

**恶意代码课程实验报告**

**实验六：Windows恶意代码**

****

学 院 网络安全学院

专 业 信息安全

学 号 2110688

姓 名 史文天

班 级 1063

1. **实验目的**

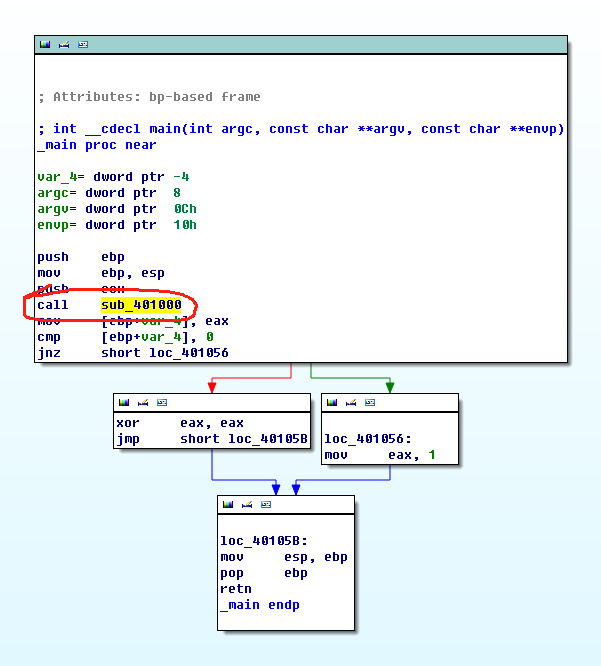
**完成课本Lab6的实验内容，编写Yara规则，并尝试IDA Python的自动化分析。**

1. **实验原理**

**配置XP虚拟机环境，安装动态、静态分析工具，使用静态和动态分析工具。**

1. **实验过程**
2. **Lab06-01**
3. **由main函数调用的唯一子过程中发现的主要代码结构是什么？**

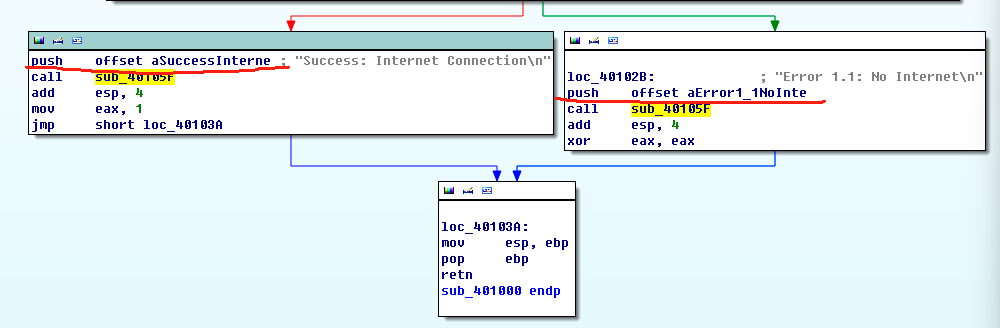
**用IDA打开该文件，使用图视图，发现调用的唯一子过程是sub\_401000。**



