

病毒分析实验报告

|  |  |
| --- | --- |
| 学生姓名 | maybeLocalhost |
| 学 号 |  |
| 专业班级 |  |
| 指导教师 |  |
| 学 院 | 计算机学院 |
| 完成时间 | 2021.01 |

目录

[一、实验简介 1](#_Toc62511317)

[1.1 实验目的 1](#_Toc62511318)

[1.2 实验内容 1](#_Toc62511319)

[1.3 实验环境 1](#_Toc62511320)

[二、实验原理 1](#_Toc62511321)

[2.1 恶意代码类型 1](#_Toc62511322)

[2.2 静态分析基础技术 2](#_Toc62511323)

[2.3 恶意代码分析步骤 3](#_Toc62511324)

[2.3.1 反病毒引擎扫描 4](#_Toc62511325)

[2.3.2 哈希值 4](#_Toc62511326)

[2.3.3 查找字符串 4](#_Toc62511327)

[2.3.4 加壳与混淆恶意代码 5](#_Toc62511328)

[2.3.5 使用Dependency Walker工具探索动态链接函数 5](#_Toc62511329)

[三、实验过程 6](#_Toc62511330)

[3.1 Chapter\_1L 6](#_Toc62511331)

[3.1.1 Lab01-01.dll和Lab01-01.exe 6](#_Toc62511332)

[3.1.2 Lab01-02.exe 12](#_Toc62511333)

[3.1.3 Lab01-03.exe 18](#_Toc62511334)

[3.1.4 Lab01-04.exe 20](#_Toc62511335)

[3.2 Chapter\_3L 24](#_Toc62511336)

[3.2.1 Lab03-01.exe 24](#_Toc62511337)

[3.2.2 Lab03-02.dll 31](#_Toc62511338)

[3.2.3 Lab03-03.exe 34](#_Toc62511339)

[3.2.4 Lab03-04.exe 37](#_Toc62511340)

[3.3 Chapter\_5L 39](#_Toc62511341)

[3.4 Chapter\_7L 47](#_Toc62511342)

[3.4.1 Lab07-01.exe 47](#_Toc62511343)

[3.4.2 Lab07-02.exe 50](#_Toc62511344)

[四、实验总结 52](#_Toc62511345)

[五、参考文献 53](#_Toc62511346)

# 一、实验简介

## 1.1 实验目的

1. 熟练掌握分析工具集的功能和方法；
2. 针对恶意代码给出静态分析和动态分析，最好能使用反汇编工具针对程序做程序分片和污点传播分析；
3. 完成实验报告，详尽描述分析过程、方法、工具，给出恶意代码的行为特征。

## 1.2 实验内容

本实验涵盖以下主题：

1. 选择合适的PE文件浏览、分析工具，静态分析工具，动态分析工具（可以参照课内教学常见的工具），常见分析工具集，学习工具的使用；
2. 针对附件中Lab1、3、5内（分析得更多加分）的可执行文件（EXE和DLL），给出恶意代码分析，分析其恶意特性、恶意功能以及恶意行为。

## 1.3 实验环境

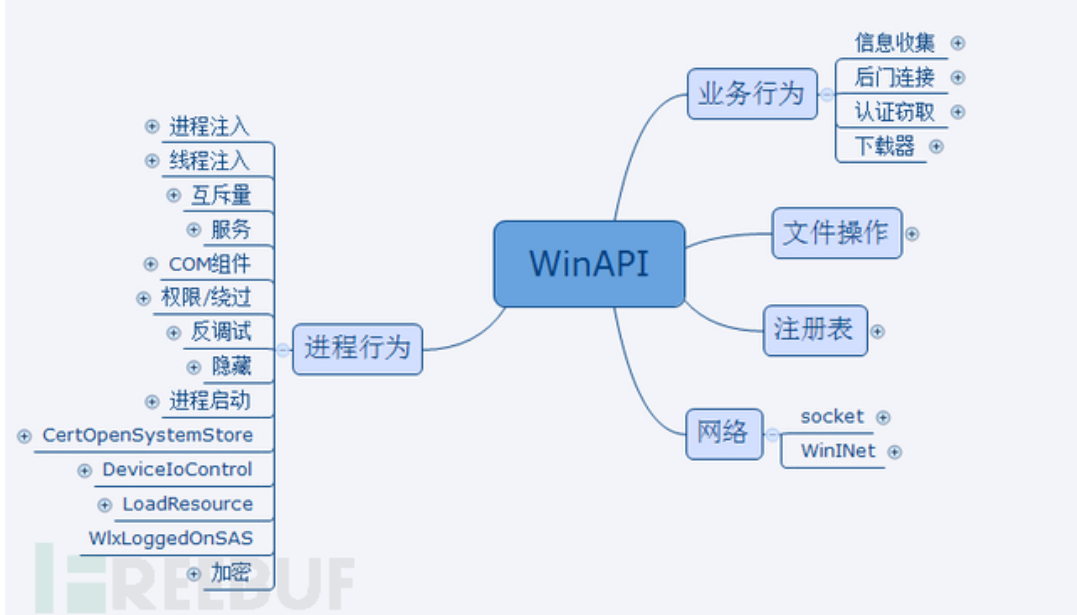
VMware Pro，Windows 10，Windows XP

# 二、实验原理

## 2.1 恶意代码类型

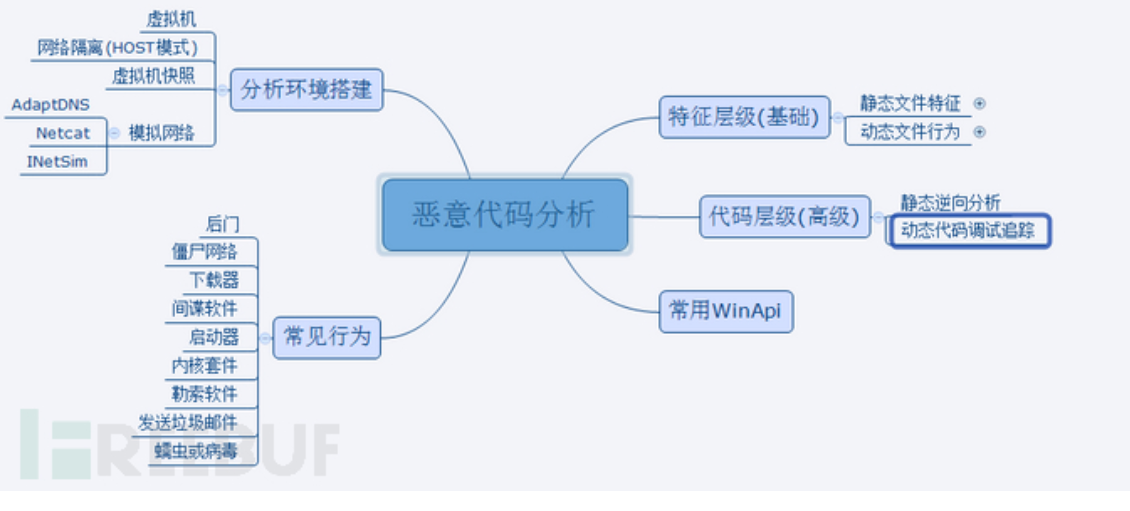
1. 后门: 恶意代码将自身安装到一台计算机来 允许攻击者来访问。后门程序通常让攻击者只需很少认证甚至无需认证，便可连接到 远程计算机上，并可以在本地系统执行命令。
2. 僵尸网络：与后门类似，也允许攻击者访问系统。但是所有被同一个僵尸网络感染的计算机将会从一台控制命令服务器收到相同的命令。
3. 下载器:这是一类只是用来下载其他恶意代码的恶意代码。下载器通常是在攻击者获得系统的访问时，首先进行安装的。下载程序会下载和安装其他的恶意代码。
4. 间谍软件:这是一类从受害计算机上收集信息并发送给攻击者的恶意代码。比如:键盘记录器。
5. 启动器: 用来启动其他恶意程序的恶意软件。一般使用一些非传统的技术，以确保其隐蔽性，或者以获取高权限访问系统。
6. 内核套件: 设计用来隐藏其他恶意代码的恶意软件。通常与其他恶意代码(后门)组合成工具套装。
7. 勒索软件:通过骚扰、恐吓甚至采用绑架用户文件等方式，使用户数据资产或计算资源无法正常使用，并以此为条件向用户勒索钱财。这类用户数据资产包括文档、邮件、数据库、源代码、图片、压缩文件等多种文件。赎金形式包括真实货币、比特币或其它虚拟货币。
8. 发送垃圾邮件的恶意代码:这类恶意代码在感染用户计算机之后，便会使用系统与网络资源来发送大量的垃圾邮件。
9. 蠕虫或计算机病毒:可以自我复制和感染其他计算机的恶意代码。

## 2.2 静态分析基础技术



静态分析要分析恶意软件执行了哪些行为，首先需要知道恶意软件常见的一些行为，如反弹shell，进程感染，文件隐藏，信息收集等。而要弄清这些行为，恶意软件需要对文件，注册表，进程进行操作，同时恶意软件要对感染的主机下发指令，传输收集到的信息，通常需要建立网络连接。要更准确的分析需要知道使用了哪些WinAPI实现了这些功能。对于初级的分析方法，只要掌握常见的知识，以及一些工具的使用即可快速的掌握。如可以使用工具分析文件的hash值，字符串，函数表，函数库依赖关系。 另外可以运行恶意代码用工具监控恶意软件对文件，注册表，网络，进程的访问与操作。要深入到代码层级去分析恶意代码，需要代码逆向分析的能力，需要读的懂汇编代码，熟练使用IDA进行分析，熟练使用ollydbg等调试工具。

## 2.3 恶意代码分析步骤





### 2.3.1 反病毒引擎扫描

当分析一个可疑的代码样本时，第一步就最好是拿多个反病毒软件扫描下这个文件，看是否有哪个引擎已经能够识别它。因为反病毒软件主要依靠一个已知的恶意代码可识别片段的特征数据库(病毒文件特征库)，以及基于行为与模式匹配的分析(启发式检测)。而攻击者可以轻易的修改自己的代码。这时，对同一个可疑恶意代码样本，运行多个不同的反病毒软件进行扫描检测 是相当有必要的，类似VirusTotal(http://www.virustotal.com/)在没有恶意代码分析所需要的一些知识和技能的情况下，使用现成的杀毒工具无疑是最好的方法。即使有这个能力，如果恶意软件样本的泄露不会造成重要资产的泄露，使用现成的工具进行扫描也能够提供很多有用的信息，减少大量的人工投入。

### 2.3.2 哈希值

哈希是一种用来唯一标识恶意代码的常用方法。恶意代码样本通过一个哈希程序，会产生出一段用于唯一标识这个样本的独特哈希值(类似一种指纹)MD5算法是恶意代码分析最为常用的一种哈希函数，SHA算法也同样流行。一旦对一个恶意样本拥有一个哈希值之后，我们可以把它做如下用途:

* 将哈希值作为标签使用；
* 与其他分析师分享哈希值，以帮助他们来识别恶意代码；
* 在线搜索这段 哈希值，看看这个文件是否被识。

### 2.3.3 查找字符串

一个程序会包含一些字符串，比如打印出的消息，连接的URL,或者复制文到某个特定的位置。从字符串中进行搜索是获取程序功能提示的一种简单方法。比如程序访问一个URL，访问的URL就是存储为程序中的一个字符串。可以使用String程序来搜索可执行文件中可打印的字符串，而一般在可执行文件中是以ACSII码和Unicode(这里指的是微软的)存储的。

### 2.3.4 加壳与混淆恶意代码

恶意代码编写者经常使用加壳或混淆技术，让程序更难被检测或分析。混淆程序是恶意代码编写者尝试去隐藏其执行过程的代码。而加壳程序则是混淆程序的一类，加壳后的恶意程序一般会被压缩，并且难以分析。合法程序大多总是会包含很多字符串，而由被加壳或混淆的恶意代码直接分析获得的字符串很少(一般包含LoadLibrary和GetProcAddress函数，他们是用来加载和使用其他函数功能的)。我们可以使用PEID(或exeinfoPE)检测加壳。

### 2.3.5 使用Dependency Walker工具探索动态链接函数

对于一个可执行文件，我们可以收集到最有用的信息之一，就是它的导入表。导入表函数是一个程序所使用的函数，但存储在其他程序中的那些函数。链接分为:静态链接，运行时链接与动态链接。

静态链接是Windows平台链接代码库最不常用的方法，当一个库被静态链接到可执行文件时，所有这个库中的代码都会被复制到可执行文件中。这会让执行程序大很多。

运行时链接在合法程序中并不流行，但是在恶意代码中是常用的，特别是当恶意代码被加壳或是混淆的时候。使用运行时链接的可执行程序，只有当需要使用函数时，才链接到库，而不是想动态链接模式那样在程序启动时就会链接。

