

**恶意代码分析课程实验报告**

**实验一**

****

学 院 计网

专 业 信息安全

学 号 2113662

姓 名 张丛

班 级 信息安全一班

1. **实验目的**

**完成书后4个实验和yara实验**

1. **实验原理**

**静态分析工具的使用，yara学习**

1. **实验过程**

**Lab 1-1**

## **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?**

## **lab01-01.exe:**

## 

## **lab01-01.dll:**

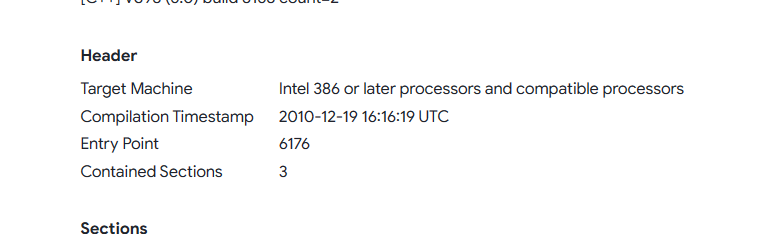
## 

可见，两个文件都已经匹配到了已有的反病毒软件特征。

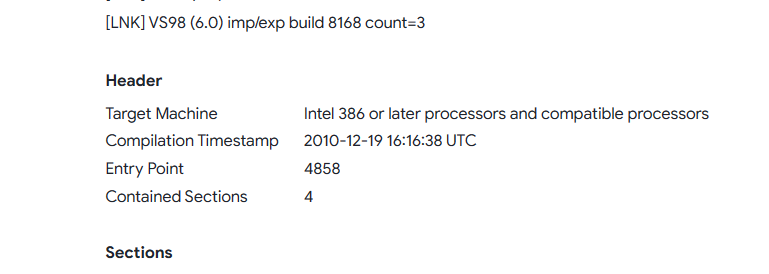
## 这些文件是什么时候编译的？

在VirusTotal给出的报告的Details部分，可以看到PE信息

lab01-01.exe:



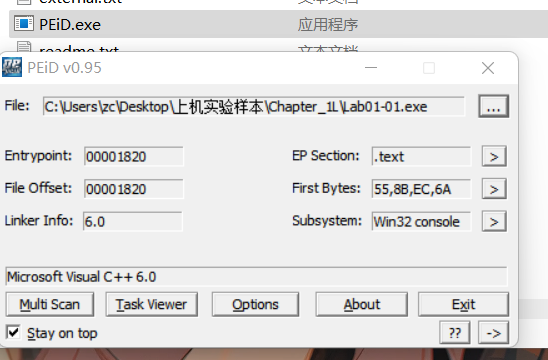
lab01-01.dll:

