170926 逆向-Reversing.kr（ImagePrc）

原创

奈沙夜影 于 2017-09-26 21:11:32 发布 565 收藏

分类专栏： CrackMe

本文链接：https://blog.csdn.net/whklhhhh/article/details/78106981

1625-5 王子昂 总结《2017年9月26日》【连续第359天总结】

A. Reversing.kr-ImagePrc

B.

ImagePrc

首先查壳，运行发现是一个光秃秃的窗口，按一下Check按钮就弹窗”Wrong”

拖入IDA查找字符串，锁定回调函数sub_401130

在https://wiki.winehq.org/List_Of_Windows_Messages和http://www.cnblogs.com/findumars/p/4999292.html查到了Windows消息说明，从而可以分析出整个程序的框架：

首先初始化，然后当鼠标移动的时候使绘图指针跟随，当鼠标按下时画点

简单来说中间的白屏部分就是个画图板（不喜欢乱点的我根本就没有发现这一点啊OTZ

其实那些都不重要……

不过字符串中倒是没有找到Correct的消息提示，核心代码如下：

```
if ( wParam == 100 )
  {
    GetObjectA(hbm, 24, &pv);
    memset(&bmi, 0, 0x28u);
    bmi.bmiHeader.biHeight = cLines;
    bmi.bmiHeader.biWidth = v16;
    bmi.bmiHeader.biSize = 40;
    bmi.bmiHeader.biPlanes = 1;
    bmi.bmiHeader.biBitCount = 24;
    bmi.bmiHeader.biCompression = 0;
    GetDIBits(hdc, (HBITMAP)hbm, 0, cLines, 0, &bmi, 0);
    v8 = (void *)sub_40150B(bmi.bmiHeader.biSizeImage);
    GetDIBits(hdc, (HBITMAP)hbm, 0, cLines, v8, &bmi, 0);
    v9 = FindResourceA(0, (LPCSTR)0x65, (LPCSTR)0x18);
    v10 = LoadResource(0, v9);
    v11 = LockResource(v10);
    v12 = 0;
    v13 = v8;
    v14 = v11 - (_BYTE *)v8;
    while ( *v13 == v13[v14] )
    {
      ++v12;
      ++v13;
      if ( v12 >= 90000 )
      {
        sub_401500(v8);
        return 0;
      }
    }
    MessageBoxA(hWnd, Text, Caption, 0x30u);  // 错误提示
    sub_401500(v8);
    return 0;
  }
```

从FindResource、LoadResource、LockResource三个函数可以看出来应该是读取程序中的某个资源，然后获取绘图板上的图像
循环体就是依次对比的部分，v12是相同计数，v13是下标
如果总共相同点达到90000个就直接返回而不报错

这里大概就可以猜出来了，毕竟要9w个像素点完全重合可以说是不可能的事；核心问题就在于内存中的这张图片了

考虑如何Dump出资源，我选择OD（啥

加载入OD，在该处下断，可以发现v13和v13[v14]两处各有长为90000的内存块，大部分都是FF，少量地方出现00，并且都是连续三个一起出现

之前查函数的时候在GetDIBits中发现RGB相关的说明，所以这里正好三个值差不多能猜到是一个像素点的RGB值了。FF FF FF代表白色，00 00 00代表黑色。