动态链接是windows里最常用的，将一些函数打包在一个dll里，然后通过程序调用dll里的函数，在程序启动时就会链接上。

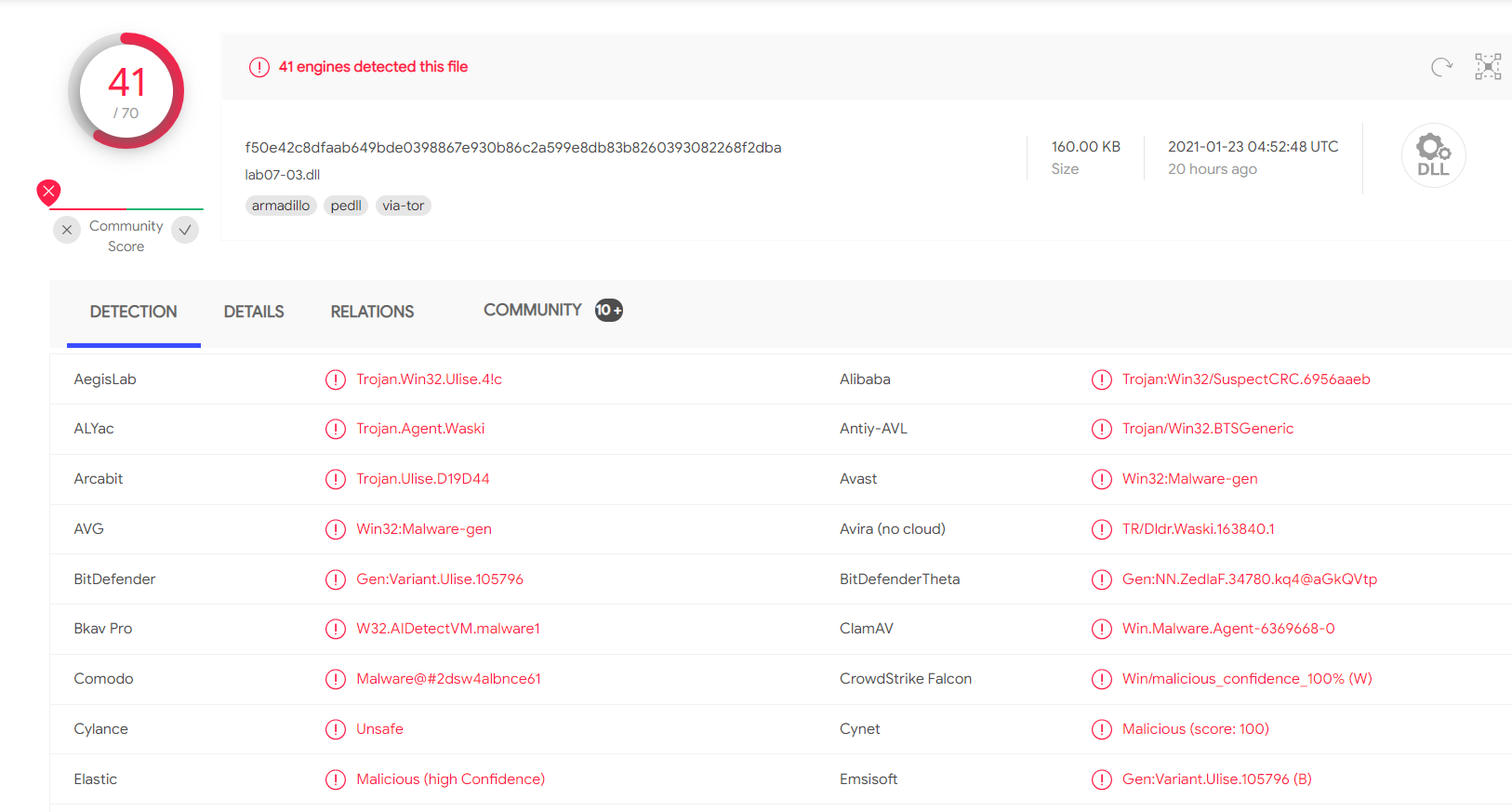
# 三、实验过程

## 3.1 Chapter\_1L

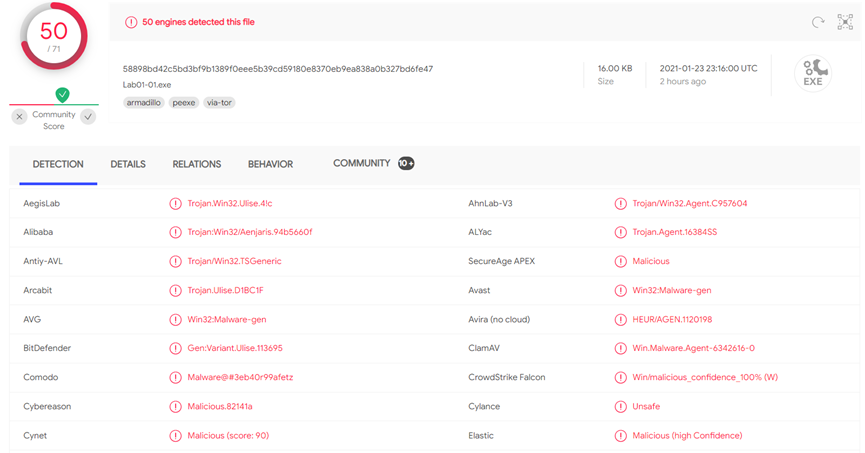
### 3.1.1 Lab01-01.dll和Lab01-01.exe

#### 1. 在线恶意代码扫描

将Lab01-01.dll拖到virustotal查看文件，可以看到文件匹配到已有的反病毒软件特征,部分截图如下：



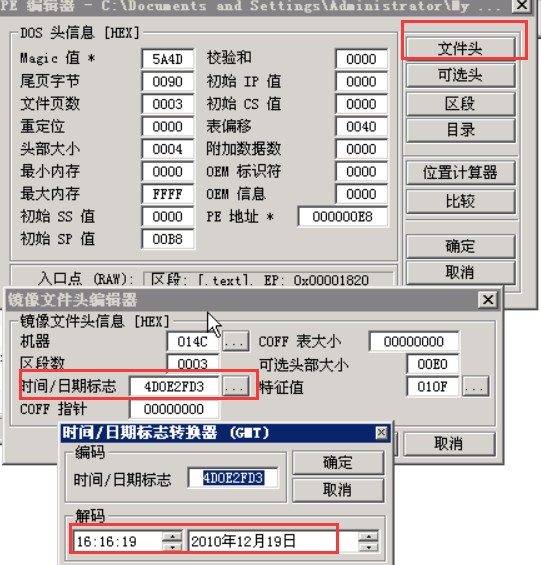
将Lab01-01.exe拖到virustotal查看文件，可以看到文件匹配到已有的反病毒软件特征,部分截图如下：



#### 2. 查看文件编译时间

查看文件编译时间一般有两种方法：

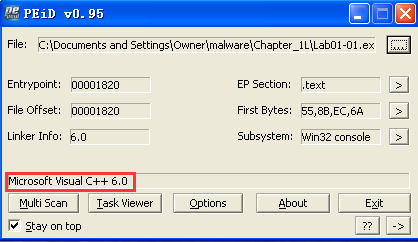
* Image\_nt\_headers –> Image\_file\_header –> Time Date Stamp查看编译时间；
* 使用工具PETools 查看dll和exe文件的编译时间：



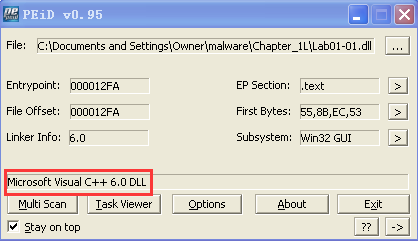


#### 3. 查看是否加壳

通过PEiD我们可以发现，该程序区段表比较完整，没有加壳。

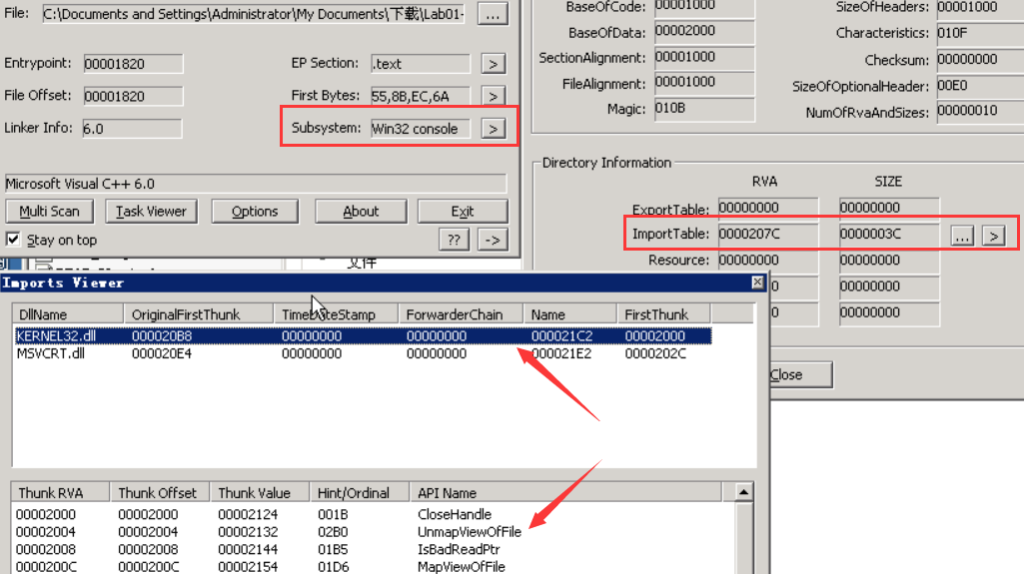


图表 1 Lab01-01.exe

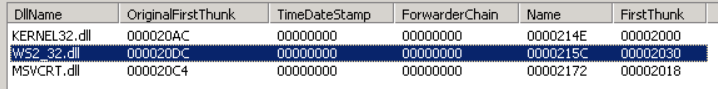


#### 4. 查看导入函数

使用PEId可以简单查看到import table中导入函数。



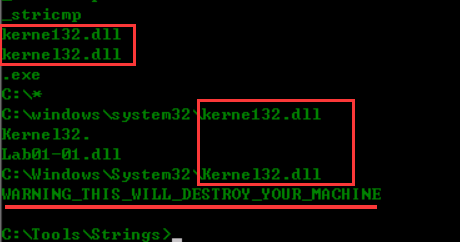
其中dll文件中存在WS2\_32.dll文件引用，是网络应用，证明此程序有联网操作。



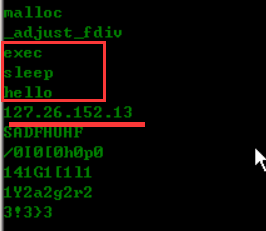
#### 5. 查看字符串

使用strings.exe文件查看所有可打印字符。

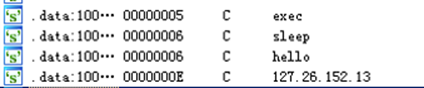
* exe文件分析：除了基本的KERNEL32.DLL和MSVCRT.DLL之外，还看到了调用CreateFileA，CopyFileA等可疑的API，通过字符分析，猜测是把程序的lab01-01.dll复制到C:\Windows\system32\kernel132.dll，改个名字，然后查找，调用该程序。



* dll文件分析：WS2\_32.dll是用于网络套接字编程的一个DLL，然后还有一个私有IP地址，估计是用于通信的IP地址，看到了CreateMutexA，OpenMutexA等API，是用于互斥锁的一个原子操作，后面还有创建句柄等等，怀疑是驻留后台的进程。除了已知的 CreateProcessA 和 Sleep，我们还注意到 exec 和 sleep。exec 可能用于通过网络给后门程序传送命令 ，再利用 CreateProcess 函数运行某个程序 。sleep 可能用于让后门程序进入休眠模式。

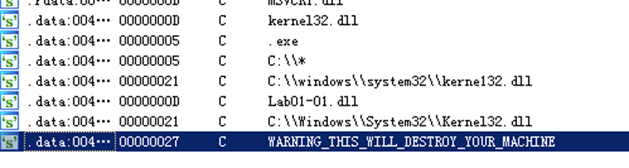


#### 6. 使用IDApro查看文件



图表 2 dll文件

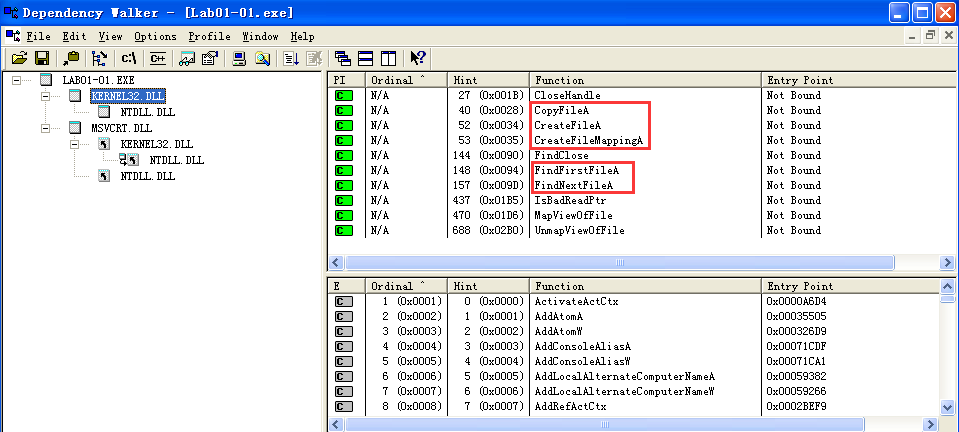
结合WS2\_32.dll推测他可能链接这个IP地址,获取exec命令来运行一个程序,一个sleep来是后门程序休眠。



有两个kernel32.dll，但仔细看有细微区别，一个是kernel32.dll，一个是kerne132.dll，推测这个恶意程序劫持了kernel32.dll。

