

**恶意代码分析与防治课程实验报告**

**实验十四：恶意代码的网络特征**

****

学 院 网络空间安全学院

专 业 信息安全

学 号 2111033

姓 名 艾明旭

班 级 信息安全一班

1. **实验目的**

**恶意代码的网络特征是指恶意软件在网络中传播和执行时所展现出的特定行为和模式。通过实验分析恶意代码的网络特征有几个目的：**

**识别恶意行为模式： 实验可以帮助识别恶意代码在网络中的行为模式，例如它们的传播方式、与服务器的通信模式、数据包的格式等。这种分析有助于了解恶意软件的传播机制和攻击方式。**

**建立特征库： 通过实验收集恶意代码的网络特征，可以建立特征库或签名数据库，用于检测和识别未来类似的恶意行为。这有助于安全团队及时发现和应对新型威胁。**

**检测与防范： 通过分析网络特征，可以开发出检测和阻止恶意代码传播的方法和工具。这种分析有助于网络安全专家设计更有效的防御策略，减少恶意代码造成的损害。**

**模拟攻击与响应： 实验也可用于模拟攻击和相应的应对。这种模拟有助于测试安全措施的有效性，评估系统在面对实际恶意攻击时的应变能力。**

**总之，通过实验分析恶意代码的网络特征，可以更好地理解和应对恶意软件对网络安全的威胁，提高系统的安全性和应对能力。**

1. **实验原理**

**捕获恶意代码样本： 首先，需要获取包含恶意代码的样本。这可以通过网络流量捕获、恶意软件沙箱分析、恶意软件样本库等方式获得。捕获的样本应包含足够的信息，使得能够对其在网络中的行为进行详细分析。**

**动态分析： 将恶意代码样本在一个受控环境中执行，监视其行为。这可以在沙箱环境中进行，以模拟真实网络环境。动态分析包括监控文件系统、注册表、进程、网络通信等活动。**

**网络流量分析： 重点关注与网络通信相关的行为。监控数据包的发送和接收，分析通信的目标、协议、数据包结构等。可以使用网络协议分析工具，例如Wireshark，对抓取到的数据包进行解析和分析。**

**行为特征提取： 基于实验中观察到的网络行为，提取特定的网络特征。这可能包括特定的网络协议、目标IP地址、使用的端口号、数据包的结构等。这些特征可以成为后续检测和防范的依据。**

**建立模型和规则： 基于实验结果，可以建立模型或规则集，用于检测类似的恶意行为。这可能涉及到机器学习模型的训练，也可以是基于规则的逻辑判定。**

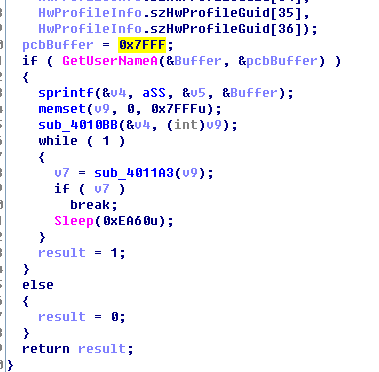
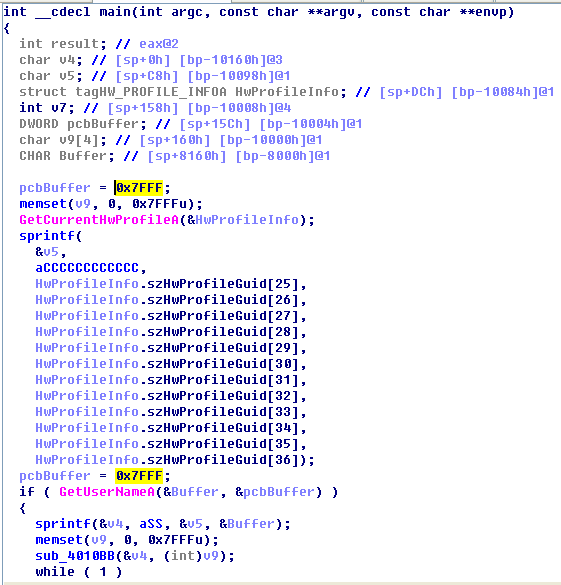
**特征库更新： 将从实验中提取的特征信息整合到特征库或签名数据库中。这有助于实时检测和防范未来的威胁。**

**评估与改进： 对建立的检测模型和规则进行评估，看其在真实环境中的性能。根据实际运行中的表现，不断改进和优化分析方法和特征库。**

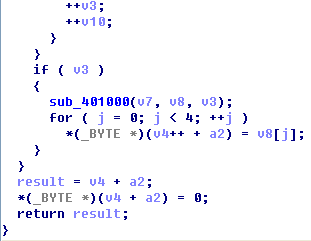
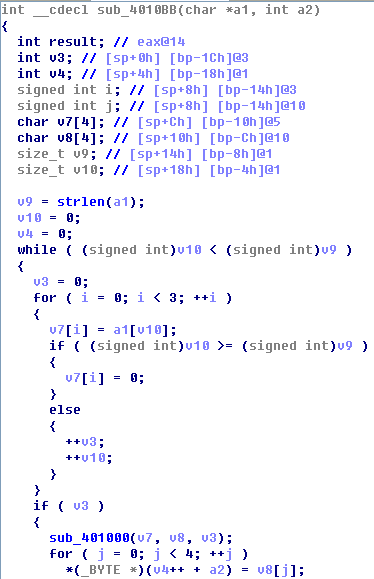
1. **实验过程**

