[ACTF新生赛2020]fungame

原创

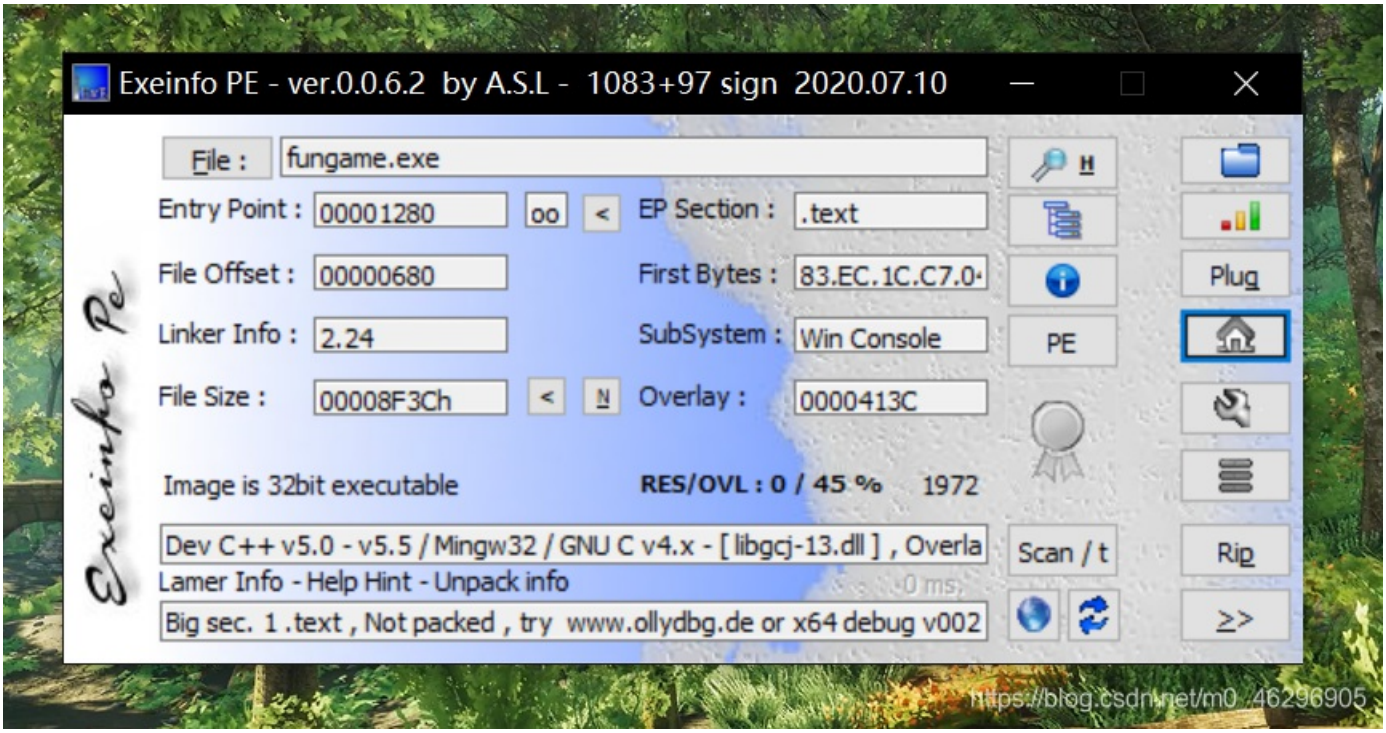
Em0s_Er1t 于 2021-05-05 23:55:14 发布 313 收藏

文章标签： python 指针 信息安全 c++ 栈

本文链接：https://blog.csdn.net/m0_46296905/article/details/116431938

题目：[ACTF新生赛2020]fungame

32位。



主要是两个函数。

```
int __cdecl sub_401340(int a1)
{
  char i; // [esp+1Fh] [ebp-9h]

  printf("Please input:");
  scanf("%s", a1);
  for ( i = 0; i <= 15; ++i )
  {
    if ( (*(i + a1) ^ *(y1 + i)) != y2[i] )
      exit(0);
  }
  return 0;
}
```

很简单的异或可以得到前16个字符

```
int __cdecl sub_401340(int a1)
{
  char i; // [esp+1Fh] [ebp-9h]

  printf("Please input:");
  scanf("%s", a1);
  for ( i = 0; i <= 15; ++i )
  {
    if ( (*(i + a1) ^ *(y1 + i)) != y2[i] )
      exit(0);
  }
  return 0;
}
```

