**恶意代码分析与防治技术实验报告**

**Lab13**

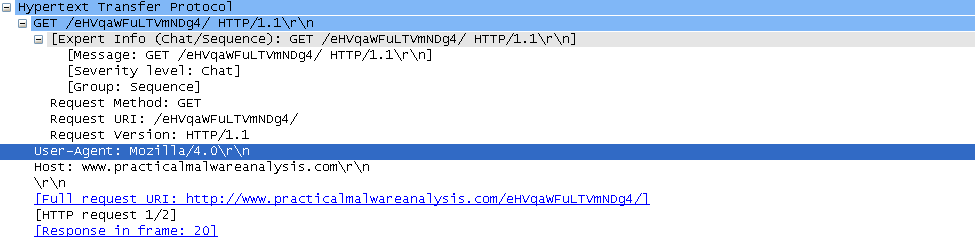
**学号：2013018 姓名：许健 专业：信息安全**

1. **实验内容**

完成课本Lab13的实验内容

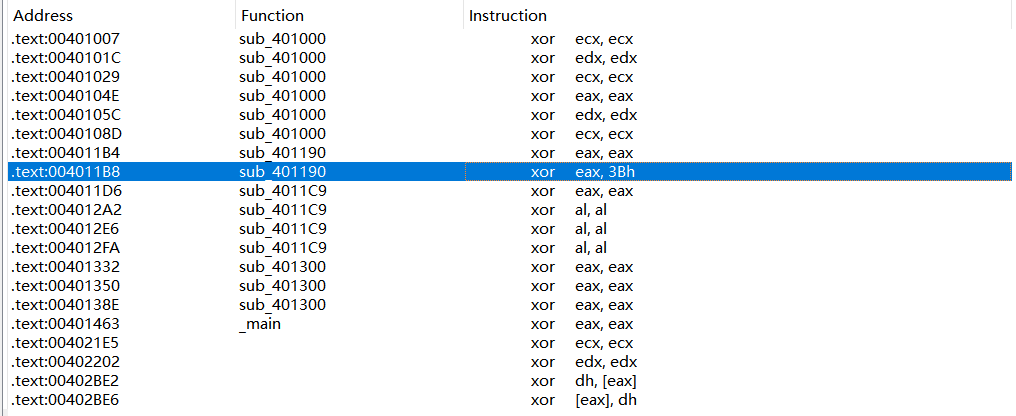
1. **问题解答**
2. Lab13-1
3. 比较恶意代码中的字符串（字符串命令的输出）与动态分析提供的有用信息，基于这些比较，哪些元素可能被加密？

通过wireshark抓包监控，过滤出http之后，主要到有个get请求，访问的是www.practicalmalwareanalysis.com，但是get后面的字符串似乎是编码或者混淆过的。网址和字符串在strings都没找到。

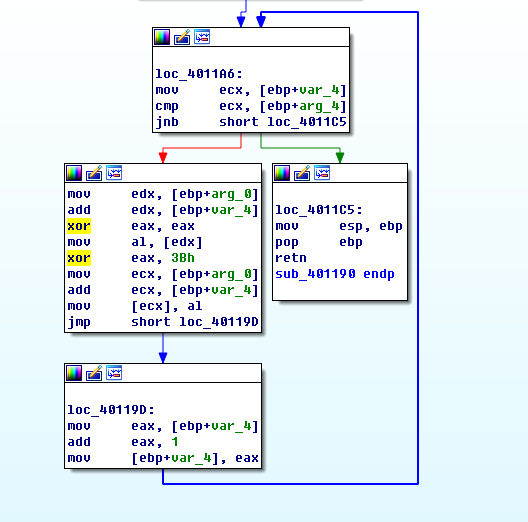


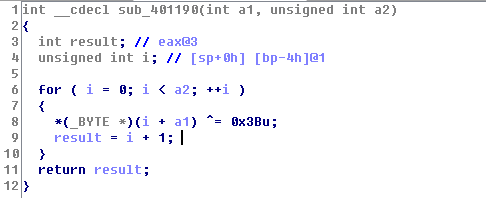
1. 使用IDA Pro搜索恶意代码中字符串‘xor’，以此来查找潜在的加密，你发现了哪些加密类型？

查找非归零XOR指令，xor eax, 3Bh



这是一段单字节XOR循环





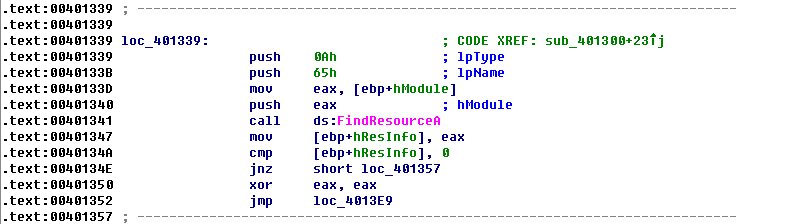
1. 恶意代码使用什么密钥加密，加密了什么内容？

可以看到这里是一个循环，循环中递增的是计算器var\_4，异或的地方在004011b8，用0x3b与eax异或，而eax来自缓冲区arg\_0，这里的arg\_4则是缓冲区的长度。所以我们知道了这里是实现用0x3b执行单字节的异或加密，并将缓冲区即其长度作为参数。给该函数重命名为xorEncode。查找xorEncode的交叉引用。

函数sub\_401300是调用xorEncode的唯一函数，跟踪他调用xorEncode之前的代码块，调用了GetModuleHandleA、FindResourceA、SizeofResource、GlobalAlloc、LoadResource和LockResource。大部分是与资源相关的函数。

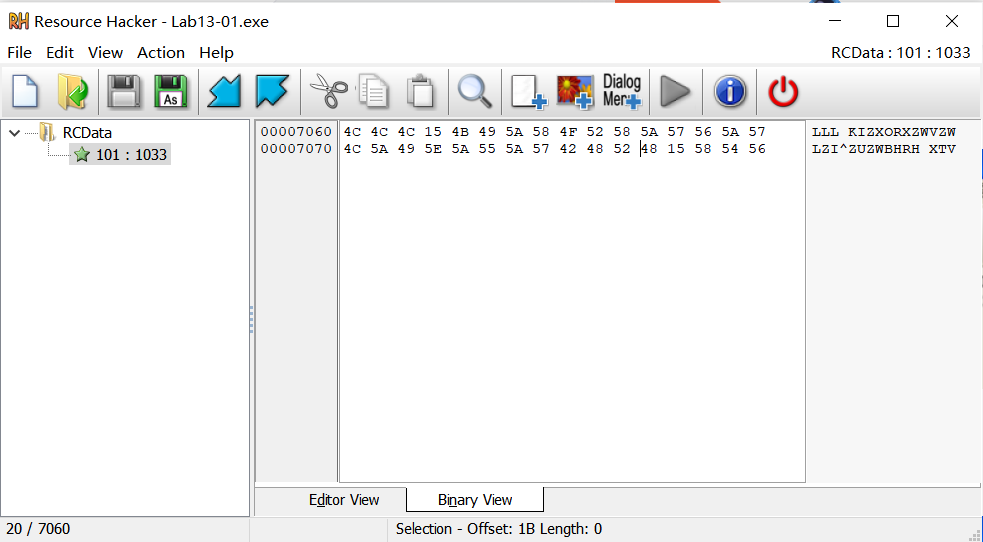
关注用于找到资源的函数FindResourceA

看到传入的lpType参数为0xA，查阅msdn知，表示资源数据时应用程序预定义的资源类型是原始数据。lpName在IDA中可以看到是索引号65h，10进制为101。

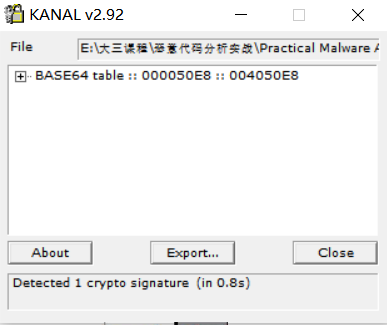


打开resource hacker，找到索引号为101的RCDATA，可以看到资源在偏移量0x7060,长度为32字节。异或后得到的字符串为77 77 77 2E 70 72 61 63 74 69 63 61 6C 6D 61 6C 77 61 72 65 61 6E 61 6C 79 73 69 73 2E 63 6F 6D。

