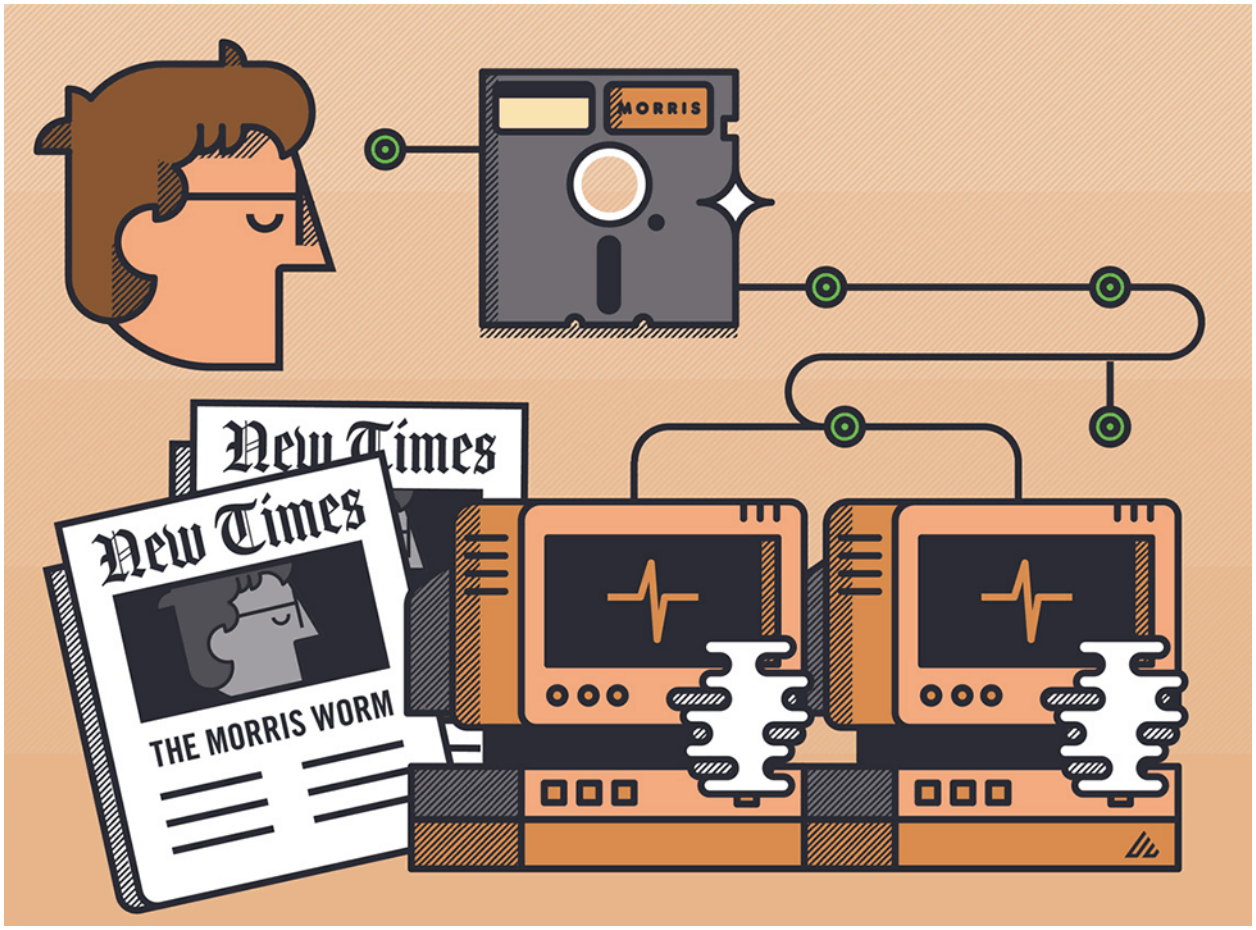


CSE 406

Malware Design: Morris Worm

Report of Findings

—

Saif Ahmed Khan
ID: 1705110
Section: B2
Dept. of CSE, BUET

Overview

The Morris worm or Internet worm of November 2, 1988, was one of the oldest computer worms distributed via the Internet, and the first to gain significant mainstream media attention.

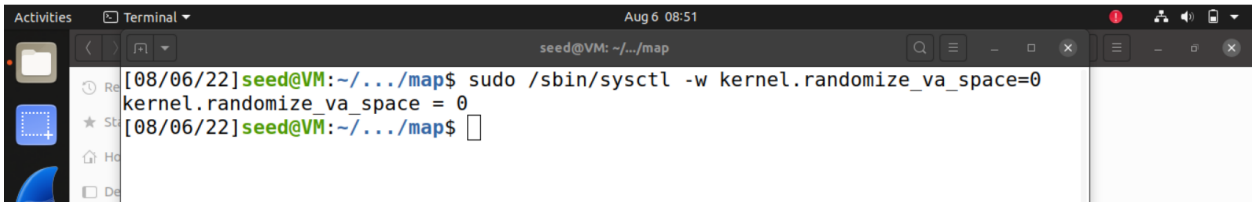
The Morris worm, named for its creator, Cornell University student Robert Tappan Morris, rapidly infected the limited (by today's standards) computers connected to the internet.

