2019新生杯 re writeup

原创

_n19hT 于 2019-11-18 23:19:41 发布 236 收藏 1

分类专栏： # reverse

本文链接：https://blog.csdn.net/weixin_43092232/article/details/103133907

reverse 专栏收录该内容

5 篇文章 0 订阅

订阅专栏

0x00 esayRE

~~本来re第一题叫easyRE很常见…结果你放了道esayRE…~~

~~没事，先运行，发现没什么东西，准备放OD跑，结果运行不了???这第一题就加壳???~~

emmm,后面发现丢进IDA，万能f5。

```
_main();
strcpy((char *)flag, "ZmxhZ3tSM19XMXRoX2I0c2U2NF8xc19zMW1wbGVfMG4zfQ==");
memset(cipher, 0, 0x30ui64);
```

得到一串字符：*ZmxhZ3tSM19XMXRoX2I0c2U2NF8xc19zMW1wbGVfMG4zfQ==*

一看，base64加密的，在线base64解密之后：

```
flag{R3_W1th_b4se64_1s_s1mple_0n3}
```

0x01 easyELF

我拿到这题先是IDA看一下逻辑，感觉不难，写个脚本就应该能跑。

(由于不是windows下的可执行文件，OD之类的都跑不了)

IDA里面主要函数：

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  unsigned int v3; // ST08_4
  __int64 v4; // rdx
  unsigned int v5; // ST0C_4
  unsigned int v6; // ST08_4
  int v7; // ST0C_4
  __int64 v9; // [rsp+10h] [rbp-40h]
  __int64 v10; // [rsp+18h] [rbp-38h]
  __int64 v11; // [rsp+20h] [rbp-30h]
  __int64 v12; // [rsp+28h] [rbp-28h]
  __int64 v13; // [rsp+30h] [rbp-20h]
  int v14; // [rsp+38h] [rbp-18h]
  __int16 v15; // [rsp+3Ch] [rbp-14h]
  char v16; // [rsp+3Eh] [rbp-12h]
  unsigned __int64 v17; // [rsp+48h] [rbp-8h]

  v17 = __readfsqword(0x28u);
  v9 = 8388319874088921696LL;
  v10 = 8176671504997244777LL;
  v11 = 7448788179923060319LL;
  v12 = 7594880320322233974LL;
  v13 = 72354192043852020 30LL;
  v14 = 1768318839;
  v15 = 32110;
  v16 = 0;
  v3 = sub_4005D6(19LL, 0LL, a3);
  v5 = sub_4005D6(9LL, 0LL, v4);
  v6 = sub_400612(v3, 19LL);
  v7 = sub_400612(v5, 9LL);
  sub_4006CB((const char *)&v9, v6, v7);
  puts("Fighting! And..... Goodbye!");
  return 0LL;
}
```

```
unsigned __int64 __fastcall sub_4006CB(const char *a1, unsigned int a2, int a3)
{
  int v4; // [rsp+0h] [rbp-90h]
  int i; // [rsp+18h] [rbp-78h]
  int v6; // [rsp+1Ch] [rbp-74h]
  char v7[104]; // [rsp+20h] [rbp-70h]
  unsigned __int64 v8; // [rsp+88h] [rbp-8h]

  v4 = a3;
  v8 = __readfsqword(0x28u);
  v6 = strlen(a1);
  for ( i = 0; i < v6; ++i )
  {
    if ( a1[i] != 123 && a1[i] != 125 && a1[i] != 95 )
    {
      a1[i] ^= 1u;
      if ( a1[i] <= 64 || a1[i] > 90 )
      {
        v7[i] = (signed int)((a1[i] - 97 - v4) * (unsigned __int64)sub_40068A(a2)) % 26 + 97;
        if ( ((unsigned __int64)sub_40068A(a2) * (a1[i] - 97 - v4) & 0x80000000) != 0LL )
          v7[i] += 26;
      }
      else
      {
        v7[i] = (signed int)((a1[i] - 65 - v4) * (unsigned __int64)sub_40068A(a2)) % 26 + 65;
        if ( ((unsigned __int64)sub_40068A(a2) * (a1[i] - 65 - v4) & 0x80000000) != 0LL )
          v7[i] += 26;
      }
    }
    else
    {
      v7[i] = a1[i];
    }
  }
  return __readfsqword(0x28u) ^ v8;
}
```

v3 v5 v6 v7是把参数带进函数算出来的,然后再把几个值带到sub_4006CB里面跑。
这个题卡了我好几个小时，一开始没搞懂给了v9-v15，为什么就只传进去一个v5的参数值，跑出来是"flag{Hero"说明缺了一堆字符…(我以为是定义的数据类型不一样导致的)后面突然想把v10-v15传进去试试,最后跑出了flag。害 还是题目做的太少！
贴一个自己写的c：

