

**恶意代码课程实验报告**

**实验七：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**

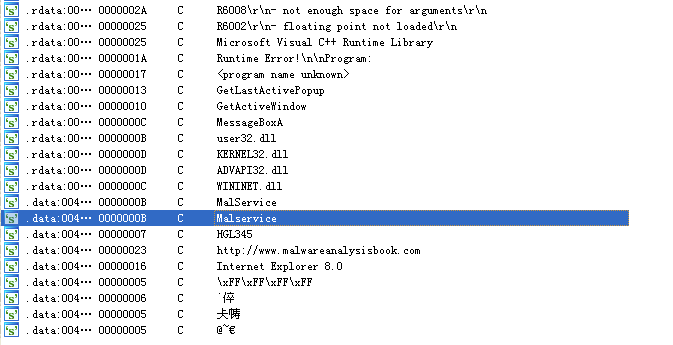
**完成课本Lab7的实验内容，编写Yara规则，并尝试IDA Python的自动化分析。**

1. **实验原理**

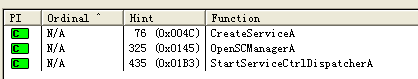
**配置XP虚拟机环境，安装动态、静态分析工具，使用静态和动态分析工具。**

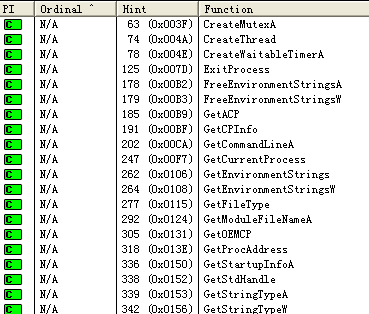
1. **实验过程**
2. **Lab07-01**

**查看字符串，看到Malservice，推测该程序创建了一个系统服务，网址信息**[**http://www.malwareanalysisbook.com**](http://www.malwareanalysisbook.com)**，用户代理信息Internet Explorer 8.0 。**



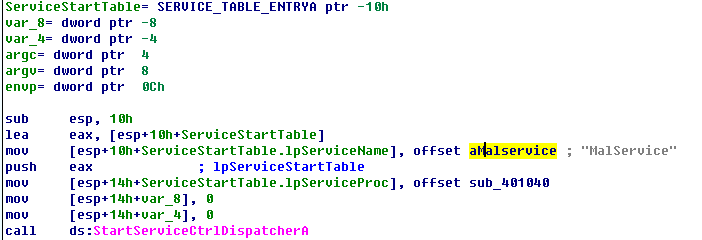
**查看导入函数，比较重要的有CreateServiceA，CreateThread,InternetOpenA等。**







