# [pwnable.kr WriteUp]

이태규(ltg2030)

## [Toddler's Bottle] leg 2PT

| 문제 설명 |
| --- |
| Daddy told me I should study arm.<br>But I prefer to study my leg!<br><br>Download : http://pwnable.kr/bin/leg.c<br>Download : http://pwnable.kr/bin/leg.asm<br><br>ssh leg@pwnable.kr -p2222 (pw:guest) |

Program Counter : 일명 PC, 다음에 실행할 명령어의 주소를 갖는다. 그러나 ARM 에서는 명령어 2개가 실행된 후 실행될 명령어의 주소를 갖는다고 한다. 영어 표현으로는 During execution, PC does not contain the address of the currently executing instruction. The address of the currently executing instruction is typically PC-8 for ARM, or PC-4 for Thumb. 라는 표현이 있는데 정확히 왜 명령어가 2개 실행된 후 실행될 명령어의 주소를 갖는지는 모르겠다.

lr : 함수가 호출된 뒤 리턴할 주소를 가지고 있는 레지스터이다. 즉 함수를 호출하는 부분에서 함수 호출 다음의 명령어 주소를 갖게된다.

문제 풀이(이론)

# [pwnable.kr WriteUp]

이태규(ltg2030)

1. 우선 http://pwnable.kr/bin/leg.asm 를 다운받아 확인했다.

2.

```
(gdb) disass main
Dump of assembler code for function main:
   0x00008d3c <+0>:     push   {r4, r11, lr}
   0x00008d40 <+4>:     add    r11, sp, #8
   0x00008d44 <+8>:     sub    sp, sp, #12
   0x00008d48 <+12>:    mov    r3, #0
   0x00008d4c <+16>:    str    r3, [r11, #-16]
   0x00008d50 <+20>:    ldr    r0, [pc, #104]  ; 0x8dc0 <main+132>
   0x00008d54 <+24>:    bl     0xfb6c <printf>
   0x00008d58 <+28>:    sub    r3, r11, #16
   0x00008d5c <+32>:    ldr    r0, [pc, #96]   ; 0x8dc4 <main+136>
   0x00008d60 <+36>:    mov    r1, r3
   0x00008d64 <+40>:    bl     0xfbd8 <__isoc99_scanf>
   0x00008d68 <+44>:    bl     0x8cd4 <key1>
   0x00008d6c <+48>:    mov    r4, r0
   0x00008d70 <+52>:    bl     0x8cf0 <key2>
   0x00008d74 <+56>:    mov    r3, r0
   0x00008d78 <+60>:    add    r4, r4, r3
   0x00008d7c <+64>:    bl     0x8d20 <key3>
   0x00008d80 <+68>:    mov    r3, r0
   0x00008d84 <+72>:    add    r2, r4, r3
   0x00008d88 <+76>:    ldr    r3, [r11, #-16]
   0x00008d8c <+80>:    cmp    r2, r3
   0x00008d90 <+84>:    bne    0x8da8 <main+108>
   0x00008d94 <+88>:    ldr    r0, [pc, #44]   ; 0x8dc8 <main+140>
   0x00008d98 <+92>:    bl     0x1050c <puts>
   0x00008d9c <+96>:    ldr    r0, [pc, #40]   ; 0x8dcc <main+144>
   0x00008da0 <+100>:   bl     0xf89c <system>
   0x00008da4 <+104>:   b      0x8db0 <main+116>
   0x00008da8 <+108>:   ldr    r0, [pc, #32]   ; 0x8dd0 <main+148>
   0x00008dac <+112>:   bl     0x1050c <puts>
   0x00008db0 <+116>:   mov    r3, #0
   0x00008db4 <+120>:   mov    r0, r3
   0x00008db8 <+124>:   sub    sp, r11, #8
   0x00008dbc <+128>:   pop    {r4, r11, pc}
   0x00008dc0 <+132>:   andeq  r10, r6, r12, lsl #9
   0x00008dc4 <+136>:   andeq  r10, r6, r12, lsr #9
   0x00008dc8 <+140>:           ; <UNDEFINED> instruction: 0x0006a4b0
   0x00008dcc <+144>:           ; <UNDEFINED> instruction: 0x0006a4bc
   0x00008dd0 <+148>:   andeq  r10, r6, r4, asr #9
End of assembler dump.
(gdb) disass key1
Dump of assembler code for function key1:
```

```
   0x00008cd4 <+0>:     push    {r11}           ; (str r11, [sp, #-4]!)
   0x00008cd8 <+4>:     add     r11, sp, #0
   0x00008cdc <+8>:     mov     r3, pc
   0x00008ce0 <+12>:    mov     r0, r3
   0x00008ce4 <+16>:    sub     sp, r11, #0
   0x00008ce8 <+20>:    pop     {r11}           ; (ldr r11, [sp], #4)
   0x00008cec <+24>:    bx      lr
End of assembler dump.
(gdb) disass key2
Dump of assembler code for function key2:
   0x00008cf0 <+0>:     push    {r11}           ; (str r11, [sp, #-4]!)
   0x00008cf4 <+4>:     add     r11, sp, #0
   0x00008cf8 <+8>:     push    {r6}            ; (str r6, [sp, #-4]!)
   0x00008cfc <+12>:    add     r6, pc, #1
   0x00008d00 <+16>:    bx      r6
   0x00008d04 <+20>:    mov     r3, pc
   0x00008d06 <+22>:    adds    r3, #4
   0x00008d08 <+24>:    push    {r3}
   0x00008d0a <+26>:    pop     {pc}
   0x00008d0c <+28>:    pop     {r6}            ; (ldr r6, [sp], #4)
   0x00008d10 <+32>:    mov     r0, r3
   0x00008d14 <+36>:    sub     sp, r11, #0
   0x00008d18 <+40>:    pop     {r11}           ; (ldr r11, [sp], #4)
   0x00008d1c <+44>:    bx      lr
End of assembler dump.
(gdb) disass key3
Dump of assembler code for function key3:
   0x00008d20 <+0>:     push    {r11}           ; (str r11, [sp, #-4]!)
   0x00008d24 <+4>:     add     r11, sp, #0
   0x00008d28 <+8>:     mov     r3, lr
   0x00008d2c <+12>:    mov     r0, r3
   0x00008d30 <+16>:    sub     sp, r11, #0
   0x00008d34 <+20>:    pop     {r11}           ; (ldr r11, [sp], #4)
   0x00008d38 <+24>:    bx      lr
End of assembler dump.
```

