个人信息

学号：1911410

姓名：付文轩

专业：信息安全

lab 4

要求

1. Lab1和Lab3样本的yara规则
2. Lab1和Lab3的yara规则扫描C盘的时间
3. 自己对yara规则进行优化的思路和建议

实验过程

编写yara规则

根据之前在lab1和lab3中strings检测的结果，可以发现这里面除了一些共有的API调用（包括系统自己也一般会使用的）之外，其他主要比较敏感、需要关注的内容就是关于**注册表信息的修改**、**文件读写**、**进程创建**、**其他exe程序调用**、**service服务**、**URL或者IP的访问**、**安装或者卸载操作**、**睡眠**、**下载文件**，基于以上这些内容编写了如下的yara规则

```
import "pe"
import "hash"

rule URLRequest {
     meta:
          description = "Maybe malware will request IP or URL"
     strings:
          $IPV4 = /((\d|[1-9]\d|1\d\d|2[0-4]\d|25[0-5])\.){3}(\d|[1-9]\d|1\d\d|2[0-4]\d|25[0-5])/
          $url = /^((ht|f)tps?):\/\/([\w\-]+(\.[\w\-]+)*\/)*[\w\-]+(\.[\w\-]+)*\/?(\?([\w\-\.,@?^=%&:\/~\+#]*)+)?/
          $dll = "WS2_32.dll"
      condition:
          $IPV4 or $url or $dll
}

rule FileRevise {
    strings:
        $CreateFile = /CreatFile[a-zA-Z]*/
        $CopyFile = /CopyFile[a-zA-Z]*/
        $WriteFile = "WriteFile"
        $MoveFile =/MoveFile[a-zA-Z]*/
    condition:
        $CreateFile or $CopyFile or $WriteFile or $MoveFile
```

```
}

rule Service {
    strings:
        $CreateService = /CreateService[a-zA-Z]*/ nocase
        $InternetOpen = /InternetOpen[a-zA-Z]*/
    condition:
        $CreateService or $InternetOpen
}

rule DownloadFile {
    strings:
        $URLDownload = /URLDownloadToFile[a-zA-Z]*/
    condition:
        $URLDownload
}

rule ReviseRegedit {
    strings:
        $KeyName =
/HKEY(_CLASSES_ROOT|_CURRENT_USER|_USERS|_LOCAL_MACHINE|_CURRENT_CONFIG)/
        $Regedit = /software(\\[a-zA-Z]*)*/ nocase
        $HKLM = /HTLM(\\[a-zA-Z]*)*/
        $RegAPI = /Reg[a-zA-Z]*/
    condition:
        $KeyName or $Regedit or $HKLM or $RegAPI
}

rule OtherEXE {
    strings:
        $exe = /[a-zA-Z0-9_]+.exe/
    condition:
        $exe
}

rule Install {
    strings:
        $InstallAPI = /(i|I)nstall[a-zA-Z0-9]*/ nocase
        $unInstall = /(U|u)ninstall[a-zA-Z0-9]*/ nocase
    condition:
        $InstallAPI or $unInstall
}

rule Sleep {
    strings:
        $Sleep = "sleep" nocase
    condition:
        $Sleep
}
```

检查Lab1中的文件

对于yara的使用，通用的情况就是`yara.exe rules sample`，当要对一个文件夹进行扫描时，需要加上一个参数-r，扫描结果如下

```
D:\Study\terms\3.Junior\FirstSemester\计算机病毒与防治技术（王志）\homework>yara64.exe -r rules.yar Chapter_1L
warning: rule "URLRequest" in rules.yar(8): string "$IPV4" may slow down scanning
FileRevise Chapter_1L\Lab01-01.exe
Service Chapter_1L\Lab01-02.exe
FileRevise Chapter_1L\Lab01-04.exe
DownloadFile Chapter_1L\Lab01-04.exe
OtherEXE Chapter_1L\Lab01-04.exe
URLRequest Chapter_1L\Lab01-01.dll
Sleep Chapter_1L\Lab01-01.dll
```

可以看见显示出了很多条检测结果，如：

01.exe中有对文件的编辑，02.exe中有对服务的操作，01.dll中有网络请求和睡眠，04.exe中有从网上下载文件、修改文件、以及对其他exe程序的操作

其中缺少的对03.exe的显示，也就是说上述的规则并没有匹配在03.exe中匹配到。经过对03.exe的观察发现这个文件是有加壳的，能直接匹配到的字符串并不多，其中可以作为标志的是`LoadLibraryA`和`GetProcAddress`以及对`Kernel32.dll`的调用，后续也可以把这个加入其中。

同时可以看见第一条中有一个提示：warning: rule "URLRequest" in rules.yar(8): string "$IPV4" may slow down scanning。也就是说写的那个对IP地址检查的正则表达式会降低检测效率，应当考虑采取优化。

