BLG460E - Secure Programming

HW-1

Ahmet Çiçekci

150140006

Part 1

P = 150140006 % 10 + 2 = 8

First, I found the address of buffer using gdb:

```
$ print &buffer
— 0xbffff27c
```

Then, I inspected the memory parts following the buffer:

```
$ x/16gx 0xbffff27c
```

I found the return address at 0xbffff28c. And, I found another way on the Internet to find return address. It is a gdb command:

```
$ info frame
```

It gives the addresses of eip, ebp… etc. Here, eip shows where the return address is. As expected, eip at 0xbffff28c here.

Now, eip - buffer, so 0xbffff28c - 0xbffff27c = 16. 8 of 16 bytes are buffer. info frame showed ebp at 0xbffff288, so 4 of 16 bytes are stack frame pointer. But I don't know what the other 4 bytes are.

Anyway, now we have seen eip, the return address is 16 bytes away from the buffer. So, I assigned buffer + 16 to the return address:

```
int* ret;
ret = buffer + 16;
```

Now, we have the return address and we want to change it so that the uid is not overwritten whit default_uid.

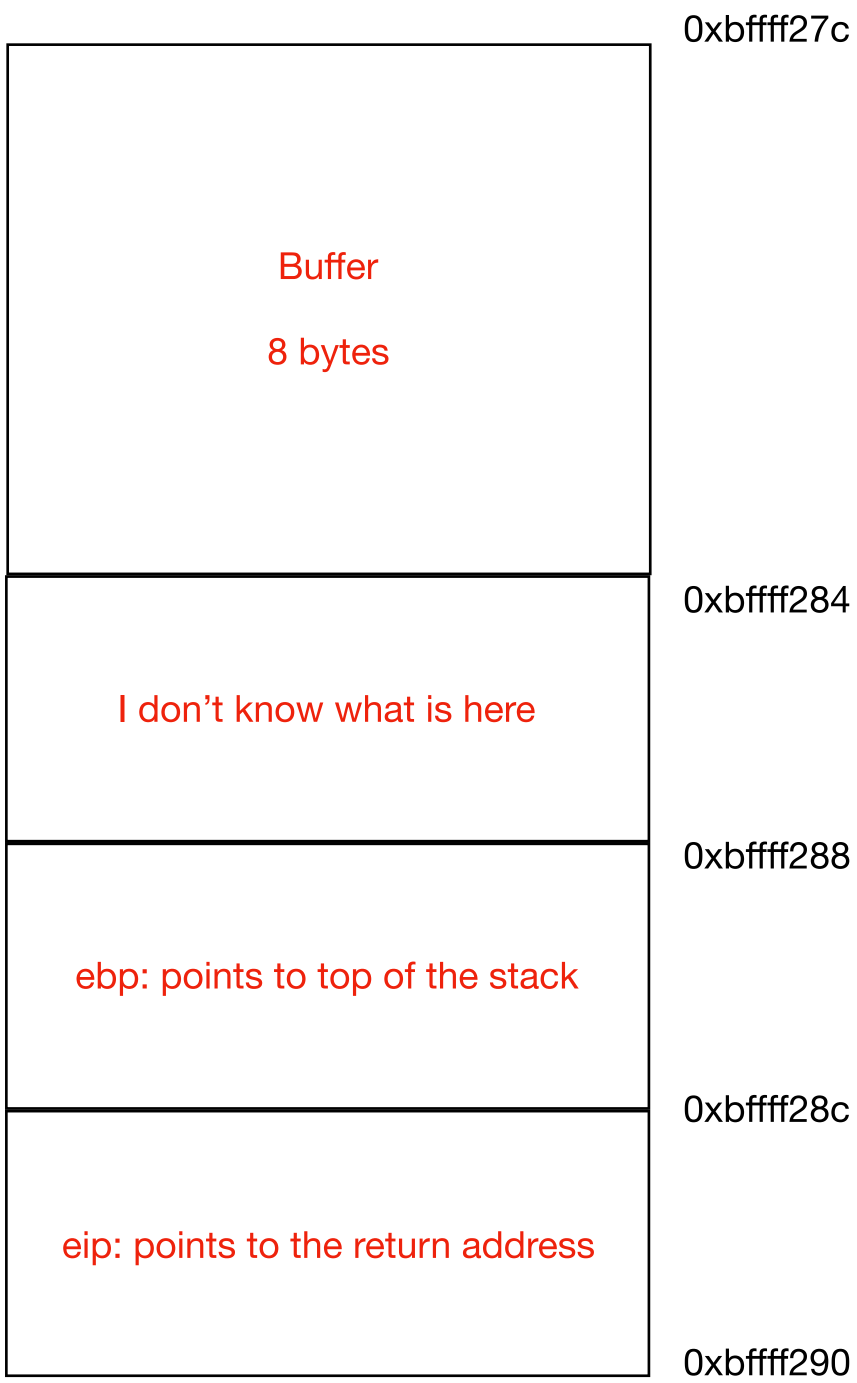
In order to see assembly code and addresses, I used this command:

```
$ disassemble /m main
```

The return address showed 0x08048664. And, in order to jump over the *uid = default_uid;* line, it should have shown 0x0804866c. 0x0804866c - 0x08048664 = 8, so I increased the return address by 8:

```
    (*ret) += 8;
```

Stack Structure:


0xbffff27c
Buffer
8 bytes
0xbffff284
I don't know what is here
0xbffff288
ebp: points to top of the stack
0xbffff28c
eip: points to the return address
0xbffff290

When I compile without -fno-stack-protector, it doesn't jump over the *uid = default_uid;* line, and it is logged in as guest.

