

**《恶意代码分析与防治技术》课程实验报告**

**实验一**

****

学 院 网络空间安全学院

专 业 信息安全

学 号 2112060

姓 名 孙蕗

班 级 信息安全1班

[《恶意代码分析与防治技术》课程Lab1实验报告](#_Toc6578)

[一、 实验目的 3](#_Toc30459)

[二、 实验原理 3](#_Toc8086)

[三、 实验过程 3](#_Toc7023)

[（一） Lab1-1 3](#_Toc3611)

[（二） Lab1-2 9](#_Toc10221)

[（三） Lab1-3 13](#_Toc11180)

[（四） Lab1-4 16](#_Toc15732)

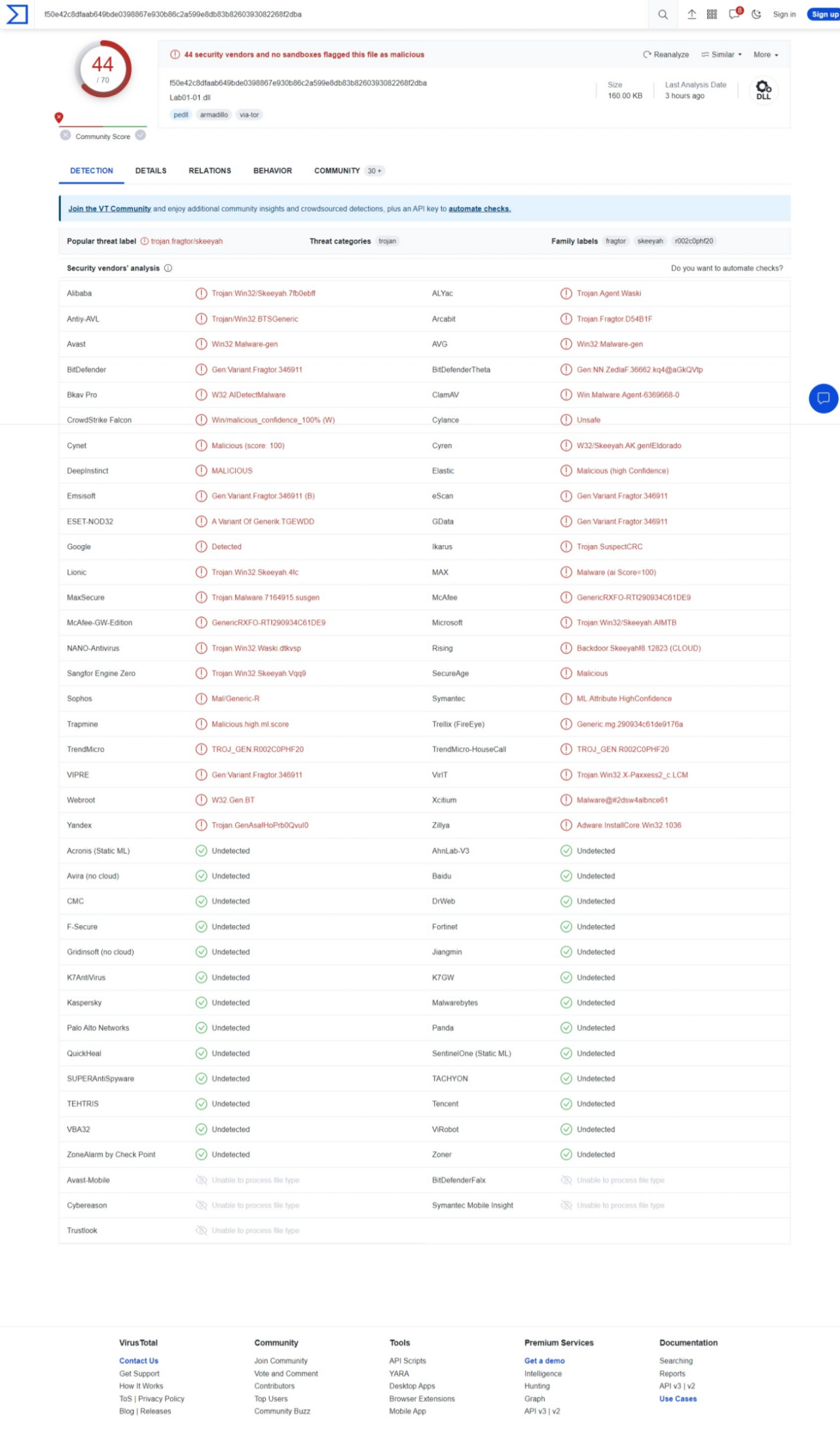
[（五） Lab1-5 22](#_Toc23262)

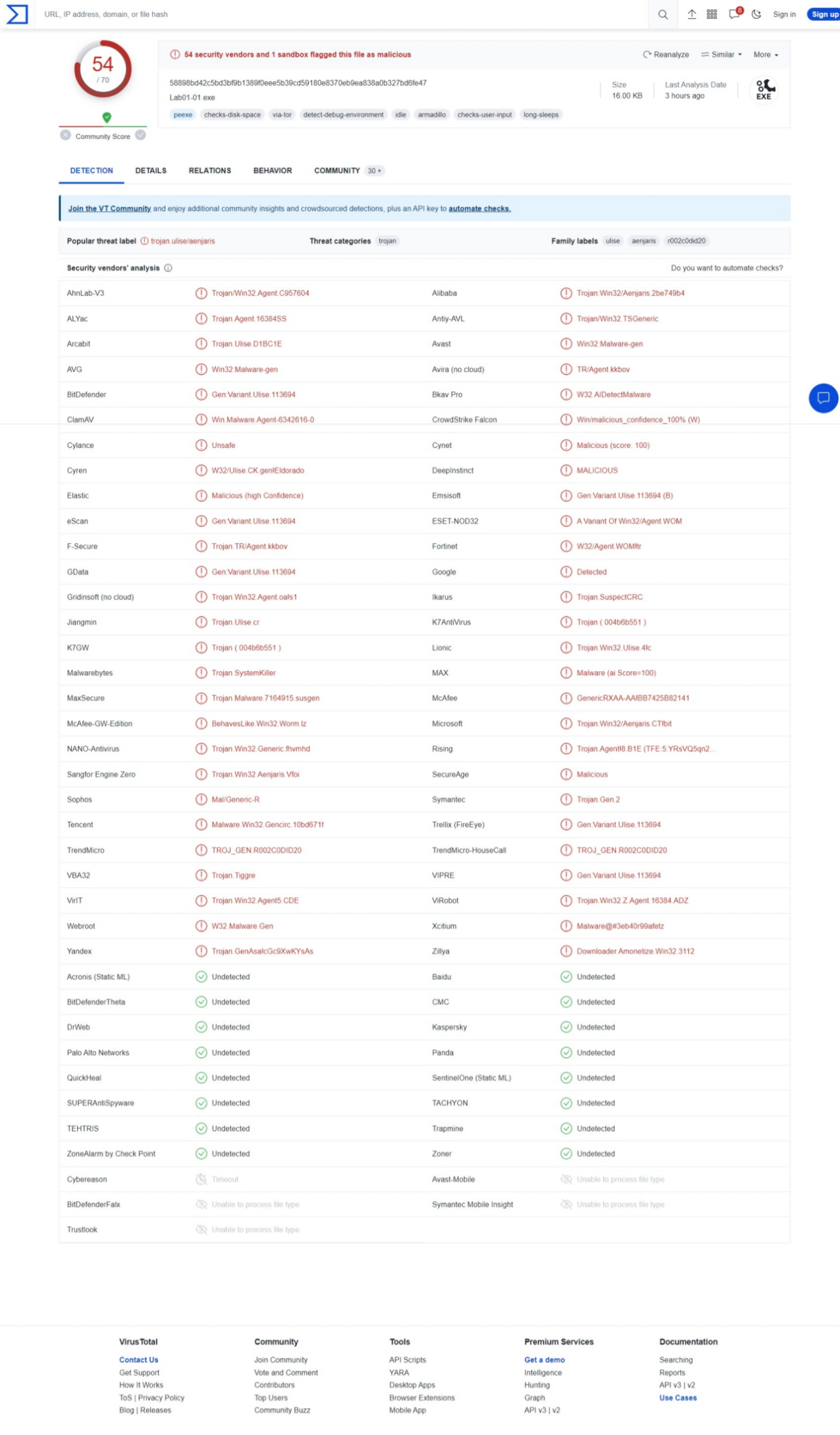
[四、 实验结论及心得体会 26](#_Toc12795)

