

**恶意代码课程实验报告**

**实验十四：恶意代码的网络行为**

****

学 院 网络安全学院

专 业 信息安全

学 号 2110688

姓 名 史文天

班 级 1063

1. **实验目的**

**完成课本Lab14的实验内容。**

1. **实验原理**

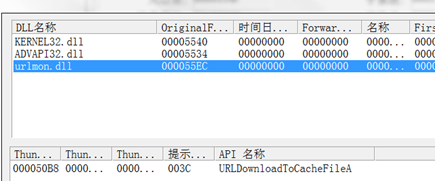
**配置XP虚拟机环境，安装动态、静态分析工具，使用静态和动态分析工具。**

1. **实验过程**

**Lab14-01**

**静态分析：**

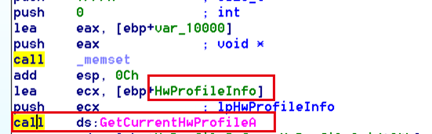
**查壳——无壳  
**

**查看输入表——  
Kernel32.dll：睡眠，创建终止进程，加载库文件等函数  
ADVAPI32.dll：用于检索或设置系统信息，获取当前用户名等函数  
Urlmon.dll：url下载文件到缓存函数  
**

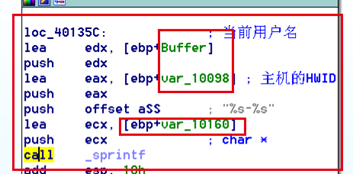
**Strings 分析  
出现base64编码格式字符串，可能使用base64加密  
在这里插入图片描述  
出现网络资源，说明恶意程序可能会联网请求网络资源  
在这里插入图片描述**

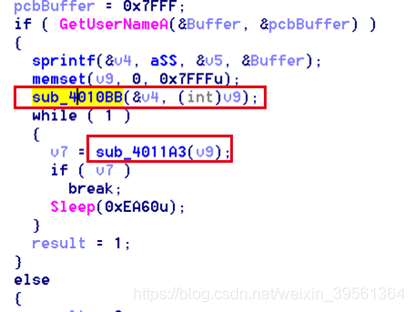
**先进行初步的动态分析**

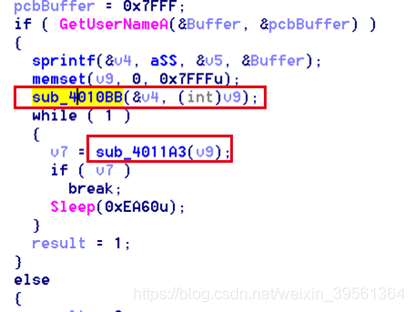
**IDA分析**

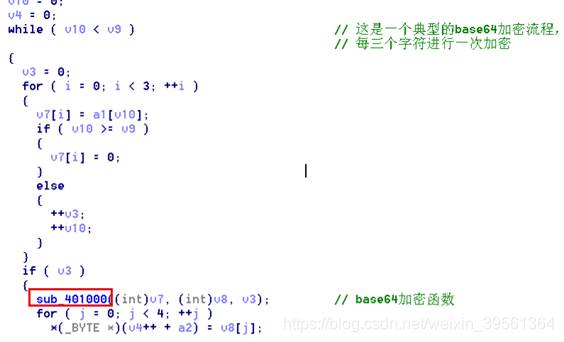
**首先获取当前主机的HWID（Hardware Identification硬件身份识别编码）  
**

**将获取到的HWID按照指定方式进行组成字符串，并且保存在变量var\_10098中  
**

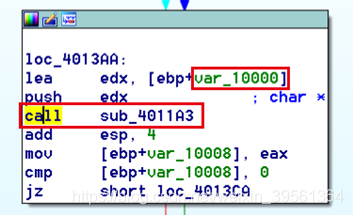
**将获取到的HWID和用户名进行拼接，保存在变量var\_10160中  
**

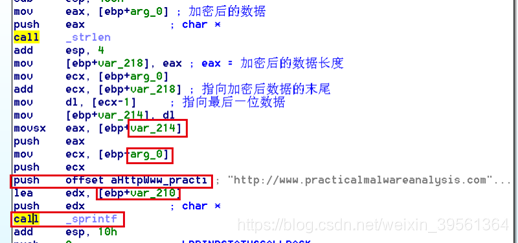
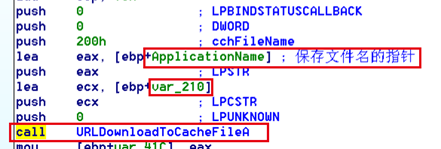
**继续往下，函数会先调用函数sub\_4010BB，然后进入一个死循环，其中的循环体调用函数sub\_4011A3  
**