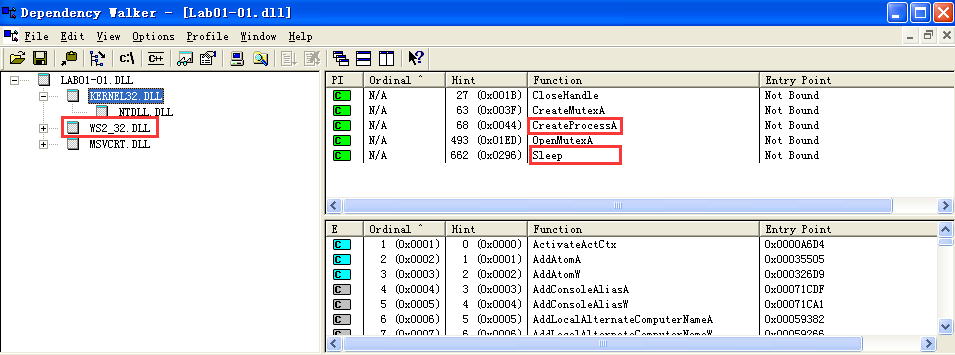
#### 7. Dependency Walkers查看程序

Lab01-01.exe 中，重点是 kernel32.dll 中同文件操作相关的导入函数。这意味着该程序可能会对系统中的文件进行搜索、调用和修改操作。



图表 3 Lab01-01.exe 中 kernel32.dll 的导入函数

Lab01-01.dll 中，重点是 CreateProcessA 和 Sleep 导入函数，这两个函数经常在后门程序中使用。WS2\_32.dll 则提供了联网功能。



图表 4 Lab01-01.dll 中 kernel32.dll 的导入函数

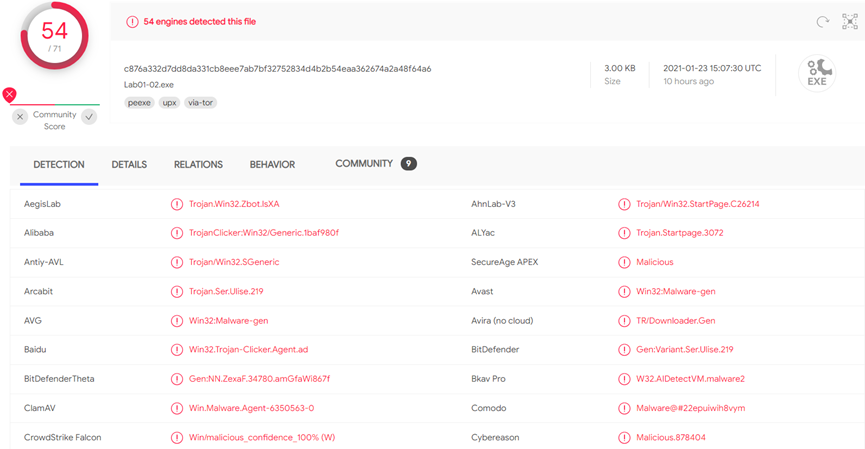
#### 8. 总结

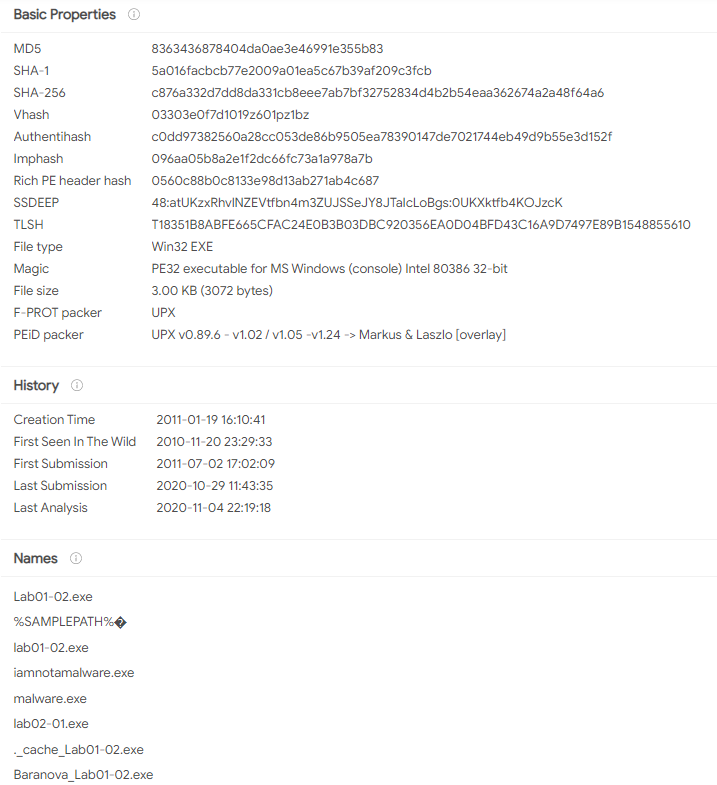
dll 文件可能是一个后门，而 exe 文件则用于安装和运行该后门。此恶意文件可能会连接某个网址，然后将自身lab01-01.dll文件重命名为kerne132.dll(数字1)并放在system32目录下。

### Lab01-02.exe

#### 1. 在线恶意代码扫描

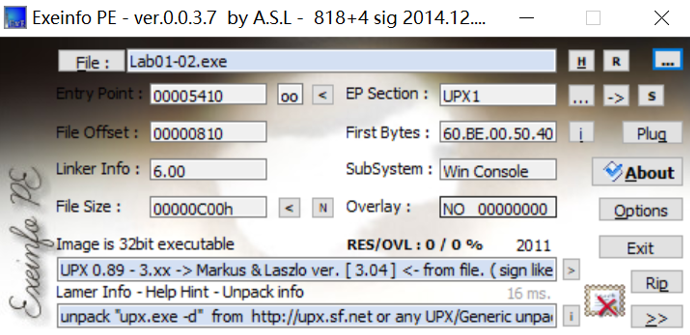
将Lab01-02.exe拖到virustotal查看文件，可以看到文件匹配到已有的反病毒软件特征,部分截图如下:





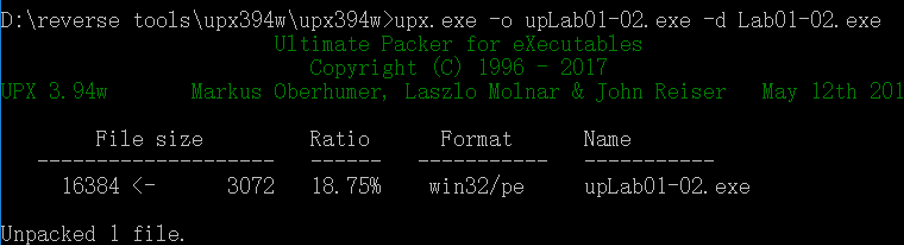
#### 2. 查看是否加壳

用Exeinfo查看，发现加了upx壳：



#### 3. 脱壳

下载这个版本的upx，用upx.exe -d命令脱壳：

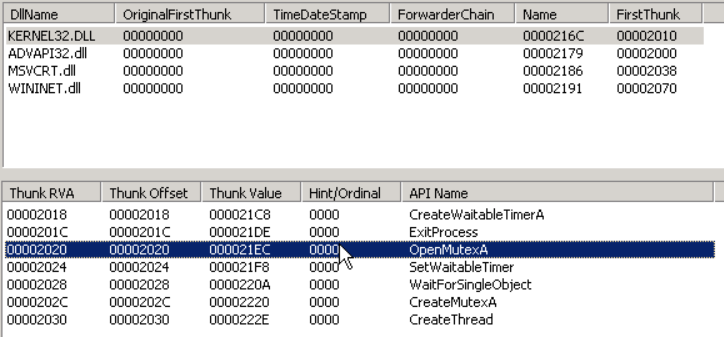


查看脱壳后的文件，发现没有壳了：

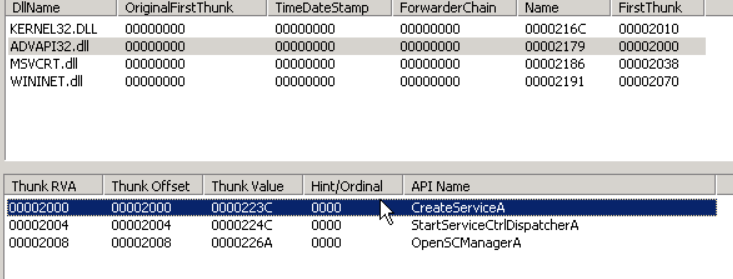


#### 4. 查看导入表

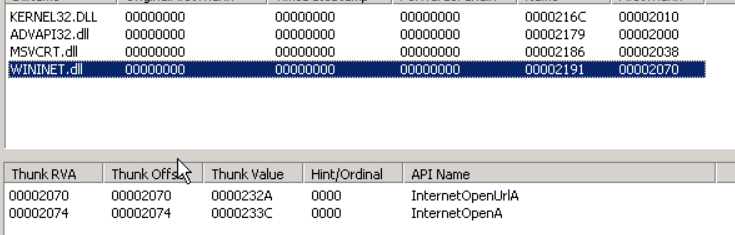
Kernel32.dll中可以看到关于互斥量和线程操作：



在advapi32.dll中有关于服务创建和打开操作：

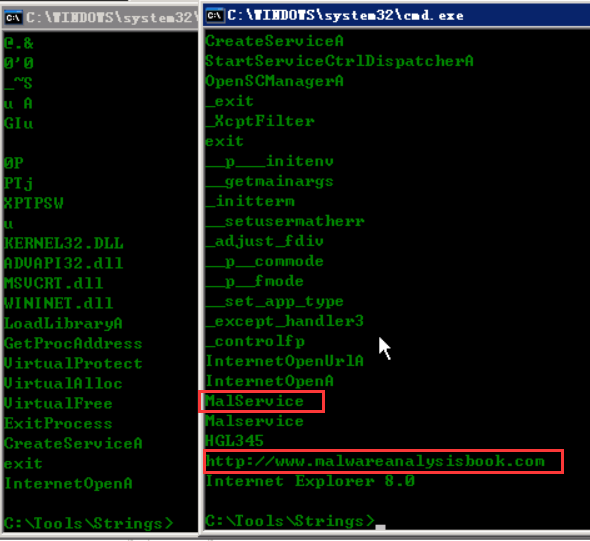


在wininet.dll中有关于打开某个url操作：

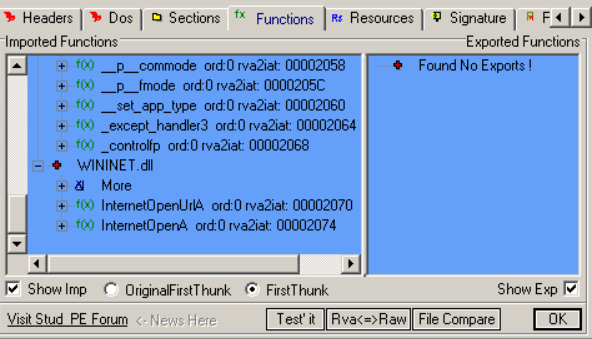


#### 5. 查看字符串

使用strings工具对脱壳前后的程序运行分析的结果是有明显区别的，脱壳后分析得到的可打印字符更多。我们可以看见一些函数，如WININET.DLL网络编程，获取模块名称，创建进程，创建互斥锁，打开IE浏览器，IE浏览器打开URL，下面还有这个URL的具体内容。



#### 6. 使用Study\_PE进行分析

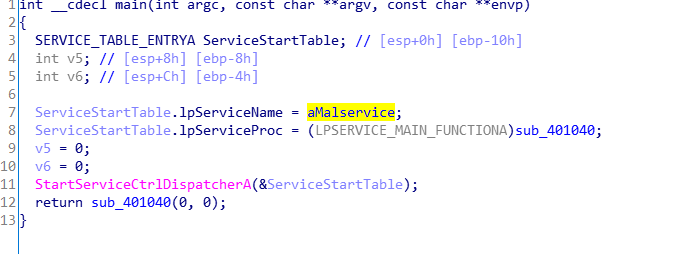


导入函数有常见的CreatMutexA,SystemTimeToFileTime,ExitProcess,CreateThread等等。

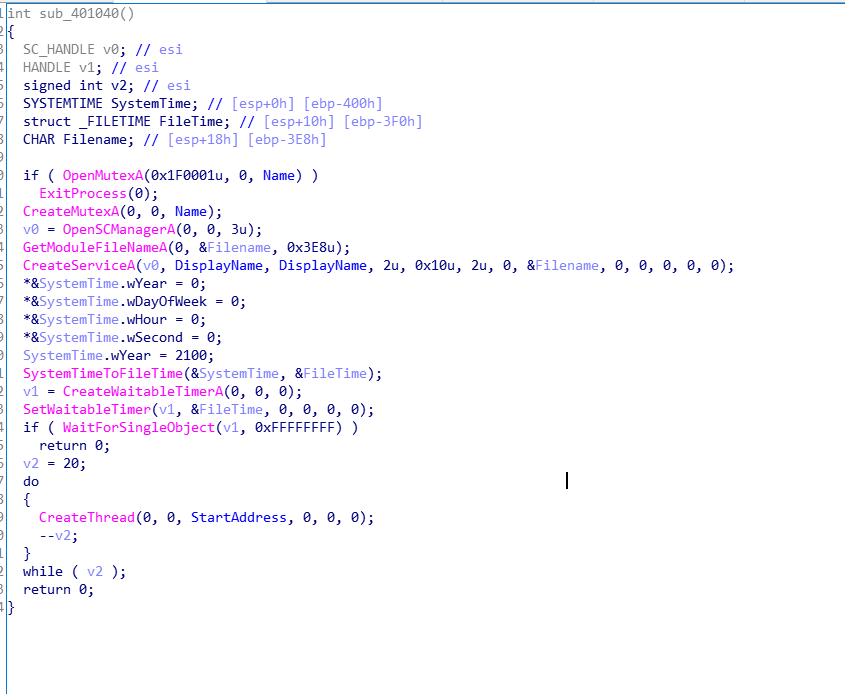
* wininet.dll是Windows的应用程序网络模块，负责网络通信，比如图中的打开URL。
* advapi32.dll是负责高级API接口的部分，负责注册表，函数与对象，事件日志等等。

