

**恶意代码课程实验报告**

**实验一：Lab1样本YARA规则编写**

****

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**

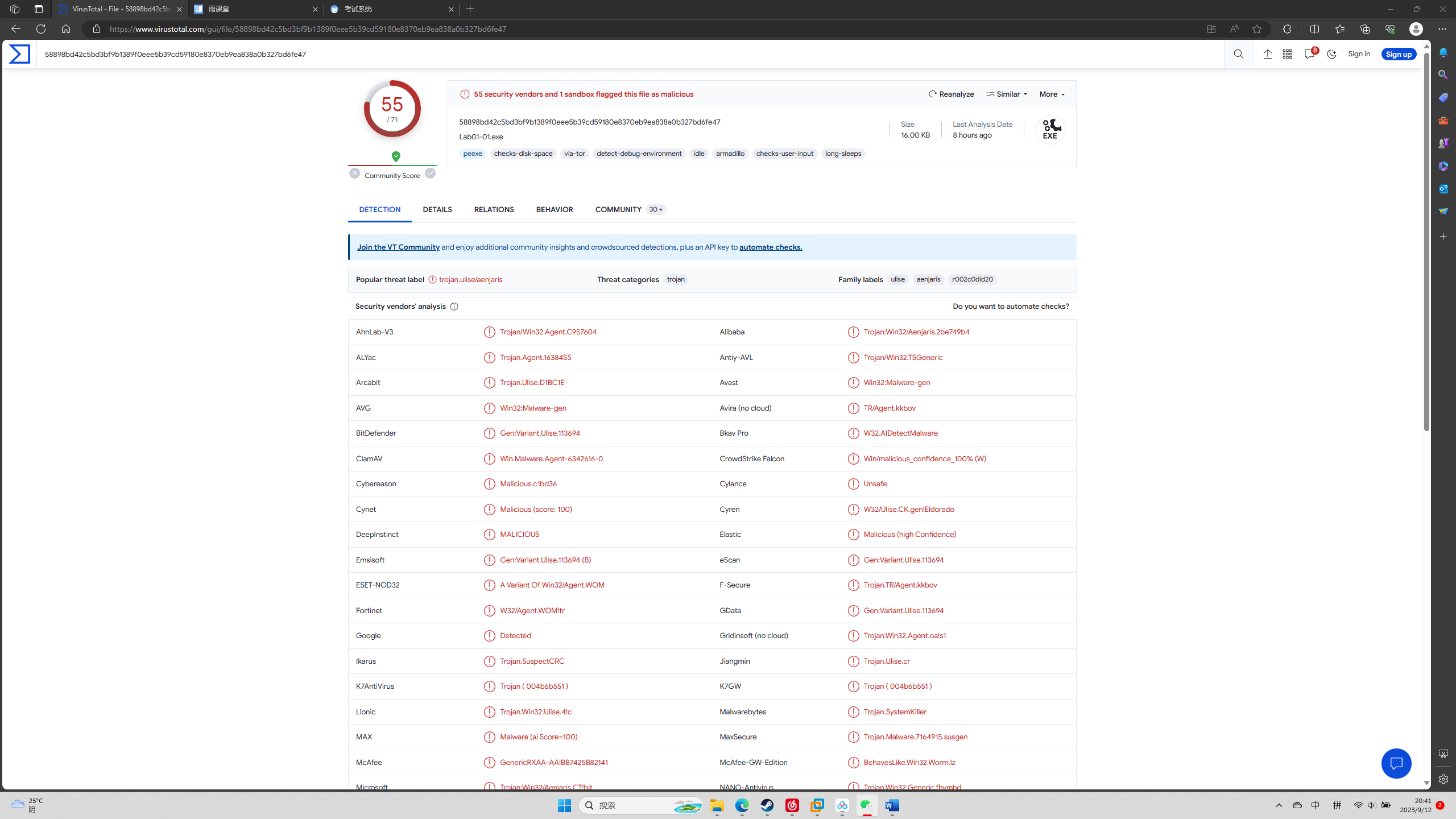
**完成教材Lab1的实验内容，编写Lab1样本的Yara引擎规则，并测试规则的执行效率。**

1. **实验原理**

**使用PEView、Yara等恶意代码检测工具分析样例代码，回答Lab1中问题，编写Yara规则，达到检测出样例代码的效果。**

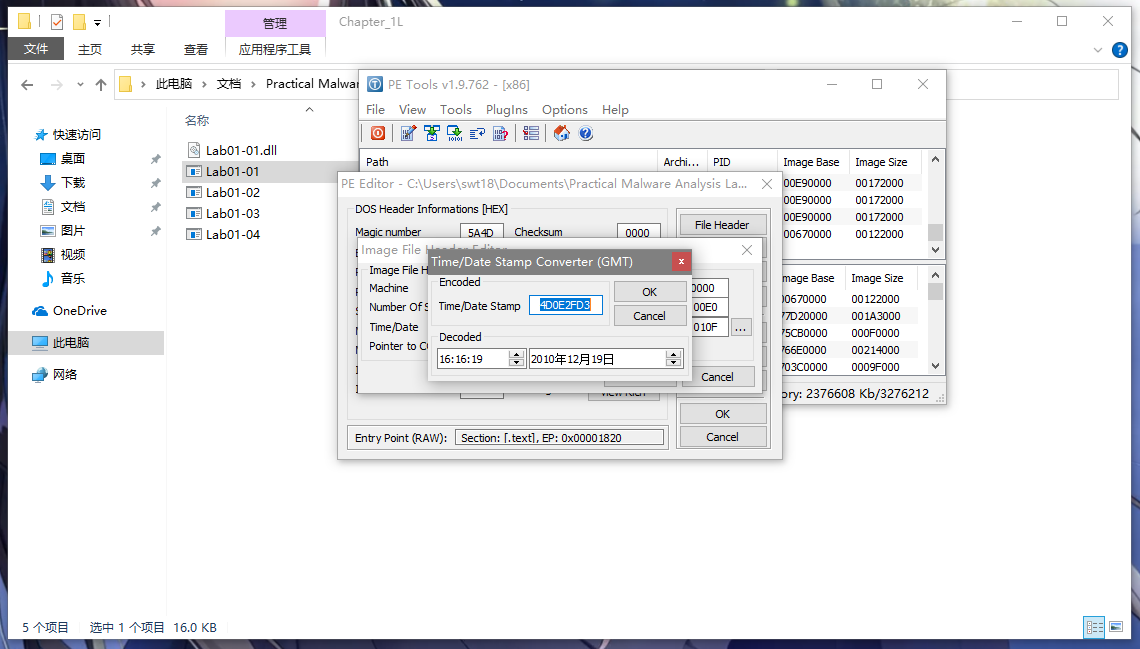
1. **实验过程**
2. **Lab01-01**
3. **将Lab01-01.exe文件上传至VIRUSTOTAL进行分析并查看其反馈报告。文件匹配到了已有的反病毒软件特征吗？**