对应字符串www.practicalmalwareanalysis.com，也就是wireshark抓到的网址。



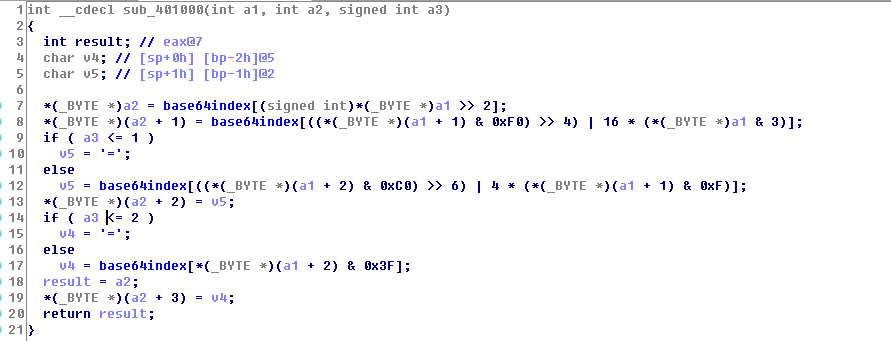
1. 使用静态工具FindCrypt2、Krypto ANALyzer（KANAL）以及IDA熵插件你发现了什么？

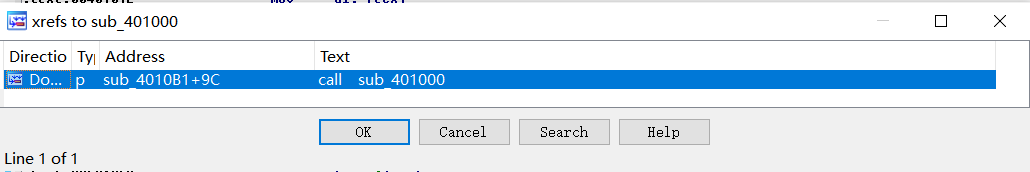


使用PEid的插件KANAL可识别出恶意代码使用标准的Base64编码字符串:ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/

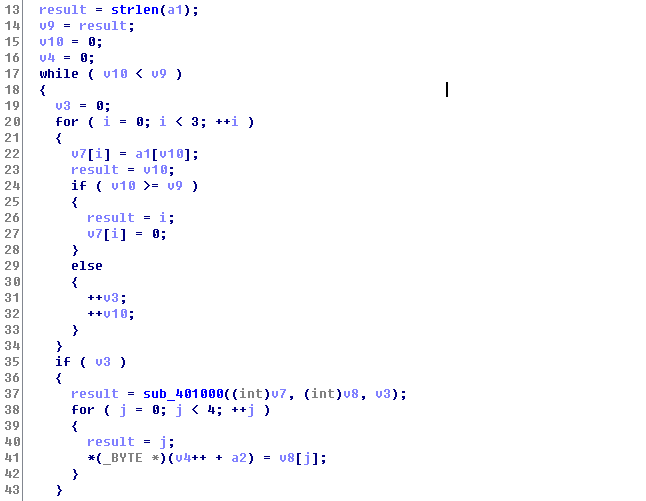
查看该字符串的交叉引用，四处都在sub\_401000中。

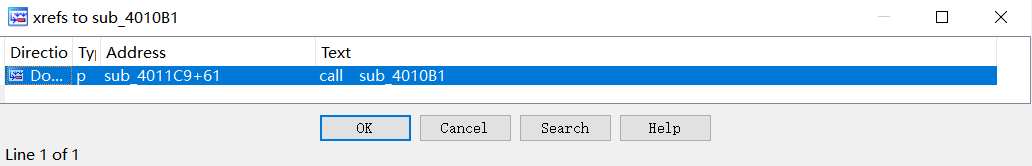
3Dh实际上就是=。因为base64加密中是用‘=’作为填充字符的，所以也印证了这是base64。



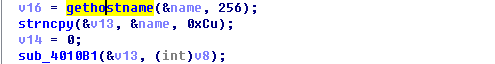


查看交叉引用，sub\_4010B1调用sub\_401000，v9是字符串长度，while的目的就是对源字符串的每个字符都进行接下来的操作。for循环是将字符串进行三字节一组的切割，也就是将字符串划分成3字节的块，保存在v7中。if-else是用于判断是否赋值完毕，完毕则在v7最后加上0表示结束，否则自增。37行看到是将3字节块传给base64index，而38行的循环这里是用于将传入的3字节加密成4字节。





查看交叉引用，F5查看反编译C代码，定位传入的参数，发现是主机名。



1. 什么类型的加密被恶意代码用来发送部分网络流量？

标准的Base64编码用来创建GET请求字符串

1. Base64编码函数在反汇编的何处？

0x004010B1处

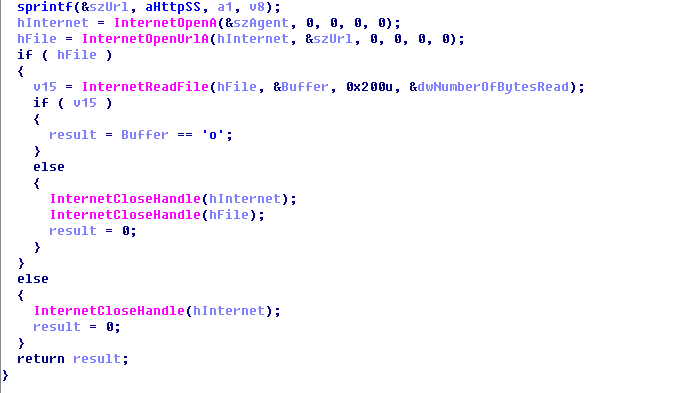
1. 恶意代码发送的Base64加密数据的最大长度是多少？加密了什么内容？

Base64 加密前，Labl3-01.exe 复制最大12个字节的主机名，这使得GET请求的字符串的最大字符个数是16,加密的内容为请求的字符串。

1. 恶意代码中，你是否在Base64加密数据中看到了填充字符（=或==）？

如果主机名小于12个字节并且不能被三整除，则可能使用填充字符。

1. 这个恶意代码做了什么？



成功返回后会在004012f1与6fh，也就是字符o进行比较。如果是o，则走左边，将eax赋1。如果不是o，则走右边，通过xor异或自身，清空eax，也就是给eax赋0

