CSE354/CSE554: Networks and Systems Security II

Exercise 1 Write-up

Madhav Kansil (2022270)

1. Set Up

In this report, I will demonstrate what all commands have been used to set up the VM configuration required by the exercise and the corresponding screenshots for the same.

The virtualization software used for this exercise is `Oracle Virtual Box`. All the VMs run `FreeBSD 14.2-RELEASE amd64`. `VM_1` and `VM_2` are on the same internal network *net_1* , similarly `VM_2` and `VM_3` are on the same internal network *net_2*. All machines also have a interface `em1` as host adapter. The following table shows the network configuration.

Machine	IP	Interface Name	Type
VM_1	10.0.2.15	em0	NAT
VM_1	192.168.1.11	em2	Internal *net_1*
VM_2	10.0.2.15	em0	NAT
VM_2	192.168.1.1	em2	Internal *net_1*
VM_2	192.168.2.2	em3	Internal *net_2*
VM_3	10.0.2.15	em0	NAT
VM_3	192.168.2.22	em2	Internal *net_2*

1.1. Commands

To assign static IPs to the interfaces of the internal network and to set destination for subnet `192.168.1.0/24` and `192.168.2.0/24` the following lines were added to `/etc/rc.conf`. Also adding a ip forwarding to `VM_2`.

```
ifconfig_<interface>="inet <ip> netmask 255.255.255.0"
```

```
# adding routes in vm1 (net_1 to net_2)
static_routes="net_1"

route_net_1="-net 192.168.2.0/24 192.168.1.1"

# adding routes in vm2 (net_2 to net_1)
static_routes="net_2"

route_net_2="-net 192.168.1.0/24 192.168.2.2"

# adding ip forwarding in vm2
gateway_enable="YES"
```

```
root@vm_1:~ # traceroute 192.168.2.22
traceroute to 192.168.2.22 (192.168.2.22), 64 hops max, 40 byte packets
 1  192.168.1.1 (192.168.1.1)  0.001 ms  0.517 ms  0.420 ms
 2  192.168.2.22 (192.168.2.22)  1.055 ms  1.144 ms  1.129 ms
root@vm_1:~ #
```

Figure 1. Traceroute

2. Enabling PF

To enable pf in FreeBSD we add `pf_enable="YES"` and `pflog_enable="YES"` to `/etc/rc.conf`. The whole pf configuration can be viewed in the Appendix Section. The commands used include `binat` for DNAT-ing and SNATing. Followed by that all incoming traffic is blocked and only TCP traffic is allowed to pass on ports 80 and 443. This is tested by SSH denial from `VM_1` to port 22 (SSH). A sanity check of SSH has been provided by removing the rules.

```
binat on $em3 from 192.168.1.11 to any -> $em3
binat on $em2 from 192.168.2.22 to any -> $em2
```

Figure 2. Binatting and Modifying Table

```
root@vm_1:~ # ssh d10nysus@192.168.1.1
ssh: connect to host 192.168.1.1 port 22: Operation timed out
root@vm_1:~ #
```

Figure 3. SSH Denied

```
root@vm_1:~ # ssh d10nysus@192.168.1.1
(d10nysus@192.168.1.1) Password for d10nysus@vm_3:
Last login: Sat Feb  8 18:09:10 2025 from 192.168.2.2
FreeBSD 14.2-RELEASE (GENERIC) releng/14.2-n269506-c8918d6c7412

Welcome to FreeBSD!

Release Notes, Errata: https://www.FreeBSD.org/releases/
Security Advisories:   https://www.FreeBSD.org/security/
FreeBSD Handbook:      https://www.FreeBSD.org/handbook/
FreeBSD FAQ:           https://www.FreeBSD.org/faq/
Questions List:        https://www.FreeBSD.org/lists/questions/
FreeBSD Forums:        https://forums.FreeBSD.org/

Documents installed with the system are in the /usr/local/share/doc/freebsd/
directory, or can be installed later with:  pkg install en-freebsd-doc
For other languages, replace "en" with a language code like de or fr.

Show the version of FreeBSD installed:  freebsd-version ; uname -a
Please include that output and any error messages when posting questions.
Introduction to manual pages:  man man
FreeBSD directory layout:      man hier

To change this login announcement, see motd(5).
Need to quickly empty a file? Use ": > filename".
                -- Dru <genesis@istar.ca>
d10nysus@vm_3:~ $
```

Figure 4. SSH Successful by removing rules

3. Webserver Setup

The webserver is chosen as `nginx`. Nginx is configured to listen on ports 80 and 443. Testing the connection through `curl` command and sending

request to VM_2's IP. A new webdir is also created by the name **nss_website** sending the request in the subdir we get the following HTML response.