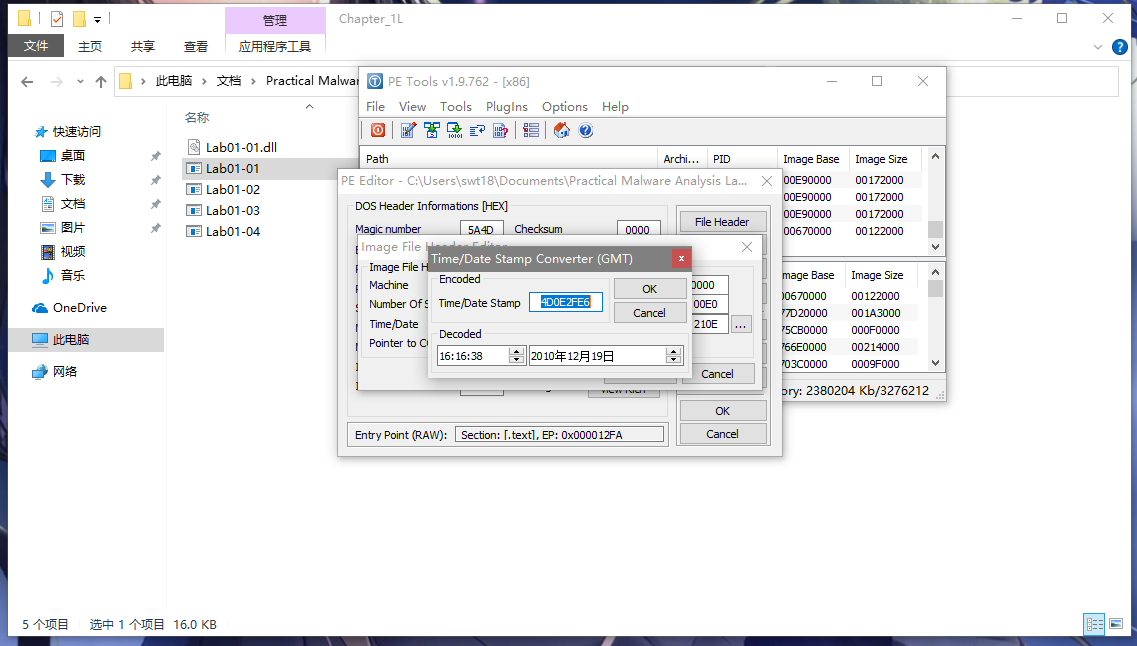
* **检测结果显示该文件被55个杀毒软件所标记。**



1. **这些文件都是什么时候编译的？**

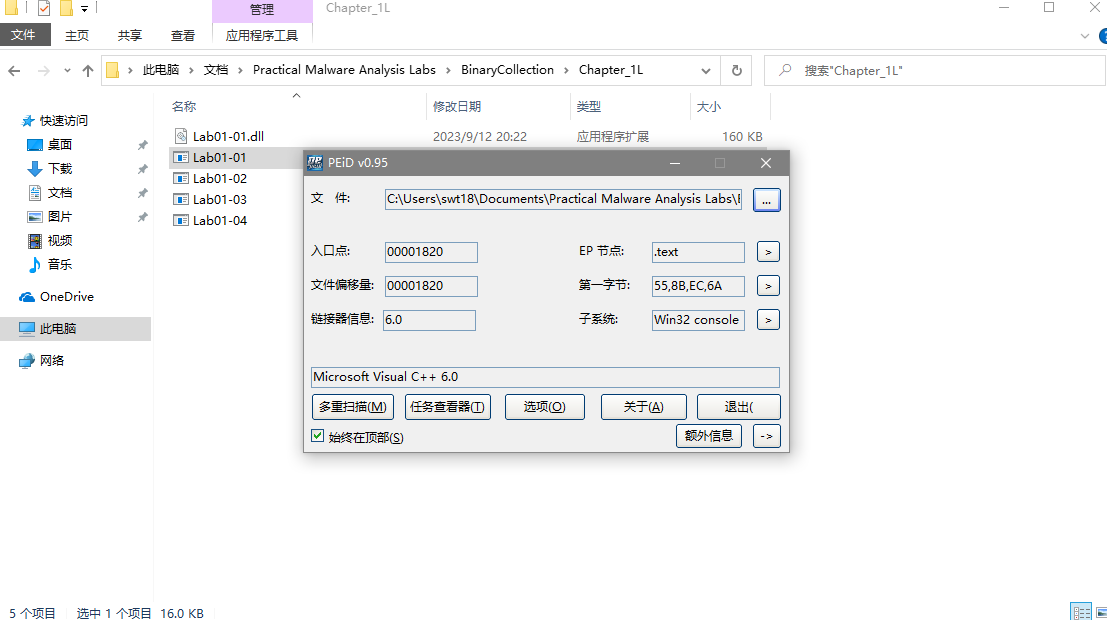
* **这两个文件都是2010年10月19日编译的，并且编译时间前后相隔时间较短，仅为19秒。**
* **由于编译时间如此接近，判断很可能是同一作者在同一时间创建了这些文件，由于DLL动态链接库文件无法自己运行，故而这个.exe 极有可能是用来是使用或安装.dll 文件的。**

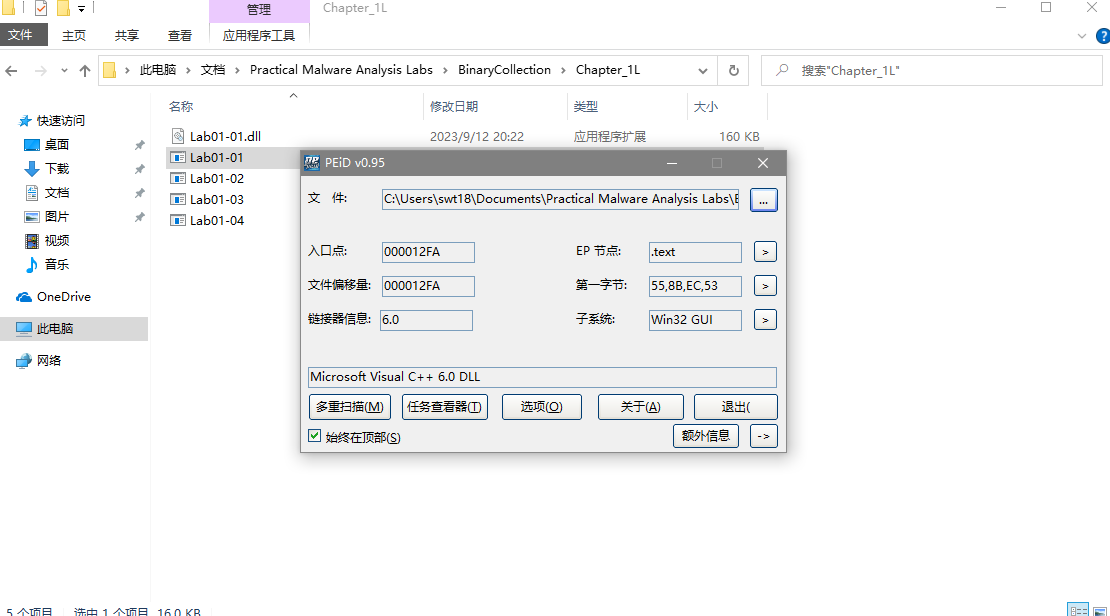




1. **这两个文件中是否存在迹象说明它们是否被加壳或混淆了？如果是，这些迹象在哪里？**

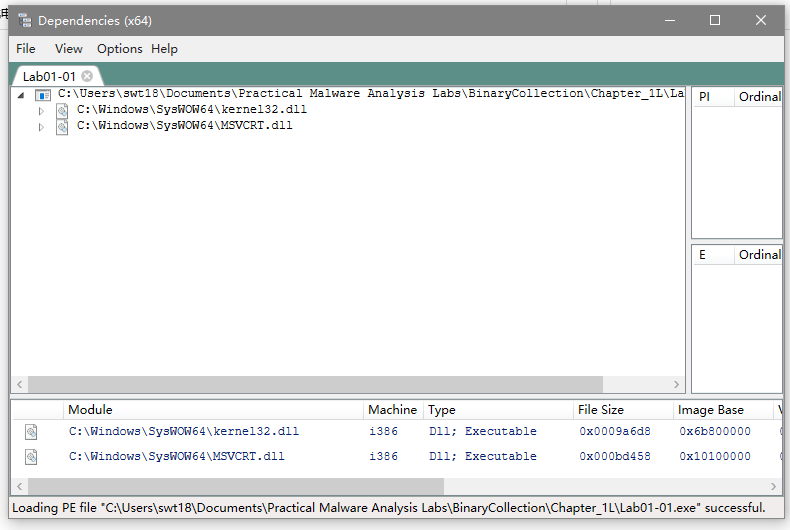
* **通过PEid工具打开.exe文件，可以看到明确的编译工具Microsoft Visual C++ 6.0，并且文件中的分节显示，虚拟大小并没有出现比原始数据大很多的情况，因此判断未被加壳或混淆。**



