Computer Security: hw3

學號：b04902053 姓名：鄭淵仁

flag: `FLAG{CAN_YOU_R34D_MY_M1ND?}`

write-up

程式的漏洞

`readme` 這支程式在一開始開了一個大小是 `0x20` 的 array，但是之後卻讀了 `0x30` bytes 進去這個 array，所以可以利用這裡來做 stackoverflow。另外，這支程式的 `RELRO` 的保護是 partial 的，所以可以改 `.got.plt` 的值。

怎麼寫入 rop chain

首先觀察 assembly code，發現在 `main` 裡面有下面這段 gadget，可以用來寫入 6 個 address 到 `rbp - 0x20`。

```
lea    rax,[rbp-0x20]
mov    edx,0x30
mov    rsi,rax
mov    edi,0x0
call   0x4004c0 <read@plt>
...
leave
ret
```

在寫入的 6 個 address 當中，倒數第 2 個 address 會被 pop 到 `rbp`，最後 1 個 address 會被當作 return address。所以可以利用最後 2 個 address 來重覆執行這段 gadget，也就可以不斷寫入 rop。

但是為了把寫入的 rop 接在一起，又不能蓋到還沒被 pop 掉的倒數第 2 個 address，所以要每次都先寫到別的位置，等那個 address 被 pop 掉之後，再回來繼續寫。所以我在 bss 後段除了找了一段 `buf` 來寫入 rop，也另外找了一段 `tmp_buf`，當作可以暫時亂寫的位置。

另外，由於這個部分的步驟很多而且要重複執行，所以我寫了一個 function 來自動化「寫入一次 `rop`，再去 `tmp_buf` 寫一些資料」這個操作。（就是 python script 裡面的 `add_4_rop()`）

有了寫入 rop 的方法之後，接下來要串出可以開 shell 的 rop。

要在 rop chain 裡面寫的內容

首先觀察到在 `libc` 的 `read` 裡面有一個 `syscall` 可以當作 gadget：

```
Dump of assembler code for function read:
   0x00000000000f7220 <+0>:     ...
   ...
   0x00000000000f722e <+14>:    syscall
   ...
   0x00000000000f7238 <+24>:    ret
```

而且 `read` 的 offset 和裡面的 `syscall` 的 offset 只有差 `0xe` ，另外雖然 `libc` 的位置可能會移動，但是後面 3 位的數字是不會變的。所以可以把 `read@plt` 的值改寫成這個 `syscall` 的位置，讓 `read` 變成 `syscall` ，之後就可以自由的 call 自己想要 call 的 syscall 了。

接下來，為了 call `syscall` ，要能改到 `rdx` 的值，這時候可以用 `__libc_csu_init` 裡面的這 2 段 gadget 來改，還可以順便 `syscall` 。

```
400690:|  4c 89 ea            |  mov    rdx,r13
400693:|  4c 89 f6            |  mov    rsi,r14
400696:|  44 89 ff            |  mov    edi,r15d
400699:|  41 ff 14 dc         |  call   QWORD PTR [r12+rbx*8]
40069d:|  48 83 c3 01         |  add    rbx,0x1
4006a1:|  48 39 eb            |  cmp    rbx,rbp
4006a4:|  75 ea               |  jne    400690 <__libc_csu_init+0x40>
4006a6:|  48 83 c4 08         |  add    rsp,0x8
```

```
4006aa:|  5b                  |  pop    rbx
4006ab:|  5d                  |  pop    rbp
4006ac:|  41 5c               |  pop    r12
4006ae:|  41 5d               |  pop    r13
4006b0:|  41 5e               |  pop    r14
4006b2:|  41 5f               |  pop    r15
4006b4:|  c3                  |  ret
```

最後，只要能把 `rax` 的值改成 `0x3b` ，就可以 call `execve` 來執行 `/bin/sh` 。但是，因為沒有修改 `rax` 的值的 gadget 可以用，所以只能利用 `syscall` 的回傳值來改 `rax` 。

但是如果剛剛在用 `read` 修改 `read@plt` 的值的時候，一次讀 `0x3b` 個 bytes 進來，同時讓最後一個 byte 蓋到 `read@plt` 的最後一個 byte 的話，那就要寫到前面不可寫的區域，這樣是不可行的。

所以，我改成先只 `read` 一個 byte 進來，讓 `rax` 變成 `1` ，這樣一來 `read` 會變成 `write` 。接下來再 `write` `0x3b` 個 bytes 出去，就可以讓 `rax` 變成 `0x3b` ，這樣一來 `read` 就會變成 `execve` ，最後就可以成功呼叫 `execve('/bin/sh', 0, 0)` 了！

script

- 可以取得 shell 的 python3 檔案： `1.py`
  (requirements: pwntools)