**LAB14-01.exe**

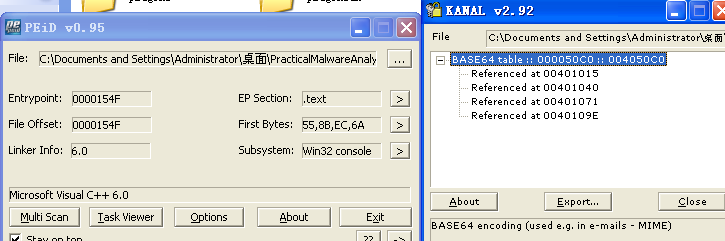
先反编译看看：



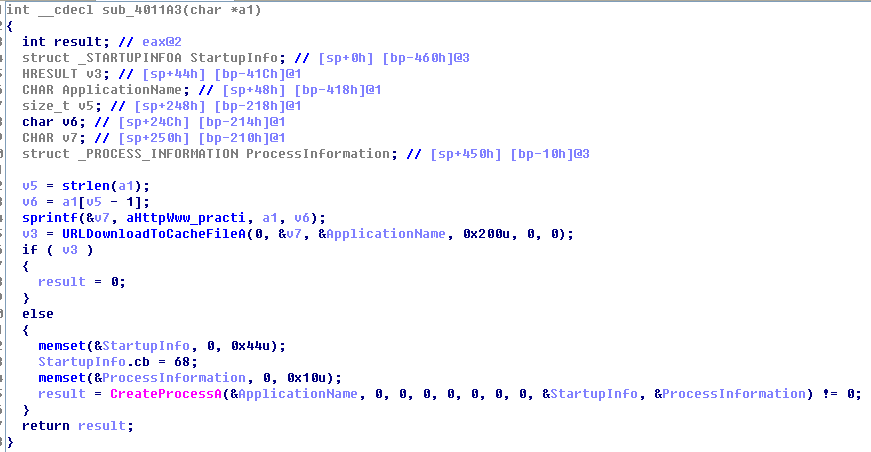
函数在做base64加密：



接下来在PEID的kanal v2.92模块当中进行验证：

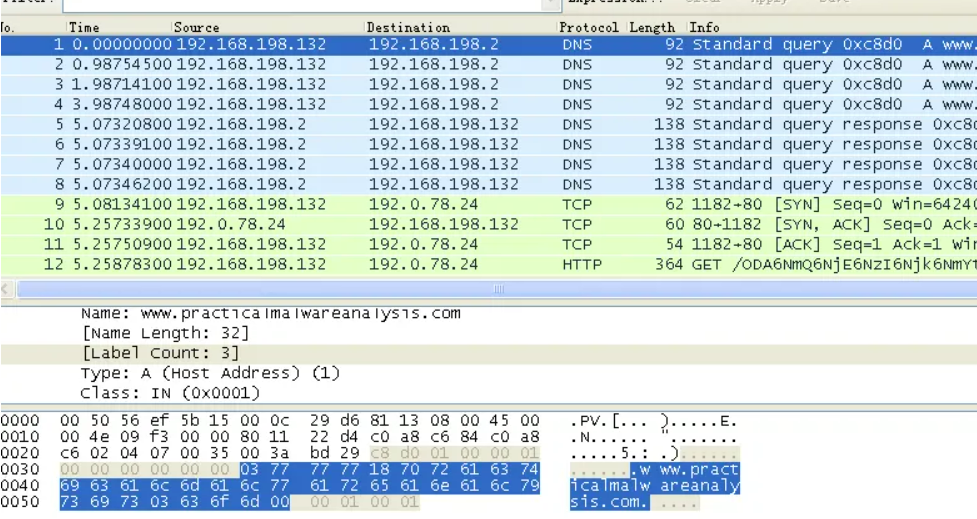


后面的功能，就是在下载执行了：

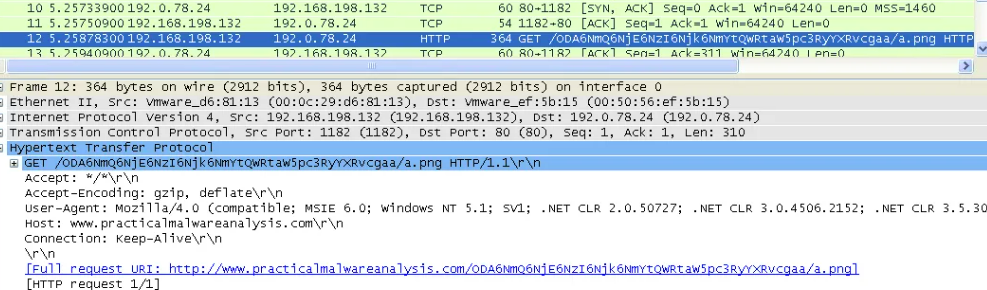


问题1：

使用 wireshark进行监控网络特征，运行试验程序Lab14-01.exe：



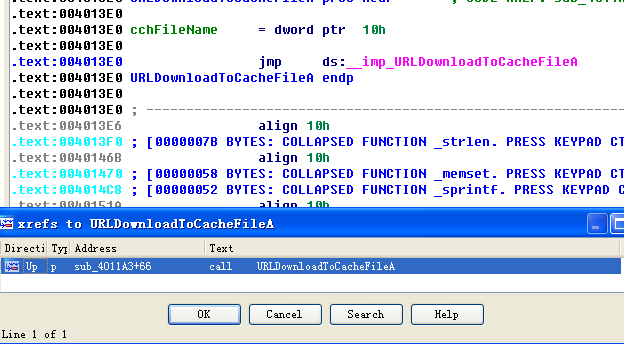
可以看到一开始就发送了一个DNS请求，目标地址就是一个网址。接着可以看到：



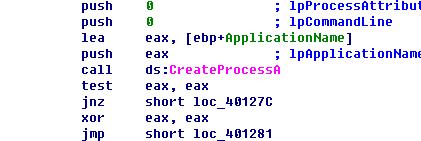
发现一个HTTP的GET请求，可以看到 字符串user-Agent并不是硬编码的。==》http c2

问题2：

使用ida打开文件Lab14-01.exe，看到函数URLDownloadToCacheFileA，这个函数的作用就是将文件下载到本地的Internet中，并且使用COM接口，这时HTTP的大部分内容都来自Windows内部，这样也就没有办法使用网络特征来进行针对性的检测。对函数URLDownloadToCacheFileA使用交叉引用，发现是sub\_4011A3调用了它：



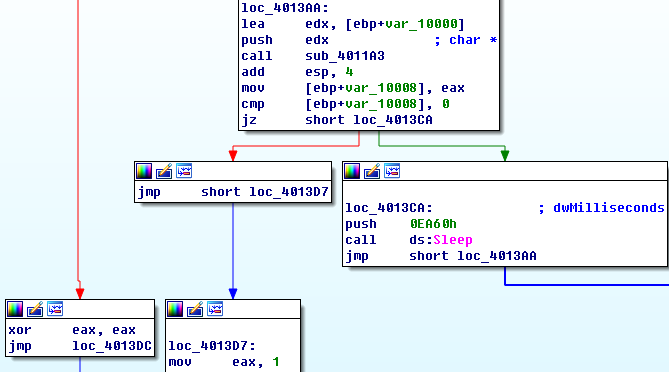
