攻防世界 Reverse高手进阶区 2分题 ReverseMe-120

原创

思源湖的鱼 于 2021-01-10 15:24:38 发布 143 收藏
分类专栏： ctf 文章标签： ctf 攻防世界 reverse base64
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/weixin_44604541/article/details/112428328
版权

CTF ctf 专栏收录该内容
200 篇文章 23 订阅
订阅专栏

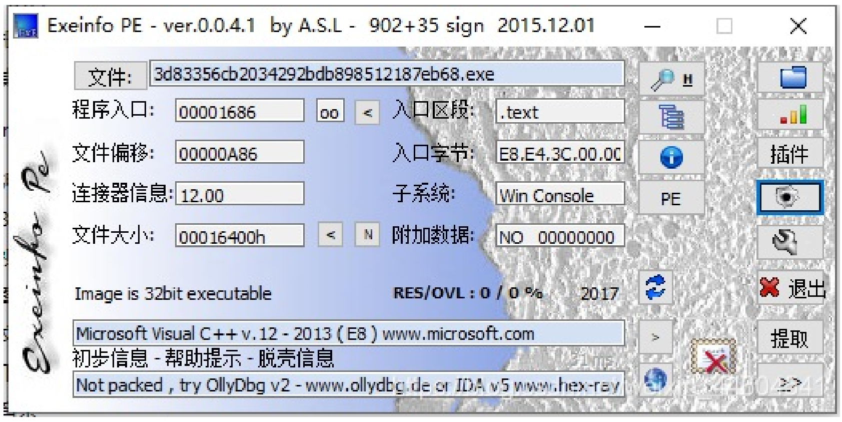
前言

继续ctf的旅程
攻防世界Reverse高手进阶区的2分题
本篇是ReverseMe-120的writeup

发现攻防世界的题目分数是动态的
就仅以做题时的分数为准了

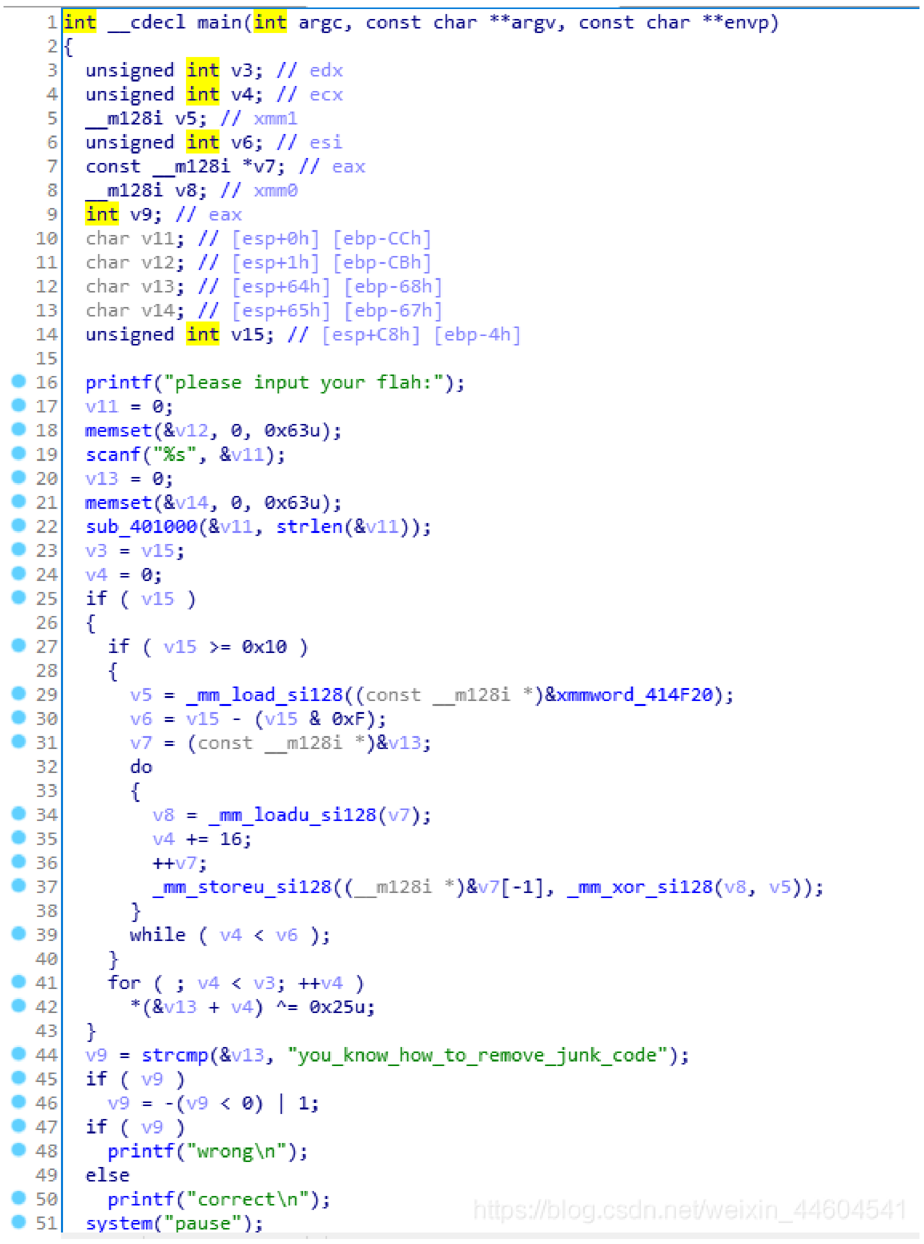
解题过程

PE查壳



```
Exeinfo PE - ver.0.0.4.1  by A.S.L -  902+35 sign  2015.12.01
文件:  3d83356cb2034292bdb898512187eb68.exe
程序入口: 00001686      入口区段: .text
文件偏移: 00000A86      入口字节: E8.E4.3C.00.00
连接器信息: 12.00       子系统: Win Console
文件大小: 00016400h     附加数据: NO  00000000
Image is 32bit executable    RES/OVL : 0 / 0 %   2017
Microsoft Visual C++ v.12 - 2013 ( E8 ) www.microsoft.com
初步信息 - 帮助提示 - 脱壳信息
Not packed , try OllyDbg v2 - www.ollydbg.de or IDA v5 www.hex-ray
```

扔进IDA



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  unsigned int v3; // edx
  unsigned int v4; // ecx
  __m128i v5; // xmm1
  unsigned int v6; // esi
  const __m128i *v7; // eax