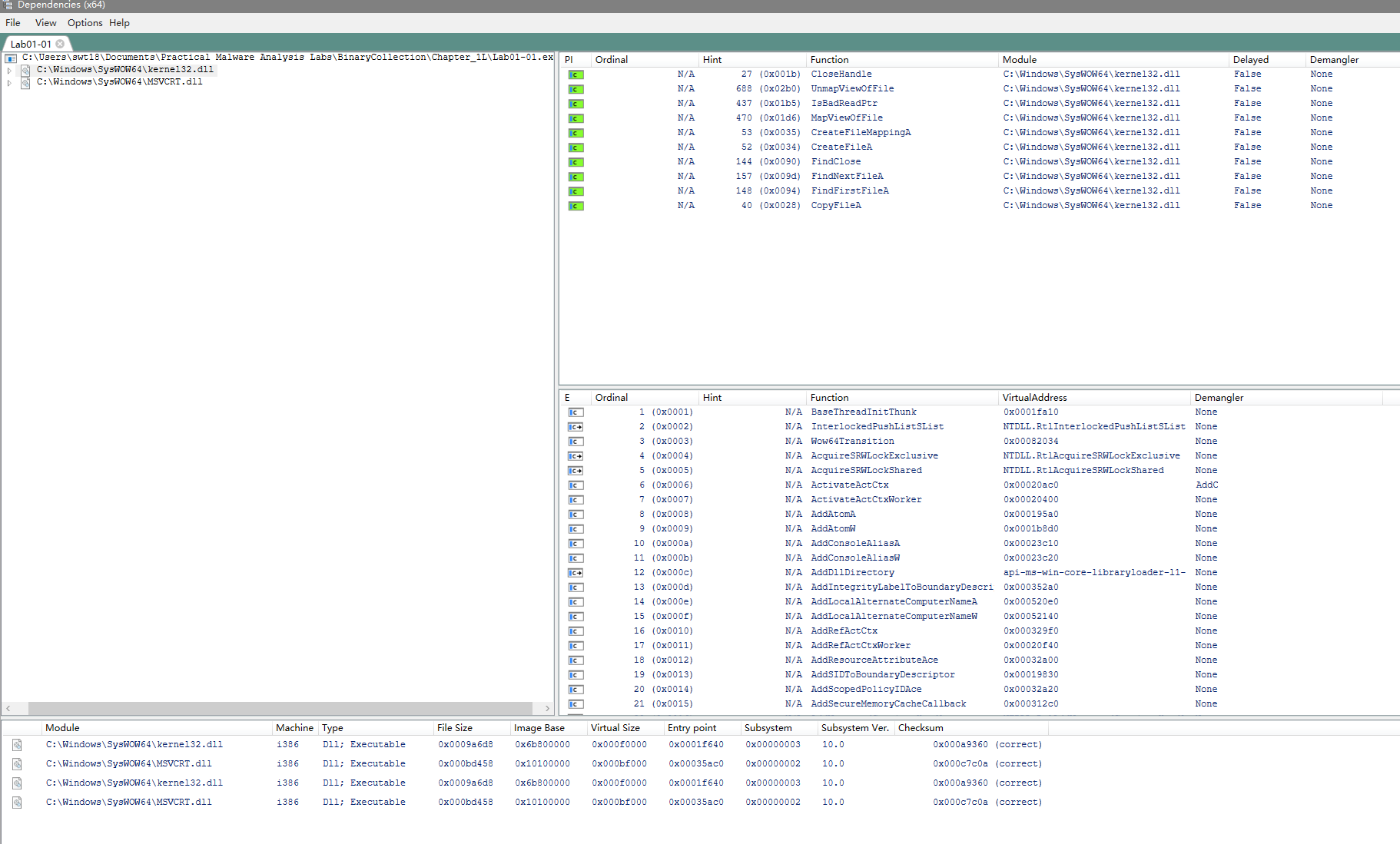


1. **是否有导入函数显示这个恶意代码的用途？如果是，有哪些导入函数。**

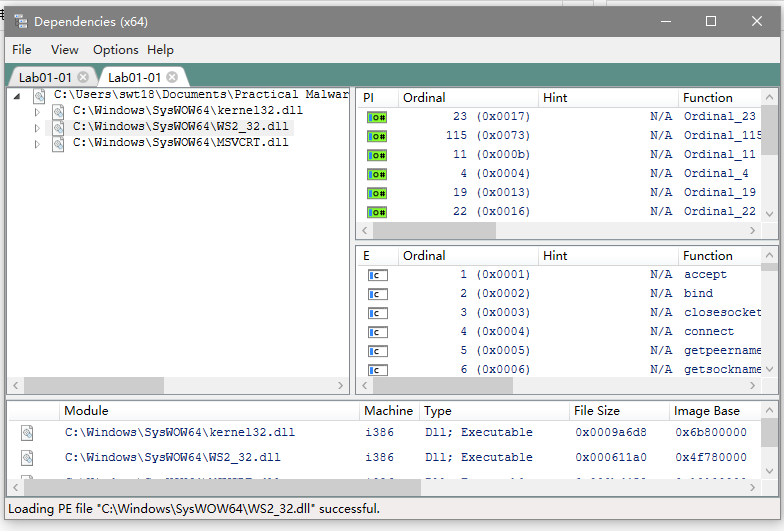
* **用Dependencies工具打开.exe文件，看到它有KERNEL32.DLL和MSVCRT.DLL两个动态链接库。**



