BUU_re_[ACTF新生赛2020]usualCrypt

原创

再搭环境是狗 于 2021-09-04 12:17:21 发布 26 收藏 1

分类专栏： 逆向 文章标签： c语言

本文链接：https://blog.csdn.net/goat_2003/article/details/120089637

逆向 专栏收录该内容

15 篇文章 0 订阅

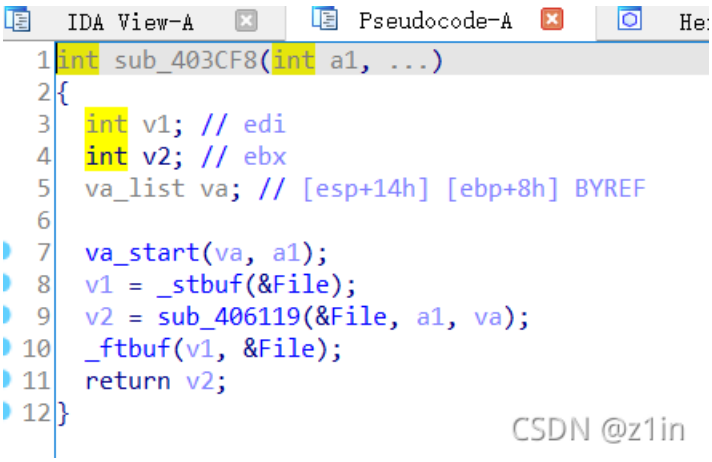
订阅专栏

首先拖入ida中

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // esi
  int result; // eax
  int v5[3]; // [esp+8h] [ebp-74h] BYREF
  __int16 v6; // [esp+14h] [ebp-68h]
  char v7; // [esp+16h] [ebp-66h]
  char v8[100]; // [esp+18h] [ebp-64h] BYREF

  sub_403CF8(&unk_40E140);
  scanf("%s", v8);
  v5[0] = 0;
  v5[1] = 0;
  v5[2] = 0;
  v6 = 0;
  v7 = 0;
  sub_401080(v8, strlen(v8), v5);
  v3 = 0;
  while ( *(v5 + v3) == byte_40E0E4[v3] )
  {
    if ( ++v3 > strlen(v5) )
      goto LABEL_6;
  }
  sub_403CF8(aError);
LABEL_6:
  if ( v3 - 1 == strlen(byte_40E0E4) )
    result = sub_403CF8(aAreYouHappyYes);
  else
    result = sub_403CF8(aAreYouHappyNo);
  return result;
}
```

初步判断一下，显示输入我们的flag，然后加密，然后再比较。

sub_403CF8函数看了大佬的教程可以知道这种函数是printf函数



然后我们再进入中间的sub_401080函数中去看看。

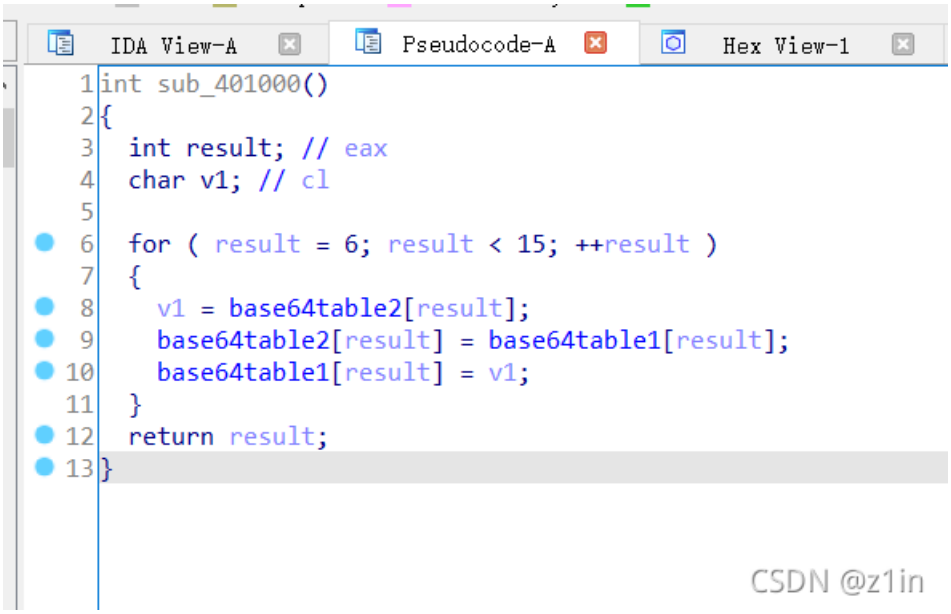
```
int __cdecl sub_401080(int a1, int a2, int a3)
{
  int v3; // edi
  int v4; // esi
  int v5; // edx
  int v6; // eax
  int v7; // ecx
  int v8; // esi
  int v9; // esi
  int v10; // esi
  int v11; // esi
  _BYTE *v12; // ecx
  int v13; // esi
  int v15; // [esp+18h] [ebp+8h]

