**前言**

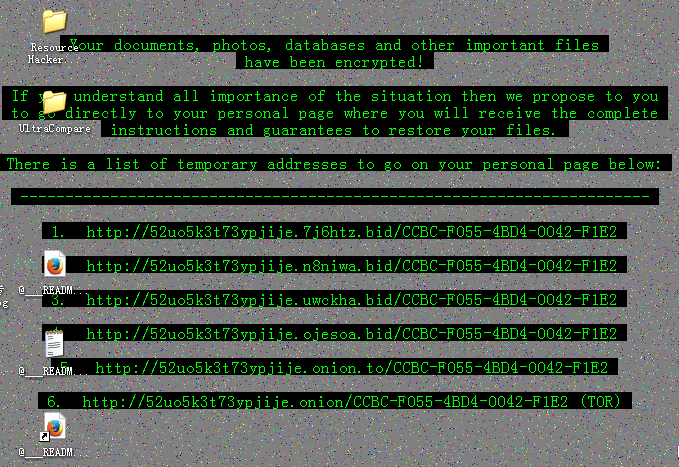
我们这次课程所讨论的是exe类恶意程序的快速分析技巧。事实上，对于exe类程序而言，在绝大多数情况下是无法对其在仅仅使用Hiew的前提下实现快速分析的，很多时候要么依靠更多的分析工具，要么直接在虚拟机中运行。这里我把在虚拟机中直接运行出的结果，作为快速分析的一种方式。毕竟只是做黑白判断，因此很多时候，只要样本没有反虚拟机手段，采用这种方式还是会很快得出黑白的结果的。

**样本分析**

这次选择的几个样本还是比较有特色的，值得我们好好研究一下：

**1、dd66bcf26c50c12f2d1036ada8cc8c14**

这个文件和上次课的文件不同，它没有导出表，它是一个exe可执行程序，通过分析入口点代码，似乎和我们之前课程中讲的知识都匹配不上，并没有发现异常的地方，那么我们不妨把它放到虚拟机里面运行看看结果。在虚拟机里运行之前，需要首先把它的后缀，也就是.exe加上，此时可以看到它的图标，似乎是一个用于下载或更新的程序，以此来打消大家的戒心，然后双击运行看看：

