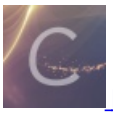
buu Reverse学习记录(26) [ACTF新生赛2020]rome

原创

EMsheep 于 2021-03-15 19:59:22 发布 55 收藏

分类专栏： buu reverse

本文链接：https://blog.csdn.net/EMsheep/article/details/114679379



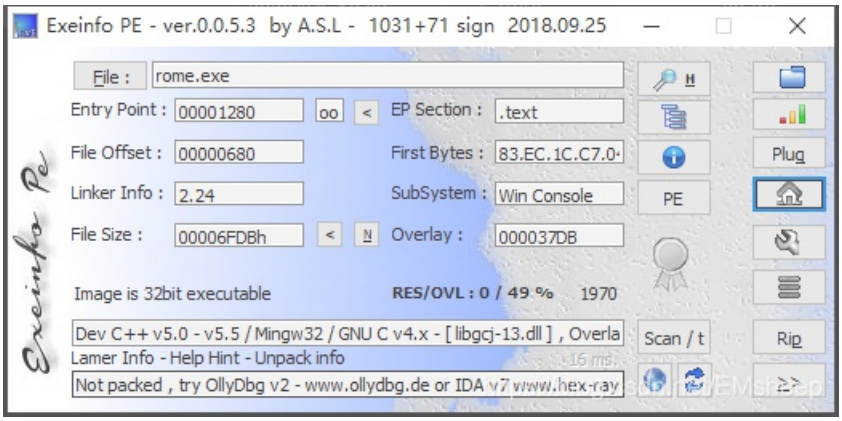
buu reverse 专栏收录该内容

30 篇文章 1 订阅

订阅专栏

题目链接：https://buuoj.cn/challenges#[ACTF%E6%96%B0%E7%94%9F%E8%B5%9B2020]rome

把题目拖进exeinfo中看一下，是个32位程序，无壳



把题目拖进IDA中，可以看到一个func()函数

```
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]
```

```
30  char v27; // [esp+47h] [ebp-11h]
31  char v28; // [esp+48h] [ebp-10h]
32  char v29; // [esp+49h] [ebp-Fh]
33  char v30; // [esp+4Ah] [ebp-Eh]
34  char v31; // [esp+4Bh] [ebp-Dh]
35  int i; // [esp+4Ch] [ebp-Ch]
36
37  v15 = 81;
38  v16 = 115;
39  v17 = 119;
40  v18 = 51;
41  v19 = 115;
42  v20 = 106;
43  v21 = 95;
44  v22 = 108;
45  v23 = 122;
46  v24 = 52;
47  v25 = 95;
48  v26 = 85;
49  v27 = 106;
50  v28 = 119;
51  v29 = 64;
52  v30 = 108;
53  v31 = 0;
54  printf("Please input:");
55  scanf("%s", &v5);
56  result = v5;
57  if ( v5 == 65 )
58  {
59    result = v6;
60    if ( v6 == 67 )
61    {
62      result = v7;
63      if ( v7 == 84 )
64      {
65        result = v8;
66        if ( v8 == 70 )
67        {
68          result = v9;
69          if ( v9 == 123 )
70          {
71            result = v14;
72            if ( v14 == 125 )
73            {
74              v1 = v10;
75              v2 = v11;
76              v3 = v12;
77              v4 = v13;
78              for ( i = 0; i <= 15; ++i )
79              {
80                if ( *((_BYTE *)&v1 + i) > 64 && *((_BYTE *)&v1 + i) <= 90 )
81                  *((_BYTE *)&v1 + i) = (*((char *)&v1 + i) - 51) % 26 + 65;
82                if ( *((_BYTE *)&v1 + i) > 96 && *((_BYTE *)&v1 + i) <= 122 )
83                  *((_BYTE *)&v1 + i) = (*((char *)&v1 + i) - 79) % 26 + 97;
84              }
85              for ( i = 0; i <= 15; ++i )
86              {
87                result = (unsigned __int8)*(&v15 + i);
88                if ( *((_BYTE *)&v1 + i) != (_BYTE)result )
89                  return result;
90              }
91              result = printf("You are correct!");
92            }
93          }
94        }
95      }
96    }
97  }
98  return result;
99}
```


这就是简单的加密函数，写个脚本破解一下。

通过观察可以发现，对大写字母的加密是加了14，对小写字母的加密是加了18
也可以一个个试，暴力破解

```
s = [81,115,119,51,115,106,95,108,122,52,95,85,106,119,64,108]
flag = ''
for i in s:
   if i > 64 and i <= 90:
       i -= 14
       if i < 65:
           i += 26
       flag += chr(i)
   elif i > 96 and i <= 122:
       i -= 18
       if i < 97:
           i += 26
       flag += chr(i)
   else:
       flag += chr(i)
print(flag)
```

flag:Cae3ar_th4_Gre@t