Lab3-01.exe用加密的主机名发送一个特定信号，直到接收特定的回应后退出。

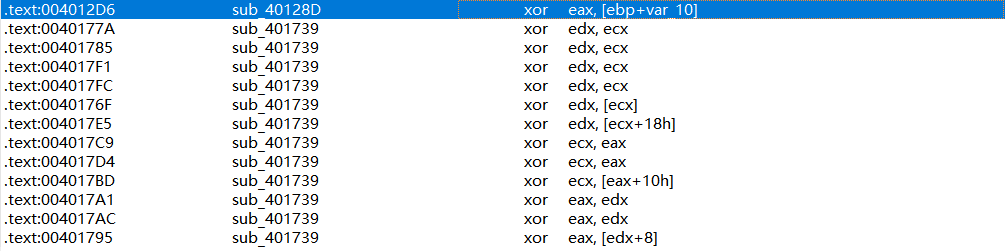
1. Lab13-2
2. 使用动态分析，确定恶意代码创建了什么？

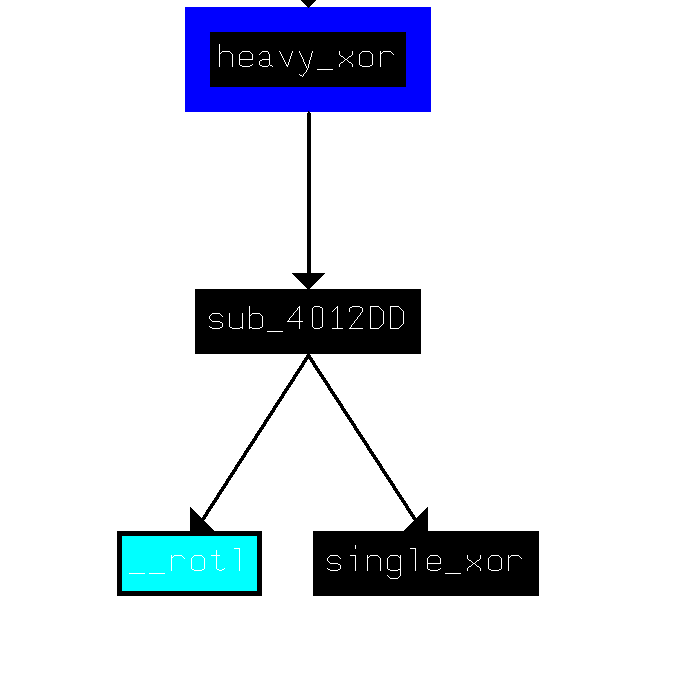


恶意代码在当前目录下创建一些较大且看似随机的文件，这些文件的命名以temp开始，以不同的8个十六进制数字结束。

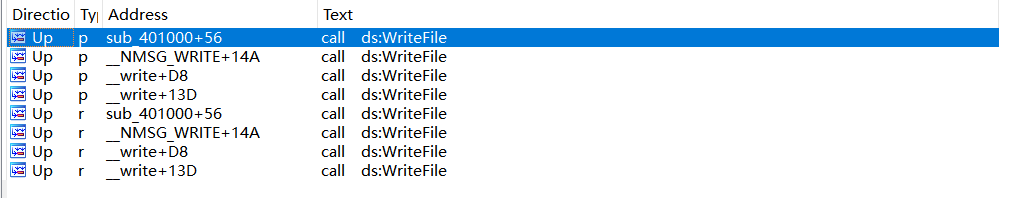
1. 使用静态分析技术，例如xor指令搜索、FindCrypt2、KANAL以及IDA熵插件，查找潜在的加密，你发现了什么？

XOR搜索技术在sub\_\_401728d(single\_xor)和sub\_401739(heavy\_xor)中识别了加密相关的函数。

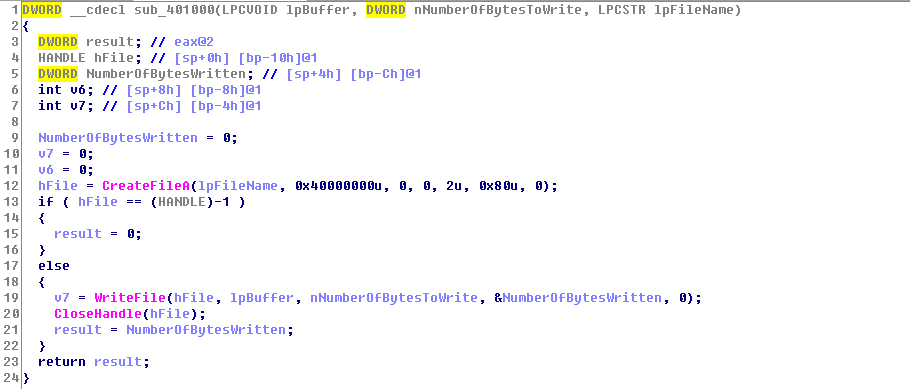




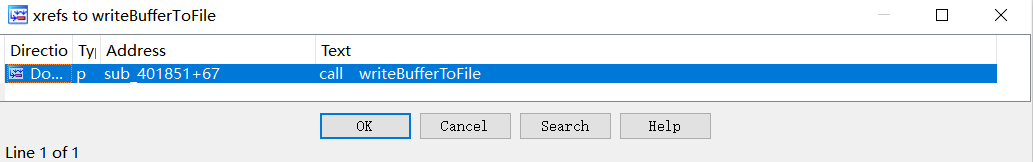
猜测heavy\_xor是一个加密函数，为了证实猜测，我们需要找到它和那些temp文件的关系。在IDA的import窗口找到writefile，查看交叉引用。



定位sub\_401000函数，作用是创建一个文件，并将缓冲区的内容写入文件。



查看该函数的交叉引用，它被sub\_401851调用，跟踪该函数。





可以看到调用了sub\_401070，sub\_40181f。其参数都为缓冲区和缓冲区的长度。之后是调用GetTickCount，根据00401895处的格式化字符串可知，这就是打印文件名的地方，文件名的后半部分是当前时间的十六机制。我们推测，sub\_401070用于获取内容，sub\_40181f用于加密内容。

1. 基于问题1的回答，哪些导入函数将是寻找加密函数比较好的一个证据？

WirteFile调用之前可能会发现加密函数。

1. 加密函数在反汇编的何处？

加密函数是sub\_40181F

1. 从加密函数追溯原始的加密内容，原始加密内容是什么？



GetDesktopWindow获取覆盖整个屏幕桌面窗口的一个句柄，函数BitsBlt，GetDIBits获取位图信息并把他们复制到缓冲区。我们猜测，这是用于抓屏。而上一层函数这是将抓屏信息加密后写到temp开头的文件里。原内容是屏幕截图。

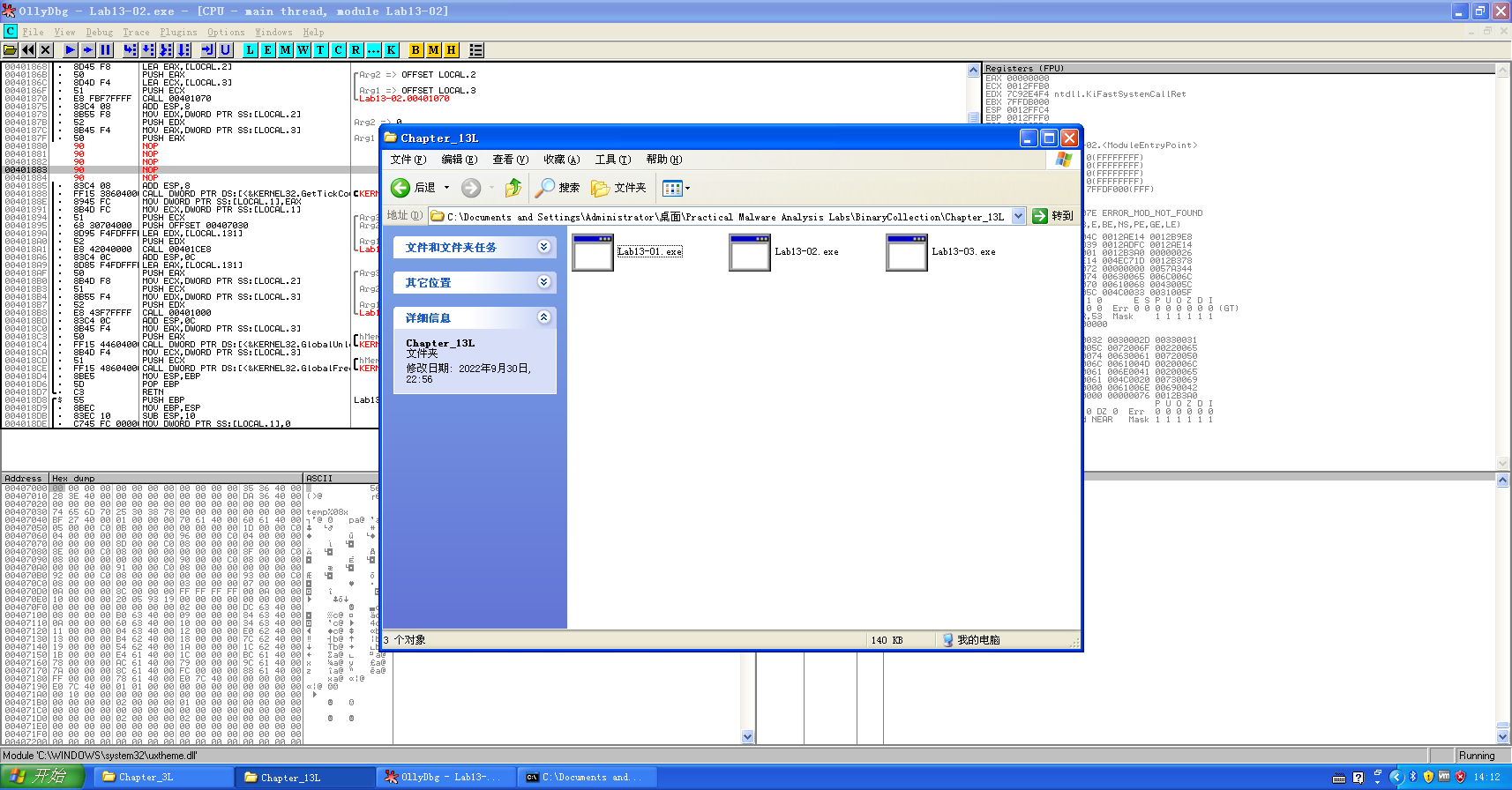
1. 你是否能够找到加密算法？如果没有，你如何解密这些内容？

加密算法是不标准算法，并且不容易识别，简单方法是通过解密工具解密流量。

1. 使用解密工具，你是否能够恢复加密文件中的一个文件到原始文件？

解密的话可以分析加密过程编写脚本来进行解密，也可以用OD解密流量，不过我选择直接不执行加密过程。

为了验证猜想，我在这里利用一个取巧的办法，程序有加密函数，那么我们便不让加密函数执行，直接在函数头返回。OD载入样本，找到加密函数0040181F这个位置，直接nop掉。然后运行程序，得到未加密的文件，加上后缀.tmp，可以看到确实是屏幕截图。

