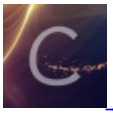
【reversing.kr逆向之旅】Easy ELF的writeup

原创

iqiqiya 于 2018-11-01 19:44:07 发布 618 收藏 1
分类专栏： 我的逆向之路 ------reversing.kr 我的CTF之路 文章标签： 【reversing.kr逆向之旅】Easy ELF的wri reversing.kr逆向Easy ELF的writeup Easy ELF writeup

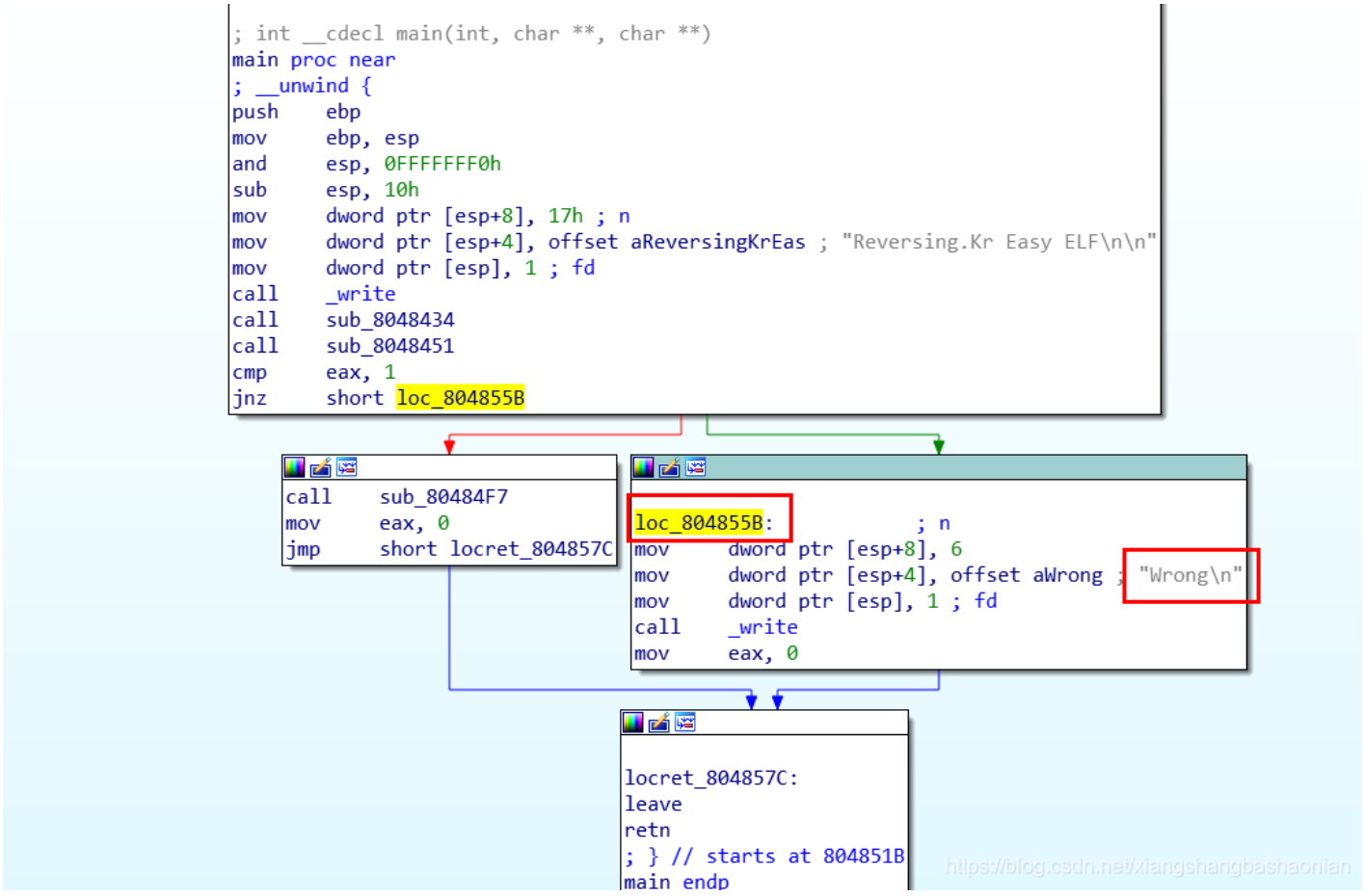

我的逆向之路 同时被 3 个专栏收录
108 篇文章 10 订阅
订阅专栏


------reversing.kr
11 篇文章 0 订阅
订阅专栏


我的CTF之路
92 篇文章 5 订阅
订阅专栏

这道题直接IDA Pro静态分析就可以

shift+f12就可以找到关键字符串　图形视图下也可以看清楚



反汇编main()

```
int __cdecl main()
{
  write(1, "Reversing.Kr Easy ELF\n\n", 0x17u);
  sub_8048434();                                // 接收我们的输入
  if ( sub_8048451() == 1 )
    sub_80484F7();                              // write(1, "Correct!\n", 9u);
  else
    write(1, "Wrong\n", 6u);
  return 0;
}
```

很显然sub_8048451()返回值等于1时　输出Correct

那我们接着F5反汇编这个函数

```
_BOOL4 sub_8048451()
{
  if ( byte_804A021 != 0x31 )                   // input[1] = 0x31
    return 0;
  input ^= 0x34u;                               // input[0] ^ 0x34
  byte_804A022 ^= 0x32u;                        // input[2] ^ 0x32
  byte_804A023 ^= 0x88u;                        // input[3] ^ 0x88
  if ( byte_804A024 != 0x58 )                   // input[4] = 0x58
    return 0;
  if ( byte_804A025 )                           // input[5] = 0x1
    return 0;
  if ( byte_804A022 != 0x7C )                   // input[2] = 0x7c
    return 0;
  if ( input == 0x78 )                          // input[0] = 0x78