  __m128i v8; // xmm0
  int v9; // eax
  char v11; // [esp+0h] [ebp-CCh]
  char v12; // [esp+1h] [ebp-CBh]
  char v13; // [esp+64h] [ebp-68h]
  char v14; // [esp+65h] [ebp-67h]
  unsigned int v15; // [esp+C8h] [ebp-4h]

  printf("please input your flah:");
  v11 = 0;
  memset(&v12, 0, 0x63u);
  scanf("%s", &v11);
  v13 = 0;
  memset(&v14, 0, 0x63u);
  sub_401000(&v11, strlen(&v11));
  v3 = v15;
  v4 = 0;
  if ( v15 )
  {
    if ( v15 >= 0x10 )
    {
      v5 = _mm_load_si128((const __m128i *)&xmmword_414F20);
      v6 = v15 - (v15 & 0xF);
      v7 = (const __m128i *)&v13;
      do
      {
        v8 = _mm_loadu_si128(v7);
        v4 += 16;
        ++v7;
        _mm_storeu_si128((__m128i *)&v7[-1], _mm_xor_si128(v8, v5));
      }
      while ( v4 < v6 );
    }
    for ( ; v4 < v3; ++v4 )
      *(&v13 + v4) ^= 0x25u;
  }
  v9 = strcmp(&v13, "you_know_how_to_remove_junk_code");
  if ( v9 )
    v9 = -(v9 < 0) | 1;
  if ( v9 )
    printf("wrong\n");
  else
    printf("correct\n");
  system("pause");
```

https://blog.csdn.net/weixin_44604541

就是一个输入转换再比对

其中一些函数查了查是

