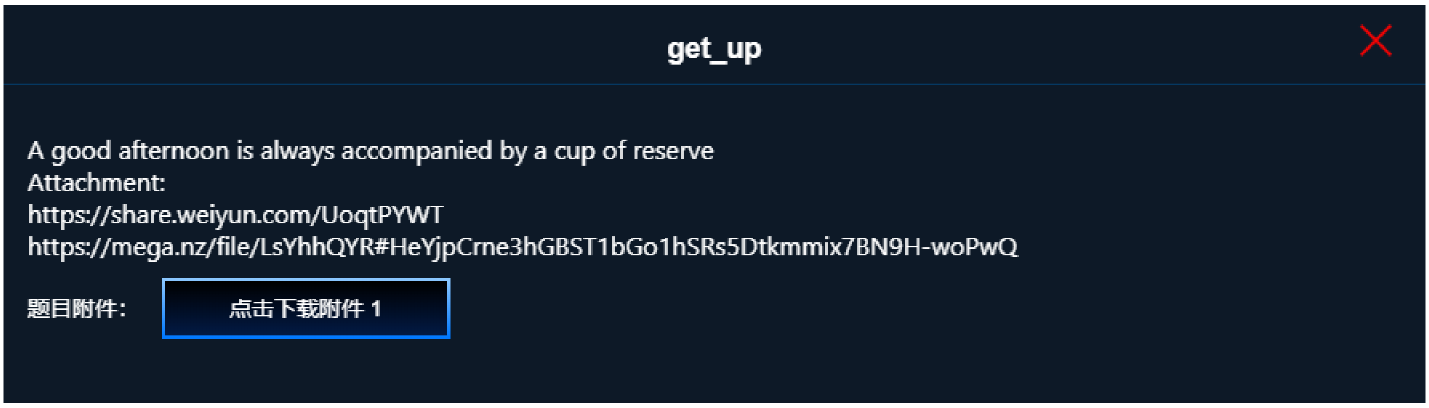
# 【SCTF2020】get_up WriteUp

古月浪子 于 2020-07-06 11:01:56 发布 363 收藏
文章标签： CTF

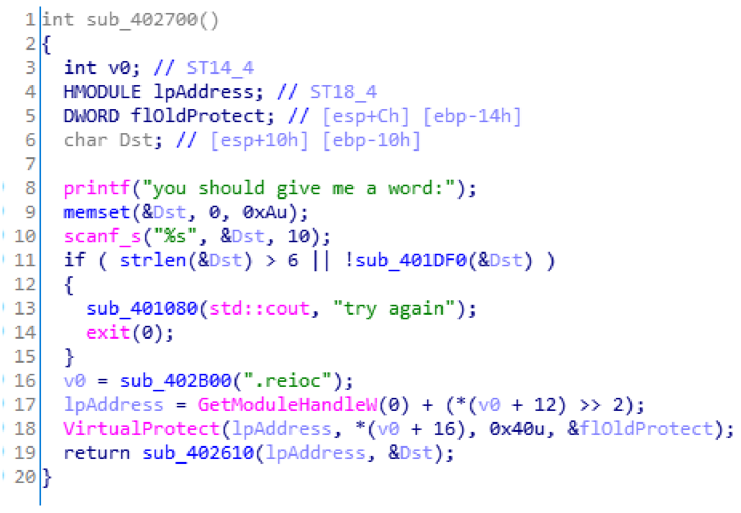
一道SCTF的逆向题，做了很久很久才做起，唉…


get_up

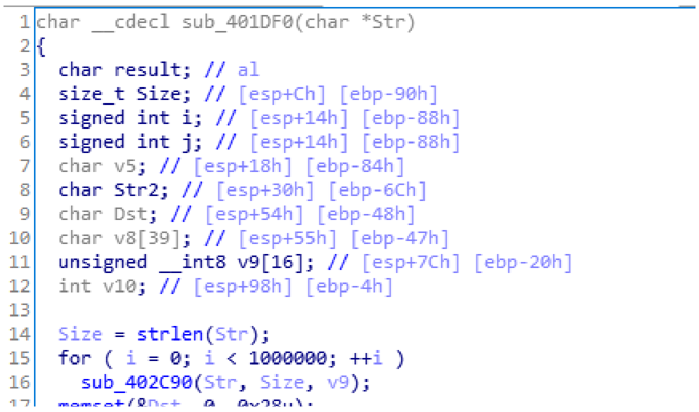
A good afternoon is always accompanied by a cup of reserve
Attachment:
https://share.weiyun.com/UoqtPYWT
https://mega.nz/file/LsYhhQYR#HeYjpCrne3hGBST1bGo1hSRs5Dtkmmix7BN9H-woPwQ

