



random

ssh로 먼저 접속해보자!

```
random@pwnable:~$ ls -al
total 40
drwxr-x---  5 root      random 4096 Oct 23  2016 .
drwxr-xr-x 116 root      root    4096 Nov 11 14:52 ..
d-----  2 root      root    4096 Jun 30  2014 .bash_history
-r--r----  1 random_pwn root     49 Jun 30  2014 flag
dr-xr-xr-x  2 root      root    4096 Aug 20  2014 .irssi
drwxr-xr-x  2 root      root    4096 Oct 23  2016 .pwntools-cache
-r-sr-x---  1 random_pwn random 8538 Jun 30  2014 random
-rw-r--r--  1 root      root     301 Jun 30  2014 random.c
random@pwnable:~$
```

접속해서 무슨 파일이 있는지 살펴보니 flag, random, random.c 파일이 보이는데 flag는 권한이 없어서 못 열어본당!

random.c을 살펴보자!

```
#include <stdio.h>

int main(){
    unsigned int random;
    random = rand();          // random value!

    unsigned int key=0;
    scanf("%d", &key);

    if( (key ^ random) == 0xdeadbeef ){
        printf("Good!\n");
        system("/bin/cat flag");
        return 0;
    }

    printf("Wrong, maybe you should try 2^32 cases.\n");
    return 0;
}
```

random 변수에 무작위 값을 상속받고, key를 입력받는데,
key와 random을 xor 했을때 그 값이 0xdeadbeef이면 system함수를 실행시켜준다!

gdb로 어떤 값을 random에 저장하는 지 살펴보자!

```
(gdb) disas main
Dump of assembler code for function main:
   0x00000000004005f4 <+0>:    push    rbp
   0x00000000004005f5 <+1>:    mov     rbp, rsp
   0x00000000004005f8 <+4>:    sub     rsp, 0x10
   0x00000000004005fc <+8>:    mov     eax, 0x0
   0x0000000000400601 <+13>:   call    0x400500 <rand@plt>
   0x0000000000400606 <+18>:   mov     DWORD PTR [rbp-0x4], eax
   0x0000000000400609 <+21>:   mov     DWORD PTR [rbp-0x8], 0x0
   0x0000000000400610 <+28>:   mov     eax, 0x400760
   0x0000000000400615 <+33>:   lea     rdx, [rbp-0x8]
   0x0000000000400619 <+37>:   mov     rsi, rdx
   0x000000000040061c <+40>:   mov     rdi, rax
   0x000000000040061f <+43>:   mov     eax, 0x0
   0x0000000000400624 <+48>:   call    0x4004f0 <__isoc99_scanf@plt>
   0x0000000000400629 <+53>:   mov     eax, DWORD PTR [rbp-0x8]
   0x000000000040062c <+56>:   xor     eax, DWORD PTR [rbp-0x4]
   0x000000000040062f <+59>:   cmp     eax, 0xdeadbeef
   0x0000000000400634 <+64>:   jne     0x400656 <main+98>
   0x0000000000400636 <+66>:   mov     edi, 0x400763
   0x000000000040063b <+71>:   call    0x4004c0 <puts@plt>
   0x0000000000400640 <+76>:   mov     edi, 0x400769
   0x0000000000400645 <+81>:   mov     eax, 0x0
   0x000000000040064a <+86>:   call    0x4004d0 <system@plt>
   0x000000000040064f <+91>:   mov     eax, 0x0
   0x0000000000400654 <+96>:   jmp     0x400665 <main+113>
   0x0000000000400656 <+98>:   mov     edi, 0x400778
   0x000000000040065b <+103>:  call    0x4004c0 <puts@plt>
   0x0000000000400660 <+108>:  mov     eax, 0x0
   0x0000000000400665 <+113>:  leave
   0x0000000000400666 <+114>:  ret
```

disas main

<main + 8 ~ +18>이 random에 값을 넣어주는 부분이다.

<main + 18>을 보면 결과 값을 담고 있는 eax레지스터를 [rbp-0x4]로 넘겨주는 것을 보니 random은 [rbp-0x4]에 저장되어 있는 것을 알수있다.

그럼 실행시켜 어떤 값을 받는지 알아보자!

```
(gdb) b *main+21
Breakpoint 1 at 0x400609
(gdb) r
Starting program: /home/random/random

Breakpoint 1, 0x000000000400609 in main ()
(gdb) x/x $rbp-0x4
0x7ffc8d89056c: 0x6b8b4567
(gdb)
```

<main+21>에 break point를 걸고
실행시킨 다음
[rbp-0x4]에 있는 값을 확인하니
0x6b8b4567이 들어있다.
그럼 우리는 key값으로
0xdeadbeef ^ 0x6b8b4567 ==
0xb526fb88을 입력해주면 되는데
int로 받으니까 3039230586을 입력해주면
된다.

계산기	
≡ 프로그래머	
6B8B4567 XOR DEADBFFF =	
B526 FB88	
HEX	B526 FB88
DEC	3,039,230,856
OCT	26 511 575 610
BIN	1011 0101 0010 0110 1111 1011 1000 1000

마저 실행해보았다!

```
(gdb) c
Continuing.
3039230856
Good!
/bin/cat: flag: Permission denied
[Inferior 1 (process 622) exited normally]
(gdb)
```

if문 안으로 들어와서 Good!은 표시해주는데 flag는 permission denied된다..

하지만 괜찮다! rand함수는 seed를 설정할 수 없어서 몇번을 해도 똑같은 값을 주기때문에
gdb 바깥으로 나와서 3039230856을 입력해보자!

```
random@pwnable:~$ ./random
3039230856
Good!
Mommy, I thought libc random is unpredictable...
random@pwnable:~$
```

flag 값을 알 수 있었다~