#### 7. 总结

该病毒是把自己变成一个服务挂在系统后台，在某个时间一到就运行自己。



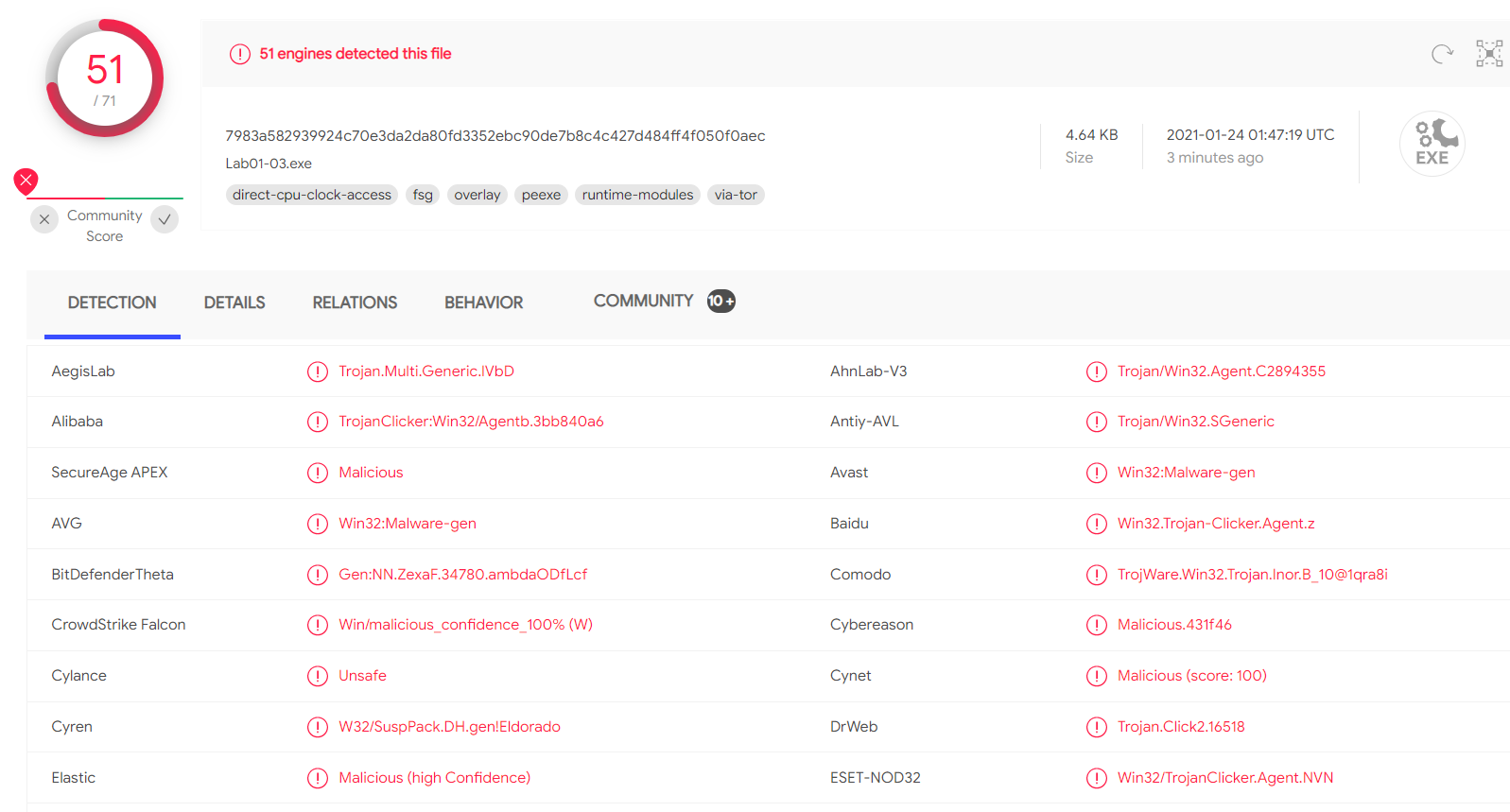
可以看到这个互斥锁的名字是HGL345，然后创建服务之后会弹出20次这个网站。



### 3.1.3 Lab01-03.exe

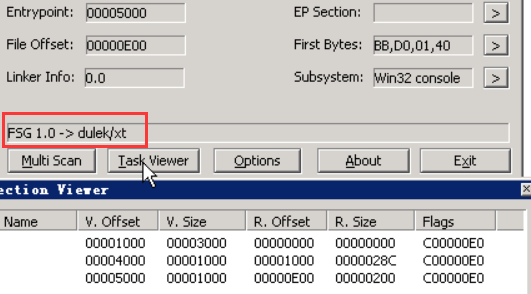
#### 1. 在线恶意代码扫描

将Lab01-03.exe拖到virustotal查看文件，可以看到文件匹配到已有的反病毒软件特征,部分截图如下:



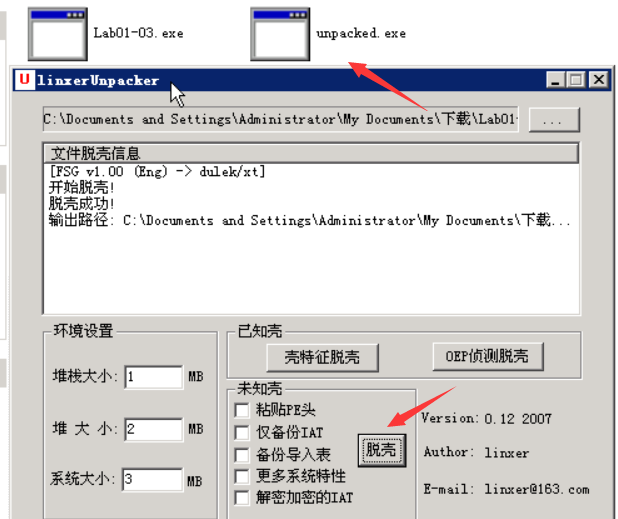
#### 2. 查看是否加壳

使用PEiD查看文件，发现该文件被加壳：



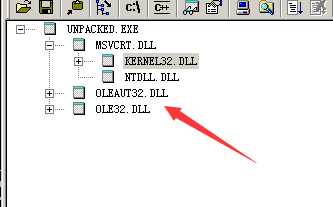
#### 3. 脱壳

使用linuxUnpacker工具自动脱壳，脱壳后程序为unpacked.exe。



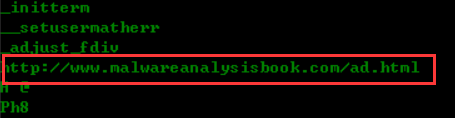
#### 4. Dependency Walkers查看程序

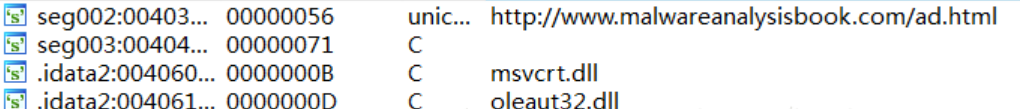
使用Dependency walker查看导入表，在ole32.dll动态链接库中调用了三个函数，使用了COM组件对象模型。第一个函数用于初始化，第二个用于创建一个实例。



#### 5. 查看字符串

找到某个url连接，可能需要打开此页面。





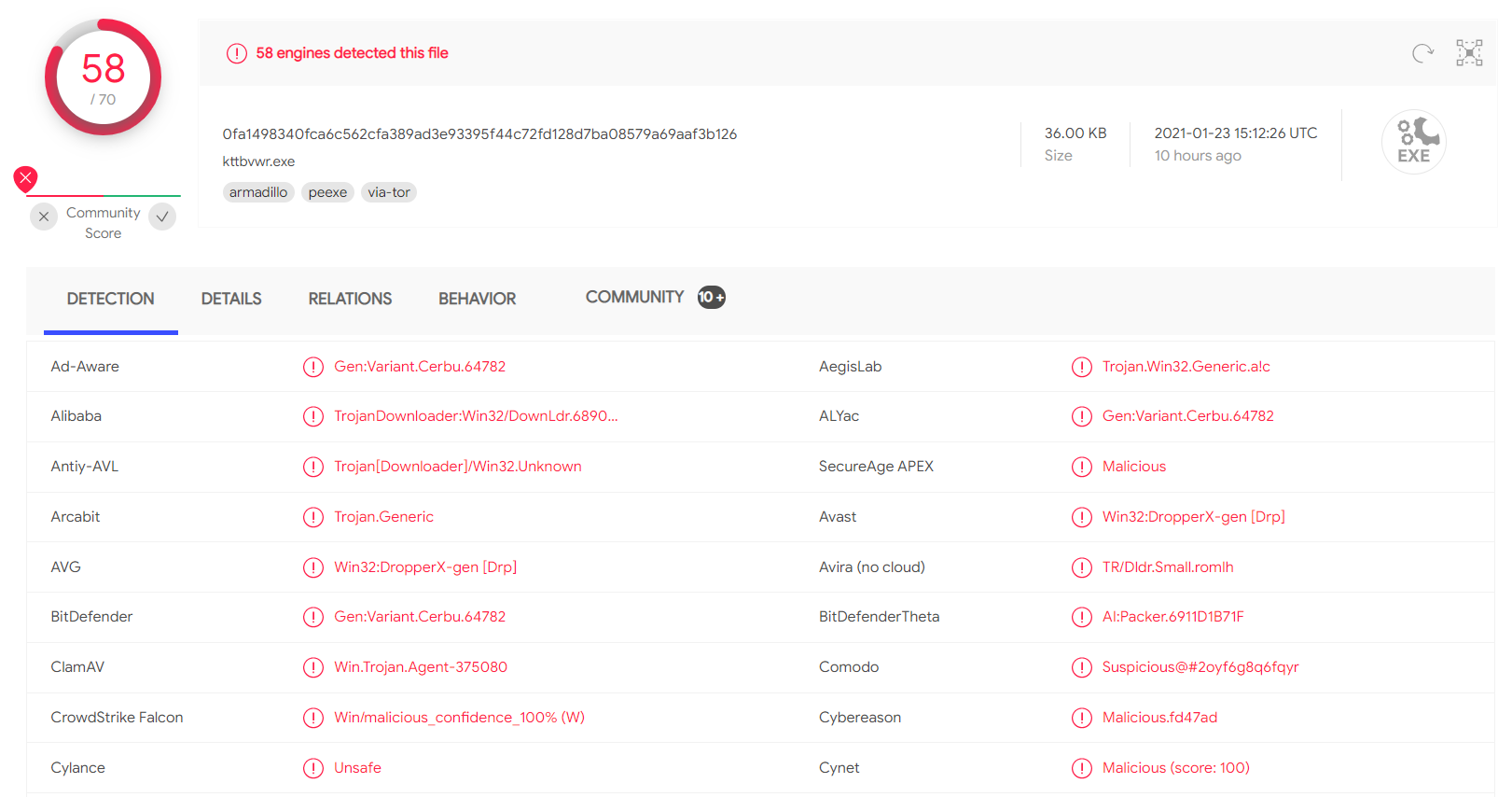
#### 6. 总结

该病毒是一个在内存中分配，然后创建一个关于网站的对象的操作。

### 3.1.4 Lab01-04.exe

#### 1. 在线恶意代码扫描

将Lab01-04.exe拖到virustotal查看文件，可以看到文件匹配到已有的反病毒软件特征,部分截图如下:



#### 2. 查看是否加壳

使用PEiD查看文件，发现该文件未被加壳：



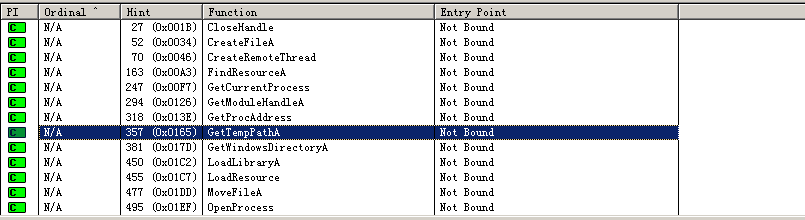
#### 3. 查看编译时间

使用工具PETools 查看文件的编译时间：

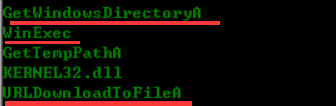


#### 4. 查看导入函数

函数导入了CreateFileA和MoveFileA这个函数，说明它可以创建一个文件和移动一个文件，还有CreateRemoteThread说明这个函数会在一个远程进程(Remote Process)里面创建一个自己的远程线程(Remote Thread)来运行恶意代码，还有FindResourceA、LoadResource和SizeofResourse这个函数，说明它再查找资源节的内容GetCurrentProcess和OpenProcess这个是获得想要获得进程的文件描述符，也是为了操作远程的进程。值得注意的是GetTempPathA这个函数，这说明这恶意代码可能会使用Temp目录。WinExec说明这个程序可以运行另一个程序代码。



