buu刷题 红帽杯2019-XX wp

原创

43v3rY0unG 于 2020-08-11 21:21:33 发布 249 收藏
分类专栏： # RE

本文链接：https://blog.csdn.net/weixin_43876357/article/details/107943529

首先找到关键函数，

找到我们输入的数据经过一系列操作之后得到的，即最终结果（注意小端序存储）：

```
156   *(_QWORD *)&v29 = 0xC0953A7C6B40BCCEi64;
157   v24 = v19 - (_BYTE *)&v29;
158   *((_QWORD *)&v29 + 1) = 0x3502F79120209BEFi64;
159   v25 = 0i64;
160   v30 = 0xC8021823;
161   v31 = 0xFA5656E7;
```

把数据dump出来：

```
int a[] = {0xCE, 0xBC, 0x40, 0x6B, 0x7C, 0x3A, 0x95, 0xC0, 0xEF, 0x9B, 0x20, 0x20, 0x91, 0xF7, 0x02, 0x35,
```

然后就来看一下整个程序对我们输入的数据进行了哪些操作：
首先确定字符串长度：

```
while ( *((_BYTE *)Code + v3) );
if ( v3 != 19 )                                 // 字符串长度为19
{
  sub_140001620(std::cout, "error\n");
  _exit((unsigned __int64)Code);
}
```

从后往前推：
下面脚本是逆向解出xxtea加密之后的结果，（三个一组进行异或）

```
#include<stdio.h>
#include<Windows.h>
int main()
{
 int count = 0;
 int b[24];
 int a[] = { 0xCE, 0xBC, 0x40, 0x6B, 0x7C, 0x3A, 0x95, 0xC0, 0xEF, 0x9B, 0x20, 0x20, 0x91, 0xF7, 0x02, 0x35


 for (int i = 23; i >=3; i--)
 {

  for (int j = 6-count; j >= 0; j--)
  {
    a[i]^=a[j];
  }
  if (i % 3 == 0)
  {
   count++;
  }

 }
 for (int i = 0; i < 24; i++)
  printf("0x%x,", a[i]);
 printf("\n");
 b[2] = a[0];
 b[0] = a[1];
 b[3] = a[2];
 b[1] = a[3];
 b[6] = a[4];
 b[4] = a[5];
 b[7] = a[6];
 b[5] = a[7];
 b[10] = a[8];
 b[8] = a[9];
 b[11] = a[10];
 b[9] = a[11];
 b[14] = a[12];
 b[12] = a[13];
 b[15] = a[14];
 b[13] = a[15];
 b[18] = a[16];
 b[16] = a[17];
 b[19] = a[18];
 b[17] = a[19];
 b[22] = a[20];
 b[20] = a[21];
 b[23] = a[22];
 b[21] = a[23];
 for (int i = 0; i < 24; i++)
  printf("0x%x,", b[i]);
 system("pause");
}
```

运行后可以得到数据

0xbc,0xa5,0xce,0x40,0xf4,0xb2,0xb2,0xe7,0xa9,0x12,0x9d,0x12,0xae,0x10,0xc8,0x5b,0x3d,0xd7,0x6,0x1d,0xdc

然后就是进行xxtea解密，密钥是输入字符的前四位，猜测是“flag”

```
#include <stdio.h>
#include <stdint.h>
#include<windows.h>
#define DELTA 0x9e3779b9
#define MX (((z>>5^y<<2) + (y>>3^z<<4)) ^ ((sum^y) + (key[(p&3)^e] ^ z)))

void btea(uint32_t *v, int n, uint32_t const key[4])
{
 uint32_t y, z, sum;
 unsigned p, rounds, e;
 if (n > 1)            /* Coding Part */
 {
  rounds = 6 + 52 / n;
  sum = 0;
  z = v[n - 1];
  do
  {
   sum += DELTA;
   e = (sum >> 2) & 3;
   for (p = 0; p<n - 1; p++)
   {
    y = v[p + 1];
    z = v[p] += MX;
   }
   y = v[0];
   z = v[n - 1] += MX;
  } while (--rounds);
 }
 else if (n < -1)      /* Decoding Part */
 {
  n = -n;
  rounds = 6 + 52 / n;
  sum = rounds*DELTA;
  y = v[0];
  do
  {
   e = (sum >> 2) & 3;
   for (p = n - 1; p>0; p--)
   {
    z = v[p - 1];
    y = v[p] -= MX;
   }
   z = v[n - 1];
   y = v[0] -= MX;
   sum -= DELTA;
  } while (--rounds);
 }
}


int main()
{
 //03e0164dd30553aa
        // uint32_t a[2] = { (unsigned int)0xd2cfdad7, 0x9ac8d5d4 };
 uint32_t v[6] = { (unsigned int)0x40cea5bc, (unsigned int)0xe7b2b2f4,(unsigned int)0x129d12a9,(unsigned in
   uint32_t const k[4] = { (unsigned int)0x67616c66, (unsigned int)0x0, (unsigned int)0X0, (unsigned int)0x
   int n = 6; //n的绝对值表示v的长度，取正表示加密，取负表示解密
```

```
   // v为要加密的数据是两个32位无符号整数
   // k为加密解密密钥，为4个32位无符号整数，即密钥长度为128位
   //printf("加密前原始数据：%x %x\n", v[0], v[1]);
   btea(v, -n, k);
   printf("加密后的数据：%x %x %x %x %x\n", v[0], v[1],v[2],v[3],v[4],v[5]);
   //btea(v, -n, k);
   //printf("解密后的数据：%x %x\n", v[0], v[1]);

   system("pause");

}
```