**跳转到该位置，看到子过程中主要的代码结构是if结构。**

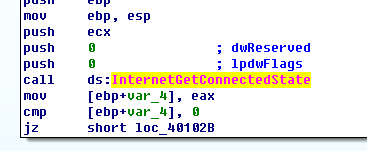
1. **位于0x40105F的子过程是什么？**

**传入该子过程的参数是两个字符串，所以该子过程的功能应该是打印字符串。**

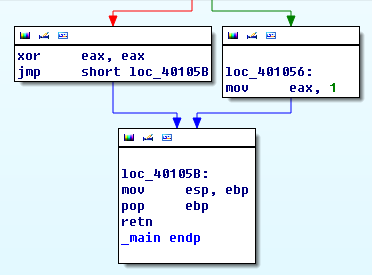


1. **这个程序的目的是什么？**

**子过程中调用了InternetGetConnectedState函数，该函数用于获得本地系统的网络连接状态。**

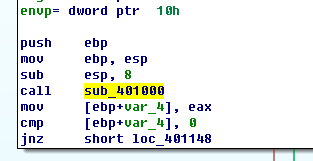


**观察分支可发现该函数返回值与子过程相同。而子过程返回1时会输出"Success: Internet Connection\n"，返回0时会输出"Error 1.1: No Internet\n"。**

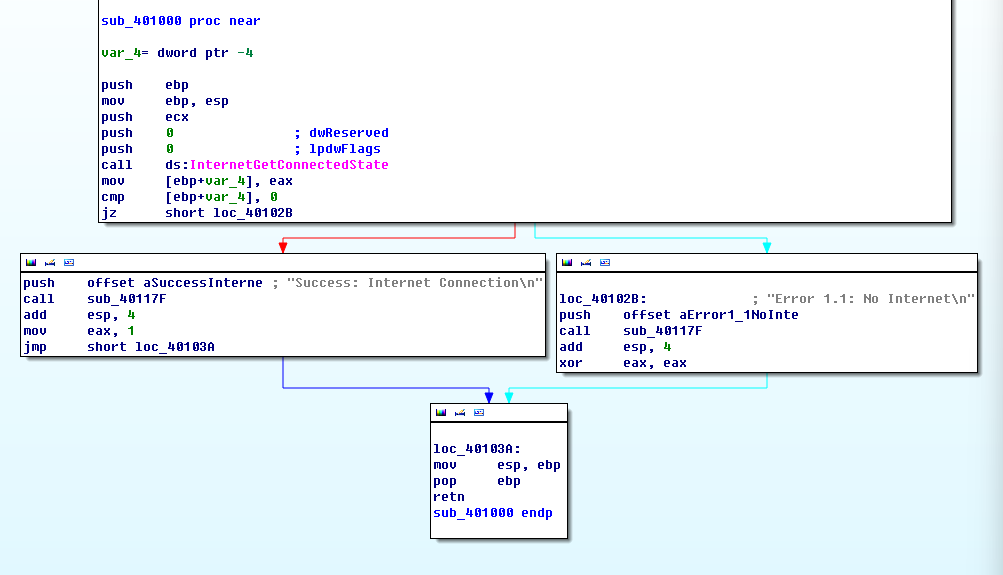


**综上，当本地有网络连接时，该函数会输出"Success: Internet Connection\n"并返回1，否则会输出"Error 1.1: No Internet\n"并返回0.**

1. **Lab06-02**
