

**恶意代码分析与防治技术课程实验报告**

**实验一：病毒监测与yara的使用**

****

学 院 计网

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**

**学会使用windows虚拟机在隔离环境下对计算机病毒进行分析与防治。**

**学会使用PEid工具进行分析加壳与脱壳，同时提前了解FGS的相关知识。**

**学会使用PEview工具进行二进制文件的分析。**

**学会使用IDApro工具对文件的字符串和链接的函数进行分析。**

**学会使用upx工具对文件在linux和windows环境下进行加壳与脱壳的操作。**

**学会使用yara工具对相关的文件编写yara规则进行目标文件的监测。**

**学会优化yara规则，并且通过powershell监测到yara监测速度的变化。**

1. **实验原理**
   1. **实验环境**

Windows10，VMWARE，Windows11

* 1. **实验工具**

STRINGS, IDAPro,PEVIEW，YARA，upx,PEID,powershell

* 1. **原理**

Yara是virustotal发布的一个开源恶意代码查杀引擎，可以用来：

识别恶意代码

查杀恶意代码

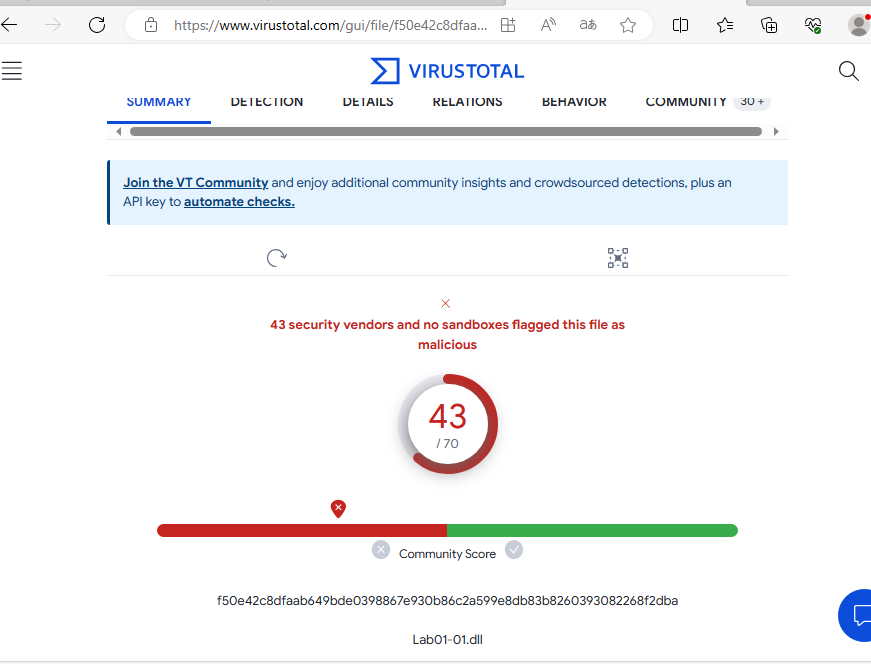
在https://virustotal.github.io/yara/上可以上传相应的恶意代码，并且生成相关的报告。

1. **实验过程**
   1. **实验为了避免病毒感染主机，在windows虚拟机环境当中运行，需要我们将老师提供的各种工具移动到虚拟机当中运行。**

Lab 1-1

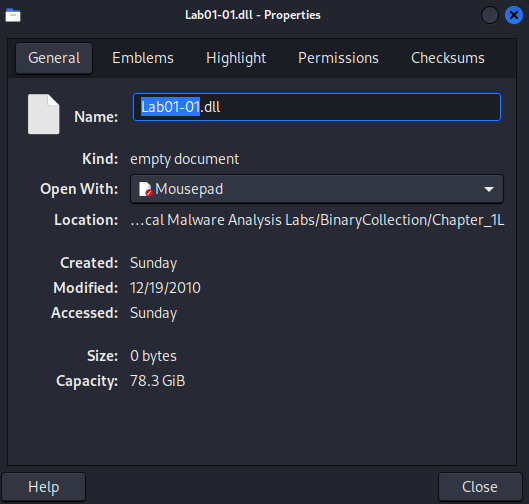
1. Upload the files to http://www.VirusTotal.com/ and view the reports. Does either file match any existing antivirus signature?

