2018湖湘杯逆向（Reverse）

原创

Thunder_J 于 2019-02-04 10:57:48 发布 998 收藏 7

分类专栏： 题目篇 文章标签： 逆向

0x00：介绍

HighwayHash64这道题放了很久一直没有去做，最近看了一下发现这道题比较特殊，重新整理了一下思路，Replace这道题比较简单，属于签到题难度。

0x01：题目

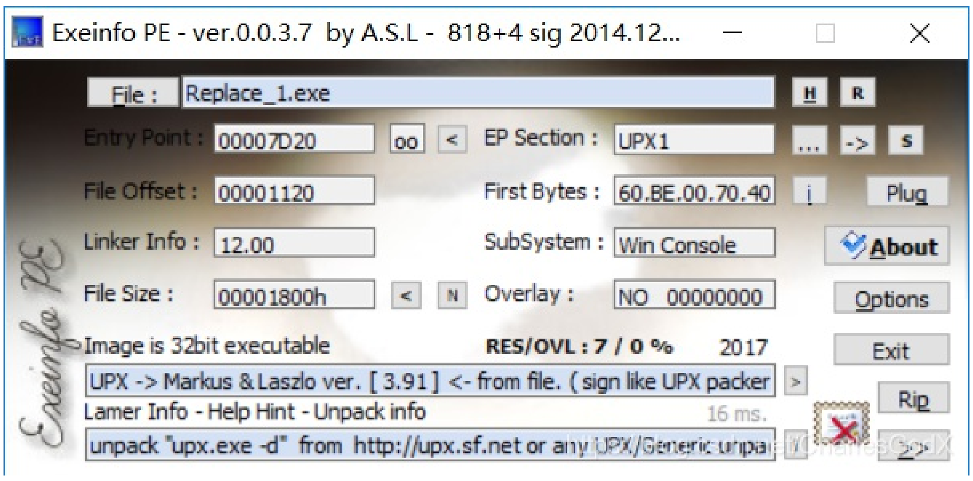
题目1：Replace

题目链接：

> 简单的…密码学~
> http://hxb2018.oss-cn-beijing.aliyuncs.com/reserves/Replace_B21DA8B2F172C13764989DF0F99B890A.rar

解题过程：

首先查壳是UPX壳，我们直接用脱壳机就可以了（当然也可以手动）


Exeinfo PE - ver.0.0.3.7 by A.S.L - 818+4 sig 2014.12...