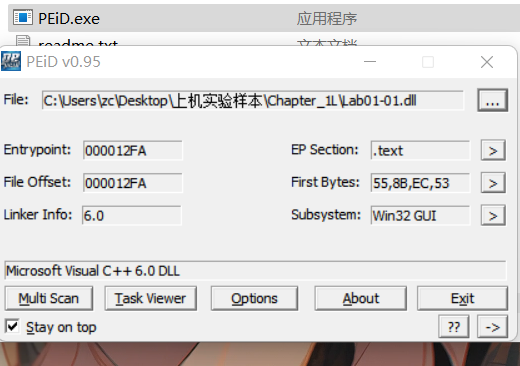


可知，两者的编译时间均为2010年12月19日16时16分，两者编译时间仅相差19秒

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

使用PEID工具对两文件进行分析：

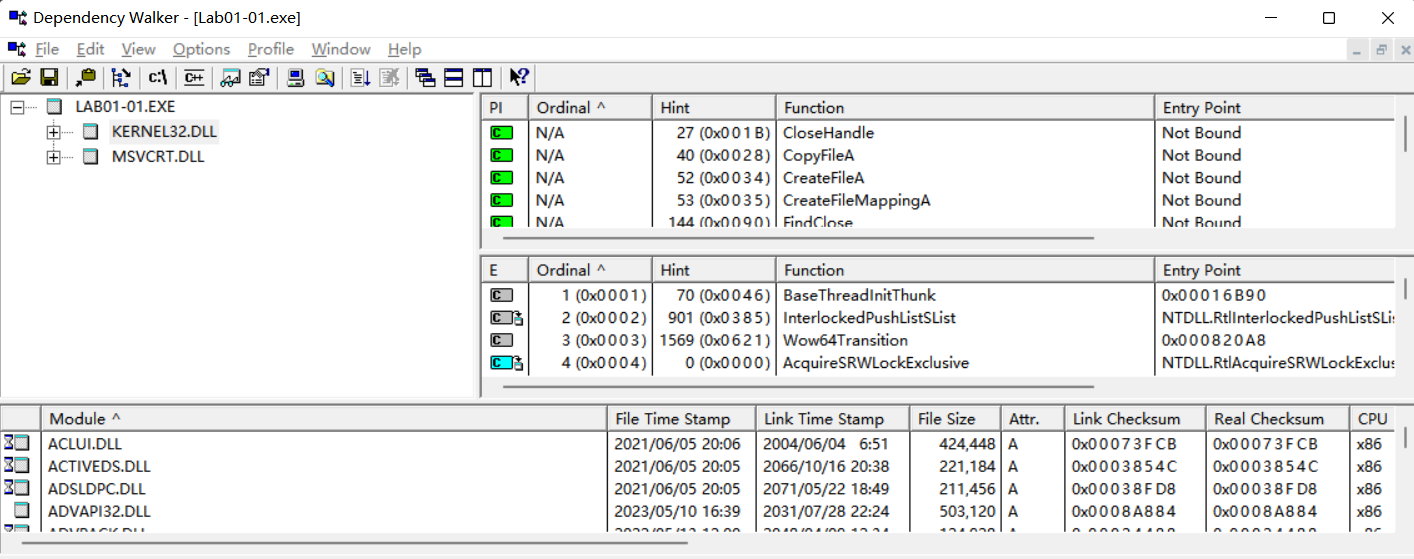




可知，两个文件没有被混淆以及加壳。

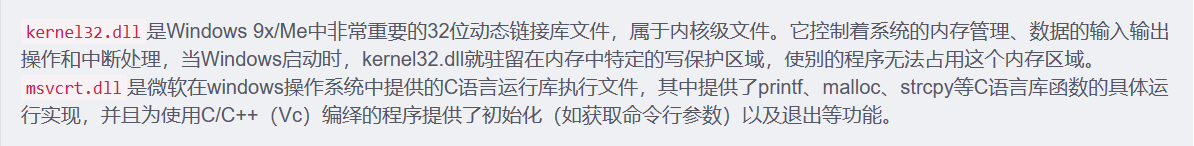
## 是否有导入函数显示出了这个恶意代码是做什么的？如果是，是哪些导入函数？

使用Dependency Walker查看lab01-01.exe的导入表函数：

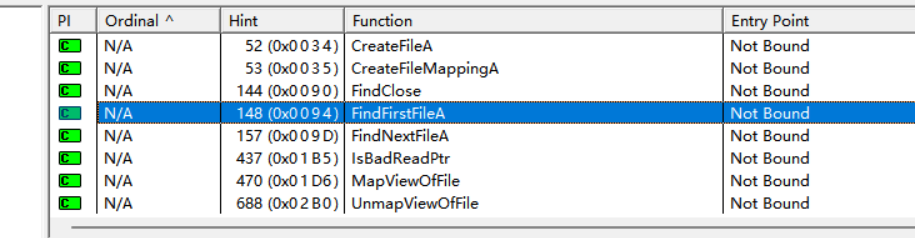