Port block can also be observed by making the webserver listen on port 7000 and can be observed that tarfic doesn't get through. Thus confirming only ports 80 and 443 can be accessed.

```
root@vm_1:~ # curl http://192.168.1.1:7000/nss_website/
^C
```

Figure 5. No traffic for port 7000

```
root@vm_1:~ # curl http://192.168.1.1/nss_website/

<!DOCTYPE html>
<html>
        <body>
                <h1>Hello from VM_3</h1>
        </body>
</html>
root@vm_1:~ #
```

Figure 6. HTML response

This confirms that VM_1 doesn't know about VM_3 and all communication happens through VM_2.

```
root@vm_1:~ # tcpdump -i em2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on em2, link-type EN10MB (Ethernet), snapshot length 262144 bytes
19:35:35.962443 ARP, Request who-has 192.168.1.1 tell 192.168.1.11, length 28
19:35:35.962483 ARP, Reply 192.168.1.1 is-at 08:00:27:00:d9:c8 (oui Unknown), length
46
19:35:35.962483 IP 192.168.1.11.45365 > 192.168.1.1.http: Flags [S], seq 3161246257,
win 65535, options [mss 1460,nop,wscale 6,sackOK,TS val 3638711776 ecr 0], length 0
19:35:35.962989 IP 192.168.1.1.http > 192.168.1.11.45365: Flags [S.], seq 446675948,
ack 3161246258, win 65535, options [mss 1460,nop,wscale 6,sackOK,TS val 650475753 ecr
 3638711776], length 0
19:35:35.963006 IP 192.168.1.11.45365 > 192.168.1.1.http: Flags [.], ack 1, win 1027,
 options [nop,nop,TS val 3638711783 ecr 650475753], length 0
19:35:35.963099 IP 192.168.1.11.45365 > 192.168.1.1.http: Flags [P.], seq 1:88, ack 1
, win 1027, options [nop,nop,TS val 3638711783 ecr 650475753], length 87: HTTP: GET /
nss_website/ HTTP/1.1
19:35:35.966348 IP 192.168.1.1.http > 192.168.1.11.45365: Flags [P.], seq 1:313, ack
88, win 1027, options [nop,nop,TS val 650475753 ecr 3638711783], length 312: HTTP: HT
TP/1.1 200 OK
19:35:35.966543 IP 192.168.1.11.45365 > 192.168.1.1.http: Flags [F.], seq 88, ack 313
, win 1027, options [nop,nop,TS val 3638711787 ecr 650475753], length 0
19:35:35.967272 IP 192.168.1.1.http > 192.168.1.11.45365: Flags [.], ack 89, win 1027
, options [nop,nop,TS val 650475757 ecr 3638711787], length 0
19:35:35.967778 IP 192.168.1.1.http > 192.168.1.11.45365: Flags [F.], seq 313, ack 89
, win 1027, options [nop,nop,TS val 650475757 ecr 3638711787], length 0
19:35:35.967803 IP 192.168.1.11.45365 > 192.168.1.1.http: Flags [.], ack 314, win 102
7, options [nop,nop,TS val 3638711787 ecr 650475757], length 0
```