题目附件： 点击下载附件 1

IDA打开，用shift+f12查找出现的字符串，再通过x交叉引用定位到主函数


```
int sub_402700()
{
  int v0; // ST14_4
  HMODULE lpAddress; // ST18_4
  DWORD flOldProtect; // [esp+Ch] [ebp-14h]
  char Dst; // [esp+10h] [ebp-10h]

  printf("you should give me a word:");
  memset(&Dst, 0, 0xAu);
  scanf_s("%s", &Dst, 10);
  if ( strlen(&Dst) > 6 || !sub_401DF0(&Dst) )
  {
    sub_401080(std::cout, "try again");
    exit(0);
  }
  v0 = sub_402B00(".reioc");
  lpAddress = GetModuleHandleW(0) + (*(v0 + 12) >> 2);
  VirtualProtect(lpAddress, *(v0 + 16), 0x40u, &flOldProtect);
  return sub_402610(lpAddress, &Dst);
}
```

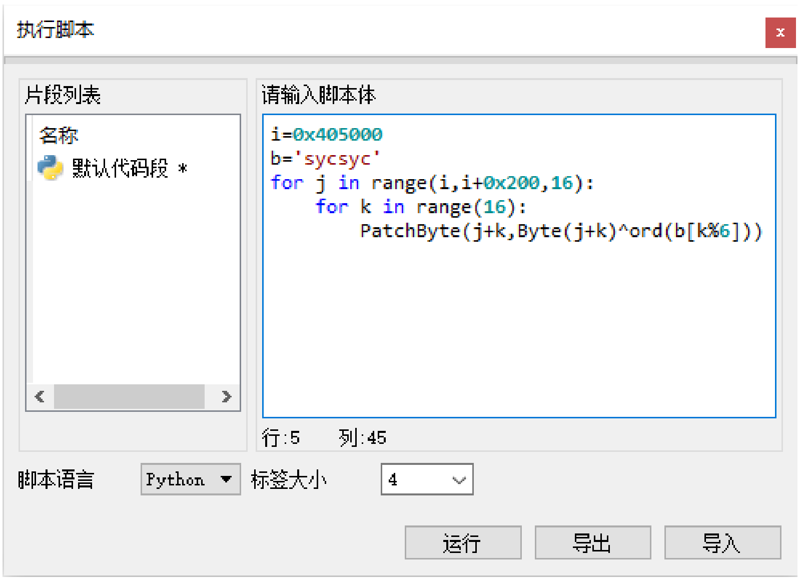
根据伪代码，大概是判断我们输入的一个长度小于等于6的字符串，跟进判断函数看一看


```
char __cdecl sub_401DF0(char *Str)
{
  char result; // al
  size_t Size; // [esp+Ch] [ebp-90h]
  signed int i; // [esp+14h] [ebp-88h]
  signed int j; // [esp+14h] [ebp-88h]
  char v5; // [esp+18h] [ebp-84h]
  char Str2; // [esp+30h] [ebp-6Ch]
  char Dst; // [esp+54h] [ebp-48h]
  char v8[39]; // [esp+55h] [ebp-47h]
  unsigned __int8 v9[16]; // [esp+7Ch] [ebp-20h]
  int v10; // [esp+98h] [ebp-4h]

  Size = strlen(Str);
  for ( i = 0; i < 1000000; ++i )
    sub_402C90(Str, Size, v9);
  memset(&Dst, 0, 0x28u);
  memset(&Dst, 0, 0x20u);
  sub_4015C0("0123456789abcdef");
  v10 = 0;
  for ( j = 0; j < 16; ++j )
  {
    *(&Dst + 2 * j) = *sub_4019A0(v9[j] >> 4);
    v8[2 * j] = *sub_4019A0(v9[j] % 16);
  }
  printf("\n");
  strcpy(&Str2, "32c1d123c193aecc4280a5d7925a2504");
  if ( !strcmp(&Dst, &Str2) )
  {
    v10 = -1;
    sub_4017E0(&v5);
    result = 1;
  }
  else
  {
    v10 = -1;
    sub_4017E0(&v5);
    result = 0;
  }
  return result;
}
```