```
__m128i _mm_load_si128 (__m128i *p);
//返回可以存放在代表寄存器的变量中的值，即*p的值

__m128i _mm_load_si128 (__m128i *p);
//返回可以存放在代表寄存器的变量中的值，即*p的值

void _mm_storeu_si128 ( __m128i *p, __m128i a);
//将__m128i 变量a的值存储到p所指定的变量中去;
```

关键函数sub_401000

```
signed int __usercall sub_401000@<eax>(unsigned int *a1@<edx>, _BYTE *a2@<ecx>, unsigned __int8 *a3, unsigned in
t a4)
{
  int v4; // ebx
  unsigned int v5; // eax
  int v6; // ecx
  unsigned __int8 *v7; // edi
```

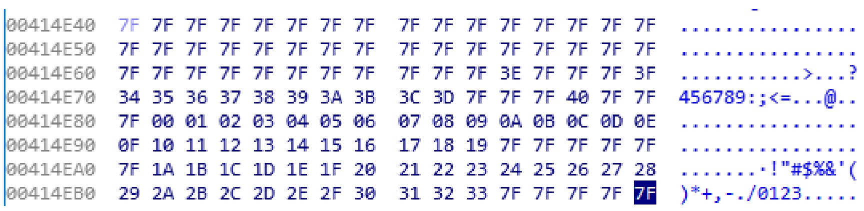
```
int v8; // edx
bool v9; // zf
unsigned __int8 v10; // cl
char v11; // cl
_BYTE *v12; // esi
unsigned int v13; // ecx
int v14; // ebx
unsigned __int8 v15; // cl
char v16; // dl
_BYTE *v18; // [esp+Ch] [ebp-Ch]
unsigned int *v19; // [esp+10h] [ebp-8h]
int v20; // [esp+14h] [ebp-4h]
unsigned int v21; // [esp+14h] [ebp-4h]
int i; // [esp+24h] [ebp+Ch]

v4 = 0;
v18 = a2;
v5 = 0;
v6 = 0;
v19 = a1;
v20 = 0;
if ( !a4 )
  return 0;
v7 = a3;
do
{
  v8 = 0;
  v9 = v5 == a4;
  if ( v5 < a4 )
  {
    do
    {
      if ( a3[v5] != 32 )
        break;
      ++v5;
      ++v8;
    }
    while ( v5 < a4 );
    v9 = v5 == a4;
  }
  if ( v9 )
    break;
  if ( a4 - v5 >= 2 && a3[v5] == 13 && a3[v5 + 1] == 10 || (v10 = a3[v5], v10 == 10) )
  {
    v6 = v20;
  }
  else
  {
    if ( v8 )
      return -44;
    if ( v10 == 61 && (unsigned int)++v4 > 2 )
      return -44;
    if ( v10 > 0x7Fu )
      return -44;
    v11 = byte_414E40[v10];
    if ( v11 == 127 || (unsigned __int8)v11 < 0x40u && v4 )
      return -44;
    v6 = v20++ + 1;
  }
  ++v5;
```