然后得到以下数据：67616c66 5858437b 646e615f 742b2b5f 7d6165
按照小端序，再把16进制转成字符就get flag了
以上是用c写的脚本。

放完整python脚本：

```
enc = 'CEBC406B7C3A95C0EF9B202091F70235231802C8E75656FA'.decode('hex')

dec1 = ''
for i in range(len(enc)/3):
    j = len(enc)/3 - i - 1
    res = ''
    for k in enc[j*3:j*3+3]:
        tmp = ord(k)
        for l in range(j):
            tmp ^= ord(enc[l])
        res += chr(tmp)
    dec1 = res + dec1
print dec1.encode('hex')

dec2 = [''] * len(dec1)
for i in range(len(dec1)):
    dec2[i] = dec1[i]

box = [1,3,0,2,5,7,4,6,9,11,8,10,13,15,12,14,17,19,16,18,21,23,20,22]
print len(box)

for i in range(len(box)):
    dec2[i] = dec1[box[i]]
dec3 = ''.join(dec2)
print dec3.encode('hex')

import struct

_DELTA = 0x9E3779B9

def _long2str(v, w):
    n = (len(v) - 1) << 2
    if w:
        m = v[-1]
        if (m < n - 3) or (m > n): return ''
        n = m
    s = struct.pack('<%iL' % len(v), *v)
    return s[0:n] if w else s
```

```
def _str2long(s, w):
    n = len(s)
    m = (4 - (n & 3) & 3) + n
    s = s.ljust(m, "\0")
    v = list(struct.unpack('<%iL' % (m >> 2), s))
    if w: v.append(n)
    return v

def encrypt(str, key):
    if str == '': return str
    v = _str2long(str, True)
    k = _str2long(key.ljust(16, "\0"), False)
    n = len(v) - 1
    z = v[n]
    y = v[0]
    sum = 0
    q = 6 + 52 // (n + 1)
    while q > 0:
        sum = (sum + _DELTA) & 0xffffffff
        e = sum >> 2 & 3
        for p in xrange(n):
            y = v[p + 1]
            v[p] = (v[p] + ((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4) ^ (sum ^ y) + (k[p & 3 ^ e] ^ z))) & 0xff
            z = v[p]
        y = v[0]
        v[n] = (v[n] + ((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4) ^ (sum ^ y) + (k[n & 3 ^ e] ^ z))) & 0xffffff
        z = v[n]
        q -= 1
    return _long2str(v, False)

def decrypt(str, key):
    if str == '': return str
    v = _str2long(str, False)
    k = _str2long(key.ljust(16, "\0"), False)
    n = len(v) - 1
    z = v[n]
    y = v[0]
    q = 6 + 52 // (n + 1)
    sum = (q * _DELTA) & 0xffffffff
    while (sum != 0):
        e = sum >> 2 & 3
        for p in xrange(n, 0, -1):
            z = v[p - 1]
            v[p] = (v[p] - ((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4) ^ (sum ^ y) + (k[p & 3 ^ e] ^ z))) & 0xff
            y = v[p]
        z = v[n]
        v[0] = (v[0] - ((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4) ^ (sum ^ y) + (k[0 & 3 ^ e] ^ z))) & 0xffffff
        y = v[0]
        sum = (sum - _DELTA) & 0xffffffff
    return _long2str(v, True)

key = 'flag'
dec4 = decrypt(dec3, key)
print len(dec4)
print dec4
```

flag：**flag{CXX_and_++tea}**

最后总结一下这个程序的整个流程：

1. 判断输入的字符串的每个字符是否包含在"qwertyuiopasdfghjklzxcvbnm1234567890"中
2. 取输入字符串的前4位字符，即"flag"，扩展为16位，作为xxtea加密的秘钥key
3. 将输入的字符串使用key加密，加密后的字符保存在字符数组v18，共24位字符
4. 打乱v18数组，保存到v19数组中
5. 将24位字符，每3位为一组，每一组异或值（具体看代码），得到新的加密字符串
6. 将新的加密字符串与已经存在的字符串比较，相同即获得胜利

难点在于判断是xxtea加密：
这里有另一种方法，装一个插件，*FindCrypt*，安装教程自行谷歌。注意，此插件只能简单识别是TEA加密。

参考博客：
https://www.cnblogs.com/playmak3r/p/12063593.html
https://impakho.com/post/redhat-2019-online-writeup
https://www.cnblogs.com/Mayfly-nymph/p/11869959.html
利用TEA算法进行数据加密