Computer Security Homework 0x08

Real name: 涂世昱

Nick name: R07922115

Student ID: R07922115

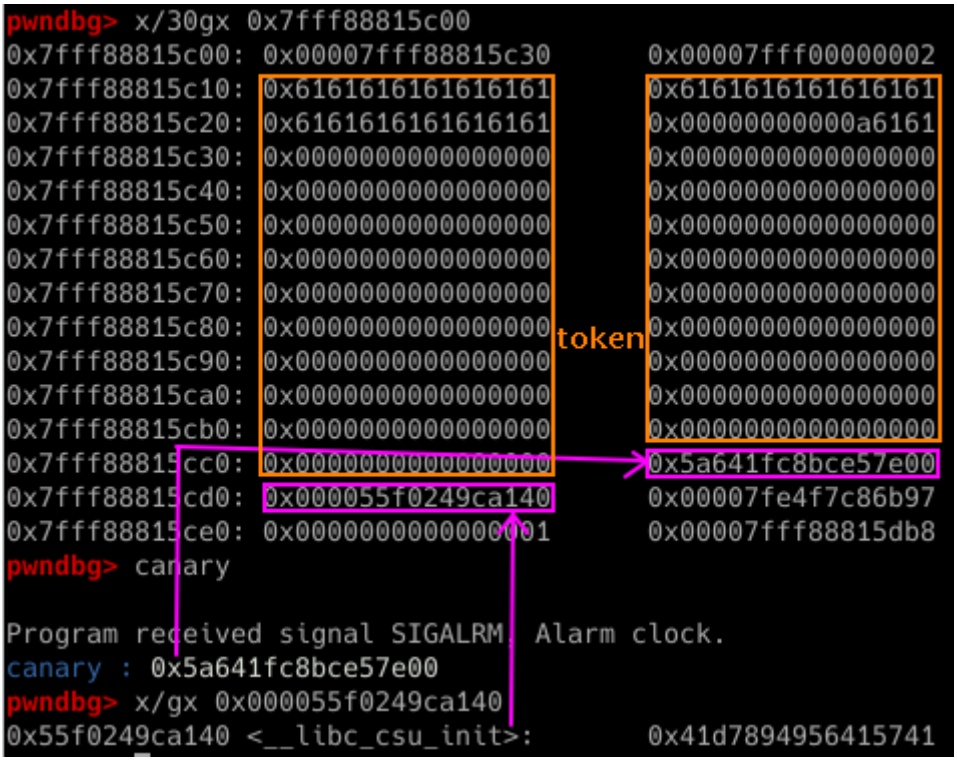
EDU 2019 election

Introduction

這題我的解法是在main function當中如果從stack當中觀察可發現在token之後緊接著的8個bytes是canary, 再接下去8個bytes為<__libc_csu_init>的位址, 將這兩個東西leak出就可以bypass canary以及pie的保護機制了, 而在後面voting function的部份因為我可以想辦法overflow msg在return address後面接ROP chain, 但是能接的東西有限, 所以要用到上課教的stack pivoting的方式將我的ROP chain蓋在其他地方, leak出libc再跳到main重新執行, 用同樣的流程蓋一條ROP chain, 執行system(sh)。