#### 5. 查看字符串



1. GetWindowsDirectoryA获取系统目录用户搜索以及拷贝等操作；
2. WinExec操作执行；
3. URLDownloadToFileA 下载某URL内容到特定文件。



可能会访问此站点下载updater.exe程序，并且可能将自己修改为system32目录下的wupdmgrd.exe恶意程序实现混淆。

#### 6. 使用IDApro查看文件

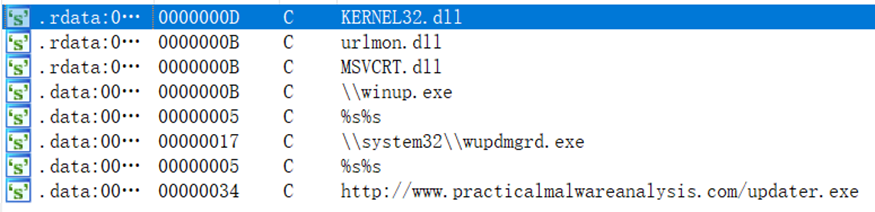
使用IDAPro打开该文件，第一行说明恶意代码创建的程序。



将资源文件用resource hacker打开：



根据“MZ”头和”PE“头判断这是一个exe文件，提出来保存成.exe文件再用IDApro打开，查看字符串可以看到一个链接，推测wupdmgrd.exe程序就是从下图链接下载得到。大致功能是这个文件下载到系统tmp目录中，然后伪装成为一个\System32\wupdmgrd.exe，并在命令行中执行这个文件。

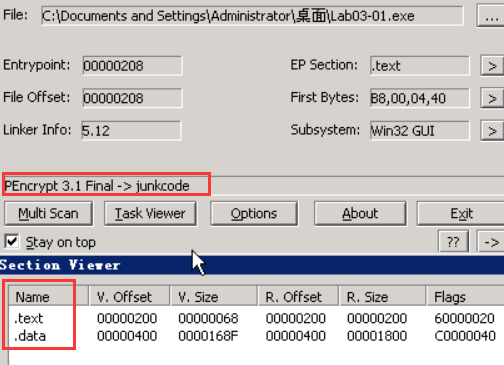


## 3.2 Chapter\_3L

### 3.2.1 Lab03-01.exe

#### 1. 查看是否加壳

使用PEiD查看文件，发现该文件被加壳：

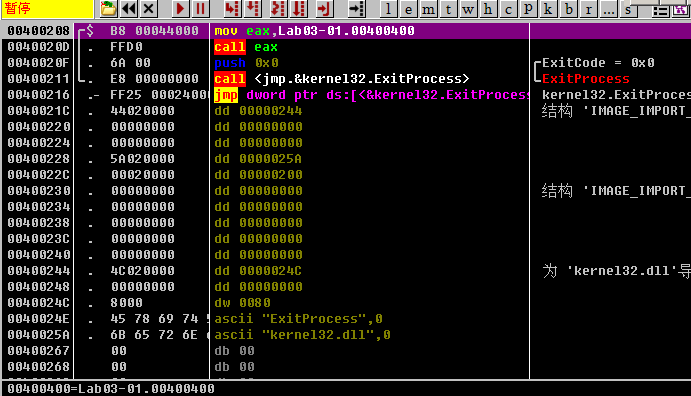


#### 2. 尝试脱壳

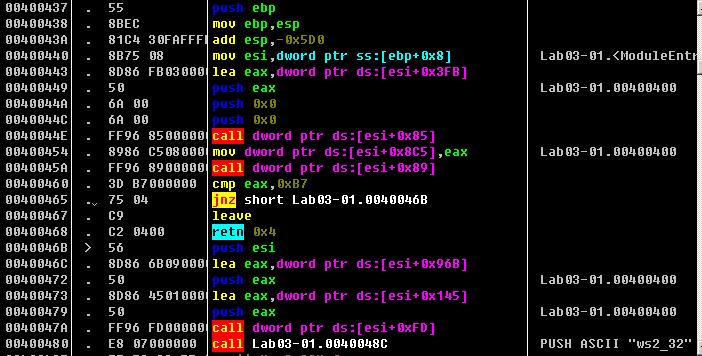
使用LinuxUnpacker脱壳失败：



尝试使用OD手动脱壳：



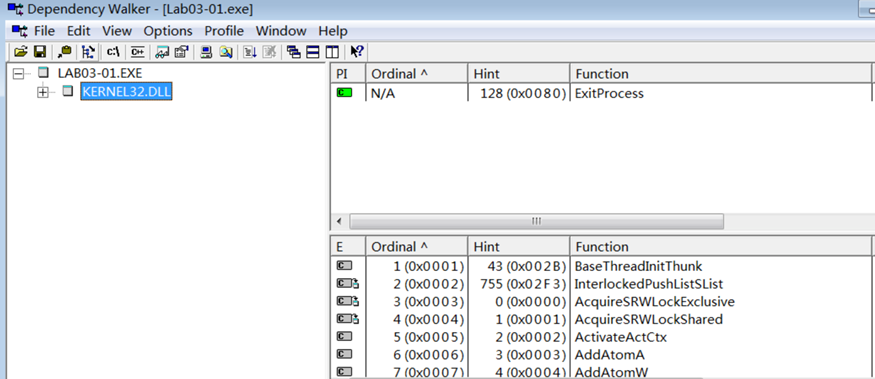
函数的运行入口点很奇怪，F8几次之后就直接结束了，我们用F7跟进这个call：



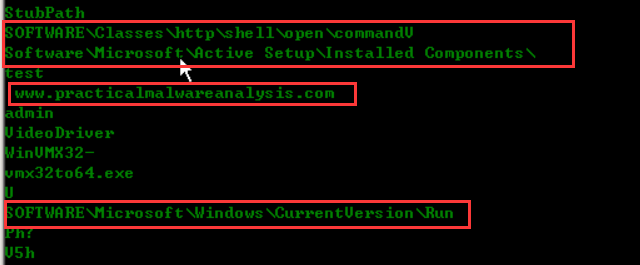
很正常的一个函数入口，甚至还看到了push ascii “ws2\_32”,然而在这个ws2\_32这一处下F4，直接结束，手动脱壳失败。

#### 3. Dependency Walker查看程序

由于无法脱壳，都可以看到的导入函数非常少.



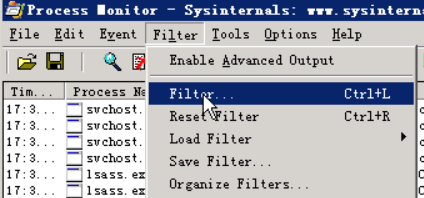
#### 4. 查看字符串



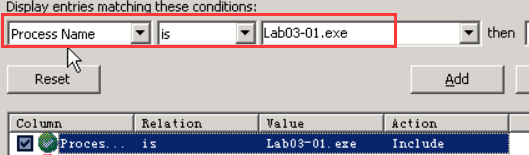
注册表操作、有一个网站链接、CurrentVersion\Run是恶意程序自启动常用的注册表路径。因为程序加壳，所以静态分析得到的结果不是很多

#### 5. 动态分析

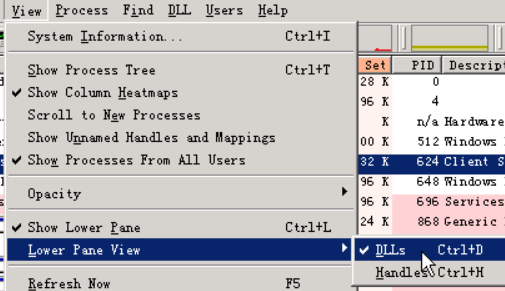
1. 启动Process Monitor，设置Filter条件



添加过滤条件为 ProcessName = Lab03-01.exe。

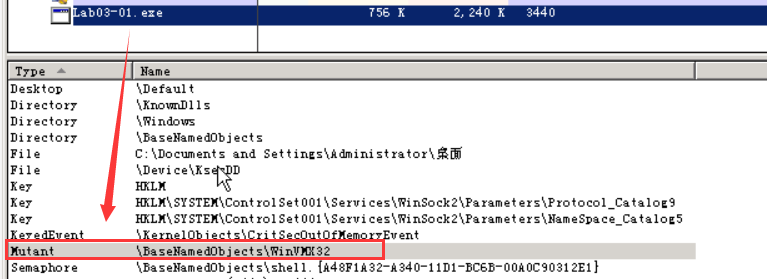


1. 启动Process Explorer，设置View –> Lower Pane View 设置显示dlls和handles

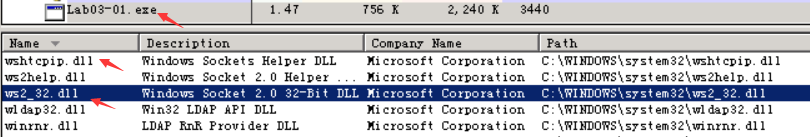


1. 运行lab03-01.exe程序

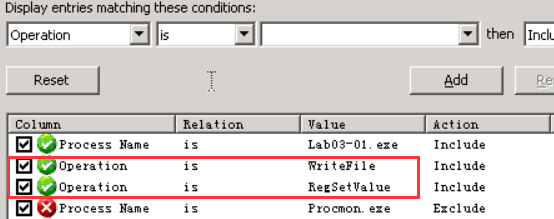
在Process Explorer中可以看到Lab03-01.exe已经运行，handles中查看到建立一个WinVMX32的互斥量



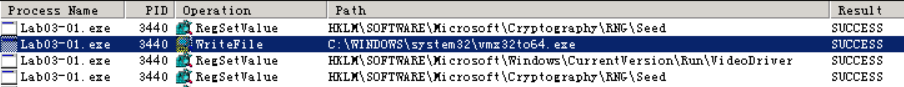
在dlls中看到有ws2\_32.dll和wshtcpip.dll的调用，说明存在网络层操作



然后在Proces Monitor中添加Filter，增加Operation的WriteFile和RegSetValue过滤

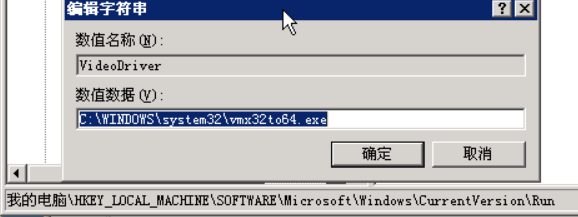


可以看到几个比较重要的操作结果

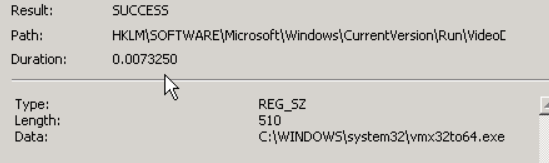


* 设置生成rng\seed随机数种子的注册表操作
* 在system32目录下创建exe文件操作
* 在currentversion\run下创建VideoDriver自启动项键值

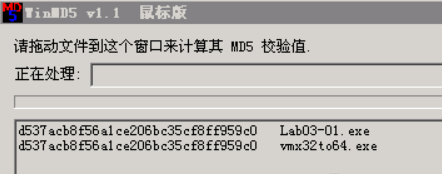
来到注册表，发现在此路径下新创建的VideoDriver键值指向的程序就是system32目录下新创建的vmx32to64.exe程序，用于自启动。



也可以在Process Monitor中双击查看到此键值数据

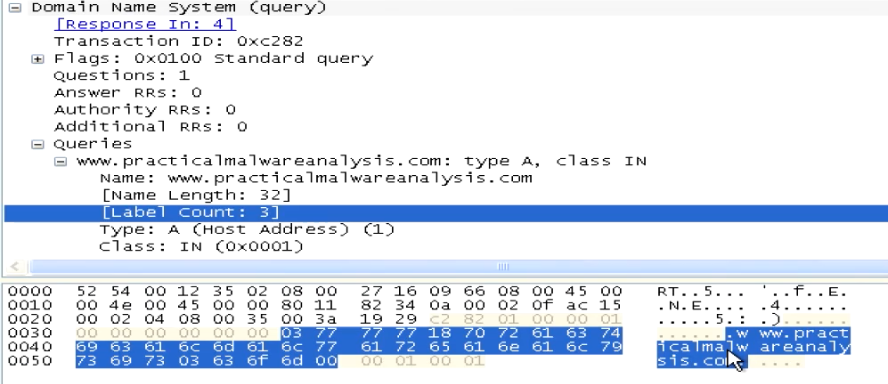


可以看到system32目录下的vmx32to64.exe程序跟实验程序Lab03-01.exe程序是一样的，也就是实现了自己的复制并拷贝到system32系统目录下。

