

level13

≡ 태그	Assembly Shell
☼ 상태	완료

풀이과정

[겪었던 어려움](#)

[풀이과정](#)

[gdb](#)

[assembly](#)

[정답](#)

[출처](#)

풀이과정

겪었던 어려움

풀이과정

gdb

strings을 통해서 주어진 level13의 파일의 내용을 읽어볼 때 `UID %d started us but we we expect %d` 이라는 내용을 볼 수 있고 실행 시켰을 때 `UID 2013 started us but we we expect 4242` 라는 문자열을 볼 수 있다.

그 이후 `gdb level13` 을 통해서 실행시킨 후 `disas(disassemble)` main을 통해서 메인 함수에 대해서 어셈블리 디버깅을 진행하면 아래의 내용을 볼 수 있다.

```
(gdb) disas main
Dump of assembler code for function main:
0x0804858c <+0>: push    %ebp
0x0804858d <+1>: mov     %esp,%ebp
0x0804858f <+3>: and     $0xffffffff0,%esp
0x08048592 <+6>: sub     $0x10,%esp
0x08048595 <+9>: call    0x8048380 <getuid@plt>
0x0804859a <+14>: cmp     $0x1092,%eax
0x0804859f <+19>: je      0x80485cb <main+63>
0x080485a1 <+21>: call    0x8048380 <getuid@plt>
0x080485a6 <+26>: mov     $0x80486c8,%edx
0x080485ab <+31>: movl    $0x1092,0x8(%esp)
0x080485b3 <+39>: mov     %eax,0x4(%esp)
0x080485b7 <+43>: mov     %edx,(%esp)
0x080485ba <+46>: call    0x8048360 <printf@plt>
```

```

0x080485bf <+51>:    movl    $0x1, (%esp)
0x080485c6 <+58>:    call   0x80483a0 <exit@plt>
0x080485cb <+63>:    movl    $0x80486ef, (%esp)
0x080485d2 <+70>:    call   0x8048474 <ft_des>
0x080485d7 <+75>:    mov     $0x8048709, %edx
0x080485dc <+80>:    mov     %eax, 0x4(%esp)
0x080485e0 <+84>:    mov     %edx, (%esp)
0x080485e3 <+87>:    call   0x8048360 <printf@plt>
0x080485e8 <+92>:    leave
0x080485e9 <+93>:    ret

```

End of assembler dump.

assembly

```

0x0804858c <+0>:    push    %ebp
: 현재 스택 프레임의 베이스 포인터 ebp를 스택에 push합니다.
0x0804858d <+1>:    mov     %esp, %ebp
: 현재 스택 포인터 esp의 값을 ebp에 복사합니다. 이로써 새로운 스택 프레임이 생성되었습니다.
0x0804858f <+3>:    and     $0xffffffff0, %esp
: esp를 16바이트 경계에 맞춰 정렬합니다.
0x08048592 <+6>:    sub     $0x10, %esp
: 스택 포인터 esp를 16바이트 내려 지역 변수를 위한 공간을 확보합니다.
0x08048595 <+9>:    call   0x8048380 <getuid@plt>
: getuid() 함수를 호출하여 현재 사용자의 UID를 가져옵니다.
0x0804859a <+14>:   cmp     $0x1092, %eax
: 가져온 UID가 0x1092(4242)인지 확인합니다.
0x0804859f <+19>:   je      0x80485cb <main+63>
: UID가 0x1092이면 0x80485cb <main+63> 위치로 점프합니다.
0x080485a1 <+21>:   call   0x8048380 <getuid@plt>
: UID가 0x1092가 아니면 다시 getuid()를 호출합니다.
0x080485a6 <+26>:   mov     $0x80486c8, %edx
: 문자열 포인터 0x80486c8을 edx에 저장합니다.
0x080485ab <+31>:   movl    $0x1092, 0x8(%esp)
: 스택에 0x1092를 저장합니다.
0x080485b3 <+39>:   mov     %eax, 0x4(%esp)
: 가져온 UID를 스택에 저장합니다.
0x080485b7 <+43>:   mov     %edx, (%esp)
: 문자열 포인터를 스택에 저장합니다.
0x080485ba <+46>:   call   0x8048360 <printf@plt>
: printf() 함수를 호출하여 오류 메시지를 출력합니다.
0x080485bf <+51>:   movl    $0x1, (%esp)
: 스택에 1을 저장합니다.
0x080485c6 <+58>:   call   0x80483a0 <exit@plt>