可见，导入了两个dll: KERNEL32.DLL和MSVCRT.DLL

查看这两个dll的功能（因为暂且记不住）：

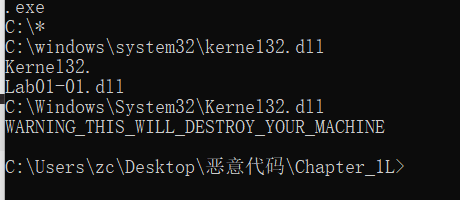


前者对应的API函数如下：

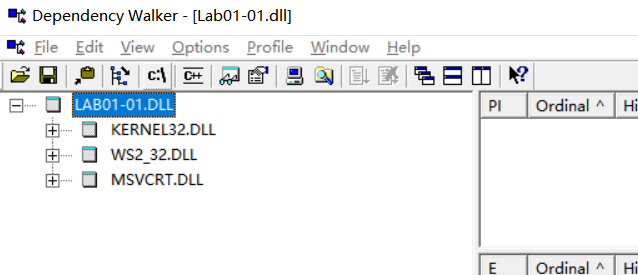


FindFirstFileA和FindNextFileA说明这个文件可能会进行搜索文件的行为。

使用strings程序搜索可打印字符串：

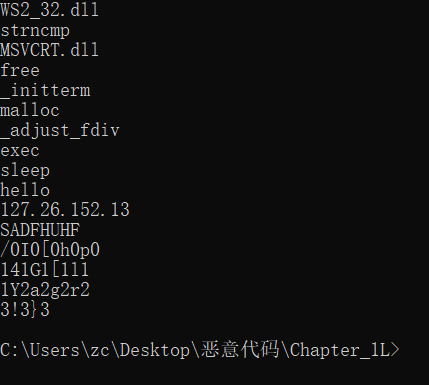


使用Dependency Walker查看lab01-01.dll的导入表函数：



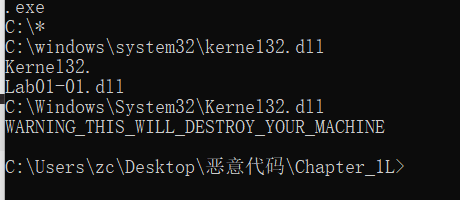
较于lab01-01.exe多出的是WS2\_32.DLL，它提供了与网络通信相关的函数和操作。