在这里发现调用了函数CreateProcessA，向上看发现了字符串'http://www.practicalmalwareanalysis.com/%s/%c.png'也就是函数URLDownloadToCacheFileA的参数，看到了/%s/%c，可以查找一下输入点在什么位置：



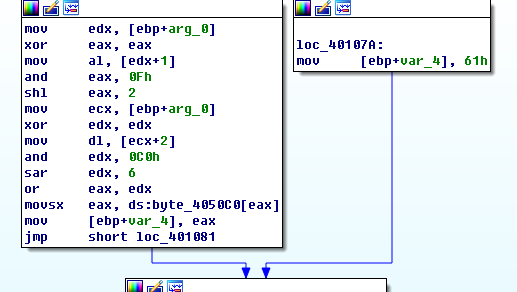


通过这段代码分析就可以知道aHttpWww\_practi字符串上方push入栈的就是%c的内容，也就是函数sub\_4011A3的参数。继续上方紧挨着的就是%s的内容。

下面分析一下这个参数arg\_0，交叉引用发现在main函数中被调用，

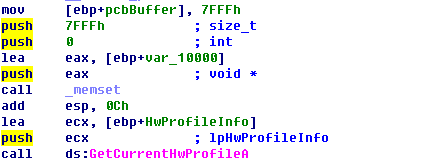


函数sub\_4011A3的参数有由函数sub\_4010BB提供，进入这个函数发现是在计算参数字符串的长度，下面调用了函数sub\_401000，进入发现通过byte\_4050C0发现了是在引用base64编码字符串。



看到了这里并没有使用标准的填充字符“=”，而是使用了小写字母“a”。

再回到main函数：



发现调用了函数GetCurrentHwProfileA，这个函数可以返回GUID也就是全局唯一标识符中的6个字节，并以MAC地址的格式打印出来，最后成为"%s-%s"中的第一个字符串，第二个字符则是用户名。

我们可以使用Base64工具来解析一下在最开始使用wireshark得到的网络数据内容ODA6NmU6NmY6NmU6Njk6NjMtsmlhbmca，解码后的结果是80:6e:6f:6e:69:63-瞚ang。

分析到这里也就确定了信令的信息来源。信息源的元素是主机GUID全局唯一标识符与用户名的一部分。GUID对于任何主机操作系统都是惟一的，信令中使用了GUID中的6个字节，也是相对唯一的。用户名会根据登录系统的用户而改变。