1. **实验目的**

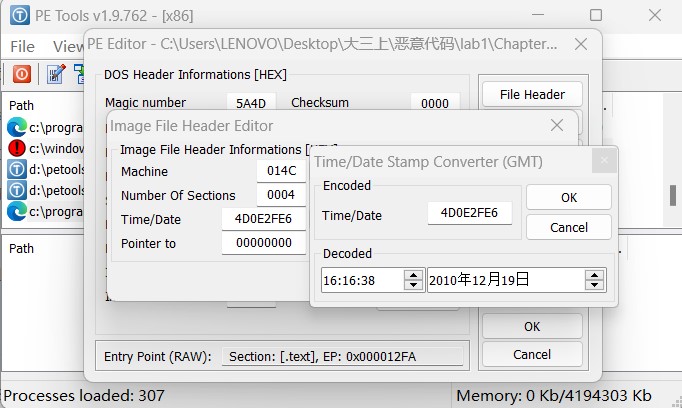
使用Lab01-01.exe、Lab01-02.exe、Lab01-03.exe、Lab01-04.exe和Lab01-01.dll文件，使用第一章描述的工具和技术分析上述文件并获取关于这些文件的信息。

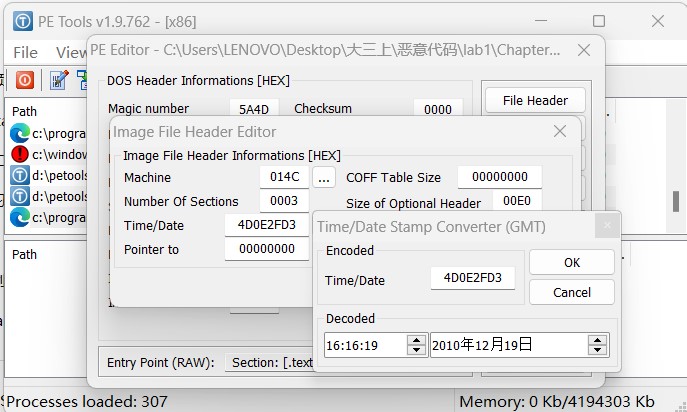
1. **实验原理**
2. VirusTotal(http://www.virustotal.com/)这样的网站允许上传一个文件，并将调用多个反病毒引擎来进行扫描。VirusTotal网站还会生成一份报告，其中提供了所有引擎对这个样本的识别情况、标识这个样本是否恶意、恶意代码名称，以及其他额外信息。
3. Strings程序可以从一个可执行程序中搜索ASCII和Unicode字符串时，它将忽略上下文和格式，所以它将分析任何文件类型，并从整个文件中检测出可打印字符串(这也意味着，它也会识别出实际上并非真正字符串的一些字符序列)。Strings程序搜索三个或以上连续的ASCI或Unicode字符，并以终结符结尾的可打印字符串。
4. 可以使用PEiD来检测加壳器的类型，或是用来链接应用程序的编译器类型。
5. 相关脱壳工具可以对使用相关技术加壳的恶意代码进行脱壳。
6. Dependency Walker工具支持列出可执行文件的动态链接函数。
7. PEview可以用来分析PE文件相关信息
8. Resource Hacker工具可以查看资源节。
9. **实验过程**
10. **Lab1-1**
11. **将文件上传至http://www.VirusTotal.com/进行分析并查看报告。文件匹配到了已有的反病毒软件特征吗?**





1. **这些文件是什么时候编译的？**

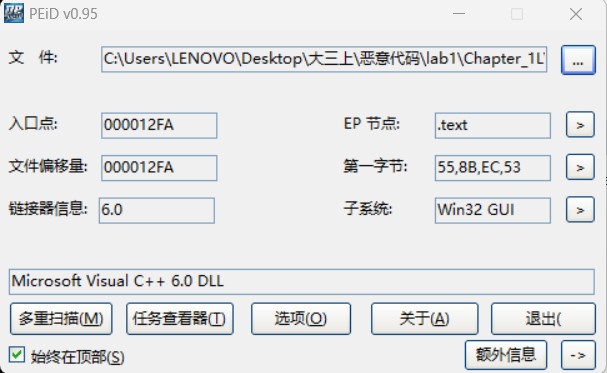
****

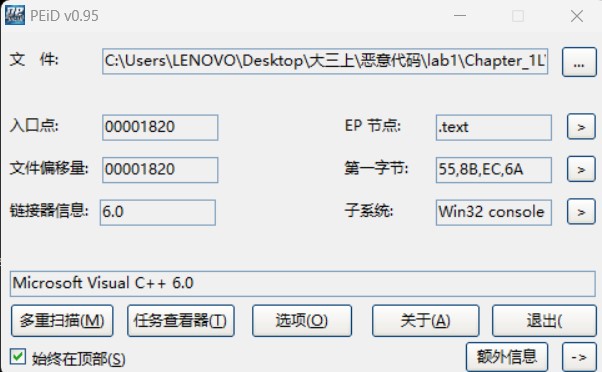
****

使用PEtools工具来打开文件。

Lab-01-01.exe 编译时间是2010年12月19日 16:16:19，Lab-01-01.dll 编译时间是2010年12月19日 16:16:38。这两个文件都是在2010年12月19日被编译的，两者编译时间在1分钟以内，基本证实了这两个文件同属一个恶意代码包。

1. **这两个文件中是否存在迹象说明它们是否被加壳或混淆了?如果是，这些迹象在哪里?**

****

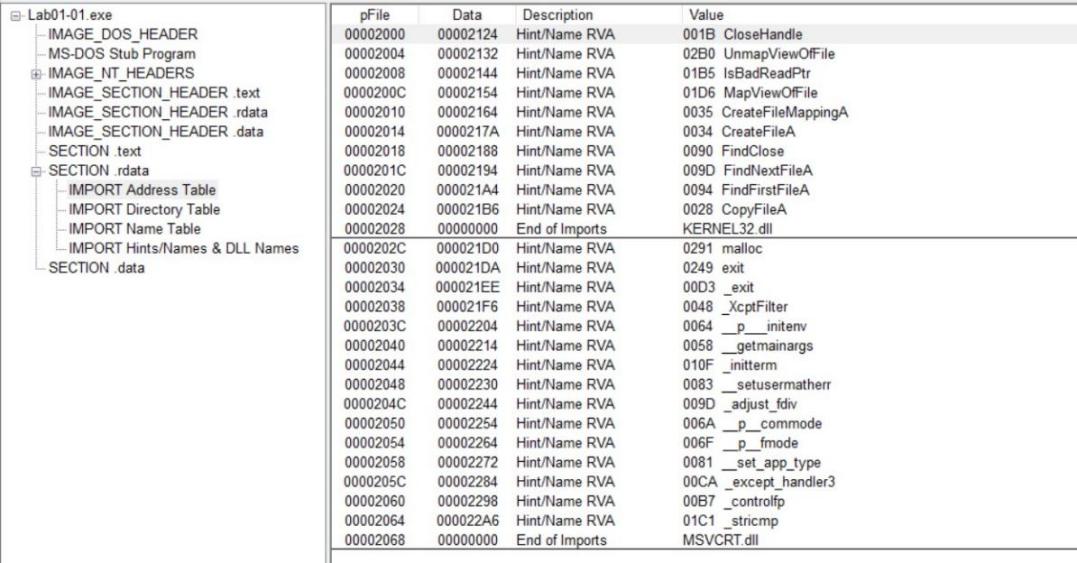
****

这两个文件的导入表数量都很少，也都有着适当大小的良好组织的文件节。

PEiD工具标记为未加壳的代码，PEiD正常检测出.dll和.exe的编译环境，EP段是正常的.text，并且是由Microsoft Visual C++编译的。

