

>> easy\_1 풀이

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    int result; // eax@9
    int v4; // edx@9
    int read_value; // [sp+18h] [bp-38h]@1
    int dummy1; // [sp+1Ch] [bp-34h]@2
    _pid_t pid; // [sp+20h] [bp-30h]@1
    int input_pid; // [sp+24h] [bp-2Ch]@4
    __int16 read_value_address; // [sp+2Eh] [bp-22h]@4
    int v10; // [sp+4Ch] [bp-4h]@1

    v10 = *MK_FP(__GS__, 20);
    pipe(&read_value);
    pid = fork();
    if ( !pid )
    {
        puts("WnOMG!!!! I forgot kid's id");
        write(dummy1, "69800876143568214356928753", 0x1Du);
        puts("Ready to exit ");
        exit(0);
    }
    read(read_value, &read_value_address, 0x1Du);
    __isoc99_scanf("%d", &input_pid);
    if ( input_pid == pid )
    {
        if ( (*(_DWORD *)((char *)lol + 3) & 0xFF) == 204 )
        {
            puts(":D");
            exit(1);
        }
        printf("WnYou got the keyWn ");
        lol((int)&read_value_address);
    }
    wait(0);
    result = 0;
    v4 = *MK_FP(__GS__, 20) ^ v10;
    return result;
}

pwndbg> checksec
^[[B[*] Checking for new versions of pwntools
To disable this functionality, set the contents of /home/kimyongsu/.pwntool
s-cache/update to 'never'.
[*] A newer version of pwntools is available on pypi (3.6.1 --> 3.8.0).
Update with: $ pip install -U pwntools
[*] '/home/kimyongsu/Desktop/\xea\x87\xb8\xeb\x83\xa5 \xed\x8f\xb4\xeb\x8d\x94/
easy_1'
Arch: i386-32-little
RELRO: Partial RELRO
Stack: Canary found
NX: NX enabled
PIE: No PIE (0x8048000)
```

문제의 main문과 보호 기법은 위와 같다.

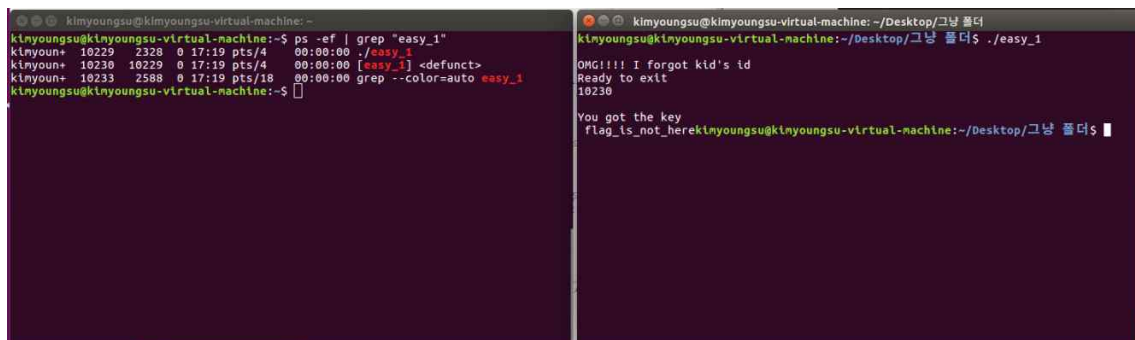
NX, 까나리, RELRO가 걸려있다. partial RELRO는 got overwrite가 먹힌다.

main의 있는 lol 함수를 보도록 한다.

```
int __cdecl lol(int read_value_address)
{
    char v2; // [sp+15h] [bp-13h]@1
    char v3; // [sp+16h] [bp-12h]@1
    char v4; // [sp+17h] [bp-11h]@1
    char v5; // [sp+18h] [bp-10h]@1
    char v6; // [sp+19h] [bp-Fh]@1
    char v7; // [sp+1Ah] [bp-Eh]@1
    char v8; // [sp+1Bh] [bp-Dh]@1

    v2 = 2 * *(_BYTE *)(read_value_address + 1);
    v3 = *(_BYTE *)(read_value_address + 4) + *(_BYTE *)(read_value_address + 5);
    v4 = *(_BYTE *)(read_value_address + 8) + *(_BYTE *)(read_value_address + 9);
    v5 = 2 * *(_BYTE *)(read_value_address + 12);
    v6 = *(_BYTE *)(read_value_address + 18) + *(_BYTE *)(read_value_address + 17);
    v7 = *(_BYTE *)(read_value_address + 10) + *(_BYTE *)(read_value_address + 21);
    v8 = *(_BYTE *)(read_value_address + 9) + *(_BYTE *)(read_value_address + 25);
    return printf("flag_is_not_here");
}
```

별다른 것은 없고 read\_value\_address로부터 떨어진 값을 변수에 넣고 flag\_is\_not\_here이  
란 문구를 출력한다. 먼저 실행을 시켜서 어떤 식인지 보도록 한다.