问题3：

攻击者可能想要跟踪运行恶意程序的主机来针对某个用户。

问题4：

是Base64编码，并不是标准的，因为使用a代替了=来作为填充符号。

问题5：

下载恶意文件并运行它。

问题6：

域名，冒号，解码后的破折号。

问题7：

分析者如果没有想到操作系统会决定恶意程序的特征元素，可能会尝试将URI的内容作为分析的重点目标。

问题8：

应该就是解码之后出现的冒号和破折号。

**LAB14-01.exe**

捕获到的数据，如下：

第一条信令：

GET /tenfour.html HTTP/1.1

User-Agent:(!<e6LJC+xnBq90daDNB+1TDrhG6aWG6p9LC/iNBqsGi2sVgJdqhZXDZoMMomKGoqx

UE73N9qH0dZltjZ4RhJWUh2XiA6imBriT9/oGoqxmCYsiYG0fonNC1bxJD6pLB/1ndbaS9YXe9710A

6t/CpVpCq5m7l1LCqR0BrWy  就是http c2内容了

Host:127.0.0.1

Cache-Control:no-cache

第二条信令：

GET /tenfour.html HTTP/1.1

User-Agent:Internet Surf

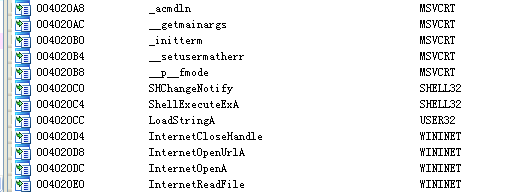
Host:127.0.0.1

Cache-Control:no-cache

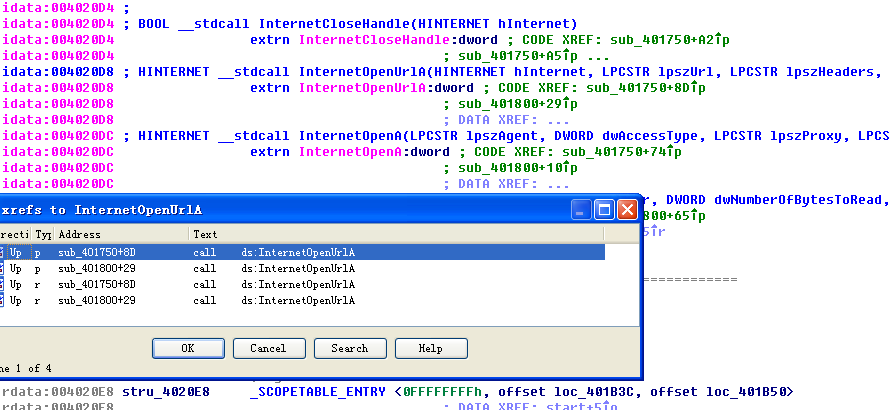
问题1：

静态IP地址比域名更加难以管理。所以攻击者会更多的选择静态IP地址，而不是域名。而使用域名攻击容许将他的恶意程序部署到任意一台主机上。

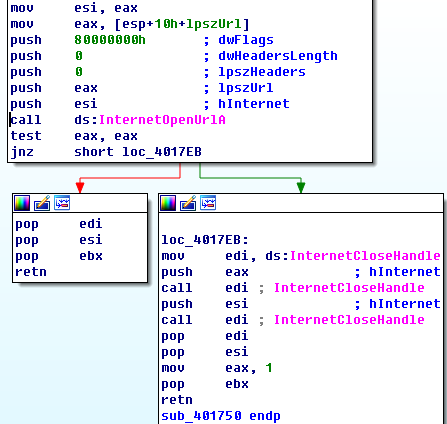
问题2：



使用ida来查看一下导入函数：可以看到从WINNET库中导入了四个API函数，双击Internetopenurla,利用交叉引用发现：

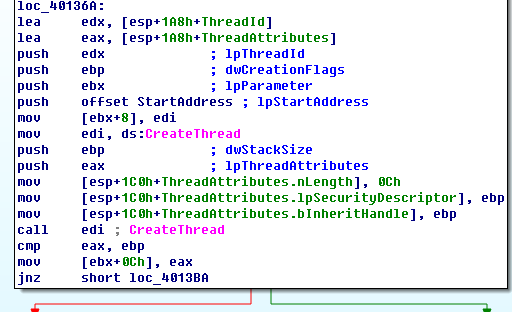


参数dwflags的值是80000000h，查阅资料可知表示INTERNET\_FLAG\_RELOAD。当这个标志被置位的时候，它会在信令中产生Cache-Control:no-cache字符串，因此会让它更容易地被识别为恶意程序。

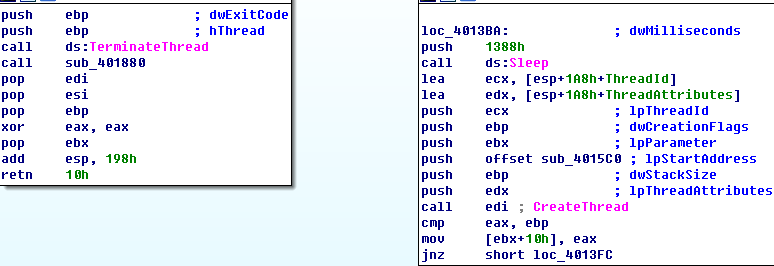


使用WINNET库的缺点就是需要提供一个硬编码的User-Agent字段。

问题3：下面来看一下main函数，

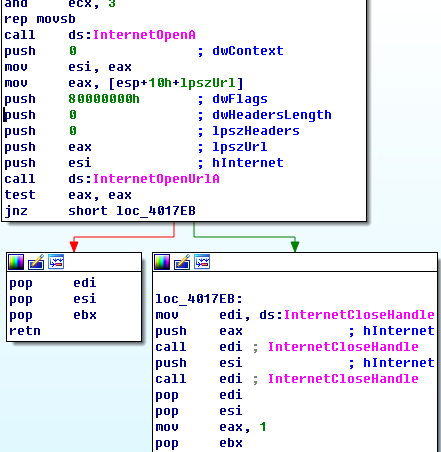


调用了函数CreateThread，用于创建一个线程，参数lpStartAddress被标记为了StartAddress，也就是线程开始的地址，向下看：



又调用了函数CreateThread，这个参数lpStartAddress被标记为了sub\_4015C0，可以将其重命名为S\_A\_thread2，表示第二个线程的开始地址。