```c
#include<stdio.h>
#include<string.h>
unsigned __int64 sub_6CB(char *a1,int a2,int a3)
{
 int v4,i;
 int v6;
 char v7[104];
 v4 = a3;
 v6 = strlen(a1);
 //printf("%d",v6);
 for(i=0;i<v6;++i)
 {
  if(a1[i]!=123&&a1[i]!=125&&a1[i]!=95)
  {
   a1[i]^=1u;
   if((a1[i]<=64)||(a1[i]>90))
   {
    v7[i]=(a1[i]-97-v4)*11%26+97;
    if((11*(a1[i]-97-v4)&0x80000000)!=0LL)
    v7[i] += 26;
   }
   else
   {
    v7[i]=(a1[i]-65-v4)*11%26+65;
     if((11*(a1[i]-65-v4)&0x80000000)!=0LL)
     v7[i]+=26;

   }

  }
  else
   v7[i]=a1[i];
 }
 printf("%s",v7);

}

int main()
{
 //__int64 v8=(signed __int64)(8388319874088921696);flag{Hero
 //__int64 v8=(signed __int64)(8176671504997244777);e_are_yoo
 //__int64 v8=(signed __int64)(7448788179923060319);_To_be_io
 //__int64 v8=(signed __int64)(7594880320322233974);nteresteo
 //__int64 v8=(signed __int64)(7235419204385202030);d_and_exo
 //__int64 v8=(signed __int64)(1768318839);cite
 //__int64 v8=(signed __int64)(32110);d}@
 //__int64 v8=(signed __int64)(0);
 //flag{Here_are_yo_To_be_interested_and_excited}
 unsigned int v5 =19;
 int v6 =9;

 sub_6CB((char *)&v8,v5,v6);
// printf("%d",strlen((char*)&v8));
 return 0;
 }
```

自己写c时要注意一下,传进去的a1类型定义为char，IDA里面伪函数定义的是 const chara1,const char 定义之后，指针对象只读，不能进行赋值计算(自己试试就知道了，这个坑也坑了我好长时间)。

但后面看大佬wp的时候，发现人家这题直接IDA下断点动态调试就出来了…
我试了下return处下断点，IDA联合ubuntu调试，然后查看v7的值(看逻辑知道v7存的就是flag),就查到了flag…

```
[stack]:00007FFE4B9452D0 db  66h ; f
[stack]:00007FFE4B9452D1 db  6Ch ; l
[stack]:00007FFE4B9452D2 db  61h ; a
[stack]:00007FFE4B9452D3 db  67h ; g
[stack]:00007FFE4B9452D4 db  7Bh ; {
[stack]:00007FFE4B9452D5 db  48h ; H
[stack]:00007FFE4B9452D6 db  65h ; e
[stack]:00007FFE4B9452D7 db  72h ; r
[stack]:00007FFE4B9452D8 db  65h ; e
[stack]:00007FFE4B9452D9 db  5Fh ; _
[stack]:00007FFE4B9452DA db  61h ; a
[stack]:00007FFE4B9452DB db  72h ; r
[stack]:00007FFE4B9452DC db  65h ; e
[stack]:00007FFE4B9452DD db  5Fh ; _
[stack]:00007FFE4B9452DE db  79h ; y
[stack]:00007FFE4B9452DF db  6Fh ; o
[stack]:00007FFE4B9452E0 db  5Fh ; _
[stack]:00007FFE4B9452E1 db  54h ; T
[stack]:00007FFE4B9452E2 db  6Fh ; o
[stack]:00007FFE4B9452E3 db  5Fh ; _
[stack]:00007FFE4B9452E4 db  62h ; b
```

0x02 ReverseMe

这题比赛的时候没写出来，逻辑都没怎么看懂，就知道是类似base64加密的过程。
那我得写个类似base64解密的逆过程吧…然后就开始头秃，当时我还奇怪怎么这题挺多人做出来了。
原来IDA里面有table表,对应着base64的table表

```
80 ; char *s2
80 s2              dq offset aWxpXdkldnfdfnf
80                                     ; DATA XREF: main+4E↑r
80                                     ; "+wXp+xDkldnFdFNFdxnzdFWpGx2m"
88                 align 20h
A0 ; char s
A0 s               db 52h              ; DATA XREF: sub_40089E+E8↑r
A0                                     ; sub_40089E+139↑r ...
A1 a9ly6nojvsipnwh db '9Ly6NoJvsIPnWhETYtHe4Sdl+MbGujaZpk102wKCr7/ODg5zXAFqQfxBicV3m8U',0
A1 _data           ends
A1
1 ; ===========================================================================
1
```

需要解密的字符

对应的table表

python脚本:

```
import base64
a = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
b = 'R9Ly6NoJvsIPnWhETYtHe4Sdl+MbGujaZpk102wKCr7/ODg5zXAFqQfxBicV3m8U'
str1='+wXp+xDkldnFdFNFdxnzdFWpGx2m'
flag = ''
for i in str1:
    flag += a[b.index(i)]
print(base64.b64decode(flag))
```

flag{bas3_1s_s0_3asy}
附百度网盘链接：Click Here