CS: hw5 - baby_heap_revenge

學號：b04902053 姓名：鄭淵仁

flag: `FLAG{YOUARENOTBABYATALL}`

script

- 可以取得 shell 的 python3 script： `baby_heap_revenge.py`

  (requirements: `pwntools`)

write-up

程式的漏洞

`baby_heap_revenge` 在 `allocate_heap()` 的時候， `malloc(size)` ，卻 `read(size+8)` 進來，也就是有 **heap overflow**。

摘要

利用 `malloc` 裡面的 `_int_free` leak 出 top chunk 的 address，再利用 house of force leak 出 `libc` 和 `stack` 的 address，就可以推算出 `read_input` 的 return address 。最後把這個 return address 蓋掉改成 `rop_gadget` 的 address，就可以拿到 shell。

使用 `malloc` call `_int_free` 的原理

首先，從 glibc 的 `malloc.c` 可以觀察到：當要 allocate 比 top chunk 的 size 還要大，並且比 `mmap` 的 threshold 要小的 chunk size 時，會先把原本的 top chunk `_int_free` 掉，並分配一塊比較大的 top chunk 出來接在原本的 top chunk 的後面。

而要做到以上的動作之前，會先檢查下列四點：

- 原本的 top chunk 的 size 是否大於 `MINISIZE` (`0x10`)
- `prev_inuse` 是不是 `1` 。
- 要 allocate 的 size 是不是比剩下的 top chunk 的 size 還要大
- `old_top + oldsize` 要 align 一個 page.

source code:

```
assert ((old_top == initial_top (av) && old_size == 0) ||
        ((unsigned long) (old_size) >= MINSIZE &&
         prev_inuse (old_top) &&
         ((unsigned long) old_end & (pagesize - 1)) == 0));
assert ((unsigned long) (old_size) < (unsigned long) (nb + MINSIZE));
```

所以只要製造出符合上述四點的情況，就可以使用 `malloc` 呼叫 `_int_free` 。

使用上述過程來 leak 出 `heap` 的 address

所以我利用上述過程來 `_int_free` 掉一塊 chunk，再 allocate 那一塊 chunk，就可以從那一塊 chunk 裡面 leak 出 `heap` 的 address 。實作細節如下：

首先用 heap overflow 把 top chunk size 改成 `0xfd1` （沒蓋掉的話會是 `0x20fd1` ），再 allocate `0x1000` ，這樣一來就會把原本的 top chunk `_int_free` 掉並放到 unsorted bin 裡面，再產生新的 top chunk 接在後面。

接下來 allocate large bin 的 size 來讓 `malloc` 從 unsorted bin 裡面切一塊出來，就可以從這一塊 trunk 裡面 leak 出一些資訊。而因為切出來之後，會先把這一塊 chunk 放到 large bin，再分配出來，所以 chunk 裡面會記錄上、下一塊 large bin 的 address ，就可以推算出 top chunk 的 address。

> 在這裡有時候 address 裡面有 null byte (`0x00`)，而 leak 到錯誤的 address。如果遇到這樣的情況，只要再執行一兩次一樣可以正確執行。

leak 出 `libc` 和 `stack` 的 address

算出 top chunk 的 address 之後，使用 house of force 移動到 `bss` 段的 `stdin@@GLIBC` (`0x602030`)，然後 print 出他的 address ，就可以推算出 `libc` 的位置。

有了 `libc` 的 address 之後，一樣使用 house of force 移動到 `environ` 就可以 print 出 `stack` 的 address。

overwrite `read_input` 的 return address

有了 `stack` 的 address 之後，就可以推算出 `read_input` 的 return address 在哪裡。再把它改成 `one_gadget` ，這樣一來在 `read` 完這一段蓋掉 `read_input` 的 return address 的 char 之後，就會執行 `one_gadget` ，就可以拿到 shell 了。