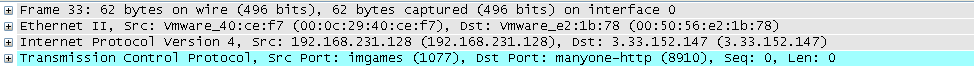
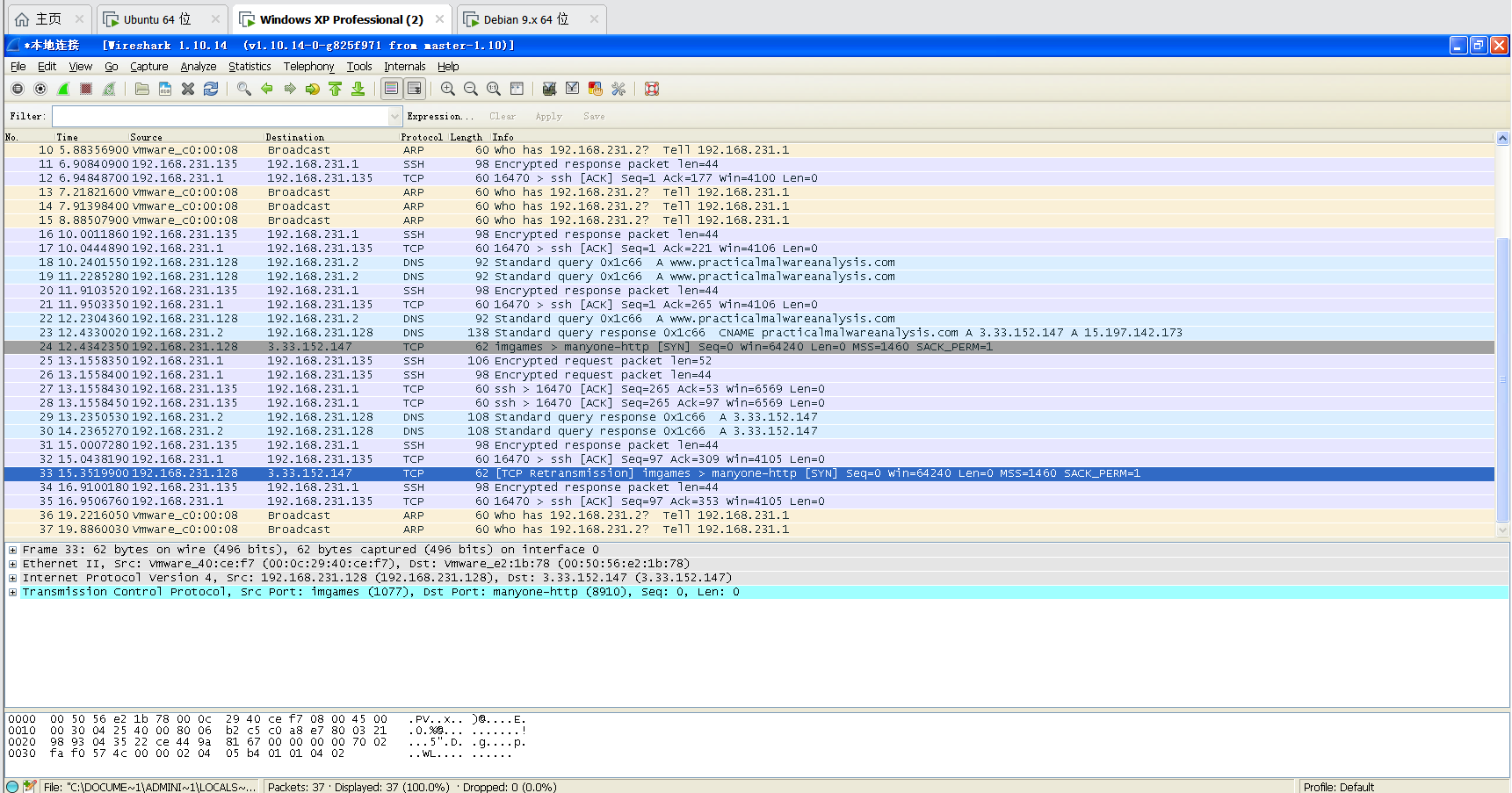


