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-nerf
這題我的 script 要多 run 幾次 flag 才會出來
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 ASLR,所以我們需要先找到 code segment 的 base address,我們可以透過 FSB 的%7$p leak 出 global 這個 buffer 的位置,因為在使用 mywrite 的時候 stack 上面會存 mywrite 的參數,而現在的 case 就是 global 的位址,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 puts 的位置,並且 got,puts@plt 都是我們有了 code 的位址之後加個 offset 就可以得到的,此時因為先 call 了一次 puts,puts 位址已經解析完畢,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,並且就算移動到 global 當作 stack,我們也會因為寫入最多只能寫 0x60 的 byte,要串 open read write ROP 會不夠,所以我們想要先串一個 read 的 ROP,用這個 ROP 讀進 open read write 的 ROP。
   - 首先先 leak 出 stack 的位址,使用 FSB %p 去 leak 出 local 這個 buffer 的位址,因為%p 會把 rsi 寫出來,因為 strncmp 第二個參數是 local,所以我們 call,printf 的時候 rsi 就是 local,這樣就可以 leak 出
     這邊我們先 stack pivoting 把 global 當成新的 stack,寫入 read ROP,讀取真正的 open read write ROP, 然後我們把 open read write ROP 寫在 read ROP 的後面,這樣 read ROP return 之後就可以剛好跳到 open read write ROP 那邊,接下來就可以讀出 flag 了

題目: fullchain

1. 首先把 cnt 寫大一點

- 只有三個 cnt,我們要盡量讓我們每個動作需要的 cnt 數變少

Leak stack address: 如果先寫到 global 在 mywrite 出來,就佔掉兩個 cnt 了
我們可以觀察 line 72 的 scanf 可以吃 10 的 char 並且比對的時候只會比對前幾個 char,我們可以這樣 write%p 來達到 leak 出 local

- FSB 寫 cnt:

FSB 寫值需要 stack 上面有那個值所在的位址,所以我們在 local 先寫入 cnt 的位址,然後再用 FSB 把他改寫。這邊要注意的是因為 cnt 此時只剩下一個,所以我們還是要使用上面那個技巧,write%16$n 把值寫到 cnt,這邊因為 printf 只 output 出 5 個 char,所以 cnt 只會是 5,我們可以再透過寫 fmt 到 global,在 FSB 寫更大的值到 cnt

2. Leak code address:這邊更上面那題的一樣所以不多介紹了
3. Leak libc:
   我們使用 FSB 把 exit_got 寫成 leave_ret 的 address
   使用這個來觸發 ROP chain, 因為 chal 可能因為優化的因素, compiler 看到 exit 就不編譯出 leave ret, 所以我們用這替代。
   我們先使用 FSB 來寫印出 puts address 的 ROP chain(更上一題一樣不多介紹)到 chal()的 return address,然後我們觸法 exit,此時 exit 已經被改寫成 leave;ret;所以就會觸發我們的 rop chain,就可以得到 puts 的 address,回來的時候記得要重新設定 cnt 和 stack 的位址

4. 有了 libc 的 address,我們就可以使用 FSB 把 memset 的 got 改成 gets 的位址,

然後 call 到 memset 的時候我們的第一個參數是 local,因為是 gets 所以可以無限地寫,我們就可以把 open read write 的 ROP 寫到 local 的 return address,寫完之後再次呼叫已經被改成 leave;ret 的 exit,再次觸法 ROP chain,然後就會讀到 flag 了

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