**先分析sub\_4010BB，该函数的参数正是HWID和用户名进行拼接得到的字符串，和一个申请的内存首地址。  
**

**进入该函数进行分析，发现该函数是一个典型的base64加密函数，其中加密的对象就是传入的参数字符串。  
**

**因此，函数sub\_4010BB就是一个base64加密函数，加密后的数据保存在变量var\_10000中。**

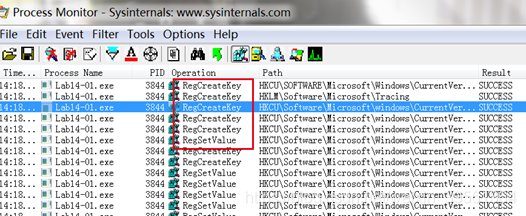
**继续往下，循环体中调用函数sub\_4011A3，其中的参数正是加密后的字符串。  
**

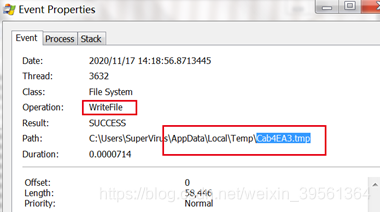
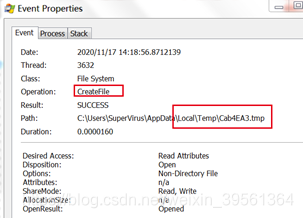
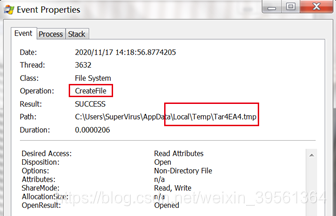
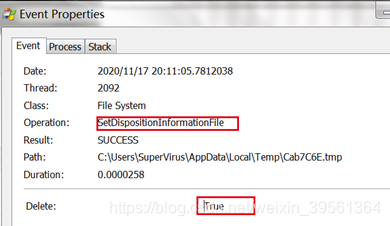
**进入该函数进行分析，  
(1) 该函数在开始进行字符串拼接，将加密后的字符、它的最后一位和域名字符串进行拼接得到一个网络资源地址，拼接后的字符串保存在var\_210中。  
在这里插入图片描述  
  
(2) 然后会调用函数URLDownloadToCacheFileA请求这个网络资源，并且进行缓存，其中缓存的文件名保存在ApplicationName变量中。  
  
(3) 接下来会创建进行执行该缓存的文件  
**

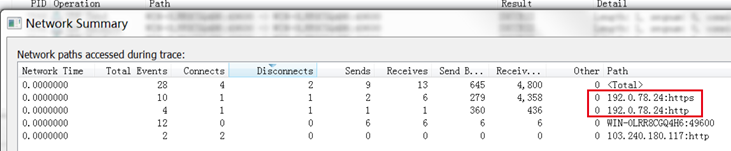
**到此，整个恶意程序的分析就完成了，现在进行恶意程序功能总结  
(1) 获取HWID和用户名，并且进行拼接  
(2) 对拼接后的字符串进行base64加密  
(3) 将加密后的字符串和指定字符串拼接得到网络资源地址  
(4) 访问网络资源地址对资源进行缓存，并且创建新的进程执行缓存文件**