先查看一下StartAddress，然后又调用了函数sub\_401750，在这个函数里面发现调用了InternetOpenA，InternetOpenUrlA，InternetCloseHandle这几个函数，可以将sub\_401750重命名为Internet1。接着看一下函数S\_A\_thread2，发现调用了函数sub\_401800，在这里面看到了函数InternetOpenA，InternetOpenUrlA，InternetReadFile， InternetCloseHandle，那么可以将sub\_401800重命名为Internet2。

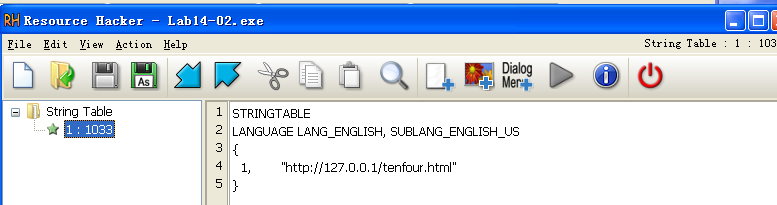
返回main函数，可以发现在在启动这两个线程之前，调用了CreatePipe，GetCurrentProcess，DuplicateHandle，CreateProcessA这几个函数，其中CreateProcessA函数用于创建一个新的cmd.exe进程。

分析到这里可以推测恶意程序的编写者是想建立一个反向的shell。

这里回到函数Internet1：

发现函数InternetOpenUrlA的一个参数是lpszUrl

同时也发现lpszUrl是Internet1的参数，因为Internet1属于StartAddress，而StartAddress的参数又取决于lpParameter，而它的内容又由LoadStringA决定。这个函数的目的就是从资源节中读取字符串。下面使用Resource Hacker来查看一下：



可以看到，这里有一个用于信令的URL。这样就可以知道，PE文件中的字符串资源节包含一个用于命令和控制的URL。在不重新编译恶意程序的情况下，可以让攻击者使用资源节来部署多个后门程序到多个命令与控制服务器的位置。

问题4：

我们现在已经知道Internet1中的一个参数是URL，那么可以知道另一个参数就是User-Agent域，看一下Internet2，和Internet1一样使用了相同的URL，User-Agent字符串被静态地定义为了“Internet Surf”。 通过分析可以知道，攻击者滥用了HTTP的User-Agent域。恶意程序创建了一个线程，来对这个域传出的信息进行编码，以及另一个线程，使用静态域表示它是通道的接收端。

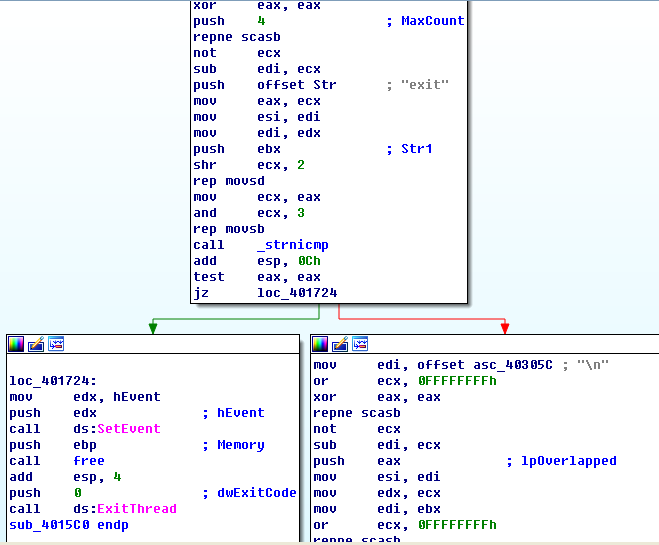
问题5：初始信息是编码后的shell提示。

问题6：攻击者只对传出的信息进行了编码，并没有对传入的命令进行编码。还可以知道服务器的依赖关系十分明显，可以将它作为特征生成的目标元素。

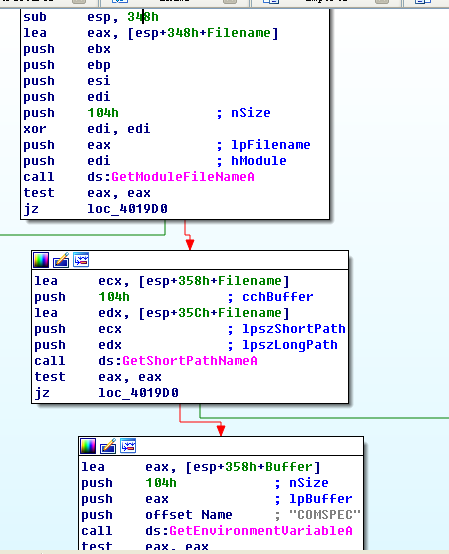
问题7：

使用的是Base64编码，但并不是标准的，使用的是一个自定义的Base64编码。

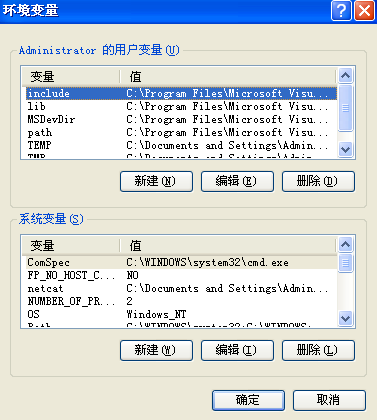
问题8：



在S\_A\_thread2中可以看到，如果接受到了exit，就会调用函数ExitThread来退出当前的线程。返回之后可以看到程序会调用函数sub\_401880：