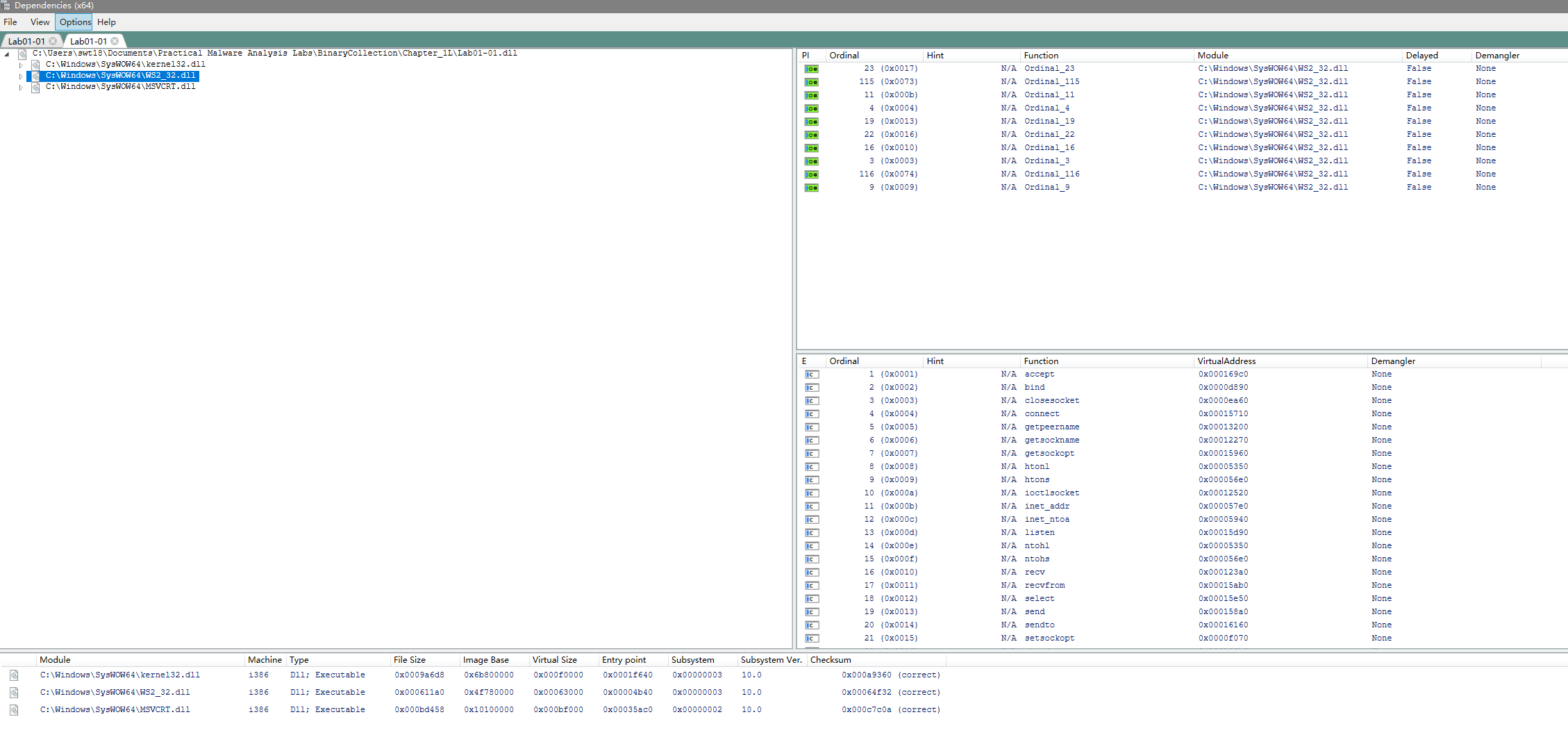
* **从MSVCRT.DLL导入的函数通常是被每一个可执行文件都包含，因为它们是作为包装代码被编译器加入可执行文件的。查看从KERNEL32.DLL导入的函数时，可以看到一些打开与操作文件的函数以及FindFirstFile、FindNextFile 和 CopyFile等，这些函数意味着，恶意代码可以对文件进行搜索、打开、复制和修改等操作，同时.exe字符串说明，恶意代码正在寻找搜索目标系统上的可执行文件。**



* **同理打开.dll文件，发现KERNEL32.DLL和WS2\_32.DLL两个动态链接库。**

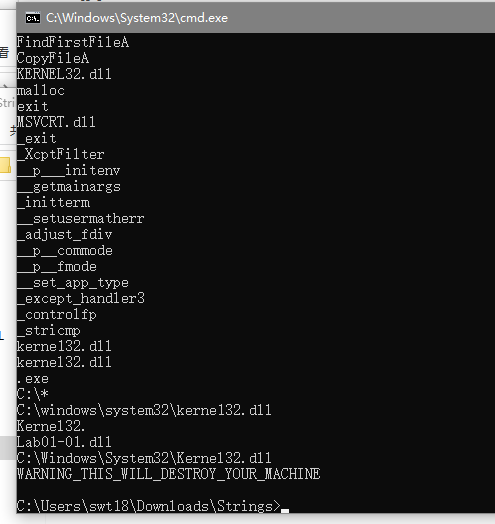


* **通过查询可知，WS2\_32.dll 提供了联网功能，打开WS2\_32.DLL发现这些导入函数都是按照序号进行导入的。打开KERNEL32.DLL可以看到CreateProcess 和 Sleep两个函数，而这两个函数普遍使用在后门程序中。**



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

* **用Strings工具检查.exe文件，发现kernel32.dll和kerne132.dll两个不同的文件，后者用数字1替换了字母l，可以猜测是为了模仿系统文件kernel32.dll的恶意文件，因此这个文件可以用来在主机作为恶意代码感染的迹象进行搜索。**



* **打开.dll文件，可以发现除了已知的 CreateProcessA 和 Sleep，我们还注意到 exec 和 sleep。**
* **经过查询可知，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 。而sleep 可能用于让后门程序进入休眠模式。**



1. **是否有基于网络的迹象可以用来发现感染机器上的恶意代码？**

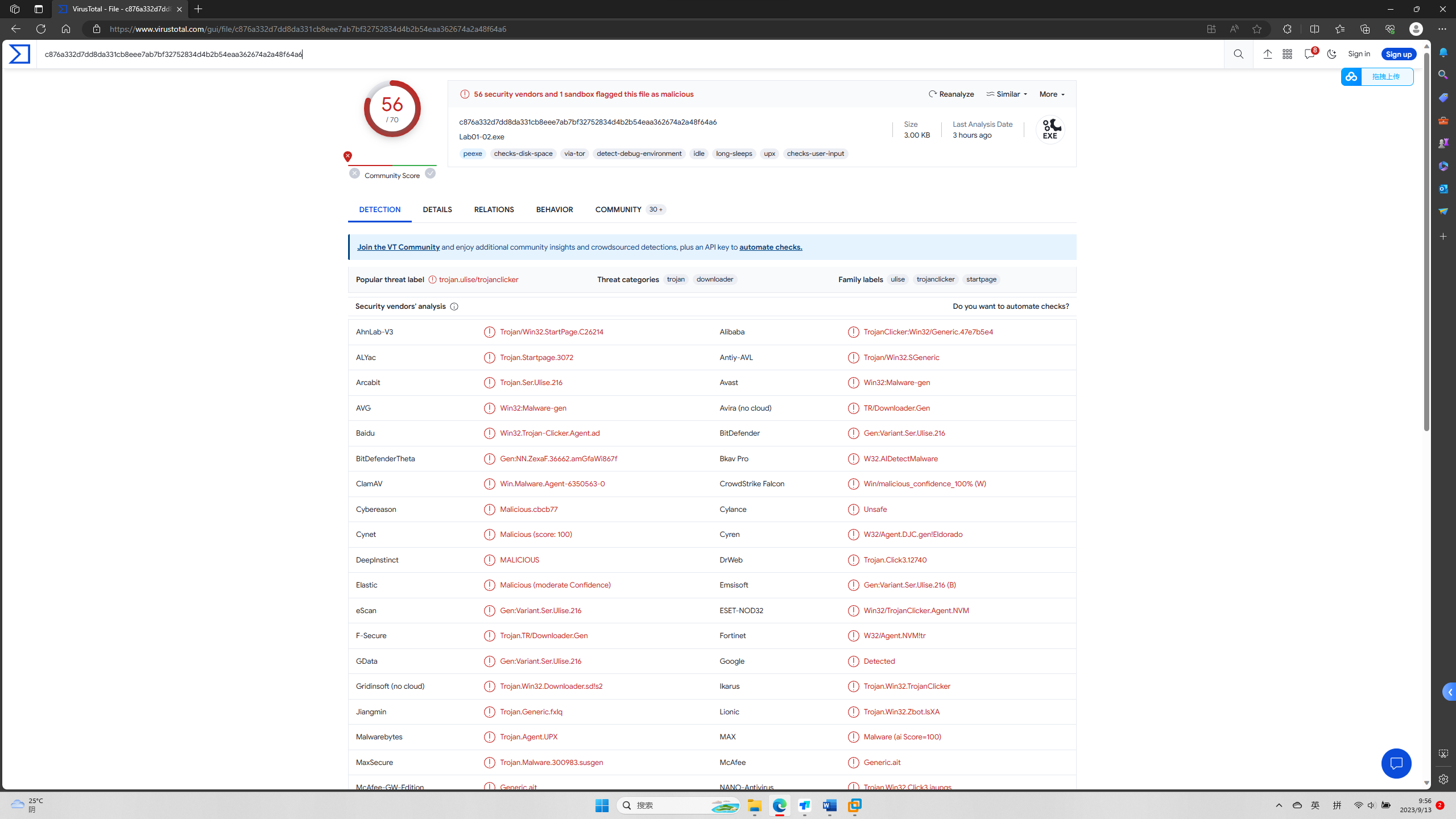
* **上述对.dll的分析发现其中包含一个私有子网IP地址 127.26.152.13的字符串。结合其调用的 WS2\_32.dll，猜测该程序可能联网通信。此次的IP地址为 127 开头的 IP 为本地地址，在此处表明是用于教学目的，而在现实中却可能指向一个具体的外网 IP。这是一个很好的基于网络的恶意代码感染迹象，可以用来识别这个恶意代码。**

