CTF-RE-JustRE （2019第三届强网杯）

原创

SuperGate 于 2019-06-06 12:38:05 发布 1053 收藏
分类专栏： CTF-RE

本文链接：https://blog.csdn.net/getsum/article/details/90580743

首先IDA点开进行静态分析：

```
int entry()
{
  char v2; // [esp+4h] [ebp-68h]
  int savedregs; // [esp+6Ch] [ebp+0h]

  puts("      #                          ######          ");
  puts("      # #    #  ####  #####     #     # ###### ");
  puts("      # #    # #        #       #     # #      ");
  puts("      # #    #  ####    #       ######  #####  ");
  puts("#     # #    #      #   #       #   #   #      ");
  puts("#     # #    # #    #   #       #    #  #      ");
  puts(" #####   ####   ####    #       #     # ###### ");
  read_string("%s", &v2);
  if ( sub_401610((int)&v2, (int)&savedregs) )
    sub_4018A0();
  puts("sorry..");
  return 0;
}
```

发现判断函数sub_401610，点进去看之后发现很长一串……前面比较好说，是将前8位字符变hex放入eax寄存器中 ， 第9，10位变hex放入ch(还是dh？)中

然后再往下面拉，可以看到很多SSE指令集，但是不是很复杂，网上可以搜到，发现都是一些比较基础数位操作。

```
LABEL_35:
  v18 = _mm_cvtsi32_si128((char)middle_2);
  v19 = _mm_unpacklo_epi8(v18, v18);
  v27 = _mm_shuffle_epi32(_mm_unpacklo_epi16(v19, v19), 0);// 对第9，10位进行128位填充
```

这个地方的v27在后面会频繁用到，值为第9，10位变hex后的128位填充

```
if ( v17 )
{
  v20 = 0;
  if ( Mark >= 2 )
  {
    v20 = 16;
    v21 = _mm_mullo_epi32(_mm_cvtepu8_epi32(_mm_cvtsi32_si128(v27.m128i_u32[0])), (__m128i)xmmword_404380);
    xmmword_405018 = (__int128)_mm_xor_si128(
                               _mm_add_epi32((__m128i)xmmword_404340, v9),
                               _mm_add_epi32(v21, (__m128i)xmmword_405018));
    xmmword_405028 = (__int128)_mm_xor_si128(
                               _mm_add_epi32(_mm_add_epi32((__m128i)xmmword_404350, (__m128i)xmmword_404340), v9),
                               _mm_add_epi32(v21, (__m128i)xmmword_405028));
    xmmword_405038 = (__int128)_mm_xor_si128(
                               _mm_add_epi32(_mm_add_epi32((__m128i)xmmword_404360, (__m128i)xmmword_404340), v9),
                               _mm_add_epi32(v21, (__m128i)xmmword_405038));
    xmmword_405048 = (__int128)_mm_xor_si128(
                               _mm_add_epi32(_mm_add_epi32((__m128i)xmmword_404370, (__m128i)xmmword_404340), v9),
                               _mm_add_epi32(v21, (__m128i)xmmword_405048));
  }
  do
  {
    *((_DWORD *)&xmmword_405018 + v20) = (v20 + first_8) ^ (0x1010101 * middle_2 + *((_DWORD *)&xmmword_405018 + v20));
    ++v20;
  }
```

这里是对xmmword_405018开始的96位的一个加密，可以看到前面405018~405048都是比较有规律的加密，其中xmmword_404340~404370是给出的常数，并且在后面的加密中不会改变。

要注意的是这里的add函数是分段加，即32位加一次，也就是把原先的128位整数分成4段依次做加法，每一段的溢出都直接舍弃。

```
while ( *((_BYTE *)&xmmword_405018 + v22) == *((_BYTE *)&loc_404148 + v22) )
```

这里是对原来数据加密之后的判断函数，显然我们要在404148处dump出来加密后的数据

写脚本跑出来前面的10位密码（上面的4组位运算方程任意取两位即可，脚本中取得是第1，2个方程）

```
Table1=[0x83EC8B55,
0xEC81F0E4,
0x00000278,
0x405004A1]
Table=[0x7BB39408,
0xEA90BD98,
0xFE0F0B1D,
0x3E5F11D5]
cnst=[0,1,2,3]
cnst1=[4,4,4,4]
Table2=[0x85CAFC37,
0xFC3AF1B4,
0xEE160D39,
0x3E507535]
Table3=[0x89C43300,
0x02742484,
0x100F0000,
0x4041A805
]

for i in range(0x100):
    val=[0,0,0,0]
    for k in range(4):
        val[k]=(i<<24)|(i<<16)|(i<<8)|i
    temp1=[0,0,0,0]
    for k in range(4):
        temp1[k]=(Table[k]+val[k])&0xffffffff
    for k in range(4):
        temp1[k]^=Table1[k]
    for k in range(4):
        if temp1[k]<cnst[k]:
            temp1[k]|=0x100000000
        temp1[k]-=cnst[k]
    temp2=[0,0,0,0]
    temp3=[0,0,0,0]
    for k in range(4):
        temp2[k]=(cnst[k]+cnst1[k]+temp1[k])&0xffffffff
    for k in range(4):
        temp3[k]=(val[k]+Table2[k])&0xffffffff
    for k in range(4):
        temp2[k]^=temp3[k]
    if temp2==Table3:
        print (hex(i),hex(temp1[0]))
```

继续跟进，在这个函数快要结尾处可以看到往内存中的某一段写入了一些东西

```
v28 = 0;
v23 = __rdtsc();
if ( HIDWORD(v23) <= HIDWORD(v26) )
  v28 = v23 - v26;
v24 = GetCurrentProcess();
WriteProcessMemory(v24, sub_4018A0, &xmmword_405018, 0x60u, 0);
return 1;
```

并且写入的地方在本判断函数结束后要被调用，所以继续跟进，发现接下来的是对剩下的16位数字进行3DES加密处理。然后再将加密后的内容与后面的数据比较

其中Padding方式为PKCS7，在对3DES加密函数的动态调试中也可以发现密钥为AFSAFCEDYCXCXACNDFKDCQXC，

加密之后的HEX为507CA9E68709CEFA20D50DCF90BB976C9090F6B07BA6A4E8，网上找个轮子就能跑出来答案：0dcc509a6f75849b

然后和前面的10个密码组合一下即可