可以看到这段代码使用了COMSPEC的方法来达到自我删除的目的。



就是运行cmd /c del！

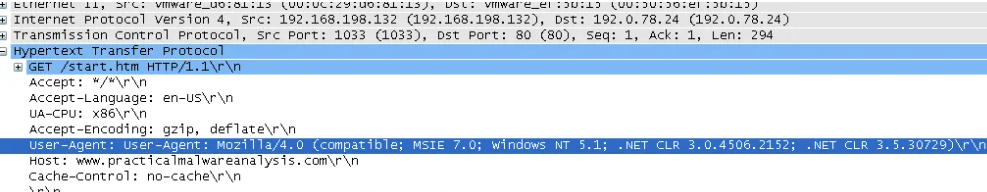
问题9：

这个恶意程序其实就是一个简单的后门程序。目的是给远端的攻击者提供一个shell命令接口。然后尝试删除自己的这个事实，可能是攻击者工具包中的一个一次性组件。

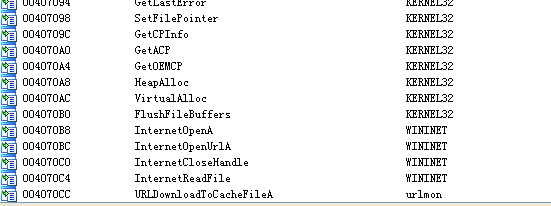
Lab14-03.exe

问题1：

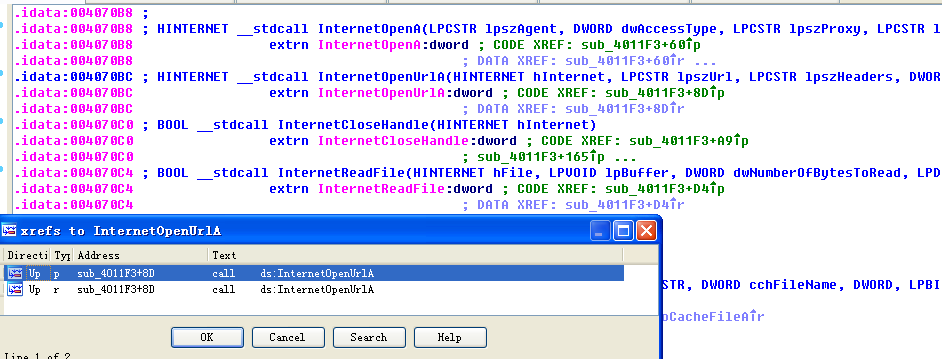
首先使用wireshark监控一下网络数据包：



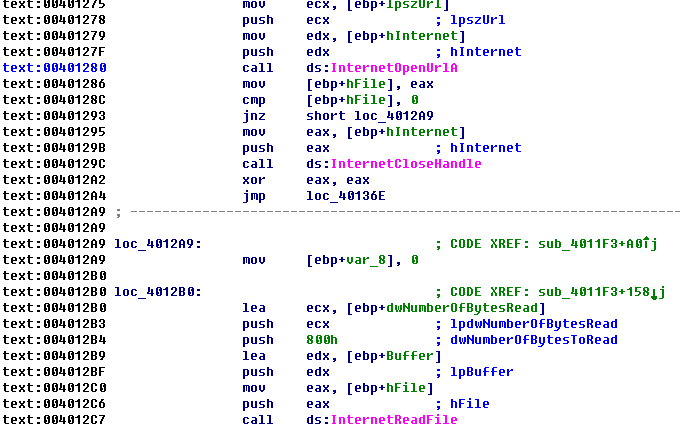
然后使用ida载入实验文件Lab14-03.exe



在这里还是看到了调用了WINNET库中的四个函数，先看一下函数InternetOpenUrlA：

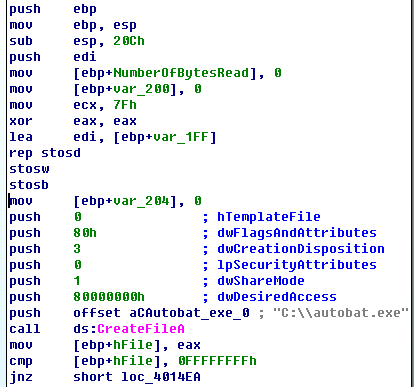


在这个函数的上方发现了两个字符串，并且还可以知道这两个字符串和前面捕获到的信令字符串是一样的，但是在网络信令中出现了“user-agent:user-agent:”这样的字符串，可以根据这个来创建了一个有效的检测特征。



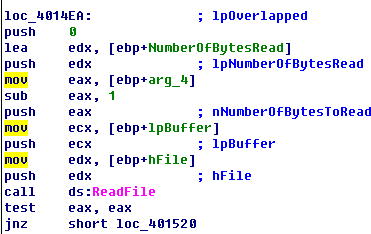
问题2：

发现在函数sub\_4011F3的其中一个参数Source作为了InternetOpenUrlA函数的lpszUrl参数，它定义的信令的目的地址URL，跟踪来源发现它在main函数中被调用，可以看到Source参数是由位于0x00401793的sub\_401457函数所调用，进入函数sub\_401457：