**动态分析**

**打开PM，运行程序进行监控**

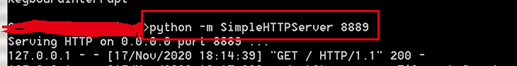
**分析注册表操作  
修改了网络连接下的多个注册表项  
**

**分析文件操作  
(1) Temp临时文件夹下创建并修改了Cab4EA3.tmp、Tar4EA4.tmp文件  
  
  
(2) 但是这些文件会被删除  
  
(3) 这些临时文件是在进行网络访问时产生的临时文件，在程序运行结束时会被删除。**

**网络操作分析，与192.0.78.24进行网络连接  
**

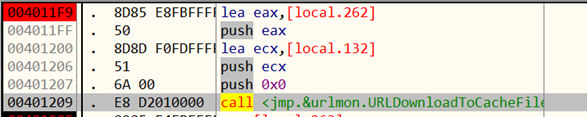
**搭建HTTP服务进行动态分析**

**在本地搭建环境进行测；进行HTTP服务搭建来进行样本分析，是因为对于函数URLDownloadToCacheFileA对网络资源进行缓存时的目录有疑惑。**

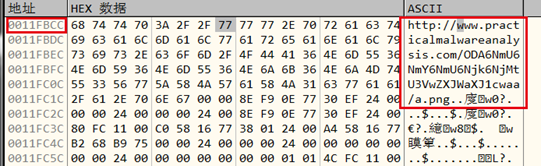
**利用python 创建一个简单的http服务，  
**

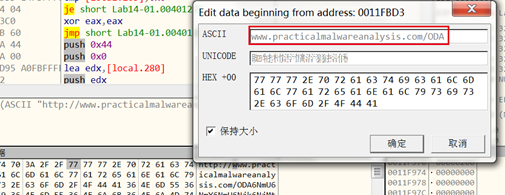
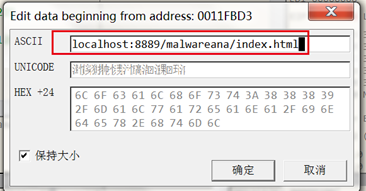
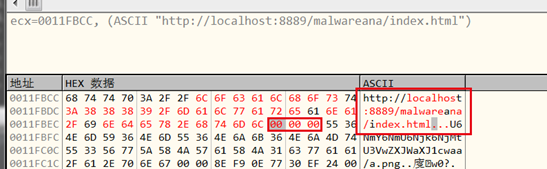
**在当前用户的根目录下创建目录和文件malwareana/index.html  
**

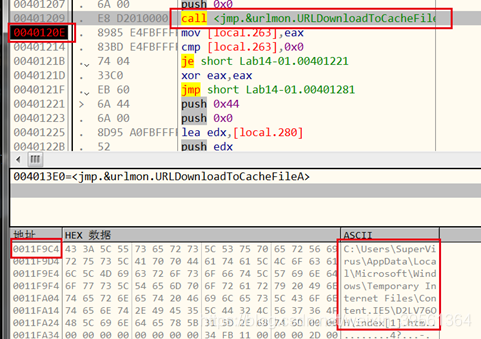
**然后打开PM进行监控**

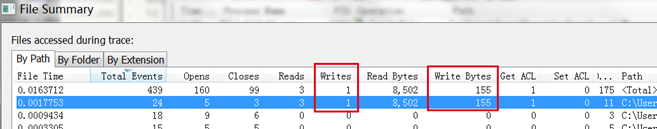
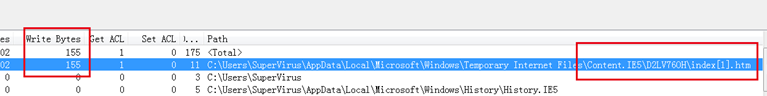
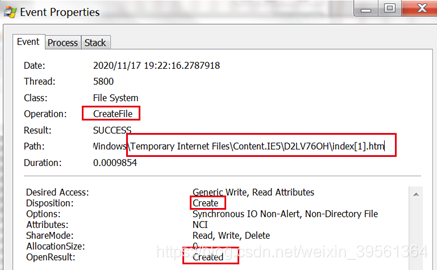
**用OD打开程序，并且在004011F9和0040120E处下断点，因为这两个地址间正是函数URLDownloadCacheFileA被调用的时刻，我们需要在此处修改网络资源的地址；同时记录下URLDownloadCacheFileA函数的其中一个参数——缓存目录ApplicationName (地址未0x0011F9C4)  
**

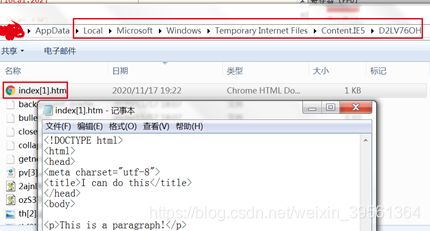
**F8单步走，出现网络资源地址  
**

**跟踪到存放网络资源内存地址  
**

**在数据窗口中修改网络资源地址  
此处修改后的地址是http://localhost:8889/malwareana/index.html (此处一定要把html后的十六进制数据改为0x00，作为字符串截断)  
  
  
修改内存内容后需要修改后面的数据为0x00h，用于字符串截断  
**

**然后在OD中继续运行程序（F9），此时程序在0040120E处断下，在数据窗口中跟踪缓存目录ApplicationName的地址——0x0011F9C4，获取到系统的缓存网络资源的目录  
**

**PM已经监控到相关的文件操作，发现创建文件index[1].html (因为缓存目录中已经存在index.html)  
  
  
**

**跟到缓存目录下，找到缓存文件，正是本地http服务端提供的文件  
**

**问题**

**恶意代码使用了哪些网络库？它们的优势是什么？  
回答：使用了URLDownloadToCacheFileA函数，该函数使用了COM接口，当恶意代码使用COM接口时，HTTP请求中的大部分内容都来自Windows内部，无法有效地使用网络特征进行针对性的检测。**