**StartServiceCtrlDispatcherA函数被系统用于实现服务，且一般立即被调用。该函数制定了服务控制管理器会调用的函数。它所制定的的是sub\_401040。**

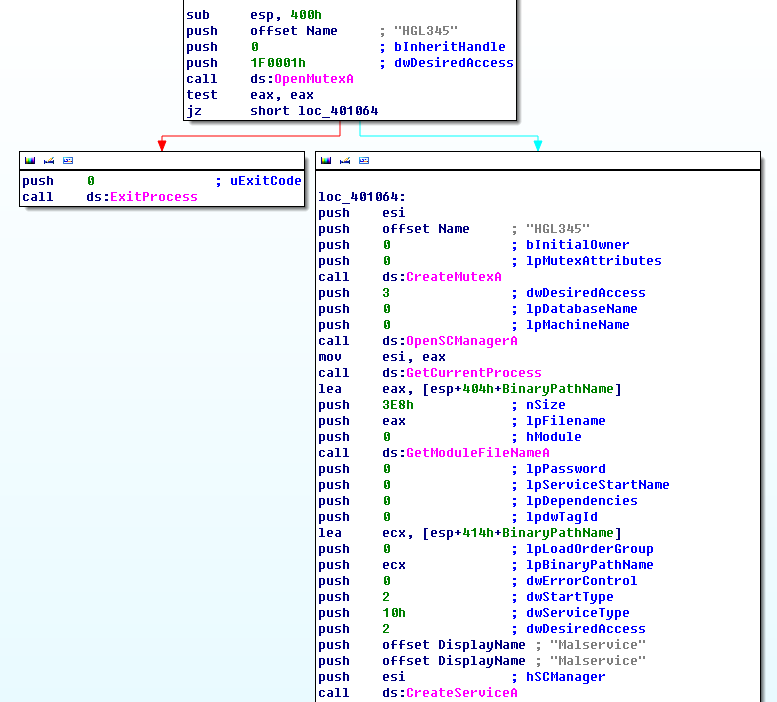


**跳转到sub\_401040，第一个函数是OpenMutexA，它尝试获取一个名为"HGL345“的互斥量句柄。如果调用成功，程序就会退出。若失败，则创建名为”HGL345"的互斥量，两处组合调用，用于保证同一时间这个程序只有一份实例在运行。因为如果有一个实例在运行了，则OpenMutexA第一次调用成功，程序就会退出。**

**OpenSCManagerA打开服务控制管理器句柄，以便该程序可以添加或修改服务。**

**GetModuleFileNameA返回当前可执行程序或一个被加载DLL的全路径名。**

**返回的全路径名被CreateServiceA用于创建一个新的服务。**

**结构体834h 表示10进制2100，表示2100年1月1日午夜。**

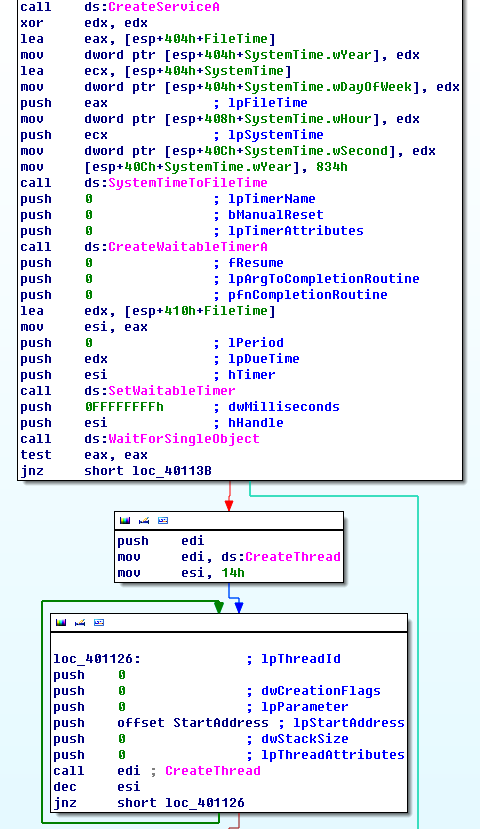
**SystemTimeToFileTime用于不同时间格式的转换。**

**SetWaitableTimer函数的lpDueTime参数，来自于刚才时间转换函数返回的FileTime。**

**随后进入WaitForSingleObject等待，直到2100年1月1日午夜执行。**

**在1处ESI设置为计数器0x14（十进制20），循环的末位，ESI在2处递减，在3处到达0时，循环退出。4处的CreateThread函数的参数中lpStartAddress可以告知当前线程的起始地址。**

**循环末尾的jmp指令是一个无条件跳转，意味着代码将永远不会停止；调用InternetOpenUrlA，并且一直下载该网址的主页。由于前面ESI被设置为20，因此会有20个线程一直调用InternetOpenUrlA函数。**



1. **当计算机重启之后，这个程序如何保证它继续运行（达到持久化驻留）？**

**创建服务 MalService，在后台随系统启动运行。**

1. **为什么这个程序会使用一个互斥量？**

**保证同一时间这个程序只有一份实例在运行。**

1. **可以用来检测这个程序的基于主机特征是什么？**

**主机上的服务Malservice和互斥量。**

1. **检测这个恶意代码的基于网络特征是什么？**

**网址信息<http://www.malwareanalysisbook.com>，用户代理信息Internet Explorer 8.0 。**

1. **这个程序的目的是什么？**

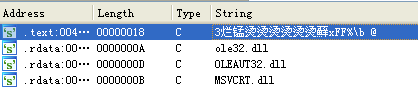
**该恶意代码目的是将自己在多台机器上安装成一个服务，进而启动DDOS攻击。如果所有的被感染机器在同一时间访问该服务器，会导致该服务器过载并无法访问该站点，导致拒绝服务攻击。**