File : Replace_1.exe

Entry Point : 00007D20

EP Section : UPX1

File Offset : 00001120

First Bytes : 60.BE.00.70.40

Linker Info : 12.00

SubSystem : Win Console

File Size : 00001800h

Overlay : NO 00000000

Image is 32bit executable

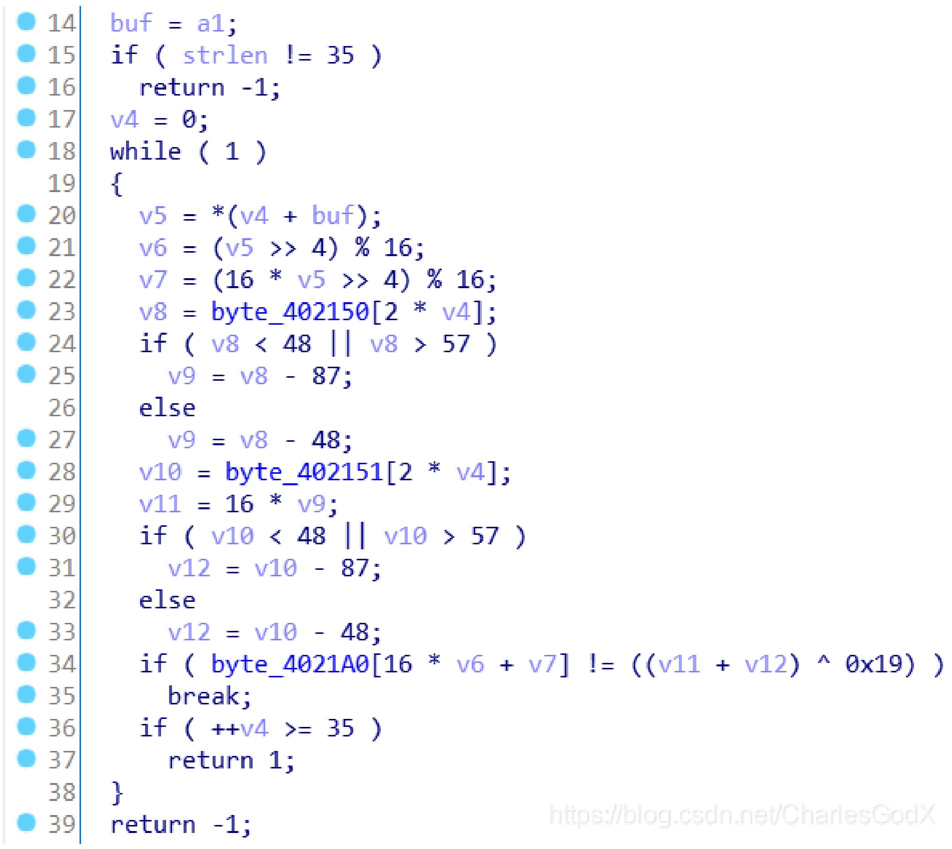
RES/OVL : 7 / 0 % 2017

UPX -> Markus & Laszlo ver. [3.91] <- from file. (sign like UPX packer

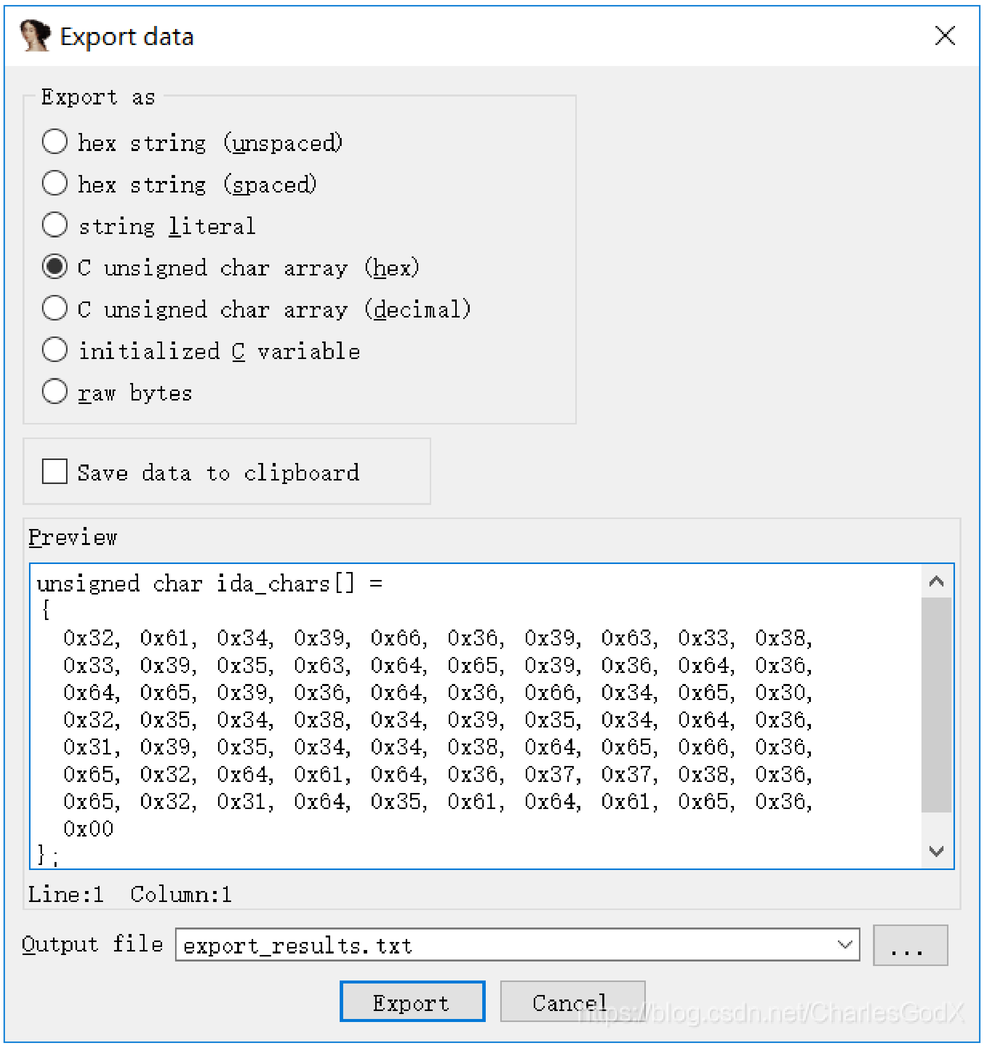
Lamer Info - Help Hint - Unpack info 16 ms.