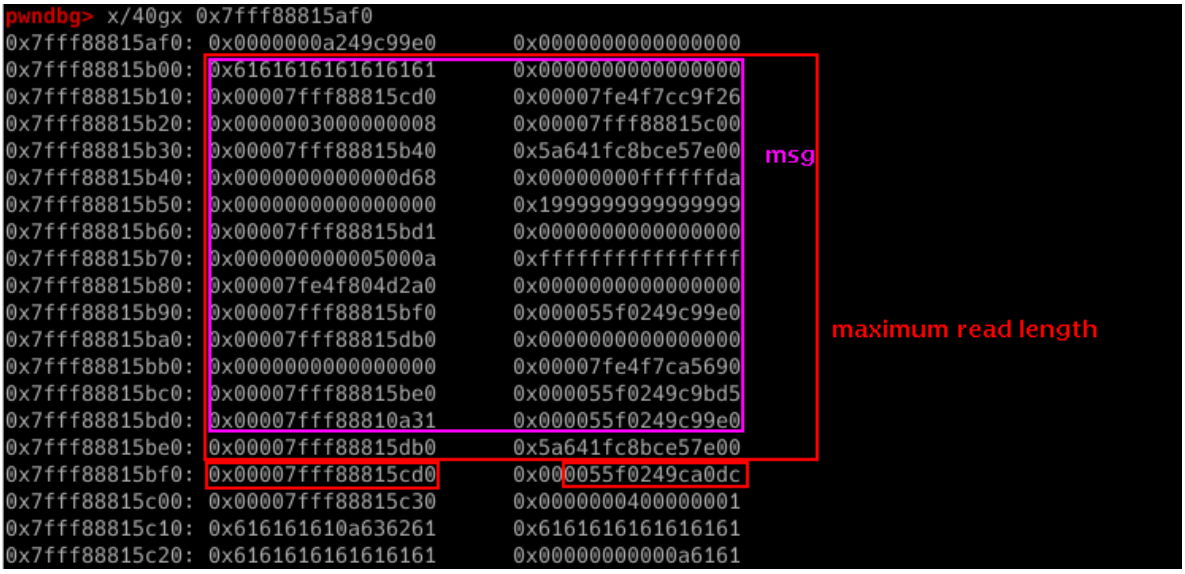
Walkthrough

bypass canary & pie



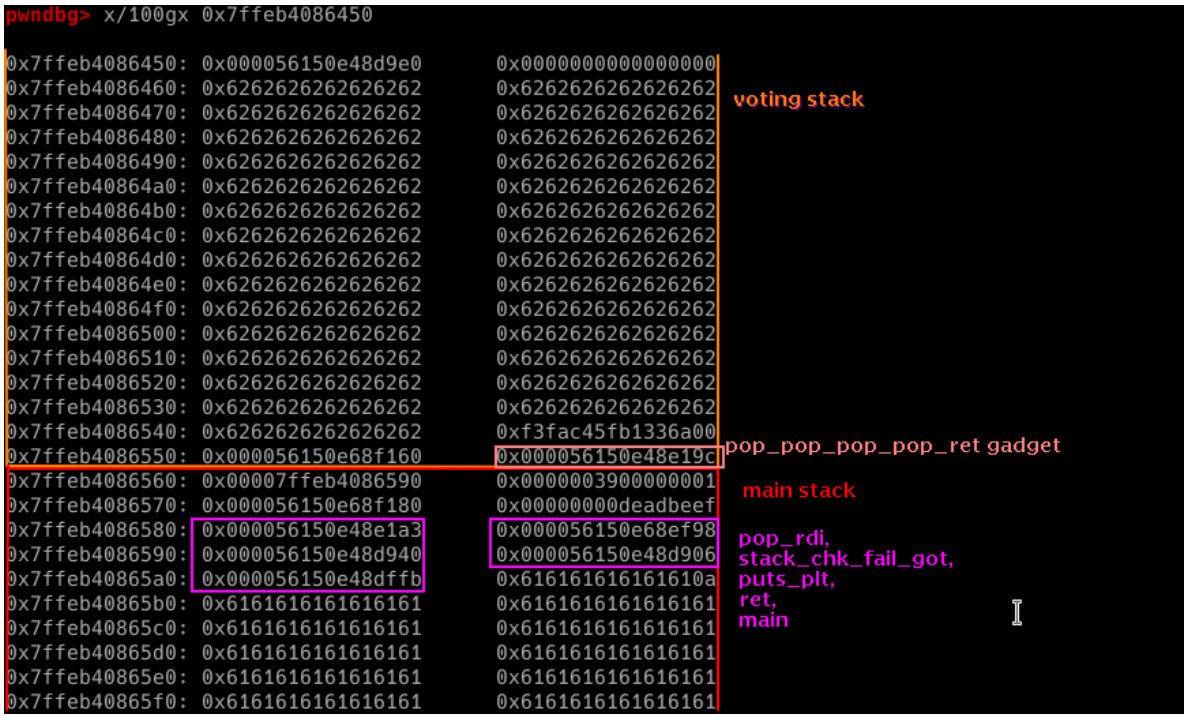
在main中可發現`memcmp( buf , token , len )`, 其中buf size為`0xc8`, token size為`0xb8`, buf比token整整多了0x10 bytes, 我就把token填滿0xb8個'a', 然後把buf也先填0xb8個'a', 接著就可以開始猜緊接在token之後的第一個byte, 如果猜中的話if條件就能滿足, 就會login, 如此一來就可以logout再繼續猜下一個byte, 直到把token後面0x10個bytes都猜完, 就把canary和pie base都leak出來了。