相关的内容和分析可以在virustotal当中得到相应的分析和关于该病毒的信息。

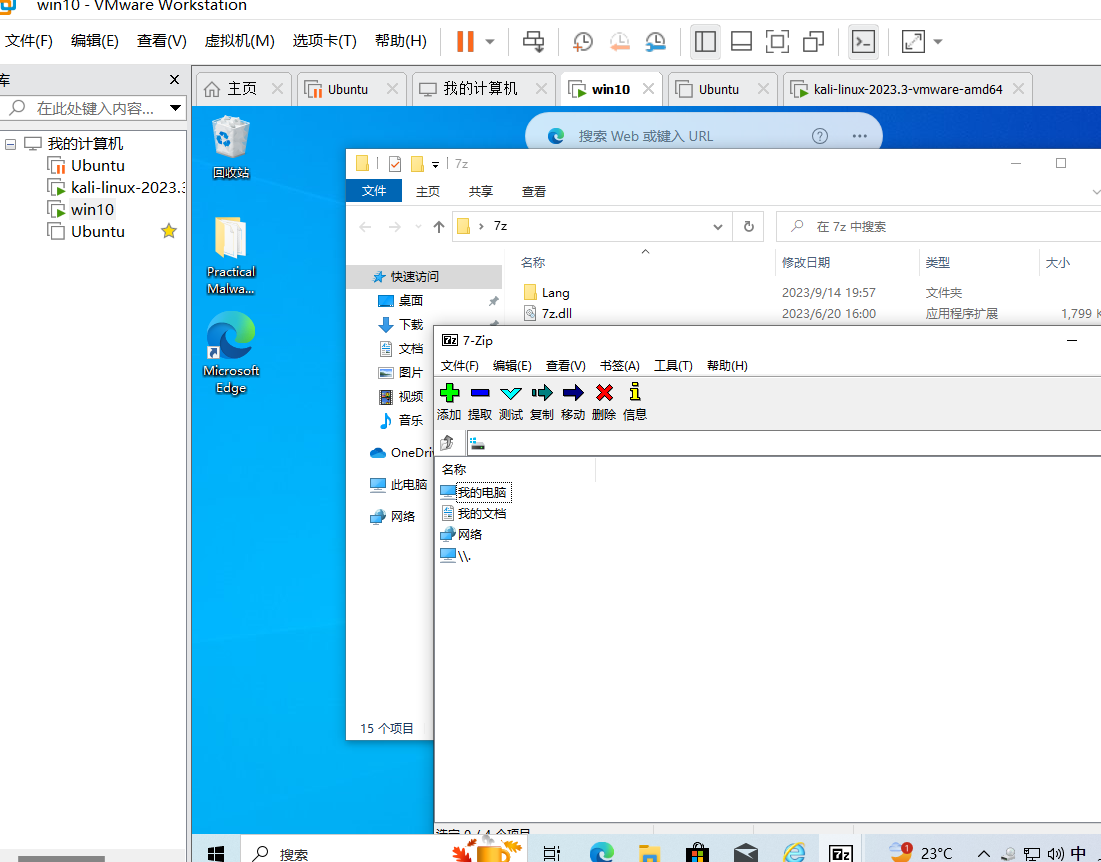


Lab01-01.exe：

Lab01-01.dll：



在kali linux当中可以看到其修改的日期。

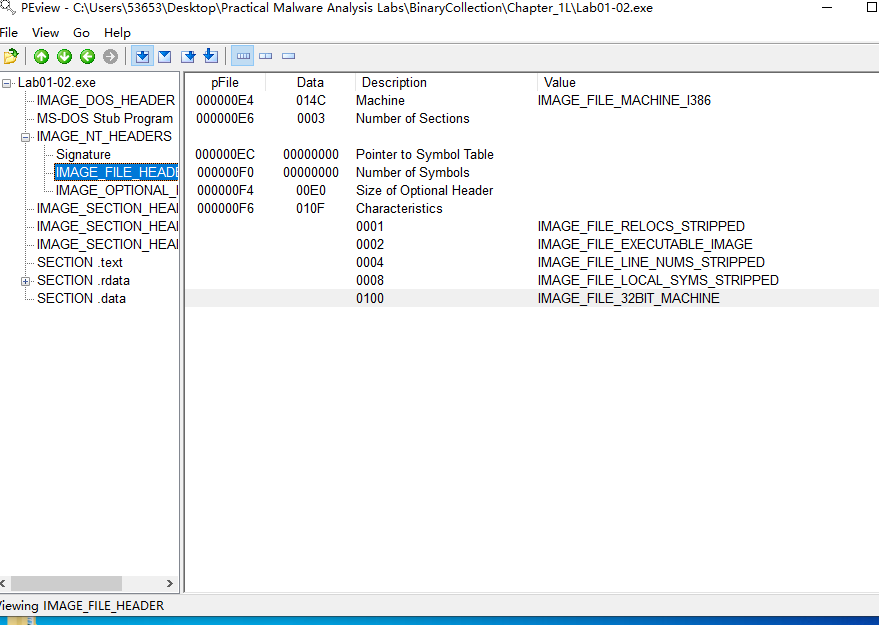


在widows当中，需要利用7z工具对相关的文件夹进行解压。

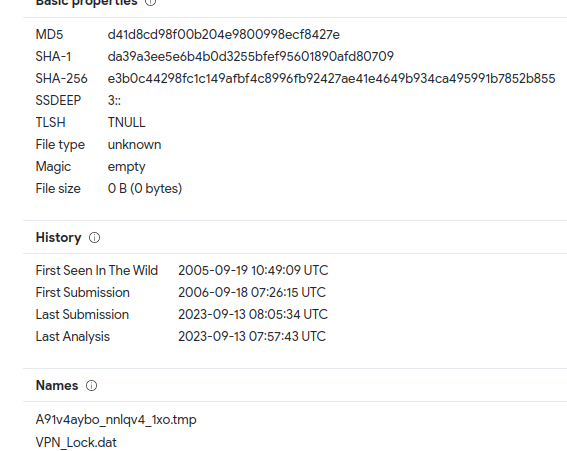
之后我尝试利用PEview工具进行观察时间戳，我尝试在

IMAGE-HEADERS-IMAGE-FILE-HEADERS里寻找关于时间戳的信息，但是出现报错后并没有找到时间戳的位置。

于是我重新打开了vitustotal的报告，找到了时间戳的位置。



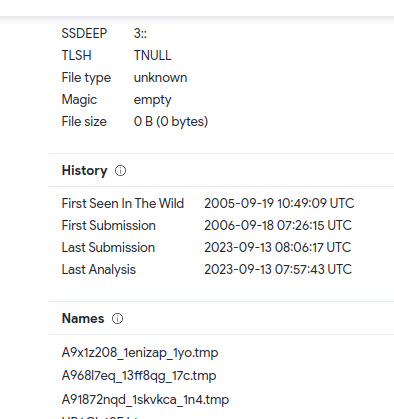
在报告当中寻找其中的编译日期。



可以看出，这两个文件都已被一些安全软件识别为病毒，匹配到了已有的反病毒软件特征。

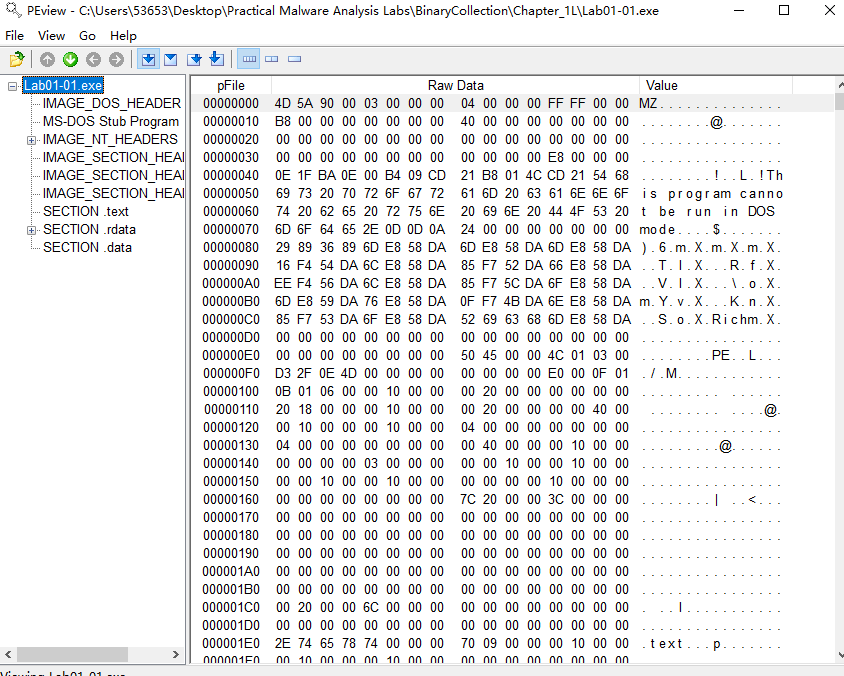
2. When were these files compiled?