unpack "upx.exe -d" from http://upx.sf.net or any UPX/Generic unpa

我们用IDA分析后发现，就是一个简单的算法

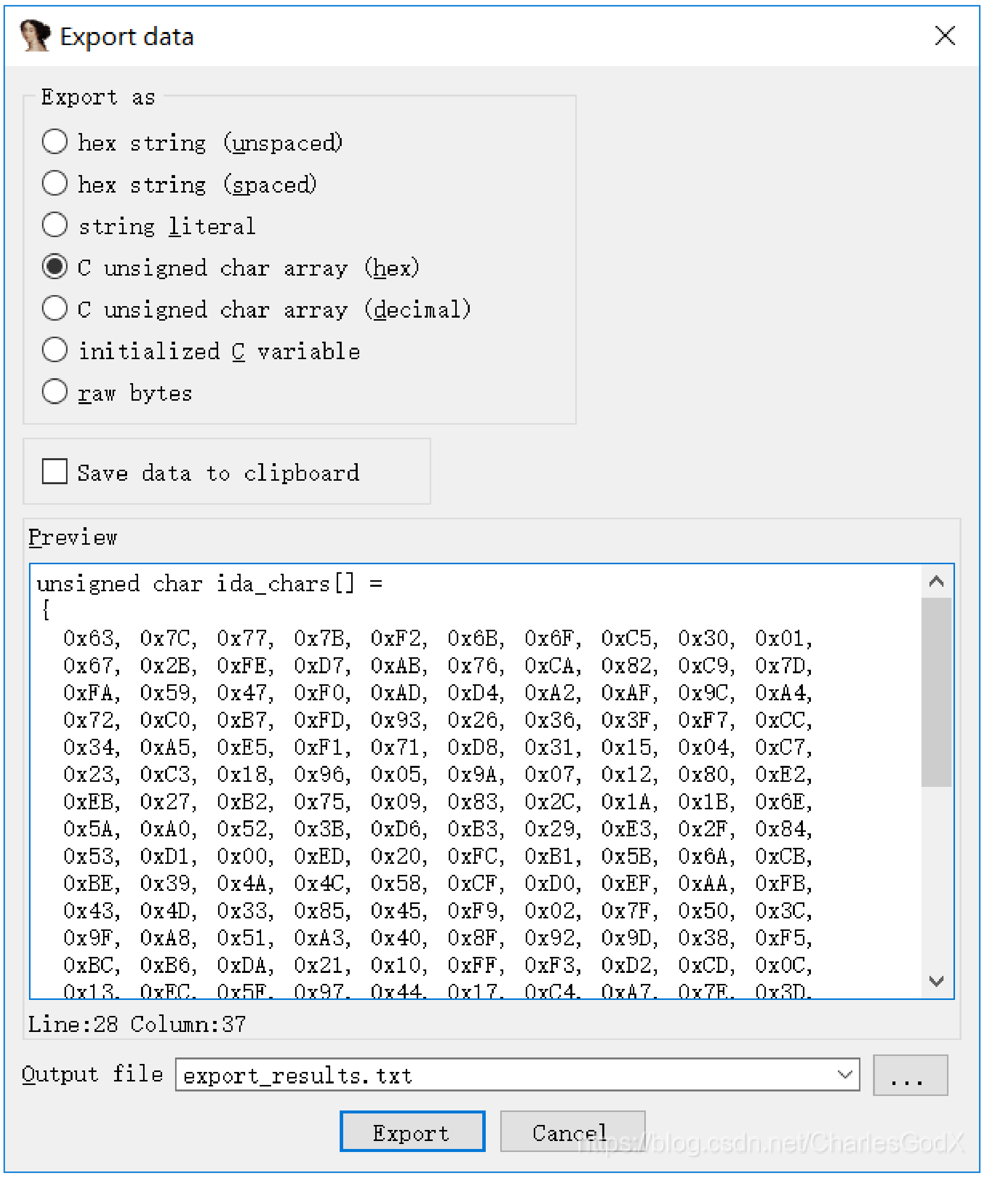

```
buf = a1;
if ( strlen != 35 )
  return -1;
v4 = 0;
while ( 1 )
{
  v5 = *(v4 + buf);
  v6 = (v5 >> 4) % 16;
  v7 = (16 * v5 >> 4) % 16;
  v8 = byte_402150[2 * v4];
  if ( v8 < 48 || v8 > 57 )
    v9 = v8 - 87;
  else
    v9 = v8 - 48;
  v10 = byte_402151[2 * v4];
  v11 = 16 * v9;
  if ( v10 < 48 || v10 > 57 )
    v12 = v10 - 87;
  else
    v12 = v10 - 48;
  if ( byte_4021A0[16 * v6 + v7] != ((v11 + v12) ^ 0x19) )
    break;
  if ( ++v4 >= 35 )
    return 1;
}
return -1;
```