这两个文件并没有被加壳或混淆过的迹象。

1. **是否有导入函数显示出了这个恶意代码是做什么的?如果是，是哪些导入函数?**

****

****

使用PEview打开两个文件。

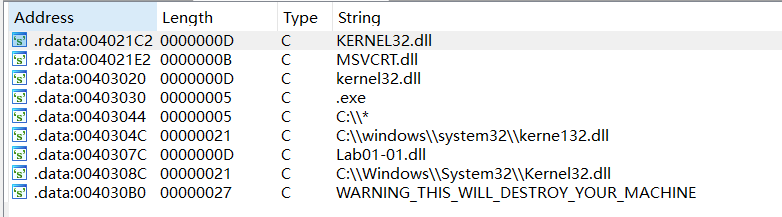
Lab01-01.dll导入函数有CreateProcess（创建新进程及其主线程）、Sleep（计算机程序进程，任务或线程进入休眠）、OpenMutexA（创建互斥量）、WS2\_32.dll（用来支持internet和网络应用程序的使用）等。这个程序很可能会创建新的进程，并且调用了WS2\_32.dll存在的联网功能。

Lab01-01.exe导入函数有FindFirstFileA（搜索目录中具有与特定名称或部分名称匹配的名称的文件或子目录）、FindNextFileA（继续从先前调用 FindFirstFile、FindFirstFileEx或FindFirstFileTransacted 函数进行文件搜索）和CopyFileA（将现有文件复制到新文件），CreateFileMappingA（为指定文件创建或打开命名或未命名的文件映射对象）等。

这个程序很有可能在搜索文件系统、打开修改文件、并复制文件等。

1. **是否有任何其他文件或基于主机的迹象，让你可以在受感染系统上查找?**

用IDApro查看.exe文件字符串



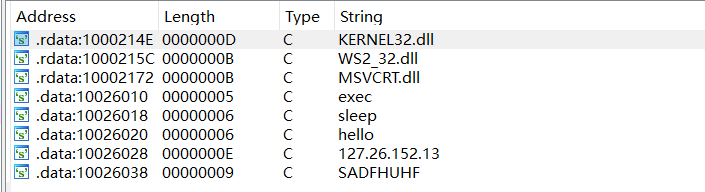
注意到kernel32.dll和kerne132.dll的区别（1和l）。

kernel32.dll文件应该是想冒充混淆为Windows的系统文件kernel32.dll。

kernel32.dll可以作为一个基于主机的迹象来发现恶意代码感染。

1. **是否有基于网络的迹象，可以用来发现受感染机器上的这个恶意代码?**

用IDApro查看.dll文件字符串



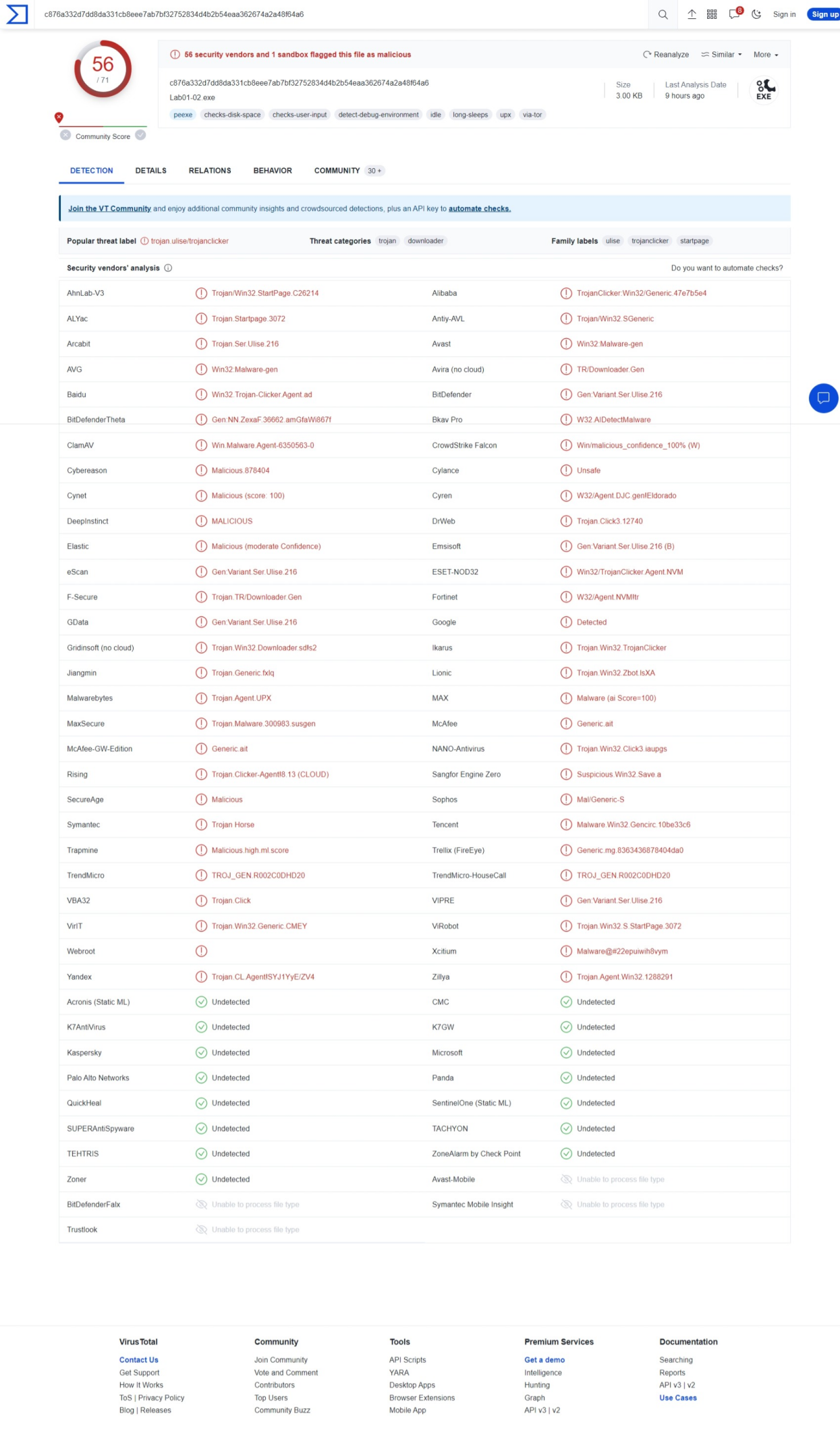
之前已发现从kernel32.dll导入了CreateProcess和Sleep函数，而这两个函数普遍在后门程序中使用。exec字符串可能是通过网络来给后门程序传送命令，让它通过CreateProcess函数运行一个程序。sleep字符串可能用于命令后门程序进入休眠模式。dll文件中还包含一个私有子网IP地127.26.152.13的字符串。