To design a simple version of the Morris worm we were given 4 tasks to perform in SEED-VM, which are described below in detail. The **LabSetup** and **Getting Familiar** parts are not shown.

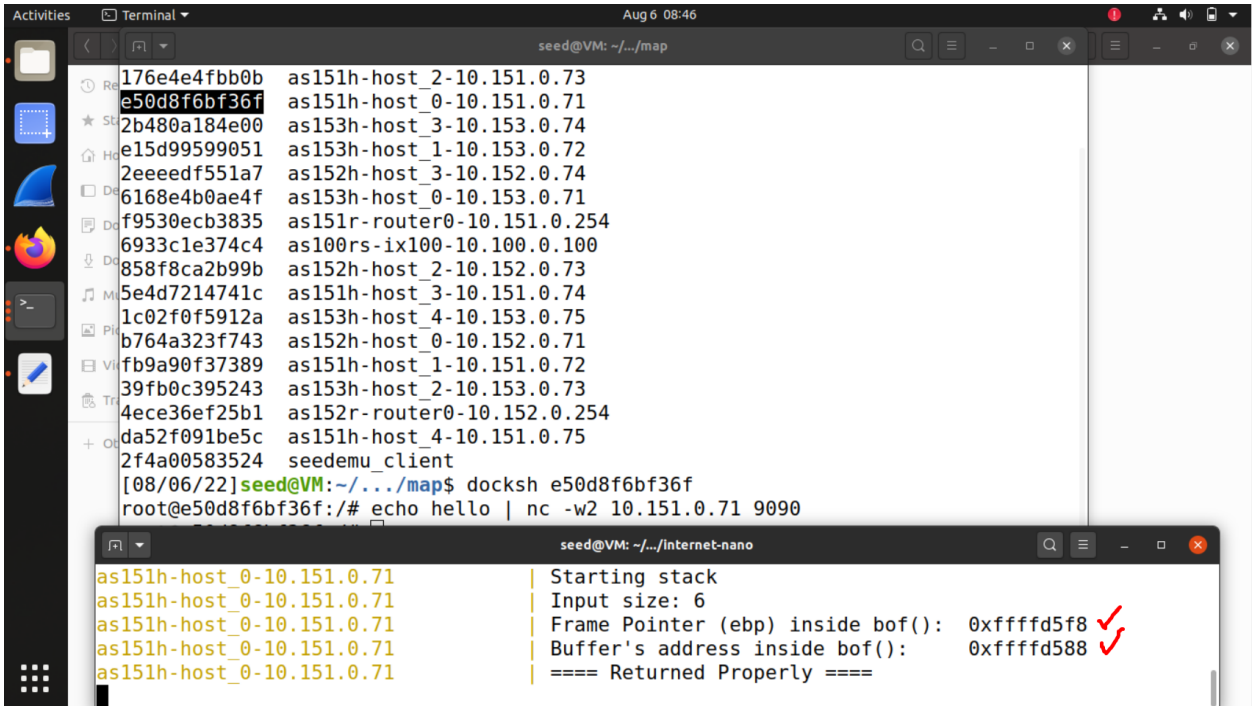
Task 1: Attacking Any Target Machine

In this task, we hard code the IP of a target machine to attack it once. This step helps us verify whether the worm can crawl to another machine or not.

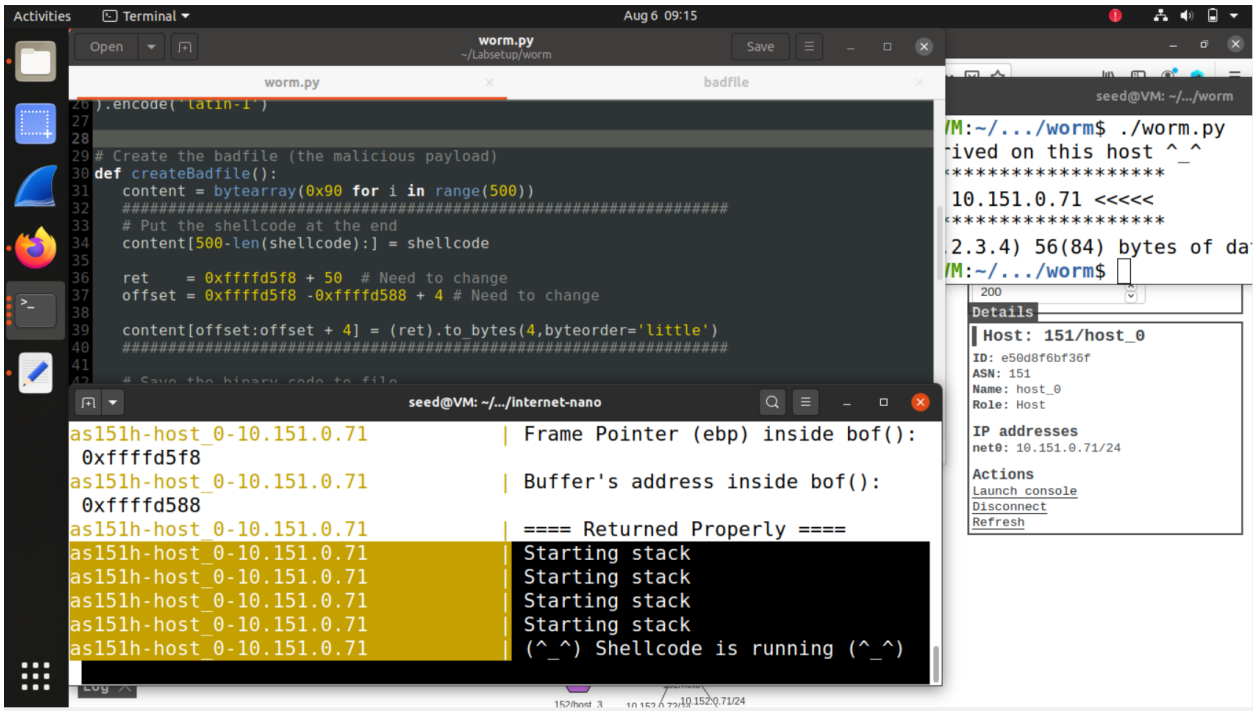
First we turn of address randomization globally for all containers for the buffer overflow attack to work:



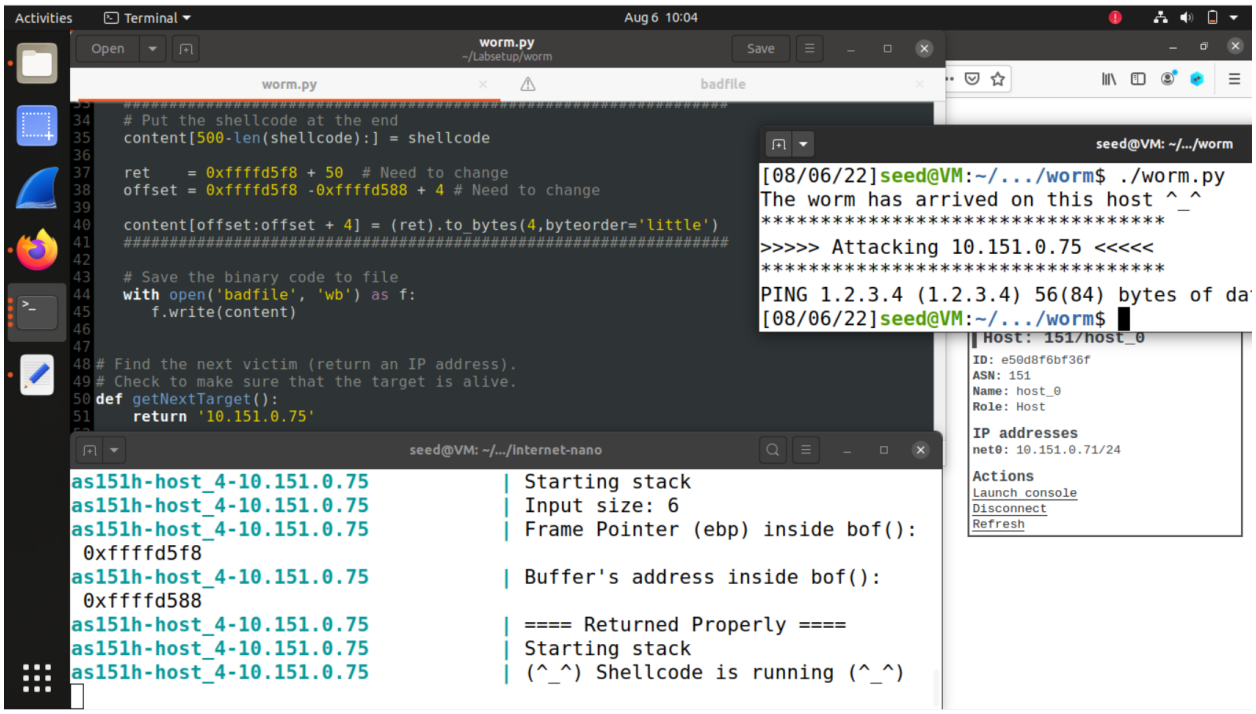
We then go into the root shell of the target machine (**10.151.0.71**) via **dockps** and **docksh** alias commands and use echo a message via netcad command. This gives us **ret** and **buffer address** values for the buffer overflow. Because address randomization is turned off, all container/host machines will have the same value for **ret** and **buffer address** which makes our lives easier:



In worm.py, we paste the **ret** and **buffer address** values and adjust ret for gdb values by adding a random integer. The **offset** is the difference of two values obtained plus 4. Now running worm.py gives us the highlighted result in internet-nano docker shell:



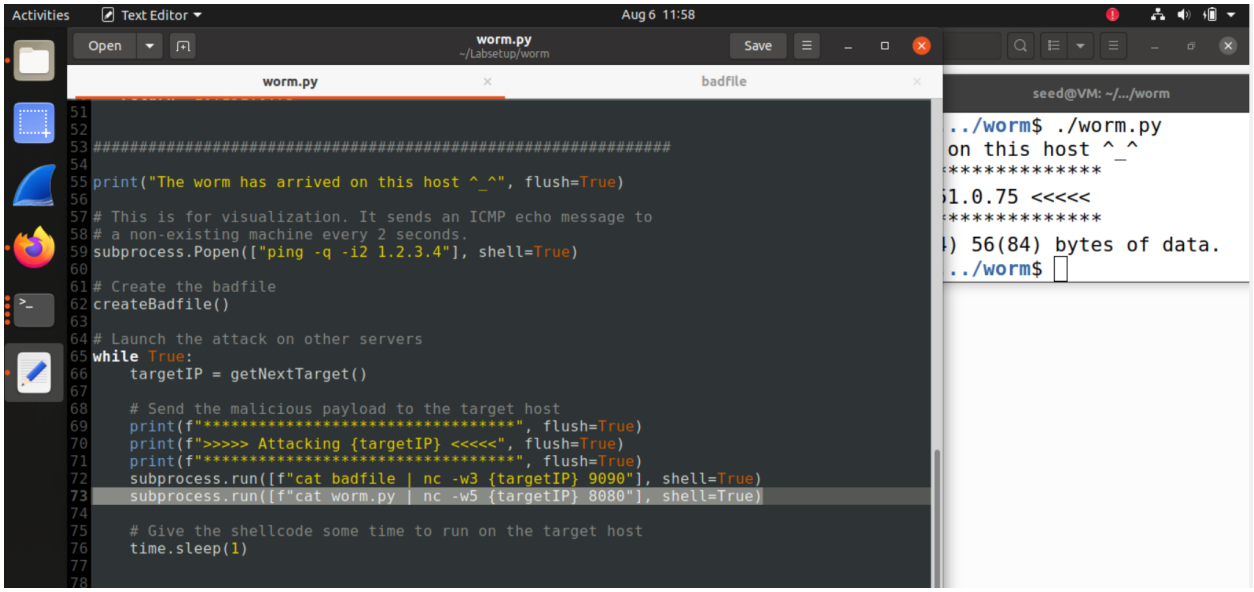
Here is task 1 again but for a different target IP (**10.151.0.75**) :



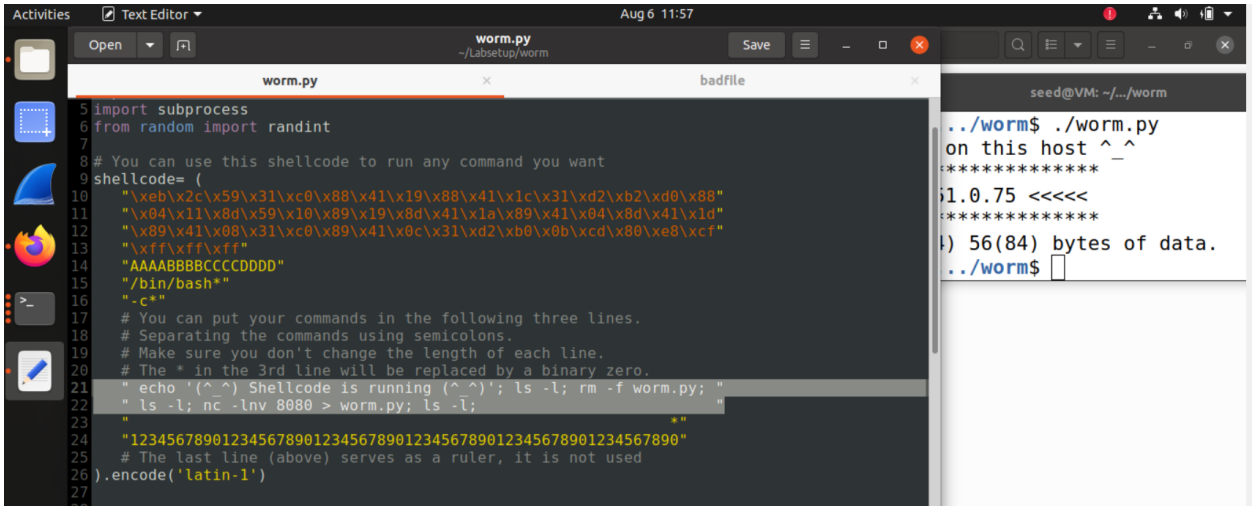
Task 2: Self Duplication

Task 1 enabled us to open a root shell in the other machine, now we can copy the worm.py to the target machine. Using approach 2, the shellcode is the “pilot code” and worm.py is the “sophisticated code”.

