恶意代码分析与防治技术实验报告

Lab2

学号：2012307　　姓名：聂志强　专业：信息安全

一、 实验环境

1. 已关闭病毒防护的 Windows10
2. VMware+Windows XP （由于 Windows 上使用 Dependency Walker 出现卡顿情况，因此在虚拟机 Windows XP 环境下使用该软件）

二、 实验工具

Yara64.exe、Dependency Walker、Strings、Hex View、IDA Pro、UPX、Pycharm

三、 实验内容

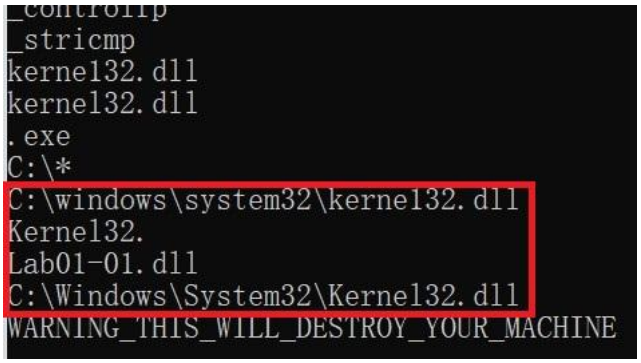
1. 对 Lab1 的样本编写 Yara 检测规则，并进行测试。

总体思想：

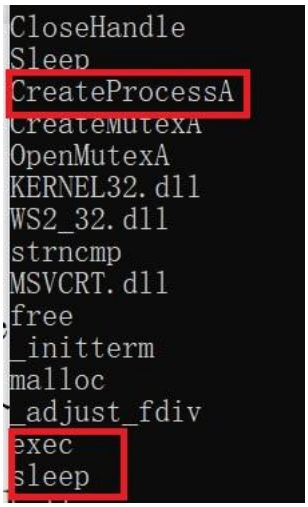
通过 Strings、Dependency Walker、IDA Pro 和 Hex View 四种工具对 Lab1 中每个文件进行分析，找到功能字符串、导入函数等特征，观察 lab1 的 5 个文件大小以及类型（PE 文件），因此均含有规则 filesize<10MB and uint16(0)==0x5A4D and uint16(uint16(0x3C))==0x00004550

具体分析：

（1） 通过 Srings 检查 Lab01-01.exe 时发现，同时观察到 C:\Windows\System32\kernel32.dll 和 C:\Windows\System32\kerne132.dll，文件 kerne132.dll，用数字 1 代替了字母 1，猜测系统文件 kernel32.dll 而自己冒充混淆为 Windows 系统文件，因此可以用 **C:\Windows\System32\kerne132.dll** 和 **kerne132.dll** 作为 lab01-01.exe 恶意代码感染的迹象。同时发现 Lab01-01.dll，猜测该.exe 文件是用来安装与运行.dll 文件的，因此 **Lab01-01.dll** 也可以作为 Yara 规则检测 lab01-01.exe 的匹配特征。