我们提取出第一处关键数组，这里可以不包括最后一个0x00：


Export data
Export as
( ) hex string (unspaced)
( ) hex string (spaced)
( ) string literal
(●) C unsigned char array (hex)
( ) C unsigned char array (decimal)
( ) initialized C variable
( ) raw bytes
☐ Save data to clipboard
Preview
Line:1 Column:1
Output file export_results.txt
...
Export
Cancel

```
unsigned char ida_chars[] =
{
  0x32, 0x61, 0x34, 0x39, 0x66, 0x36, 0x39, 0x63, 0x33, 0x38,
  0x33, 0x39, 0x35, 0x63, 0x64, 0x65, 0x39, 0x36, 0x64, 0x36,
  0x64, 0x65, 0x39, 0x36, 0x64, 0x36, 0x66, 0x34, 0x65, 0x30,
  0x32, 0x35, 0x34, 0x38, 0x34, 0x39, 0x35, 0x34, 0x64, 0x36,
  0x31, 0x39, 0x35, 0x34, 0x34, 0x38, 0x64, 0x65, 0x66, 0x36,
  0x65, 0x32, 0x64, 0x61, 0x64, 0x36, 0x37, 0x37, 0x38, 0x36,
  0x65, 0x32, 0x31, 0x64, 0x35, 0x61, 0x64, 0x61, 0x65, 0x36,
  0x00
};
```

第二处关键数组：



Export as: [✓ C unsigned char array (hex)]

```
unsigned char ida_chars[] =
{
  0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01,
  0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76, 0xCA, 0x82, 0xC9, 0x7D,
  0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4,
  0x72, 0xC0, 0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC,
  0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15, 0x04, 0xC7,
  0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2,
  0xEB, 0x27, 0xB2, 0x75, 0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E,
  0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84,
  0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB,
  0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF, 0xD0, 0xEF, 0xAA, 0xFB,
  0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C,
  0x9F, 0xA8, 0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5,
  0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2, 0xCD, 0x0C,
  0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D,
```

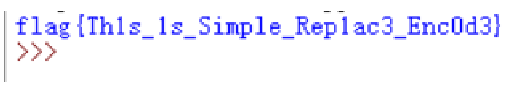
Line:28 Column:37

Output file: export_results.txt

提取完之后，就可以写爆破脚本了：

```
a = [
  0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01,
  0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76, 0xCA, 0x82, 0xC9, 0x7D,
  0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4,
  0x72, 0xC0, 0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC,
  0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15, 0x04, 0xC7,
  0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2,
  0xEB, 0x27, 0xB2, 0x75, 0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E,
  0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84,
  0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB,
  0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF, 0xD0, 0xEF, 0xAA, 0xFB,
  0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C,
  0x9F, 0xA8, 0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5,
  0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2, 0xCD, 0x0C,
  0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D,
  0x64, 0x5D, 0x19, 0x73, 0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A,
  0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB,
  0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3,
  0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79, 0xE7, 0xC8, 0x37, 0x6D,
  0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A,
  0xAE, 0x08, 0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6,
  0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A, 0x70, 0x3E,
  0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9,
  0x86, 0xC1, 0x1D, 0x9E, 0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9,
  0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,
  0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99,
  0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16
]
b = [
  0x32, 0x61, 0x34, 0x39, 0x66, 0x36, 0x39, 0x63, 0x33, 0x38,
  0x33, 0x39, 0x35, 0x63, 0x64, 0x65, 0x39, 0x36, 0x64, 0x36,
  0x64, 0x65, 0x39, 0x36, 0x64, 0x36, 0x66, 0x34, 0x65, 0x30,
  0x32, 0x35, 0x34, 0x38, 0x34, 0x39, 0x35, 0x34, 0x64, 0x36,
  0x31, 0x39, 0x35, 0x34, 0x34, 0x38, 0x64, 0x65, 0x66, 0x36,
  0x65, 0x32, 0x64, 0x61, 0x64, 0x36, 0x37, 0x37, 0x38, 0x36,
  0x65, 0x32, 0x31, 0x64, 0x35, 0x61, 0x64, 0x61, 0x65, 0x36,
  0x00
]

flag = ""

v4 = 0
while(1):
    v8 = b[2 * v4]
    if (v8 < 48 | v8 > 57):
        v9 = v8 - 87
    else:
        v9 = v8 - 48
    v10 = b[2 * v4+1]
    v11 = 16 * v9
    if ( v10 < 48 | v10 > 57 ):
        v12 = v10 - 87
    else:
        v12 = v10 - 48

    for v5 in range(0,127):
        v6 = (v5 >> 4) % 16
        v7 = (16 * v5 >> 4) % 16
        if a[16 * v6 + v7] == (v11 + v12) ^ 0x19:
            flag += chr(v5)
            break

    v4 += 1

    if (v4 >= 35):
        break
print(flag)
```