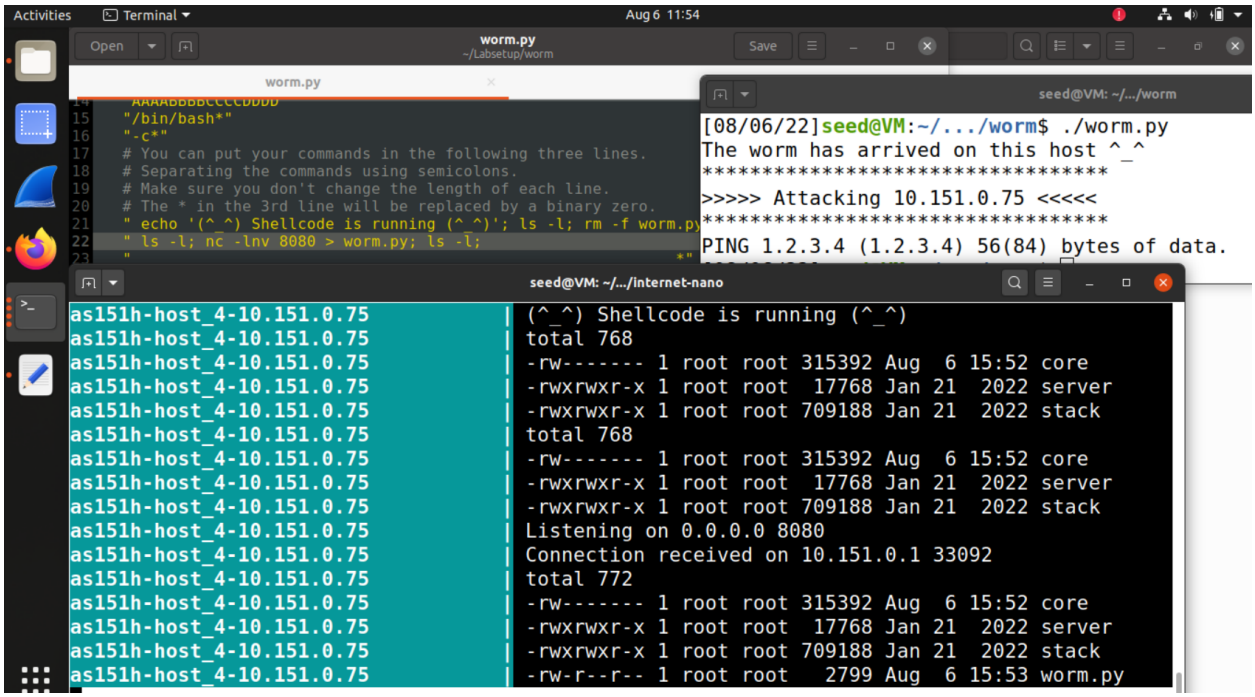
We modeled the host or attacker machine to be the client and the victim to be the server. Host machine (client) sends worm.py [Line 73] through port 8080:



The victim’s shell now runs “nc -lv 8080” command as it is a server receiving worm.py from the client/attacker:



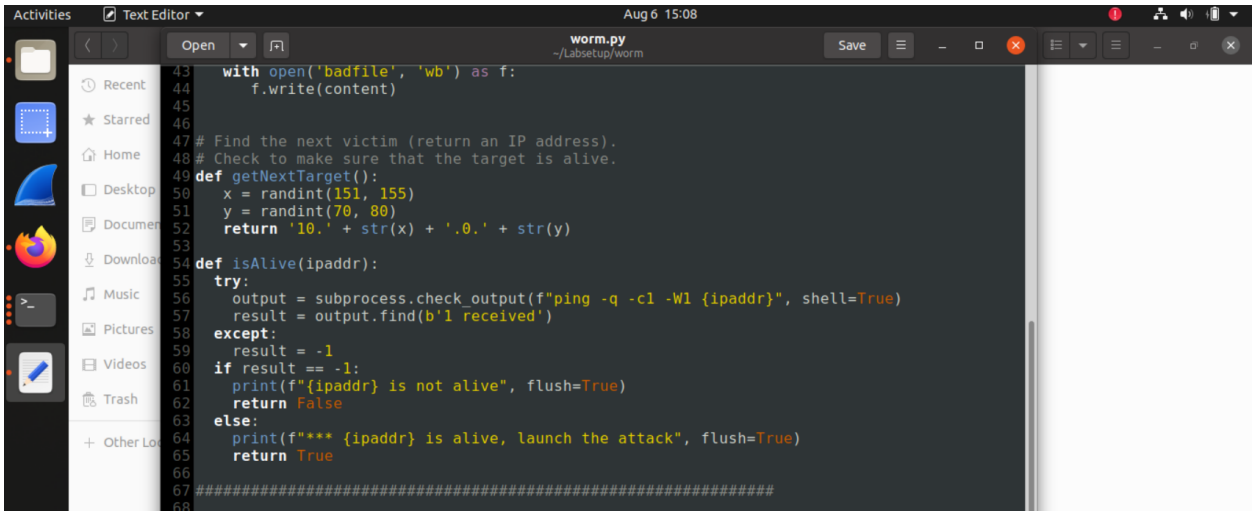
The following result is obtained. The first “ls” command shows no worm.py. After the “nc -lnv 8080” command, we can see in the highlighted terminal that a new file named worm.py has been created:



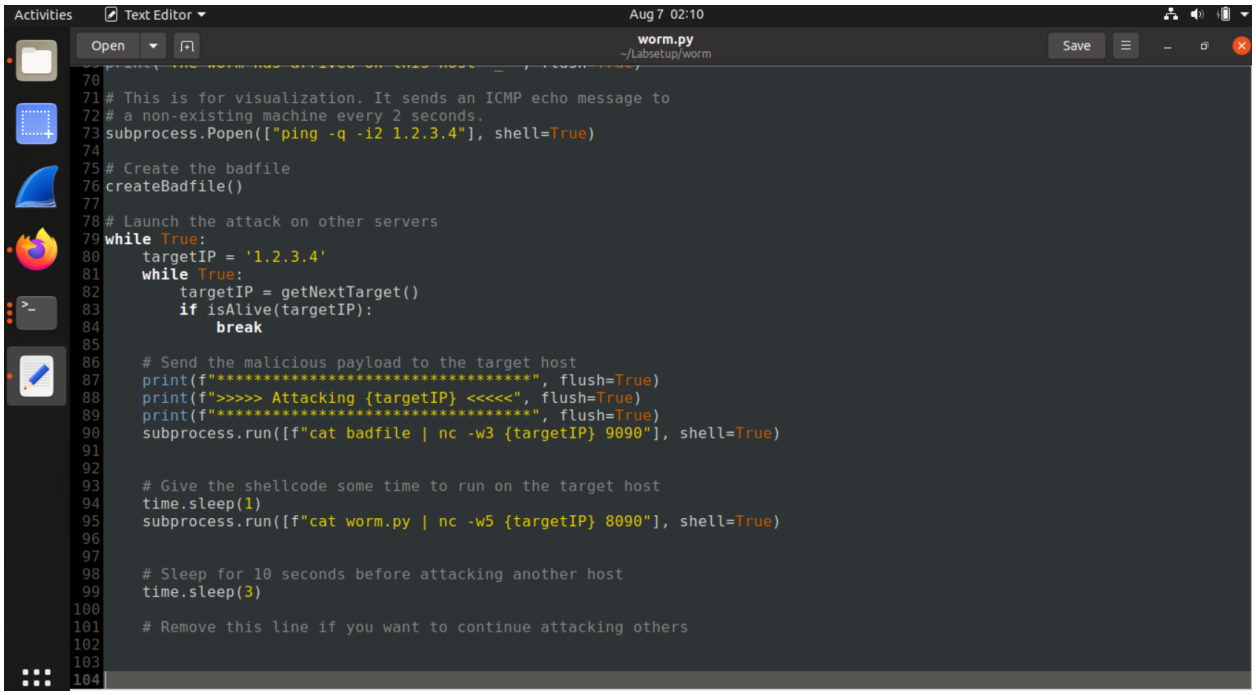
Task 3: Propagation

We have managed to copy worm.py to other machines but we need to make those machines send a copy to even more machines creating a propagation of worm infection.

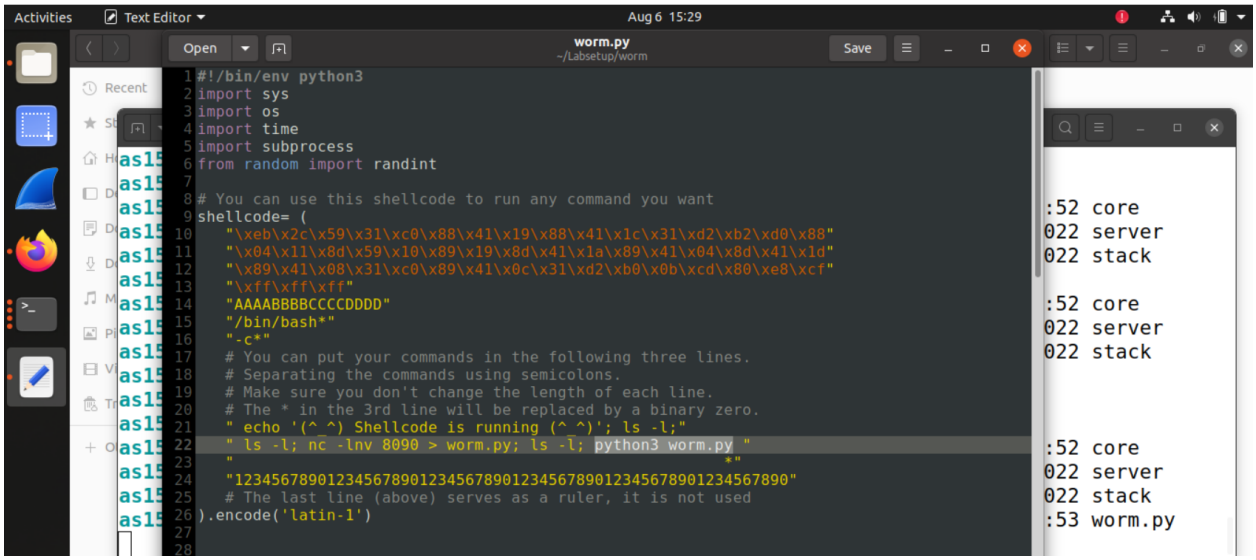
We find random IPs to target using a modified **getNextTarget**() by removing hard coded IP and check if it is alive using **isAlive**() method:



Now in the while loop for attacking others, we check for new alive targets to attack [Lines 80-84] and also remove “exit(0)” to keep propagation going :

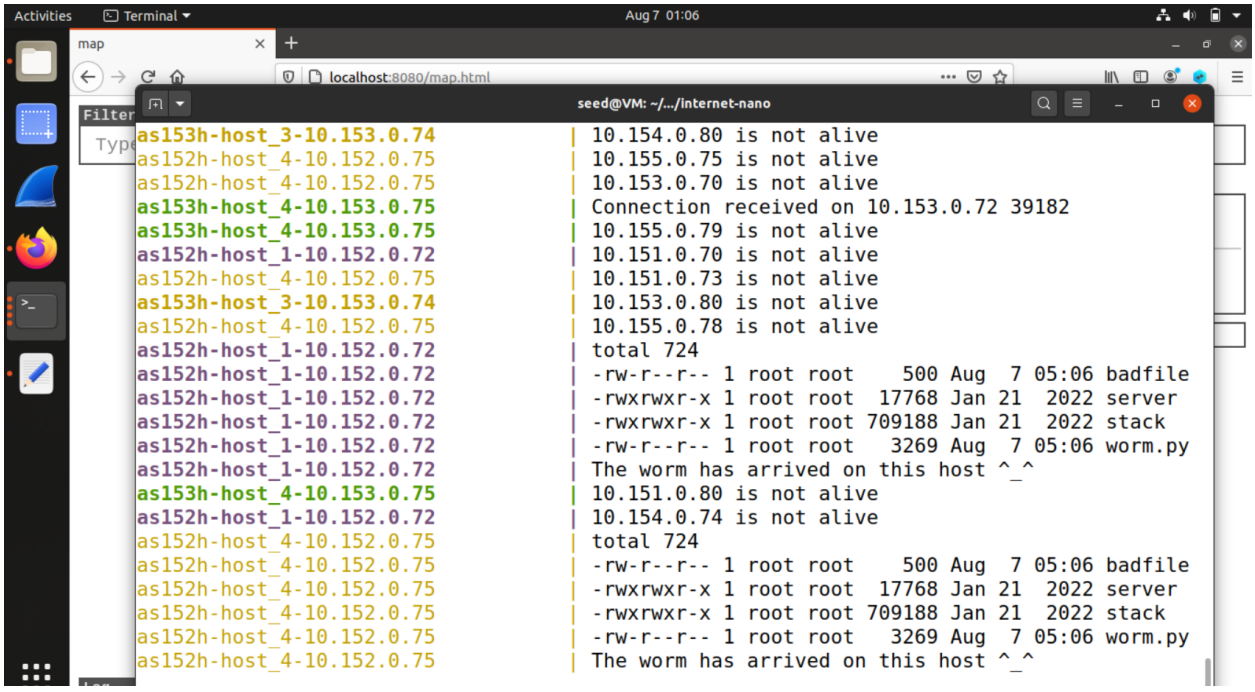


Now that we are sending copies of worm.py to many machines, we need to execute worm.py in the machine which receives it so that is will start doing what the host machine did, that is, spread the infection:



We then run “worm.py” again:

And obtain the following results. Different machines are now propagating the worm and executing it to keep sending it to other machines:



Task 3 does not check whom to send the file, so one machine will end up with multiple copies and executions of the worm.py. This leads to overuse of CPU resources and cause the VM to crash!!!

