

# 【reversing.kr逆向之旅】 Easy Keygen的writeup

原创

iqiqiya 于 2018-10-31 21:18:16 发布 425 收藏

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 Keygen的 Easy Keygen的 writeup](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/xiangshangbashaonian/article/details/83590603>

版权



[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



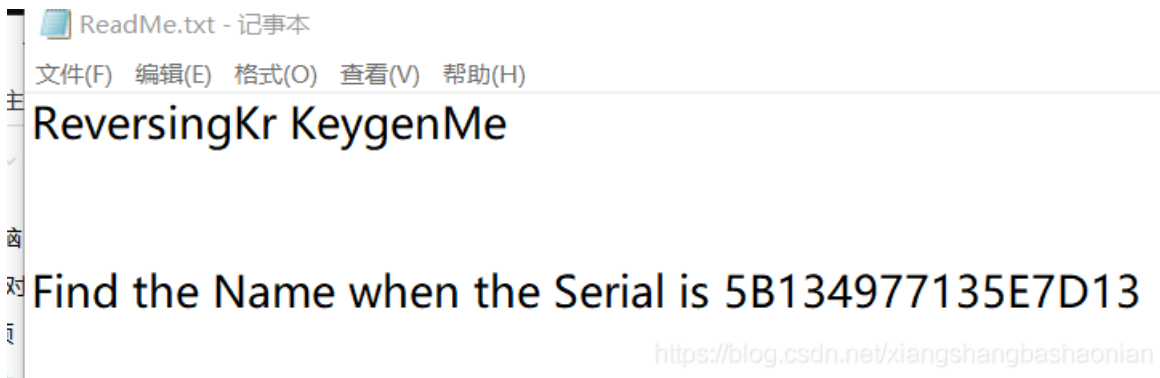
[我的CTF之路](#)

92 篇文章 5 订阅

订阅专栏

先看ReadMe.txt

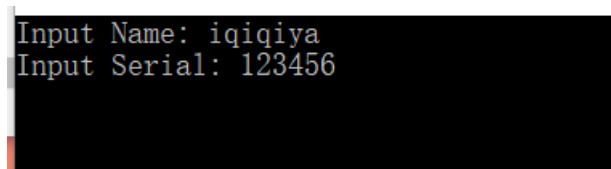
意思是让我们找到当序列号是5B134977135E7D13时对应的名字



查壳 无壳 vc++程序



运行发现当我们Name与Serial不对应时 程序直接退出



查看关键字符串

.rdata:0...	0000000D	C	KERNEL32.dll
.data:00...	00000007	C	Wrong\n
.data:00...	0000000A	C	Correct!\n
.data:00...	0000000F	C	Input Serial:
.data:00...	00000007	C	%s%02X
.data:00...	0000000D	C	Innut Name:

双击字符串 再双击引用 F5 然后进行分析

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    signed int j; // ebp
    signed int i; // esi
    char v6; // [esp+Ch] [ebp-130h]
    char v7; // [esp+Dh] [ebp-12Fh]
    char v8; // [esp+Eh] [ebp-12Eh]
    char input; // [esp+10h] [ebp-12Ch]
    char v10; // [esp+11h] [ebp-12Bh]
    __int16 v11; // [esp+71h] [ebp-CBh]
    char v12; // [esp+73h] [ebp-C9h]
    char v13; // [esp+74h] [ebp-C8h]
    char v14; // [esp+75h] [ebp-C7h]
    __int16 v15; // [esp+139h] [ebp-3h]
    char v16; // [esp+13Bh] [ebp-1h]

    input = 0;
    v13 = 0;
    memset(&v10, 0, 0x60u);
    v11 = 0;
    v12 = 0;
    memset(&v14, 0, 0xC4u);
    v15 = 0;
    v16 = 0;
    v6 = 16;
    v7 = 32;
    v8 = 48;
    printf(aInputName); // 输出Input Name:
    scanf(aS, &input); // 这个input用来保存Name
    j = 0; // 这个for循环 是对Name的操作
    for ( i = 0; j < strlen(&input); ++i )
    {
        if ( i >= 3 ) // 每次取三位 给sprintf进行操作 生成Serial放在v13所在内存区域
            i = 0;
        sprintf(&v13, aS02x, &v13, *(&input + j++) ^ *(&v6 + i)); // %s%02X %02x用来控制格式,以十六进制输出,2为指
    }
    memset(&input, 0, 0x64u);
    printf(aInputSerial); // Input Serial:
    scanf(aS, &input); // 这个input用来保存Serial
    if ( !strcmp(&input, &v13) ) // 真码与假码的比较
        printf(aCorrect);
    else
        printf(aWrong);
    return 0;
}
```

这个程序的思路就是 将我们输入的名称进行一系列操作 生成一个Serial

再与我们自己输入的Serial进行对比

如果一致 就输出 correct

刚开始这个函数参数好乱啊 大概可以猜到

```
sprintf(&v13, aS02x, &v13, *(&input + j++) ^ *(&v6 + i));
```

但还是决定IDA 动态调试看一下汇编

这里是我们的输入

```
debug007:0019FE18 db 69h ; i
debug007:0019FE19 db 71h ; q
debug007:0019FE1A db 69h ; i
debug007:0019FE1B db 71h ; q
debug007:0019FE1C db 69h ; i
debug007:0019FE1D db 79h ; y
debug007:0019FE1E db 61h ; a
```

```
debug007:0019FE14 db 10h
debug007:0019FE15 db 20h
debug007:0019FE16 db 30h
```

每次取Name中三个字符一次与0x10,0x20,0x30进行异或 再赋值给v13 生成下面的序列

```
debug007:0019FE7C db 37h ; 7
debug007:0019FE7D db 39h ; 9
debug007:0019FE7E db 35h ; 5
debug007:0019FE7F db 31h ; 1
debug007:0019FE80 db 35h ; 5
debug007:0019FE81 db 39h ; 9
debug007:0019FE82 db 36h ; 6
debug007:0019FE83 db 31h ; 1
debug007:0019FE84 db 34h ; 4
debug007:0019FE85 db 39h ; 9
```

图形视图更清晰一点

```
loc_40107E:
movsx ecx, [esp+esi+13Ch+v6]
movsx edx, [esp+ebp+13Ch+input] ; 每次取出Name的一个字符 放进edx中
xor ecx, edx ; 异或
lea eax, [esp+13Ch+v13]
push ecx
push eax
lea ecx, [esp+144h+v13]
push offset aS02x ; "%s%02X"
push ecx ; char *
call _sprintf
add esp, 10h
inc ebp ; j++
lea edi, [esp+13Ch+input]
or ecx, 0FFFFFFFh
xor eax, eax
inc esi ; i++
repne scasb ; repne scasb指令, 用于扫描字符串, 计算字符串的长度
not ecx
dec ecx
cmp ebp, ecx ; 比较j与len(Name)
jl short loc_401077 ; 比较i与3的大小

loc_4010B6:
mov ecx, 19h
xor eax, eax
lea edi, [esp+13Ch+input]
```

<https://blog.csdn.net/xiangshangbashaonian>

好啦 我们一直知道了怎样通过Name生成Serial

那我们只要逆一下就行啦

脚本来自[夜影师傅](#)

```
Serial = "5B134977135E7D13"
Name = ''
a = [16, 32, 48]
for i in range(len(Serial)//2):
    Name += chr(int(Serial[2*i:2*i+2], 16) ^ a[i%3])
print(Name)
#K3yg3nm3
```

参考链接: <https://blog.csdn.net/whklhxxx/article/details/78079929>