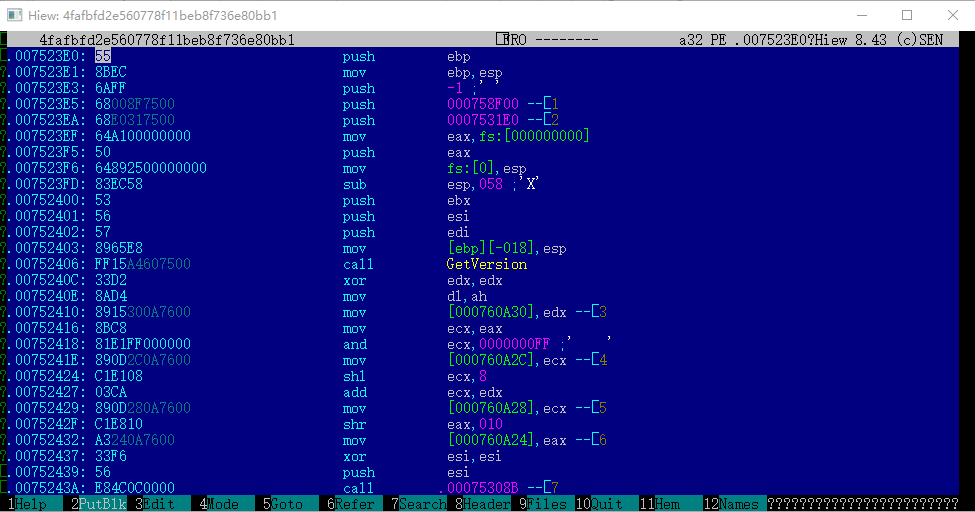


可以看到，这是一个典型的勒索程序，那么我们就可以直接将其归类为：Trojan-Ransom.Win32.Zerber.a。

**2、4fafbfd2e560778f11beb8f736e80bb1**

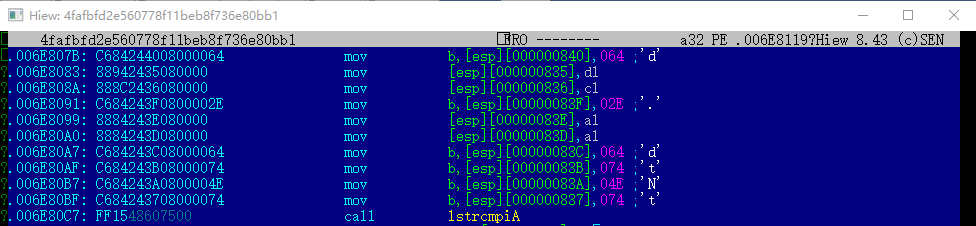
我们再看一个相对复杂的exe程序的情况。实际上，分析exe程序本身就需要分析师具备一定的分析经验，而在不借助其它分析工具的情况下，只使用Hiew时，其实更加考验分析师的综合能力了，需要分析师深入理解恶意程序，知道常用的混淆手段，更加应该熟悉PE文件的特点，是需要时间与一定的样本量作为积累的，所以病毒分析，特别是快速分析技巧，绝对是一个技术活。

对于当前程序来说，我们利用Hiew打开以后，利用F8、F5跳转到程序的入口点，然后切换到反汇编代码的模式：

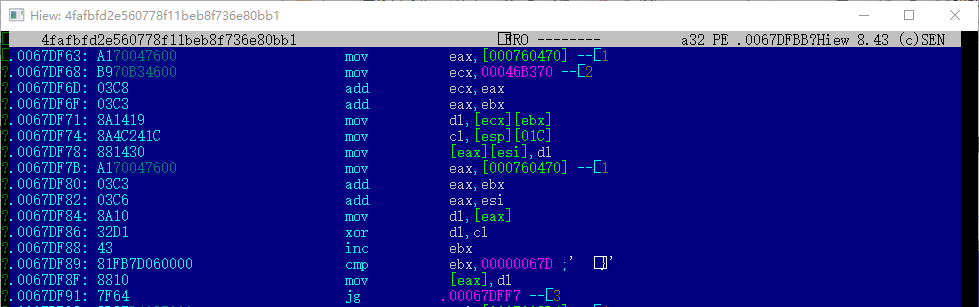


这段代码就是这个程序最开始执行的代码，但事实上，我们并不需要分析这段代码，因为这是编译系统自动为我们生成的，可以说几乎所有的类似的PE程序在起始位置都会包含这样的一段代码。因此分析时我们需要找到main函数的位置。这里就需要经验了，我们往下看，在GetVersion、GetCommandLineA、GetStartupInfoA、GetModuleHandleA这一系列函数的后面有一个call，其实这个call所调用的就是main函数了。按下相应的快捷键，就可以跳转到0x00613B50，即main函数的位置了，这个位置，也就是这个样本实际要执行的代码，是我们重点分析的部分。

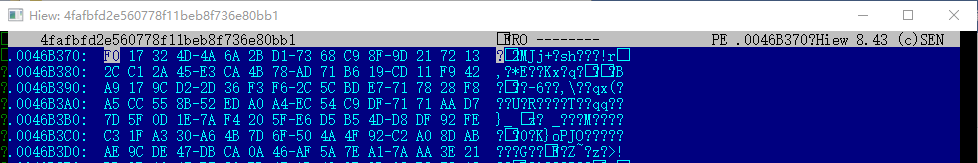
在这个main函数中，我们主要分析的是call，比如看一下位于0x00613BAD这个位置的call，利用快捷键跳转过去：



在这段代码中，我们可以看到非常多的将单个字符移入内存的情况，如前所述，这就是非常可疑的地方了。继续往下看，在0x006E824A的位置也有一个call，跳转过去看一下：



从0x0067DF63这个位置开始，实际上是一个解密过程，为什么这么说呢？首先，在0x0067DF68的位置，有一个mov的赋值语句，也就是将0x0046B370这个地址的内容赋给ecx，那么我们不妨看一下这个地址里面的内容：

