jarvis oj reverse 病毒数据分析 writeup

原创

charlie_heng 于 2018-02-19 09:33:38 发布 307 收藏

分类专栏： 二进制-逆向工程

本文链接：https://blog.csdn.net/charlie_heng/article/details/79336476

这题之前做过，但是做到一半就放弃了，现在又回来做一下

首先这题其实还是有点难度的

我们先来分析下main函数部分，首先它做了反debug判断，获取父进程信息
其次它还判断了程序所在位置，过了这几个判断之后，获取当前时间作为seed给srand

然后在注册表里面拿到文档所在的目录

接着在目录里面搜索docx后缀的文件，将其加密之后发送出去

在加密那里，它做了几件事
1. 生成8字节随机数
2. new了一个缓冲区，首先往里面放了文件的大小，然后在里面填了bufFileContent这个字符串，之后往里面塞文件内容，然后在最后加上bufFileName，然后在加上文件名
3. 之后在加密函数里面，先拿缓冲区前8个字节，用tea加密，加密的key可以从数据包中拿到，加密完之后，xor一下生成的8字节随机数
4. 之后再拿缓冲区8个字节，异或前一个8字节tea加密之后的数据，然后用tea加密这个数据，然后再异或前8个没加密字节，之后一直循环这样 （感觉我也说不清楚....还是自己调试或者看下我下面放出来的代码吧

有了加密的过程，逆推就得到未加密的数据
这里一个关键是，如何得到srand的seed？

因为这个时间可以在数据包中获取，但是这个是不准的，我试了一下，发现准确的seed是 1465461209

下面就是解密的代码

```
import struct
import binascii

def rev(x):
    return struct.unpack('>L',struct.pack('<L',x))[0]

def u32(x):
    return struct.unpack('<L',x)[0]

def p32(x):
    return struct.pack('<L',x)

def d32(x):
```

```
    x=x.strip()
    x=x.replace(" ",'')
    return binascii.a2b_hex(x)


def enc(y,z,key,round=16):
    sum=0
    delta=0x9e3779b9
    for i in range(round):
        sum+=delta
        sum&=0xffffffff
        y += ((z << 4) + key[0]) ^ (z + sum) ^ ((z >> 5) + key[1])
        y&=0xffffffff
        z += ((y << 4) + key[2]) ^ (y + sum) ^ ((y >> 5) + key[3])
        z&=0xffffffff
        print(hex(y),hex(z))
    return y,z


def dec(y,z,key,round=16):
    delta=0x9e3779b9
    sum=delta<<4
    for i in range(round):
        z -= ((y << 4) + key[2]) ^ (y + sum) ^ ((y >> 5) + key[3])
        z+=0xffffffff+1
        z&=0xffffffff
        y -= ((z << 4) + key[0]) ^ (z + sum) ^ ((z >> 5) + key[1])
        sum -= delta
        sum+=0xffffffff+1
        y+=0xffffffff+1
        y&=0xffffffff
        sum&=0xffffffff
    return y,z

#key=[0x84be2329 ,0xaed56ce1 ,0x7c80ef87 ,0xd212ffb0]
#time=1465461209

key=[0x2923be84 ,0xe16cd6ae ,0x87ef807c ,0xb0ff12d2]
rand1=u32(b'\xc3\xe6H\xf6')
rand2=u32(b'\xe0\xf7"~')


f2=open('./t1','rb')
enc=f2.read()
f2.close()


t1=u32(enc[:4])
t2=u32(enc[4:8])
t1_d=t1^rand1
t2_d=t2^rand2
t1_dec,t2_dec=dec(rev(t1_d),rev(t2_d),key)


ans=[]
ans.append(t1_dec)
ans.append(t2_dec)

for i in range(8,len(enc),8):
```

```
        tt1=rev(u32(enc[i:i+4]))
        tt2=rev(u32(enc[i+4:i+8]))
        tt1_d=tt1^t1_dec
        tt2_d=tt2^t2_dec
        tt1_dd,tt2_dd=dec(tt1_d,tt2_d,key)
        tt1_dec=tt1_dd^rev(t1_d)
        tt2_dec=tt2_dd^rev(t2_d)

        t1_dec=tt1_dec
        t2_dec=tt2_dec
        t1_d=rev(tt1_d)
        t2_d=rev(tt2_d)
        ans.append(t1_dec)
        ans.append(t2_dec)

    f1=open('./ans','wb')
    for i in ans:
        f1.write(p32(rev(i)))
    f1.close()
```