1. **你猜这些文件的目的是什么？**

* **基于以上分析，该.dll 文件可能是一个后门文件，而.exe文件则是用来安装与运行该dll文件。**

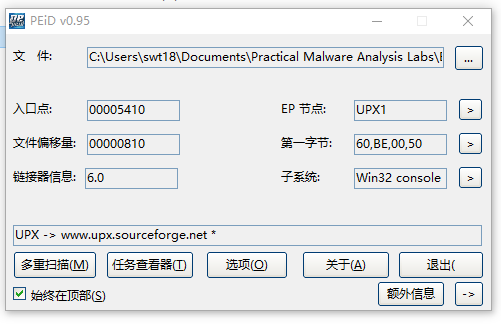
1. **Lab01-02**
2. **将Lab01-02.exe文件上传至VIRUSTOTAL进行分析并查看其反馈报告。文件匹配到了已有的反病毒软件特征吗？**

* **结果显示被56个杀毒软件标记。**

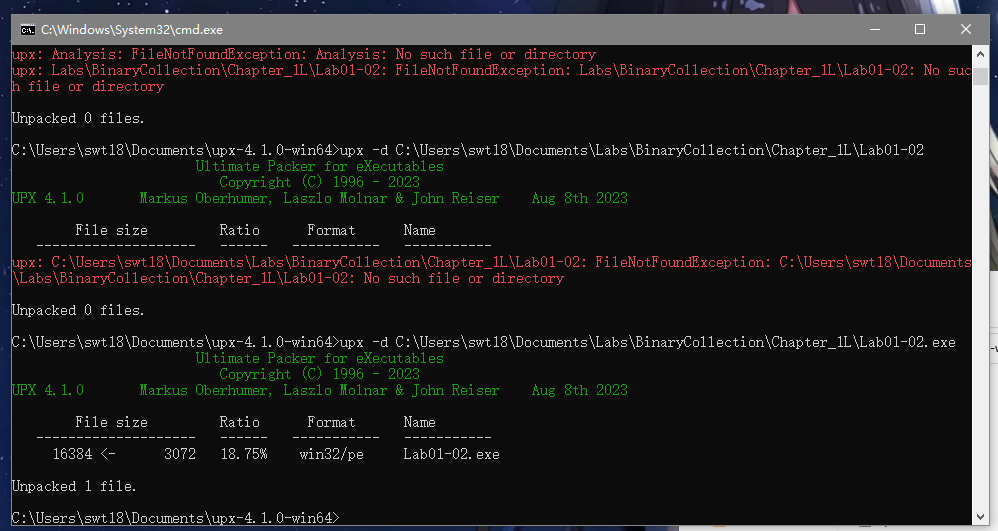


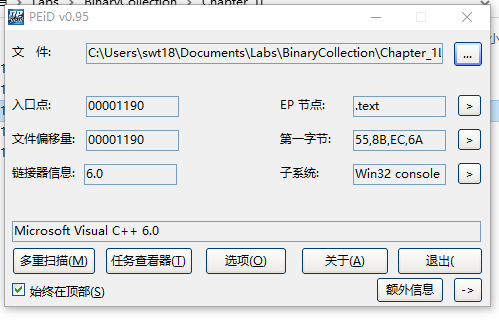
1. **是否有这个文件被加壳或混淆的任何迹象？如果是这样，这些迹象是什么？如果文件被加壳，请进行脱壳。**

* **用PEid打开，发现该文件被UPX加壳。**



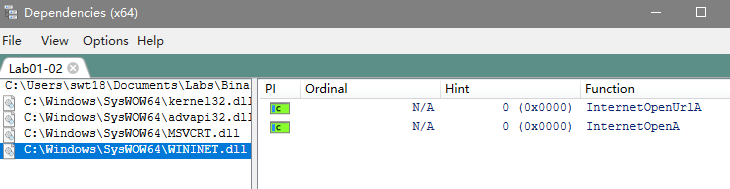
* **用UPX工具进行脱壳后，可以看到正常信息，如编译器为Microsoft Visual C++ 6.0等。**

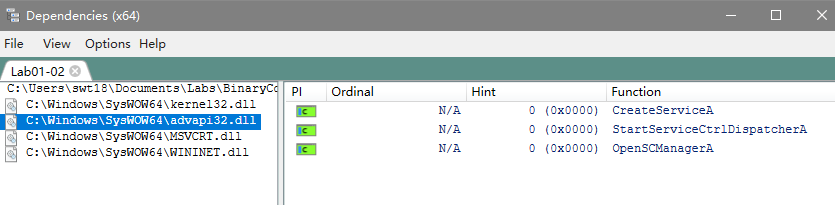




1. **有没有任何导入函数能够暗示出这个程序的功能？如果是，是哪些导入函数，他们会告诉你什么？**

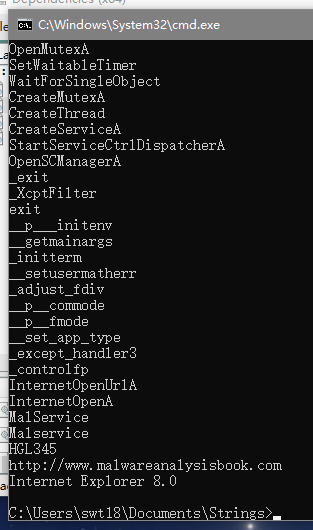
* **用Denpendencies工具分析脱壳后的.exe文件，看到从WININET.DLL导入的函数InternetOpenA和InternetOpenUrl表示这个恶意代码会进行联网操作。从ADVAPI32.DLL导入的函数CreateService表示这个代码会创建一个服务。**