1. Lab13-3
2. 比较恶意代码的输出字符串和动态分析提供的信息，通过这些比较，你发现哪些元素可能被加密？

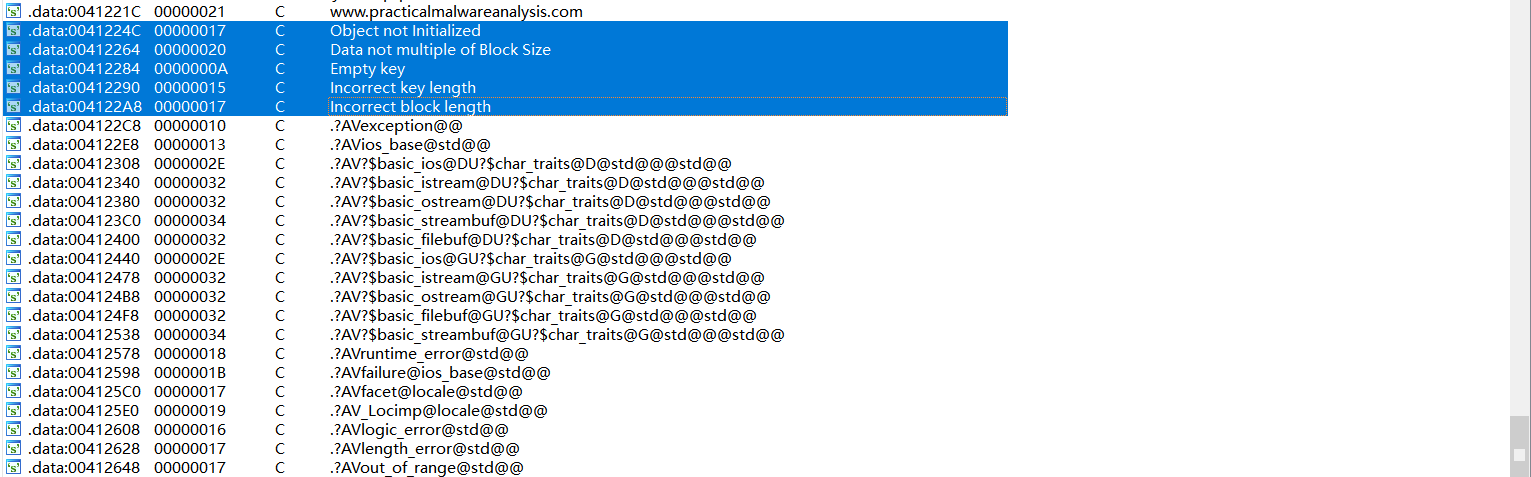
首先启动wireshark进行监控，然后双击运行lab13-03.exe

可以看到恶意代码尝试通过dns解析域名www.practicalmalwareanalysis.com

并且尝试建立tcp连接，连接到远程主机的8910端口



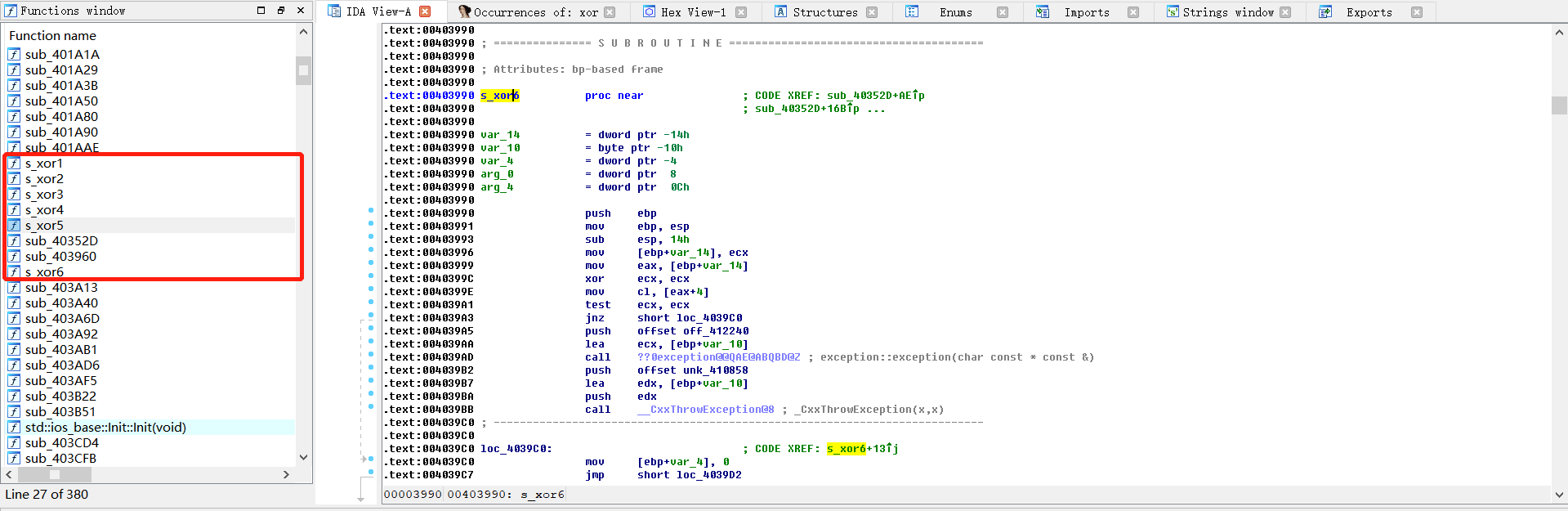
看到了在wireshark中的发现的域名，下面的字符串疑似与某种加密有关，从字面意义可知，如空密钥，不正确的密钥长度，不正确的块长度等等。



1. 使用静态分析搜索字符串xor来查找潜在的加密，通过这种方式，你发现什么类型的加密？

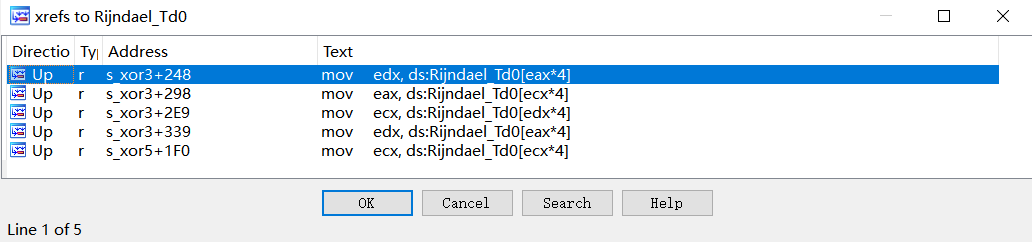
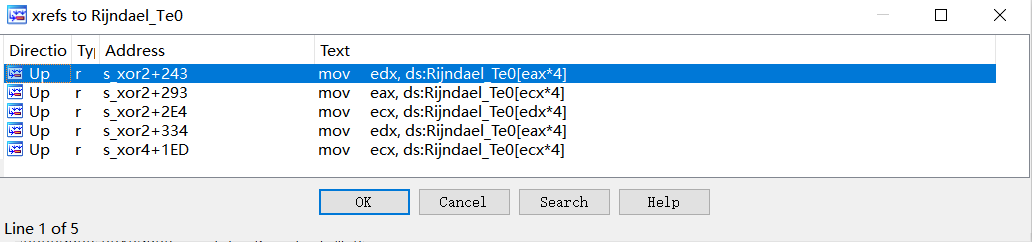
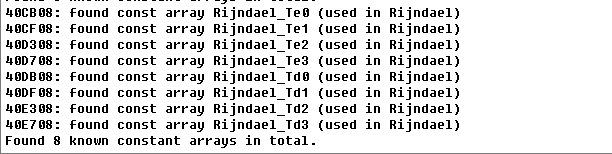
搜索xor指令发现了6个可能与加密相关的单独函数，但是加密的类型看不出来。

将这些函数重命名为s\_xor1-6



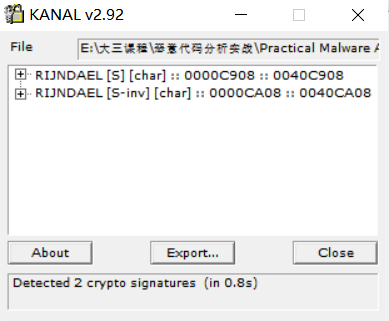
1. 使用静态工具，如FindCrypt2、Krypto ANALyzer（KANAL）以及IDA熵插件识别一些其他类型的加密机制，发现的结果与搜索字符XOR结果比较如何？

FindCrypt2插件查找出AES算法

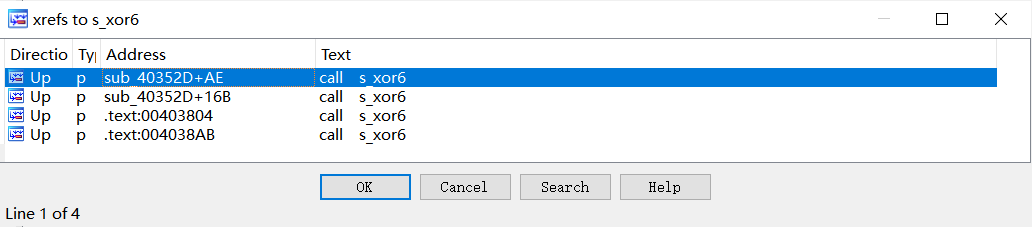


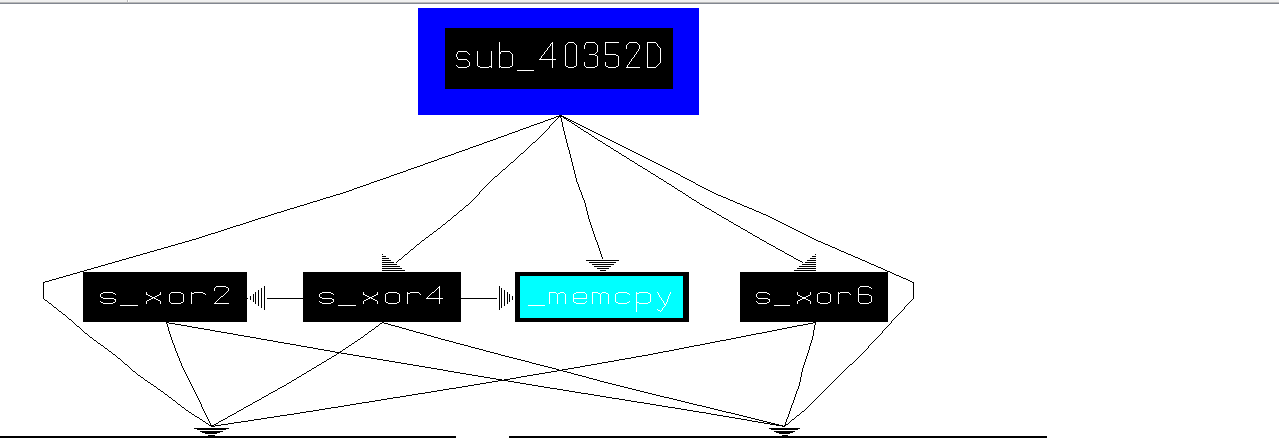
s\_xor2，s\_xor4和AES加密相关，而s\_xor3，s\_xor5与AES解密相关。

KANAL识别出的S，S-inv参考了S-box结构，它是一些加密算法的基本机构，和IDA识别出的加密算法是一样的。



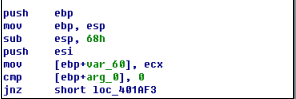
查看s\_xor6的交叉引用，跟入sub\_40352d查看xrefs from