使用strings程序：



可见有一个ip地址：127.26.152.13

## 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？



通过strings程序找到kerne132.dll这个程序，用数字1代替了字母l，是为了看起来像是系统文件kernel32.dll。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。

## 6.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？

## 

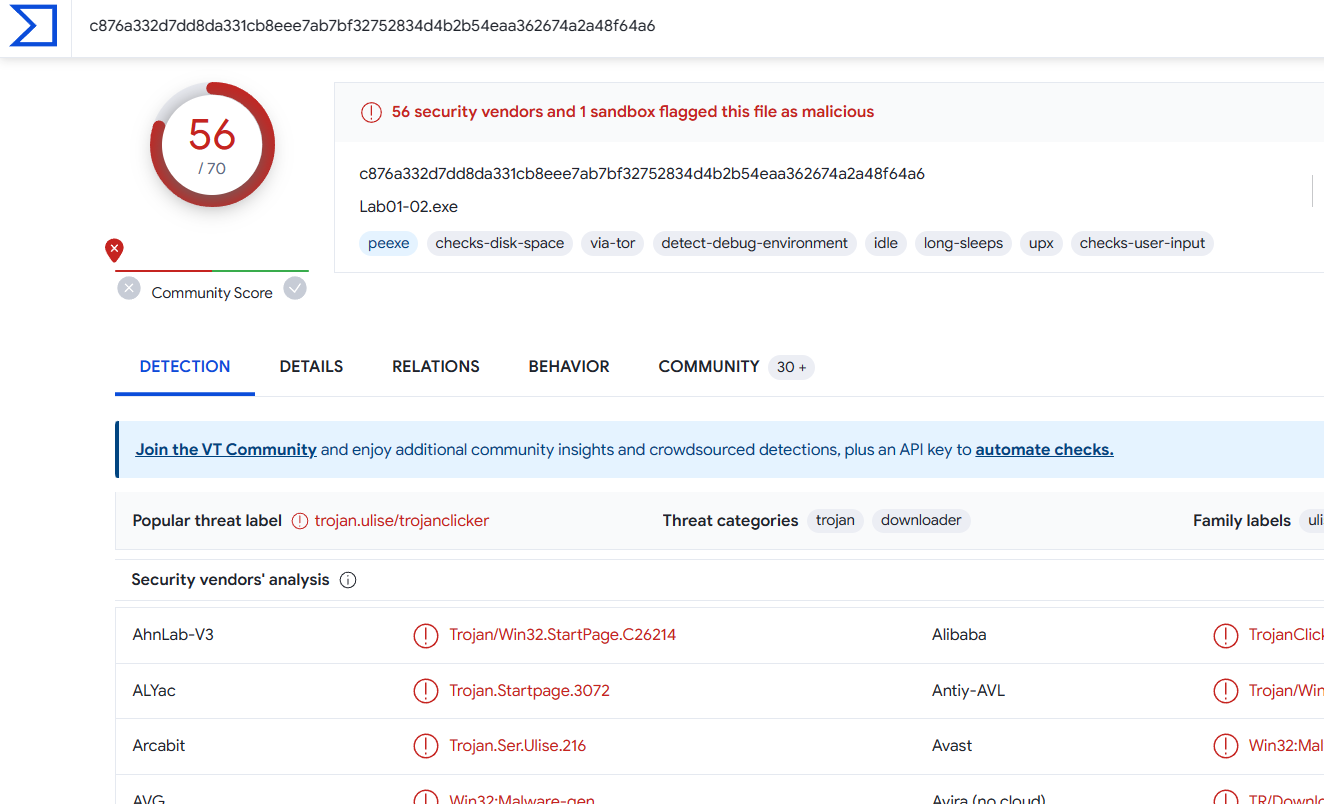
发现dll文件中有ip地址，sleep，exec等函数，可能会进入此ip地址进行下载文件，可能是后门程序。