查看 Virus Total 给出的报告，可以查到 PE 头的相关信息，其中包含文件的编译时间。



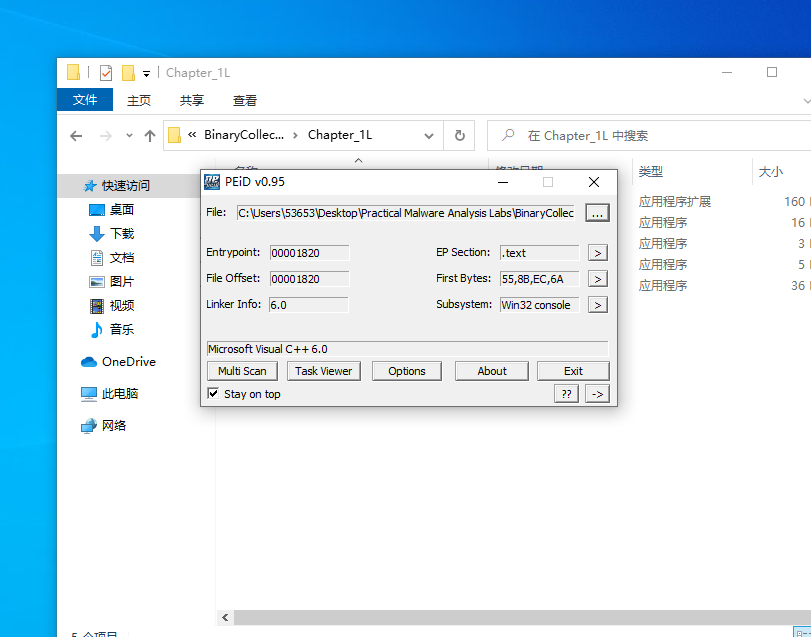
从而知道，Lab01-01.exe 的编译时间为 2010-12-19 16:16:19；Lab01-01.dll 的编译时间为

2010-12-19 16:16:38。二者的编译时间都在1min之内。

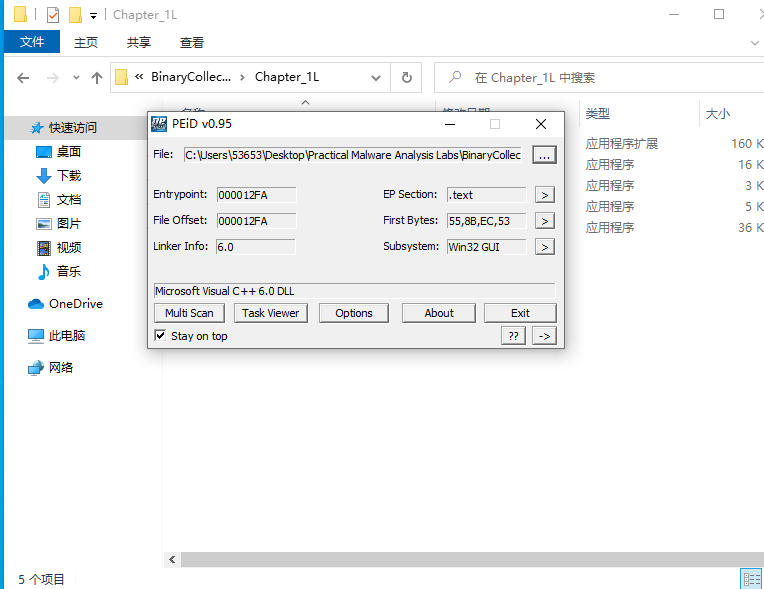


3. Are there any indications that either of these files is packed or obfuscated? If so, what are these indicators?

直接使用查壳工具 PEid 扫描文件，结果如下，可见没有加壳和混淆。



Lab01-01.exe和lab01-01.dll都是没有加壳的文件，都可以直接通过PEid扫描到其中的很多信息。



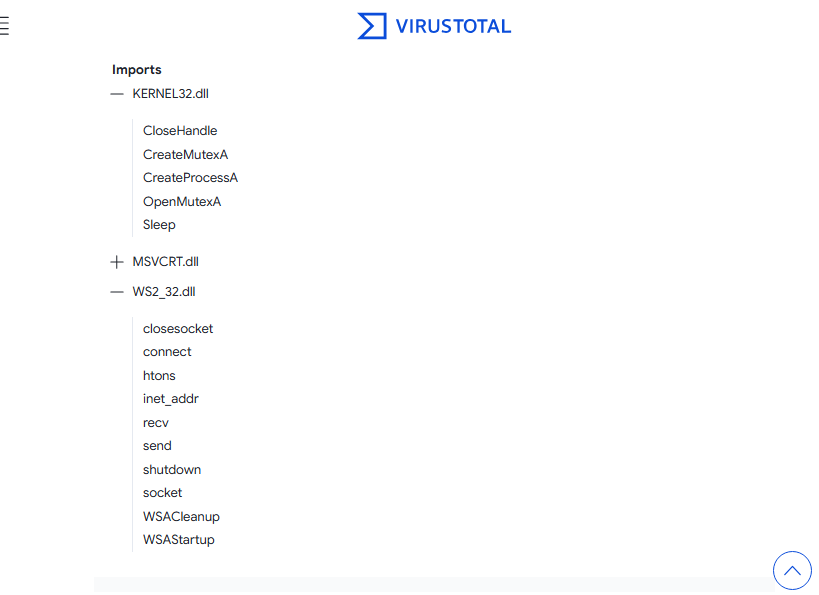
Lab01-01.exe： Lab01-01.dll：

4. Do any imports hint at what this malware does? If so, which imports are they?

查看 Virus Total 给出的报告，可以知道文件中包含了哪些导入函数。

Lab01-01.exe：

其中值得关注的是 FindFirstFileA、FindNextFileA 和 CopyFileA，这三个函数配合使用，可以搜索文件系统里的所有文件并复制。文件当中只有很少的函数，说明他们可能只是一些小程序或者小漏洞。