从上面LockResource的返回值可以看出，v13[v14]的地方是原有图片，所以我们Dump它。

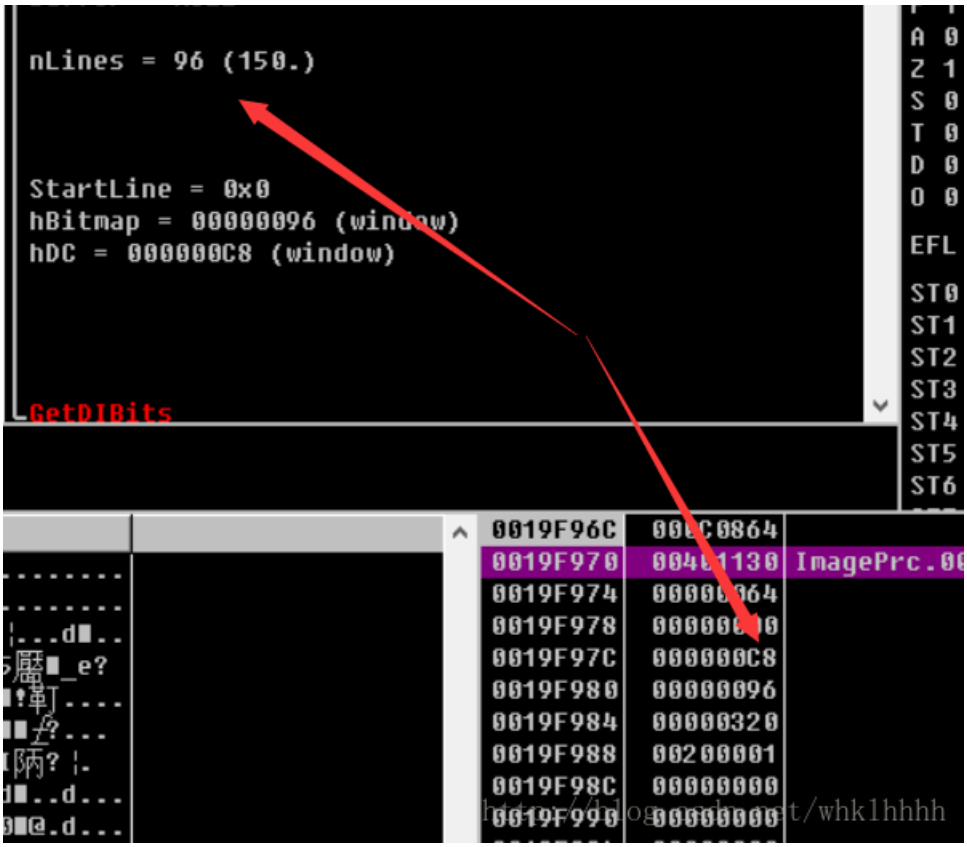
之后看别人的WriteUp学习时发现它是直接用eXeScope这个工具的，我是复制OD数据区后再进行清洗加工的。

复制下来以后由于还包含了地址和ASCII值，需要进一步清洗，通过python的split(' ')方法就可以分隔开无用数据和有效RGB了

下一步考虑如何将得到的数据流转换成图片。
首先绘图我使用的是Tkinter的Canvas，即GUI画布
每个像素点通过画长1宽1的直线即可实现

（别人的WriteUp显示PIL库提取RGB可能会更方便些，由于我没有经验所以就暴力的想到啥用啥了）
其次坐标点(x, y)转换成数据流是第（y * height + x）个像素点，每个像素点有3个值，即得到了(x, y)对应(y * height + x) * 3开始的3个值
由于黑色和白色的RGB是相同的，所以只需要考虑每个像素点的第一个值是FF还是00即可
OD中复制下来的数据清洗过后是每行16个值，再转换过去就是flag[p//16][p%16]

另一个关键的问题就是height和weight是多少
很容易发现Dump下来的数据是90000，也就是说30000个像素点
随便试了80、100，绘出来的图像都不可见
想起来IDA中识别出了GetDIBits的参数是有高和宽的，可惜不是常量；去OD中找一下，发现：



也就是说是150x200
这样就全部分析完成了

脚本为：

```
import tkinter
file = open("ImagePrc/dump.txt", 'rb')
list = file.readlines()
flag = []
for i in list:
    flag.append(i.split(b' '))
height = 150
weight = 200
top = tkinter.Tk()
c = tkinter.Canvas(top, bg='white')
for y in range(height):
    for x in range(weight):
        p = (y*weight + x)*3
        if(flag[p//16][2+p%16]==b'00'):#+2是因为清先后前面有2个无关数据，排除
            c.create_line((x, y, x+1, y+1))
c.pack()
top.mainloop()
```

得到图像：

不知道为什么是倒着的，不过也差不多能看出来是TOG/GOT，后者有意义，果然正确

话说我刚拿到题目就很好奇Prc是什么意思，查也没找到…感觉会有点提示啥的，吃了文化的亏*(:3」∠)*

另一篇使用eXeScope和PIL库使过程简单很多的WriteUp：
http://blog.csdn.net/yuanyunfeng3/article/details/49791067

C. 明日计划
Reversing.kr