## 7.你猜这些文件的目的是什么？

dll文件可能是一个后门，exe文件用来安装与运行dll文件

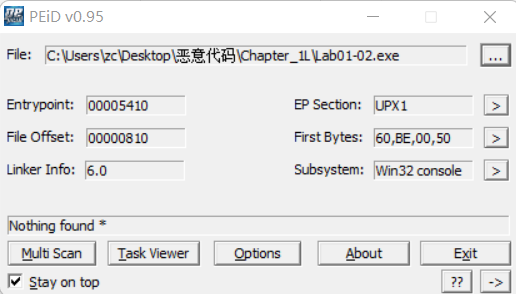
**Lab 1-2**

## 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?



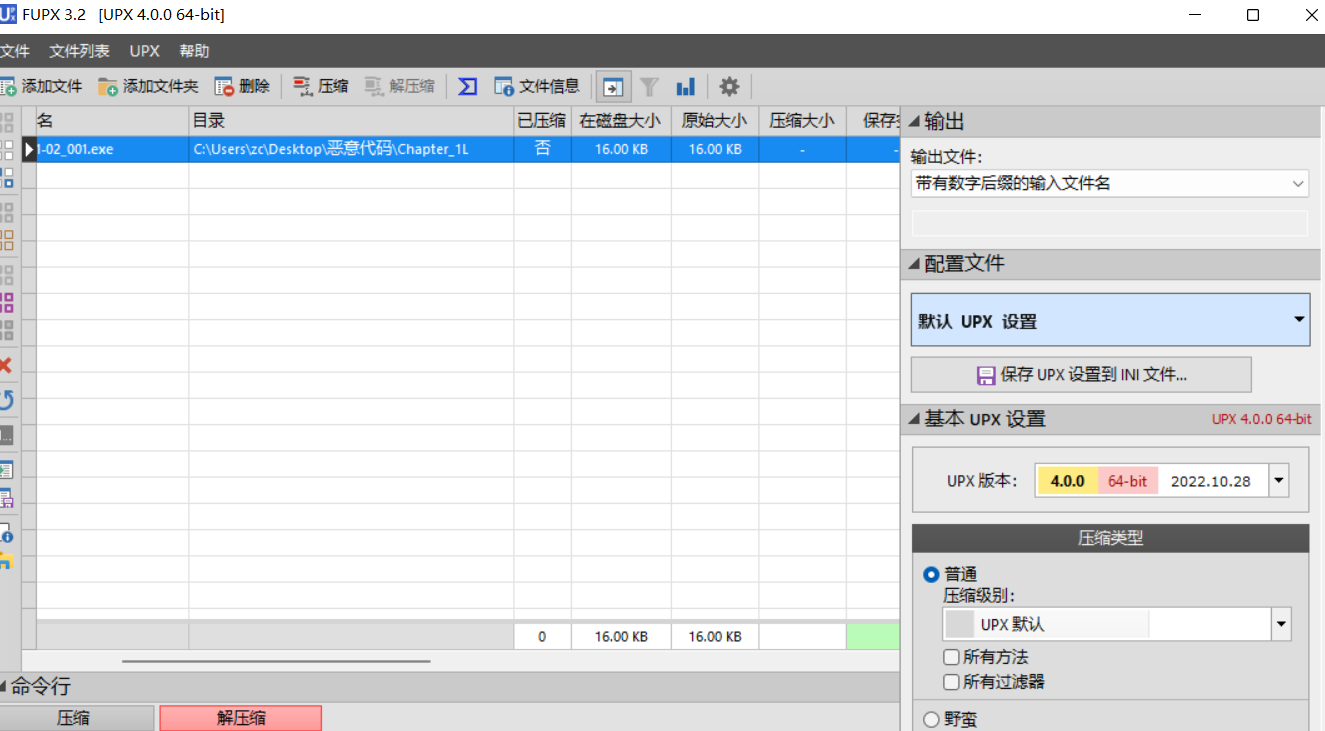
匹配到，且较于书后答案有增长

## 是否有这个文件被加壳或混淆的任何迹象？如何是这样，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

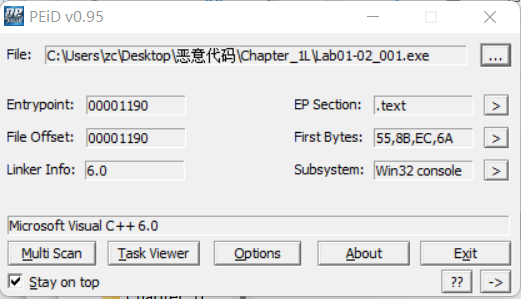


通过EP段为UPX1和”Nothing found”可以推断这个样本存在加壳的情况。

使用Fre UPX脱壳：



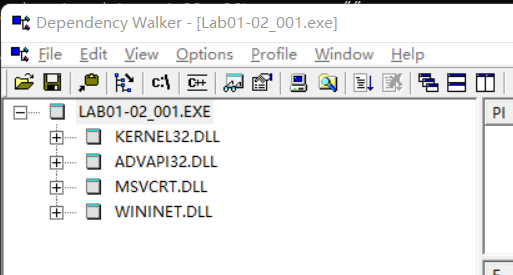
PEID分析脱壳后的exe文件：

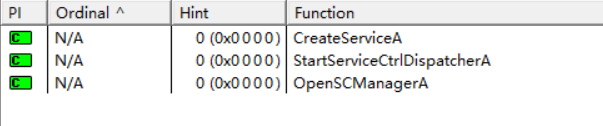


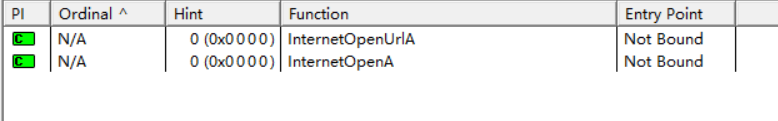
成功脱壳。

## 有没有任何的导入函数能够暗示这个程序的功能？如果是，是哪些导入函数，它们会告诉你什么？

对已脱壳新程序，分析导入函数：







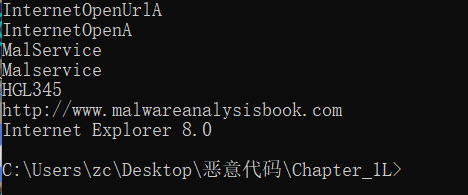
CreateService：创建一个可以在启动时刻运行的服务。CreateService这个函数来持久化、隐藏，或是启动内核驱动。

