re学习笔记（20）i春秋-re-Nonstandard

原创

Forgo7ten 于 2019-12-29 11:53:59 发布 263 收藏

分类专栏： ctf小白成长ing # reverse 文章标签： ctf i春秋 Nonstandard reverse

本文链接：https://blog.csdn.net/Palmer9/article/details/103588203

新手一枚，如有错误（不足）请指正，谢谢！！

题目下载：下载地址

参考资料：

base16，base32，base64 编码方式的通俗讲解

Base16,Base32,Base64编码的介绍

IDA载入，找到main函数

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    FILE *v3; // eax
4    FILE *v4; // eax
5    FILE *v5; // eax
6    char Buf[16]; // [esp+0h] [ebp-24h]
7    __int64 v8; // [esp+10h] [ebp-14h]
8    int v9; // [esp+18h] [ebp-Ch]
9    __int16 v10; // [esp+1Ch] [ebp-8h]
10
11   v9 = 0;
12   _mm_storeu_si128((__m128i *)Buf, (__m128i)0i64);
13   v10 = 0;
14   v8 = 0i64;
15   v3 = _iob_func();
16   fputs("Place Input Flag:\n", v3 + 1);
17   v4 = _iob_func();
18   fgets(Buf, 29, v4);
19   if ( sub_401480(Buf) == 1 )
20   {
21     v5 = _iob_func();
22     fputs("yes\n", v5 + 1);
23   }
24   return 0;
25 }
```


Buf为读入的字符串，如果sub_401480()函数的返回值等于1的话则输出成功

则进入sub_401480()查看

```
signed int __thiscall sub_401480(const char *this)
{
  const char *v1; // esi
  const char *v2; // eax
  unsigned int v3; // eax
  unsigned int v4; // kr04_4
  signed int result; // eax
  char v6; // [esp+4h] [ebp-38h]
  char Dst; // [esp+5h] [ebp-37h]

  v6 = 0;
  v1 = this;                                    // v1为输入的字符串
  memset(&Dst, 0, 0x31u);                       // 对Dst数组清零
  if ( strlen(v1) != 28 )                       // 长度要等于28
    goto LABEL_10;
  v2 = sub_401070((int)v1, 28u);                // v2为sub_401070的返回值
  strncpy_s(&v6, 50u, v2, 48u);                 // 将v2复制给v6
  v3 = 0;
  v4 = strlen(&v6);                             // v4等于v6的长度
  if ( !v4 )
    goto LABEL_10;
  do
  {
    if ( byte_402120[v3] != *(&v6 + v3) )       // 对每位进行遍历，要相等
                                                // 其中byte_402120 = "nAdtxA66nbbdxA71tUAE2AOlnnbtrAp1nQzGtAQGtrjC7==="
      break;
    ++v3;
  }
  while ( v3 < v4 );
  if ( v3 == 48 )                               // 遍历完48个字符都相等则返回值为1
    result = 1;
  else
LABEL_10:
    result = -1;
  return result;
}
```


所以先看看sub_401070()函数中对v1做了怎样的变换

```
int v35; // [esp+44h] [ebp-4h]