2. **main函数调用的第一个子过程执行了什么操作？位于0x40105F的子过程是什么?**

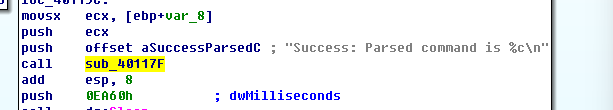


**跳转到该位置发现与6-1相同，是检查本地网络链接。**



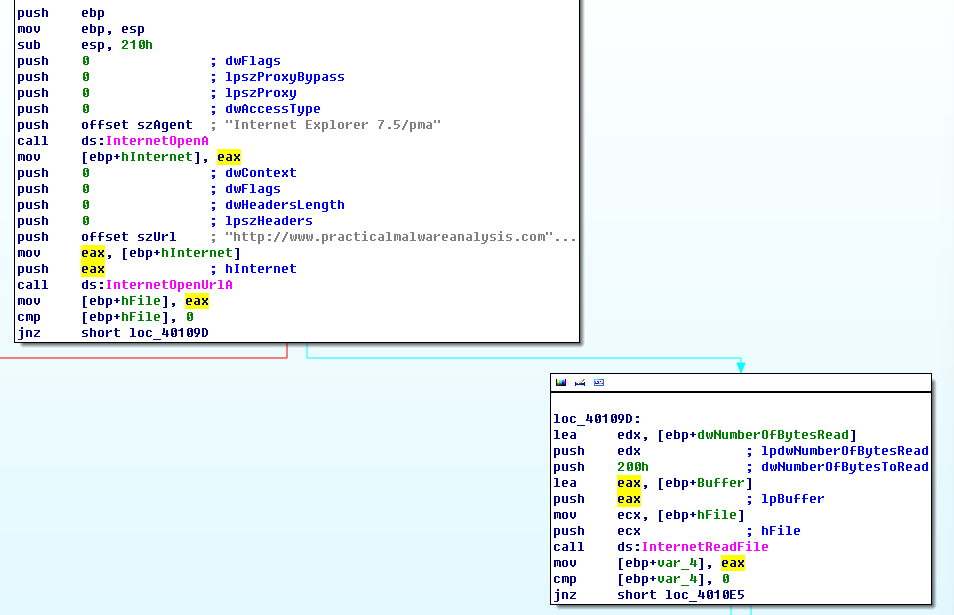
1. **位于0x40117F的子过程是什么？**

**与6-1同理是输出字符串。**

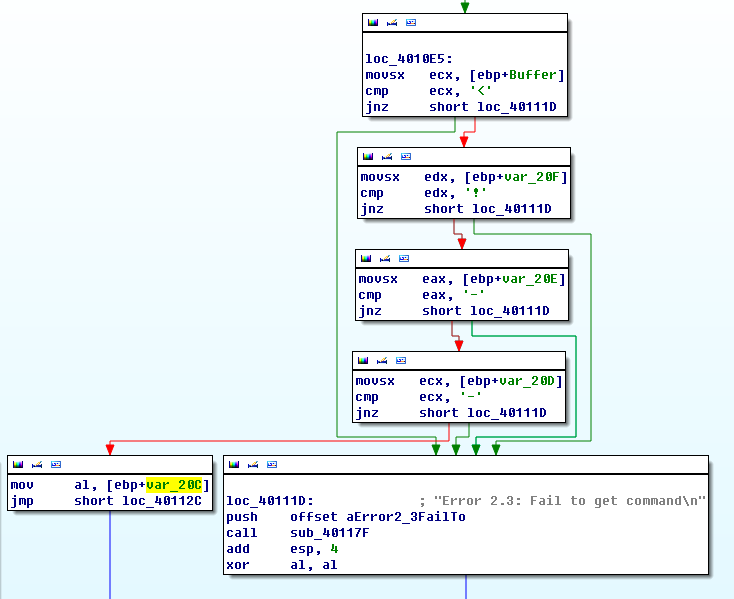


1. **被main函数调用的第二个子过程做了什么？**

**首先该进程以"Internet Explorer 7.5/pma"参数调用InternetOpenA函数，又以“http://www.practicalmalwareanalysis.com/cc.htm”为参数调用InternetOpenUrlA函数，之后将结果“hFile”和数组“Buffer”作为参数传给函数InternetReadFile，从该url读取数据并以字符形式存储在Buffer中。**