运行得到flag



```
flag{Th1s_1s_Simple_Replac3_Enc0d3}
>>>
```

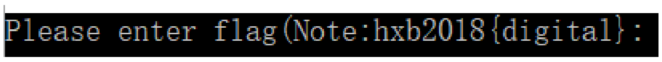
题目2：HighwayHash64

题目链接：

> 口算哈希说的就是你吧~
> http://hxb2018.oss-cn-beijing.aliyuncs.com/reserves/reverse_1CE475F54D2A3264A8ED743FDFEF24A8.zip

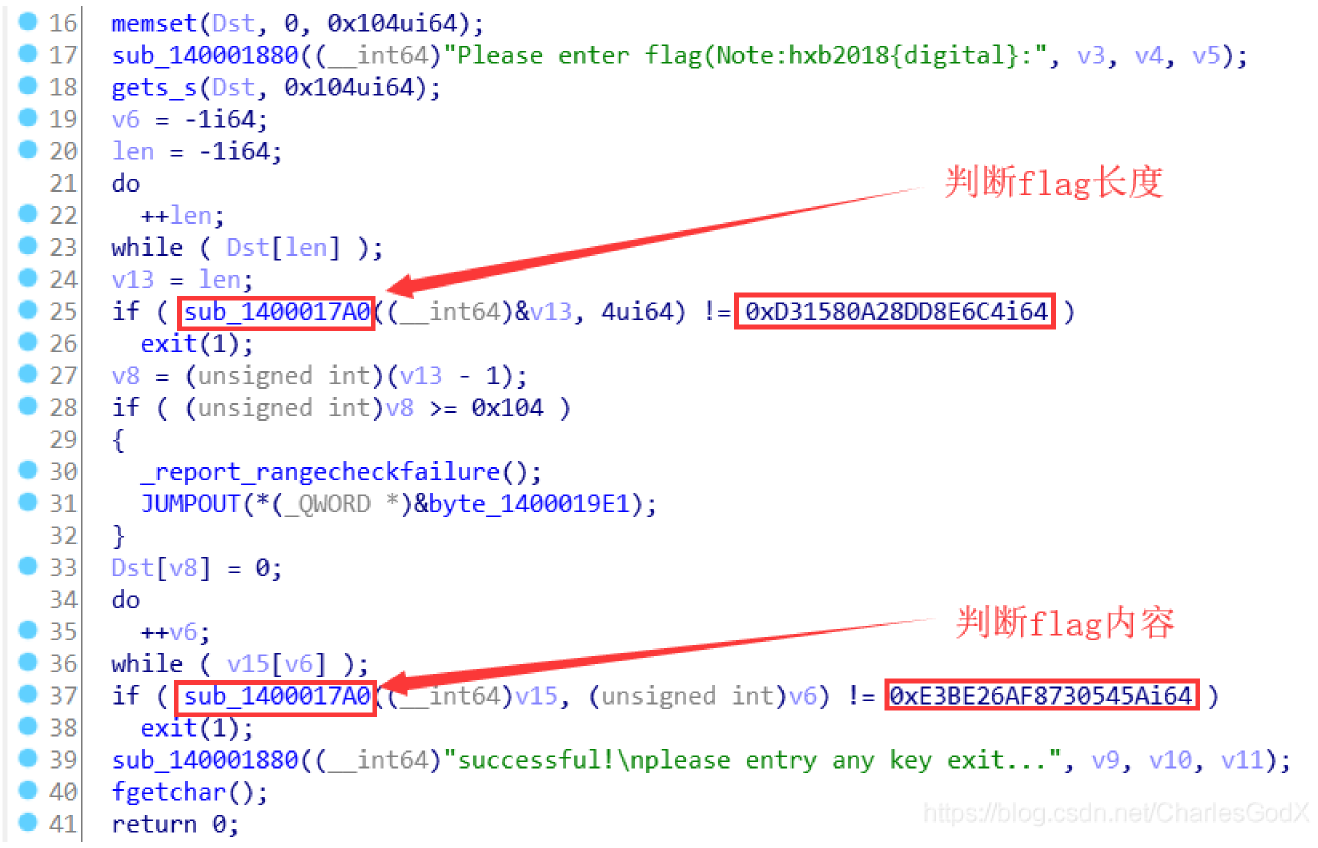
解题过程：

拿到题运行之，提示要我们输入flag，随便输入之后程序自动退出，放入IDA中分析一下



```
Please enter flag(Note:hxb2018{digital}:
```