  v3 = 0;
  v4 = 0;
  sub_401000();
  v5 = a2 % 3;
  v6 = a1;
  v7 = a2 - a2 % 3;
  v15 = a2 % 3;
  if ( v7 > 0 )
  {
    do
    {
      LOBYTE(v5) = *(a1 + v3);
      v3 += 3;
      v8 = v4 + 1;
      *(v8 + a3 - 1) = base64table[(v5 >> 2) & 0x3F];
      *(++v8 + a3 - 1) = base64table[16 * (*(a1 + v3 - 3) & 3) + ((*(a1 + v3 - 2) >> 4) & 0xF)];
      *(++v8 + a3 - 1) = base64table[4 * (*(a1 + v3 - 2) & 0xF) + ((*(a1 + v3 - 1) >> 6) & 3)];
      v5 = *(a1 + v3 - 1) & 0x3F;
      v4 = v8 + 1;
      *(v4 + a3 - 1) = base64table[v5];
    }
    while ( v3 < v7 );
    v5 = v15;
  }
  if ( v5 == 1 )
  {
```

```
    LOBYTE(v7) = *(v3 + a1);
    v9 = v4 + 1;
    *(v9 + a3 - 1) = base64table[(v7 >> 2) & 0x3F];
    v10 = v9 + 1;
    *(v10 + a3 - 1) = base64table[16 * (*(v3 + a1) & 3)];
    *(v10 + a3) = 61;
LABEL_8:
    v13 = v10 + 1;
    *(v13 + a3) = 61;
    v4 = v13 + 1;
    goto LABEL_9;
  }
  if ( v5 == 2 )
  {
    v11 = v4 + 1;
    *(v11 + a3 - 1) = base64table[(*(v3 + a1) >> 2) & 0x3F];
    v12 = (v3 + a1 + 1);
    LOBYTE(v6) = *v12;
    v10 = v11 + 1;
    *(v10 + a3 - 1) = base64table[16 * (*(v3 + a1) & 3) + ((v6 >> 4) & 0xF)];
    *(v10 + a3) = base64table[4 * (*v12 & 0xF)];
    goto LABEL_8;
  }
LABEL_9:
  *(v4 + a3) = 0;
  return sub_401030(a3);
}
```

还是看了一些大佬的得知，这大概是一个base64加密。为什么呢？可以通过base64转换表得知，点开数组，可以看到储存的数据为：

```
.data:0040E0A0 ; char base64table[10]
.data:0040E0A0 base64table     db 41h                  ; 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 byte_40E0AA     db 4Bh                  ; DATA XREF: sub_401000+B↑r
.data:0040E0AA                                         ; sub_401000+11↑w
.data:0040E0AB aLmnopqrstuvwxy db 'LMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/',0
.data:0040E0E1                 align 4
```

这个便是base64转换表。接下来看一下开头的sub_401000函数，



这个函数改变了部分对应表。但base64加密的规则并没有改变。

再进入下面的sub_401030函数中看一下

```
int __cdecl sub_401030(const char *a1)
{
  __int64 v1; // rax
  char v2; // al

  v1 = 0i64;
  if ( strlen(a1) )
  {
    do
    {
      v2 = a1[HIDWORD(v1)];
      if ( v2 < 97 || v2 > 122 )
      {
        if ( v2 < 65 || v2 > 90 )
          goto LABEL_9;
        LOBYTE(v1) = v2 + 32;
      }
      else
      {
        LOBYTE(v1) = v2 - 32;
      }
      a1[HIDWORD(v1)] = v1;
LABEL_9:
      LODWORD(v1) = 0;
      ++HIDWORD(v1);
    }
    while ( HIDWORD(v1) < strlen(a1) );
  }
  return v1;
}
```

可以看出这是一个转换字母大小写的函数。
那么现在开始最重要的环节：写脚本

```
# -*- coding:utf-8 -*-
import base64

table = list("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/")
model = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
for i in range(6,15):  #实现了变表
    table[i],table[i+10]=table[i+10],table[i]
table = ''.join(table)
swap="zMXHz3TIgnxLxJhFAdtZn2fFk3lYCrtPC2l9".swapcase()  #转换字母大小写
dec=''
for i in range(len(swap)):
    dec += model[table.find(swap[i])]    #恢复为变表前的字符串
print(base64.b64decode(dec))
```

得出flag：flag{bAse64_h2s_a_Surprise}