**用于构建网络信令的信息源元素是什么，什么样的条件会引起信令的改变？  
回答：发送的信息中包含了主机的HWID和用户名；其中HWID是固定的，用户名则是可变的。**

**为甚攻击者可能对嵌入在网络信令中的信息感兴趣？  
回答：攻击者想获取特定的用户和主机，在下一步进行针对性地攻击。**

**恶意代码是否使用了标准的Base64编码？如果不是，编码是如何不寻常的？  
回答：不是常用的Base64编码，因为常见的base64编码在长度不够时采用“=”进行末尾填充，而此恶意程序采用‘a’进行末尾填充**

**恶意代码的 主要目的是什么？  
回答：将被感染主机的HWID和用户名发送到到C&C，然后从网站下载并运行其他恶意代码**

**使用网络特征可能有效探测到恶意代码通信中的什么元素？  
回答：域名特征、base64编码特征和HTTP请求中的URI的后缀是PNG文件，**

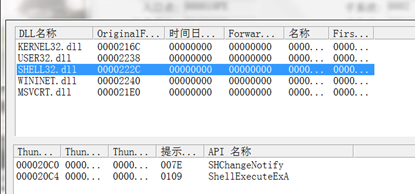
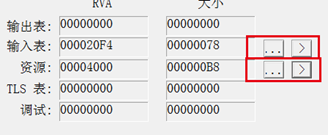
**分析者尝试为这个恶意代码开发一个特征时，可能会犯什么错误？  
回答：——**

**哪些特征集可能检测到这个恶意代码（以及新的变种）？  
回答：——**

**Labs-14-02**

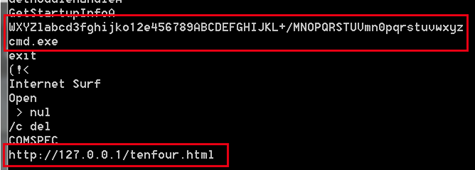
**静态分析：**

**查壳——无壳  
**

**查看输入表——发现其他资源  
KERNEL32.dll：创建终止进程、复制句柄、创建管道、独写文件等函数  
USER32.dll：加载字符串  
SHELL32.dll：执行shell  
WININET.dll：打开URL，网络读取文件  
MSVCRT.dll：——  
  
**

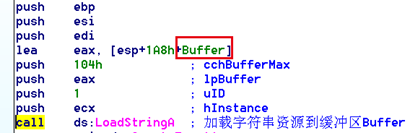
**查看隐藏的资源——网络资源地址  
**

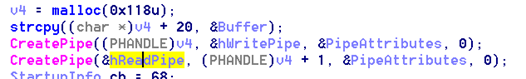
**Strings分析**

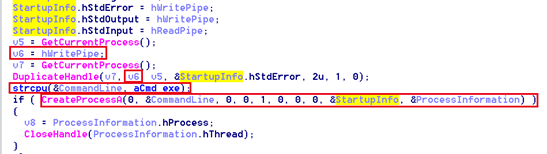
**出现常见的base64加密使用的字符串，推测可能采用base64进行加密  
出现“cmd.exe”，推测可能会调用cmd执行恶意命令  
出现“http:…127.0.0.1/tenfour.html”，推测会访问网络资源  
**

**先进行初步的动态分析**

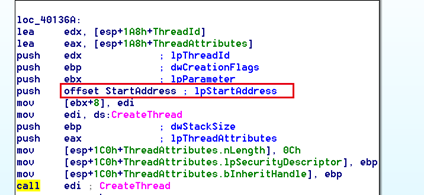
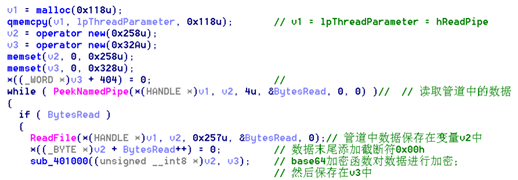
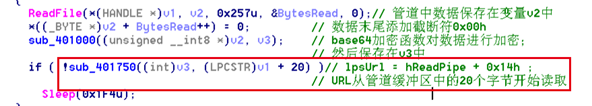
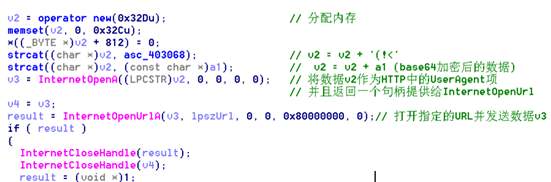
**IDA分析**