InternetOpenA：初始化WinINet中的一些高层次互联网访问函数，比如InternetOpenUrl和InternetReadFile。

InternetOpenUrlA：使用FTP、HTTP或HTTPS协议连接来打开一个特定的URL，如果URL固定，则可以作为基于网络的特征码。

故而程序可能有联网以及开启、创建服务的功能。

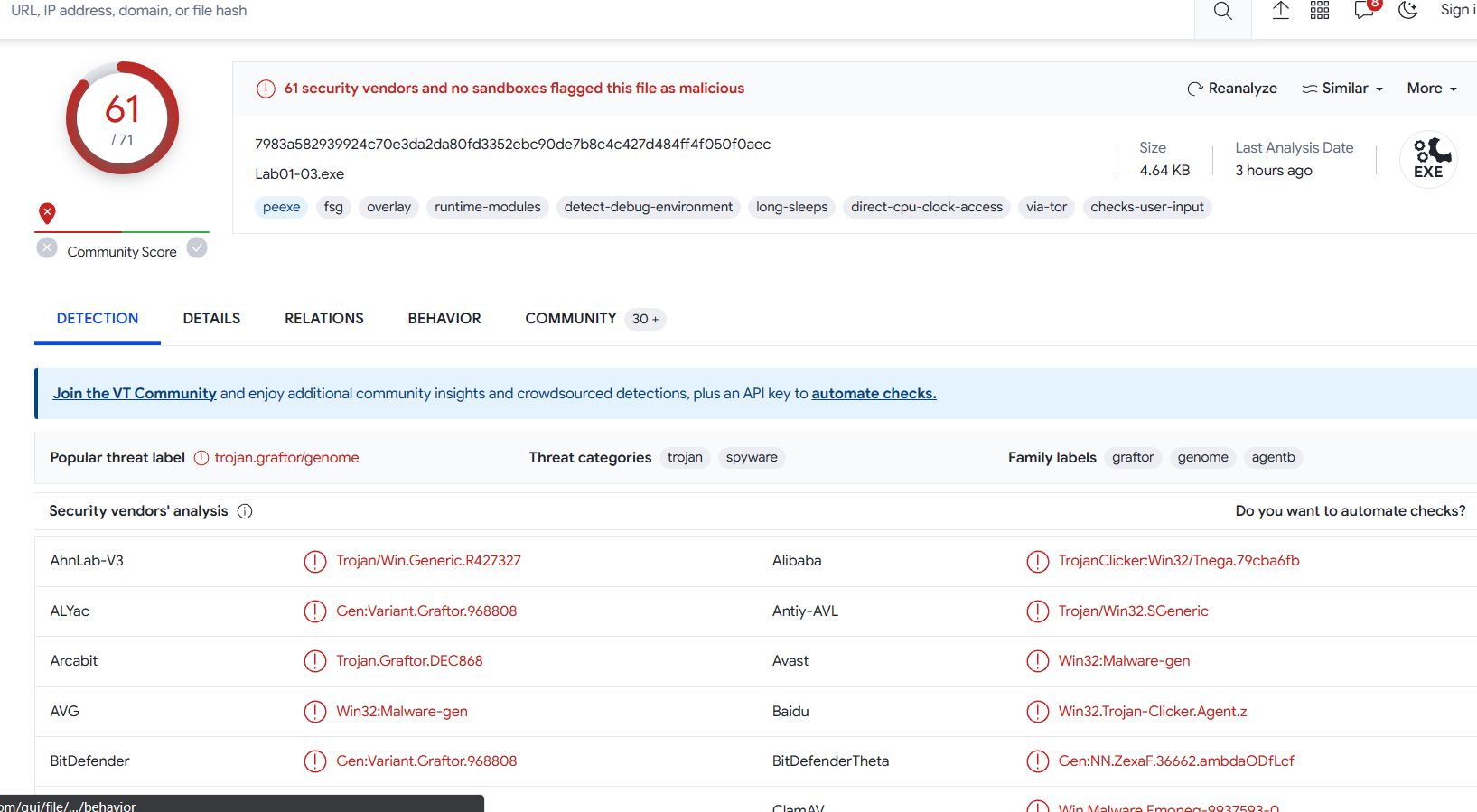
## 哪些基于主机或基于网络的迹象可以用来被确定被这个恶意代码所感染的？



strings分析，可以看到可疑的服务MalService和网址http://www.malwareanalysisbook.com。

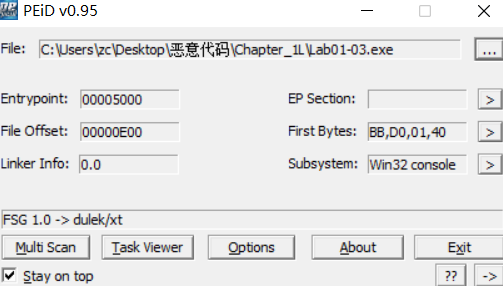
**Lab 1-3**

## **将Lab01-03.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？**



## 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？

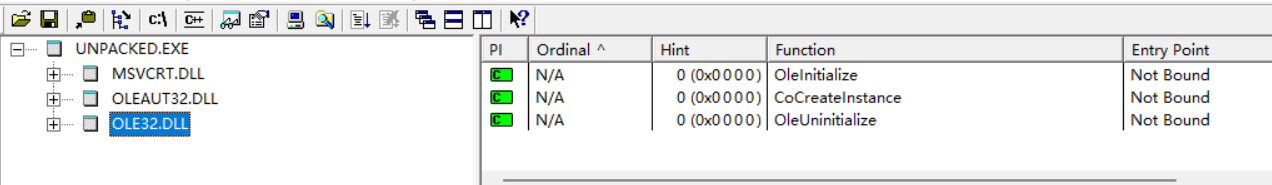
有加壳（ FSG 1.0 -> dulek/xt）:



因为加壳类型有变化，之前的FreeUPX工具无法脱壳，使用LinxerUnpacker工具成功脱壳：

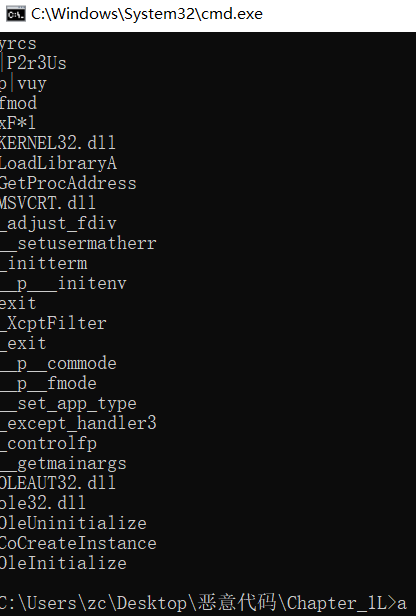
## 

## 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？



通过调用CoCreateInstance，程序可以根据提供的类标识创建特定的COM对象。类标识指定了实现COM对象的文件，因此可以暗示程序使用了特定文件中的COM功能。