1. **这个程序什么时候完成执行？**

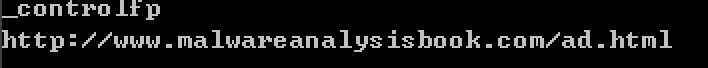
**循环末尾的jmp指令是一个无条件跳转，意味着代码将永远不会停止**

1. **Lab07-02**

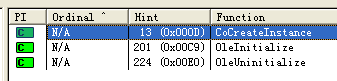
**查看字符串时看到乱码。**



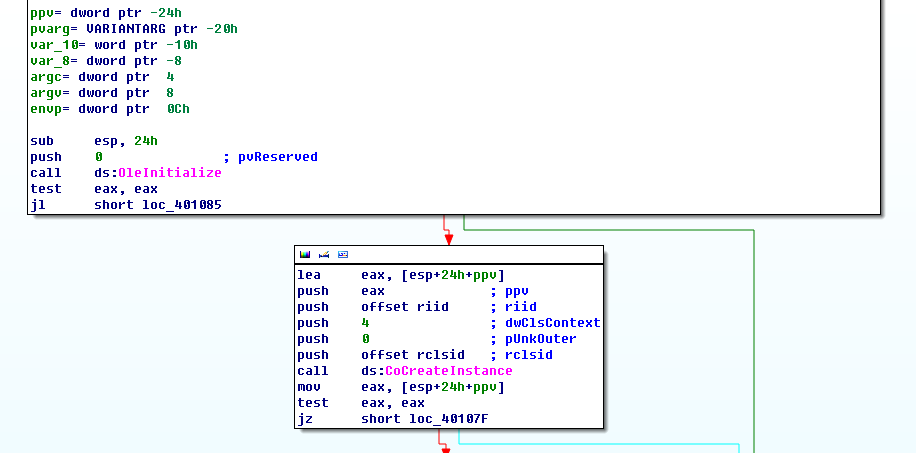
**因为是Unicode所以乱码，实际为如下网站。**



**查看导入函数。**



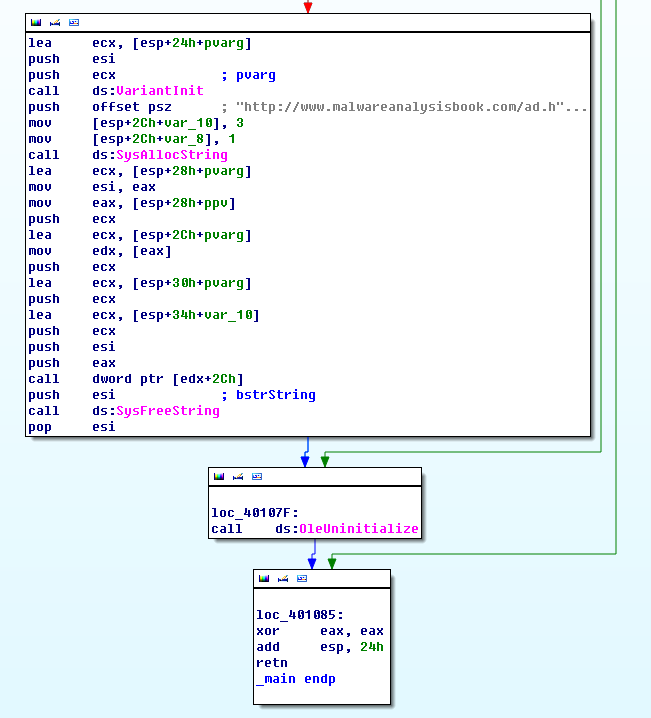
**该恶意代码先初始化COM，调用OleInitialize函数和CoCreateInstance获得一个COM对象。返回的COM对象被保存为ppv。riid和rclsid分别表示接口标识符（IID)和类标识符（CLSID）。**