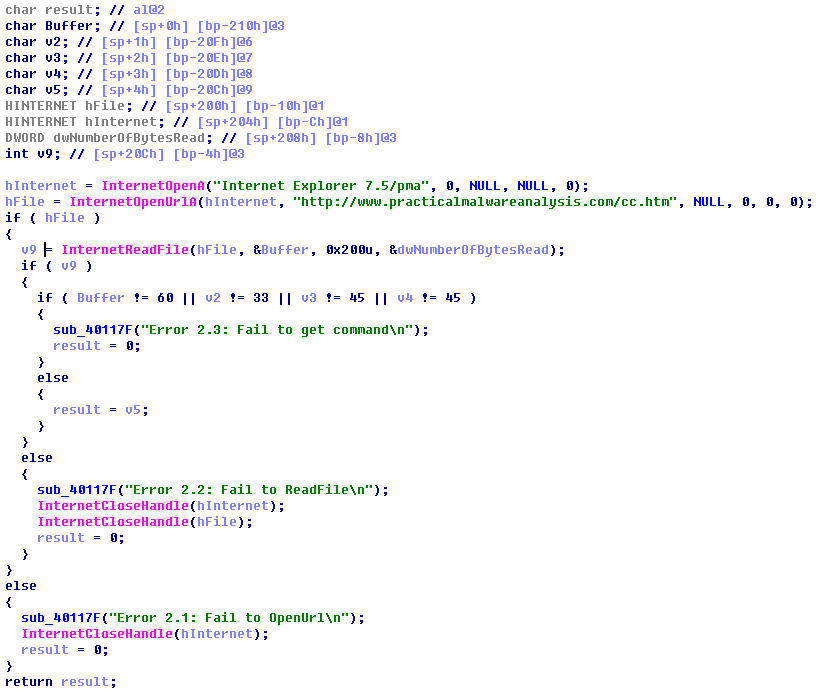


**之后其通过四个cmp判断Buffer的开头是否为“** <!—**”的命令开头。若是则将随后的一个字符存在al中。**



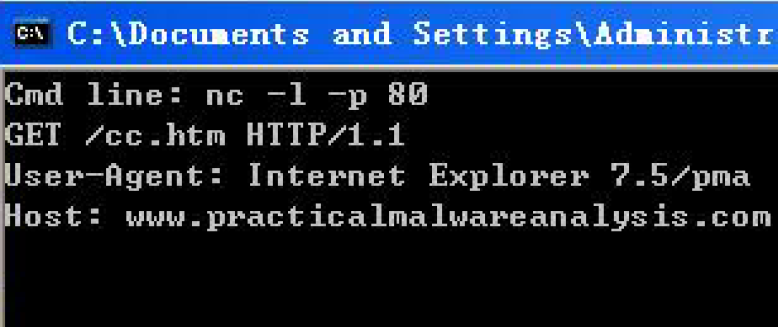
1. **在这个子过程中使用了什么类型的代码结构？**

**用伪代码窗口分析，看到使用了字符数组和if结构。**



1. **在这程序中有任何基于网络的指示吗？**

**使用netcat等工具监测，很容易发现该文件访问了上述url。**

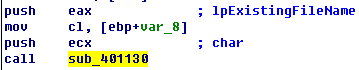
****

1. **这个恶意代码的目的是什么？**

**恶意代码首先判断是否存在一个可用的Internet连接，如果不存在就终止运行；如果存在，则使用一个独特的用户代理尝试下载一个网页。该网页包含了一段由“<!--”开始的HTML注释，程序解析之后的那个字符，进行逐字比对，并打印“Success: Parsed command is %c\n”，其中%c就是从该字符。如果解析成功，程序会休眠60秒，然后终止运行。**

1. **Lab06-03**
2. **比较在main函数与实验6-2 的main函数的调用。从main中调用的新的函数是什么？**

**是sub\_401130。**



1. **这个新的函数使用的参数是什么？**

**传入的第一个参数是char类型，即此前读出的HTML字符。第二个参数是指向文件名字符串的指针（实际上是标准main函数的argv[0]，即该程序自己的文件名）。**

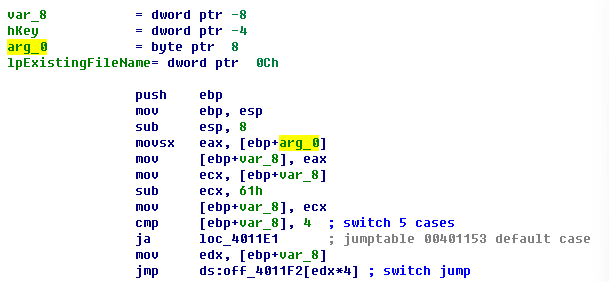
1. **这个函数包含的主要代码结构是什么？**

**用伪代码窗口打开，看到主要结构是switch结构。**



1. **这个函数能够做什么？**

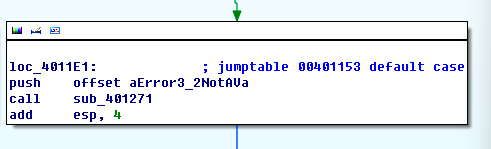
**变量arg\_0是IDA自动生成的标签，表示第一个参数（最后一个压入栈中的参数）,将arg\_0的值赋给var\_8,将var\_8自减61h（对应ASCII 字符‘a’），若该字符减’a’大于4（非‘a’、‘b’、‘c’、‘d’、‘e’），则跳到loc\_4011E1，否则，将该值赋给edx，进入switch语句。**