key1, key2, key3 함수 호출 후 r0 값을 모두 더한 값을 알아내는 것이 중요하다.

따라서 먼저 key1에서 r0의 변화를 보면

```
   0x00008cdc <+8>:     mov     r3, pc
   0x00008ce0 <+12>:    mov     r0, r3
   0x00008ce4 <+16>:    sub     sp, r11, #0
```

이므로 key1()을 호출한 뒤 r0 값은 0x00008ce4 이다.

# [pwnable.kr WriteUp]

이태규(ltg2030)

key2에서 r0는

```
   0x00008d04 <+20>:    mov     r3, pc
   0x00008d06 <+22>:    adds    r3, #4
   0x00008d08 <+24>:    push    {r3}
   0x00008d0a <+26>:    pop     {pc}
   0x00008d0c <+28>:    pop     {r6}            ; (ldr r6, [sp], #4)
   0x00008d10 <+32>:    mov     r0, r3
```

이렇게 변하므로 0x00008d08+4 의 값을 갖고

key3에서 r0는

```
   0x00008d28 <+8>:     mov     r3, lr
   0x00008d2c <+12>:    mov     r0, r3
```

즉 key3의 lr 값을 갖는 것을 알 수 있는데 key3의 lr값을 확인하면

```
   0x00008d7c <+64>:    bl      0x8d20 <key3>
   0x00008d80 <+68>:    mov     r3, r0
```

로 0x00008d80 임을 알 수 있다. 따라서 이 셋을 더한 108400을 입력하면 flag를 얻을 수 있다.

문제 풀이(실습)

```
/ $ ls -l
total 628
drwxrwxr-x    2 root     0                0 Nov 10  2014 bin
drwxrwxr-x    2 root     0                0 Nov 10  2014 boot
drwxrwxr-x    2 root     0                0 Nov 10  2014 dev
drwxrwxr-x    3 root     0                0 Nov 10  2014 etc
-r--------    1 1001     0               38 Nov 10  2014 flag
---s--x---    1 1001     1000        636419 Nov 10  2014 leg
lrwxrwxrwx    1 root     0               11 Nov 10  2014 linuxrc -> bin/busybo
dr-xr-xr-x   32 root     0                0 Jan  1 00:00 proc
drwxrwxr-x    2 root     0                0 Nov 10  2014 root
drwxrwxr-x    2 root     0                0 Nov 10  2014 sbin
drwxrwxr-x    2 root     0                0 Nov 10  2014 sys
drwxrwxr-x    4 root     0                0 Nov 10  2014 usr
/ $ ./leg
Daddy has very strong arm! : 108400
Congratz!
My daddy has a lot of ARMv5te muscle!
```

Congratz! 가 출력된 뒤 flag를 cat 하기 때문에 My daddy has a lot of ARMv5te muscle! 이 flag임을 알 수 있다.

실행화면