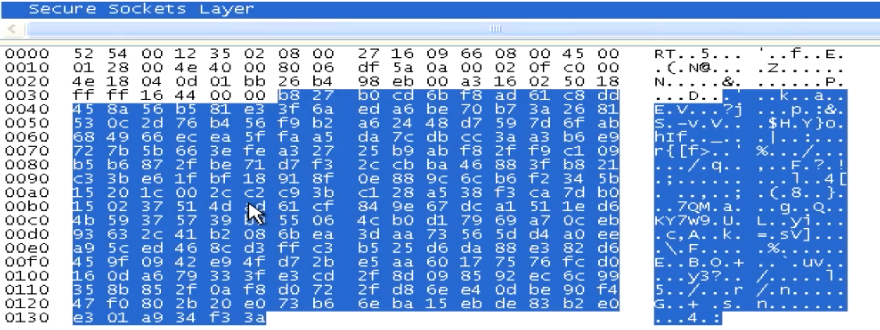


1. wireshark抓包

发现对www.practicalmalwareanalysis.com这个域名进行了解析。



并且在SSL协议上发送了16\*16=256 个字节的数据。



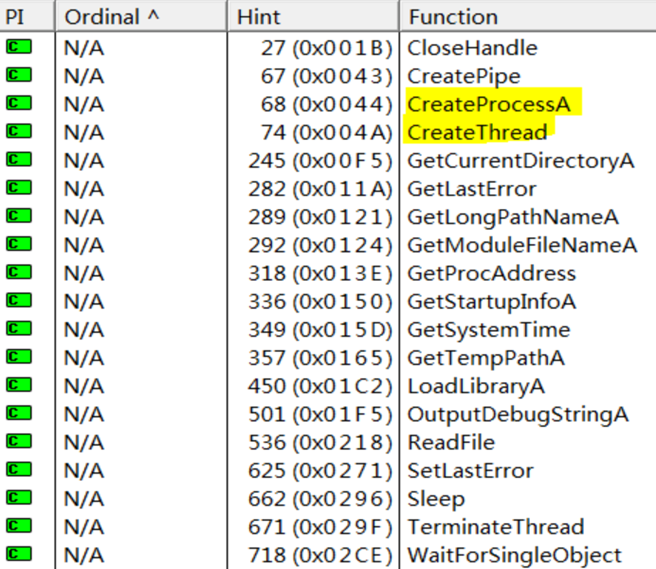
### 3.2.2 Lab03-02.dll

#### 1. 查看导出函数

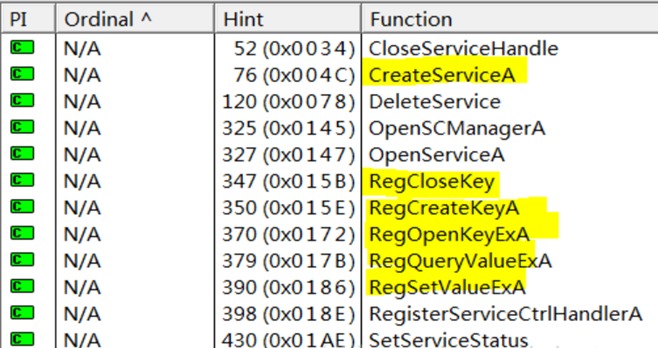
先使用静态分析技术，使用PEview找到五个导出函数：



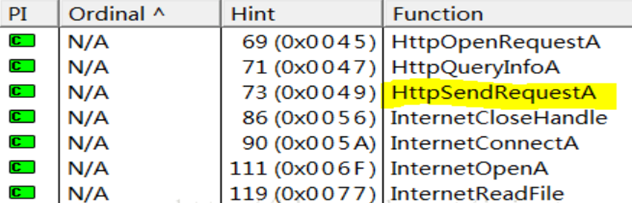
#### 2. Dependency Walkers查看程序



说明代码会创建进程，线程。

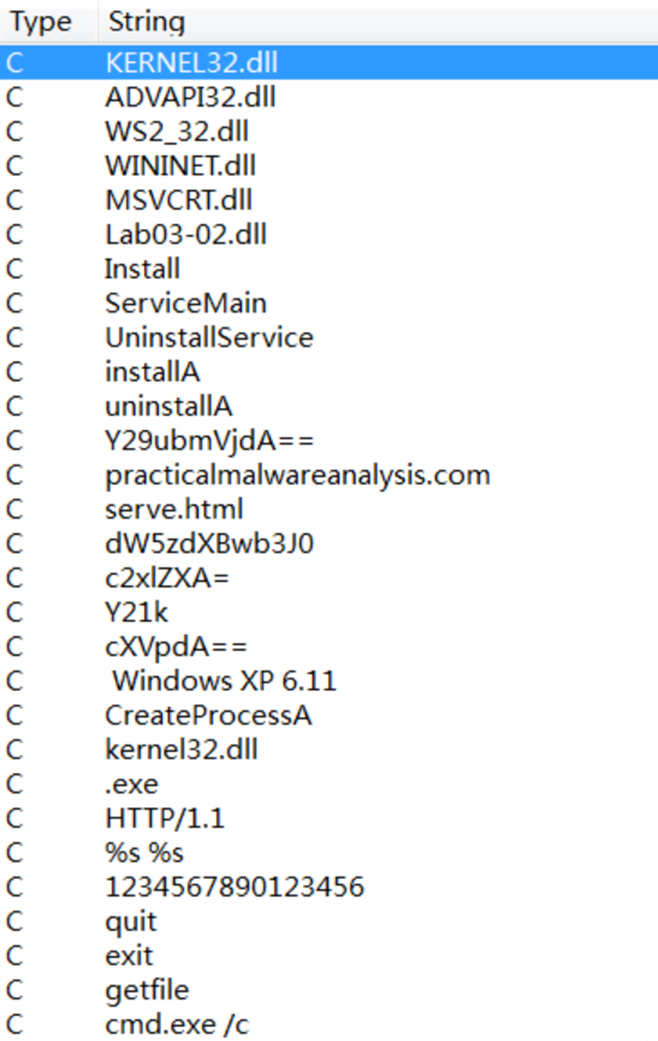


说明代码会操作服务，还有操作注册表。



猜测这个恶意代码还使用网络连接。

#### 3. 查看字符串



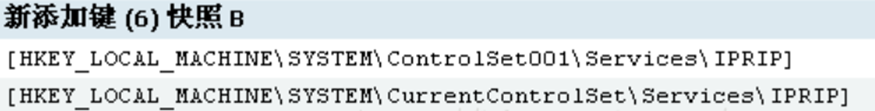
字符串中我们可以发现譬如域名、函数、网页、下载文件的指令、运行文件的指令，还有一些注册表操作语句。

#### 4. 在虚拟机中运行该程序

运行该程序前，进行注册表快照存储，process explorer打开，输入：

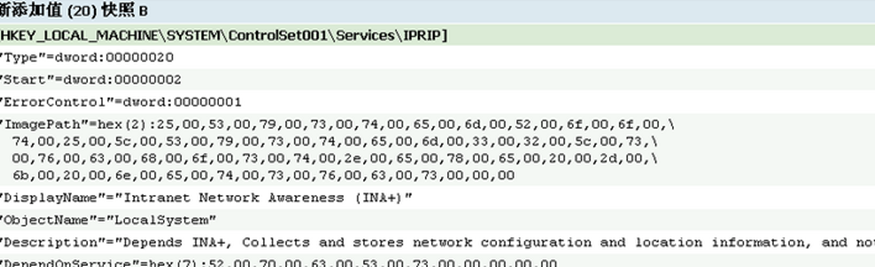
rundll32.exe Lab03-02.dll,install

然后查看注册表对比，发现：



该DLL创建了一个叫IPRIP的服务。这些新建的注册表如下：

[HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet001\Services\IPRIP\Parameters]   
"ServiceDll"=hex(2):43,00,3a,00,5c,00,44,00,6f,00,63,00,75,00,6d,00,65,00,6e,\  
 00,74,00,73,00,20,00,61,00,6e,00,64,00,20,00,53,00,65,00,74,00,74,00,69,00,\  
 6e,00,67,00,73,00,5c,00,36,00,34,00,5c,00,4c,68,62,97,5c,00,42,00,69,00,6e,\  
 00,61,00,72,00,79,00,43,00,6f,00,6c,00,6c,00,65,00,63,00,74,00,69,00,6f,00,\  
 6e,00,5c,00,43,00,68,00,61,00,70,00,74,00,65,00,72,00,5f,00,33,00,4c,00,5c,\  
 00,4c,00,61,00,62,00,30,00,33,...



ImagePath中的值，翻译过来就是下面的命令语句：

%SystemRoot%\System32\svchost.exe -k netsvcs

这个DLL会将自己安装成为一个服务IPRIP，然后由于这个恶意代码是个DLL文件，它就依赖一个可执行文件来执行它，然后它依赖的就是svhost.exe这个可执行文件来启动。

### 3.2.3 Lab03-03.exe

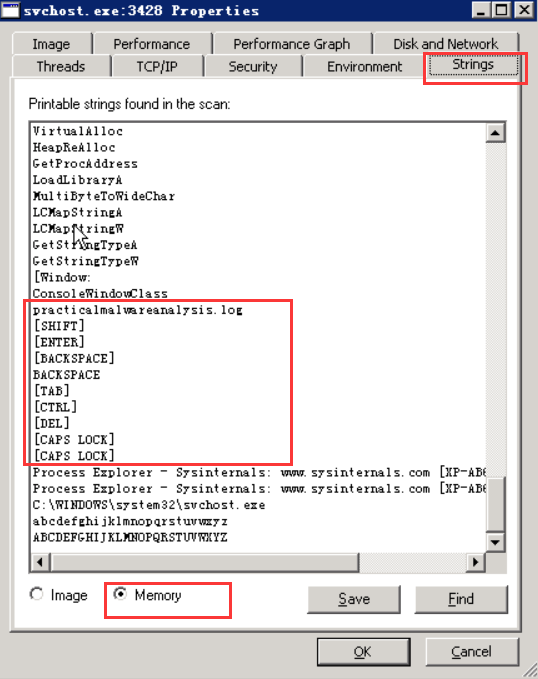
打开Process monitor & process explorer，运行exe程序，在PE中发现程序Lab 03-03.exe运行后一闪而过。



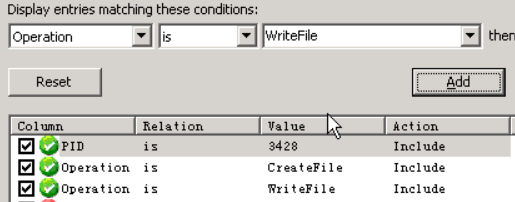
并且只留下一个svchost孤儿进程。



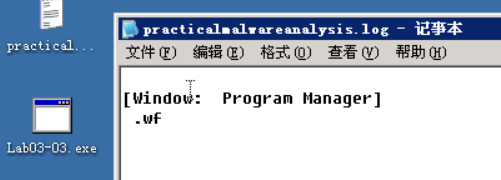
双击svchost孤儿进程，在strings中查看memory，发现一个log文件，有键盘特殊按键的标识。



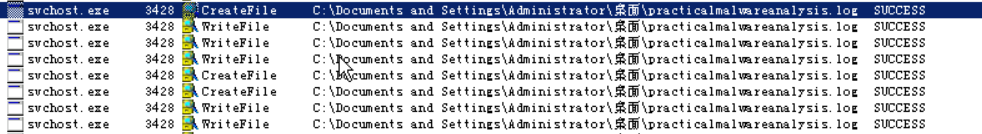
在PM中过滤查找pid = 3428此进程，并且添加对创建文件以及写入文件操作的过滤。



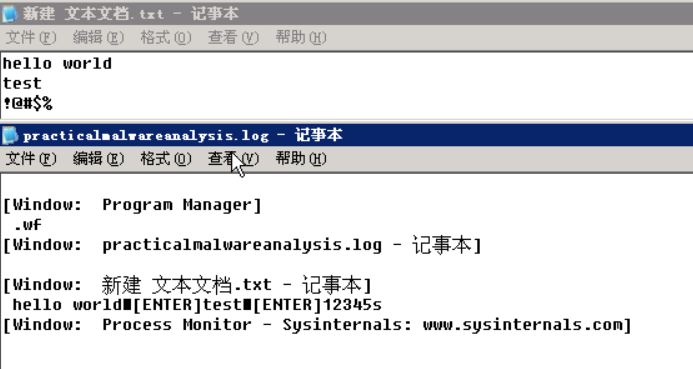
在同一路径下，已经可以看到自动创建的log文件。



可以监视到对此log文件的不断读写操作。



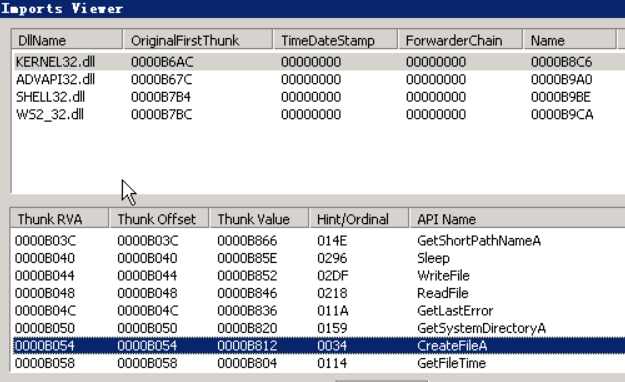
新建一个txt文件，随便输入一些内容，然后就可以在此log文件中查看到输入的记录。