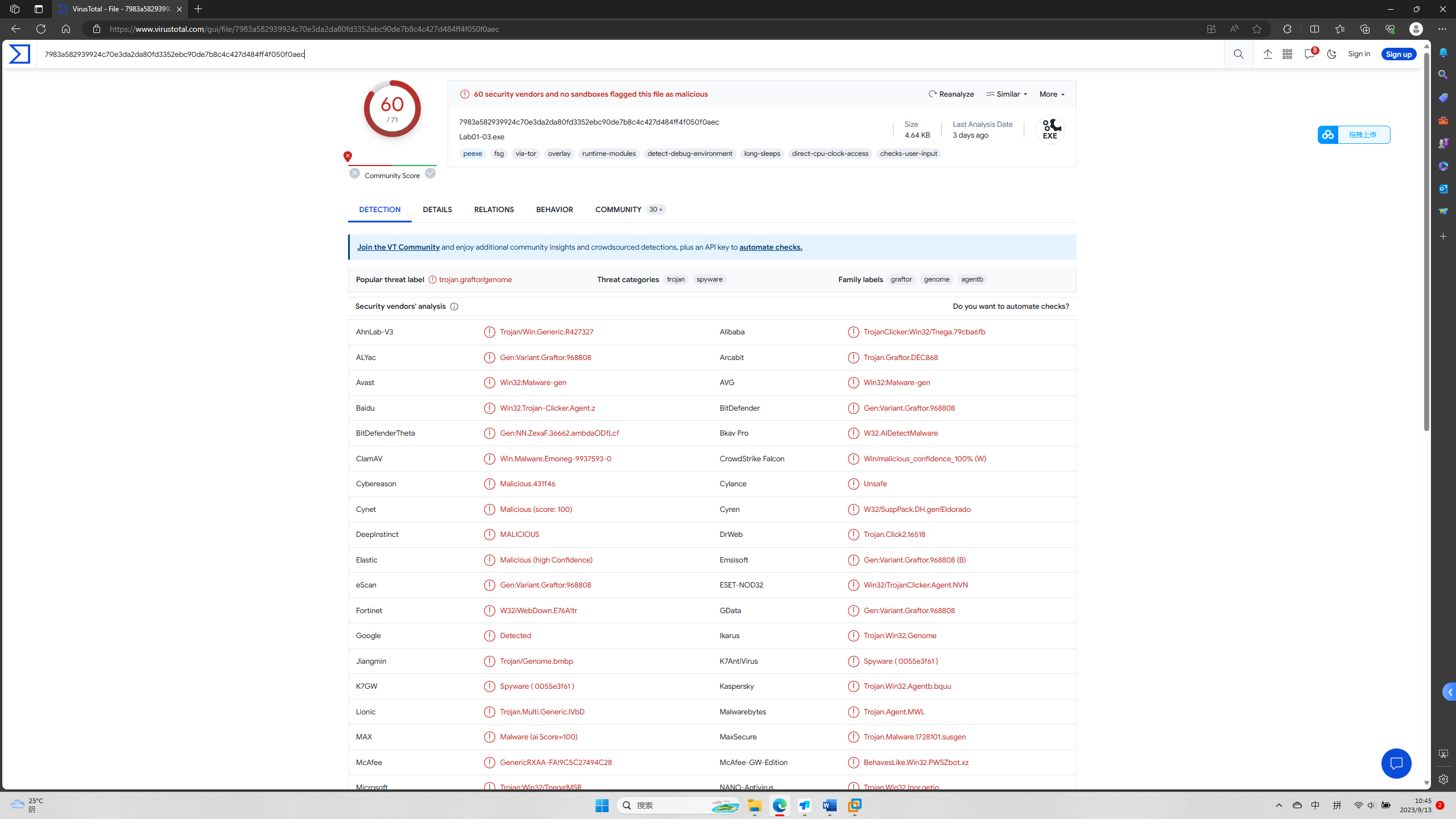
1. **哪些基于主机或基于网络的迹象，可以被用来确定这个恶意代码所感染的机器？**

* **通过Strings工具检查字符串列表，发现http://www.malwareanalysisbook.com ，推测可能是InternetOpenURL函数中所打开的URL。除此之外，还发现Malservice字符串，所以应该通过一个名为Malservice的服务，并通过 http://www.malwareanalysisbook.com 的网络流量，来检查恶意代码感染的主机。**



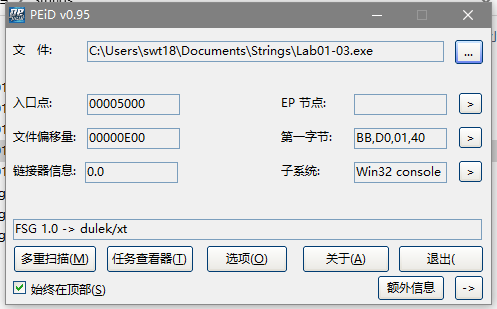
1. **Lab01-03**
2. **将Lab01-03.exe文件上传至VIRUSTOTAL进行分析并查看其反馈报告。文件匹配到了已有的反病毒软件特征吗？**

* **发现被60个杀毒软件标记。**

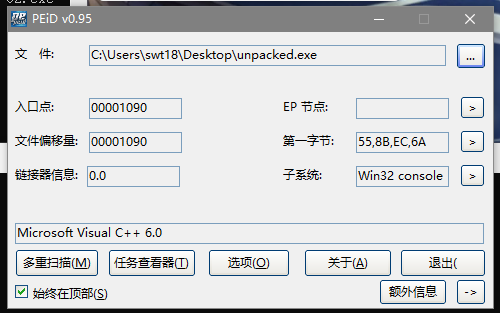


1. **是否有这个文件被加壳或混淆的任何迹象？如果是这样，这些迹象是什么？如果文件被加壳，请进行脱壳，如果可能的话。**

* **使用PEid工具分析.exe文件，看到加壳类型为FSG1.0.**

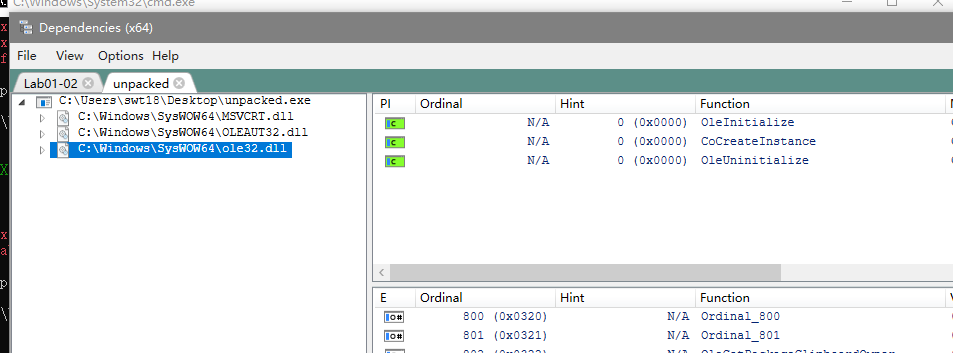


* **使用工具UnpackSDK脱壳后，正常显示编译工具。**



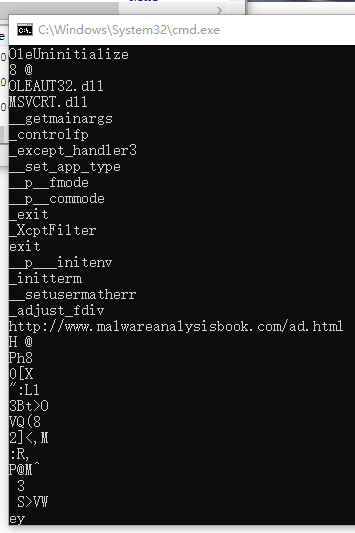
1. **有没有任何导入函数能够暗示出这个程序的功能？如果是，是哪些导入函数，他们会告诉你什么？**