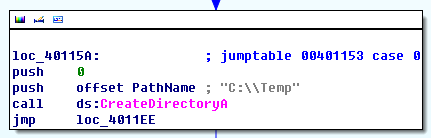
**Switch结构通过将arg\_0与一张跳转表off\_4011F2对比选择跳转位置 (包含‘a’~e’的分支)，加上loc\_4011E1（default 分支），共有6个分支。**



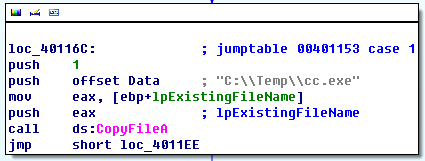
**若该字符非‘a’~e’，则输出'Error 3.2: Not a valid command provided'。**



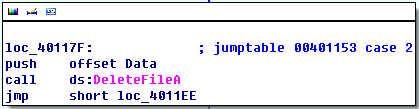
**若为‘a’，则调用CreateDirectory创建一个文件夹“C:\Temp”。**



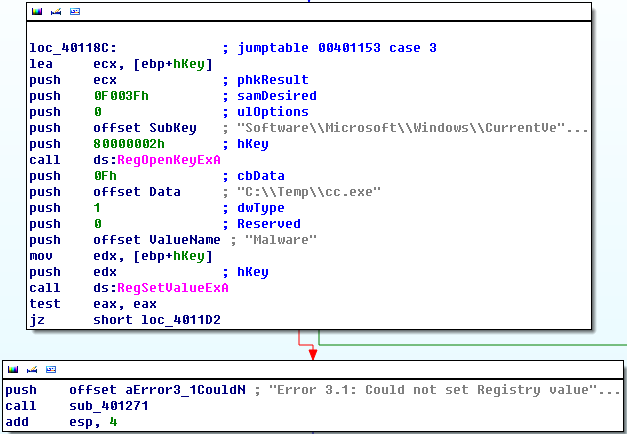
**若为‘b’，则调用CopyFile复制文件： 源文件是lpExistingFileName， 前文提过是argv[0]， 也即该程序自己的文件名“Lab06-03.exe”；目标文件是“C:\Temp\cc.exe”。即该分支将Lab06-03.exe复制到C:\Temp\cc.exe。**



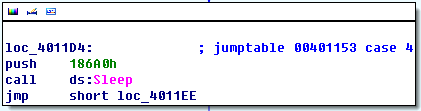
**若为‘c’，则会调用C:\Temp\cc.exe删除文件C:\Temp\cc.exe。**



**若为‘d’，则会先调用RegOpenKeyEx打开注册表键“Software\Microsoft\Windows\CurrentVersion\Run”，然后再在该键下创建一个新的键“...\Malware”，其值为“C:\Temp\cc.exe”。这样系统启动时，如果C:\Temp\cc.exe存在，则也会跟随系统启动，自动运行。**



