BUU ctf 之[ACTF新生赛2020]rome

原创

不会掉发的小鲁 于 2020-07-26 22:29:44 发布 819 收藏 1

分类专栏： 安全 ctf

本文链接：https://blog.csdn.net/weixin_47158947/article/details/107601037

 安全 同时被 2 个专栏收录

15 篇文章 0 订阅

订阅专栏

 ctf

16 篇文章 0 订阅

订阅专栏

1.这是一道简单的写脚本的题吧，不是很难。

2.查看字符串，定位，查看伪代码

```
int func()
{
  int result; // eax
  int v1; // [esp+14h] [ebp-44h]
  int v2; // [esp+18h] [ebp-40h]
  int v3; // [esp+1Ch] [ebp-3Ch]
  int v4; // [esp+20h] [ebp-38h]
  unsigned __int8 v5; // [esp+24h] [ebp-34h]
  unsigned __int8 v6; // [esp+25h] [ebp-33h]
  unsigned __int8 v7; // [esp+26h] [ebp-32h]
  unsigned __int8 v8; // [esp+27h] [ebp-31h]
  unsigned __int8 v9; // [esp+28h] [ebp-30h]
  int v10; // [esp+29h] [ebp-2Fh]
  int v11; // [esp+2Dh] [ebp-2Bh]
  int v12; // [esp+31h] [ebp-27h]
  int v13; // [esp+35h] [ebp-23h]
  unsigned __int8 v14; // [esp+39h] [ebp-1Fh]
  char v15; // [esp+3Bh] [ebp-1Dh]
  char v16; // [esp+3Ch] [ebp-1Ch]
  char v17; // [esp+3Dh] [ebp-1Bh]
  char v18; // [esp+3Eh] [ebp-1Ah]
  char v19; // [esp+3Fh] [ebp-19h]
  char v20; // [esp+40h] [ebp-18h]
  char v21; // [esp+41h] [ebp-17h]
  char v22; // [esp+42h] [ebp-16h]
  char v23; // [esp+43h] [ebp-15h]
  char v24; // [esp+44h] [ebp-14h]
  char v25; // [esp+45h] [ebp-13h]
  char v26; // [esp+46h] [ebp-12h]
  char v27; // [esp+47h] [ebp-11h]
  char v28; // [esp+48h] [ebp-10h]
  char v29; // [esp+49h] [ebp-Fh]
```

```
char v30; // [esp+4Ah] [ebp-Eh]
char v31; // [esp+4Bh] [ebp-Dh]
int i; // [esp+4Ch] [ebp-Ch]

v15 = 81;
v16 = 115;
v17 = 119;
v18 = 51;
v19 = 115;
v20 = 106;
v21 = 95;
v22 = 108;
v23 = 122;
v24 = 52;
v25 = 95;
v26 = 85;
v27 = 106;
v28 = 119;
v29 = 64;
v30 = 108;
v31 = 0;
printf("Please input:");
scanf("%s", &v5);
result = v5;
if ( v5 == 'A' )
{
  result = v6;
  if ( v6 == 'C' )
  {
    result = v7;
    if ( v7 == 'T' )
    {
      result = v8;
      if ( v8 == 'F' )
      {
        result = v9;
        if ( v9 == '{' )
        {
          result = v14;
          if ( v14 == '}' )
          {
            v1 = v10;
            v2 = v11;
            v3 = v12;
            v4 = v13;
            for ( i = 0; i <= 15; ++i )
            {
              if ( *((_BYTE *)&v1 + i) > 64 && *((_BYTE *)&v1 + i) <= 90 )
                *((_BYTE *)&v1 + i) = (*((char *)&v1 + i) - 51) % 26 + 65;
              if ( *((_BYTE *)&v1 + i) > 96 && *((_BYTE *)&v1 + i) <= 122 )
                *((_BYTE *)&v1 + i) = (*((char *)&v1 + i) - 79) % 26 + 97;
            }
            for ( i = 0; i <= 15; ++i )
            {
              result = (unsigned __int8)*(&v15 + i);
              if ( *((_BYTE *)&v1 + i) != (_BYTE)result )
                return result;
            }
            result = printf("You are correct!");
          }
```

```
          }
        }
      }
    }
  }
  return result;
}
```

下面是字符串操作的部分

```
if ( *((_BYTE *)&v1 + i) > 64 && *((_BYTE *)&v1 + i) <= 90 )
                  *((_BYTE *)&v1 + i) = (*((char *)&v1 + i) - 51) % 26 + 65;
                if ( *((_BYTE *)&v1 + i) > 96 && *((_BYTE *)&v1 + i) <= 122 )
                  *((_BYTE *)&v1 + i) = (*((char *)&v1 + i) - 79) % 26 + 97;
```

你得和上面的V15后面的字符串相等才可以。下面是代码

```
 for ( i = 0; i <= 15; ++i )
              {
                result = (unsigned __int8)*(&v15 + i);
                if ( *((_BYTE *)&v1 + i) != (_BYTE)result )
                  return result;
              }
```

脚本

```
import string

model = [81,115,119,51,115,106,95,108,122,52,95,85,106,119,64,108]

s1 = string.ascii_lowercase
s2 = string.ascii_uppercase

flag = ""

for i in model:
    if i > 64 and i <= 90:
        flag += s2[i-14-65]
    elif i > 96 and i <= 122:
        flag += s1[i-18-97]
    else:
        flag += chr(i)
print ('flag{'+flag+'}')
```

flag{Cae3ar_th4_Gre@t}