Lab01-01.dll：

其中 CreateMutexA 函数用于创建一个互斥对象，可以和 OpenMutexA 一起操作一个互斥对象；CreateProcess 函数创建并启动一个新进程，如果恶意软件创建一个新的进程，新的流程需要分析；Sleep 函数可以使计算机程序进入休眠。由此可以猜想，这个 DLL 会创建一个互斥变量以保证一个资源同时只能被一个进程使用，如果这个互斥变量被锁，则其他的进程就执行 Sleep 函数等待。

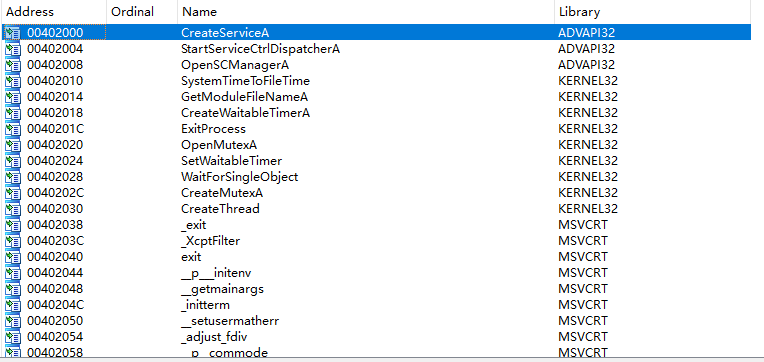
而 WS2\_32.dll 提供联网功能，该恶意软件很可能要执行网络相关的任务。

5. Are there any other files or host-based indicators that you could look for on infected systems?

使用 IDA 分析文件 Lab01-01.exe，观察 Strings 窗口，如下：

出现了 kerne132.dll，这里企图用 1(one)混淆了 l，说明这台主机已经被感染了。

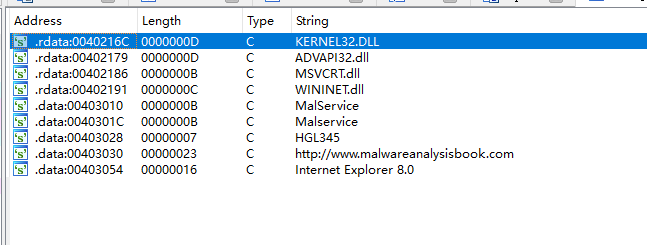
此外，在下一行还会发现这个.exe 在运行时会导入 Lab01-01.dll，在上一题中已经分析了这个.dll 文件可能出现的恶意行为。

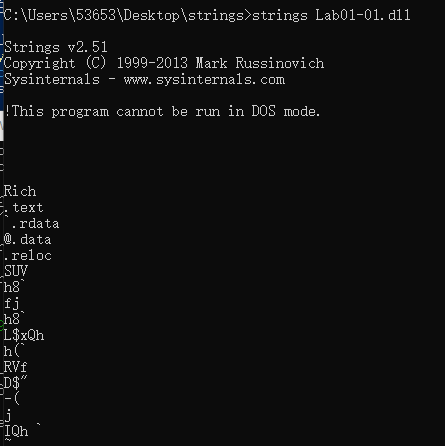


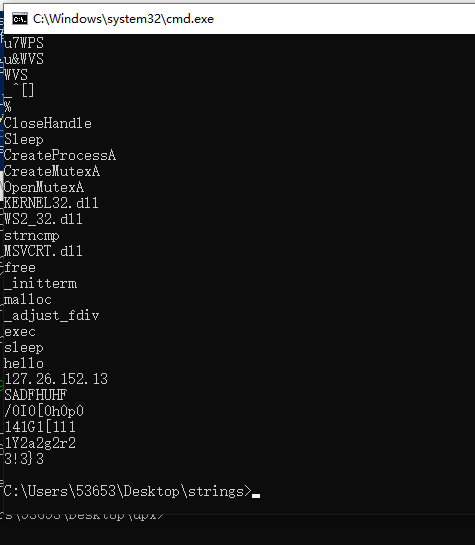
6. What network-based indicators could be used to find this malware on infected machines?

使用 IDA 分析文件 Lab01-01.dll，观察 Strings 窗口，如下：

可以发现这里有一个 ip 地址，而在第四题分析导入函数时已经知道 WS2\_32.dll 提供联网功能，该恶意软件很可能要执行网络相关的任务，因此可以推测是要与这里提到的网址联网通信。







7. What would you guess is the purpose of these files?

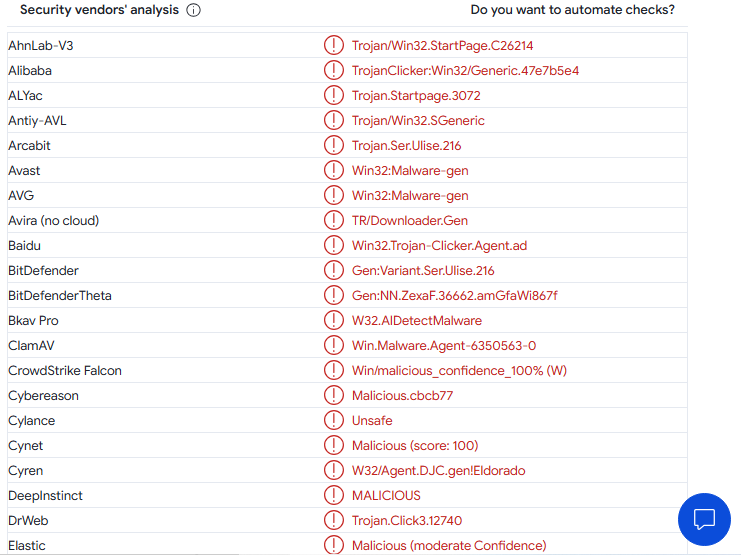
通过以上分析可以推测，.dll 文件可能是一个后门，.exe 文件是用来安装与运行.dll 文件的。

Lab 1-2

1. Upload the Lab01-02.exe file to http://www.VirusTotal.com/ and view the reports. Does it match any existing antivirus signature?