找到main函数，这两处有明显的比较，结合题目意思，这里应该是hash摘要计算，我们需要爆破hash的值

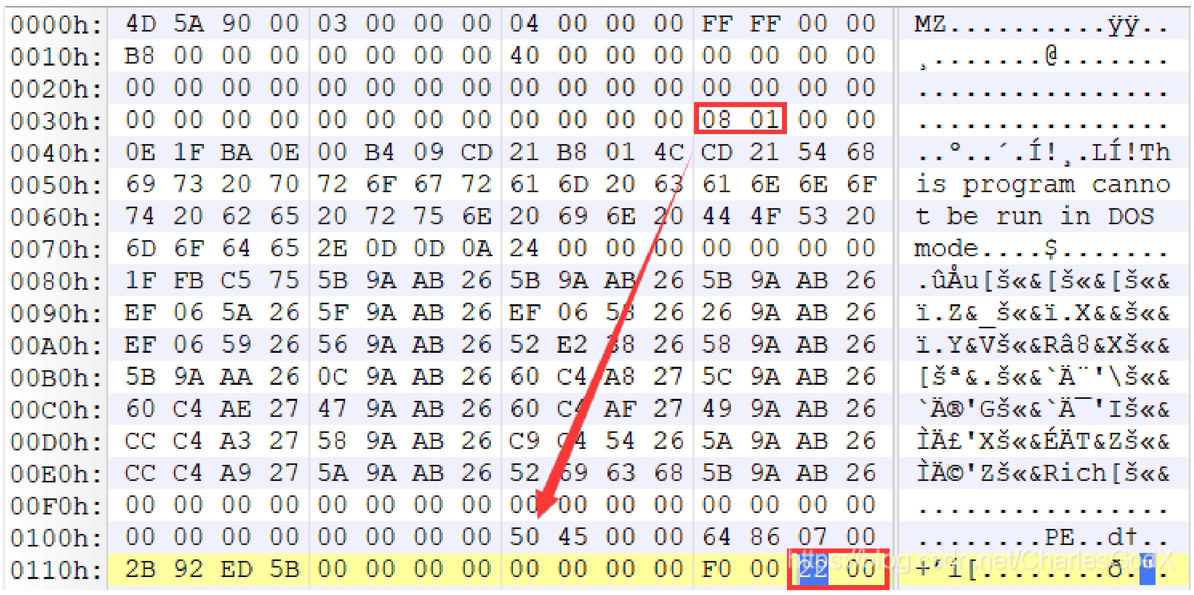


```
memset(Dst, 0, 0x104ui64);
sub_140001880((__int64)"Please enter flag(Note:hxb2018{digital}:", v3, v4, v5);
gets_s(Dst, 0x104ui64);
v6 = -1i64;
len = -1i64;
do
  ++len;
while ( Dst[len] );
v13 = len;
if ( sub_1400017A0((__int64)&v13, 4ui64) != 0xD31580A28DD8E6C4i64 )
  exit(1);
v8 = (unsigned int)(v13 - 1);
if ( (unsigned int)v8 >= 0x104 )
{
  _report_rangecheckfailure();
  JUMPOUT(*(_QWORD *)&byte_1400019E1);
}
Dst[v8] = 0;
do
  ++v6;
while ( v15[v6] );
if ( sub_1400017A0((__int64)v15, (unsigned int)v6) != 0xE3BE26AF8730545Ai64 )
  exit(1);
sub_140001880((__int64)"successful!\nplease entry any key exit...", v9, v10, v11);
fgetchar();
return 0;
```

我们逐个分析，可以分析得出第一个函数负责计算flag长度，第二个函数负责计算flag括号中的内容，既然是自定义的hash函数，我们可以考虑新写一个exe程序调用这个程序的函数，既然是调用那我们肯定要将exe文件修改为dll文件，工具使用010Editor，修改的方法主要是：

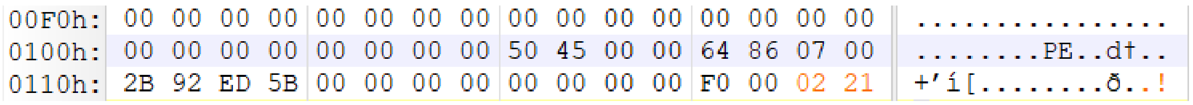
- IMAGE_FILE_HEADER->Characteristics(文件属性)->2102h（DLL文件一般是2102h）
- IMAGE_OPTIONAL_HEADER->AddressOfEntryPoint（程序执行入口RVA）->0000h