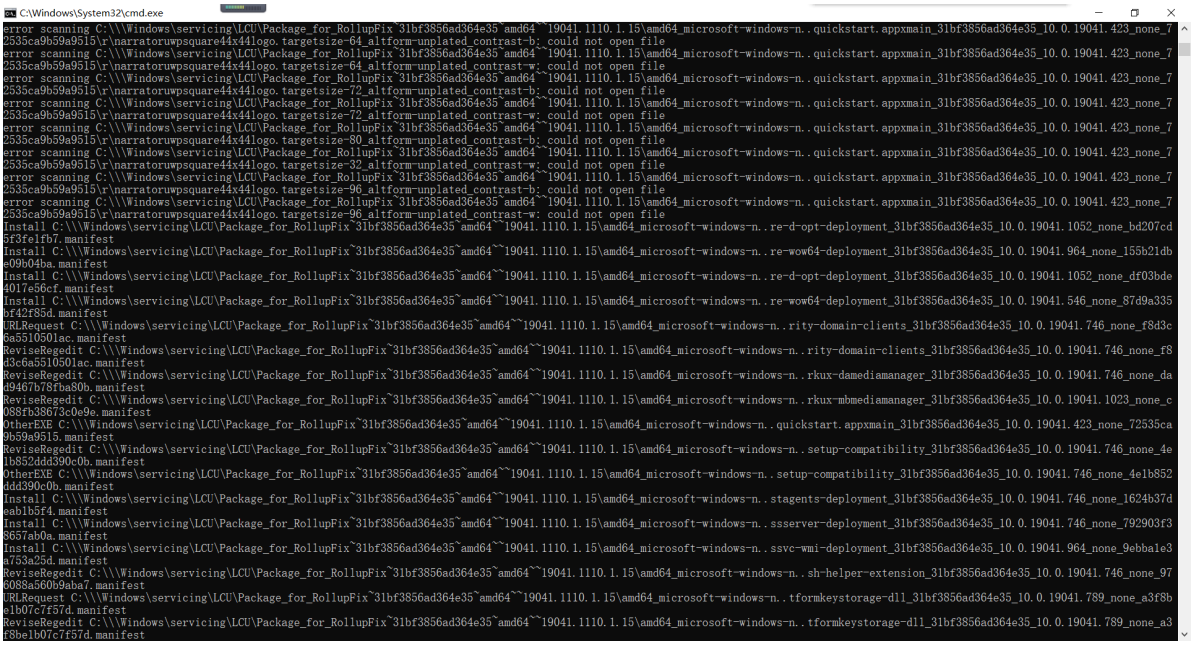
检查Lab3中的文件

```
D:\Study\terms\3.Junior\FirstSemester\计算机病毒与防治技术（王志）\homework>yara64.exe -r rules.yar Chapter_3L
warning: rule "URLRequest" in rules.yar(8): string "$IPV4" may slow down scanning
ReviseRegedit Chapter_3L\Lab03-01.exe
OtherEXE Chapter_3L\Lab03-01.exe
Install Chapter_3L\Lab03-01.exe
FileRevise Chapter_3L\Lab03-03.exe
OtherEXE Chapter_3L\Lab03-03.exe
Sleep Chapter_3L\Lab03-03.exe
URLRequest Chapter_3L\Lab03-04.exe
FileRevise Chapter_3L\Lab03-04.exe
Service Chapter_3L\Lab03-04.exe
ReviseRegedit Chapter_3L\Lab03-04.exe
OtherEXE Chapter_3L\Lab03-04.exe
Sleep Chapter_3L\Lab03-04.exe
URLRequest Chapter_3L\Lab03-02.dll
Service Chapter_3L\Lab03-02.dll
ReviseRegedit Chapter_3L\Lab03-02.dll
OtherEXE Chapter_3L\Lab03-02.dll
Install Chapter_3L\Lab03-02.dll
```

检查结果类似于Lab1，这里就不多描述了

检查C盘中的文件

由于C盘中的文件有非常多，显示的结果也非常多，但并不是说C盘中就有很多的病毒，只是因为C盘是系统盘，其中很多文件都会有这些操作，这里只截取了一小部分的扫描结果



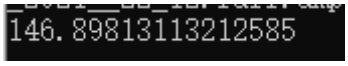
可以看到其中有一部分扫描的时候出错，猜测应该是因为权限不足的问题。如果换成使用管理员模式打开cmd应该可以解决这个问题（这里就体现出linux中的方便，直接在命令行中加上sudo就可以提升到管理员权限，或者su root）

为了方便时间的统计，写了一个简单的python的脚本执行扫描和时间统计，代码如下：

```python
#!/usr/bin/env python3
# -*- coding:utf-8 -*-

import os, time
begin_time = time.time()
os.system(r"yara64.exe -r rules.yar C:\\")
end_time = time.time()
print(end_time - begin_time)
```

得到扫描的时间如下：



可以看出扫描了接近147s，也就是接近2分半

优化思路和建议

首先yara如果说的直白一点，其实就是一个将文件以二进制格式打开，去对规则一个个进行匹配，全盘扫描的时间肯定是比直接定位检测要慢很多的。提出的优化思路有如下几点：

1. 有的字符串能直接定位检测的就直接定位扫描，就比如说是检查是否是PE文件，可以直接读取开头的16位，看他是不是4D5A就行。那么此时的速度肯定是会快很多的
2. 尽量减少xor的应用
3. 使用精简的字符串或者效率高的正则表达式。像之前我在对IPV4进行匹配的时候，写的那个正则表达式确实是能匹配出所有的IPV4形式，但是在效率上就会差很多，那么这里就可以考虑进行一个精简，提高效率
4. 找到关键部分，去除冗余。在我之前写的对URL检测的正则表达式中，其实只需要`^((ht|f)tps?)`这一部分也就可以检测了，毕竟现在网页端除了用IP的地址访问，其他通常都是用这部分开头，之后的内容都可以直接忽略不看。

5. 加上对文件大小的限制。这一条在明确知道自己需要检测的文件大小大概范围的时候是可以加上的，这样就能在一定程度上减少检查的数目、提高检查效率，但是这个的使用是有一定限制的，或者说适用的范围相对较小