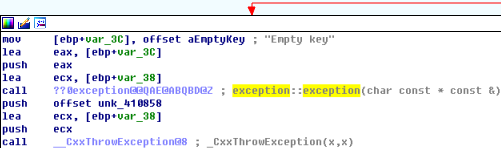


从上图中可以看到s\_xor6和s\_xor2,s\_xor4相关，都是加密相关的。

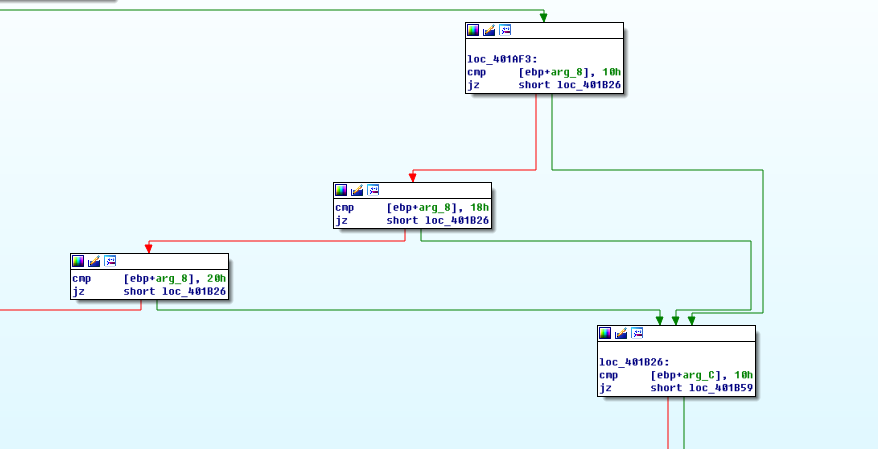
还有s\_xor1不知道是做什么的，可以去看看



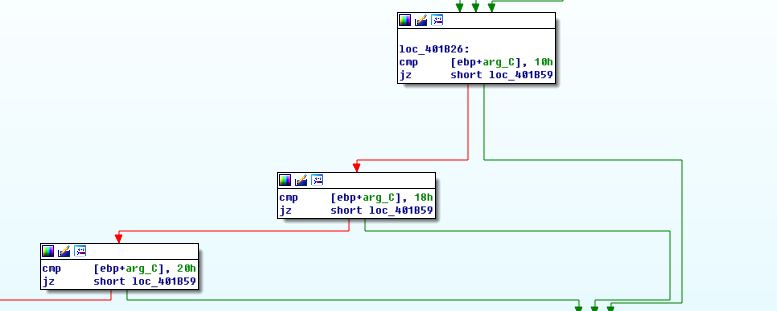
通过cmp将arg\_0与0比较。如果不正确，往左边走



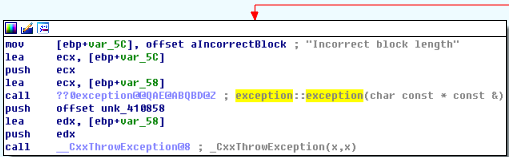
会报错空密钥。如果arg\_0通过检测，则往右走



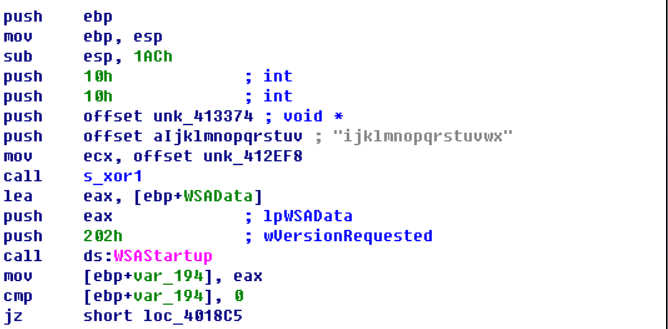
会检测arg\_8是否正确，如果错误，往左走，会报错不正确的密钥长度  
如果arg\_8正确，往右走



此时会检测arg\_c，如果错误，则往左

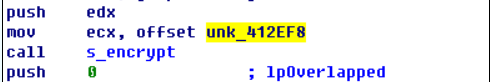


会报错不正确的块长度。也就是说s\_xor1是密钥的初始化代码。查看s\_xor的交叉引用



调用s\_xor1之前，存在unk\_412ef8的引用，00401890通过ecx将这个偏移量传入s\_xor1。

我们看看unk\_412ef8的其他引用，跟踪第三处。



00401429将unk\_412ef8的偏移量载入ecx，而随后就是s\_encrypt的调用  
可知unk\_412ef8是一个表示AES加密器的C++对象，并且s\_xor1是加密器的初始化函数

前面分析的时候说过，比较arg\_0参数，如果失败则会提示空密钥，说明arg\_0参数就是一个密钥。s\_xor1设置的AES密码就是之前看到的那串字符串

1. 恶意代码使用哪两种加密技术？

恶意代码使用AES和自定义的Base64加密



1. 对于每一种加密技术，它们的密钥是什么？

AES的密钥是ijklmnopqrstuvwx，自定义的Base64 加密的索引字符串是

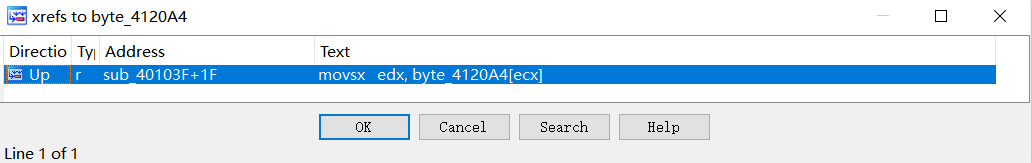
CDEFGHIJKLMNOPQRSTUVWXYZABcdefghi jklmnopqrstuvwxyzab0123456789+/.

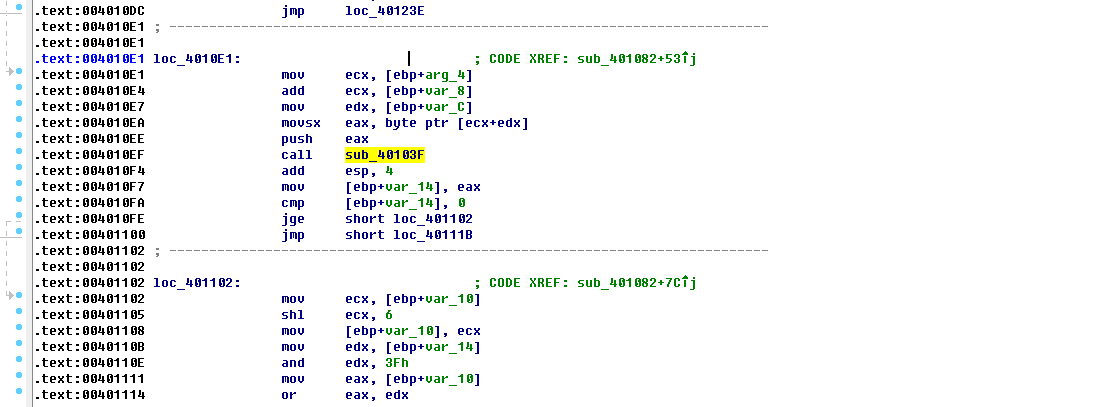
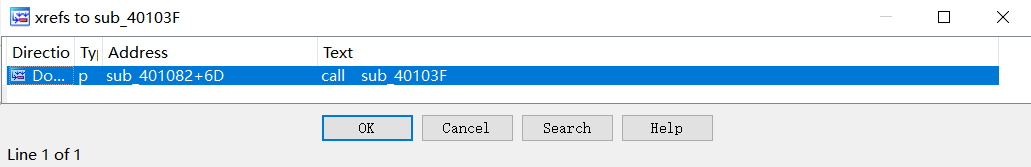
1. 对于加密算法，它的密钥足够可靠吗？另外你必须知道什么？

对于自定义Base64加密的实现，索引字符串已经足够了。但是对于AES， 实现解密可能需要密钥之外的变量。如果使用密钥生成算法，则包括密钥生成算法、密钥大小、操作模式，如果需要还包括向量的初始化等。