我们先通过MS-DOS头部找到PE文件头，然后在通过偏移找到上面的两处位置，如下图：



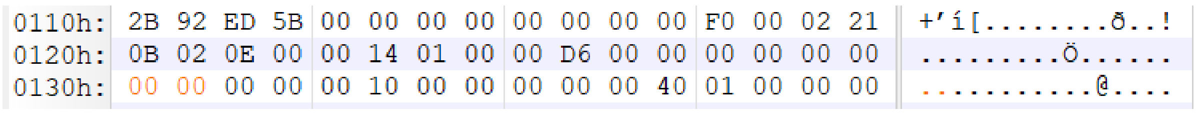
```
0000h: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
0010h: B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
0020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0030h: 00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00  ................
0040h: 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
0050h: 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
0060h: 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
0070h: 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
0080h: 1F FB C5 75 5B 9A AB 26 5B 9A AB 26 5B 9A AB 26  .ûÅu[š«&[š«&[š«&
0090h: EF 06 5A 26 5F 9A AB 26 EF 06 58 26 26 9A AB 26  ï.Z&_š«&ï.X&&š«&
00A0h: EF 06 59 26 56 9A AB 26 52 E2 38 26 58 9A AB 26  ï.Y&Vš«&Râ8&Xš«&
00B0h: 5B 9A AA 26 0C 9A AB 26 60 C4 A8 27 5C 9A AB 26  [šª&.š«&`Ä¨'\š«&
00C0h: 60 C4 AE 27 47 9A AB 26 60 C4 AF 27 49 9A AB 26  `Ä®'Gš«&`Ä¯'Iš«&
00D0h: CC C4 A3 27 58 9A AB 26 C9 C4 54 26 5A 9A AB 26  ÌÄ£'Xš«&ÉÄT&Zš«&
00E0h: CC C4 A9 27 5A 9A AB 26 52 69 63 68 5B 9A AB 26  ÌÄ©'Zš«&Rich[š«&
00F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0100h: 00 00 00 00 00 00 00 00 50 45 00 00 64 86 07 00  ........PE..d†..
0110h: 2B 92 ED 5B 00 00 00 00 00 00 00 00 F0 00 22 00  +'í[........ð.".
```

第一处修改：



```
00F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0100h: 00 00 00 00 00 00 00 00 50 45 00 00 64 86 07 00  ........PE..d†..
0110h: 2B 92 ED 5B 00 00 00 00 00 00 00 00 F0 00 02 21  +'í[........ð..!
```

第二处修改：



```
0110h: 2B 92 ED 5B 00 00 00 00 00 00 00 00 F0 00 02 21  +'í[........ð..!
0120h: 0B 02 0E 00 00 14 01 00 00 D6 00 00 00 00 00 00  .........Ö......
0130h: 00 00 00 00 00 10 00 00 00 00 00 40 01 00 00 00  ...........@....
```

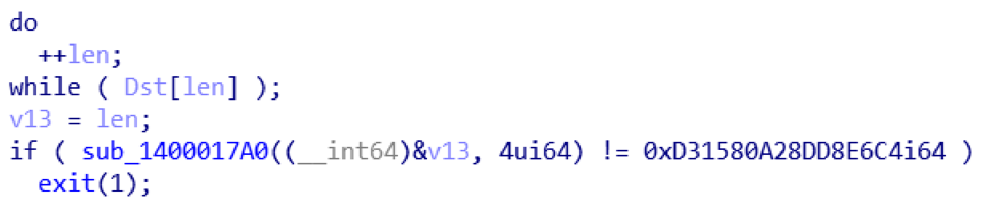
修改之后我们将后缀改为.dll然后就可以实现链接了，根据汇编我们可以看出是__fastcall的调用约定，因为是用rcx和rdx传的参数，编译的时候也需要用64位编译，因为我们调用的dll是64位

```
.text:000000014000193D BA 04 00 00 00       mov     edx, 4
.text:0000000140001942 89 44 24 20          mov     [rsp+158h+var_138], eax
.text:0000000140001946 48 8D 4C 24 20       lea     rcx, [rsp+158h+var_138]
.text:000000014000194B E8 50 FE FF FF       call    sub_1400017A0
```