* **用Dependencies工具打开脱壳后的.exe文件，看到从ole32.dll导入了函数CoCreateInstance，该函数用于在本机上创建一个对象。**

****

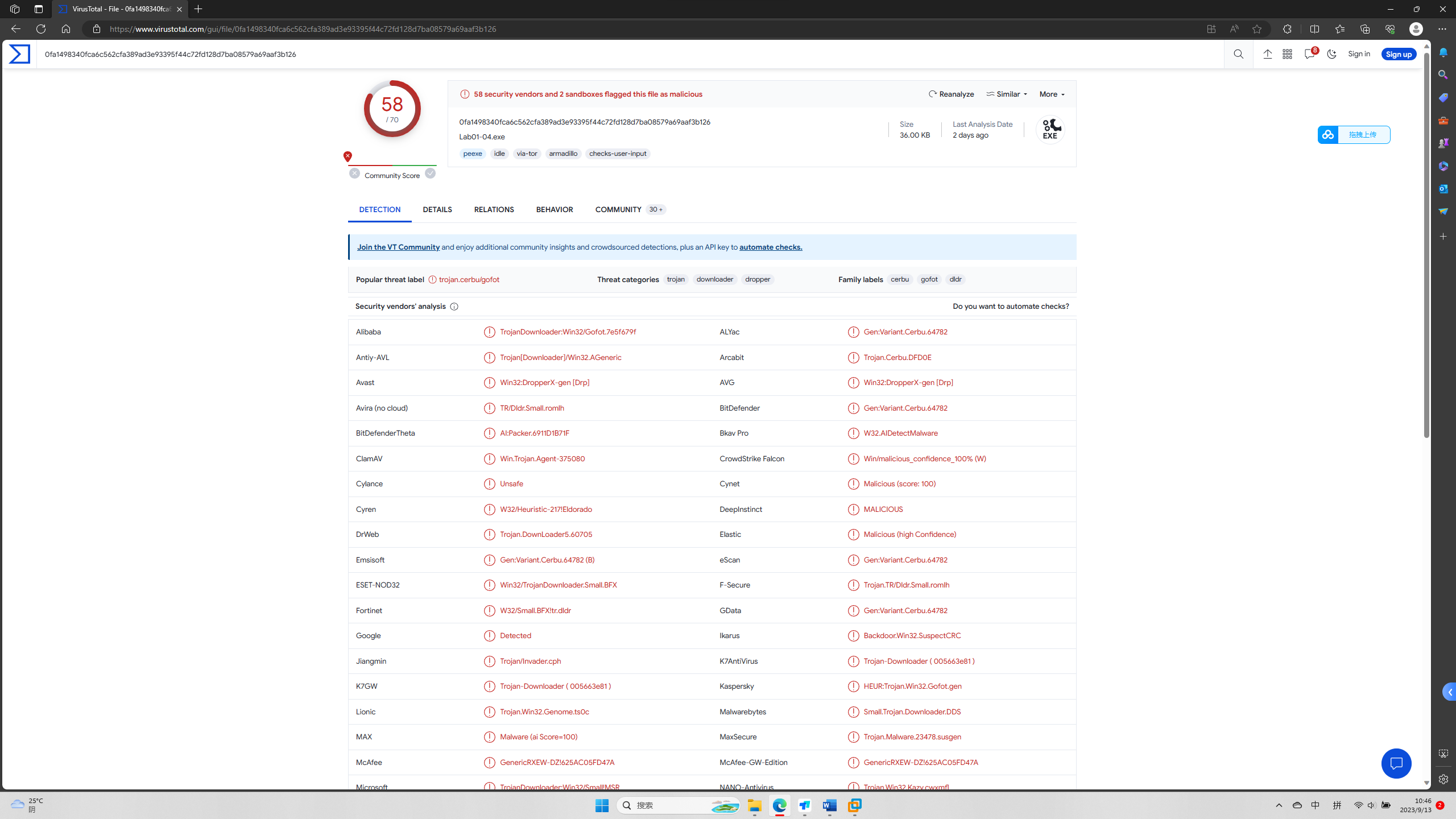
1. **哪些基于主机或基于网络的迹象，可以被用来确定这个恶意代码所感染的机器？**

* **用Strings工具检查脱壳后的.exe文件，看到字符串**[**http://www.malwareanalysisbook.com/ad.html**](http://www.malwareanalysisbook.com/ad.html)**，推测该文件是用于打开该网站。**



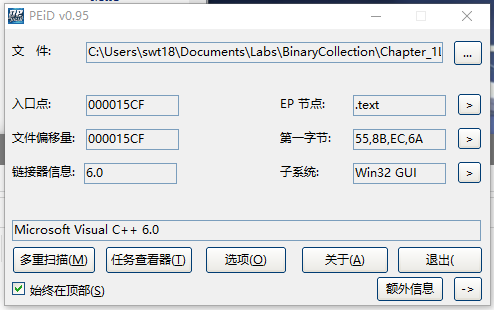
1. **Lab01-04**
2. **将Lab01-03.exe文件上传至VIRUSTOTAL进行分析并查看其反馈报告。文件匹配到了已有的反病毒软件特征吗？**

* **发现被58个杀毒软件标记**



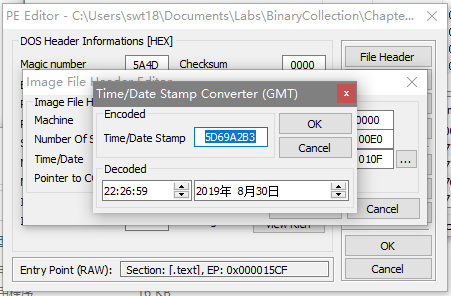
1. **是否有这个文件被加壳或混淆的任何迹象？如果是这样，这些迹象是什么？如果文件被加壳，请进行脱壳，如果可能的话。**

* **用PEid工具打开，看到没有被加壳。**



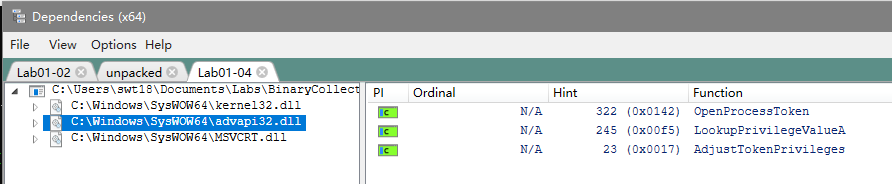
1. **这个文件是什么时候编译的？**

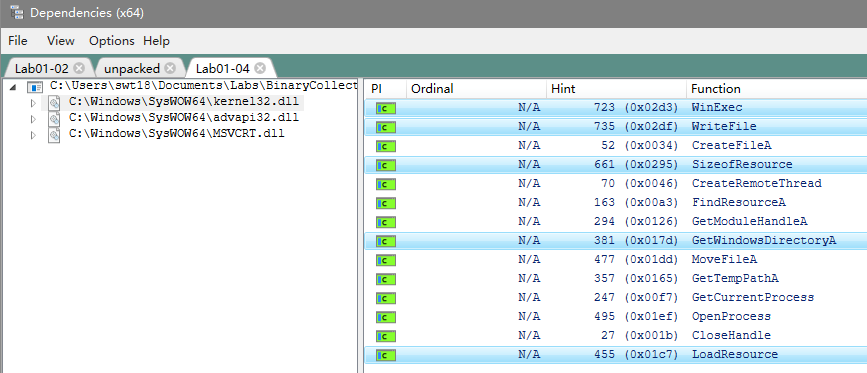
* **用PE tools工具打开，看到编译时间为2019年8月30日，该时间和之前的文件相比未免过新，很可能是伪造的。**



