter packets and accept only those packets that are sent on ports 80 and 443. For that, we change the iptables FORWARD chain. Also, we would block any packets generated at vm2. The modified iptables looks like this:

```
[artix2 ~]# 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 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   21  1308 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80
    4   240 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:443
    7   704 ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED

Chain OUTPUT (policy DROP 7 packets, 540 bytes)
 pkts bytes target     prot opt in     out     source               destination
[artix2 ~]# _
```

Commands executed for getting this iptable configuration:

```
iptables -P FORWARD DROP
iptables -A FORWARD -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -p tcp --dport 443 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED, RELATED -j ACCEPT
iptables -P OUTPUT DROP
```

c.

I'm sending an HTTP request packet from vm1 to vm3. Below are the output of tcpdump which monitors the packet exchange on all these vms:

vm1:

```
[artix1 ~]# cat /mnt/hgfs/shared/task1/a1.pcap
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
12:39:22.905754 IP 10.0.0.1.49086 > 20.0.0.2.http: Flags [S], seq 3431481268, win 64240, options [mss 1460,sackOK,TS val 1191012
434 ecr 0,nop,wscale 7], length 0
12:39:22.906767 IP 20.0.0.2.http > 10.0.0.1.49086: Flags [S.], seq 376046717, ack 3431481269, win 65160, options [mss 1460,sackO
K,TS val 3382830020 ecr 1191012434,nop,wscale 7], length 0
12:39:22.906800 IP 10.0.0.1.49086 > 20.0.0.2.http: Flags [.], ack 1, win 502, options [nop,nop,TS val 1191012435 ecr 3382830020]
, length 0
12:39:22.906956 IP 10.0.0.1.49086 > 20.0.0.2.http: Flags [P.], seq 1:73, ack 1, win 502, options [nop,nop,TS val 1191012435 ecr
3382830020], length 72: HTTP: GET / HTTP/1.1
12:39:22.907735 IP 20.0.0.2.http > 10.0.0.1.49086: Flags [.], ack 73, win 509, options [nop,nop,TS val 3382830021 ecr 1191012435
], length 0
12:39:22.907930 IP 20.0.0.2.http > 10.0.0.1.49086: Flags [P.], seq 1:238, ack 73, win 509, options [nop,nop,TS val 3382830021 ec
r 1191012435], length 237: HTTP: HTTP/1.1 200 OK
12:39:22.907941 IP 10.0.0.1.49086 > 20.0.0.2.http: Flags [.], ack 238, win 501, options [nop,nop,TS val 1191012436 ecr 338283002
1], length 0
12:39:22.908078 IP 20.0.0.2.http > 10.0.0.1.49086: Flags [P.], seq 238:357, ack 73, win 509, options [nop,nop,TS val 3382830021
ecr 1191012435], length 119: HTTP
12:39:22.908088 IP 10.0.0.1.49086 > 20.0.0.2.http: Flags [.], ack 357, win 501, options [nop,nop,TS val 1191012436 ecr 338283002
1], length 0
12:39:22.917362 IP 10.0.0.1.49086 > 20.0.0.2.http: Flags [F.], seq 73, ack 357, win 501, options [nop,nop,TS val 1191012445 ecr
33828300211, length 0
```

Here, we can see that it appears to vm1 that it is directly communicating with vm3. All the requests/responses are between 10.0.0.1 (eth0 on vm1) and 20.0.0.2 (eth0 on vm3). Finally, we also get an OK HTTP response from the server.

The response is displayed by curl:

```
[artix1 ~]# curl 20.0.0.2
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <p>This is a test website.</p>
  </body>
</html>
```

vm2 (eth0):

```
[artix2 ~]# tcpdump -p
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
12:35:29.857051 IP 10.0.0.1.49086 > 20.0.0.2.http: Flags [S], seq 3431481268, win 64240, options [mss 1460,sackOK,TS val 1191012
434 ecr 0,nop,wscale 7], length 0
12:35:29.857579 IP 20.0.0.2.http > 10.0.0.1.49086: Flags [S.], seq 376046717, ack 3431481269, win 65160, options [mss 1460,sackO
K,TS val 3382830020 ecr 1191012434,nop,wscale 7], length 0
12:35:29.857976 IP 10.0.0.1.49086 > 20.0.0.2.http: Flags [.], ack 1, win 502, options [nop,nop,TS val 1191012435 ecr 3382830020]
, length 0
12:35:29.858140 IP 10.0.0.1.49086 > 20.0.0.2.http: Flags [P.], seq 1:73, ack 1, win 502, options [nop,nop,TS val 1191012435 ecr
3382830020], length 72: HTTP: GET / HTTP/1.1
12:35:29.858585 IP 20.0.0.2.http > 10.0.0.1.49086: Flags [.], ack 73, win 509, options [nop,nop,TS val 3382830021 ecr 1191012435
], length 0
12:35:29.858779 IP 20.0.0.2.http > 10.0.0.1.49086: Flags [P.], seq 1:238, ack 73, win 509, options [nop,nop,TS val 3382830021 ec
r 1191012435], length 237: HTTP: HTTP/1.1 200 OK
12:35:29.858889 IP 20.0.0.2.http > 10.0.0.1.49086: Flags [P.], seq 238:357, ack 73, win 509, options [nop,nop,TS val 3382830021
ecr 1191012435], length 119: HTTP
12:35:29.859117 IP 10.0.0.1.49086 > 20.0.0.2.http: Flags [.], ack 238, win 501, options [nop,nop,TS val 1191012436 ecr 338283002
1], length 0
12:35:29.859237 IP 10.0.0.1.49086 > 20.0.0.2.http: Flags [.], ack 357, win 501, options [nop,nop,TS val 1191012436 ecr 338283002
1], length 0
12:35:29.868624 IP 10.0.0.1.49086 > 20.0.0.2.http: Flags [F.], seq 73, ack 357, win 501, options [nop,nop,TS val 1191012445 ecr
33828300211, length 0
12:35:29.869105 IP 20.0.0.2.http > 10.0.0.1.49086: Flags [F.], seq 357, ack 74, win 509, options [nop,nop,TS val 3382830031 ecr
1191012445], length 0
12:35:29.869452 IP 10.0.0.1.49086 > 20.0.0.2.http: Flags [.], ack 358, win 501, options [nop,nop,TS val 1191012446 ecr 338283003
1], length 0
```