```
kimyoungsu@kimyoungsu-virtual-machine: ~$ ps -ef | grep easy_1
kimyoungsu 10229 2328 0 17:19 pts/4 00:00:00 ./easy_1
kimyoungsu 10230 10229 0 17:19 pts/4 00:00:00 [easy_1] <defunct>
kimyoungsu 10233 2588 0 17:19 pts/18 00:00:00 grep --color=auto easy_1
kimyoungsu@kimyoungsu-virtual-machine: ~$

kimyoungsu@kimyoungsu-virtual-machine: ~/Desktop/그냥 폴더$ ./easy_1
OMG!!!! I forgot kid's id
Ready to exit
10230
You got the key
flag_is_not_herekimyoungsu@kimyoungsu-virtual-machine: ~/Desktop/그냥 폴더$
```

ps -ef로 pid를 찾고 그 값을 넣어보았다. 근데 흠... 진짜 flag가 안 뜬다. (뜨면 이상할 듯)  
flag\_is\_not\_here을 보니 도저히 포너블 문제가 아니고 코드 패치 문제 같아서 리버싱 문제  
로 생각해보았다. 그래서 main이랑 lol을 디스어셈블해서 분기점마다 보기로 했다.  
main에서는 딱히 특이한 점이 보이지는 않았는데 lol의 디스어셈블을 보니 이상한 점을 발견  
할 수 있었다.

```

8696      mov     eax, [ebp+read_value_address]
8699      add     eax, 9
869C      movzx   eax, byte ptr [eax]
869F      mov     edx, eax
86A1      mov     eax, [ebp+read_value_address]
86A4      add     eax, 19h
86A7      movzx   eax, byte ptr [eax]
86AA      lea     eax, [edx+eax]
86AD      mov     [ebp+var_D], al
86B0      mov     [ebp+var_C], 0
86B7      cmp     [ebp+var_C], 1
86BB      jnz     short loc_80486D3
86BD      mov     eax, offset format ; "%s"
86C2      lea     edx, [ebp+var_13]
86C5      mov     [esp+4], edx
86C9      mov     [esp], eax ; format
86CC      call    _printf
86D1      jmp     short locret_80486E0

```

저 부분을 보면 ebp+var\_C의 값을 0으로 바꿔주고 1과 비교하는데 jnz는 ZF가 0이 아닐 때 실행이 된다. 근데 0과 1을 비교하면 ZF가 언제나 0이 아니기 때문에 계속 실행되는데 아마 이 부분을 코드 패치하면 풀 수 있을 것 같다. 색칠한 부분 중 mov 부분을 hex로 보면

```

080486A0  C2 8B 45 08 83 C0 19 0F
080486B0  C7 45 F4 00 00 00 00 83
080486C0  04 08 8D 55 ED 89 54 24
080486D0  CC CB AD D0 C2 00 A1 A0

```

이렇게 나오는데 00 00 00 00 이 부분이 아마도 뒤에 mov A, B 중 B의 operand인 것 같다. 인터넷에 찾아보니 그렇다고 한다. 이제 hxd에서 00 부분을 01로 바꾸고 실행을 시켜보기로 한다.

```

000006A0  C2 8B 45 08 83 C0 19 0F
000006B0  C7 45 F4 01 00 00 00 83
000006C0  04 08 8D 55 ED 89 54 24

```

이제 이 파일을 실행시키면 ebp+var\_C에 1을 넣고 1과 비교하니 아마도 성공할 것 같다. 이제 처음과 같이 실행을 하면??

```

kimyoungsu@kimyoungsu-virtual-machine:~$ ps -ef | grep "easy_1"
root      2536   2379  0 21:34 pts/4    00:00:00 sudo ./easy_1
root      2537   2536  0 21:34 pts/4    00:00:00 ./easy_1
root      2538   2537  0 21:34 pts/4    00:00:00 [easy_1] <defunct>
kimyoung+ 2573   2561  0 21:35 pts/18    00:00:00 grep --color=auto easy_1
kimyoungsu@kimyoungsu-virtual-machine:~$

kimyoungsu@kimyoungsu-virtual-machine: /mnt/hgfs/vm_shared
rm: 쓰기 보호된 디렉토리 'cdrom'을(를) 지울까요? y
rm: 'cdrom'을 지울 수 없음: 허가 거부
kimyoungsu@kimyoungsu-virtual-machine:/mnt$ sudo rm -r cdrom
kimyoungsu@kimyoungsu-virtual-machine:/mnt$ ls
hgfs
kimyoungsu@kimyoungsu-virtual-machine:/mnt$ cd hgfs/
kimyoungsu@kimyoungsu-virtual-machine:/mnt/hgfs$ ls
kimyoungsu@kimyoungsu-virtual-machine:/mnt/hgfs$ cd vm_shared/
kimyoungsu@kimyoungsu-virtual-machine:/mnt/hgfs/vm_shared$ ls
easy_1
kimyoungsu@kimyoungsu-virtual-machine:/mnt/hgfs/vm_shared$ ls -al
lrwxrwxrwx 1 root root 0 8월 5 21:20 .
dr-xr-xr-x 1 root root 4192 8월 5 21:34 ..
-rwxrwxrwx 1 root root 7540 8월 5 19:17 easy_1
kimyoungsu@kimyoungsu-virtual-machine:/mnt/hgfs/vm_shared$ sudo ./easy_1
OMG!!!! I forgot kid's ld
Ready to exit
2538
You got the key
rhe1heg@kimyoungsu@kimyoungsu-virtual-machine:/mnt/hgfs/vm_shared$

```

rhe1heg하면서 짚린 키 값이 나오는데 성공한 것 같다. 클리어!