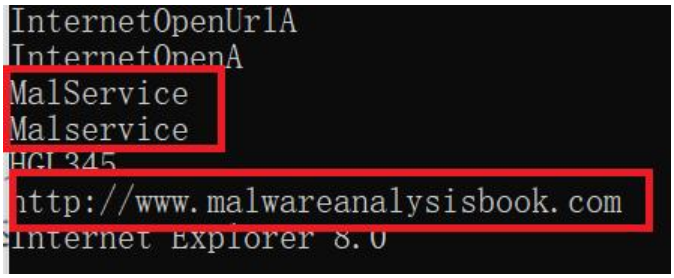


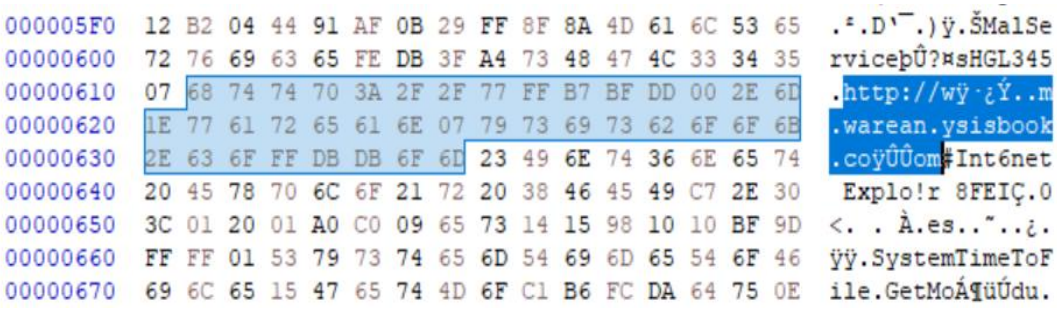
（2） 通过 Srings 和 Dependency Walker 工具检查 Lab01-01.dll 的字符串和导入函数。我们注意到 exec 、sleep 和 CreateProcessA，exec 可能用于通过网络给后门程序传送命令 ，再利用 CreateProcess 函数运行某个程序 ，sleep 可能用于让后门程序进入休眠模式。因此编写 Yara 规则时把同时出现 Sleep + **exec** + CreateProcess 这三个特征字符串猜测为后门程序。



（3） 通过 Strings 和 Dependency Walker 工具检查进行初步时，发现 InternetOpenUrl 、 InternetOpen 、 CreateService 可能存在恶意操作的函数以及字符串 MalService 、sHGL345，想起 Lab1 实验中 Lab01-02 是进行了加壳的程序，那么猜想很多信息被隐藏，所以我进而 Lab01-02 进行脱壳处理，发现 www.malwareanalysisbook.com 字符串，yara 规则是针对静态分析的，不支持动态调试过程中进行匹配， 对于加壳后的恶意程序，会出现字符串特征被加壳程序处理而无法直接匹配的现象，所以使用 Hex View 视图，在十六进制机器码以及对应的数据下进行观察可以看到一些破碎的特征信息，查找到对应

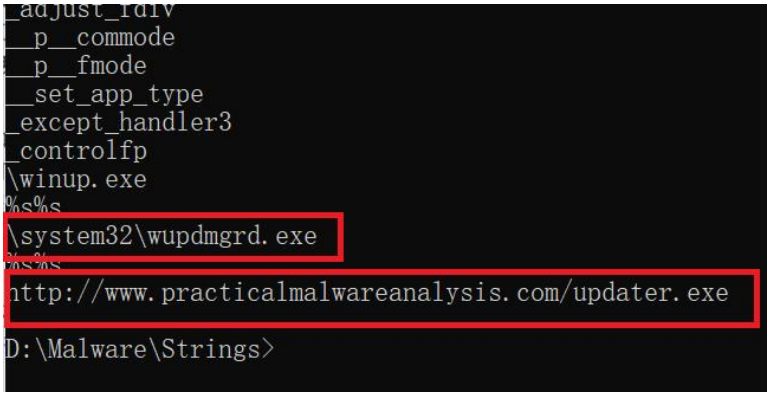
的十六进制机器码，综合以上信息进行编写 Yara 规则。





（4） 进行 lab01-03.exe 分析前，想到上次 lab1 实验时对于该文件脱壳不太会，因此没办法考虑脱壳后分析，仅能对脱壳前可疑字符串等信息编写 Yara 规则。

（5） 通过 Strings 和 Dependency Walker 工具对 Lab01-04.exe 进行分析时，出现字符串 \system32\wupdmgrd.exe（Windows 升级管理器），结合导入函数 GetWindowDirectory 的调用，表明恶意代码在 C：Windows\system32\wupdmgrd.exe 位置创建或修改一个文件。 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。综合上述 \system32\wupdmgrd.exe 和 www.malwareanalysisbook.com/updater.exe 字符串特征编写 Yara 规则。