看到32位的a-f+0-9的字符串，盲猜哈希，又因为输入的长度小于等于6，肯定能爆破出来，于是丢到cmd5网站上，解出来“sycsyc”

```
void __cdecl sub_402610(int a1, char *Str)
{
  char v2; // bl
  signed int i; // [esp+8h] [ebp-4h]

  for ( i = 0; i < 16; ++i )
  {
    v2 = *(i + a1);
    *(i + a1) = Str[i % strlen(Str)] ^ v2;
  }
}
```

成功绕过第一个判断以后，下面的代码大致意思是查找.reioc段并且改为可写，然后用刚刚输入的字符串进行异或，这里用IDCPython脚本处理一下



```
i=0x405000
b='sycsyc'
for j in range(i,i+0x200,16):
    for k in range(16):
        PatchByte(j+k,Byte(j+k)^ord(b[k%6]))
```

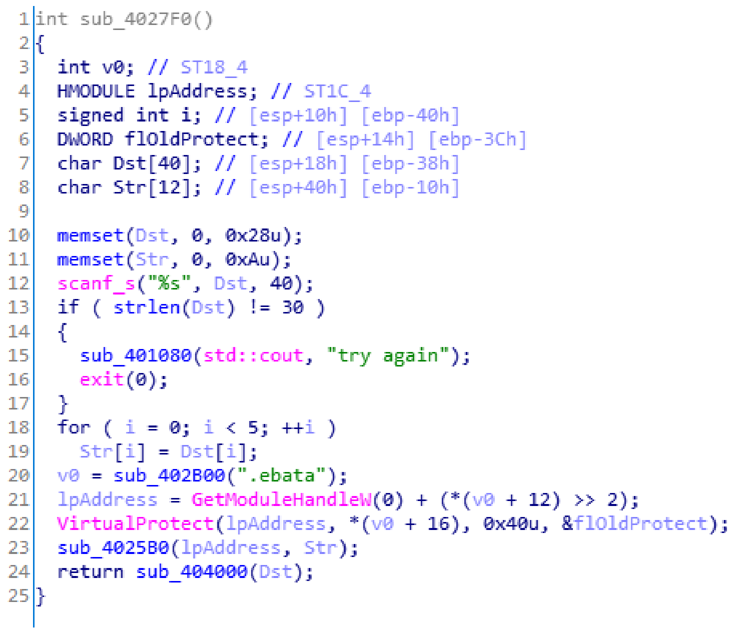
行:5 列:45

脚本语言 Python 标签大小 4

可以看到处理完后.reioc段变成了这个样子（先u将数据转为未定义，再p转为函数）

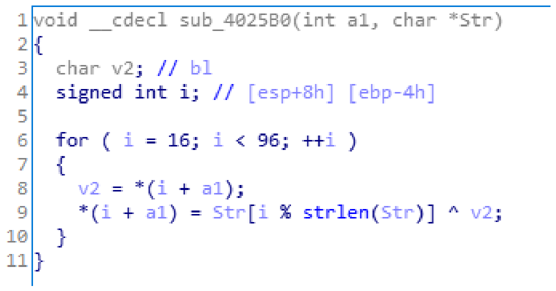
```
.reioc:00405000 sub_405000      proc near               ; CODE XREF: sub_403230+8↑p
.reioc:00405000                 push    ebp
.reioc:00405001                 mov     ebp, esp
.reioc:00405003                 call    sub_403120
.reioc:00405008                 call    sub_4027F0
.reioc:0040500D                 pop     ebp
.reioc:0040500E                 retn
.reioc:0040500E sub_405000      endp
```