一开始没留意后面的字符串复制直接写了exp提交，发现不对，

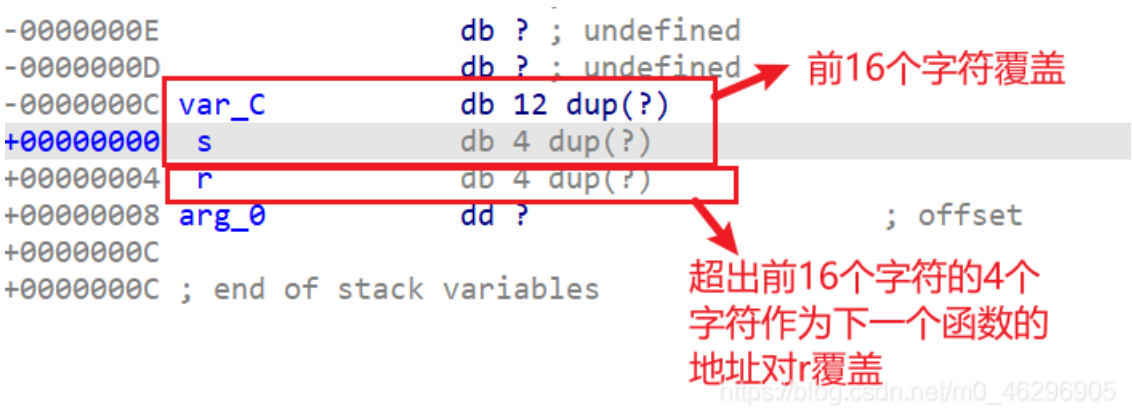
仔细观察发现是栈溢出，这个函数先把flag复制到了只能容纳12个字节的Destination，必定发生溢出，而第二次则复制到了x

```
int __cdecl sub_4013BA(char *Source)
{
  char Destination[12]; // [esp+1Ch] [ebp-Ch] BYREF

  strcpy(Destination, Source);
  strcpy(x, Source);
  return 0;
}
```

交叉调用看一下x，发现了另一个函数，实现的是base64编码。

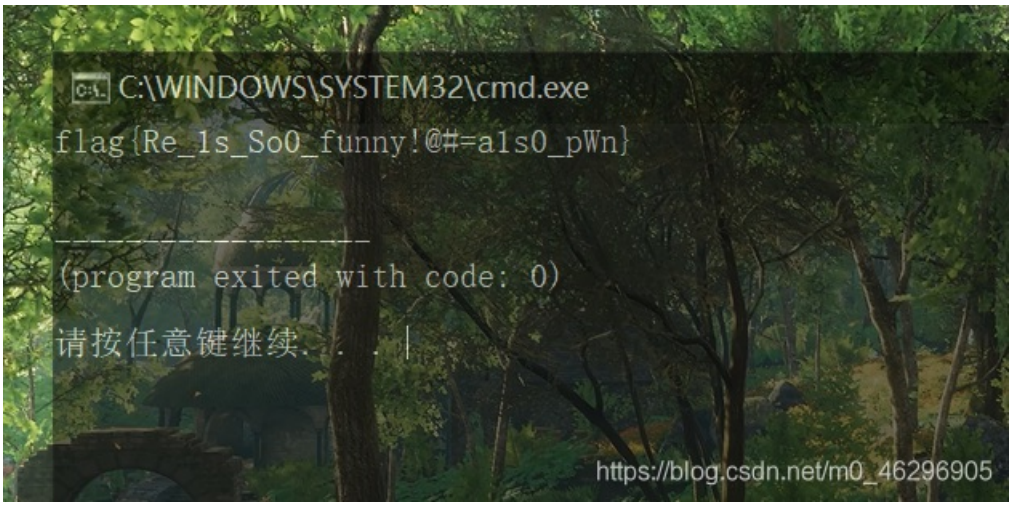
```
void __noreturn sub_40233D()
{
  char Str2[13]; // [esp+13h] [ebp-35h] BYREF
  char Str1[16]; // [esp+20h] [ebp-28h] BYREF
  char Str[12]; // [esp+30h] [ebp-18h] BYREF
  size_t v3; // [esp+3Ch] [ebp-Ch]

  printf("Please input again:");
  strcpy(Str2, "YTFzMF9wV24=");
  memset(Str, 0, sizeof(Str));
  memset(Str1, 0, sizeof(Str1));
  scanf("%s", Str);
  v3 = strlen(Str);
  sub_402421(Str, v3, Str1);//base64
  if ( !strcmp(Str1, Str2) )
  {
    printf("%s%s", x, Str);
    exit(0);
  }
  exit(0);
}
```

原来是通过输入的flag产生溢出，然后用flag中的隐藏函数的地址（地址为40233D）覆盖，促使程序调用隐藏函数sub_40233D。



EXP

```
import base64
y1=[0x23,0x61,0x3E,0x69,0x54,0x41,0x18,0x4D,0x6E,0x3B,0x65,0x53,0x30,0x79,0x45,0x5B]
y2=[0x71,0x04,0x61,0x58,0x27,0x1E,0x4B,0x22,0x5E,0x64,0x3,0x26,0x5E,0x17,0x3C,0x7A]
flag='flag{'
for i in range(len(y1)):
 flag+=chr(y1[i]^y2[i])
flag+=chr(0x40)+chr(0x23)+chr(0x3D)+str(base64.b64decode('YTFzMF9wV24='),encoding = "utf-8") #bytes转str
print(flag+'}')
```




```
flag{Re_1s_So0_funny!@#=a1s0_pWn}
```