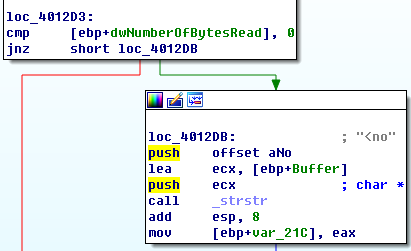


如果CreateFileA执行失败就会执行下面的代码：

发现了C:\autobat.exe和http://www.practicalmalwareanalysis.com/start.htm这两个字符串。如果文件不存在，它将反馈到这个信令的URL。

再来看一下如果CreateFileA执行成功

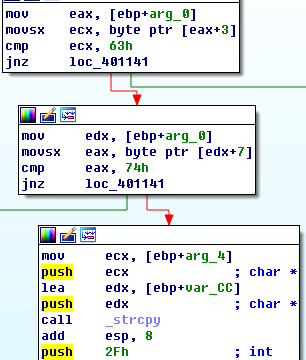
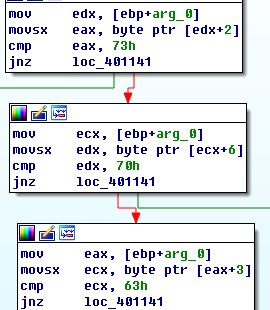
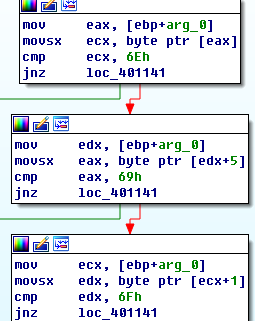
程序会调用函数ReadFile来读取C:\\autobat.exe这个文件的内容，并保存到lpBuffer中，lpBuffer是InternetOpenUrlA函数的lpszUrl参数，所以呢就可以推测autobat.exe是存储URL明文的配置文件。

分析到这里我们就可以知道，当配置文件不可用时，域名和URL路径都会采用硬编码。硬编码的URL与所有配置文件一起来构造特征。然而，以硬编码组件作为检测目标，比结合硬编码组件与动态URL链接，检测效果可能会更好。因为使用的URL存储在配置文件中，并且随着命令而改变，它是临时的，所以不是持久的网络特征。

问题3：

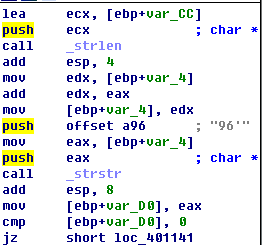
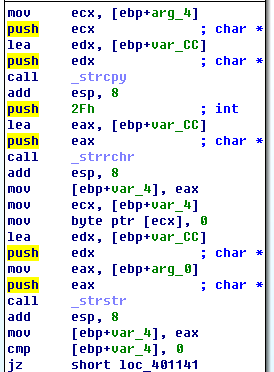
在函数sub\_4011F3发现调用了函数\_strstr

这个函数的目的就是搜索一个字符串在另一个字符串中第一次出现的位置，参数是<no，判断是否是<no开头，具体的判断内容在函数sub\_401000中，进去看一下：



看到了打乱顺序的判断，判断的字符串就是noscript。

继续向下看：

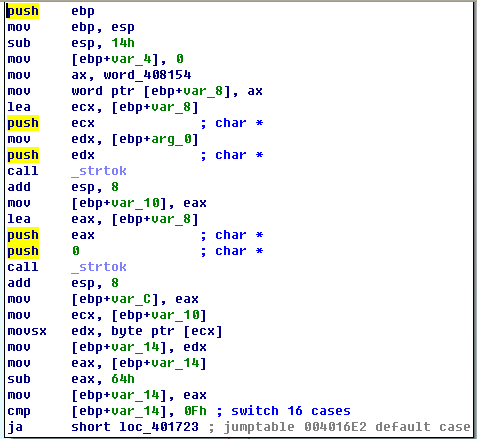


这是在截取/与96之间的内容。

这样我们就可以知道，恶意代码从web页面上noscript标签中某些特定组件来获得命令。使用这种技术，代码可以向一个合法的网页发生信令，并且接受合法的内容，这使防御者在区分是恶意流量还是合法流量变得更加困难。

问题4：

回到main函数，进入sub\_401684函数，

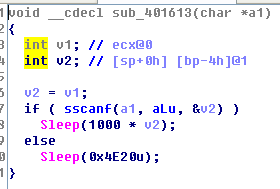
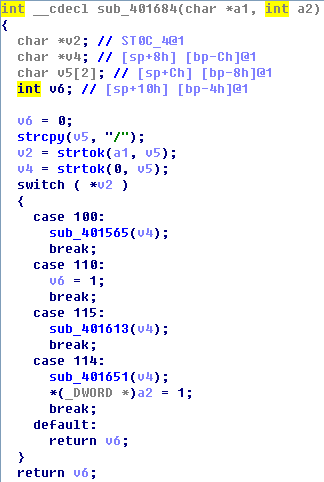