上述内容的存在预示着极有可能休眠后门连接远程。这个ip地址可用于识别基于网络的恶意代码感染迹象，可用于识别恶意代码。

1. **你猜这些文件的目的是什么?**

.dll文件可能是一个后门，.exe文件是用来安装与运行DLL文件的。

1. **Lab1-2**
2. **将Lab01-02.exe文件上传至http://www.VirusTotal.com/进行分析并查看报告。文件匹配到了已有的反病毒软件特征吗?**



1. **是否有这个文件被加壳或混淆的任何迹象?如果是这样，这些迹象是什么?如果该文件被加壳，请进行脱壳，如果可能的话。**

使用PEiD检测，存在UPX壳。



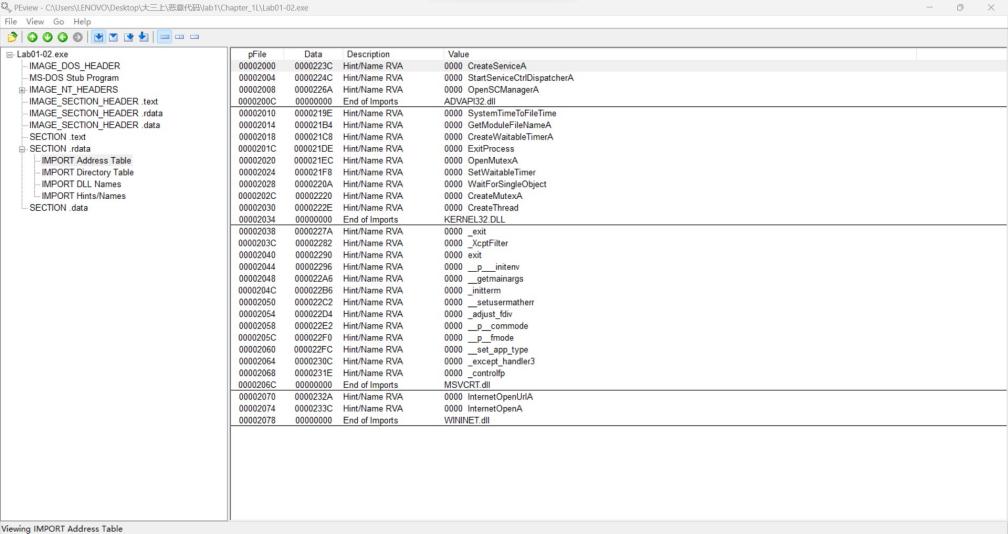
使用FUPX进行脱壳。



使用PEiD检测，已脱壳。



1. **有没有任何导入函数能够暗示出这个程序的功能?如果是，是哪些导入函数，它们会告诉你什么?**

****

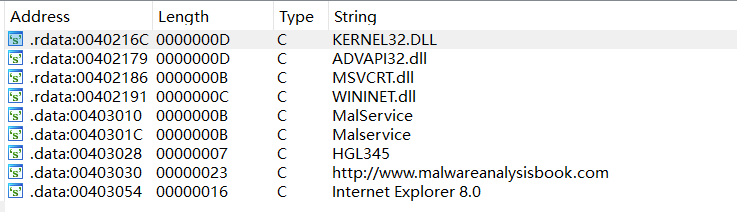
使用PEview打开已脱壳的.exe文件，kernel32.dll和msvcrt.dll,而这些函数几乎被每个程序都导入，关于恶意代码的信息不多。

advapi32.dll的导入函数有CreateServiceA（创建服务对象并将其添加到指定的服务控制管理器数据库）、StartServiceCtrlDispatcherA（将服务进程的main线程连接到服务控制管理器）、OpenSCManagerA（与指定计算机上的服务控制管理器建立连接，并打开指定的服务控制管理器数据库）函数，这三个函数一般可用于创建互斥、进程、服务。这个恶意代码很可能会创建一个服务。

从wininet.dll的导入函数有InternetOpenUrlA（打开由完整的 FTP 或 HTTP URL 指定的资源）、InternetOpenA（初始化应用程序对 WinINet 函数的使用）函数，这两个函数一般可用于进行联网操作，启动一个链接。这个恶意代码很可能将会进行联网操作。

综上所述，这个恶意代码很可能会创建一个服务并进行联网操作。

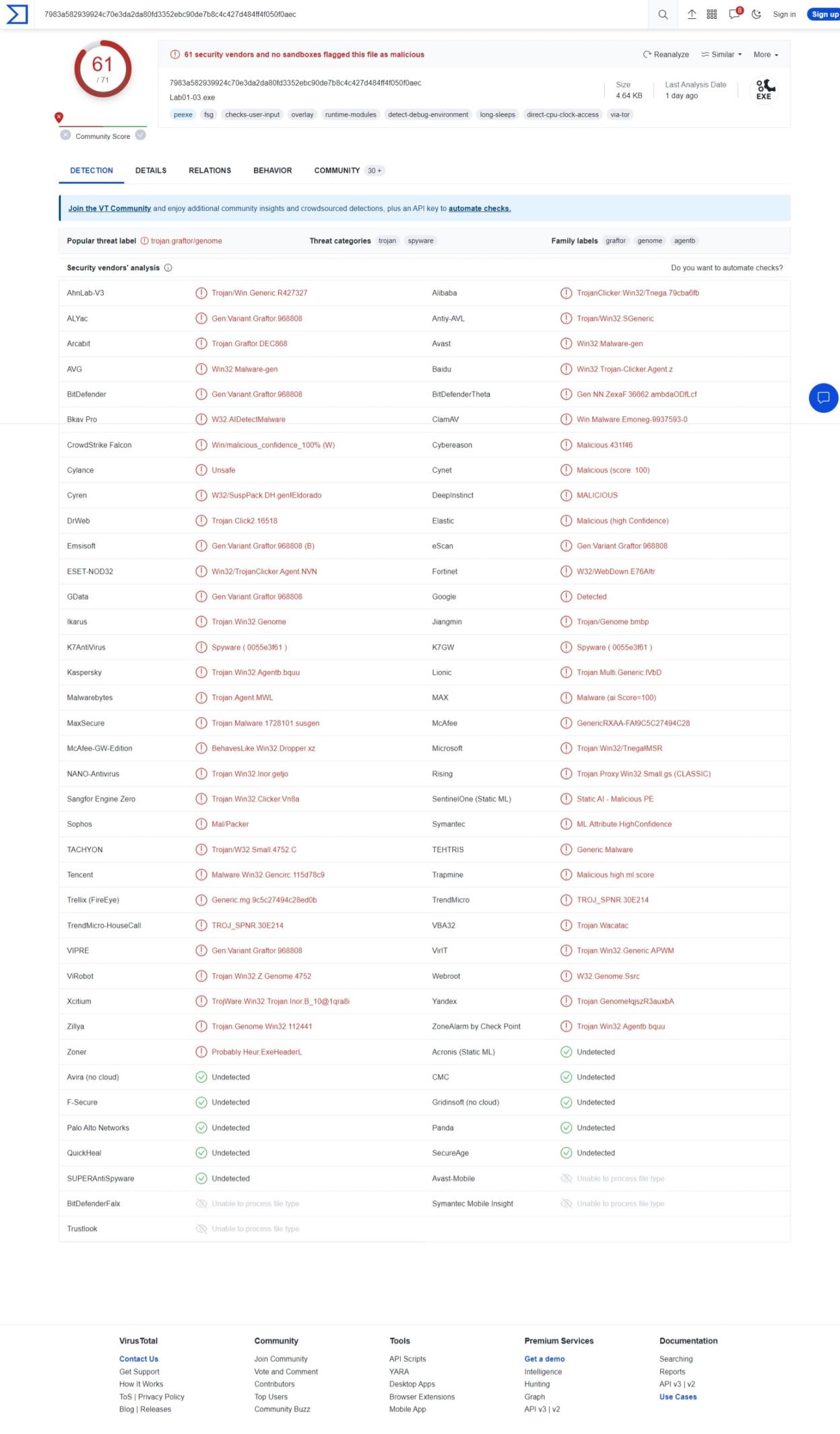
1. **哪些基于主机或基于网络的迹象可以被用来确定被这个恶意代码所感染的机器?**