vm2 (eth1):

Note: I have requested HTTP packet twice from vm3 for capturing activity on both the ports of vm2. Here the time and port number are different because this is the packet capture for 2nd interaction.

```
[artix2 ~]# tcpdump -p -i eth1
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
12:59:26.496442 IP 20.0.0.1.49094 > 20.0.0.2.http: Flags [S], seq 3458735978, win 64240, options [mss 1460,sackOK,TS val 1192449
073 ecr 0,nop,wscale 7], length 0
12:59:26.497549 IP 20.0.0.2.http > 20.0.0.1.49094: Flags [S.], seq 2636607439, ack 3458735979, win 65160, options [mss 1460,sack
OK,TS val 3384266659 ecr 1192449073,nop,wscale 7], length 0
12:59:26.498499 IP 20.0.0.1.49094 > 20.0.0.2.http: Flags [.], ack 1, win 502, options [nop,nop,TS val 1192449075 ecr 3384266659]
, length 0
12:59:26.498715 IP 20.0.0.1.49094 > 20.0.0.2.http: Flags [P.], seq 1:73, ack 1, win 502, options [nop,nop,TS val 1192449075 ecr
3384266659], length 72: HTTP: GET / HTTP/1.1
12:59:26.499739 IP 20.0.0.2.http > 20.0.0.1.49094: Flags [.], ack 73, win 509, options [nop,nop,TS val 3384266662 ecr 1192449075
], length 0
12:59:26.500056 IP 20.0.0.2.http > 20.0.0.1.49094: Flags [P.], seq 1:238, ack 73, win 509, options [nop,nop,TS val 3384266662 ec
r 1192449075], length 237: HTTP: HTTP/1.1 200 OK
12:59:26.500357 IP 20.0.0.2.http > 20.0.0.1.49094: Flags [P.], seq 238:357, ack 73, win 509, options [nop,nop,TS val 3384266662
ecr 1192449075], length 119: HTTP
12:59:26.500729 IP 20.0.0.1.49094 > 20.0.0.2.http: Flags [.], ack 238, win 501, options [nop,nop,TS val 1192449077 ecr 338426666
2], length 0
12:59:26.500972 IP 20.0.0.1.49094 > 20.0.0.2.http: Flags [.], ack 357, win 501, options [nop,nop,TS val 1192449078 ecr 338426666
2], length 0
12:59:26.515335 IP 20.0.0.1.49094 > 20.0.0.2.http: Flags [F.], seq 73, ack 357, win 501, options [nop,nop,TS val 1192449092 ecr
3384266662], length 0
12:59:26.515919 IP 20.0.0.2.http > 20.0.0.1.49094: Flags [F.], seq 357, ack 74, win 509, options [nop,nop,TS val 3384266678 ecr
1192449092], length 0
12:59:26.516381 IP 20.0.0.1.49094 > 20.0.0.2.http: Flags [.], ack 358, win 501, options [nop,nop,TS val 1192449093 ecr 338426667
8], length 0
```

In vm2, we see that addresses are being translated. A temporary port for communication between vm1 and vm3 is being allocated on vm2.

