x64逆向:基本运算和分支结构

通用找main的方法:从start开始进去,第二个call进去找exit函数,一般第一个exit函数上一个函数就是main。高版本vs比较特殊,在exit上面在第二个函数(注意main有三个参数)

基本运算

- 1. 加减法
 - Tea 比例因子寻址代替 add、sub ,同x86一样

x >= 0

x / c = x * M >> 32 >> n

- 2. 乘法
 - o 在x86上优化成多条指令的情况中, x64会直接用 imul
- 3. 除法
 - 。 优化公式

```
在x86下,指数为32。在x64下,指数为64。
以下统一以32为例:
   1. 无符号2的幂
       x / 2 \wedge n = x >> n
   2. 无符号非2的幂1
       x / c = x * M >> 32 >> n
   3. 无符号非2的幂2
       x / c = (((x - (x * M >> 32)) >> n1) + (x * M >> 32)) >> n2
   4. 有符号正2的幂
       x >= 0
          x / 2 \wedge n = x >> n
       x < 0
           x / 2^n = (x + (2^n - 1)) >> n
   5. 有符号负2的幂
       同情况1, -(x / 2^n)
   6. 有符号正非2的幂1
          x / c = x * M >> 32 >> n
          此时, M > 0
       x < 0
          x / c = (x * M >> 32 >> n) + 1
   7. 有符号正非2的幂2
       x >= 0
           x / c = (x * M >> 32) + x >> n
       x < 0
          x / c = ((x * M >> 32) + x >> n) + 1
   8. 有符号负非2的幂1
```

```
此时, M < 0
    x < 0
    x / c = (x * M >> 32 >> n) + 1

9. 有符号负非2的幂2
    x >= 0
    x / c = (x * M >> 32) - x >> n
    x < 0
    x / c = ((x * M >> 32) - x >> n) + 1

计算除数c, 其中n为移位总次数:

1. c = 2^n / M
    c > 0 时, 使用

2. c = 2^n / (2^32 - M)
    c < 0 时, 使用

3. c = 2^n / (2^32 + M)
    乘减移加移时, 使用
```

。 总结

```
重要公式有三:

1. x * M >> 32 >> n

M > 0 时, x / c

M < 0 时, x / -c

2. (x * M >> 32) + x >> n

x / c

3. (x * M >> 32) - x >> n

x / -c
```

4. 取模

。 优化公式

5. 三目运算

- o x == 0 ? 8 : 9 这种两值相差一的, 利用 sete reg 条件设置指令
- o 其他情况利用 cmove 条件传送指令

分支结构

看图IDA和x64dbg的跳转线条,实线是 jmp ,虚线是条件跳,中间包含的圆点就是汇编语句,注意线的 开头有个判断

1. if

- o 单分支 if, 虚线
- o if-else, 虚线 if + 实线 else, 两部分有重叠
- o && 相当于 if 的嵌套
- 。 两个虚线交叉是 11 ,满足条件1跳,不满足条件2走

```
. .....
- - . text:00000001400010A8
                                     jz
lea
                                            short loc 1400010B8
   .text:00000001400010AA
                                            eax, [rsi-1]
                                     lea
   .text:00000001400010AD
                                           rcx, aN1Argc1 ; "n
 .text:00000001400010B4
                                     test eax, eax
_ .text:00000001400010B6
                                    jz short loc_1400010BF
   .text:00000001400010B8
.text:00000001400010B8 loc_1400010B8:
                                                          ; COE
                                     lea rcx, aN1Argc1_0; "n
   .text:00000001400010BF
.text:00000001400010BF loc_1400010BF:
                                                          ; COE
                                    call sub 140001010
```

2. 循环

- o do-while,线条向上
- o while 优化成 do-while
- o for优化为if + do-while
- o break 线条直接跳出
- o continue 线条上跳

```
; C
                                           movsxd rax, ebx
    .text:00000001400010C4
 .text:00000001400010C7
                                                   rdi, rax
                                           add
 .text:00000001400010CA
                                           cmp
                                                   rdi, 0Ah
.text:00000001400010CE
.text:00000001400010D0
.text:00000001400010D2
                                                   short loc 1400010D7
                                            ja
                                           inc
                                                   ebx
                                           cmp ebx, 0Ah
jl short loc_1400010C4
- text:00000001400010D5
    .text:00000001400010D7
.text:00000001400010D7 loc_1400010D7:
.text:00000001400010D7
                                                                    ; C
                                           mov rax, rdi
```

3. switch

。 线太多, 不看线看特征