用IDApro查看.exe文件字符串，看到了所创建的服务名称Malservice。还有www.malwareanalysisbook.com，这可能是InternetOpenURL函数中所打开的URL。

可以通过名为Malservice的服务,通过到http://www.malwareanalysisbook.com的网络流量，来检查被恶意代码感染的主机。

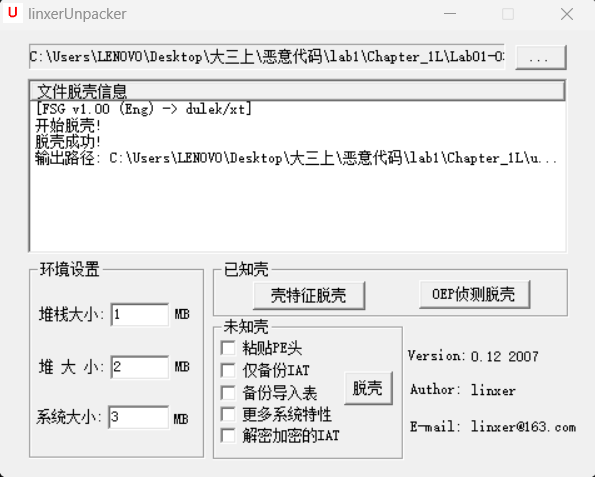
1. **Lab1-3**
2. **将Lab01-03.exe文件上传至htp://www.VirusTotal.com/进行分析并查看报告。文件匹配到了已有的反病毒软件特征吗?**

****

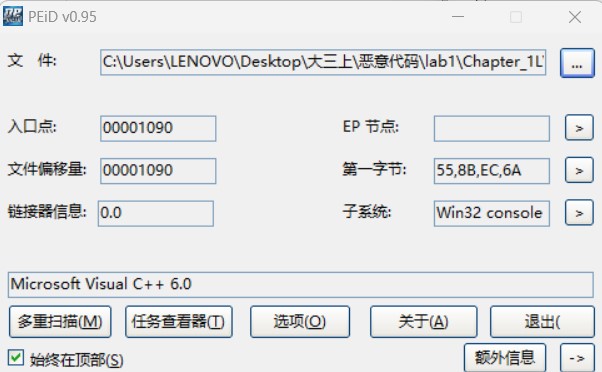
1. **是否有这个文件被加壳或混淆的任何迹象?如果是这样，这些迹象是什么?如果该文件被加壳，请进行脱壳，如果可能的话。**

****

使用PEiD检测，可能存在FSG壳

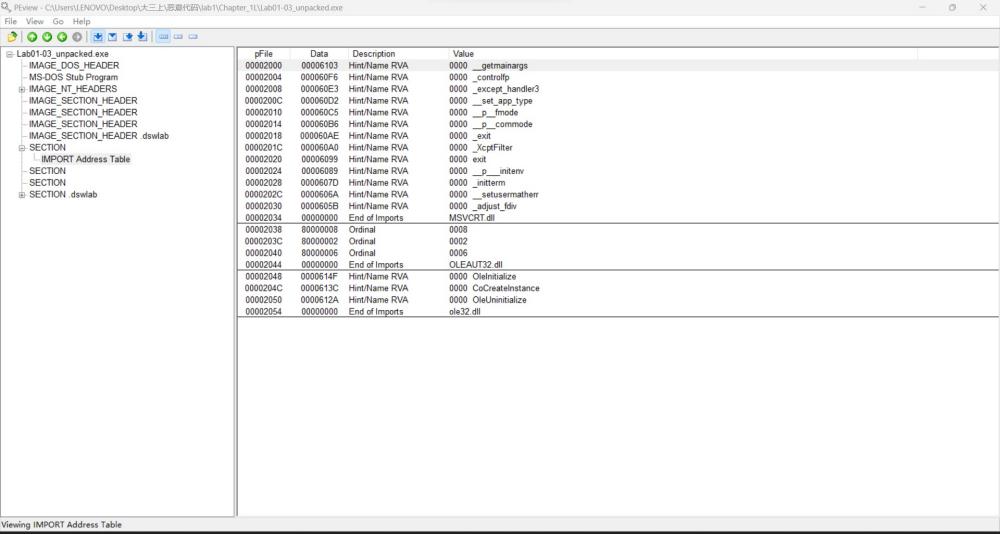


使用linxerUnpacker进行脱壳。



使用PEiD检测，已脱壳。

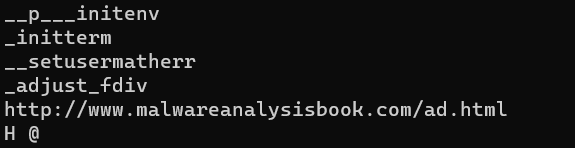
1. **有没有任何导入函数能够暗示出这个程序的功能?如果是，是哪些导入函数，它们会告诉你什么?**

****

使用PEview打开脱壳后的.exe文件,发现OleInitialize（初始化当前单元的COM库）、CoCreateInstance（创建并默认初始化与指定 CLSID 关联的类的单个对象）、OleUninitialize（关闭单元上的 COM 库，并释放相关对象及资源）三个函数。