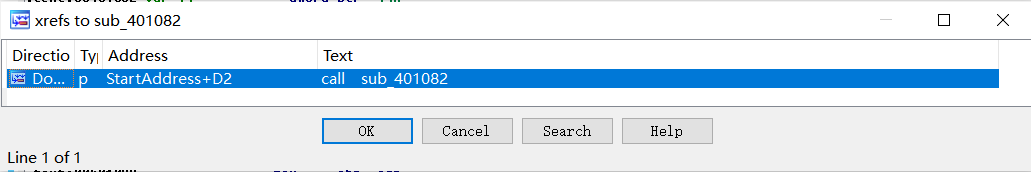
1. 恶意代码做了什么？

查看自定义的Base64加密的索引字符串的交叉引用，跟入sub\_40103f查看其交叉引用

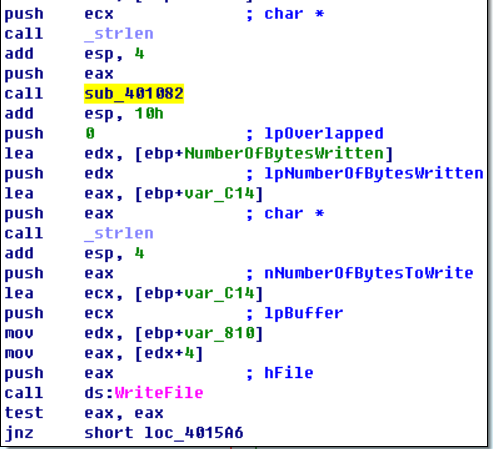


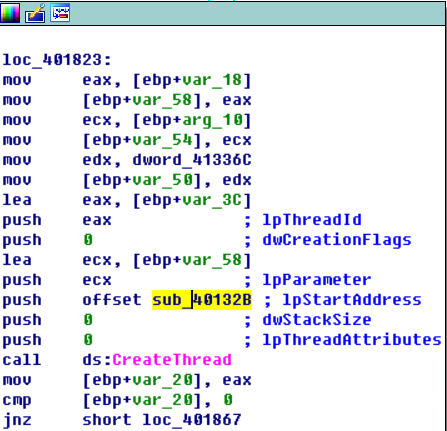


查看函数sub\_401082的交叉引用，跟入StartAddress



可以看到sub\_401082是在ReadFIle和WriteFile之间被调用的，这和我们前面分析的AES很像，前面我们知道s\_encrypt是在sub\_40132b处被调用的。查看其交叉引用来到sub\_4015b7。可以看到sub\_10432b是一个新线程的开始，这个新线程使用CreateThread创建，所以给sub\_40132b重命名为aes\_thread。