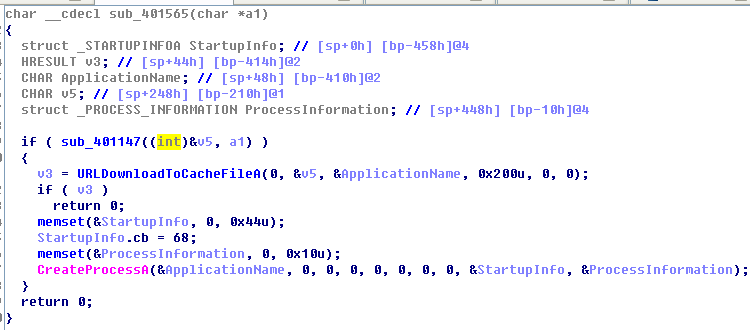
这里调用了函数\_strtok，选择函数，并将命令的内容分为两个部分，保存在两个变量里。继续向下看可以知道恶意程序中必须有一个noscript的标签，后面会跟随一个ULR,这里包含的域名和原始网页请求的域名相同，并且要求是以96位结尾。包含的命令的第一个字符必须和程序支持的命令相对应。当搜索noscript标签时，会先搜索<no，然后通过比较来确定noscript标签，当命令匹配时，程序仅仅会考虑第一个字符，这就会对我们的分析早成一定的困扰。

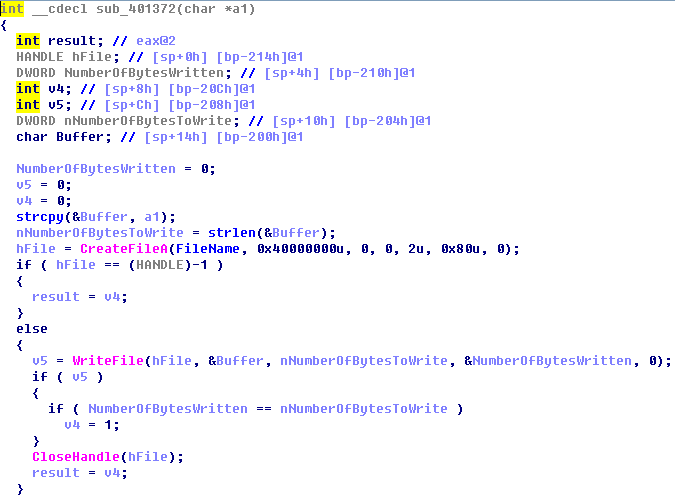
问题5：复制 /abcdefghijklmnopqrstuvwxyz0123456789:.

优点就是这并不是标准的base64编码，若要理解其内容需要进行逆向分析，缺点就是一致性，开头部分总是字符串

问题6：包括quit,download,sleep,redirect。







问题7：一个下载器。

问题8：

1.

静态定义的域名和路径，以及动态发现的URL中相似信息有关的特征。

信令中静态组件有关的特征。

能够识别命令初始请求的特征。

能够识别命令与参数对特定属性的特征。

问题9：

alert tcp $HOME\_NET any -> $EXTERNAL\_NET $HTTP\_PORTS (msg:”PM14.3.1 Specific

User-Agent with duplicate header”; content:”User-Agent|3a20|User-Agent|3a20|

Mozilla/4.0|20|(compatible\;|20|MSIE|20|7.0\;|20|Windows|20|NT|20|5.1\;|20|

NET|20|CLR|20|3.0.4506.2152\;|20|.NET|20|CLR|20|3.5.30729)”; http\_header;

Sid:20001431; rev:1;)

1. **实验结论及心得体会**

**由此，本学期的恶意代码分析工作已经到达了尾声，本书后面还有几章实验内容，但课程范围之外的内容有一定的挑战性，将会是我们继续进行恶意病毒分析的一把很好的钥匙。在本学期我学习了恶意代码病毒分析的多种原理以及各种各样的解决不同问题的方法，使我受益匪浅。希望未来我能够将本学期学到的宝贵财富应用于实践当中，检验我学习到的诸多技能。**

**本次实验当中遇到的很大的问题来源于恶意代码的网络行为，在前文的分析当中，我们通过静态分析等等查找相应的网页是否有可能出现，虽然相当简洁，但是在实际的运用当中仍然具有一定的不确定性，因此我们不能笼统的将具有相应的网址或者访问网络特征的行为归类为具有网络行为。本次实验采用wireshark抓包检测，更加准确，验证的实验更加具有说服力。**

**本次实验还遇到一个问题，旧版的wireshark在使用过程当中具有一定的不便性。在此，我首先尝试了在新版windows上运行恶意代码以进行抓包，但是由于位数不同，并不能捕获到目标网址。这也与之前实验当中的尝试相类似。可见在早期windows上进行运行的程序一直存在着有可能不被新版windows兼容的问题。说明了产品的更新迭代一定会随着实践的变化二层出不穷，恶意代码的功能也会随着技术的不断进步和革新而逐渐变得更加强大。**

**但与此同时，我们的分析技术和经验也会将我们的分析变得更加容易。在查找到旧版wireshark使用方法过后，我能够顺利的将恶意病毒的网络信息抓包捕获，并且将相应的目标网址通过一定的方法截获，这说明计算机便高度分析与防治人员是可以通过努力，减少恶意代码带来的危害，并且一定程度上改善计算机病毒环境。**

**本学期的计算机病毒之旅是我受益匪浅，在此衷心的感谢王老师，邓老师，以及助教们的殷勤付出。我将会继续努力，在推进恶意代码病毒分析的道路上继续前进，再接再励。**