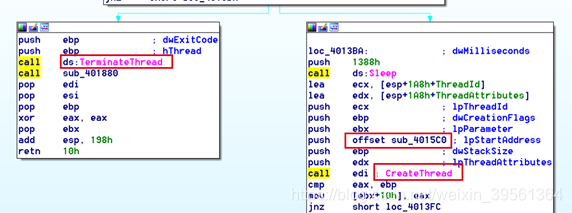
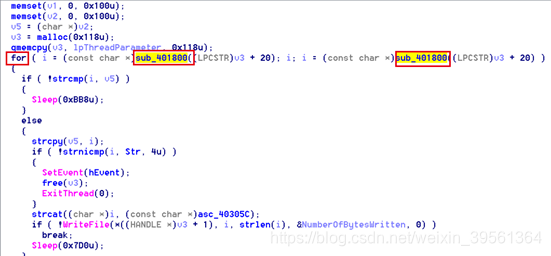
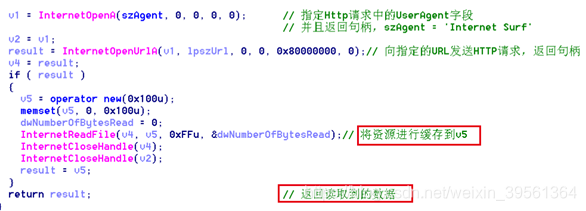
**Main函数首先加载资源中的字符串（前面动态分析已经知道字符串是一个网络资源地址），然后保存到Buffer缓冲区中  
**

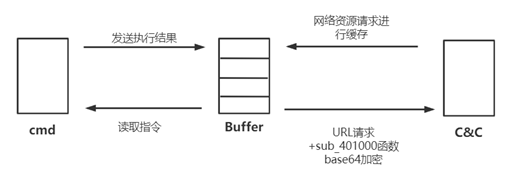
**然后将字符串赋值到变量v4中，同时创建两个管道，同时将变量v4分别作为输入和输出  
**

**管道的另一端则是进程信息StartupInfo结构特，通过创建一个新的cmd进程来实现管道通信  
**

**也就是说cmd通过管道接收缓冲区Buffer的数据进行执行，然后将执行的结果通过管道传输到缓冲区中**

**接下来会创建一个线程，线程的地址是StartAddress  
  
(1) 跟入该地址，该模块会读取管道的数据，然后利用函数sub\_401000进行base64加密，并将加密后的数据保存在v3中  
  
(2) 接下来会调用函数sub\_401750对加密后的数据进行操作，该函数拥有两个参数，一个是加密后的数据，另一个是追溯回去正是缓冲区原始的数据，但是存在0x14字节的偏移（也就是网络资源的字符串）  
  
(3) 进入sub\_401750函数进行分析，该函数将加密后的数据作为HTTP协议中的UserAgent字段，然后访问指定的URL  
**

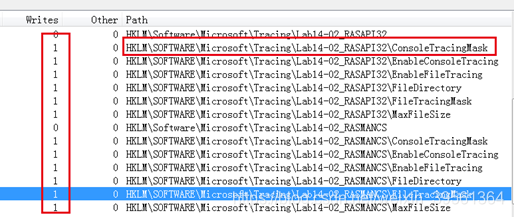
**继续往下，整个逻辑进行分支，一个分支终止进程(TerminateThread)，另一分支创建另一个线程，线程起始地址是sub\_4015C0  
  
(1) 进入该地址进行分析，该地址存在一个for循环，循环体中不断利用函数sub\_401800进行网络访问  
  
(2) Sub\_401800的参数和前面分析的sub\_401750函数的一样，同样是是从管道中进行读取，并且读取的偏移都是0x14h (十进制20)——指定的URL，可以猜测这两个函数在功能上应该有相似之处，跟进该函数进行分析  
  