第一个call是printf让你输入flag，第二个call比较关键。先读取输入的字符，要求长度为30，然后找到.ebata段并对其进行动态patch



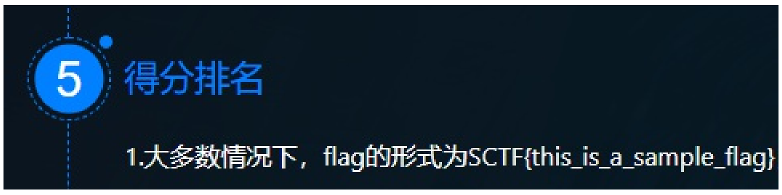
```
int sub_4027F0()
{
  int v0; // ST18_4
  HMODULE lpAddress; // ST1C_4
  signed int i; // [esp+10h] [ebp-40h]
  DWORD flOldProtect; // [esp+14h] [ebp-3Ch]
  char Dst[40]; // [esp+18h] [ebp-38h]
  char Str[12]; // [esp+40h] [ebp-10h]

  memset(Dst, 0, 0x28u);
  memset(Str, 0, 0xAu);
  scanf_s("%s", Dst, 40);
  if ( strlen(Dst) != 30 )
  {
    sub_401080(std::cout, "try again");
    exit(0);
  }
  for ( i = 0; i < 5; ++i )
    Str[i] = Dst[i];
  v0 = sub_402B00(".ebata");
  lpAddress = GetModuleHandleW(0) + (*(v0 + 12) >> 2);
  VirtualProtect(lpAddress, *(v0 + 16), 0x40u, &flOldProtect);
  sub_4025B0(lpAddress, Str);
  return sub_404000(Dst);
}
```

具体的patch方法是，取输入的前5个字符，进行异或

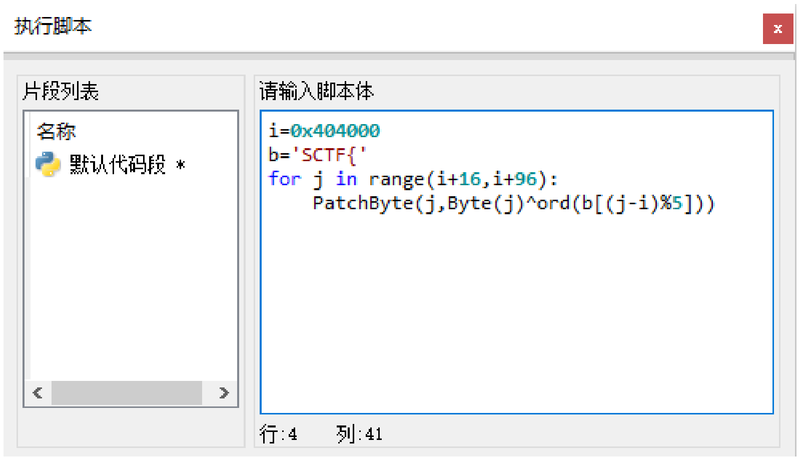


```
void __cdecl sub_4025B0(int a1, char *Str)
{
  char v2; // bl
  signed int i; // [esp+8h] [ebp-4h]

  for ( i = 16; i < 96; ++i )
  {
    v2 = *(i + a1);
    *(i + a1) = Str[i % strlen(Str)] ^ v2;
  }
}
```

再写脚本处理一下。前5个字符到底是什么呢？~~盲猜是4个字母+1个大括号，因此有“flag{”、“sctf{”、“SCTF{”、“FLAG{”四种可能的开头方式（以我心目中的可能性降序排序），因此全部试一遍~~ 比赛官网上面写了flag格式



1.大多数情况下，flag的形式为SCTF{this_is_a_sample_flag}

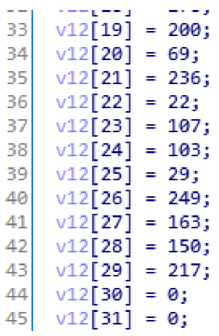


```
i=0x404000
b='SCTF{'
for j in range(i+16,i+96):
    PatchByte(j,Byte(j)^ord(b[(j-i)%5]))
```