**riid为0D30C1661-0CDAF-11D0-8A3E-00C04F0C90E26E对应IWebBrowser2。**

**rclsid为0002DF01-0000-0000-C000-000000000046对应Internet Explorer。**

**返回的COM对象在1处被后面的一些函数访问。紧跟这条指令，在2处，EAX被解引用并且指向这个COM对象的基址。在3处，这个对象中偏移0x2C是Navigate函数。该函数被调用后，Internet Explorer将导航至http://www.malwareanalysisbook.com/ad.html。该函数执行一些清理函数，但是没有持久化驻留，仅简单显示一个广告页面。**



1. **这个程序如何完成持久化驻留？**

**该程序没有完成持久化驻留。运行一次后退出。**

1. **这个程序的目的是什么？**

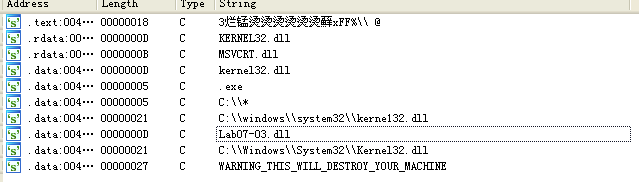
**给用户显示一个广告网页http://www.malwareanalysisbook.com/ad.html。**

1. **这个程序什么时候完成执行？**

**显示广告页后立即完成执行。**

1. **Lab07-03**

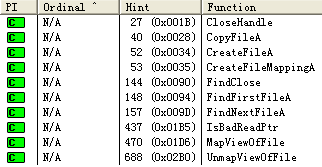
**查看字符串。看到一些动态链接库相关的信息和一条警告。**



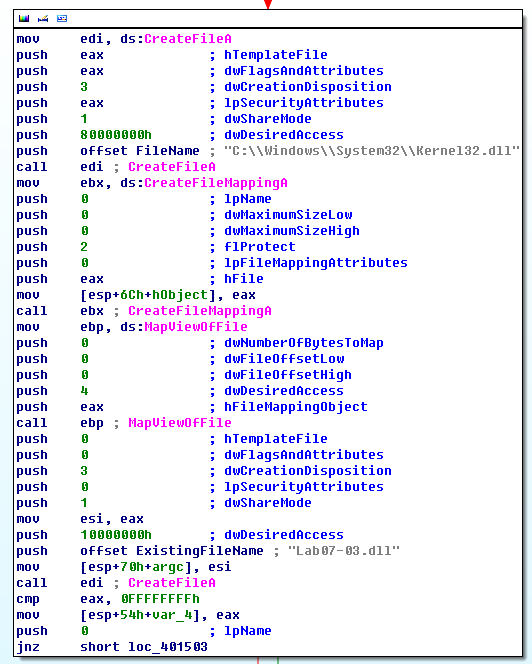
**127.26.15.13，恶意代码可能访问该IP地址。**

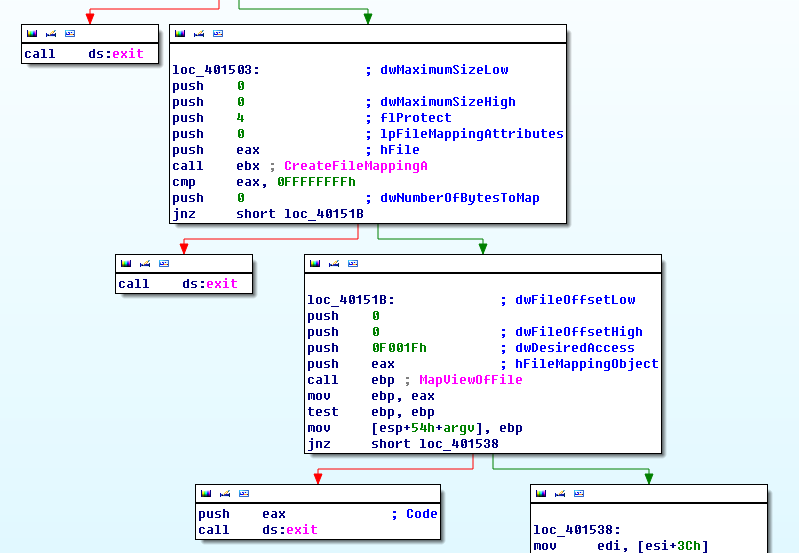


**查看导入函数，看到许多对文件进行操作的函数。CreateFileA 、 CreateFileMappingA以及MapViewOfFile表明该程序可能打开一个文件，并将其映射到内存中。这个 FindFirstFileA 和FindNextFileA函数组合，可能会搜索文件，并使用CopyFileA 对找到的文件进行复制操作。该程序没有导入Lab 07-03.dll（DLL中的函数）、LoadLibrary或者GetProcAddress，暗示它可能没有在运行时加载那个DLL。**