Part 2

I found the return address using the same way in the part 1. First, I found the address of password:

```
$ print &password
— 0xbffff248
```

I inspected the memory parts following the password, and found the return address at 0xbffff25c. 0xbffff25c - 0xbffff248 = 20. Using the info frame command, I found the stack parts:

0xbffff248 - 0xbffff250: password, 8 bytes

0xbffff250 - 0xbffff254: esi, source address for memcpy, 4 bytes

0xbffff254 - 0xbffff258: edi, destination address for memcpy, 4 bytes

0xbffff258 - 0xbffff25c: ebp, 4 bytes

And,

0xbffff25c - 0xbffff260: eip, return address

In order to overflow buffer, we should give a 20 bytes of input, and then 4 bytes of return address that we want to jump. So 24 bytes of input:

4142434445464748 92f2ffbf92f2ffbfb8f2ffbf02870408

- 8 bytes, 4142434445464748: this is the password. This can be anything other than 00, doesn't matter.

- 4 bytes, 92f2ffbf: this is esi. Because in the memory it is written in reverse order, actually it is 0xbffff292. Actually, esi was 0x00000000, but 00 is null character and I didn't know what to do, and just gave the same address as edi.

- 4 bytes, 92f2ffbf: edi, 0xbffff292. When I run the program, it was 0xbffff292, and in order to avoid changing any part of the stack except password and return address, I gave this as input.

- 4 bytes, b8f2ffbf: ebp, 0xbffff2b8. I tried to avoid breaking the structure of the stack, again.

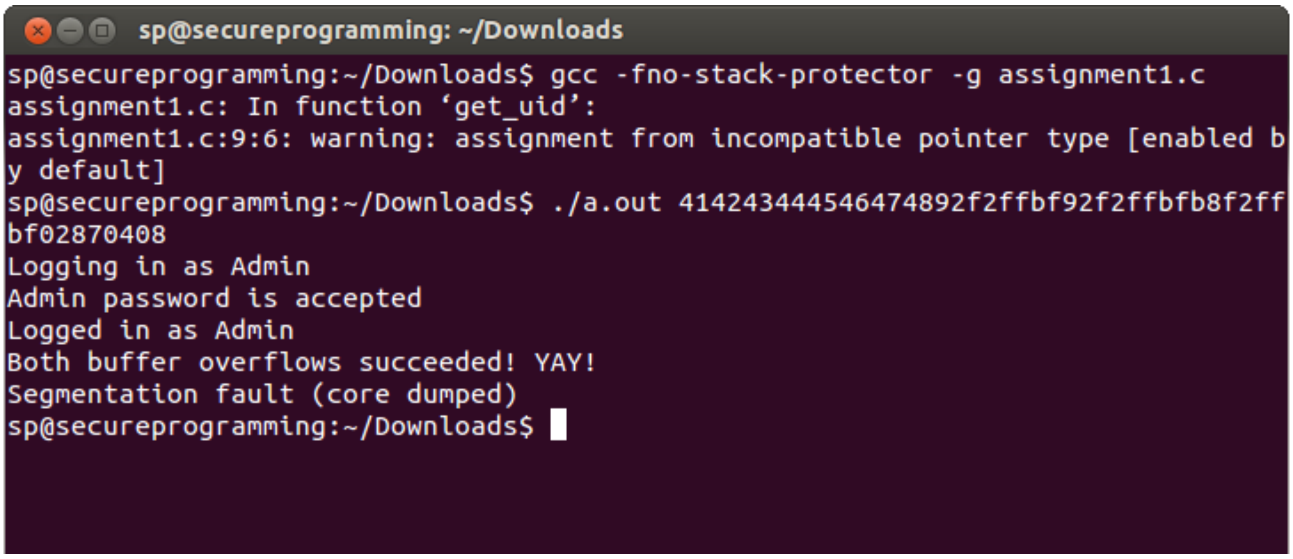
- 4 bytes, 02870408: and the return address, the main part for us. At first, the return address was 0x080486e9. By inspecting the assembly code and addresses (disassemble /m main), I found the address of else statement: 0x08048702. Because it is written to the memory in reverse order, I gave this input: 02870408.

So, the program jumped to the else statement.

After copying operation;

- the password at stack has changed: 414243…
- ebp and edi haven't changed because I gave the appropriate input, but I had to change esi, because it was 00000000, I didn't know what to do, to avoid changing this part.
- return address has changed from 0x080486e9 to 0x08048702.

When I didn't use *sudo sysctl kernel.randomize va space=0,* I still succeeded at overflow. I don't know why.



```
sp@secureprogramming: ~/Downloads
sp@secureprogramming:~/Downloads$ gcc -fno-stack-protector -g assignment1.c
assignment1.c: In function ‘get_uid’:
assignment1.c:9:6: warning: assignment from incompatible pointer type [enabled b
y default]
sp@secureprogramming:~/Downloads$ ./a.out 414243444546474892f2ffbf92f2ffbfb8f2ff
bf02870408
Logging in as Admin
Admin password is accepted
Logged in as Admin
Both buffer overflows succeeded! YAY!
Segmentation fault (core dumped)
sp@secureprogramming:~/Downloads$
```