Lab01-02.exe：

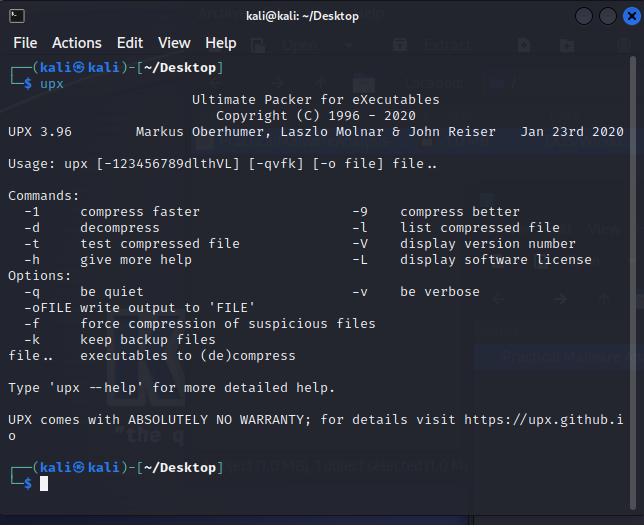


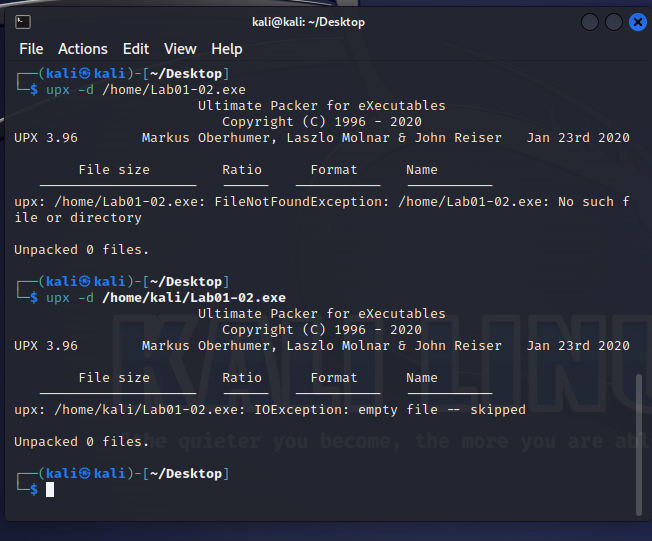


可以看出，这个文件都已被一些安全软件识别为病毒，匹配到了已有的反病毒软件特征。

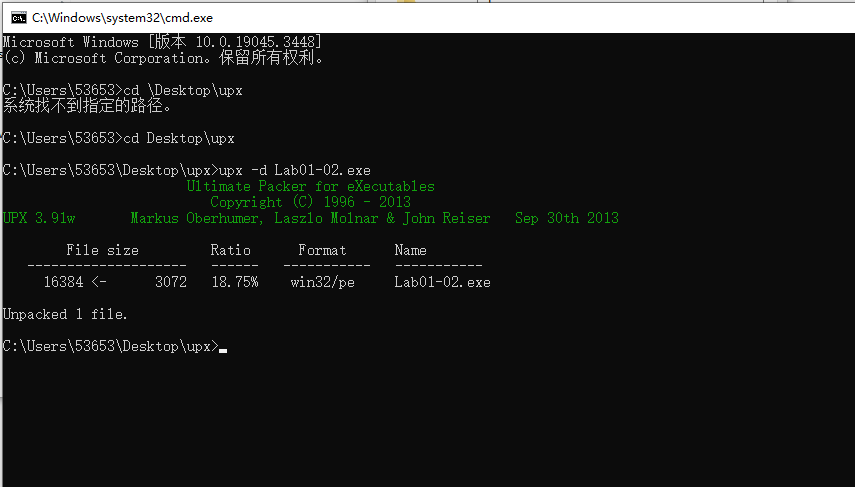
2. Are there any indications that this file is packed or obfuscated? If so, what are these indicators? If the file is packed, unpack it if possible. 使用查壳工具 PEid 扫描文件，结果如下，说明该文件已经被加壳了。