(3) 循环体中除了sub\_401800函数进行操作外，还对网络数据进行了缓存。  
**

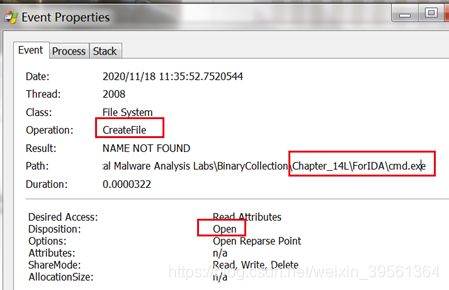
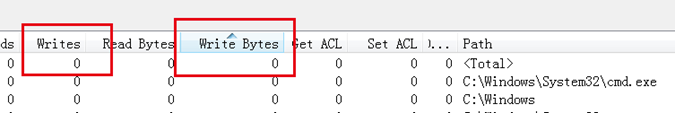
**到此，整个恶意程序的分析基本结束了，关于整个恶意程序创建的管道通信模型可以简化为：  
**

**现在对整个恶意程序的功能进行总结  
（1） 加载字符串资源  
（2） 创建管道进程间进行通信  
（3） 创建新进程执行cmd.exe  
（4） 创建线程进行数据加密，然后发送网络数据请求  
（5） 创建新线程发送另一网络数据请求，并且将请求的资源进行缓存**

**动态分析**

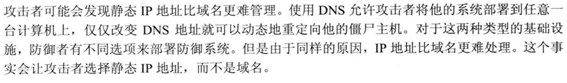
**打开PM进行监控，运行程序**

**分析注册表操作，发现创建多个注册表  
**

**分析文件操作  
打开了cmd.exe  
  
并无写文件操作  
**

**网络操作分析——向本地127.0.0.1发送请求**

**问题**

**恶意代码编写时直接使用IP地址的好处和坏处各是什么？  
回答：   
**

**恶意代码使用了哪些网络库？使用这些库的好处和坏处是什么？  
回答：使用了WinNet库。优点是与Winsock API相比，操作系统可以提供较多的网络字段元素；缺点就是该库的网络函数调用需要提供一个硬编码的UserAgent字段。**

**恶意代码信令中URL的信息源是什么？这个信息源提供了哪些优势？  
在这里插入图片描述**

**恶意代码利用了HTTP协议的哪个方面，来完成它的目的？  
回答：利用UserAgent字段来发送信息，并且该信息被加密；通过两个管道通信来实现远程控制和任意命令执行。**

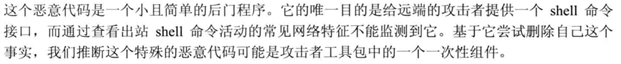
**在恶意代码的初始信令中传输的是哪种信息？  
回答：打开cmd返回的信息，但是被加密。**

**这个恶意代码通信信道的设计存在什么缺点？  
在这里插入图片描述**

**恶意代码的编码方案是标准的吗？**

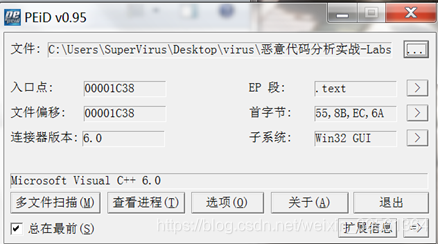
**是自定义的base64编码。**

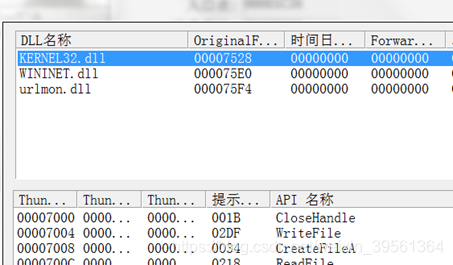
**通信是如何被终止的？使用关键字exit来终止通信，退出时恶意代码会试图删除自己**

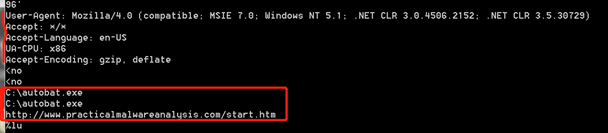
**这个恶意代码的目的是什么？在攻击者的工具中，它可能会起什么作用？  
**

**Labs-14-03**

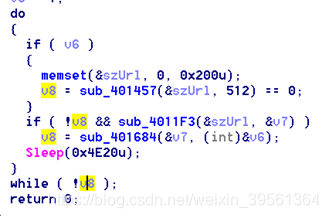
**静态分析：**

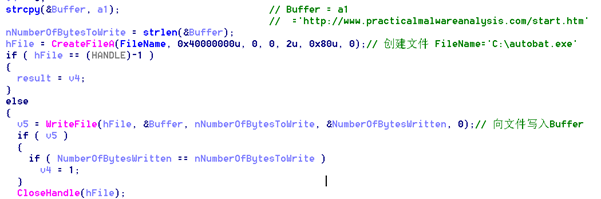
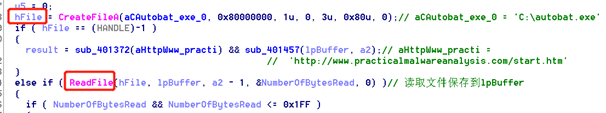
**查壳——无壳  
**