v2 = a1;
v30 = a2;
v35 = a1;
sub_401000();
v3 = 0;
v4 = 0;
v28 = 0;
v5 = 0;
v25 = 0;
v6 = 0;
if ( v30 )                                      // Base32加密
{
  do
  {
    if ( !*(_BYTE *)(v6 + v2) )                 // v2为字符串首地址，v6为0开始遍历
      break;
    ++v6;
    v5 += 8;
    ++v3;
  }
  while ( v6 < v30 );                           // 循环完，v5为将所有字符转换为二进制位、的位数
                                                //
  v28 = v3;
}
switch ( v5 % 40 )                              // 这里对位数是否能整除40做判断，来判断是否添加'='
                                                // v4等于多少就添加几个'='
```


下面是Base32加密，可参考文章Base16,Base32,Base64编码详细学习

然后分析一下sub_401000()函数，主要是对内存中字符串进行处理

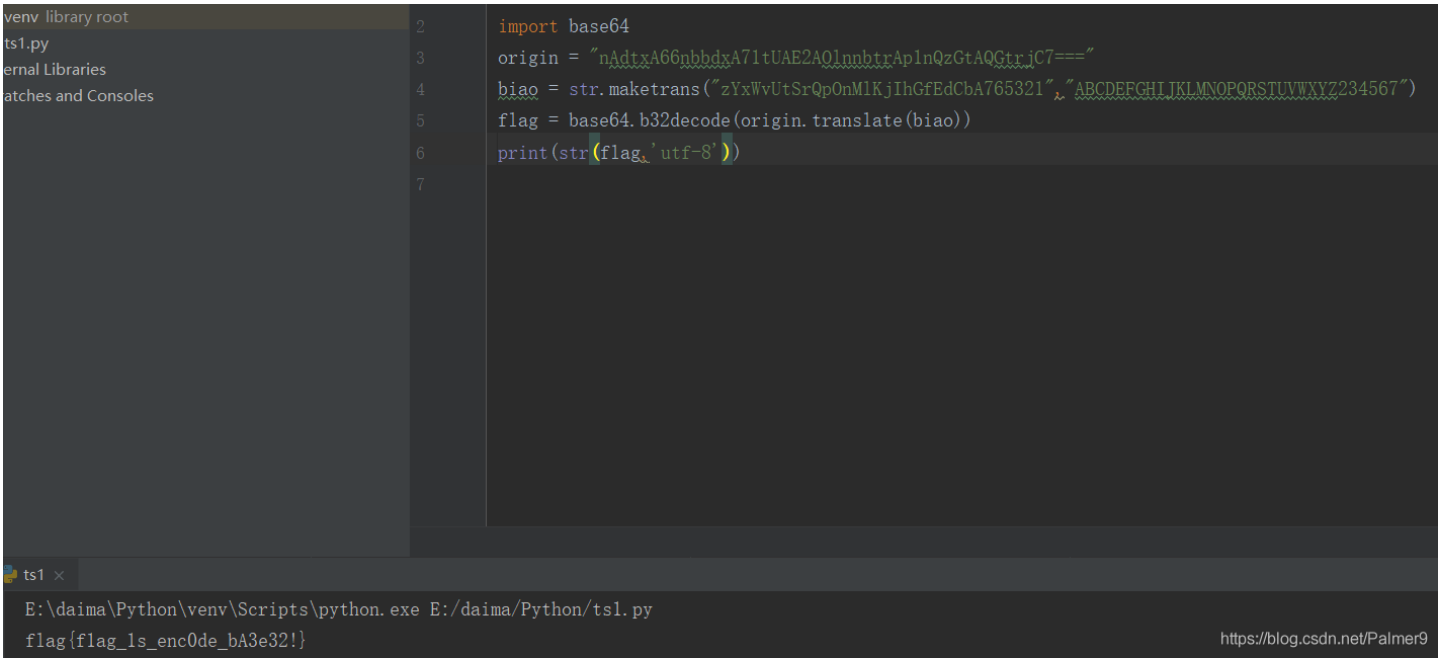
```
signed __int16 sub_401000()
{
  signed int v0; // eax
  int v1; // esi
  char *v2; // edx
  char v3; // cl
  signed __int16 result; // ax

  v0 = 1;                                       // 存储完应该为：zYxW
  do
  {
    byte_403020[v0] += 32;                      // 对索引为奇数的元素+32
                                                // 也就是对偶数位的元素+32
                                                // 将其转换为小写
    v0 += 2;
  }                                             // 修改完
                                                // byte_403020变为
                                                // AbCdEfGhIjKlMnOpQrStUvWxYz
  while ( v0 < 26 );
  v1 = 0;
  v2 = &aMnopqrstuvwxyz[13];                    // v2等于最后一位
  do
  {
    v3 = byte_40301F[++v1];                     // 将v3赋值  从这个数组头开始，每次往后移动一个字节
    byte_40301F[v1] = *v2;                      // 将v2给从数组头开始遍历的……
    *v2-- = v3;                                 // v2等于从数组头开始遍历的……然后地址向低地址移动1字节
  }
  while ( (signed int)v2 > (signed int)aMnopqrstuvwxyz );// 两边是十三个，也就是将其对称换一下位置
                                                // 这个循环的作用：将内存中的字母表倒序排列
  *(_DWORD *)&aMnopqrstuvwxyz[14] = '3567';     // 计算机中是按小端序存储
  result = '12';
  word_40303E = '12';
  byte_403040 = 0;                              // 存储完应该为：zYxWvUtSrQpOnMlKjIhGfEdCbA765321
  return result;
}
```


也就是一个base32加密，字符集是`zYxWvUtSrQpOnMlKjIhGfEdCbA765321`
加密后的密文是`nAdtxA66nbbdxA71tUAE2AOlnnbtrAp1nQzGtAQGtrjC7===`

然后开始写python脚本

```
import base64
origin = "nAdtxA66nbbdxA71tUAE2AOlnnbtrAp1nQzGtAQGtrjC7==="
biao = str.maketrans("zYxWvUtSrQpOnMlKjIhGfEdCbA765321","ABCDEFGHIJKLMNOPQRSTUVWXYZ234567")
flag = base64.b32decode(origin.translate(biao))
print(str(flag,'utf-8'))
```



最终flag为flag{f1ag_1s_enc0de_bA3e32!}