XCTF EASYHOOK

原创

酸酸菜鱼 于 2020-08-03 17:16:35 发布 195 收藏
分类专栏： CTF

1. 使用动态调试的方式发现在哪里进行加密，得出加密函数sub_401000。
2. 有个this_is_not_flag的幌子
3. 算法函数在sub_401000中

```
i = 0;
if ( a2 > 0 )
{
  do                                    // 输入长度为18
  {
    if ( i == 18 )
    {
      *(_BYTE *)(input + 18) ^= 0x13u;  // 输入最后一个要^0x13  input[18]^0x13
    }
    else                                // 根据i%2不同的值，进行不同的计算
    {
      if ( i % 2 )
        v3 = *(_BYTE *)(i + input) - i;       // input[i]=input[i]-i
      else
        v3 = *(_BYTE *)(i + input + 2);       // input[i]=input[i+2]^i
      *(_BYTE *)(i + input) = i ^ v3;
    }
    ++i;
  }
  while ( i < a2 );
}
v4 = 0;
if ( a2 <= 0 )
  return 1;
v5 = 0;
while ( byte_40A030[v5] == *(_BYTE *)(v5 + input) )// byte_40A030 = [0x61, 0x6A, 0x79, 0x67, 0x6B, 0x46, 0x6D, 0x2E, 0x7F, 0x5F,
{                                       // 最后与该字符串数组比较
  v5 = ++v4;
  if ( v4 >= a2 )
    return 1;
```

```
"""
正向思路：
1.对输入的末位计算
2.i%2为奇数时：
    (input[i]-i)^i
  i%2为偶数时：
    (input[i+2])^i
逆向思路
1.求出末位
2.反过来求flag[i]和flag[i+2]
"""
str_list = [0x61, 0x6A, 0x79, 0x67, 0x6B, 0x46, 0x6D, 0x2E, 0x7F, 0x5F, 0x7E, 0x2D, 0x53, 0x56, 0x7B, 0x38,
print(len(str_list))
flag = list("1234567890123456789")
v3 = 0
for i in range(18):
    if i % 2:
        flag[i] = chr((str_list[i] ^ i) + i)
    else:
        flag[i+2] = chr(str_list[i] ^ i)

print("".join(flag)+chr(str_list[18] ^ 0x13))

# lag{Ho0k_w1th_Fun}
#要在前边补个f，得出最终结果 flag{Ho0k_w1th_Fun}
```