```
    ++v5;
  }
  while ( v5 < a4 );
  if ( !v6 )
    return 0;
  v12 = v18;
  v13 = ((unsigned int)(6 * v6 + 7) >> 3) - v4;
  if ( v18 && *v19 >= v13 )
  {
    v21 = 3;
    v14 = 0;
    for ( i = 0; v5; --v5 )
    {
      v15 = *v7;
      if ( *v7 != 13 && v15 != 10 && v15 != 32 )
      {
        v16 = byte_414E40[v15];
        v21 -= v16 == 64;
        v14 = v16 & 0x3F | (v14 << 6);
        if ( ++i == 4 )
        {
          i = 0;
          if ( v21 )
            *v12++ = BYTE2(v14);
          if ( v21 > 1 )
            *v12++ = BYTE1(v14);
          if ( v21 > 2 )
            *v12++ = v14;
        }
      }
      ++v7;
    }
    *v19 = v12 - v18;
    return 0;
  }
  *v19 = v13;
  return -42;
}
```

看得人有点麻

看了眼byte_414E40

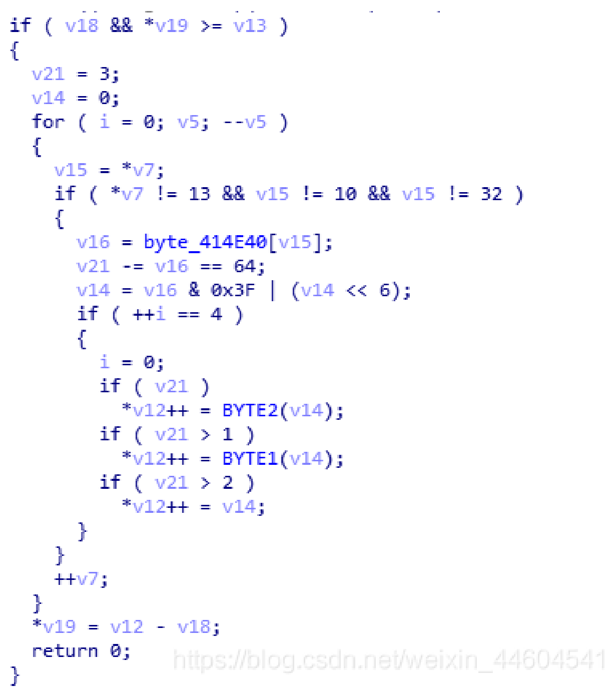


```
00414E40  7F 7F 7F 7F 7F 7F 7F 7F  7F 7F 7F 7F 7F 7F 7F 7F  ................
00414E50  7F 7F 7F 7F 7F 7F 7F 7F  7F 7F 7F 7F 7F 7F 7F 7F  ................
00414E60  7F 7F 7F 7F 7F 7F 7F 7F  7F 7F 7F 3E 7F 7F 7F 3F  ...........>...?
00414E70  34 35 36 37 38 39 3A 3B  3C 3D 7F 7F 7F 40 7F 7F  456789:;<=...@..
00414E80  7F 00 01 02 03 04 05 06  07 08 09 0A 0B 0C 0D 0E  ................
00414E90  0F 10 11 12 13 14 15 16  17 18 19 7F 7F 7F 7F 7F  ................
00414EA0  7F 1A 1B 1C 1D 1E 1F 20  21 22 23 24 25 26 27 28  ....... !"#$%&'(
00414EB0  29 2A 2B 2C 2D 2E 2F 30  31 32 33 7F 7F 7F 7F 7F  )*+,-./0123.....
```

还是没什么想法

查了查
这个函数就是base64

关键代码



```
if ( v18 && *v19 >= v13 )
{
  v21 = 3;
  v14 = 0;
  for ( i = 0; v5; --v5 )
  {
    v15 = *v7;
    if ( *v7 != 13 && v15 != 10 && v15 != 32 )
    {
      v16 = byte_414E40[v15];
      v21 -= v16 == 64;
      v14 = v16 & 0x3F | (v14 << 6);
      if ( ++i == 4 )
      {
        i = 0;
        if ( v21 )
          *v12++ = BYTE2(v14);
        if ( v21 > 1 )
          *v12++ = BYTE1(v14);
        if ( v21 > 2 )
          *v12++ = v14;
      }
    }
    ++v7;
  }
  *v19 = v12 - v18;
  return 0;
}
```

https://blog.csdn.net/weixin_44604541

每三个字节处理变成四个字节，生成一个长字节
再从这个长字节中查四次表生成对应的四个字符
这就是base64的特征

那就简单了
先xor再base64解密

```python
import base64

s='you_know_how_to_remove_junk_code'
flag=''
for i in s:
    flag += chr(ord(i)^0x25)
print(base64.b64encode(flag))
```

得到flag：`XEpQek5LSlJ6TUpSelFKeldASEpTQHpPUEtOekZKQUA`

结语

关键是base64的识别
看到wp里有自己分析源码的
看着就很痛苦