[BUUCTF]REVERSE——[ACTF新生赛2020]usualCrypt

原创

Angel~Yan 于 2020-11-30 14:28:54 发布 294 收藏 1

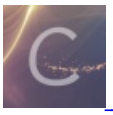
分类专栏： REVERSE BUUCTF刷题记录

本文链接：https://blog.csdn.net/mcmuyanga/article/details/110382800


REVERSE 同时被 2 个专栏收录

75 篇文章 1 订阅

订阅专栏


BUUCTF刷题记录

198 篇文章 14 订阅

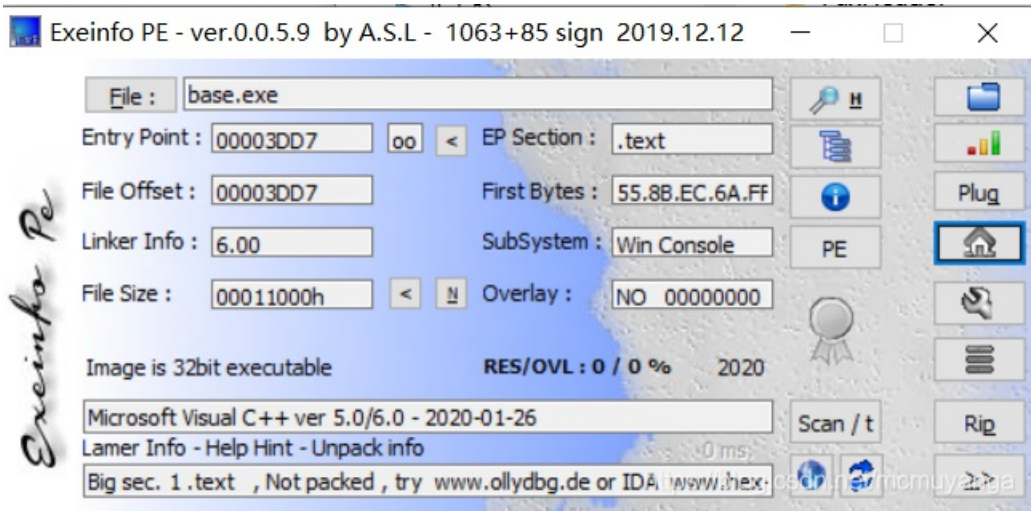
订阅专栏

[ACTF新生赛2020]usualCrypt

附件

步骤：

例行检查，无壳，32位程序



32位ida载入，直接看main函数

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // esi
  int result; // eax
  int v5; // [esp+8h] [ebp-74h]
  int v6; // [esp+Ch] [ebp-70h]
  int v7; // [esp+10h] [ebp-6Ch]
  __int16 v8; // [esp+14h] [ebp-68h]
```

```
  char v9; // [esp+16h] [ebp-66h]
  char v10; // [esp+18h] [ebp-64h]

  sub_403CF8((int)&unk_40E140);
  scanf(aS, &v10);
  v5 = 0;
  v6 = 0;
  v7 = 0;
  v8 = 0;
  v9 = 0;
  sub_401080((int)&v10, strlen(&v10), (int)&v5);
  v3 = 0;
  while ( *((_BYTE *)&v5 + v3) == byte_40E0E4[v3] )
  {
    if ( ++v3 > strlen((const char *)&v5) )
      goto LABEL_6;
  }
  sub_403CF8((int)aError);
LABEL_6:
  if ( v3 - 1 == strlen(byte_40E0E4) )
    result = sub_403CF8((int)aAreYouHappyYes);
  else
    result = sub_403CF8((int)aAreYouHappyNo);
  return result;
}
```


逻辑很简单，一开始让我们输入一个字符串，然后该字符串经过sub_401080（）函数加密，加密后得到byte_40E0E4里面的数据 `zMXHz3TIgnxLxJhFAdtZn2fFk3lYCrtPC2l9`

看一下sub_401080函数

```
int __cdecl sub_401080(int a1, int a2, int a3)
{
  int v3; // edi
  int v4; // esi
  int v5; // edx
  int v6; // eax
  int v7; // ecx
  int v8; // esi
  int v9; // esi
  int v10; // esi
  int v11; // esi
  _BYTE *v12; // ecx
  int v13; // esi
  int v15; // [esp+18h] [ebp+8h]

  v3 = 0;
  v4 = 0;
  sub_401000();
  v5 = a2 % 3;
  v6 = a1;
  v7 = a2 - a2 % 3;
  v15 = a2 % 3;
  if ( v7 > 0 )
  {
    do
    {
      LOBYTE(v5) = *(_BYTE *)(a1 + v3);
      v3 += 3;
      v8 = v4 + 1;
      *(_BYTE *)(v8++ + a3 - 1) = byte_40E0A0[(v5 >> 2) & 0x3F];
      *(_BYTE *)(v8++ + a3 - 1) = byte_40E0A0[16 * (*(_BYTE *)(a1 + v3 - 3) & 3)
                                           + (((signed int)*(unsigned __int8 *)(a1 + v3 - 2) >> 4) & 0xF)];
      *(_BYTE *)(v8 + a3 - 1) = byte_40E0A0[4 * (*(_BYTE *)(a1 + v3 - 2) & 0xF)
                                          + (((signed int)*(unsigned __int8 *)(a1 + v3 - 1) >> 6) & 3)];
      v5 = *(_BYTE *)(a1 + v3 - 1) & 0x3F;
      v4 = v8 + 1;
      *(_BYTE *)(v4 + a3 - 1) = byte_40E0A0[v5];
    }
    while ( v3 < v7 );
    v5 = v15;
  }
  if ( v5 == 1 )
```