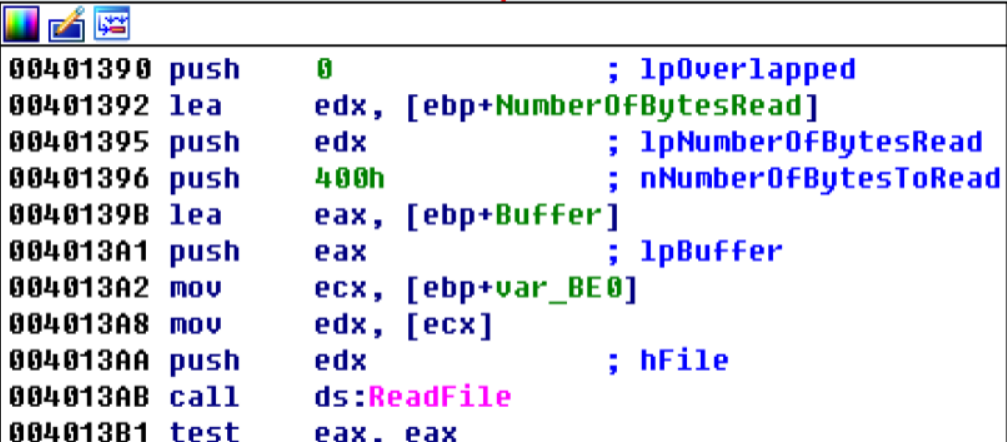




传递给线程的参数保存在lpParameter，也就是var\_58中。00401826处var18移入var\_58。

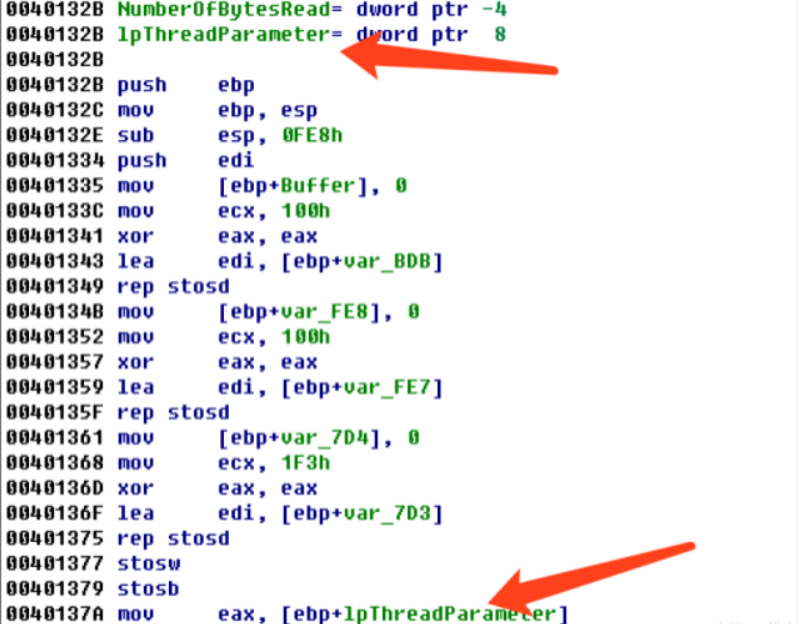
00401835处dwrord\_41336c移入var\_50

然后跟进aes\_thread看看这些参数在函数中的流程。先看ReadFile

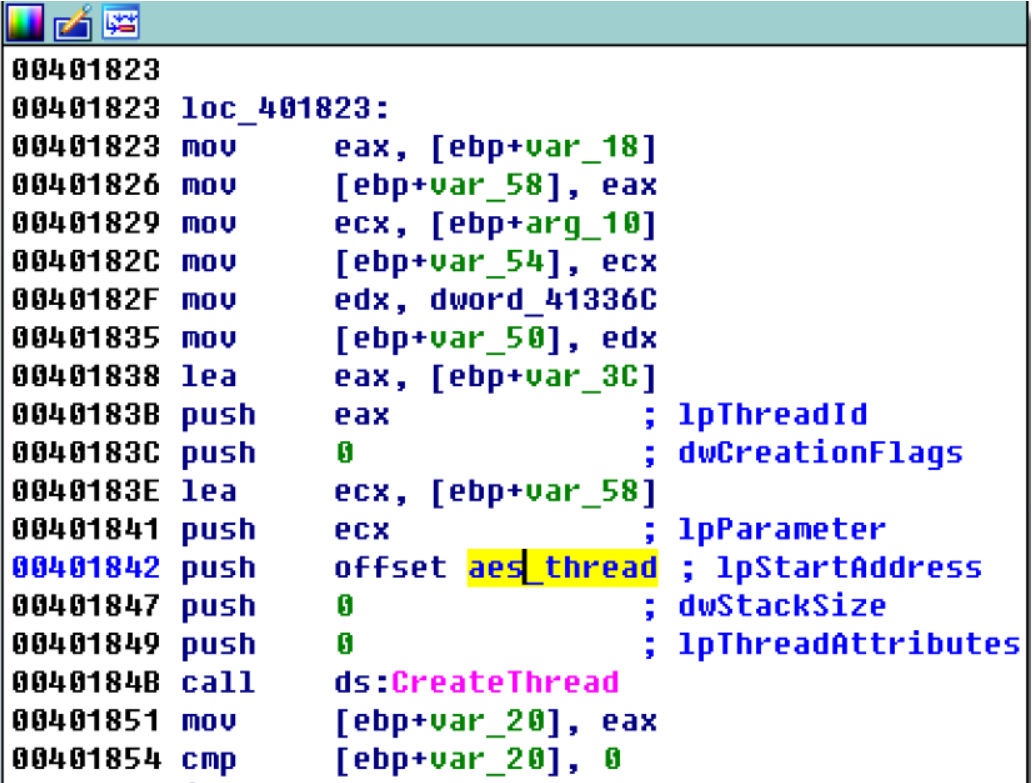


其参数hFile来自var\_BE0

往上回溯，可知其来自该函数的唯一参数



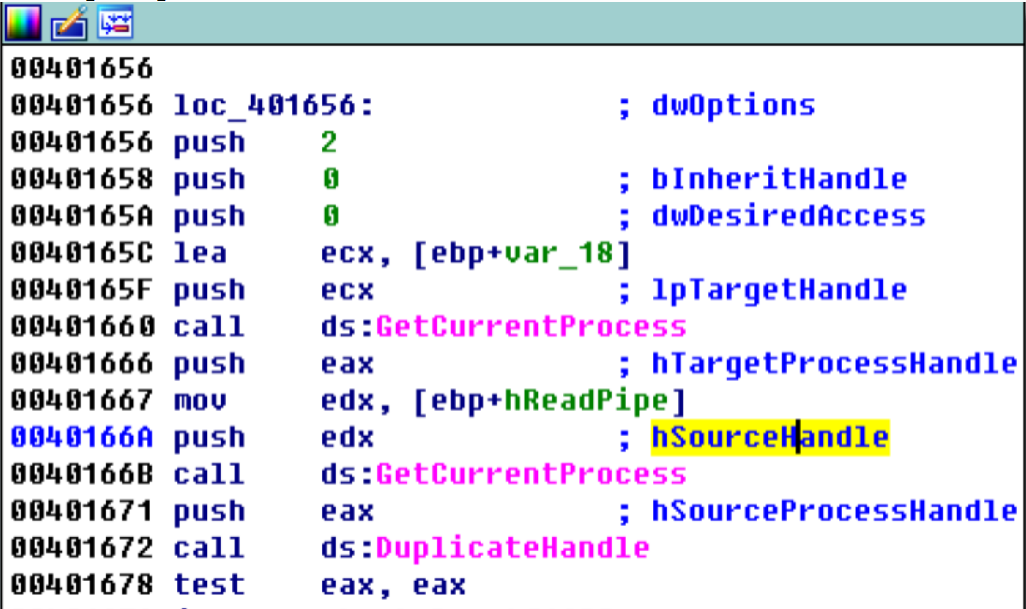
回到上层函数



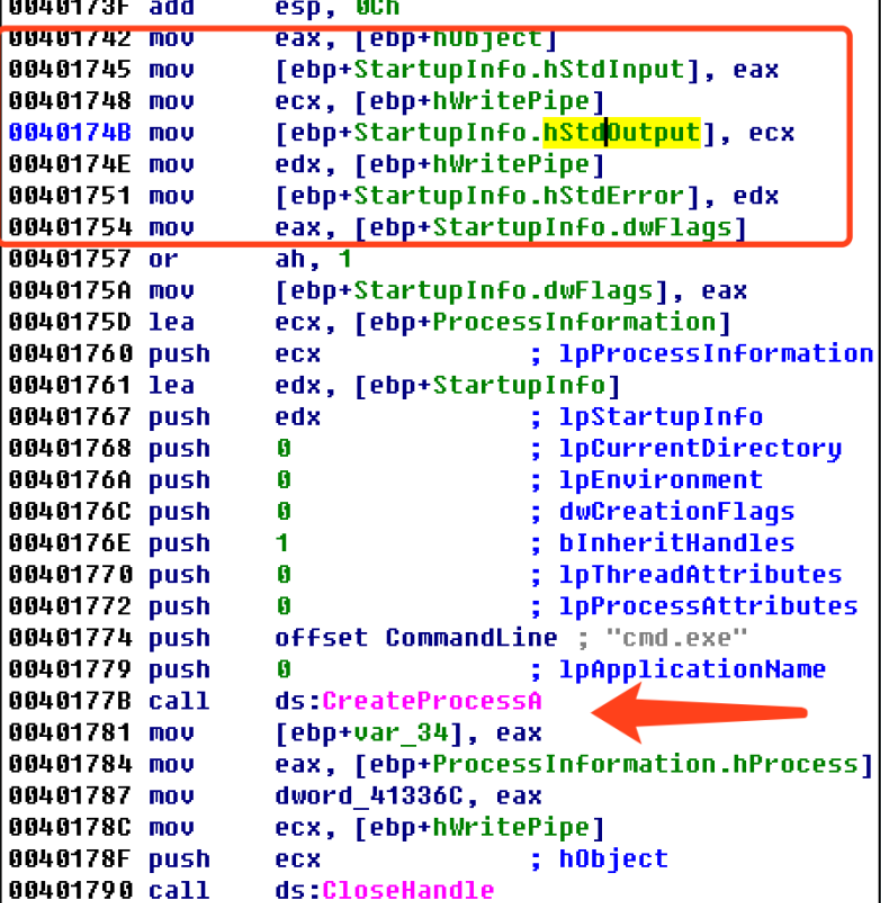
可以看到唯一的lpParameter参数来自var\_58,或者说来自var\_18

再看看ase\_thread的WriteFile。参数hFIle来自var\_BE0+4，就是var\_54或者说是arg\_10

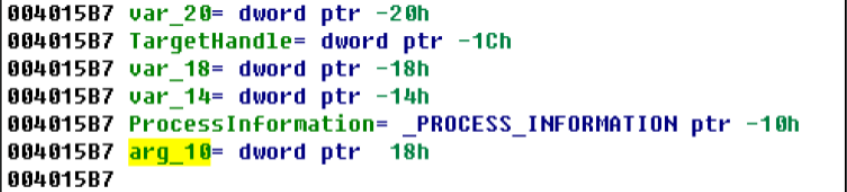
我们看到var\_58和var\_18持有一个管道的句柄，并且这个管道的与一个shell命令的输出相连接。



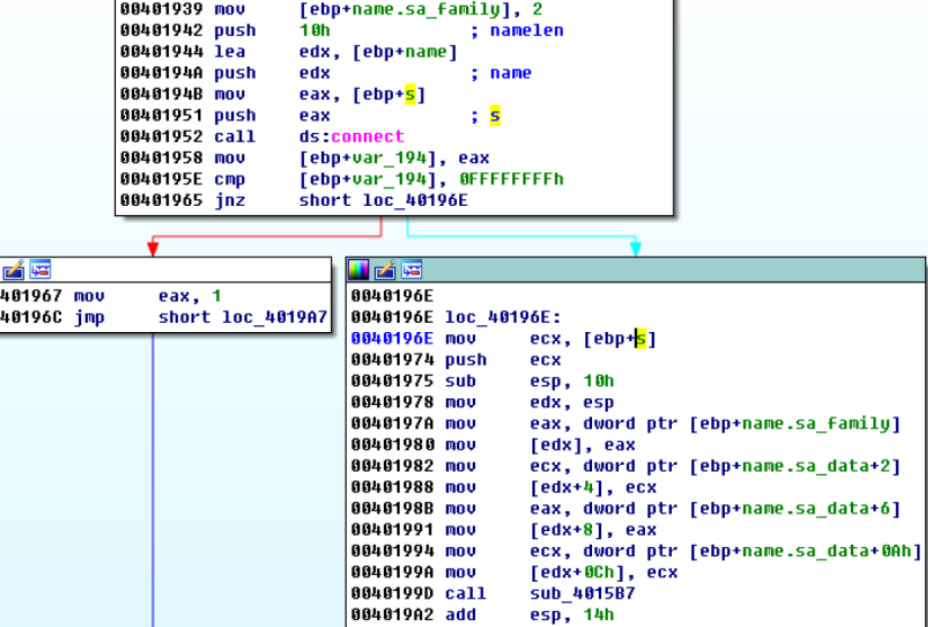
命令hSourceHandle通过DuplicateHandle复制到shell命令的标准输出和标准错误



这条shell命令由CreateProcess启动，而var\_54或者说arg\_10往前回溯



看到是sub\_4015b7的唯一参数，查看交叉引用来到main



可知其参数来自ebp+s,而它是用connnect调用创建的一个网络套接字  
那么我们也就知道了，aes\_thread用于读取shell命令的输出结果，在写入网络套接字之前加密它。而base64加密函数也在一个由他们宿主进程启动的函数中使用，所以我们猜测base64线程会读取远程套接字的内容作为输入，经过解密后，再将结果发送作为命令shell的输入。

恶意代码使用以自定义Base64加密算法加密传入命令和以AES加密传出shell命令响应来建立反连命令shell。

1. 构造代码来解密动态分析过程中生成的一些内容，解密后的内容是什么？

解密BInaEi==，使用自定义base64解码得到dir

使用wireshark捕获信息，然后使用AES解密得到