```

```

: exit(1) 함수를 호출하여 프로그램을 종료합니다.
0x080485cb <+63>: movl    $0x80486ef, (%esp)
: 문자열 포인터 0x80486ef를 스택에 저장합니다.
0x080485d2 <+70>: call   0x8048474 <ft_des>
: ft_des() 함수를 호출합니다.
0x080485d7 <+75>: mov     $0x8048709, %edx
: 문자열 포인터 0x8048709를 edx에 저장합니다.
0x080485dc <+80>: mov     %eax, 0x4(%esp)
: ft_des() 함수의 반환값을 스택에 저장합니다.
0x080485e0 <+84>: mov     %edx, (%esp): 문자열 포인터를 스택에 저장합니다.
0x080485e3 <+87>: call   0x8048360 <printf@plt>: printf() 함수를 호출하여 결과를 출
0x080485e8 <+92>: leave: 현재 스택 프레임을 해제합니다.
0x080485e9 <+93>: ret: 함수를 반환합니다.

```

여기서 *main + 14에서 %eax가 0x1092와 같은지 확인하는 부분이 있는데, 이 hex 값을 dec으로 바꾸면 4242에 해당하는 값이므로 `UID 2013 started us but we we expect 4242` 에 해당하는 4242인지 확인하는 부분이라는 것을 알 수 있다.

종단점을 break를 통해서 *main+14으로 잡는 `break *main+14` 명령어를 실행시킨 후 `r(run)` 을 통해서 실행시키면 해당 부분에서 일시정지를 하게 되는데 \$eax에 우리가 원하는 UID값인 4242를 `set $eax=4242` 명령어를 입력 한 후 `cont(continue)` 를 통해서 실행시키면 원하는 flag값을 얻을 수 있다.

정답

```

(gdb) break *main+14
Breakpoint 1 at 0x804859a
(gdb) r
Starting program: /home/user/level13/level13

Breakpoint 1, 0x0804859a in main ()
(gdb) set $eax=4242
(gdb) cont
Continuing.
your token is 2A31L79asukciNyi8uppkEuSx

```

`"2A31L79asukciNyi8uppkEuSx"`

출처

gdb 명령어 정리

 시스템이나 포너블을 시작할 때, 일반적인 대회 문제들은 (몇몇을 제외한) LOB나 FTZ와 같이 ...

<https://blog.naver.com/mathboy7/220227929322>


```

bra_1stFloor -ls
bra_1stFloor -ls
bra_1stFloor -ls
bra_1stFloor -ls ls
Iron_golem.c
bra_1stFloor -ls gdb ./iron_golem
ed 1st Linux (6.1post-1.2008669.41rh)
2004 Free Software Foundation, Inc.
see software, covered by the GNU General Public License, and you are
to change it and/or distribute copies of it under certain conditions.
w copying" to see the conditions.
absolutely no warranty for GDB. Type "show warranty" for details.
was configured as "i386-redhat-linux-gnu"... (no debugging symbols found)... Using host
ry "/lib/tls/libthread_db.so.1".

```

GDB 사용 시 set 명령어

gdb 사용시 set 명령어를 이용하면 자신이 원하는대로 setting을 해서 진행해볼 수 있다. 예를 들어 아래와 같은 소스코드가 있다. 해당 코드를 컴파일 한 뒤 gdb를 열고 got overwrite를 진행해보도록 하겠다. 컴파일 후 gdb를 이용하여 어셈블리어 코드로 보았다. printf@plt 호출 부분을 들여다

 <https://xn--vj5b11biyw.kr/124>


```
~/Desktop/pwn_study_code# cat got.c
stdio.h>

printf("ls");
return 0;

~/Desktop/pwn_study_code# gcc -o got
~/Desktop/pwn_study_code# gdb got -q
```

0X1092 to decimal

What is 0X1092 to decimal? How to convert 0X1092 to decimal. What is 0X1092 in decimal? (0X1092 hex to decimal or 1092 hex to decimal)

 <https://decimal.info/hex-to-decimal/1/how-to-convert-0X1092-to-decimal.html>