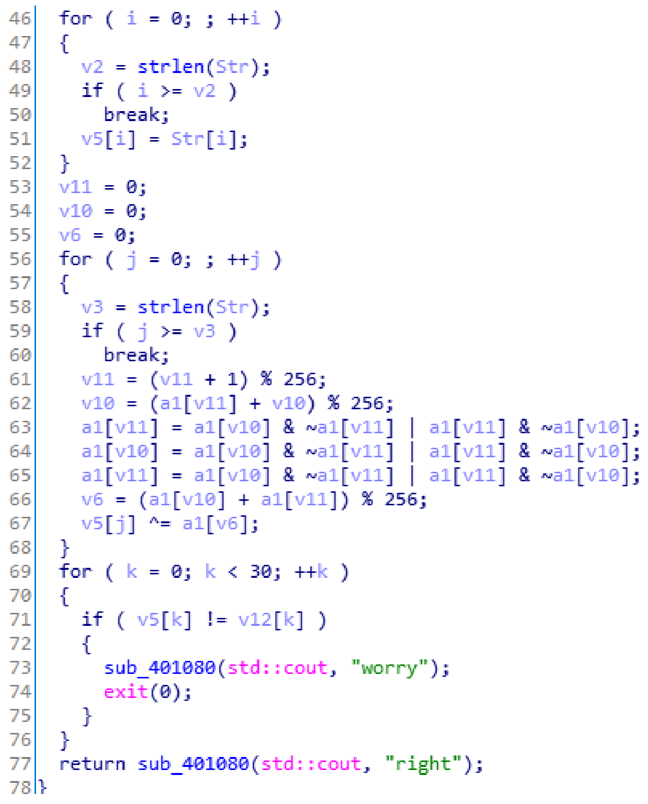
老样子，将被动态patch的数据转为函数，即可在f5里看到sub_*开头的函数啦

```c
int __cdecl sub_404000(char *Str)
{
  size_t v1; // eax
  int v3[300]; // [esp+0h] [ebp-9ACh]
  int v4; // [esp+4B0h] [ebp-4FCh]
  int v5; // [esp+4B4h] [ebp-4F8h]
  size_t i; // [esp+4B8h] [ebp-4F4h]
  unsigned int j; // [esp+4BCh] [ebp-4F0h]
  int k; // [esp+4C0h] [ebp-4ECh]
  int v9[300]; // [esp+4C4h] [ebp-4E8h]
  char Dst[40]; // [esp+974h] [ebp-38h]
  char v11[9]; // [esp+99Ch] [ebp-10h]

  v11[0] = 's';
  v11[1] = 'y';
  v11[2] = 'c';
  v11[3] = 'l';
  v11[4] = 'o';
  v11[5] = 'v';
  v11[6] = 'e';
  v11[7] = 'r';
  v11[8] = 0;
  memset(Dst, 0, 0x28u);
  for ( i = 0; i < strlen(Str); ++i )
    Dst[i] = Str[i];
  for ( j = 0; j < 256; ++j )
  {
    v9[j] = j;
    v1 = strlen(v11);
    v3[j] = v11[j % v1];
  }
  v5 = 0;
  for ( k = 0; k < 256; ++k )
  {
    v5 = (v3[k] + v9[k] + v5) % 256;
    v4 = v9[k];
    v9[k] = v9[v5];
    v9[v5] = v4;
  }
  return sub_401A70(v9, Dst);
}
```

```c
int __cdecl sub_401A70(int *a1, char *Str)
{
  size_t v2; // eax
  size_t v3; // eax
  int v5[30]; // [esp+4h] [ebp-114h]
  int v6; // [esp+7Ch] [ebp-9Ch]
  int k; // [esp+80h] [ebp-98h]
  unsigned int i; // [esp+84h] [ebp-94h]
  unsigned int j; // [esp+88h] [ebp-90h]
  int v10; // [esp+8Ch] [ebp-8Ch]
  int v11; // [esp+90h] [ebp-88h]
  int v12[32]; // [esp+94h] [ebp-84h]

  v12[0] = 128;
  v12[1] = 85;
  v12[2] = 126;
  v12[3] = 45;
  v12[4] = 209;
  v12[5] = 9;
  v12[6] = 37;
  v12[7] = 171;
  v12[8] = 60;
  v12[9] = 86;
  v12[10] = 149;
  v12[11] = 196;
  v12[12] = 54;
  v12[13] = 19;
  v12[14] = 237;
  v12[15] = 114;
  v12[16] = 36;
  v12[17] = 147;
  v12[18] = 178;
```



```
v12[19] = 200;
v12[20] = 69;
v12[21] = 236;
v12[22] = 22;
v12[23] = 107;
v12[24] = 103;
v12[25] = 29;
v12[26] = 249;
v12[27] = 163;
v12[28] = 150;
v12[29] = 217;
v12[30] = 0;
v12[31] = 0;
```



```
  for ( i = 0; ; ++i )
  {
    v2 = strlen(Str);
    if ( i >= v2 )
      break;
    v5[i] = Str[i];
  }
  v11 = 0;
  v10 = 0;
  v6 = 0;
  for ( j = 0; ; ++j )
  {
    v3 = strlen(Str);
    if ( j >= v3 )
      break;
    v11 = (v11 + 1) % 256;
    v10 = (a1[v11] + v10) % 256;
    a1[v11] = a1[v10] & ~a1[v11] | a1[v11] & ~a1[v10];
    a1[v10] = a1[v10] & ~a1[v11] | a1[v11] & ~a1[v10];
    a1[v11] = a1[v10] & ~a1[v11] | a1[v11] & ~a1[v10];
    v6 = (a1[v10] + a1[v11]) % 256;
    v5[j] ^= a1[v6];
  }
  for ( k = 0; k < 30; ++k )
  {
    if ( v5[k] != v12[k] )
    {
      sub_401080(std::cout, "worry");
      exit(0);
    }
  }
  return sub_401080(std::cout, "right");
}
```