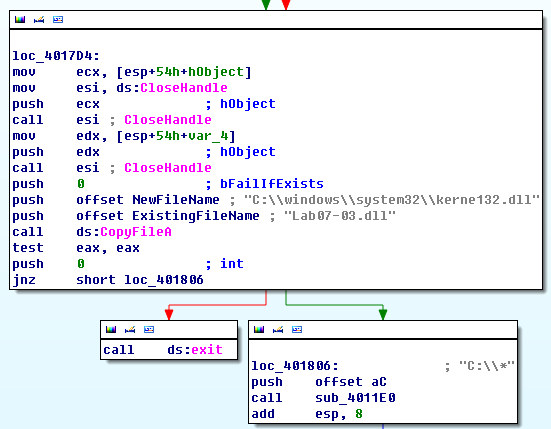


**CreateFileA、CreateFileMappingA 以及MapViewOfFile表明该程序打开文件，并将其映射到内存中。**

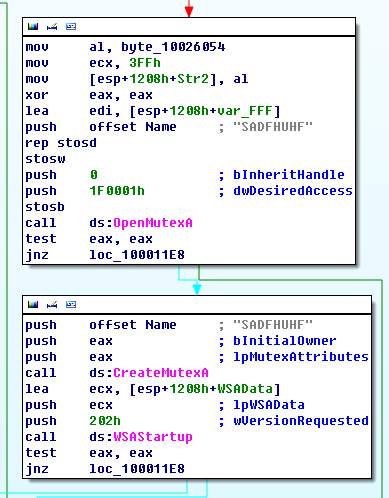




**运行Lab07-03.exe程序后Lab07-03.dll名称修改为kerne132.dll，并且有导出函数，该导出函数是kernel32.dll中的导出函数，并且作了重定向，所有功能还是kernel32.dll中的功能。