Silver Bullet - pwnable.tw

flag

```
FLAG{uS1ng_S1lv3r_bu1l3t_7o_Pwn_th3_w0rld}
```

Solution

overflow 到 return address

觀察完 assembly 後可以整理出一個 `struct bullet` ：

```
struct bullet {
  char desc[48];
  int power;
};
```

另外 `man 3 strncat` 有下面這一段：

> If src contains n or more bytes, strncat() writes n+1 bytes to dest (n from src plus the terminating null byte). Therefore, the size of dest must be at least strlen(dest)+n+1.

所以可以利用上述現象來 overflow bullet 的 desc，方法如下：

1. `create` 47 byte 的 `bullet` 。
2. `power_up` 1 byte，這時候 overflow 的 1 byte 會蓋到 `bullet.power` ，讓原本的 power 變成 0，之後會再加上這次的 strlen 變成 1。
3. `power_up` 47 byte，這裡新的 `desc` 會被接到 `bullet.power` 的後面，所以就可以讓 power 變得很大，順便蓋到 return address，也就可以串 ROP 了。

串 ROP

這裡使用 stack migration，也就是利用 `leave return` 來搬移 stack 的 address。

在每一次 migration 的最後，可以利用 `read_input` 來讀新的 ROP 到另一塊 stack 上。

所以在 overflow 完之後第一次 ROP 先放 `read_input` 讀 ROP 到 bss 的那一塊 stack 上，再 `leave return` 到剛 `read_input` 的 address。

接下來就可以串 ROP 了：

1. `puts` `puts@GOT` ，這樣一來就可以 leak 出 `system` 的 address。
2. `system('/bin/sh')`

這樣一來就可以拿到 shell 了。

script

- 可以拿到 shell 的 python3 script： `silver_bullet.py`