**若为‘e’，则调用Sleep休眠186A0h=100000毫秒。**



1. **这个恶意代码中有什么本地特征吗？**

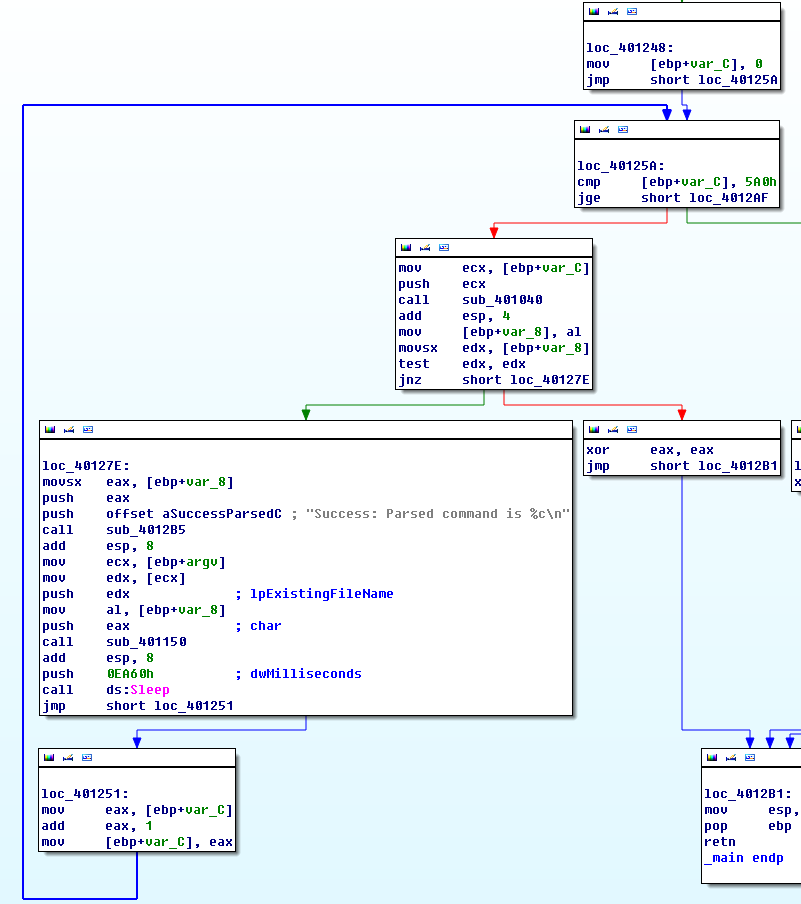
**注册表键Software\Microsoft\Windows\CurrentVersion\Run\Malware和本地文件C:\Temp\cc.exe。**

1. **这个恶意代码的目的是什么?**

**该程序先检查是否存在有效的Internet连接。如果找不到，程序直接终止。否则，该程序会尝试下载一个网页，该网页包含了一段以“<!--” 开头的HTML注释。该注释的第一个字符被用于switch语句来决定程序在本地系统运行的下一步行为，包括是否删除一个文件、创建一个目录、设置一个注册表run键、复制一个文件或者休眠100秒。**

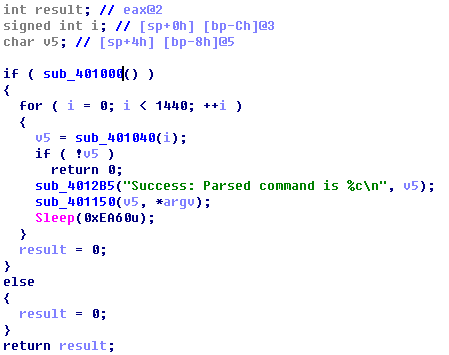
1. **Lab03-04**
2. **在实验6-3 和6-4 的 main 函数中的调用之间的区别是什么？**

**Main函数中有一个for循环结构，循环次数由var\_C记录，当var\_C < 1440时继续运行。循环体中调用了sub\_401040。**



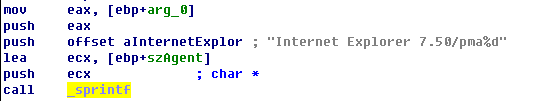
1. **什么新的代码结构已经被添加到 main 中？**

**用伪代码窗口打开，看到添加了for循环结构。**



1. **这个实验的解析HTML的函数和前面实验中的那些有什么区别？**

**arg\_0是从main函数中传入的计数器（var\_C）。arg\_0与一个格式化字符串及一个目标地址一起被压入栈，然后可以看到调用了sprintf，后者创建一个字符串，并将其存储在局部变量szAgent中， szAgent被传给了InternetOpenA，也就是说，每次计数器递增了，User-Agent也会随之改变。**



1. **这个程序会运行多久？（假设它已经连接到互联网。)**

**每次循环由于sleep指令，至少要运行1分钟，循环1440次，则至少运行24h。**

1. **在这个恶意代码中有什么新的基于网络的迹象吗？**

**User-Agent会不断改变。**

1. **这个恶意代码的目的是什么？**

**该恶意代码会检查是否存在可用的Internet连接，如果连接不存在，程序终止运行，否则，程序使用一个独特的User-Agent下载一个网页，这个User-Agent中包含了一个循环结构的计数器，该计数器中是程序已经运行的时间，下载的网页里包含HTML注释，会被读到一个字符数组里，并与”<!--“进行比较，然后从注释中抽取下一个字符，用于一个switch结构来决定接下来在本地系统的行为，这些行为是已经硬编码的，包括删除一个文件、创建一个文件夹、设置一个注册表run键、复制一个文件以及休眠60s。该程序会运行1440分钟后终止。**

