題目:sandbox
new_code_buf 有設置執行的權限
觀察 code,可以發現要寫進去 new_code_buf 的 code 會把 syscall 和 call register 都替換掉, 但是可以發現他最後 copy 進去 new_code_buf 的 epilogue 裡面含有 0f05 這個 syscall 的 pattern,我們可以利用
shellcode 前面準備好參數,之後 jmp 到 epilogue 的 pttern 那邊就可以成功的執行 syscall 取得 shell

題目:fullchain-nert
有開 PIE 和 ASLR

0. 首先可以看到有執行的次數限制,不過我們可以輕易地使用 buffer overflow 的技巧把次數限制改掉。
1. Leak code base address:
   想要做 ROP,就必須得到 code 真實的位置,因為有開 PIE,所以我們需要先找到 code segment 的 base address,我們可以透過 FSB 的%p leak 出 global 這個 buffer 的位置,扣掉 global 的 offset 就得到 code 的 base address
2. Leak libc:
   seccomp_rule 擋掉了一些 syscall,看來是要用 open read write 的那個 syscall 去做 ROP,並且實際在串 ROP 的時候用 ROPgadget 發現少了很多 gadget 尤其是最重要的 syscall,所以我們想要使用 libc 裡面的 gadget 來串 ROP。
   問題來了,因為有開 ASLR 所以 libc 每次的 address 都是不固定的,所以我們需要 leak libc 的位置,這可以使用 puts@plt(puts@got)這個技巧把 plt 給 leak 出來,具體作法也是 ROP,使用 pop_rdi,puts_got,puts_plt 來讓 puts 印出 got puts 的位置,此時因為先 call 了一次 puts,所以印出來的是真實的 puts 位址,再扣掉 puts 在 library 裡面的 offset 就是 library 的 base address。
   要注意的是,要做 ROP 就需要從 function 裡面 return,我們從 chal()裡面 return 觸發了 ROP,但是做完 ROP 後我們還想要回來 chal(),只要在 ROP 最後串上一個 chal 的位址就可以了,因為 puts 會 call ret, ret 執行 pop rip,chal()的 address 就會被 load 到 rip
3. 串 open read write 的 ROP
   可以發現,buffer overflow 的長度太短,不足以塞下 ROP,所以我們需要做 stack pivoting, 首先先 leak 出 stack 的位址,使用 FSB 去 leak 出 local 這個 buffer 的位址,然後取 stack 上的新的一塊位址去放主要的 ROP chain,然後使用 libc 裡面的 I/O function 寫 ROP chain 到這塊 stack 上,最後在 local 那邊做 stack pivoting 移轉到這塊新的 stack 上,執行完 ROP chain 就可以得到 flag 了。
   不過我在做這邊的時候出問題了,就是我打算使用 libc 裡面的 gets 來將 open read write 的 ROP 寫到那塊新的 stack 上面,問題是我在串 gets 的 ROP 的時候不管怎麼用程式都沒有反應,原本想改成 scanf,不過發現 scanf 的 ROP 太長會

超過 overflow 的長度,最後就沒有做出來了。

題目: fullchain
此題沒時間看,不過簡短看了一下可以發現 buffer overflow 基本上都沒有發揮餘地

Ref: retlibc https://tech-blog.cymetrics.io/posts/crystal/pwn-intro-2/