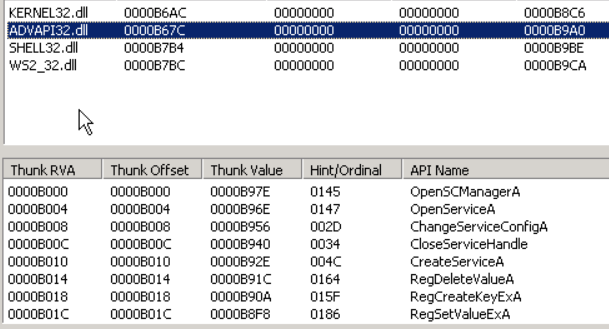
推测此恶意程序为一个键盘记录器

### 3.2.4 Lab03-04.exe

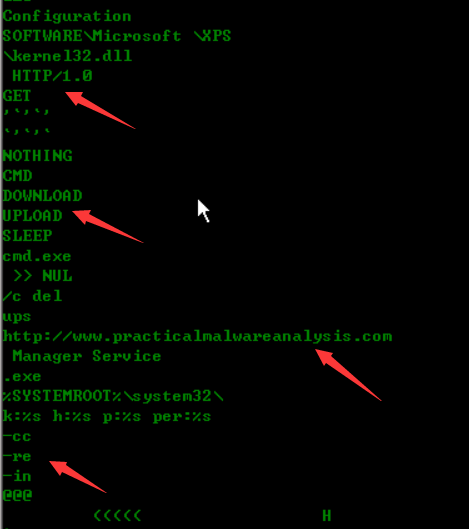
PEid查看导入表，在kernel32.dll中有关于获取系统目录，以及文件创建、写入、拷贝、删除等的操作。



在advapi32.dll中有关于service服务的操作，调用ws2\_32.dll会用到联网操作。



使用strings查看字符，发现有疑似GET访问某个站点www.pracxxx.com，并且有-cc -re -in等疑似后门命令。



运行程序，发现在PE中监控到exe调用子进程cmd.exe并且退出，而Lab 03-04.exe会自动删除。



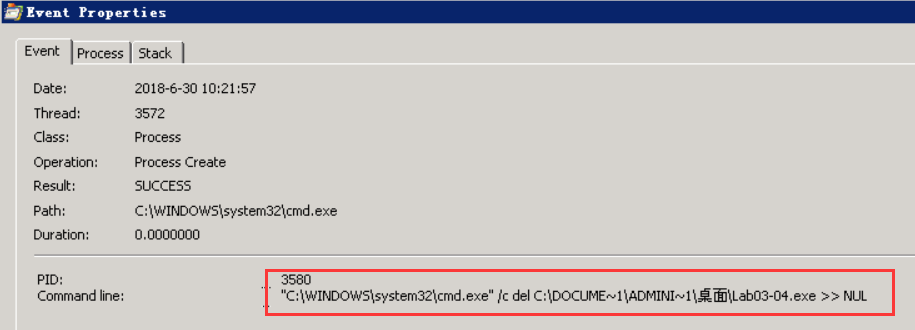
添加一个Filter。



创建了一个cmd.exe进程。

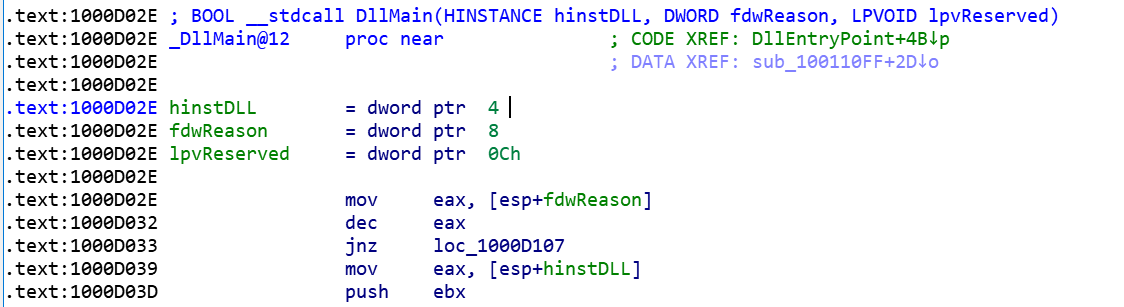


此cmd的命令就是调用del删除自身，实现自毁。

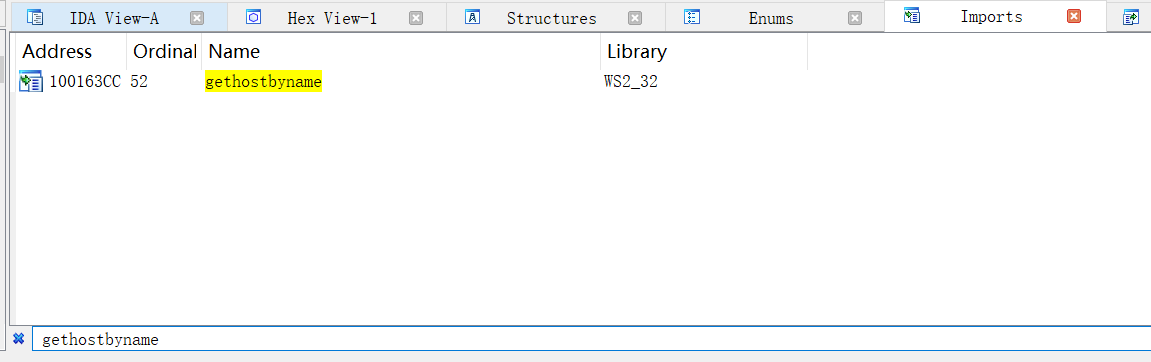


## 3.3 Chapter\_5L

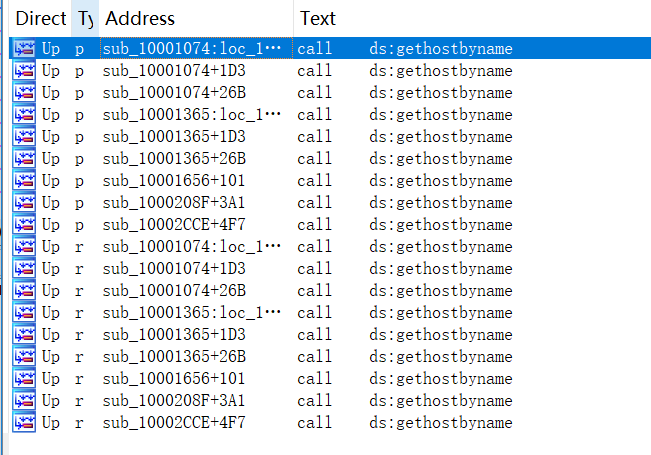
1. 直接用IDAPro打开.dll文件,就直接来到Dllmain处,可以知道地址为0x1000D02E。



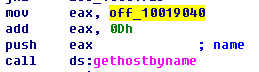
1. 在import窗口中搜索一下gethostbyname函数。



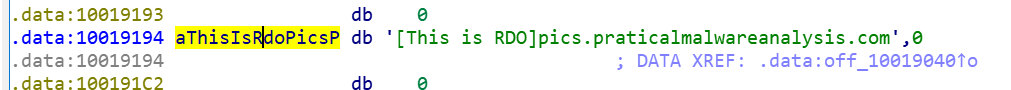
1. 双击进去按快捷键”X”查看交叉引用，可以发现出现了5个不同的函数调用了它。



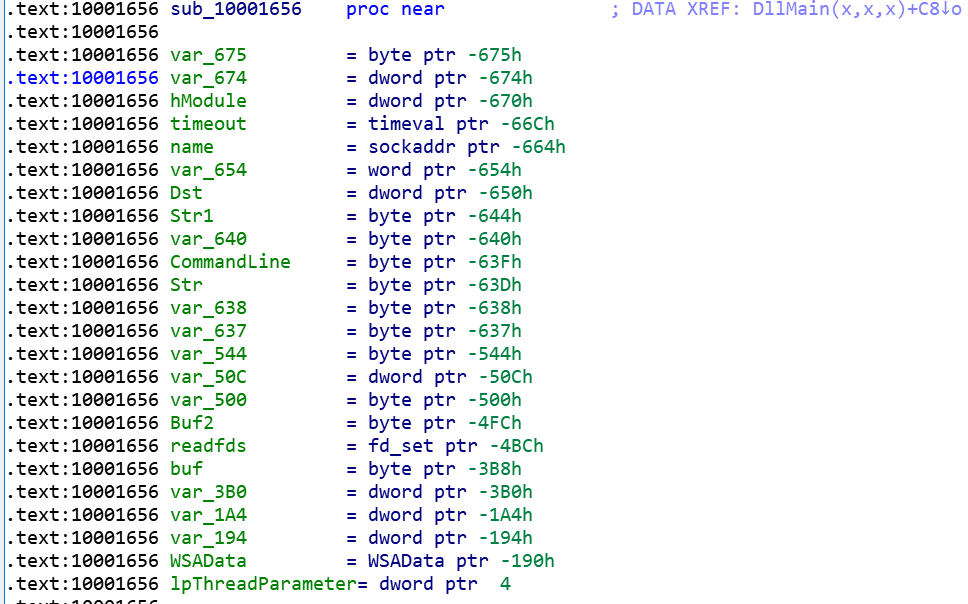
1. 在上题的交叉引用表中，找到0x10001757(sub\_10001656+101)双击到达代码处,分析如下,0x100019040处的全局变量地址赋值给eax,eax+0dh得到要请求的DNS名字,再压入栈作为gethostbyname的参数,所以双击查看10019040处代码,再双击跳转到1001994处看到字符串,[This is RDO]pics.praticalmalwareanalysis.com,去掉前头0dh个字符得到pics.praticalmalwareanalysis.com。



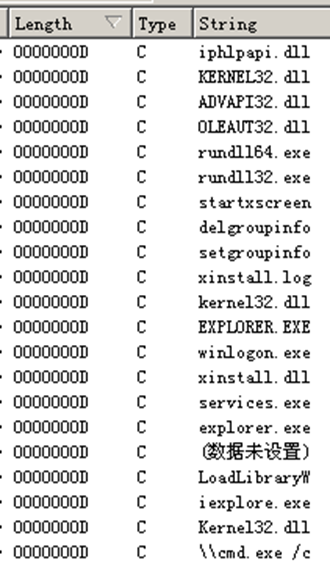




1. 按G跳转到0x10001656往上翻看,得到下图,分析可得,有23个局部变量一个(最后一个)参数,大于ebp的是参数,小于的是局部变量。

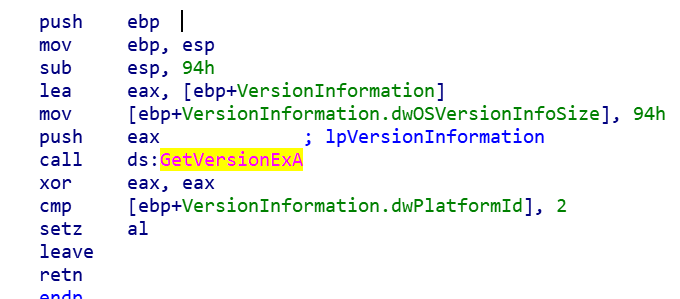


1. cmd.exe /c加一个转移字符,再加一个结束符一共13个字符,在setup里面设置一下最小长度为13,再按length低到高排列可以快速找到,双击到达位置.10095B34处。

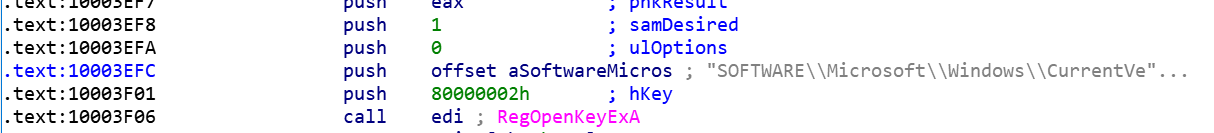


1. 在上题处选中该局部变量,然后按x查看那个地方调用这个局部变量,可以发现只有 一个地方,双击跳转到该处。可以看到下面有个recv应该是接受远端发来的命令,然后下面一串的命令比较看运行哪一个命令,可以推测这是一个远程shell会话。
2. 在0x100101c处选中该局部变量按x可以看到就三个地方调用它,其中只有一个是mov指令其他都是比较,所以双击查看该mov指令,可以看到局部变量给赋予从函数10003695返回的值.再双击查看sub\_10003695处的代码,可以看到如果得到的ID==2则al设为1否则为0,为2说明VER\_PLATFORM\_WIN32\_NT win32nt平台



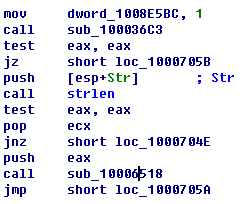


1. 在10010444处,成功则执行sub\_100052A2函数双击跳转到该函数位置查看代码,看到两个关键的函数调用,打开一个注册表项成功跳转到10005309处,失败则关闭返回的注册表句柄,再看10005309处代码,还是几个关键函数调用,regwueryvalue和sprintf和100038EE,双击查看100038EE可以知道这是一个send函数,那么很明显了,得到某注册表值并且发送给某个远端.