1. **Yara规则**

**根据以上分析结果，得到Yara规则如下。**

rule lab0601

{

strings:

$string1 = "Error 1.1: No Internet"

$string2 = "Success: Internet Connection"

$string3 = "InternetGetConnectedState"

condition:

filesize < 100KB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) == 0x00004550 and all of them

}

rule lab0602

{

strings:

$string1 = "http://www.practicalmalwareanalysis.com/cc.htm"

$string2 = "Error 2.3: Fail to get command"

$string3 = "Internet Explorer 7.5/pma"

condition:

filesize < 100KB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) == 0x00004550 and all of them

}

rule lab0603

{

strings:

$string1 = "Software\\Microsoft\\Windows\\CurrentVersion\\Run"

$string2 = "C:\\Temp\\cc.exe"

$string3 = "C:\\Temp"

condition:

filesize < 100KB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) == 0x00004550 and all of them

}

rule lab0604

{

strings:

$string1 = "Success: Parsed command is %c"

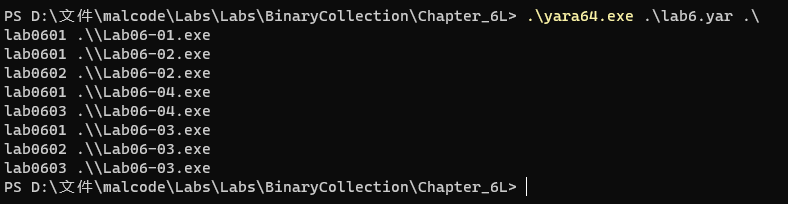
$string2 = "DDDDDDDDDDDDDD"

condition:

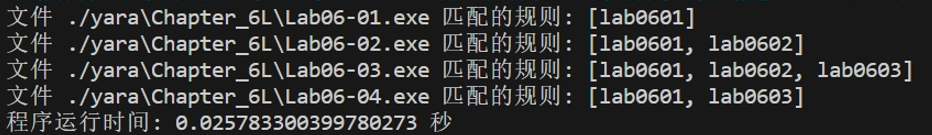
filesize < 100KB and uint16(0) == 0x5A4D and uint16(uint16(0x3C)) == 0x00004550 and all of them

}

**运行结果如下，可看到成功匹配所有文件。**



**在存放所有样例的文件夹下测试其检测效率，结果如下。**



1. **IDA Python脚本编写**

**遍历所有函数，排除库函数或简单跳转函数，当反汇编的助记符为call或者jmp且操作数为寄存器类型时，输出该行反汇编指令。**

import idautils

for func in idautils.Functions():

flags = idc.GetFunctionFlags(func)

if flags & FUNC\_LIB or flags & FUNC\_THUNK:

continue

dism\_addr = list(idautils.FuncItems(func))

for line in dism\_addr:

m = idc.GetMnem(line)

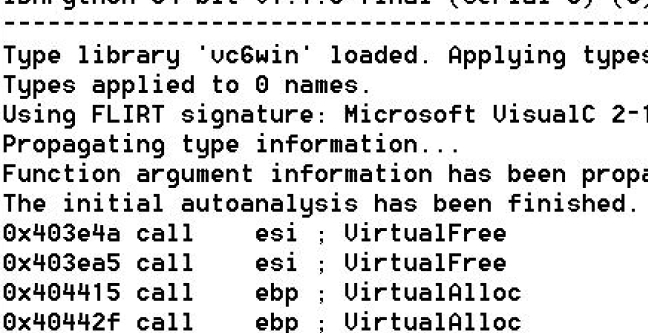
if m == 'call' or m == 'jmp':

op = idc.GetOpType(line,0)

if op == o\_reg:

print '0x%x %s' % (line,idc.GetDisasm(line))

**结果如下**

****

1. **实验结论及心得体会**
2. **对程序动态分析有了进一步了解。**
3. **了解了更多Windows恶意代码的形式和特征。**