使用 Kali 的 upx -d 命令脱壳，脱壳成功！



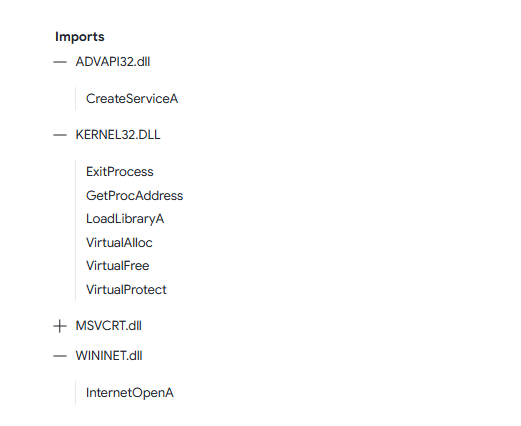


同样也可以使用windows下的upx软件进行脱壳，upx脱壳的结果可以用于我们继续在windows系统下进行PEid的分析和脱壳后函数的观察。

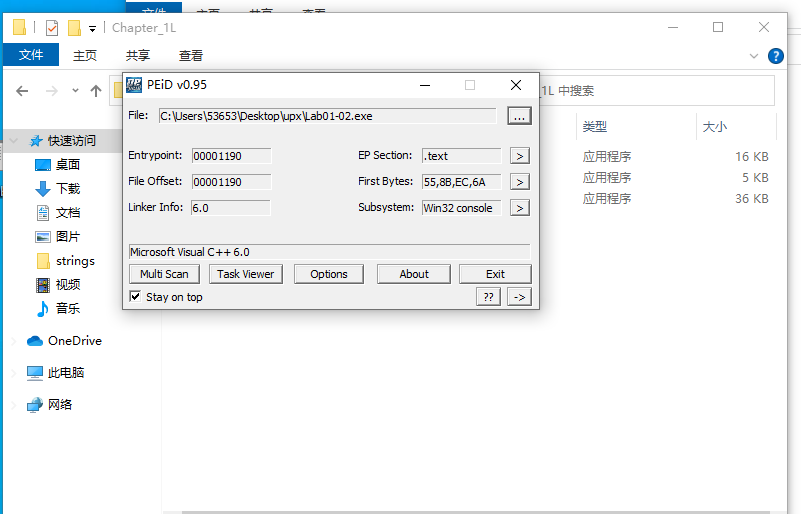


3. Do any imports hint at this program’s functionality? If so, which imports are they and what do they tell you?

将脱壳后的 Lab01-02.exe 文件上传到 Virus Total 网站上，得到与脱壳前不同的导入函数，如下：



进而可以说明脱壳可以给文件的功能发生一定的变化。

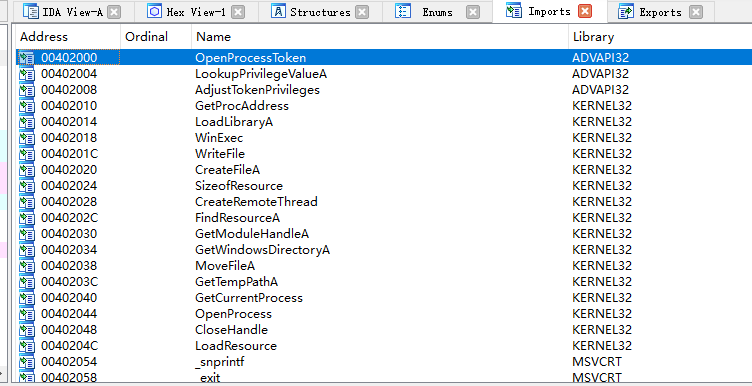


推测该程序将有开启或创建服务、联网等行为。

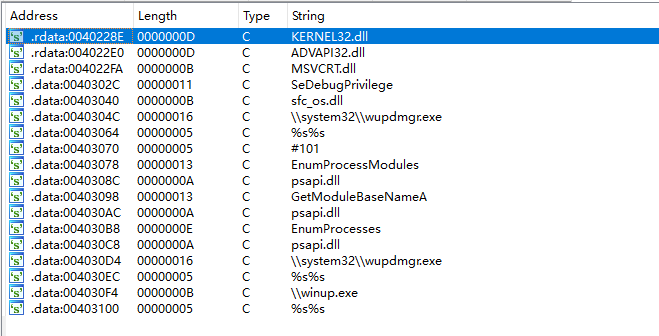
4. What host- or network-based indicators could be used to identify this malware on infected machines?

使用 IDA 分析文件 Lab01-02.exe，观察其函数窗口和 Strings 窗口，如下：

有一些可疑字符串和可疑网址。



其中的一些可以字符串我们可以用来进行yara规则的编写。

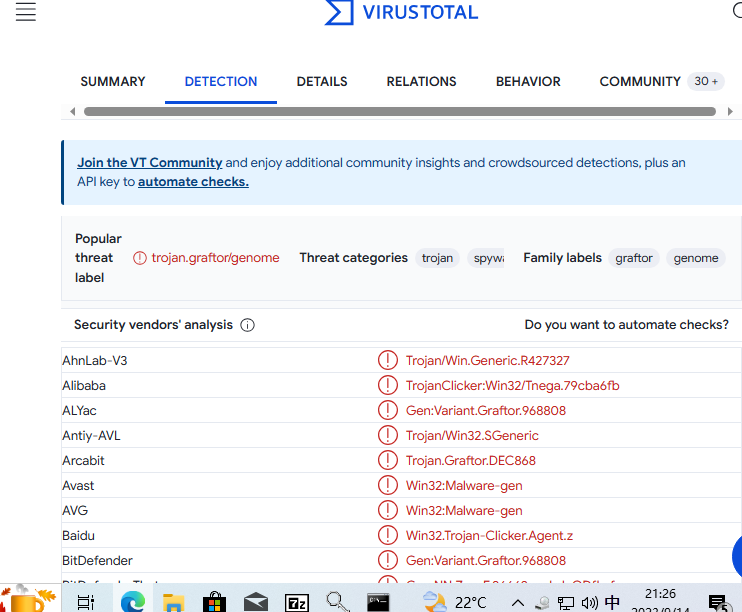


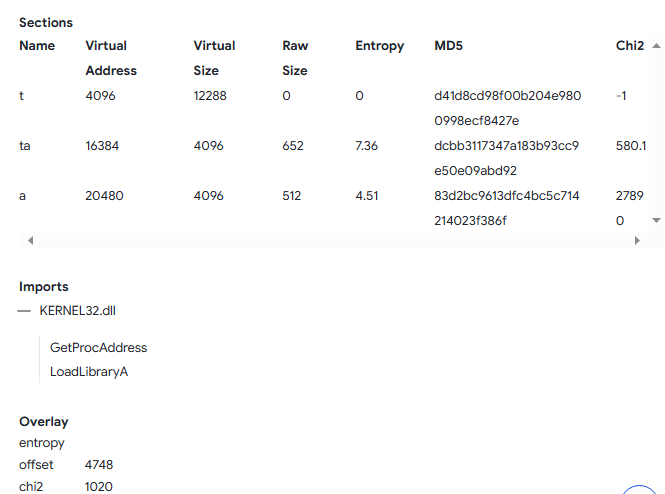
Lab 1-3

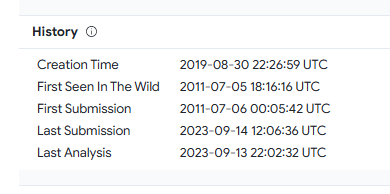
1. Upload the Lab01-03.exe file to http://www.VirusTotal.com/ and view the reports. Does it match any existing antivirus signature?

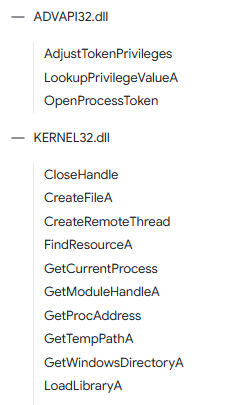
Lab01-03.exe：

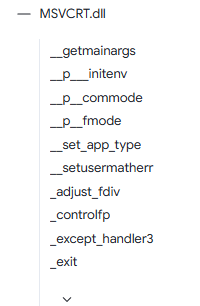
可以看出，这个文件都已被一些安全软件识别为病毒，匹配到了已有的反病毒软件特征。











1. Are there any indications that this file is packed or obfuscated? If so, what are these indicators? If the file is packed, unpack it if possible. 使用查壳工具 PEid 扫描文件，结果如下，说明该文件已经被加壳了。