## 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？



似乎找不到可疑的字符串。

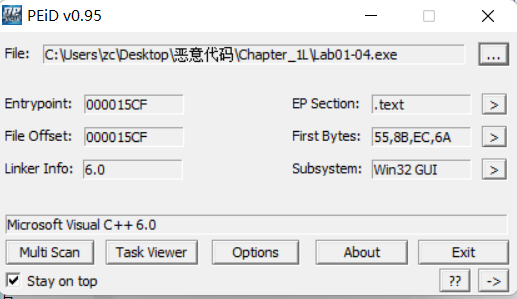
**Lab 1-4**

## **将Lab04.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?**

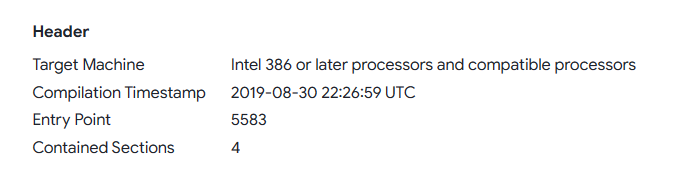
## 

## 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？

没有加壳：

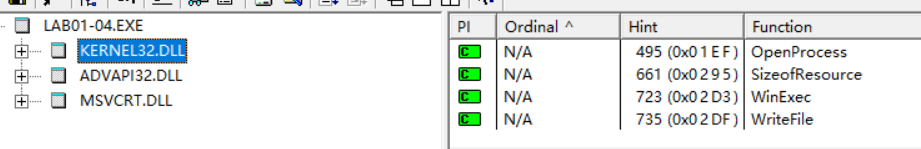


## 这个文件是什么时候被编译的？

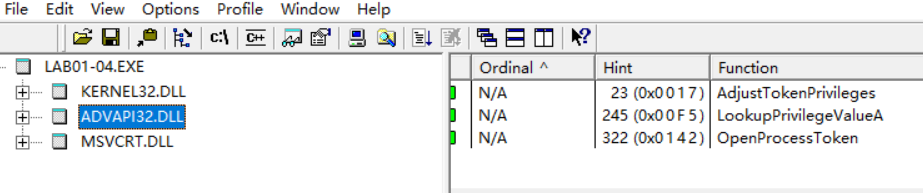


编译时间：2019.08.30 22:26:59 UTC

## 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？

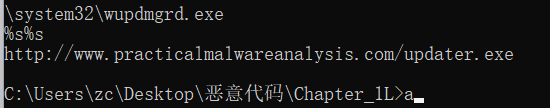


从kernel32.dll的api函数可以看出这个样本可能涉及到文件的读写、资源的查找等



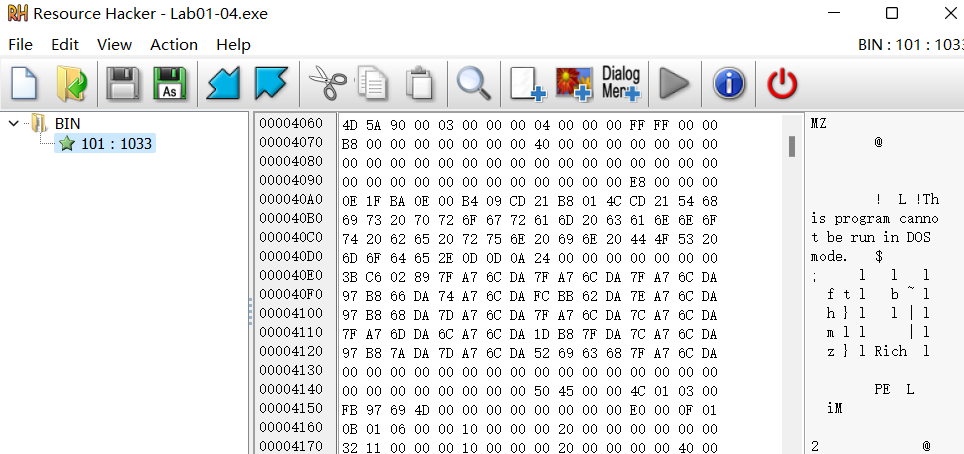
从advapi32.dll函数，可能和系统的权限有关。

## 哪些基于主机或基于网络的迹象可以用来被确定被这个恶意代码所感染的？



字符串wupdmgrd.exe表示程序可能在这个位置创建或者修改文件，下面的网址可能表示额外恶意代码的网路存储路径。

## 这个文件在资源段中包含一个资源，使用Resource Hacker工具来检测资源，然后抽取资源。从资源中你能发现什么？



资源节包含另一个PE可执行文件，用于下载额外恶意代码。

**Lab 1-5**

## 对Lab1和Lab3的样本编写Yara规则。

rule Lab1

{

meta:

description = "rules for Lab1 "

date = "202x/xx/xx"

author = "LYT"

strings:

$a = "kerne132.dll" wide ascii

$b = "127.26.152.13" wide ascii

$c = "http://www.malwareanalysisbook.com" wide ascii

$d = "wupdmgr" wide ascii

condition:

any of them

}

对于lab01-03.exe，使用strings命令并没有显示可疑字符串（多为正常dll文件或者无意义乱码）

rule Lab3

{

meta:

description = "rules for Lab3 "

date = "202x/xx/xx"

author = "LYT"

strings:

$a = "kerne132.dll" wide ascii

$b = "LoadLibraryA" wide ascii

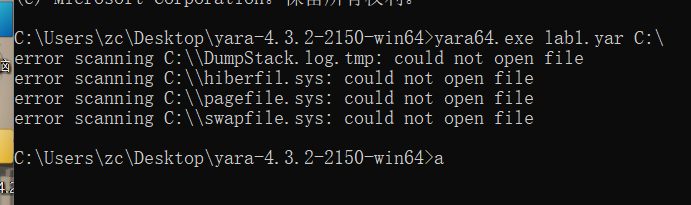
condition:

any of them

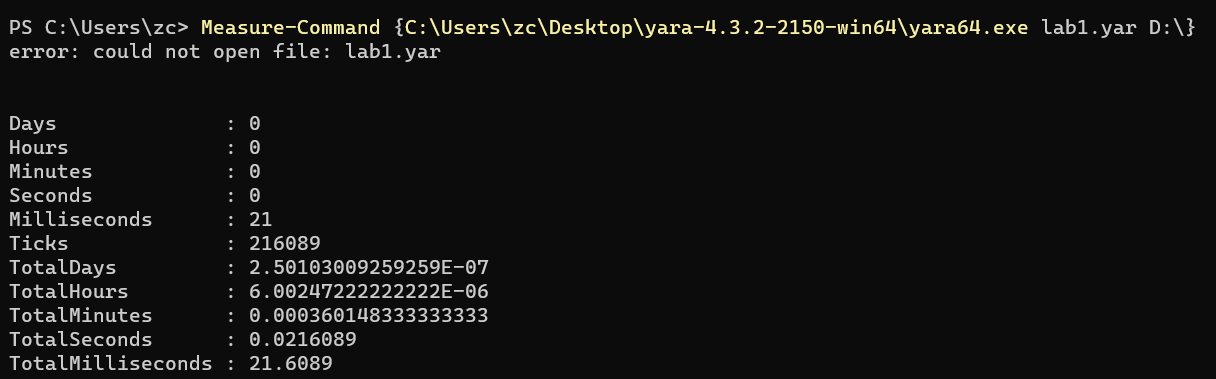
}

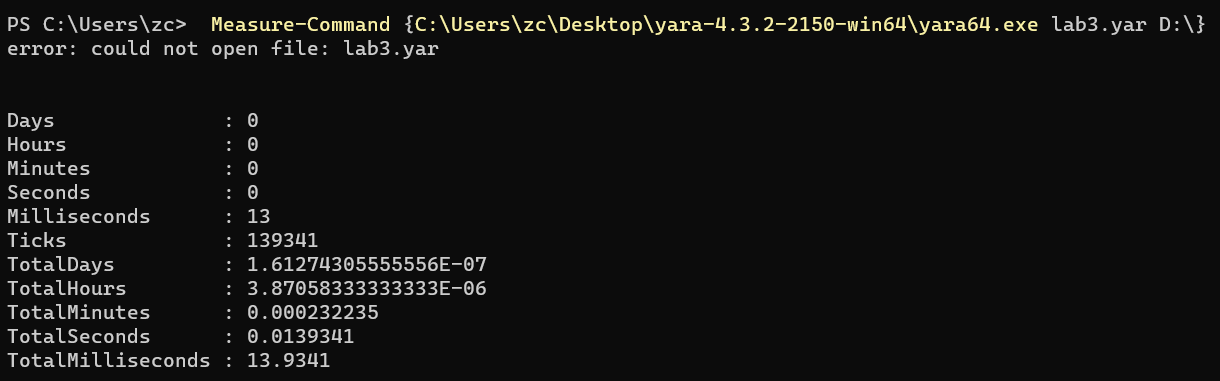
## 使用编写的规则对自己电脑的C盘进行Yara扫描，记录扫描所用时间。

因为C盘权限问题会出现下面情况：



故扫描D盘，以下是两者的时间对比：





## 讨论哪些Yara条件执行效率高，哪些低。如何改进效率低的Yara条件。

**高效条件：**

使用较短的固定字符串进行匹配，避免使用正则表达式或通配符。

例如，使用$string来匹配一个具体的字符串，而不是使用$string\*来匹配以该字符串开头的任意字符串。

**低效条件：**

使用复杂的正则表达式或通配符进行匹配，尤其是在大量字符串上进行匹配。

例如，使用$string\*来匹配以某个字符串开头的任意字符串，这可能会导致性能下降。

**改进低效的Yara条件**：

优化正则表达式和通配符的使用。尽量使用更具体的模式，避免使用过于宽泛的模式。

将复杂的匹配逻辑分解为多个简单的条件。这样可以减少每个条件的匹配数量，提高效率。

使用nocase修饰符来进行大小写不敏感的匹配。这可以减少匹配的复杂性。

使用for循环将匹配范围限制在特定的字符串列表中。这样可以减少匹配的字符串数量。

1. **实验结论及心得体会**

本次实验经过书后习题，亲手练习使用了各种静态分析工具，且根据提供的实验样本的不同，操作和分析过程也有所区别。

总之熟练了基础工具的使用，对以后的学习会有帮助。

在编写yara规则的实验部分，也进行了yara的扫描。因为是初次接触yara，对于其安装和使用也花了很多时间。

然后根据实验样本编写了简单的规则，并在电脑上进行时间的对比分析。

相信以后使用yara会更得心应手。