然后我们就可以开始实现链接了：

```
typedef __int64(__fastcall *f)(__int64 buff, unsigned __int64 len);

int main()
{
 HINSTANCE hdll;
 hdll = LoadLibrary(TEXT("F:\\reverse.dll"));
 if (hdll == NULL)
 {
  printf("Load dll Error: %d\n", GetLastError());
  return 0;
 }
 printf("Dll base is %llx\n", hdll);

 f func = ((f)((char*)hdll + 0x17A0));
}
```

第一处是需要我们爆破长度，长度最后减去hxb2018{}这9位：

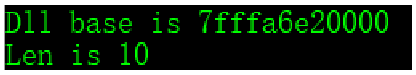


```
do
  ++len;
while ( Dst[len] );
v13 = len;
if ( sub_1400017A0((__int64)&v13, 4ui64) != 0xD31580A28DD8E6C4i64 )
  exit(1);
```

代码如下：

```
int i;
unsigned long long  result;
for (i = 0; i<50; i++)
{
 result = func((long long)&i, 4);
 if (result == 0xD31580A28DD8E6C4)
 {
  printf("Len is %d\n", i - 9);
 }
}
```

运行可以看到长度为10

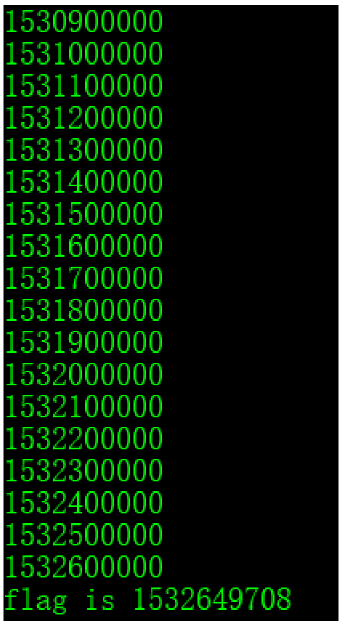


```
Dll base is 7fffa6e20000
Len is 10
```

第二次是需要我们爆破内容，通过sprintf快速制作10个字节的十进制数，然后穷举：

```
unsigned long long j;
unsigned long long result2;
char buff[20];
for (j = 0; j < 10000000000; j++)
{
 sprintf_s(buff, "%0.10llu", j);
 if (j % 100000 == 0)
 {
  printf("%0.10llu\n", j);
 }
 result2 = func((long long)buff, 10);
 if (result2 == 0xE3BE26AF8730545A)
 {
  printf("flag is %lld\n", j);
  return 0;
 }
}
```

最后爆破一段时间得到flag



```
1530900000
1531000000
1531100000
1531200000
1531300000
1531400000
1531500000
1531600000
1531700000
1531800000
1531900000
1532000000
1532100000
1532200000
1532300000
1532400000
1532500000
1532600000
flag is 1532649708
```

最后贴一个总代码

```
#include "stdafx.h"
#include<Windows.h>

typedef __int64(__fastcall *f)(__int64 buff, unsigned __int64 len);

int main()
{
 HINSTANCE hdll;
 hdll = LoadLibrary(TEXT("F:\\reverse.dll"));
 if (hdll == NULL)
 {
  printf("Load dll Error: %d\n", GetLastError());
  return 0;
 }
 printf("Dll base is %llx\n", hdll);

 f func = ((f)((char*)hdll + 0x17A0));

 int i;
 unsigned long long  result;
 for (i = 0; i<50; i++)
 {
  result = func((long long)&i, 4);
  if (result == 0xD31580A28DD8E6C4)
  {
   printf("Len is %d\n", i - 9);
  }
 }

 unsigned long long j;
 unsigned long long result2;
 char buff[20];
 for (j = 0; j < 10000000000; j++)
 {
  sprintf_s(buff, "%0.10llu", j);
  if (j % 100000 == 0)
  {
   printf("%0.10llu\n", j);
  }
  result2 = func((long long)buff, 10);
  if (result2 == 0xE3BE26AF8730545A)
  {
   printf("flag is %lld\n", j);
   return 0;
  }
 }
 return 0;
}
```

Tips：每个人的flag在复现的时候可能不同哦

0x02：总结

HighwayHash64这道题可以帮助我们熟悉PE文件结构，是一道很不错的题目，还有一道题目More efficient than JS我没有记录，大概是关于WebAssembly逆向的，可以使用idawasm插件在IDA中进行分析，也可以使用wasmdec 生成伪 c 代码分析，就是环境搭建比较麻烦。

参考的wp：

https://www.anquanke.com/post/id/164604#h2-6

https://impakho.com/post/hxb-2018-writeup