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，他们会告诉你什么？**

* **从ADVAPI32.DLL导入的函数，表示程序做了一些与权限有关的事情，试图访问使用了特殊权限进行保护的文件。**
* **从KERNEL32.DLL的导入函数告诉我们这个程序从资源节中装载数据（LoadResource、FindResource和SizeofResource），并写一个文件到磁盘上（CreateFile和WritFile），接着执行一个磁盘上的文件（WinExec）。因为它调用了GetWindowDirectory函数，可以猜测这个程序将文件写入到了系统目录。**





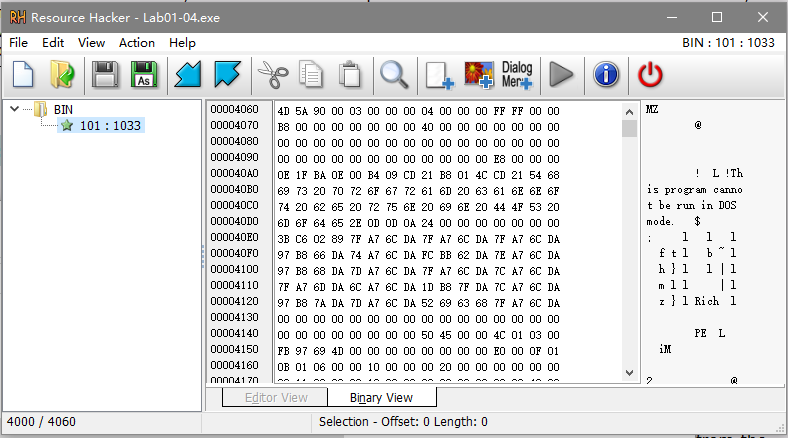
1. **哪些基于主机或基于网络的迹象，可以被用来确定这个恶意代码所感染的机器？**

* **使用Strings工具检查字符串，出现 \system32\wupdmgrd.exe （Windows 升级管理器），结合GetWindowDirectory函数调用，表明恶意代码在 C：Windows\system32\wupdmgrd.exe 位置创建或修改一个文件。**
* **而 www.malwareanalysisbook.com/updater.exe 很可能是要下载的恶意代码的存储位置，或者是伪装成这个文件。**

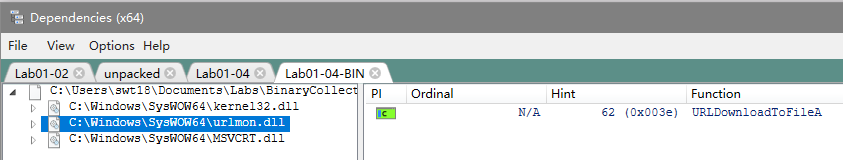


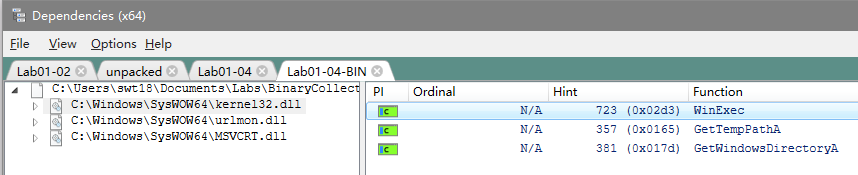
1. **这个文件在资源段中包含一个资源，使用Resource Hacker工具来检查资源，然后抽取资源，资源中你能发现什么吗？**

* **使用Resource Hacker工具来检查资源，发现资源段中还有一个可执行文件，并且有PE文件头的特征。**



* **将其提取出来用Dependencies工具分析，看到其调用了URLDownloadToFileA函数和WinExec函数，可能是试图下载并执行某个恶意文件。**





1. **Yara规则编写**

**作为PE文件，上述文件均满足filesize<10MB and uint16(0)==0x5A4D and uint16(uint16(0x3C))==0x00004550的条件，因此将其作为全局规则。**

1. **Lab01-01**

* **.exe文件调用了Lab01-01.dll和“kerne132.dll”等特有的动态链接库，可以作为检测特征。**

1. rule lab1\_1\_exe
2. {
3. strings:
4. $string1 = "Lab01-01.dll"
5. $string2 = "kerne132.dll"
6. condition:
7. filesize < 10MB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) == 0x00004550 and $string1 and $string2
8. }

* **.dll文件调用了sleep、CreateProcessA等函数，可以作为检测特征。**

1. rule lab1\_1\_dll
2. {
3. strings:
4. $string1 = "sleep"
5. $string2 = "CreateProcessA"
6. condition:
7. filesize < 10MB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) ==
8. 0x00004550 and $string1 and $string2
9. }
10. **Lab01-02**

* **由于原程序经过加壳，用Strings工具分析原.exe文件可以检测出MalService等字符串，可以作为检测特征。**

1. rule lab1\_2\_exe
2. {
3. strings:
4. $string1 = "HGL345"
5. $string2 = "MalService"
6. condition:
7. filesize < 10MB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) ==
8. 0x00004550 and $string1 and $string2
9. }
10. **Lab01-03**

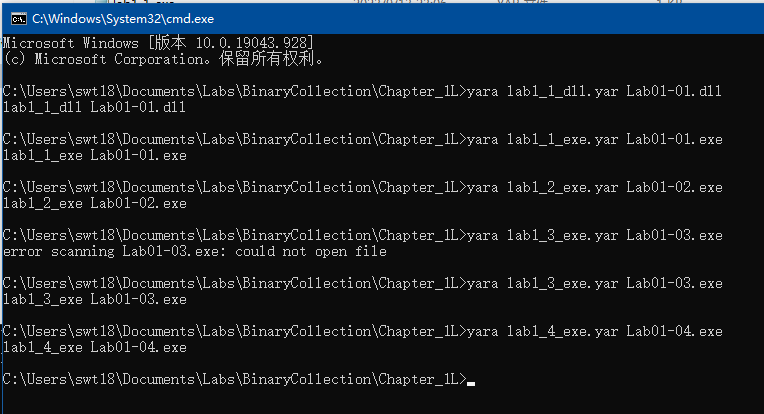
* **由于原程序经过加壳，用Strings工具分析原.exe文件可以检测出ole32.vd等字符串，可以作为检测特征。**

1. rule lab1\_3\_exe
2. {
3. strings:
4. $string1 = "ole32.vd"
5. $string2 = "mal"
6. $string3 = "\_getmas"
7. condition:
8. filesize < 10MB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) ==
9. 0x00004550 and $string1 and $string2 and $string3
10. }
11. **Lab01-04**

* **其调用了函数LoadResource,并且字符串分析中出现了某下载链接,可作为检测特征。**

1. rule lab1\_4\_exe
2. {
3. strings:
4. $string1 = "LoadResource"
5. $string2 = "http://www.practicalmalwareanalysis.com/updater.exe"
6. condition:
7. filesize < 10MB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) ==
8. 0x00004550 and $string1 and $string2
9. }
10. **规则测试**

* **逐个利用所编写规则测试文件，可以看到每个文件可以被所编写规则检测出。（第一次检测Lab01-03.exe失败是因为该文件进行脱壳后名称改变了）**



1. **实验结论及心得体会**
2. **对恶意代码调用函数实现攻击的方式有了进一步了解。**
3. **对文件加壳脱壳有了初步了解，学会了如何判断加壳类型，以及如何脱壳。**
4. **初步了解了Yara规则的编写，可以针对文件编写对应的Yara规则。**