这里只要合理的用y来修改一下变量类型，就能更好的看清楚算法流程啦~
我们用c++写一下爆破脚本：

```cpp
#include <iostream>

using namespace std;

int sub_401A70(int *a1, char *Str) {
 int v5[30];
 int v6 = 0;
 int v10 = 0;
 int v11 = 0;
 int v12[30] = { 128,85,126,45,209,9,37,171,60,86,149,196,54,19,237,114,36,147,178,200,69,236,22,107,103,29,249,
163,150,217 };
 for (int i = 0; i < strlen(Str); ++i)
  v5[i] = Str[i];
 for (int j = 0; j < strlen(Str); ++j) {
  v11 = (v11 + 1) % 256;
  v10 = (a1[v11] + v10) % 256;
  a1[v11] = a1[v10] & ~a1[v11] | a1[v11] & ~a1[v10];
  a1[v10] = a1[v10] & ~a1[v11] | a1[v11] & ~a1[v10];
  a1[v11] = a1[v10] & ~a1[v11] | a1[v11] & ~a1[v10];
  v6 = (a1[v10] + a1[v11]) % 256;
  v5[j] ^= a1[v6];
 }
 for (int k = 0; k < 30; ++k)
  if (v5[k] != v12[k]) {
   return k + 1;
  }
 return 0;
}

int sub_404000(char *Str) {
 int v3[300];
 int v5 = 0;
 int v9[300];
 char Dst[40];
 char v11[9] = "syclover";
 memset(Dst, 0, 0x28u);
 for (int i = 0; i < strlen(Str); ++i)
  Dst[i] = Str[i];
 for (int j = 0; j < 256; ++j) {
  v9[j] = j;
  v3[j] = v11[j % 8];
 }
 for (int k = 0; k < 256; ++k) {
  v5 = (v3[k] + v9[k] + v5) % 256;
  int tmp = v9[k];
  v9[k] = v9[v5];
  v9[v5] = tmp;
 }
 return sub_401A70(v9, Dst);
}

int main()
{
 char flag[] = "SCTF{111111111111111111111111}";
 for (int i = 5; i < 29; i++)
  flag[i] = 1;
 for (int i = 5; sub_404000(flag);)
  if (sub_404000(flag) == i + 1)
   flag[i]++;
  else
   i++;
 cout << flag;
}
```

成功爆破出flag！

> SCTF{zzz~(|3[___]_rc4_5o_e4sy}