1. 看导出表,找到PSLIST,双击查看几个关键call调用,一个个分析,第一个sub\_100036c3,双击查看可以知道同样是一个判断操作系统是否是win2000以上的函数dwpaltformid==2,且dwMajorVersion>5。



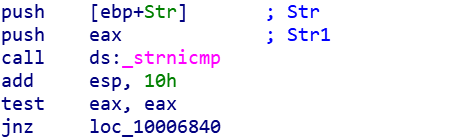
strlen函数看str参数是否为空,如果为空调用sub\_10006518函数,否则跳转1000704E处调用sub\_1000664c函数。

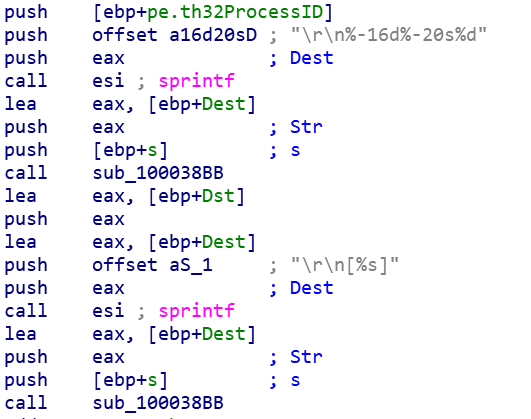
现在看sub\_1000664c函数查看函数的调用右键选择xrefs from可以看到几个关键函数

Getmodulfilename,strcmp,process32next process32first

再看函数流程图具体分析可以知道,函数列出并发送所以str指定的模块名字的进程流程图分析关键点截图。

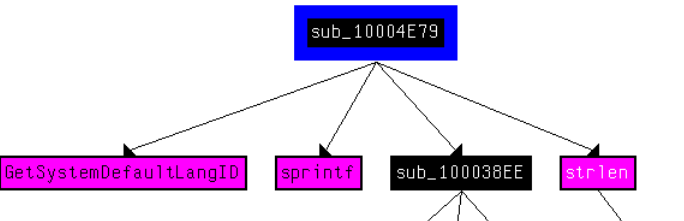






再看10006518同样的方法发现他们很相似,用到的函数基本相同,但是你会发现10006518少了strcmp函数相对于1000664c,可以知道因为没有指定str,所以程序列出所以进程信息,并发送.这就是PSLIST函数作用.正如他的函数名所暗示的一样。

1. xrefs from截图如下:可以看到关键函数send,那么他send什么呢?看getsystemdefaultlangid和sprintf可以知道send的是和系统缺省语言ID,故命名为send\_langID。



1. 直接右键xrefs from只能看到密密麻麻一片黑数不清多少个函数调用,选择view graphs user xrefs chart选择深度为1,为2的时候已经多的看不清楚了。



1. 查看关键代码知道参数eax决定睡眠多长,可以看到一系列对eax的操作查看初始10019020处的数据得到30,30乘以3e8h(1000)得到30000毫秒





1. #define AF\_INET 2协议域

#define SOCK\_STREAM 2 socket类型tcp

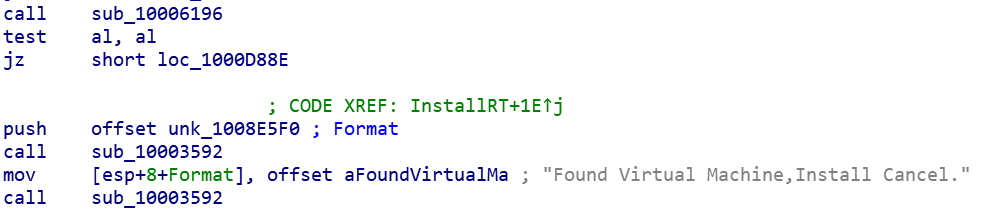
#define IPPROTO\_TCP 6tcp协议



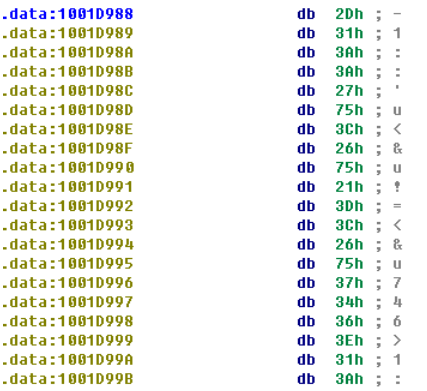


1. 选择searsh text 选择find all,需要一点时间,双击到达100061DB处查看代码,存在VMXh,再看交叉引用选择一个双击查看代码,可以看到关键字符串输出。

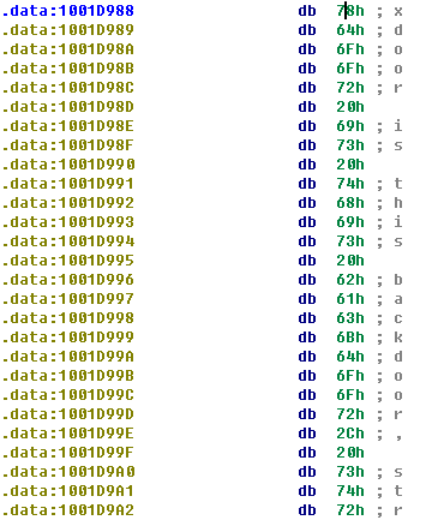




1. 未识别16进制数



1. 数值改变



1. 按A看到字符串内容如下

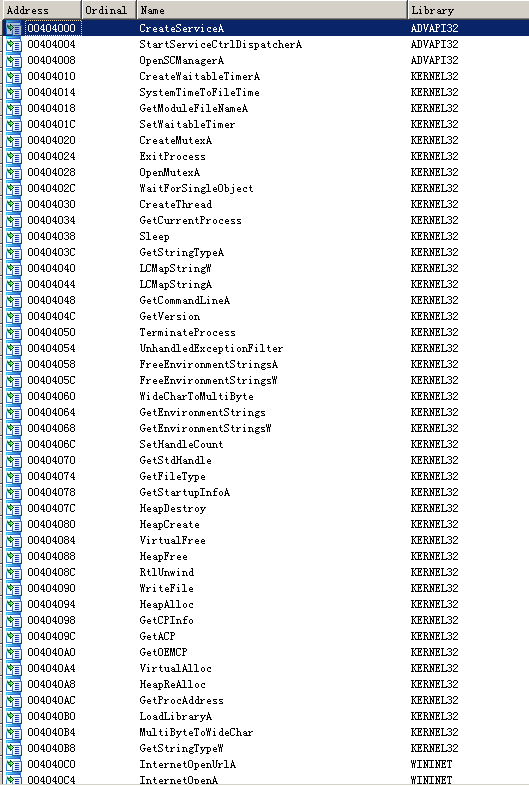
xdoor is this backdoor, string decoded for Practical Malware Anal

## 3.4 Chapter\_7L

### 3.4.1 Lab07-01.exe

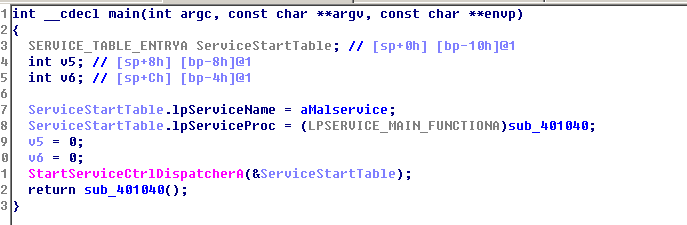
#### 1. 使用IDA分析整个程序

通过IDA分析整个程序的逻辑与疑点，先打开导入函数表，大致浏览程序的功能。



发现了一个CreateServiceA，StartServiceCtrlDispatcherA，OpenSCManagerA 等等有关于系统服务的函数，说明该程序具有操作系统的操作，InternetOpenUrlA，InternetOpenA函数用于打开一个网址，CreateMutexA，OpenMutexA用于创建和打开互斥锁。

#### 2. 使用IDA分析main函数



这个main()函数是用来创建一个服务列表，名字是MalService，主程序是sub\_401040()，通过StartServiceCtrlDispatcherA来执行这样的一个函数，我们跟进服务主程序去看看。

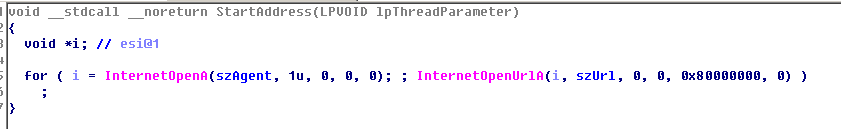


函数功能首先是打开互斥锁HGL345，如果打开成功就创建这样的互斥锁，如果不成功就退出程序，是保证程序只有一个程序运行。然后打开服务管理器，获取当前进程，找到当前进程的名字和文件目录Filename，创建基于这个随机获取到的文件目录来创建自己的服务，参数解释如下：



这个程序注册一个自己的服务开机运行，潜伏在程序运行时的当前目录的二进制文件中。

继续向下分析，发现是设定一个时间2100/0/0 00:00:00，然后获取系统时间，如果到了这个时间，那么就船舰20个线程，每个线程都是从StartAddress开始执行，我们进入这个StartAddress函数：



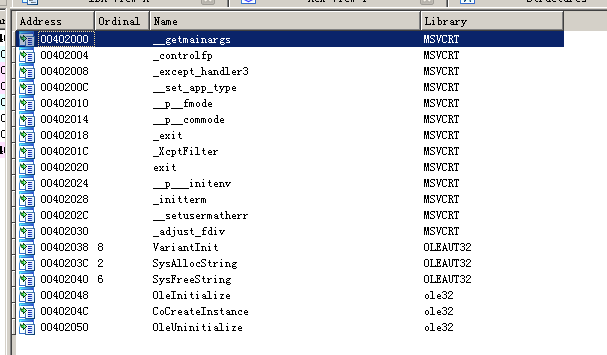
发现是打开Internet版本是Internet Explorer 8.0，然后打开网址http://www.malwareanalysisbook.com，字符串参考列表如下：



其功能是将自己写进开机启动中，到了一个指定的时间之后，打开20个线程同时访问该网站，估计是将病毒植入抓到的肉鸡中，然后指定一个时间进行DDOS攻击。

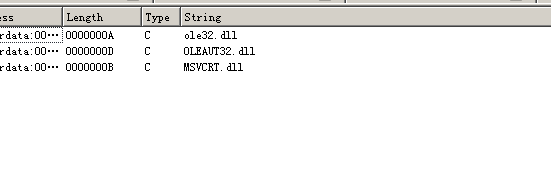
### 3.4.2 Lab07-02.exe

先查看函数导入表：



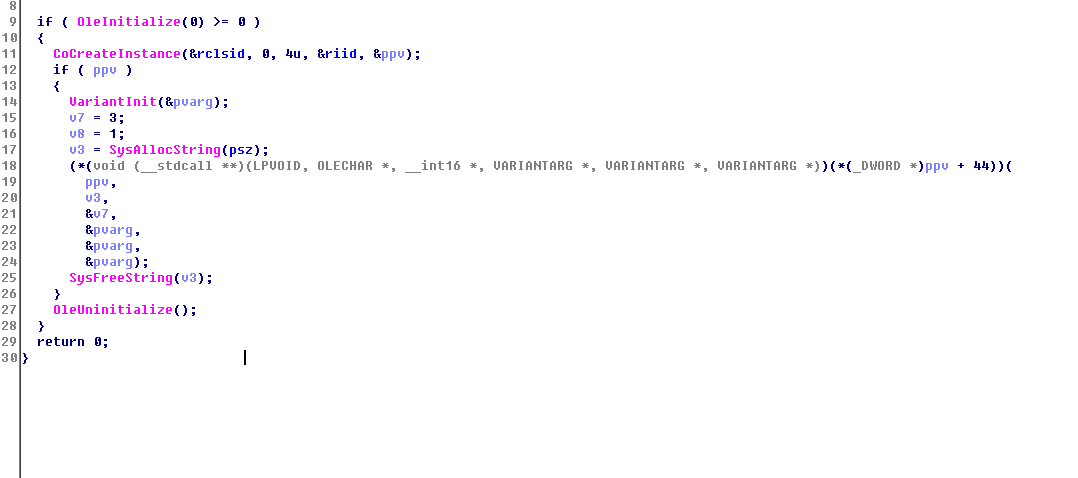
发现它的函数导入并不多，OleInitialize用于创建ole初始化对象，CoCreateInstance用于创建一个Com实例，OleUninitialize用于删除一个ole对象，上面的SysAllocString和SysFreeString用于字符串的操作。

我们再看看字符串列表：





看到了一个网址，可能有用来打开这个网址的操作，进入main函数看一看：



大概是首先创建实例，如果创建成功了，那么就运行ppv+44处函数指针指向的函数，其中v3是分配了一个字符串，跟进去看一看：



发现就是网址，我们把这个网址改一下名字，然后查询old函数的操作，发现这个CoCreateInstance函数的参数如下：



猜测该程序会打开浏览器导航，然后释放变量，关闭对象，结束main。

# 四、实验总结

通过本次实验，我学习了恶意代码分析的基本方法与步骤，加深了对恶意代码的了解，并能够使用多种安全工具对恶意程序进行测试。学会了使用工具分析文件的hash值，字符串，函数表，函数库依赖关系。 另外还可以运行恶意代码用工具监控恶意软件对文件，注册表，网络，进程的访问与操作。

# 五、参考文献

1. Hansimov. 恶意代码分析实战. https://hansimov.gitbook.io/malware-analysis/part1-basic-analysis/ch03-basic-dynamic-analysis
2. damaoooo. 恶意代码分析笔记. https://damaoooo.github.io/tags/%E6%81%B6%E6%84%8F%E4%BB%A3%E7%A0%81%E5%88%86%E6%9E%90%E7%AC%94%E8%AE%B0/
3. Atom Kid. 逆向/安全. http://www.atomsec.org/%E5%AE%89%E5%85%A8/