**分析输入表——  
KERNEL32.DLL：读写，创建文件，创建文件，加载库文件等  
WININET.DLL：URL访问，网络文件读取等  
urlmon.dll：URLDownloadtoCacheFileA函数下载文件到缓存  
**

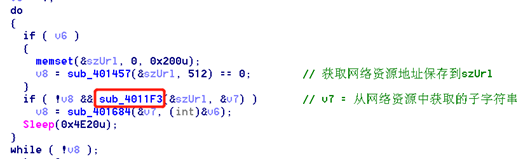
**String分析  
出现base64相似的编码字符串，猜测可能使用base64进行数据加密  
在这里插入图片描述  
出现HTTP协议相关字段，猜测会进行网络请求；  
出现exe程序，可能会执行  
出现网络资源，可能会请求网络资源  
**

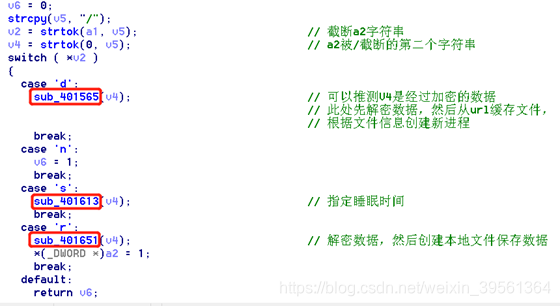
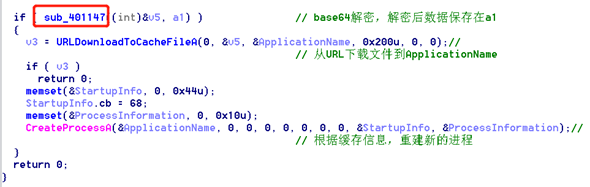
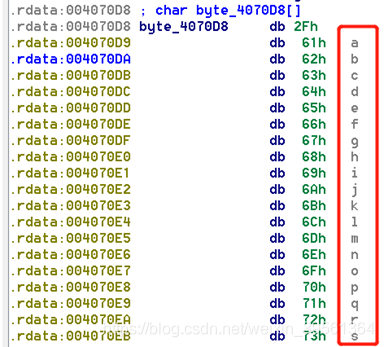
**IDA分析**

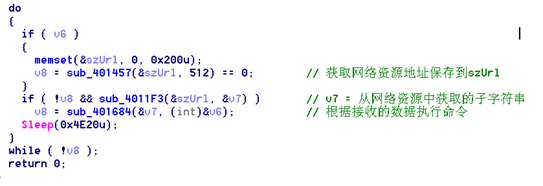
**Main函数的逻辑比较简答，主要是一个do while循环：  
**

**对循环体进行分析，首先调用了函数sub\_401457，该函数由两个参数，一个是字符串szUrl，另一个是数字；跟进该函数进行分析  
(1) 该函数内部首先调用CreateFileA函数打开文件’C:\autobat.exe’ ，然后调用sub\_401372  
  
(2) 进入函数sub\_401372，该函数的参数是字符串’http://www.practicalmalwareanalysis.com/start.htm’ ，一个网络资源；进入该函数发现该函数将创建文件’C:\autobat.exe’，并且将字符串’http://www.practicalmalwareanalysis.com/start.htm’ 写入到文件中。  
  
(3) 回到函数sub\_401457，继续往下，调用函数ReadFile将文件中的数据读取到缓冲区lpBuffer中；此处的文件正是前面创建的’C:\autobat.exe’，它内部存储的数据正是网络资源字符串’http://www.practicalmalwareanalysis.com/start.htm’。  
**