    return byte_804A023 == 0xDDu;               // input[3] = 0xdd
  return 0;
}
```

这里我分析的是有6个变量 以数组的形式

双击任一变量　看下内存顺序

(ps:BSS段通常是指用来存放程序中未初始化的全局变量和静态变量的一块内存区域

特点是可读写的，在程序执行之前BSS段会自动清0)

```
.bss:0804A01C dword_804A01C   dd ?                    ; DATA XREF: sub_80483B0+10↑r
.bss:0804A01C                                         ; sub_80483B0+33↑w ...
.bss:0804A020 input           db ?                    ; DATA XREF: sub_8048434+B↑o
.bss:0804A020                                         ; sub_8048451:loc_8048469↑r ...
.bss:0804A021 byte_804A021    db ?                    ; DATA XREF: sub_8048451+3↑r
.bss:0804A022 byte_804A022    db ?                    ; DATA XREF: sub_8048451+27↑r
.bss:0804A022                                         ; sub_8048451+31↑w ...
.bss:0804A023 byte_804A023    db ?                    ; DATA XREF: sub_8048451+36↑r
.bss:0804A023                                         ; sub_8048451+40↑w ...
.bss:0804A024 byte_804A024    db ?                    ; DATA XREF: sub_8048451+45↑r
.bss:0804A025 byte_804A025    db ?                    ; DATA XREF: sub_8048451:loc_80484A8↑r
.bss:0804A026                 db    ? ;
```


由sub_8048451()对应关系

可以得到结果

```
a = 0x78 ^ 0x34
b = 0x31
c = 0x7c ^ 0x32
d = 0xdd ^ 0x88
e = 0x58
f = 0x1
input = [a,b,c,d,e,f]
for i in input:
 print chr(i)
#L1NUX
```

这里f也就是对应的input[5] 转成字符是乱码 直接去掉吧 可能是IDA分析的缘故

提交L1NUX 正确