**

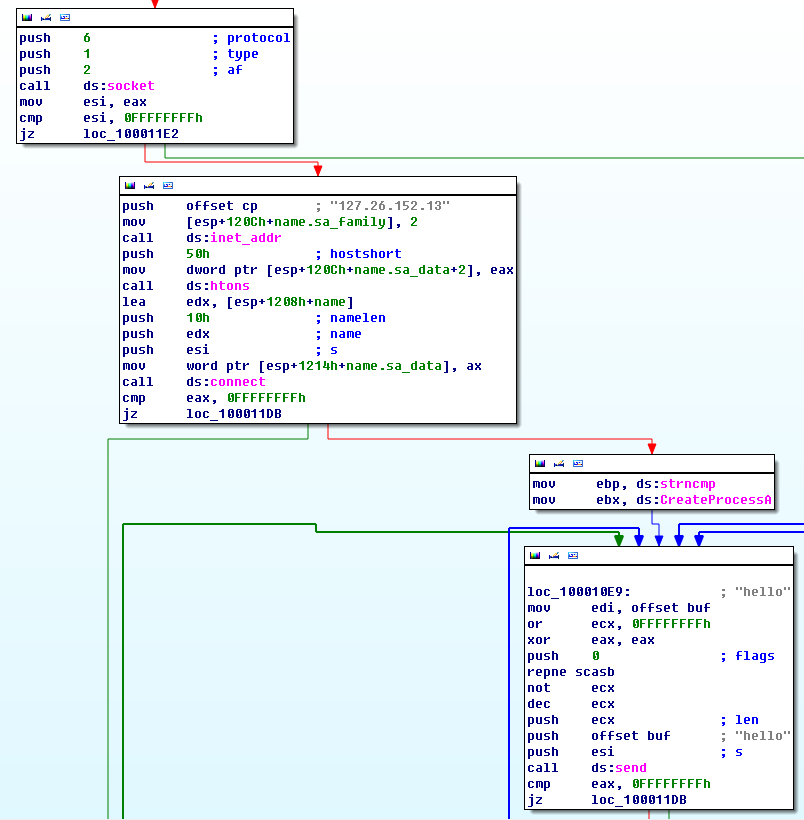


**再查看DLL文件，看到首先使用了OpenMutexA、CreateMutexA 创建互斥量。**

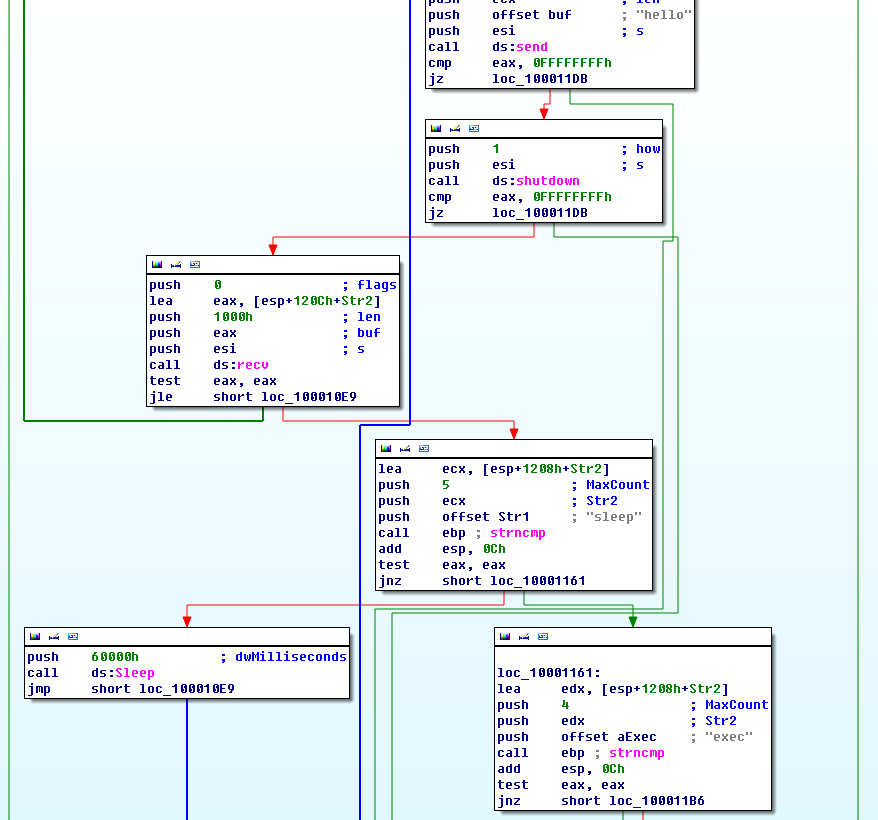


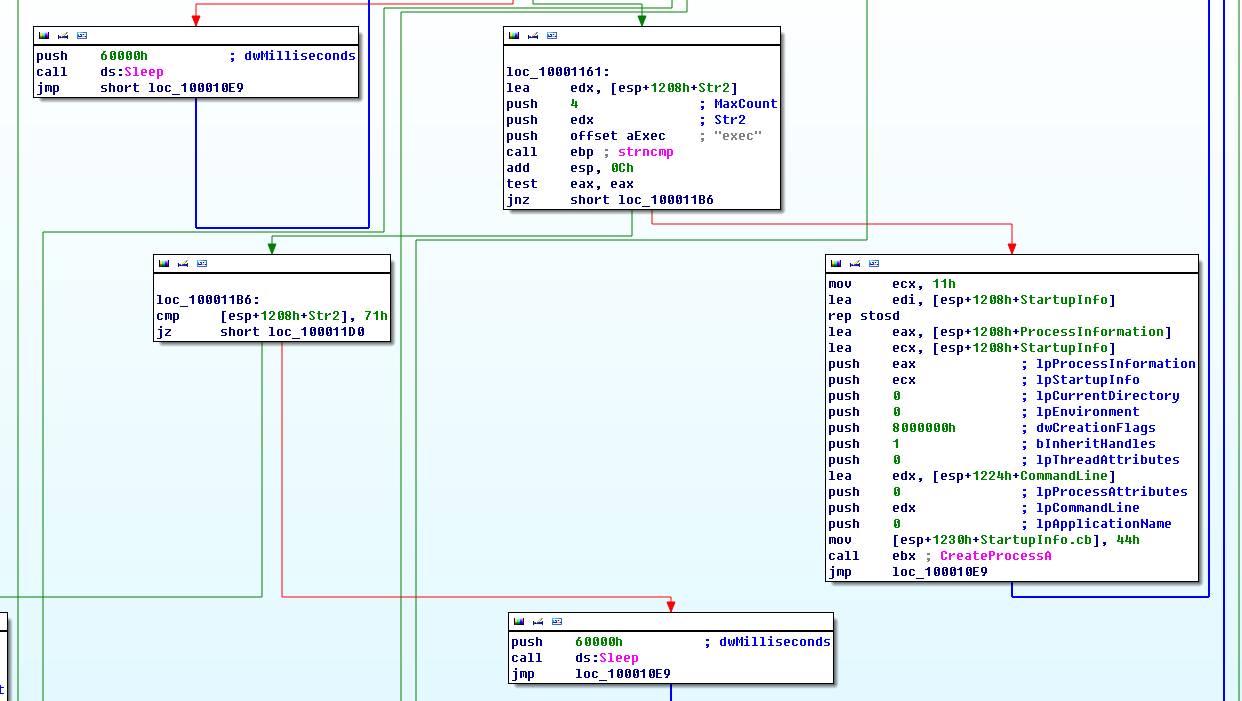
**随后创建了远程socket，建立连接，使用了固定ip地址127.26.152.13。端口0x50，为80端口，常用于Web流量。**