vm3:

```
[artix3 html]# tcpdump -p
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
05:09:09.546813 IP 20.0.0.1.49086 > 20.0.0.2.http: Flags [S], seq 3431481268, win 64240, options [mss 1460,sackOK,TS val 1191012
434 ecr 0,nop,wscale 7], length 0
05:09:09.546887 IP 20.0.0.2.http > 20.0.0.1.49086: Flags [S.], seq 376046717, ack 3431481269, win 65160, options [mss 1460,sackO
K,TS val 3382830020 ecr 1191012434,nop,wscale 7], length 0
05:09:09.547677 IP 20.0.0.1.49086 > 20.0.0.2.http: Flags [.], ack 1, win 502, options [nop,nop,TS val 1191012435 ecr 3382830020]
, length 0
05:09:09.547876 IP 20.0.0.1.49086 > 20.0.0.2.http: Flags [P.], seq 1:73, ack 1, win 502, options [nop,nop,TS val 1191012435 ecr
3382830020], length 72: HTTP: GET / HTTP/1.1
05:09:09.547893 IP 20.0.0.2.http > 20.0.0.1.49086: Flags [.], ack 73, win 509, options [nop,nop,TS val 3382830021 ecr 1191012435
], length 0
05:09:09.548091 IP 20.0.0.2.http > 20.0.0.1.49086: Flags [P.], seq 1:238, ack 73, win 509, options [nop,nop,TS val 3382830021 ec
r 1191012435], length 237: HTTP: HTTP/1.1 200 OK
05:09:09.548229 IP 20.0.0.2.http > 20.0.0.1.49086: Flags [P.], seq 238:357, ack 73, win 509, options [nop,nop,TS val 3382830021
ecr 1191012435], length 119: HTTP
05:09:09.548801 IP 20.0.0.1.49086 > 20.0.0.2.http: Flags [.], ack 238, win 501, options [nop,nop,TS val 1191012436 ecr 338283002
1], length 0
05:09:09.548936 IP 20.0.0.1.49086 > 20.0.0.2.http: Flags [.], ack 357, win 501, options [nop,nop,TS val 1191012436 ecr 338283002
1], length 0
05:09:09.558342 IP 20.0.0.1.49086 > 20.0.0.2.http: Flags [F.], seq 73, ack 357, win 501, options [nop,nop,TS val 1191012445 ecr
3382830021], length 0
05:09:09.558421 IP 20.0.0.2.http > 20.0.0.1.49086: Flags [F.], seq 357, ack 74, win 509, options [nop,nop,TS val 3382830031 ecr
1191012445], length 0
05:09:09.559159 IP 20.0.0.1.49086 > 20.0.0.2.http: Flags [.], ack 358, win 501, options [nop,nop,TS val 1191012446 ecr 338283003
1], length 0
```

In vm3, we observe that it looks like all the traffic is coming from 20.0.0.1 (eth1 on vm2). The web server machine is unaware of the actual producer of this traffic, which is sitting in front of the firewall. All the responses are sent back to the firewall machine, which then returns the response to the actual user.

2.

a. Nginx binary is owned by root. So on setting *setuid* bit, the executor of the program should get privileges of root. However, on closer inquiry, I found that the nginx worker process responsible for reading web server files is run by a special user, *http*.

```
[artix3 nginx]# ps -aux | grep nginx
root         831  0.0  0.1   2252   736 ?        Ss   08:43   0:00 runsv nginx
root         851  0.0  0.9   9268  6404 ?        S    08:43   0:00 nginx: master process nginx -g daemon off;
http         858  0.0  0.3   9764  2756 ?        S    08:43   0:00 nginx: worker process
root        3665  0.0  0.2   3260  1644 tty1     S+   09:30   0:00 grep --colour=auto nginx
```

So, even though the setuid bit of nginx binary is set, the nginx worker process is run by *http* user. I was not able to figure out the proper reason for it. But, we can observe that the worker process will not be able to access the webserver pages. The HTTP request fails and returns a 403 (Forbidden) response.

I also observed that on changing the owner of the webserver directory to *http* user, we can actually access the website with 200 (OK) response from the server.

b.

```
[temphttp@artix3 nginx]$ sudo chown -hR temphttp html
[temphttp@artix3 nginx]$ chmod 700 html
[temphttp@artix3 nginx]$ ls -l
total 4
drwx------ 2 temphttp root 4096 Feb  7 03:45 html
```

```
[temphttp@artix3 sbin]$ ls -l | grep nginx
-rwxr-xr-x 1 root root 1524728 Nov 28 21:48 nginx
[temphttp@artix3 sbin]$ sudo chmod 4755 nginx
[temphttp@artix3 sbin]$ ls -l | grep nginx
-rwsr-xr-x 1 root root 1524728 Nov 28 21:48 nginx
```

```
[hadron@artix3 ~]$ curl 20.0.0.2
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.20.2</center>
</body>
</html>
```

c. We can explicitly set extra permissions using ACL for *http* user to read, write, execute. After this, we can access the webserver. The reason is, the service worked responsible for reading the webserver files is run by http user.

Note: For ACL to work, we must mount our partition with acl option first. I added that in *fstab*, and rebooted my system.

```
[artix3 nginx]# setfacl -m u:http:rwx html
[artix3 nginx]# getfacl html
# file: html
# owner: temphttp
# group: root
# flags: s--
user::rwx
user:http:rwx
group::---
mask::rwx
other::---

[artix3 nginx]# cd html/
[artix3 html]# ls -l
total 4
-rw-rw-rw-+ 1 root root 119 Feb  7 03:46 index.html
[artix3 html]# curl 20.0.0.2
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <p>This is a test website.</p>
  </body>
</html>
[artix3 html]#
```