```
43  {
44    LOBYTE(v7) = *(_BYTE *)(v3 + a1);
45    v9 = v4 + 1;
46    *(_BYTE *)(v9 + a3 - 1) = byte_40E0A0[(v7 >> 2) & 0x3F];
47    v10 = v9 + 1;
```



```
48    *(_BYTE *)(v10 + a3 - 1) = byte_40E0A0[16 * (*(_BYTE *)(v3 + a1) & 3)];
49    *(_BYTE *)(v10 + a3) = 61;
50LABEL_8:
51    v13 = v10 + 1;
52    *(_BYTE *)(v13 + a3) = 61;
53    v4 = v13 + 1;
54    goto LABEL_9;
55  }
56  if ( v5 == 2 )
57  {
58    v11 = v4 + 1;
59    *(_BYTE *)(v11 + a3 - 1) = byte_40E0A0[((signed int)*(unsigned __int8 *)(v3 + a1) >> 2) & 0x3F];
60    v12 = (_BYTE *)(v3 + a1 + 1);
61    LOBYTE(v6) = *v12;
62    v10 = v11 + 1;
63    *(_BYTE *)(v10 + a3 - 1) = byte_40E0A0[16 * (*(_BYTE *)(v3 + a1) & 3) + ((v6 >> 4) & 0xF)];
64    *(_BYTE *)(v10 + a3) = byte_40E0A0[4 * (*v12 & 0xF)];
65    goto LABEL_8;
66  }
67LABEL_9:
68  *(_BYTE *)(v4 + a3) = 0;
69  return sub_401030(a3);
70}
```


头部有一个sub_401000函数，中间看运算特征码可以判断是base64加密，尾部一个sub_401030函数

先从sub_401000函数函数开始看起

```
1signed int sub_401000()
2{
3  signed int result; // eax
4  char v1; // cl
5
6  result = 6;
7  do
8  {
9    v1 = byte_40E0AA[result];
10   byte_40E0AA[result] = byte_40E0A0[result];
11   byte_40E0A0[result++] = v1;
12 }
13 while ( result < 15 );
14 return result;
15}
```


将两个数组里的数据进行了交换，看地址两个数组是连在一起的，其实也可以连在一起当成一个数组看，从下标为6开始到下标为15，往后偏移了10（0xA）位，也就是`QRSTUVWXY`和`GHIJKLMNOP`相互交换了一下

```
.data:0040E0A0 ; char byte_40E0A0[10]
.data:0040E0A0 byte_40E0A0     db 'A'                  ; DATA XREF: sub_401000:loc_401005↑r
.data:0040E0A0                                         ; sub_401000+17↑w ...
.data:0040E0A1                 db  42h ; B
.data:0040E0A2                 db  43h ; C
.data:0040E0A3                 db  44h ; D
.data:0040E0A4                 db  45h ; E
.data:0040E0A5                 db  46h ; F
.data:0040E0A6                 db  47h ; G
.data:0040E0A7                 db  48h ; H
.data:0040E0A8                 db  49h ; I
.data:0040E0A9                 db  4Ah ; J
.data:0040E0AA ; char byte_40E0AA[]
```

```
.data:0040E0AA byte_40E0AA      db 'K'                  ; DATA XREF: sub_401000+B↑r
.data:0040E0AA                                          ; sub_401000+11↑w
.data:0040E0AB aLmnopqrstuvwxy db 'LMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/',0
.data:0040E0E1                 align 4
```


所以原始用来加密的base64密码表是`ABCDEFQRSTUVWXYGHIJKLMNOPZabcdefghijklmnopqrstuvwxyz0123456789+/`

再看一下sub_401030函数，对字符串进行了大小写的转换

```
1  int __cdecl sub_401030(const char *a1)
2  {
3    __int64 v1; // rax
4    char v2; // al
5
6    v1 = 0i64;
7    if ( strlen(a1) != 0 )
8    {
9      do
10     {
11       v2 = a1[HIDWORD(v1)];
12       if ( v2 < 'a' || v2 > 'z' )
13       {
14         if ( v2 < 'A' || v2 > 'Z' )
15           goto LABEL_9;
16         LOBYTE(v1) = v2 + 32;
17       }
18       else
19       {
20         LOBYTE(v1) = v2 - 32;
21       }
22       a1[HIDWORD(v1)] = v1;
23 LABEL_9:
24       LODWORD(v1) = 0;
25       ++HIDWORD(v1);
26     }
27     while ( HIDWORD(v1) < strlen(a1) );
28   }
29   return v1;
30 }
```


程序理清楚了，我们可以反向推导，

第一步首先要对进行byte_40E0E4数组进行大小写转换，也就是我们输入的数据进行了base64加密后的状态

第二步是还原经 base64（更改密钥表后）加密字符的原含义，还原规则即sub_401000()的交换

第三步最后得到了真实的nbase64n加密字符串，解密即可得到我们输入的字符串，一般都是flag

```python
import base64

flag = ''
dict = {}
offset = 10

string = 'zMXHz3TIgnxLxJhFAdtZn2fFk3lYCrtPC2l9'.swapcase() #sub_401030()
print ('转换后的字符串：'+string)

myb = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'


for i in range(len(myb)):
    dict[myb[i]] = myb[i]



for i in range(6, 15): #sub_401000()
    dict[myb[i]] , dict[myb[i+offset]] = dict[myb[i+offset]] , dict[myb[i]] # 恢复base64密钥表



print ('**********')
for i in dict:
    print (i,dict[i])
print ('**********')

for i in range(len(string)):
    flag += dict[myb[i]]


flag = base64.b64decode(flag)

print(flag)
```

```
/ /
**********
b'flag{bAse64_h2s_a_Surprise}'
```