M10915027 石成峰 HW9

[shelllab]

1. 從原始碼知道程式提供了寫入binary並執行的操作。
2. 使用pwntools打開遠端程式。
3. 製造出可以打開某個檔案，讀取並寫入到記憶體的binary payload。
4. 在這段payload後面寫入檔案路徑的字串以供syscall當參數讀取。
5. 送出payload並接收遠端程式回傳的flag。

[ROPlab]

1. 查看程式資訊知道有開啟NX和stack的保護。
2. 可以先從libc-2.29.so得出的資訊有libc\_start\_main的offset

(0x26a80)以及回傳的行數(+235)，需要的gadget位置則有libc\_system的offset(0x52fd0)、pop\_rdi,ret 指令的offset(0x26542)、單獨ret的offset(0x2535f)和/bin/sh字串的address offset(0x1afb84)。

1. 需要從遠端得到的資訊有用來檢查的隨機數canary和libc的base，canary可以透過把輸入蓋到canary的位址讀取到，libc的base則是蓋到return回libc\_start\_main的位址讀取後減掉offset和行數得出。
2. 之後便可以構造出payload為寫到canary的padding+讀到的canary+隨便一個8byte的rbp值(0)+將本來的return address寫成libc\_base+pop\_rdi的address+要被pop進rdi的/bin/sh字串address+用來修正stack對齊問題的單獨libc\_base + ret的位址+最後就是需要呼叫的libc\_base+system的位址。
3. 執行後會先通過canary的檢查之後跳到pop rdi的地方將/bin/sh放進rdi，執行一個ret把rsp pop掉一個對齊後呼叫把rdi當作參數的system，後得到shell即可找到flag。

[GOTlab]

1. 查看原始碼後知道可以在任意位址寫入任意8byte的資料，並且沒有開啟PIE，因此執行時載入的address是固定的。
2. 先leak出printf@plt的連結address，減去libc\_printf的offset後得到libc\_base。
3. 之後將exit@plt的連結位址改寫成main，使得執行exit時會呼叫main，得到多次修改的機會。
4. 現在需要一個動態連結並且參數為動態位址的函數，在initproc中找到了setvbuf。
5. 接著將setvbuf連結到libc\_base+system，使的呼叫setvbuf時變成呼叫system。
6. 最後再將stdin的位址改成libc\_base+/bin/sh的offset即可當作參數執行system(/bin/sh)來get shell，之後便可找到flag。

[fmtlab]

1. 先列出需要修改的記憶體有return address需要指向win，還有end需要改成0來結束迴圈執行ret指令。
2. 可以先從libc-2.29.so得出的資訊有libc\_start\_main的offset

(0x26a80)以及回傳的行數(+235)，需要的gadget則有單獨ret的offset(0x2535f)。

1. 可以從程式binary得出的資訊有和win在同區塊的csu\_init offset(0x80012c0)，還有win的offset(0x8001201)，以及執行printf後rbp增加的offset(224)。
2. 需要從遠端得到的資訊有存放win的位址，由於可以從%14$p的記憶體位址leak出來csu\_init的位址，減去csu\_init的offset得到code\_base再加上win的offset即可得到win的位置，還可以從%12$p的位址leak出目前的rbp減去offset後可以得到main中rbp的位址，最後由於還需要呼叫ret對齊stack所以需要從%15$p中leak出libc\_start\_main的回傳址，減去行數和offset後可以得到libc\_base。
3. 之後開始使用printf的fmt中%hhn參數開始修改記憶體數值，先把本來的return address(main rbp+8)寫入libc\_base+ret的指令位址，用來對齊stack。
4. 再把ret後的return address(main rbp+0x10)寫入win的位址。
5. 最後再把end變數的位址(main rbp-0x34)寫入0結束回圈，執行2次ret後跳到win中執行system(/bin/sh)，get shell後找到flag。