我们还可以搜索相应的导入表，却发现其没有明显的导入表，说明这是一个加壳后的文件。

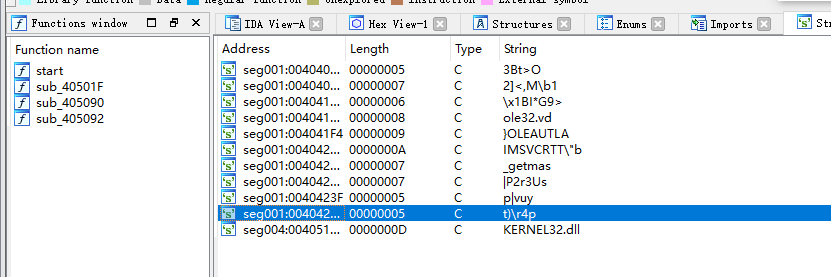
使用脱壳工具 Linxer Unpacker 进行壳特征脱壳，脱壳成功！

3. Do any imports hint at this program’s functionality? If so, which imports are they and what do they tell you?

将脱壳后的 Lab01-03.exe 文件上传到 Virus Total 网站上，得到与脱壳前不同的导入函数，该程序可能将有组件对象模型 COM、联网等相关操作。

4. What host- or network-based indicators could be used to identify this malware on infected machines?

使用 IDA 分析文件 Lab01-03.exe，观察 Strings 窗口，如下：可以看到一些可疑字符串。



Lab 1-4

1. Upload the Lab01-04.exe file to http://www.VirusTotal.com/ and view the reports. Does it match any existing antivirus signature?

Lab01-04.exe：

可以看出，这个文件都已被一些安全软件识别为病毒，匹配到了已有的反病毒软件特征。

2. Are there any indications that this file is packed or obfuscated? If so, what are these indicators? If the file is packed, unpack it if possible. 使用查壳工具 PEid 扫描文件，结果如下，说明该文件没有被加壳或混淆。

3. When were these files compiled?

查看 Virus Total 给出的报告，可以查到 PE 头的相关信息，其中包含文件的编译时间。

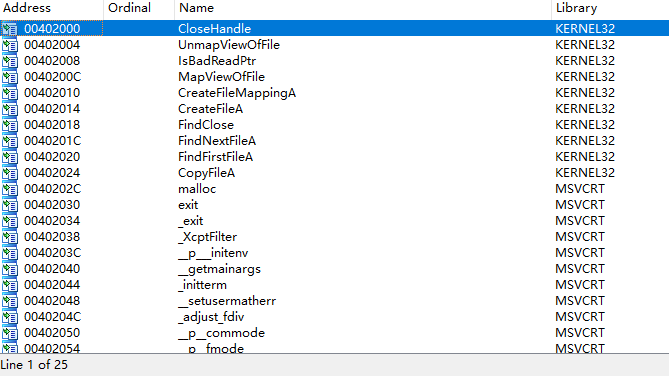
因此这个文件的编译时间为 2019-08-30 22:26:59。（但我认为这是个不可信的时间）

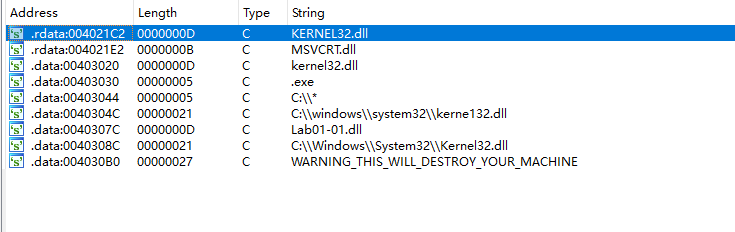
4. Do any imports hint at this program’s functionality? If so, which imports are they and what do they tell you?

该程序可能将有操作权限、创建可执行文件并运行、联网等行为。

5. What host- or network-based indicators could be used to identify this malware on infected machines?

使用 IDA 分析文件 Lab01-04.exe，观察 Strings 窗口，如下：可以看到一些可疑字符串。其中字符串”\\system32\\wupdmgr.exe”表示这个程序会在这个位置创建或者修改文件。





6. This file has one resource in the resource section. Use Resource Hacker to examine that resource, and then use it to extract the resource. What can you learn from the resource?