Figure 7. tcpdump on VM_1

```
19:40:06.315501 IP 192.168.1.11.45365 > 19 | 19:40:06.315516 IP 192.168.2.2.45365 > 192
2.168.1.1.http: Flags [F.], seq 88, ack 31 | .168.2.22.http: Flags [F.], seq 88, ack 31
3, win 1027, options [nop,nop,TS val 36387 | 3, win 1027, options [nop,nop,TS val 36387
11787 ecr 650475753], length 0             | 11787 ecr 650475753], length 0
19:40:06.315954 IP 192.168.1.1.http > 192. | 19:40:06.315940 IP 192.168.2.22.http > 192
168.1.11.45365: Flags [.], ack 89, win 102 | .168.2.2.45365: Flags [.], ack 89, win 102
7, options [nop,nop,TS val 650475757 ecr 3 | 7, options [nop,nop,TS val 650475757 ecr 3
638711787], length 0                      | 638711787], length 0
19:40:06.316215 IP 192.168.1.1.http > 192. | 19:40:06.316201 IP 192.168.2.22.http > 192
168.1.11.45365: Flags [F.], seq 313, ack 8 | .168.2.2.45365: Flags [F.], seq 313, ack 8
9, win 1027, options [nop,nop,TS val 650d7 | 9, win 1027, options [nop,nop,TS val 65047
5757 ecr 3638711787], length 0             | 5757 ecr 3638711787], length 0
19:40:06.316635 IP 192.168.1.11.45365 > 19 | 19:40:06.316649 IP 192.168.2.2.45365 > 192
2.168.1.1.http: Flags [.], ack 314, win 10 | .168.2.22.http: Flags [.], ack 314, win 10
27, options [nop,nop,TS val 3638711787 ecr | 27, options [nop,nop,TS val 3638711787 ecr
 650475757], length 0                      |  650475757], length 0
[0] 0:tcpdump*                                                    "vm_2" 19:47 10-Feb-21
```

Figure 8. tcpdump on VM_2

```
root@vm_3:~ # tcpdump -i em2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on em2, link-type EN10MB (Ethernet), snapshot length 262144 bytes
19:45:08.663101 ARP, Request who-has 192.168.2.22 tell 192.168.2.2, length 46
19:45:08.663101 ARP, Reply 192.168.2.22 is-at 08:00:27:e9:ea:5c (oui Unknown), length
 28
19:45:08.663101 IP 192.168.2.2.45365 > 192.168.2.22.http: Flags [S], seq 3161246257,
win 65535, options [mss 1460,nop,wscale 6,sackOK,TS val 3638711776 ecr 0], length 0
19:45:08.663101 IP 192.168.2.22.http > 192.168.2.2.45365: Flags [S.], seq 446675948,
ack 3161246258, win 65535, options [mss 1460,nop,wscale 6,sackOK,TS val 650475753 ecr
 3638711776], length 0
19:45:08.663102 IP 192.168.2.2.45365 > 192.168.2.22.http: Flags [.], ack 1, win 1027,
 options [nop,nop,TS val 3638711783 ecr 6504757753], length 0
19:45:08.663102 IP 192.168.2.2.45365 > 192.168.2.22.http: Flags [P.], seq 1:88, ack 1
, win 1027, options [nop,nop,TS val 3638711783 ecr 6504757753], length 87: HTTP: GET /
nss_website/ HTTP/1.1
19:45:08.663102 IP 192.168.2.22.http > 192.168.2.2.45365: Flags [P.], seq 1:313, ack
88, win 1027, options [nop,nop,TS val 650475753 ecr 3638711783], length 312: HTTP: HT
TP/1.1 200 OK
19:45:08.667216 IP 192.168.2.2.45365 > 192.168.2.22.http: Flags [F.], seq 88, ack 313
, win 1027, options [nop,nop,TS val 3638711787 ecr 6504757753], length 0
19:45:08.667234 IP 192.168.2.22.http > 192.168.2.2.45365: Flags [.], ack 89, win 1027
, options [nop,nop,TS val 650475757 ecr 3638711787], length 0
19:45:08.667495 IP 192.168.2.22.http > 192.168.2.2.45365: Flags [F.], seq 313, ack 89
, win 1027, options [nop,nop,TS val 650475757 ecr 3638711787], length 0
19:45:08.668334 IP 192.168.2.2.45365 > 192.168.2.22.http: Flags [.], ack 314, win 102
7, options [nop,nop,TS val 3638711787 ecr 650475757], length 0
```

Figure 9. tcpdump on VM_3

4. Task 2

A new user **temphttp** was created and was made owner of the web directory residing in /var/www/nginx/.