推测该程序可以通过COM接口访问一个网址。

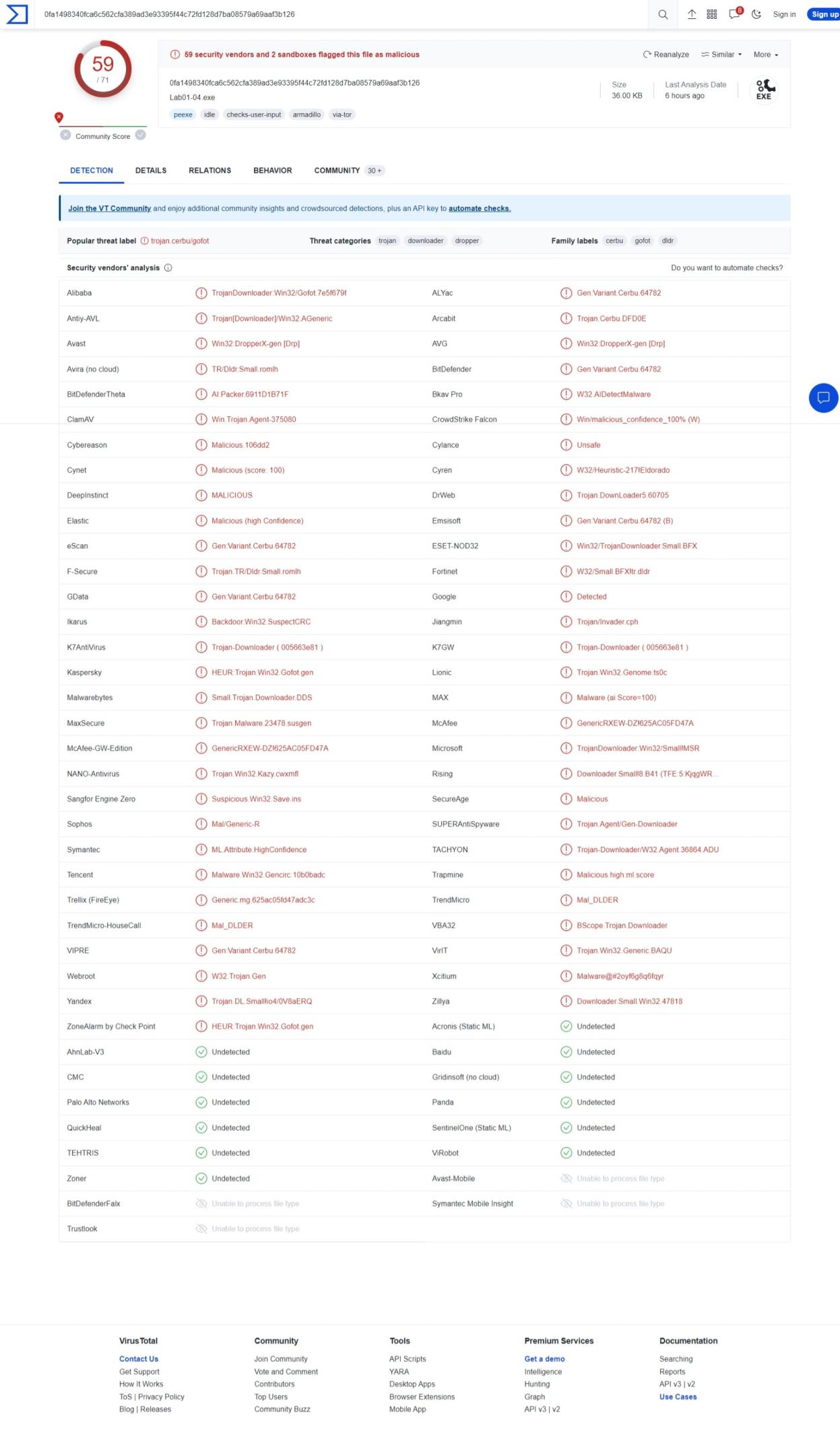
1. **有哪些基于主机或基于网络的迹象，可以被用来确定被这个恶意代码所感染的机器?**



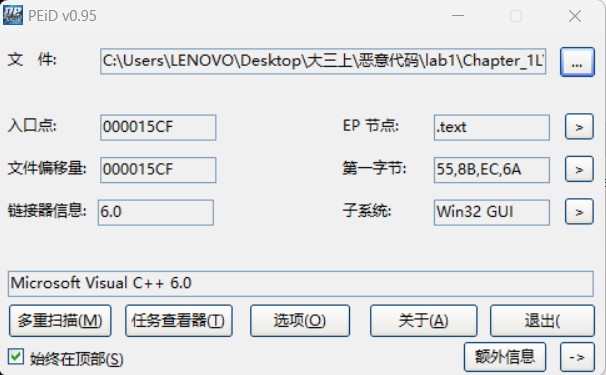
使用strings.exe查看Lab01-03\_unpacked.exe字符串信息，可以看到一个网址，因此该恶意代码应该是利用ole相关组件实现了对该网址的访问，监视网络行为。

若有访问该网址的行为，则该机器应已被恶意代码感染。

1. **Lab1-4**
2. **将Lab01-04.exe文件上传至http://www.VirusTotal.com/进行分析并查看报告。文件匹配到了已有的反病毒软件特征吗?**

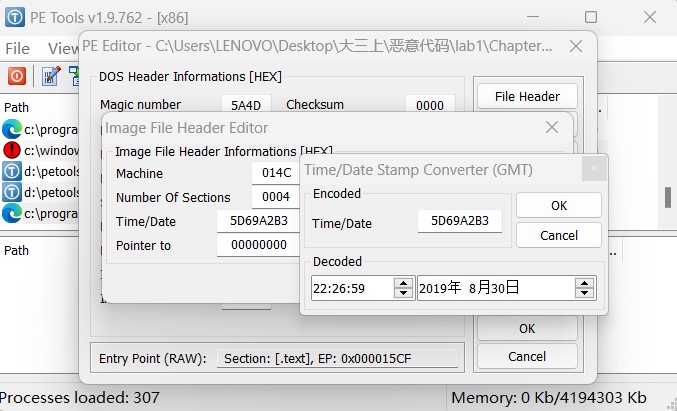
****

1. **是否有这个文件被加壳或混淆的任何迹象?如果是这样，这些迹象是什么?如果该文件被加壳，请进行脱壳，如果可能的话。**

****

没有迹象显示这个文件是被加过壳或混淆过的。

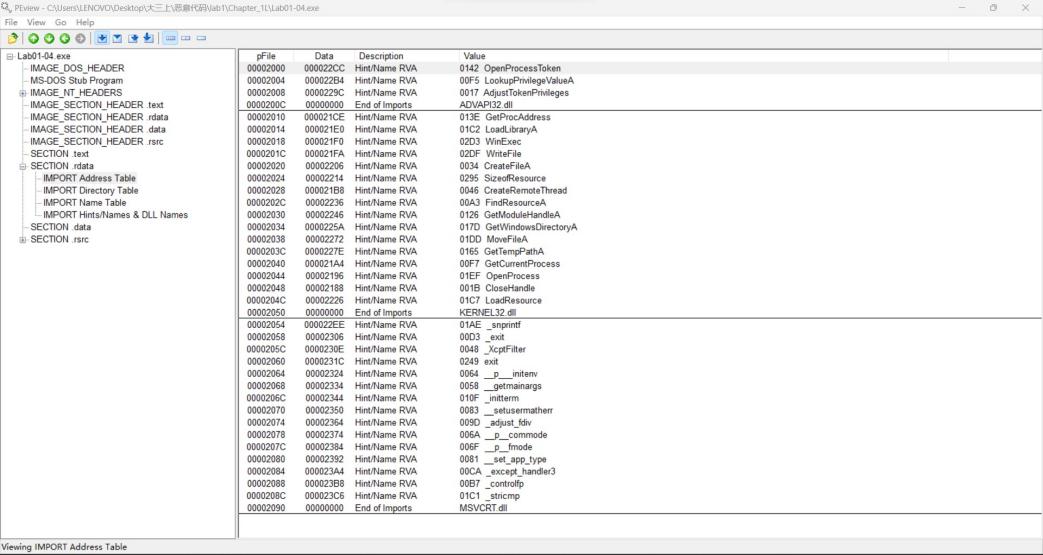
1. **这个文件是什么时候被编译的?**

****

使用PEtools工具来打开文件。

根据文件头的信息，这个文件是在2019年8月编译的。但推测该编译时间应该是伪造的，仅根据当前信息还不能确定这个文件到底是什么时候被编译的。

1. **有没有任何导入函数能够暗示出这个程序的功能?如果是，是哪些导入函数，它们会告诉你什么?**

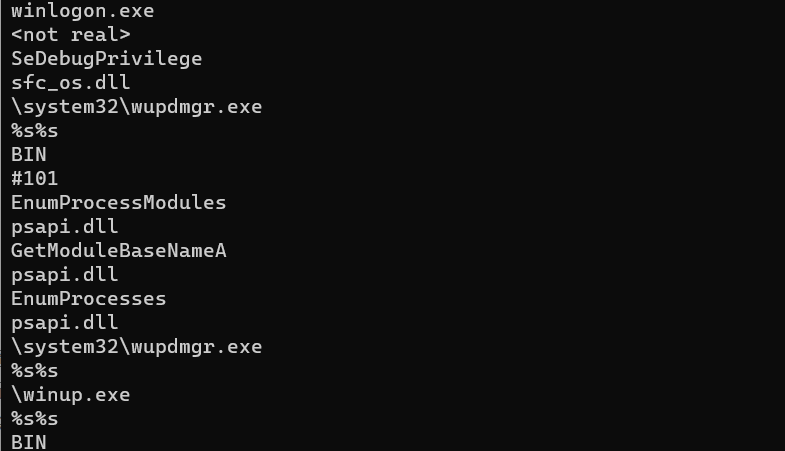
****

从advapi32.dll有3个导入函数，OpenProcessToken（打开与进程关联的访问令牌）、LookupPrivilegeValueA（检索本地唯一标识符 (LUID) 指定系统上用于本地表示指定特权名称）、AdjustTokenPrivileges（启用或禁用指定访问令牌中的特权），这3个函数可能和系统的权限有关。