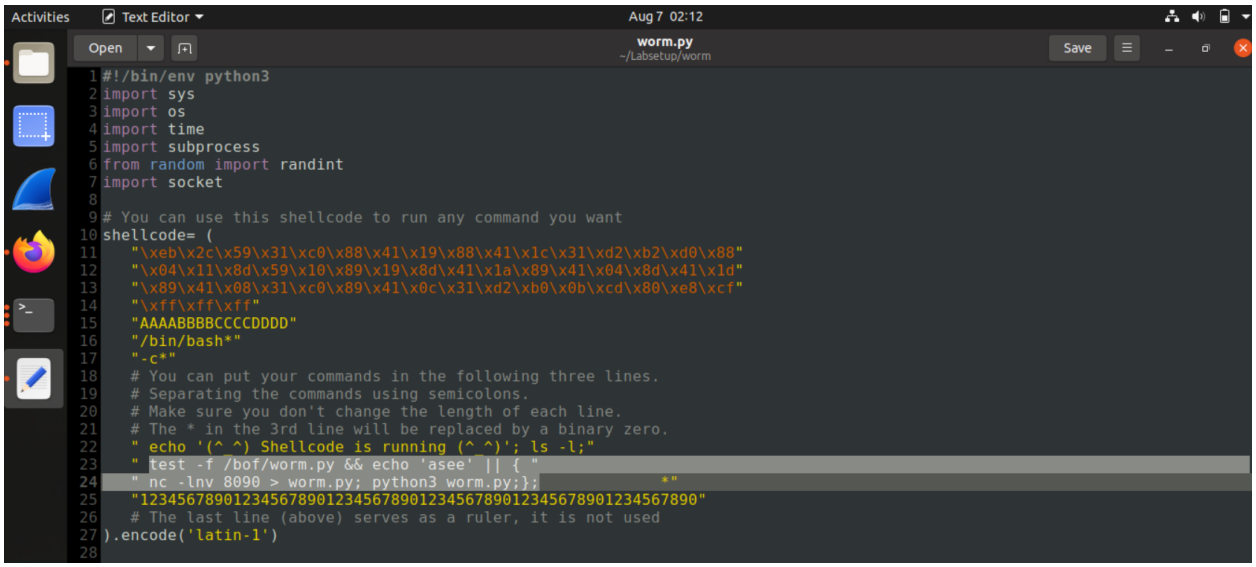
Task 4: Preventing Self-Infection

In the previous task, we kept on sending and executing worm.py to everyone. There is a certain chance that the worm would come back to our own machine and infect us which we obviously do not want, unless you are up to no good!

Also we need to only send one copy of the worm.py to a machine and execute it once, multiple copies and executions are not needed as worm.py runs in a loop to keep attacking other machines.

So we provide a simple checking mechanism. If the file worm.py already exist in the **/bof** folder, then we do not need to open a connection to receive worm.py and execute it again because it is already running in that machine.

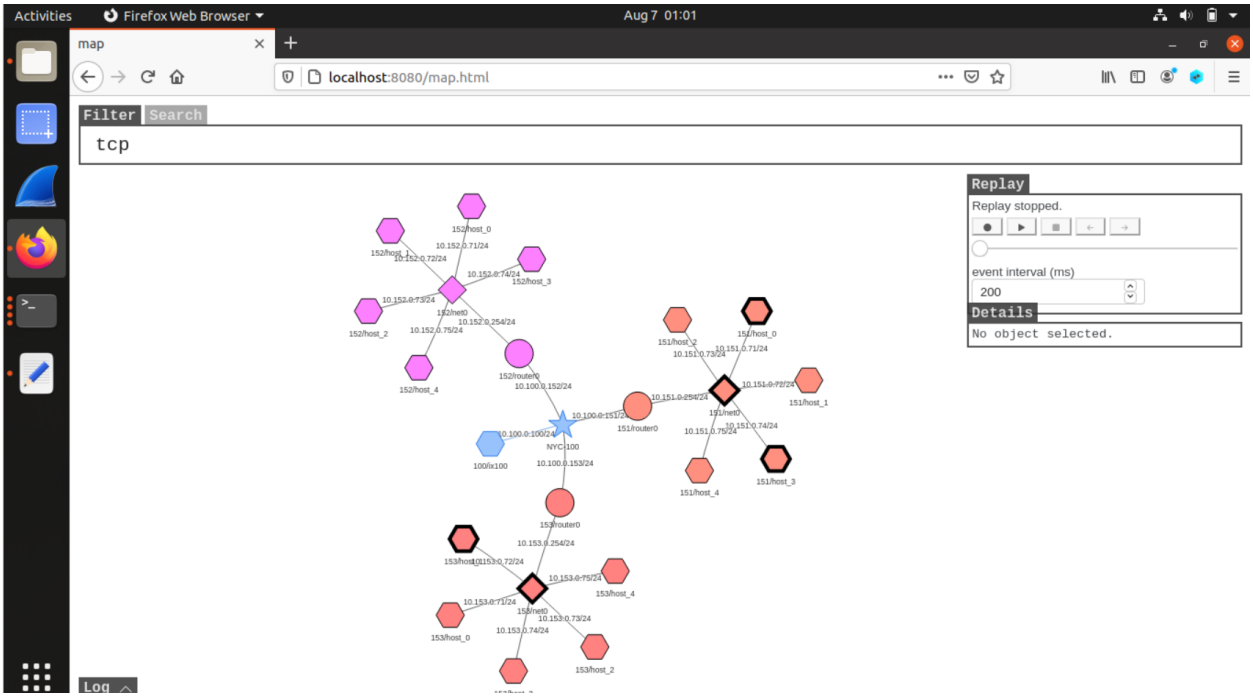
Here are the changes to the shell code. Test to see if the file exist, if not only then open a connection, receive worm.py and execute it:



Final Result: The Worms in Action

The map application, when set to **tcp packets** filter, can help us visualize the propagation of the worm which is sent via tcp connections [server-client]. It can be found when we go to the url: **http://localhost:8080/map.html**

This was the result when worm.py was executed once on the host machine after the completion of **Task 4**:



The infection was observed to be smoother and more controlled than that in Task 3 after the checking mechanism was implemented.