The webserver is hosted with **nginx**. Nginx works with a master and worker thread. The master thread runs with root privileges and spawn a worker thread. As stated in documentation "The main purpose of the master process is to read and evaluate configuration files, as well as maintain the worker processes." A worker thread can be assigned **uid** and **guid** by the master process. If the master thread is running as root, then nginx will **setuid()**/**setgid()** to USER or GROUP. If GROUP is not specified, then nginx uses the same name as USER. By default it's nobody user and nobody or nogroup group [1].

I chmoded my index.html to **700** since worker thread would only be able to read the file if permission to **others** were provided.

To access the index.html in my web directory I tried setting the **setuid** bit for **nginx** binary in **sbin**. However this did not help in accessing and reading of the html file.

```
root@vm_1:~ # curl http://192.168.1.1/nss_website/
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.26.2</center>
</body>
</html>
root@vm_1:~ #
```

Figure 10. Forbidden to access

```
-r-sr-xr-x   1 root wheel  1225200 Jan 30 12:07 nginx
```

Figure 11. `nginx` bit set

The switching of ownership of the `nginx` binary from root to `temphttp` would not be resulting in anything since to listen below ports 1024 you need root privileges to start those services.

Just to be sure that this wasn't a nginx internal implementation issue, I used Apache24 webserver to replicate the same procedure and with apache too I wasn't able to access the index.html file in the web directory.

4.1. Read through ACLs

To enable ACLs on FreeBSD we have to edit `/etc/fstab` file and add ACL support. After that we set acls using `setfacl`. The command to enable acl on index.html is `setfacl -m u:www:r-x index.html`. To delete them command is `setfacl -b <filename>`

The user is assigned as `www` because that is the default username of the worker thread. This allows us to access the file through ACL while the file Non ACL permissions are still `700`.

```
root@vm_3:/var/www/nginx/nss_website # getfacl index.html
# file: index.html
# owner: temphttp
# group: temphttp
user::rwx
user:www:r-x
group::---
mask::r-x
other::---
root@vm_3:/var/www/nginx/nss_website #
```

Figure 12. ACL set on index.html

4.2. Webdir Access with nginx "user" modification

Since setting the `setuid` bit for nginx executable didn't result in anything there is another way which uses the implementation of `setuid` for the worker thread which gets set by the master thread when `user` is defined in `nginx.conf`.

```
user  temphttp;
worker_processes  1;
```

Figure 13. Changing `nginx.conf`

We can see that the worker process is running with the permissions of the user `temphttp` by using the command `ps aux | grep nginx`.

This results in accessing the file in the webdirectory even when the permissions of the file are set to `700`.

```
root@vm_3:/var/www/nginx/nss_website # ps aux | grep nginx
root     1193   0.0  0.1 22472 8632  -  Is   20:56    0:00.00 nginx: master process
temphttp 1194   0.0  0.1 22472 9108  -  I    20:56    0:00.01 nginx: worker process
```

Figure 14. ps output

Even though this is not traditional "`setuid`" method this could be less riskier than setting the uid of the binary. Setting the `setuid` of the worker thread could result in security risks. For example, a file could be owned by root and have permissions `700` this could result in execution as root but shouldn't have been since the permissions to others and group were `0` respectively.

```
root@vm_1:~ # curl http://192.168.1.1:443/

<!DOCTYPE html>
<html>
        <body>
                <h1>Hi! Add /nss_website/ for subdir</h1>
        </body>
</html>
root@vm_1:~ # curl http://192.168.1.1:443/nss_website/

<!DOCTYPE html>
<html>
        <body>
                <h1>Hello from VM_3</h1>
        </body>
</html>
root@vm_1:~ #
```

Figure 15. cURL successful

5. Appendix

5.1. /etc/pf.conf

```
# Declaring interfaces
em2="em2"
em3="em3"

# Making network corresponding it
net_1=$em2:network
net_2=$em3:network

# Declaring ports
allowed_port="{http,https}"

# Binatting
binat on $em3 from 192.168.1.11 to any -> $em3
binat on $em2 from 192.168.2.22 to any -> $em2

# Blocking Traffic
block in all
pass proto tcp from $net_2 to port $allowed_port
pass proto tcp from $net_1 to port $allowed_port
```