leak libc



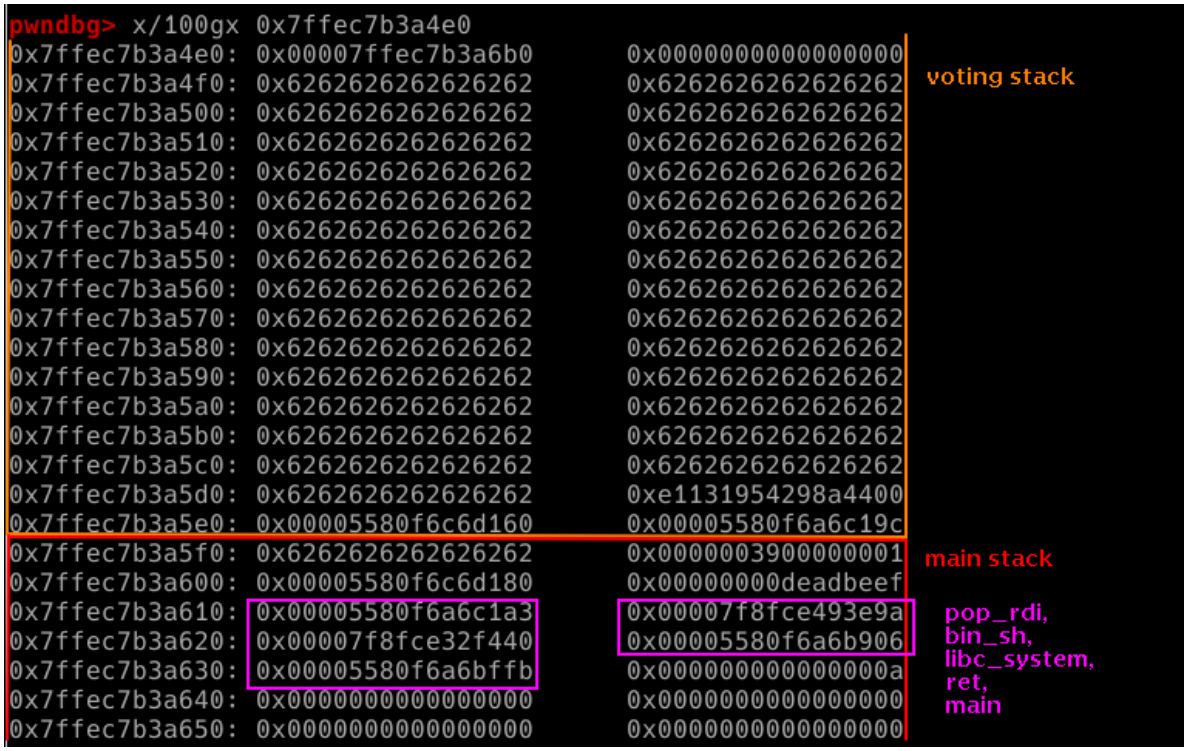
在voting function可以發現後面會`read( 0 , msg , candidates[idx].votes )`, 其中`msg` size 為`0xe0`, `candidates[idx].votes`在`struct Candidate`發現其中`votes`的type為`uint8_t`, 最大為`0xff`, 所以最多可以read `0xff`個字, 剛剛又已經把canary leak出來了, 所以就可以覆蓋掉return address。於是就可以一直register新的token, 然後把票頭給`candidates[0]`直到他的`votes`為`0xff`為止, 就可以成功overflow了。

因為`votes`最多只能為`0xff`, 恰好可以蓋一個return address, 後面就塞不下了, 所以必須得用stack pivoting的方式另起爐灶, 我這邊選擇的方式是把真正的ROP chain放在`main` stack當中的token裡面, 把`stack_chk_fail_got`位置當作`puts`參數印出來最後再跳回`main`。結論如下,



call system(sh)

跳回`main`之後再用相同的方式造ROP chain讓它去執行`system(sh)`, 如下圖


pwndbg> x/100gx 0x7ffec7b3a4e0
0x7ffec7b3a4e0: 0x00007ffec7b3a6b0 0x0000000000000000
0x7ffec7b3a4f0: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a500: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a510: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a520: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a530: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a540: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a550: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a560: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a570: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a580: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a590: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a5a0: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a5b0: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a5c0: 0x6262626262626262 0x6262626262626262
0x7ffec7b3a5d0: 0x6262626262626262 0xe1131954298a4400
0x7ffec7b3a5e0: 0x00005580f6c6d160 0x00005580f6a6c19c
0x7ffec7b3a5f0: 0x6262626262626262 0x0000003900000001
0x7ffec7b3a600: 0x00005580f6c6d180 0x00000000deadbeef
0x7ffec7b3a610: 0x00005580f6a6c1a3 0x00007f8fce493e9a
0x7ffec7b3a620: 0x00007f8fce32f440 0x00005580f6a6b906
0x7ffec7b3a630: 0x00005580f6a6bffb 0x000000000000000a
0x7ffec7b3a640: 0x0000000000000000 0x0000000000000000
0x7ffec7b3a650: 0x0000000000000000 0x0000000000000000
voting stack
main stack
pop_rdi,
bin_sh,
libc_system,
ret,
main