Filenote:

1. 首先,把 File 的 fileno 寫成 stdout
2. 因為目前 fp 的 FILE 上面沒有 buffer,先 fwrite 一次讓 sys 幫我們 allocate write_buf_base,因為我們沒有 heap 的 address 也不行自己寫入 buffer address
3. 我們使用 gets 送 payload 把 write_buf_base 的 LSB 變成\x00,也就是說可以多輸出原本 buffer 指向的位置前面的位置,所以如果前面有 libc 的 address (像是 File struct 本身的就有 vtable 的 pointer 會指向 libc),我們就會得到 libc 的 address

   並且這邊還有一個重點就是因為我們不知道 heap 的 address,所以我們沒辦法透過 read_end == write_base 的方法規避 do_write 裡面的檢查,所以我們可以設立 flag _IO_IS_APPENDING,避免進入檢查,如下圖

```
if (fp->_flags & _IO_IS_APPENDING)
  fp->_offset = _IO_pos_BAD;
else if (fp->_IO_read_end != fp->_IO_write_base
  {
    off64_t new_pos
= _IO_SYSSEEK (fp, fp->_IO_write_base - fp->_IO_read_end, 1);
    if (new_pos == _IO_pos_BAD)
return 0;
    fp->_offset = new_pos;
```

4. 接下來是寫 gadget 到_IO_file_overflow

   我們可以透過 fwrite(note_buf,, fp) 是將 note_buf 裡面的東西寫到 fp,並且中間會經過 fp 的 write_buf,所以我們先用 gets 把 fp 的 IO_write_base,IO_write_ptr 指到_IO_file_overflow 的開頭,在將 IO_write_end 指到_IO_file_overflow 的結尾,此時我們再用 gets 得到 execve 這個 gadget,並且使用 fwrite 經過上面提到的機制寫到_IO_file_overflow,並且這邊要注意這個 gadget 的要求要滿足,我們可以看執行到 fwrite 的時候時候 register 的情況,可以發現其中一個需要滿足的 register 會在 gets 寫的到的地方,所以我們 gets 就將那編寫成\x00 就可以了 ,fwrite 完自動會執行_IO_file_overflow,就可以讓我們拿到 shell
5. 解決 Remote 不能動問題:

   在上面 get libc 的方法找不到東西,猜是因為 remote 的 buffer 的 mode 跟我們 local 不太一樣,可能是_IOFBF,遇到\n 也不會寫出來(事實上,如果加上\n 在裡面的話會 EOF,不知道為啥),需要把它填滿才會出來,且 remote 的 system 在 allocate write_buffer 的時候 write_buffer_end 設比較大,造成一次 fwrite 填不滿這塊 buffer,我們就一值 call fwrite 直到有東西噴出來