从kernel32.dll中有16个导入函数，GetProcAddress（从指定的动态链接库DLL检索导出函数或变量的地址）、LoadLibraryA（将指定的模块加载到调用进程的地址空间中）、WinExec（运行指定的应用程序）、WriteFile（将数据写入指定的文件或输入/输出I/O设备）、CreateFileA（创建或打开文件或 I/O 设备）、SizeofResource（以字节为单位检索指定资源的大小）、CreateRemoteThread（创建在另一个进程的虚拟地址空间中运行的线程）、FindResourceA（确定指定模块中具有指定类型和名称的资源的位置）、GetModuleHandleA（检索指定模块的模块句柄）、GetWindowsDirectoryA（检索 Windows 目录的路径）、MoveFileA（移动现有文件或目录，包括其子级）、GetTempPathA（检索为临时文件指定的目录的路径）、GetCurrentProcess（检索当前进程的伪句柄）、OpenProcess（打开现有的本地进程对象）、CloseHandle（关闭打开的对象句柄）、LoadResource（检索可用于获取指向内存中指定资源第一个字节的指针的句柄）。

可以看出这个程序不仅涉及到系统权限，还涉及到文件的读写、资源的查找等。假设它试图访问使用了特殊权限进行保护的文件，这个程序从资源节中装载数据(LoadResource、FindResource、SizeofResource),并写一个文件到磁盘上(CreateFile、WriteFile),接着执行一个磁盘上的文件(WinExec),并调用GetWindowsDirectory将文件写入到了系统目录。

1. **有哪些基于主机或基于网络的迹象，可以被用来确定被这个恶意代码所感染的机器?**





用strings.exe查看Lab01-04.exe文件字符串，出现了很多dll，在系统目录下出现了一些exe文件。

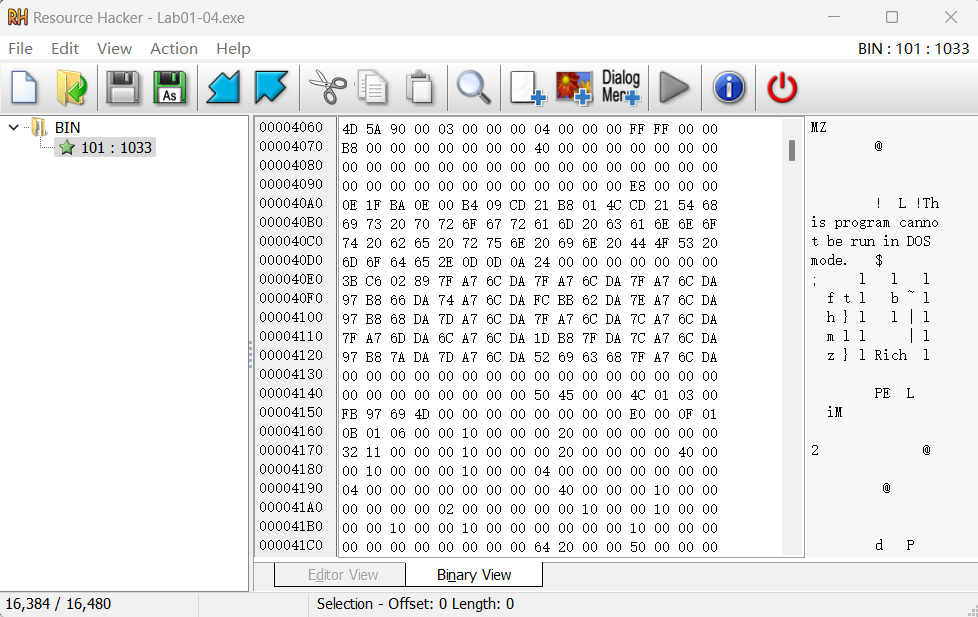
winlogon.exe是Windows NT 用户登陆程序，用于管理用户登录和退出；wupdmgr.exe是大多数 Windows系统的自动升级程序，但病毒木马也经常将自己伪装成这两个exe来运行。结合GetWindowsDirectory函数调用，表明恶意代码在C:\WindowslSystem32\wupdmgr.exe位置创建或者修改了一个文件。

winup.exe不是Windows系统自带的exe文件，而是至今仍被广泛使用的infecter病毒技术的被感染文件。推测该恶意代码的资源节中存放了winup.exe。

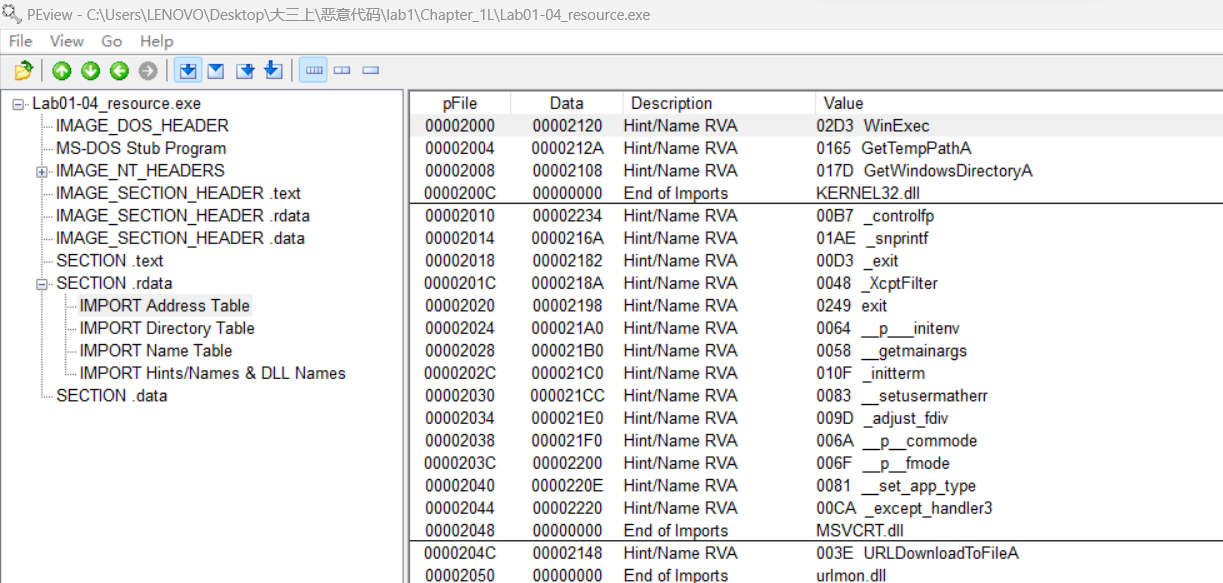
导入函数URLDownloadToFileA（从互联网下载并将其保存到文件，可以与winexec结合使用）和一个网址“http://www.practicalmalwareanalysis.com/updater.exe”表示这个exe文件将从该网址下载updater.exe。

