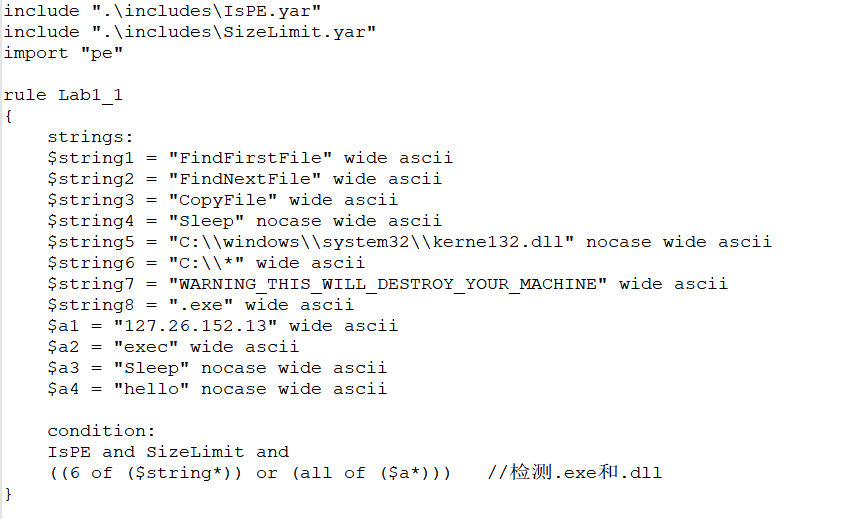
**Lab 1.1Yara规则编写**

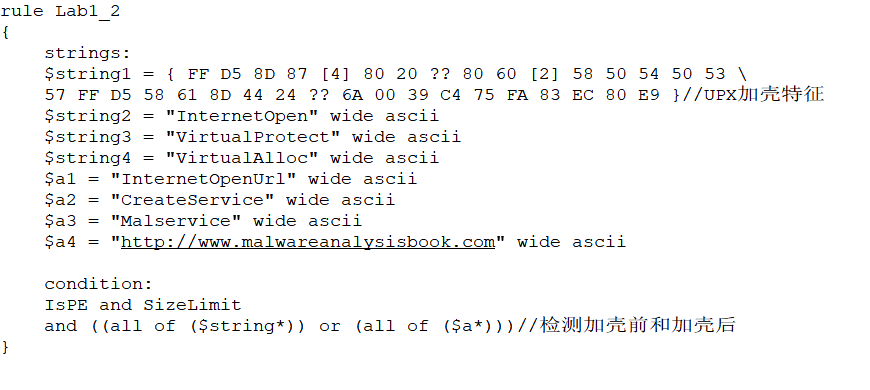
**许健 2013018**

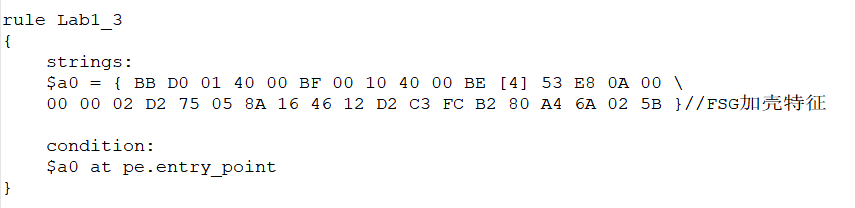
1. 对Lab1的样本编写Yara规则。

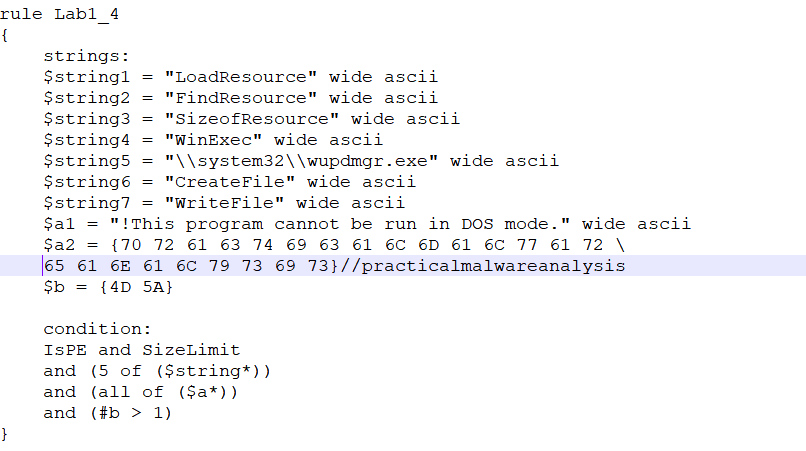
对于Lab1中的四个恶意代码分别编写了检测规则，其中对于Lab1\_3的加壳程序并没有找到什么可用特征，只检测了FSG壳的特征。

对于ISPE和SizeLimit则放在了其他文件里，private不输出，其中SizeLimit定义global先检测。寻找的特征主要是恶意程序中可疑的字符串或者是一段字节序列。对于更加有效的病毒特征目前还不是特别清楚如何提取，比如在某一段内存区域会有什么字节序列等。



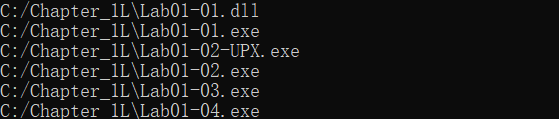


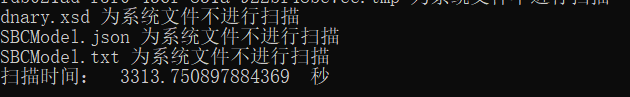




1. 使用自己编写的规则对自己电脑的C盘进行Yara引擎的扫描，记录扫描所用时间。

编写Python脚本运行yara规则，import time模块统计时间，输出结果如下，可以看到恶意的代码文件被检测出来，但是C盘占用300G且包含许多动态链接库、可执行文件(PE文件)，扫描很慢，如果使用上文的yara规则，执行效率相当低下。





1. 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？

全局匹配的Yara条件执行效率较低，比如字符串查找，需要遍历文件，可以多用限定范围的yara规则，多用in和at，如在程序的入口点附近查找特定的字节序列，程序执行效率会高很多；其次是减少与恶意程序关联不大的字符串，只选取两三个代表性的字符串即可，在执行效率和准确性之间做一个折中。

减少扫描文件的数量也是有必要的，恶意程序通常不会太大，可以使用SizeLimit限制文件的大小，但是带来的问题是黑客可能将多个恶意文件合并在一起以躲过程序的检测，实际使用要慎重权衡。