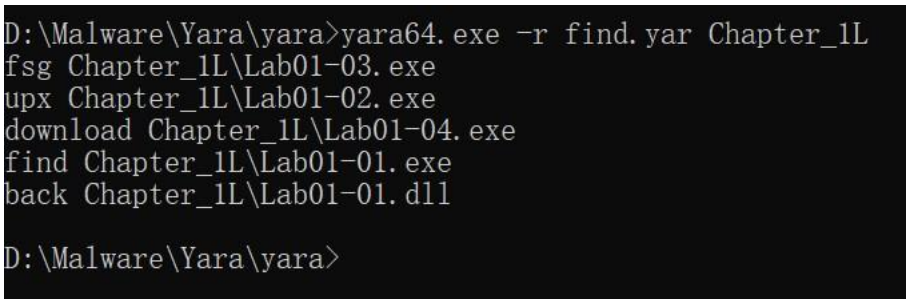
Yara 规则：

```
rule find
{
strings:
        $string1 = "Lab01-01.dll"
        $string2 = "kerne132.dll"
        $string3 = "C:\\windows\\system32\\kerne132.dll"
condition:
        filesize<10MB and uint16(0)==0x5A4D and uint16(uint16(0x3C))==0x00004550 and 1 of them
}

rule back
{
strings:
        $string1 = "sleep"
        $string2 = "exec"
        $string3 = "CreateProcessA"
condition:
        filesize<10MB and uint16(0)==0x5A4D and uint16(uint16(0x3C))==0x00004550 and all of them
}

rule upx
{
strings:
        $string1 = "MalService"
        $string2="HGL345"
        $num = {68 74 74 70 3A 2F 2F 77 FF B7 BF DD 00 2E 6D 1E 77 61 72 65 61 6E 07 79 73 69 73
62 6F 6F 6B 2E 63 6F FF DB DB 6F 6D}
condition:
        filesize<10MB and uint16(0)==0x5A4D and uint16(uint16(0x3C))==0x00004550 and $string1
and $string2    and $num
}

rule fsg
{
strings:
        $string0 = "}OLEAUTLA"
        $string1 = "ole32.vd"
        $string2 = "_getmas"
condition:
        all of them
}
```

```
rule download
{
strings:
        $string1 = "\\system32\\wupdmgrd.exe"
        $string2 = "http://www.practicalmalwareanalysis.com/updater.exe"
condition:
        uint16(0) == 0x5a4d and filesize < 100KB and all of them
}
```

运行结果：

在 Chapter_1L 文件夹中成功识别 5 个病毒程序

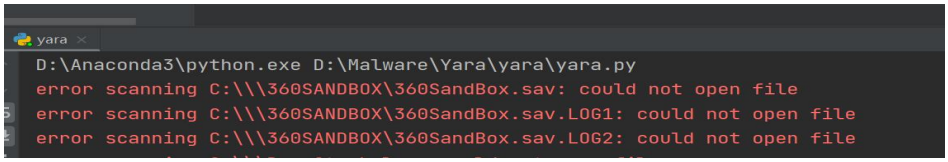


2. **使用自己编写的规则对自己电脑的 C 盘进行 Yara 引擎的扫描，记录扫描所用时间。**

（1） 为了方便时间的统计，通过 pycharm 平台编写了一个 python 的脚本执行扫描和时间统计，代码如下：

```
1.    import json
2.    import os, time
3.
4.    begin_time = time.time()
5.    os.system(r"yara64.exe -r find.yar C:\\")
6.    end_time = time.time()
7.
8.    print(end_time - begin_time)
```

可以看到其中有一部分扫描的时候出现“could not open file”报错，应该是因为权限不足的问题。



（2） 成功找到 C 盘中存放的 Chapter_1L 中的 5 个程序，且仅有少量 C 盘中程序的误判为恶意代码。

```
find C:\\\Chapter_1L\Lab01-01.exe
upx C:\\\Chapter_1L\Lab01-02.exe
back C:\\\Chapter_1L\Lab01-01.dll
download C:\\\Chapter_1L\Lab01-04.exe
fsg C:\\\Chapter_1L\Lab01-03.exe
```

（3） 扫描时间如下图所示，约为 192s

```
192.15387201309204

进程已结束,退出代码0
```

3. 讨论哪些 Yara 条件执行效率高，哪些 Yara 条件执行效率低。如何改进那些执行效率低的 Yara 条件？

（1） 结合理论课所讲知识和实验中亲自对比，当检查特征字符串时如果使用全局搜索效率会很低，当指定地址或地址范围时 Yara 条件执行效率会变高。例如根据 IDA Pro 提供的信息对字符串在文件中的偏移值或虚拟地址的偏移值进行计算，使用“at”指定它的特定地址，这样避免了查找整个程序，只需要找到特定地址去匹配。

（2） 对于 16 进制字符串匹配，??的效率比[...]效率更高一点。

四、 实验心得

1. 通过对全局搜索、范围搜索和特定地址搜索的不同 Yara 规则比较，验证了对条件执行效率的高低。

2. 为了找到十六进制地址，探索使用 IDA Pro 对脱壳程序进行分析并成功实现