**接下来就是使用yara规则和yara程序对相应的病毒文件进行扫描**

**首先我们编写并利用以下的yara规则进行扫描。**

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

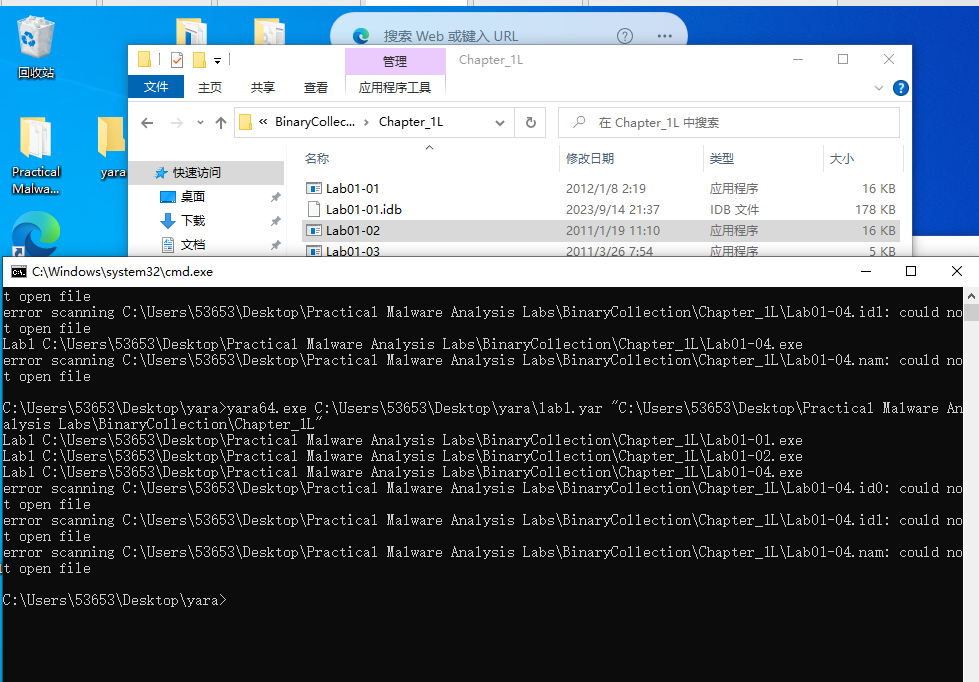
$d = "wupdmgr" wide ascii

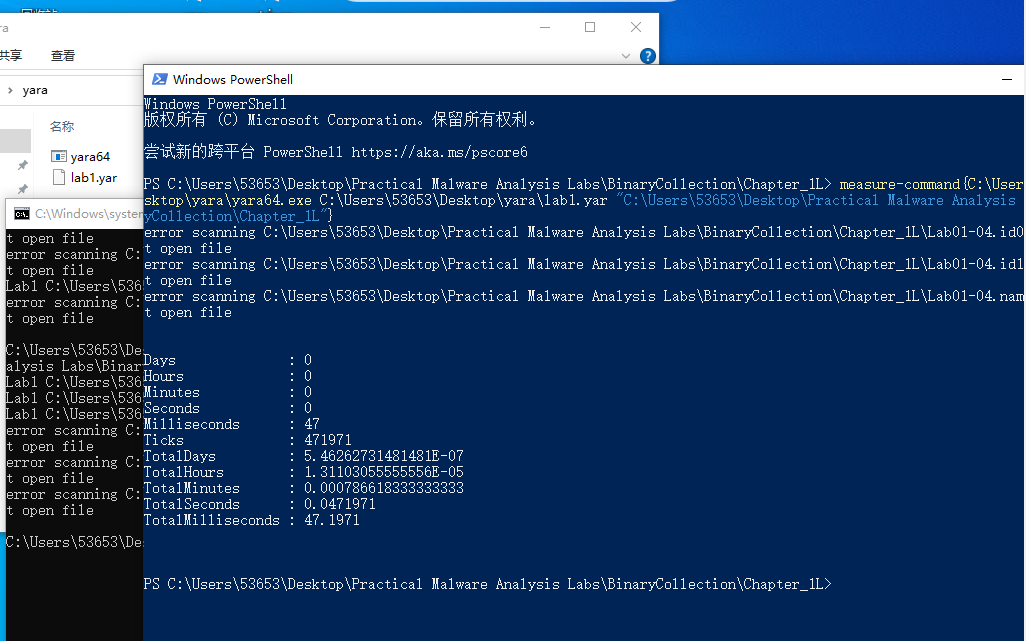
condition:

any of them

}

**得到以下的输出结果，并且在powershell当中得到相应的运行时间。**

****

****

**接下来我们更改yara规则，试图对我们扫描的过程进行优化。**

rule Lab1

{

meta:

description = "rules for Lab1"

date = "202x/xx/xx"

author = "LYT"

type = "malware-analysis"

strings:

$kernel\_dll = /kerne\d+\.dll/ nocase

$ip\_address = /127\.26\.152\.13/

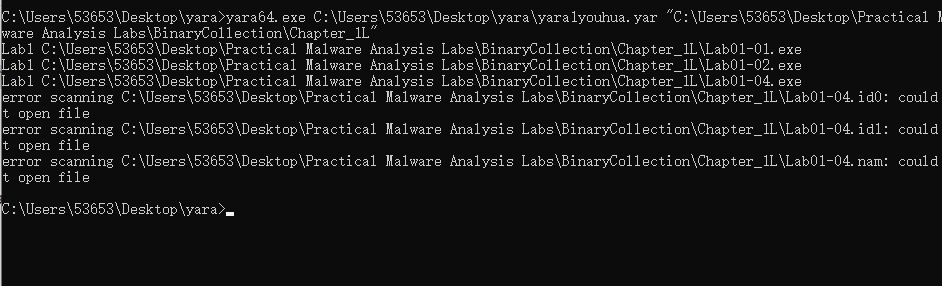
$url = /http:\/\/www\.malwareanalysisbook\.com/

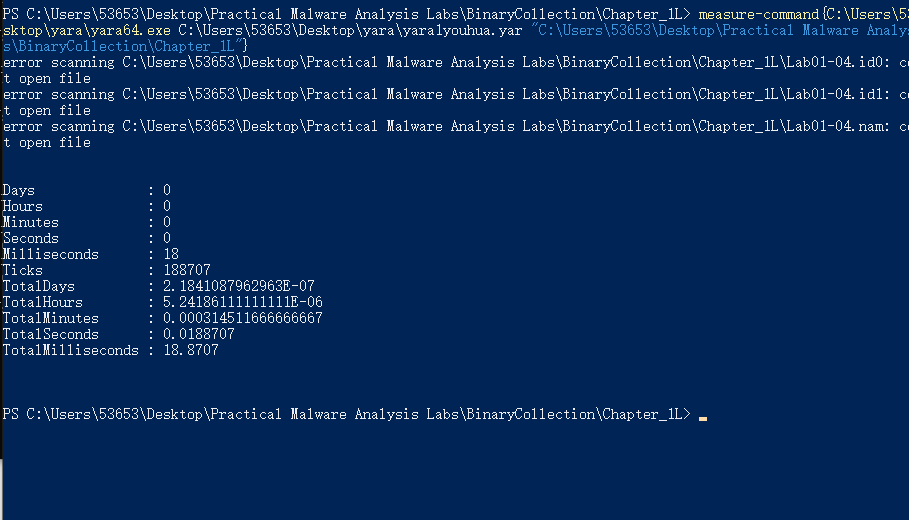
$process\_name = /wupdmgr/

condition:

any of them

}

****

****

**可以看到运行时间从47.1971降低到了18.8707，可以说明我们得到了一个比较好的结果，成功进行了相应的优化。**

1. **实验结论及心得体会**

**通过本次实验，我进行了许多病毒程序的分析和yara扫描的工作，我发现了其中有很多的应用软件需要得到运用，还有许多的软件其中具有很多不同的功能。例如PEview可以将PE文件的许多问题都能展现，PEid可以很好的扫描加壳，strings对字符串的扫描非常精准，upx的脱壳简单有效，yara的扫描方便快捷，ida的使用一直十分全面方便。**

**yara规则的使用有很多的优化空间**

**优化字符串匹配：**

**1.使用正则表达式来匹配字符串，可以提高灵活性和减少规则中的字符串数量。**

**对于IP地址和URL等可能存在变化的值，可以使用通配符或正则表达式部分匹配。**

**2.精确定义字符串类型：**

**指定更具体的字符串类型，如wide ascii替换为wide或ascii，以减少匹配的计算量。**

**3.考虑添加更多具有辨别力的字符串：**

**根据你的需求，可以添加更多独特而与恶意行为相关的字符串。**

**4.使用元数据进行分类和分组：**

**使用元数据来对规则进行分类和分组，以便更好地管理和组织规则集。**

**通过对以上的yara规则优化方案的使用，我们可以很好的发现yara规则可以有效的减少相应的程序运行时间，提高程序运行的效率。**