**因此函数sub\_401457的功能就是在本地创建文件保存网络资源地址**

**回到循环体，继续往下，调用函数sub\_4011F3对网络资源进行操作，进入该函数，  
  
(1) 该函数首先初始化HTTP字段元素，然后根据传入的URL进行网络资源访问  
(2) 从网络资源读取数据后保存到Buffer中，然后匹配‘<no’字符串  
  
(3) 然后调用sub\_401000进行进一步匹配  
  
(4) 返回从网络中获取的满足匹配条件的字符串**

**继续往下，将获取的字符串传入函数sub\_401684  
  
(1) 进入函数sub\_401684进行分析，发现该函数先将字符串进行截断，然后进行一个switch跳转，跳转的条件就是字符串的首字母，那么可疑猜测这就是根据C&C发送的指令执行不同的操作；每个命令的方式都已经分析如下。  
  
(2) 需要注意的是，其中的加密函数部分都调用函数sub\_401147  
  
(3) 该函数与传统的base64并不相同，它采用了自定义的编码字符串  
**

**到此，整个恶意程序的分析就完成了  
**

**现在对恶意程序的功能进行总结  
（1） 创建本地配置文件  
（2） 根据配置文件发送网络请求，其中的HTTP部分元素已经被硬编码  
（3） 将网络请求获取的数据进行缓存，并且进行字符串检查  
（4） 对返回的数据进行解密，不同的指令执行不同的操作，其中包括睡眠sleep，重定向redirect（修改本地配置文件），下载并执行其他恶意文件download，不执行任何操作nop。**

**问题**

**1.在初始信令中硬编码元素是什么？什么元素能够用于创建一个好的网络特征？**

**2.初始信令中的什么元素可能不利于可持久使用的网络特征？**

**3.恶意代码是如何获得命令的？本章中的什么例子用来类似的方法？这种技术的有点是什么？**

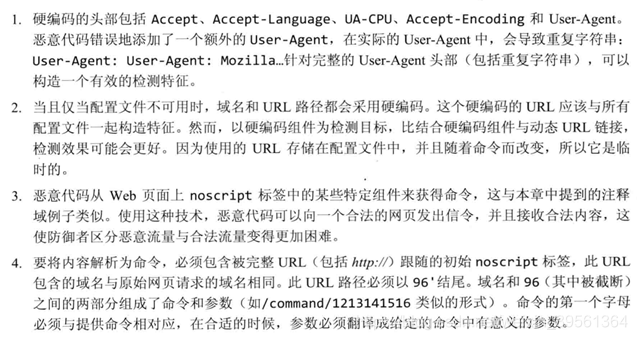
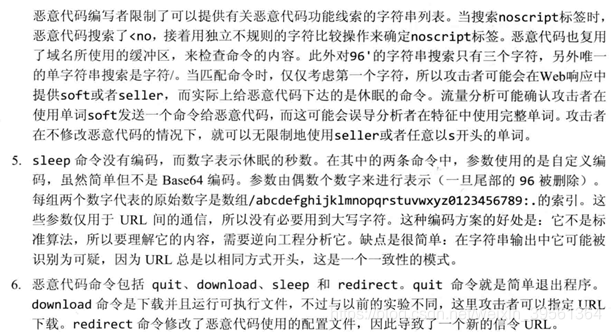
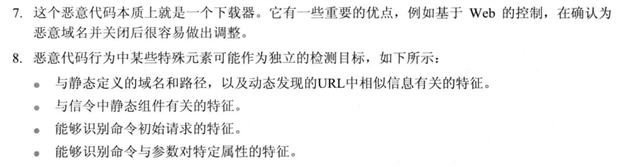
**4.当恶意代码接受到输入时，在输入上执行什么检查可以决定它是否是一个有用的命令？攻击者如何隐藏恶意代码正在寻找的命令列表？**

**什么类型的编码用于命令参数？它与Base64编码有什么不同？它提供的有点和缺点各是什么？**

**5.这个恶意代码会接受哪些命令？**

**6.这个恶意代码的目的是什么？**

**7.本章介绍了用独立的特征，来针对不同位置代码的想法，以增加网络特征的鲁棒性。那么在这个恶意代码中，可以针对哪些区段的代码，或是配置文件，来提取网络特征？**

**8.什么样的网络特征集应该被用于检测恶意代码？  
  
  
**

1. **实验结论及心得体会**
2. **对恶意代码网络行为相关的知识有了更深入、直观的了解，对恶意代码利用网络的目的等有了明确的认识。**
3. **了解了更多Windows恶意代码的形式和特征。**
4. **对恶意代码各种分析工具的使用，各种原理的运用有了更加熟练的认识和掌握。**