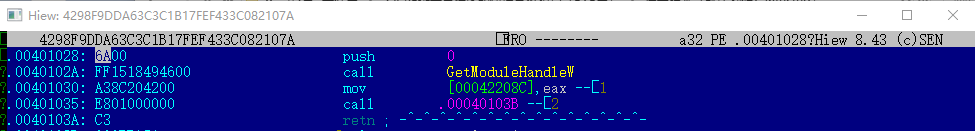


可以看到，这里就是一堆乱码，很可能是经过加密的。接下来则是一系列的运算过程，有add加法运算，有xor异或运算等，其中的这个异或运算实际上是恶意程序经常采用的一种解密手段。从最后的inc自增操作以及cmp比较操作可以得知，ebx保存的是想要解密的二进制代码的个数，这里是0x67D。

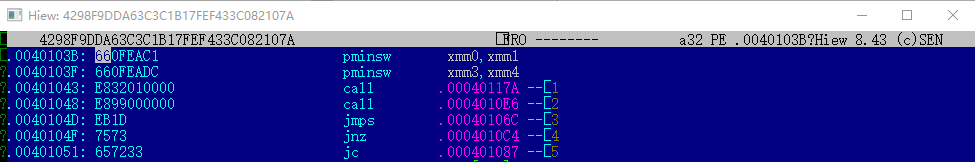
分析至此，我们就已经获得了非常充分的证据来证明这是一个恶意程序了。因为一个正常的程序是完全没有必要通过解密自身的方式来获得代码或者数据的，恶意程序之所以这么做，目的就是为了增加分析师的工作量，但是这种掩耳盗铃的方式反而引起了我们的怀疑。那么这里我们就可以把它归类为：Trojan.Win32.Agent.b。

**3、4298F9DDA63C3C1B17FEF433C082107A**

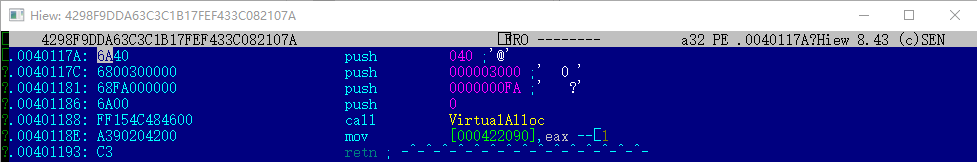
类似的样本还有这个4298F9DDA63C3C1B17FEF433C082107A，来到入口点可以看到一个call：



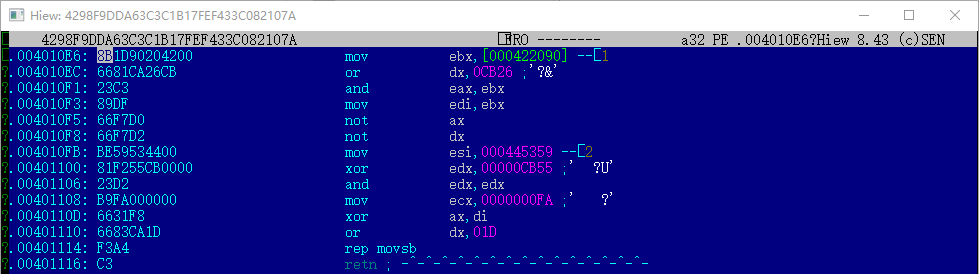
这里可以看到，程序在获取了自身模块句柄之后，将返回值（保存在了eax中）赋给了一个地址，然后就是一个call。这里跟随这个call看看它做了什么：



首先跳转到第一个call里面看一下内容：



可以看到，这个call的目的就是利用VirtualAlloc函数来分配内存空间。有理由相信，解密后的代码，很可能就会保存在这里。返回上一级，查看一下第二个call的内容：



可以发现，这里又是利用and、not以及xor等运算符做了一些解密运算，这也是不应该出现在一个正常程序中的，所以就可以直接把它报毒了：Trojan.Win32.Agent.c。

其实这是病毒保护自己的一种手段，叫做混淆，或者也可以理解为是病毒作者为自己的恶意程序所编写的一种壳。我会在未来的进阶篇中，为大家详细分析一下这类恶意程序。

**小结**

通过本次课程对PE文件的快速分析，大家可以发现快速分析技术是需要建立在分析过大量样本的前提下，才能够积累起足够的经验的。我们最后讨论的那两个样本，可以说是我们日常分析工作中，在仅仅利用Hiew的前提下，分析得最麻烦的恶意程序了。在这里也希望大家多多积累经验，多多分析，能够把这几次课程中的知识点全都掌握下来。