综上所述，如果电脑中出现了winup.exe，在C:\Windows\system32文件夹下出现了wupdmgrd.exe，则该机器被恶意代码所感染。监视网络行为，如果有向“hxxp://www.practicalmalwareanalysis.com/updater.exe”发送请求，则该机器被恶意代码所感染。

1. **这个文件在资源段中包含一个资源。使用Resource Hacker工具来检查资源，然后抽取资源。从资源中你能发现什么吗?**



单击Action→Save \*.bin resource...,将文件存储为.exe格式，并在PEview中打开查看导入表。



kernel32.dll文件中有3个导入函数，WinExec（运行指定的应用程序）、GetTempPathA（检索为临时文件指定的目录的路径）、GetWindowsDirectoryA（检索 Windows 目录的路径）。urlmon.dll文件中有URLDownLoadToFileA（从互联网下载并将其保存到文件中）。

综上所述，嵌入文件通过访问一些网络函数，调用了一个由恶意下载器普遍使用的函数URLDownloadToFile和WinExec函数下载了updater.exe，并且还可能执行了下载到的文件。

1. **Lab1-5**
2. **对Lab1和Lab3的样本编写Yara规则。**

**Lab1：**

|  |  |
| --- | --- |
| 1 | rule Lab1 |
| 2 | { |
| 3 | meta: |
| 4 | description = "rules for Lab1 " |
| 5 | date = "202x/xx/xx" |
| 6 | strings: |
| 7 | $a = "kerne132.dll" wide ascii |
| 8 | $b = "127.26.152.13" wide ascii |
| 9 | $c = "http://www.malwareanalysisbook.com" wide ascii |
| 10 | $d = "wupdmgr" wide ascii |
| 11 | condition: |
| 12 | any of them |
| 13 | } |

**Lab3:**

|  |  |
| --- | --- |
| 1 | rule Lab3 |
| 2 | { |
| 3 | meta: |
| 4 | description = "rules for Lab3 " |
| 5 | date = "202x/xx/xx" |
| 6 | strings: |
| 7 | $a = "vmx32to64" wide ascii |
| 8 | $b = "serve.html" wide ascii |
| 9 | $c = "http://www.malwareanalysisbook.com" wide ascii |
| 10 | $d = "svchost" wide ascii |
| 11 | $e = "practicalmalwareanalysis.log" wide ascii |
| 12 | condition: |
| 13 | any of them |
| 14 | } |

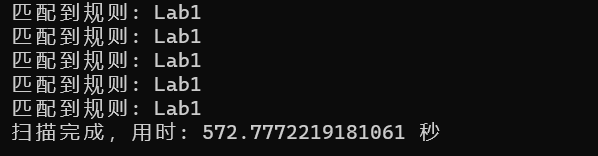
1. **使用自己的规则对自己电脑的C盘进行Yara引擎的扫描，记录扫描所用时间。**

下载并安装好yara、python后，使用pip install yara-python命令安装Yara Python模块。

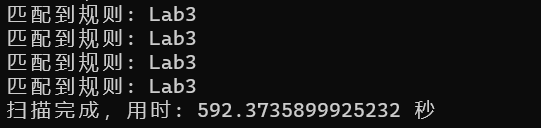
Python脚本：

|  |  |
| --- | --- |
| 1 | import os |
| 2 | import time |
| 3 | import yara |
| 4 |  |
| 5 | # 载入Yara规则 |
| 6 | rules = yara.compile('lab1.yara')# lab3使用的文件更换为lab3.yara |
| 7 |  |
| 8 | # 指定要扫描的目录 |
| 9 | target\_directory = 'C:\\' |
| 10 |  |
| 11 | start\_time = time.time() |
| 12 |  |
| 13 | matches = [] |
| 14 |  |
| 15 | # 遍历目录中的所有文件 |
| 16 | for root, dirs, files in os.walk(target\_directory): |
| 17 | for file in files: |
| 18 | file\_path = os.path.join(root, file) |
| 19 | try: |
| 20 | # 执行扫描 |
| 21 | file\_matches = rules.match(file\_path) |
| 22 | if file\_matches: |
| 23 | for match in file\_matches: |
| 24 | matches.append(match) |
| 25 | except Exception as e: |
| 26 | pass |
| 27 |  |
| 28 | end\_time = time.time() |
| 29 |  |
| 30 | # 输出匹配结果 |
| 31 | if matches: |
| 32 | for match in matches: |
| 33 | print(f"匹配到规则: {match.rule}") |
| 34 | else: |
| 35 | print("未发现匹配的规则") |
| 36 |  |
| 37 | # 计算并输出扫描时间 |
| 38 | scan\_time = end\_time - start\_time |
| 39 | print(f"扫描完成，用时: {scan\_time} 秒") |

Lab1扫描用时：



Lab3扫描用时：



1. **讨论哪些yara条件执行效率高，哪些yara条件执行效率低，以及如何改进那些执行效率低的yara条件。**
2. **执行效率高**
3. 使用静态规则
4. 使用必要的规则，精简规则的数目
5. 使用较少较短的字符串并且只在必要时才进行规则匹配
6. 条件简单，使用较少的逻辑运算符和条件嵌套
7. **执行效率低**
8. 使用动态规则
9. 使用规则的数量过大，且其中存在冗余的规则
10. 使用了大量较长的字符串进行规则匹配
11. 使用了大量逻辑运算符、通配符（如\*和？）、较为复杂的正则表达式（如具有多个嵌套、条件判断等）等使用过多复杂条件的规则
12. 扫描大型二进制文件，文件过大
13. **如何改进**
14. 相似的规则进行合并，减少规则数目
15. 简化规则条件，减少逻辑运算符等的使用
16. 减少不必要通配符和正则表达式的使用
17. 精确匹配范围，减少不必要的计算
18. 优化规则顺序，将最可能匹配的规则放在前面
19. 缓存已匹配的结果，减少重复运算
20. **实验结论及心得体会**
21. **实验结论**
22. Lab1-1实验结论： 通过VirusTotal和相关工具的分析，发现Lab01-01.exe和Lab01-01.dll是在2010年12月19日编译的，没有明显的加壳或混淆迹象，但存在对文件进行操作、创建新的进程并与互联网通信的迹象，可能是一个后门程序。
23. Lab1-2实验结论： Lab01-02.exe经过脱壳后，发现了UPX壳，还有与互联网通信的迹象，可能是一个恶意下载器。
24. Lab1-3实验结论： Lab01-03.exe存在FSG壳，使用脱壳工具后得到清晰代码。导入函数暗示它可以通过COM接口访问网址，可能涉及到下载和执行文件。
25. Lab1-4实验结论： Lab01-04.exe并没有加壳或混淆，但存在一些与系统权限和网络通信相关的导入函数，可能涉及到文件操作和下载。
26. Lab1-5实验结论： 编写了Yara规则并在C盘上进行了扫描，可以用于检测与恶意代码相关的特征。
27. **心得体会**
28. 通过本次实验，学习了如何使用不同的工具和技术来分析恶意代码，包括VirusTotal网站、PEview、脱壳工具等。这些工具对于恶意代码分析非常有用，可以帮助我们识别潜在的威胁。
29. 实验中还练习了如何观察恶意代码中的迹象，例如编译时间、导入函数、字符串等，这些信息可以帮助理解恶意代码的功能和行为。
30. 编写Yara规则帮助巩固了课堂的知识，可以帮助我们自动化恶意代码检测的过程，并及时发现潜在的威胁。
31. 学到了优化Yara规则的重要性，以提高执行效率。合并相似的规则、简化条件、减少不必要的通配符和正则表达式都是优化规则的有效方法。
32. 本次实验提供了有关恶意代码分析的基本知识和技能，使我能够更好地理解和应对潜在的安全威胁。