**实行的操作是使用send函数发送“hello”。**

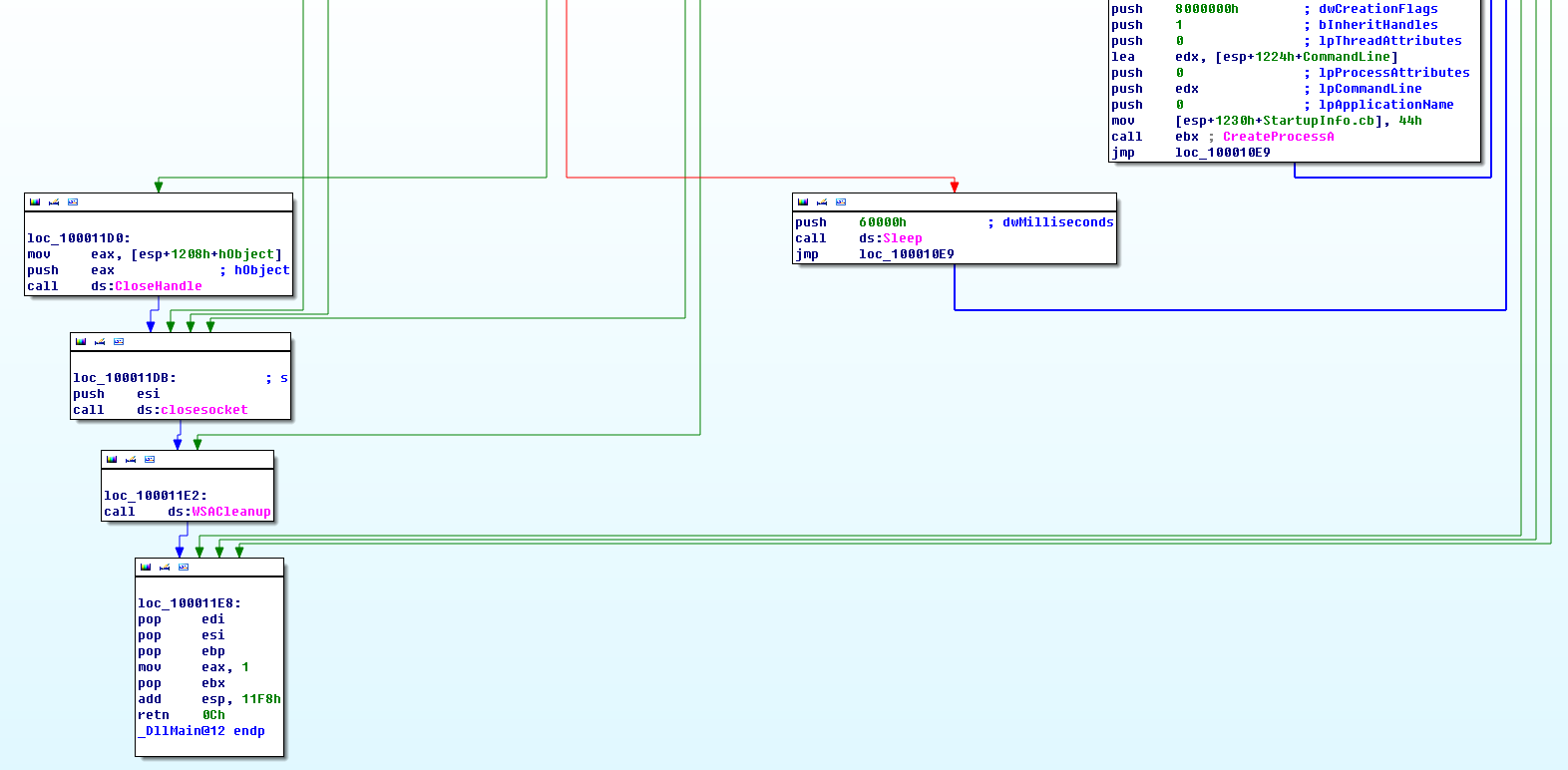


**在使用send后使用了recv接收发送的字符串，之后使用strcmp判断接受的字符串是否为“sleep”，如果是，则会调用sleep函数休眠60秒。**





**DLL的最后使用CloseHandle和CLoseSocket等函数结束链接。**



1. **这个程序如何完成持久化驻留，来确保在计算机被重启后它能继续运行？**

**通过DLL到C:\windows\system32\，并修改系统上每一个导入它的.exe文件，达到持久化驻留。**

1. **这个恶意代码的两个明显的基于主机特征是什么？**

**硬编码使用文件名kerne132.dll。**

**硬编码使用一个互斥量SADFHUHF。**

1. **这个程序的目的是什么？**

**创建后门程序来接远程主机，且难以被删除。两个命令分别用于执行和休眠。**

1. **一旦这个恶意代码被安装，你如何移除它？**

**难以删除，因为感染了系统上每一个使用kerne132.dll的exe文件。**

1. **Yara规则**

**根据以上分析结果，得到Yara规则如下。**

rule lab0701

{

strings:

$string1 = "Malservice"

$string2 = "[**http://www.malwareanalysisbook.com**](http://www.malwareanalysisbook.com)"

$string3 = " Internet Explorer 8.0"

condition:

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

}

rule lab0702

{

strings:

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

$string2 = "CoCreateInstance"

condition:

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

}

rule lab0703

{

strings:

$string1 = "127.26.15.13"

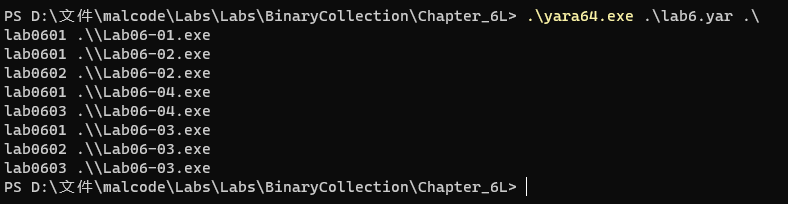
$string2 = "WARNING\_THIS\_WILL\_DESTORY\_YOUR\_MACHINE"

$string3 = "Lab07-03.dll"

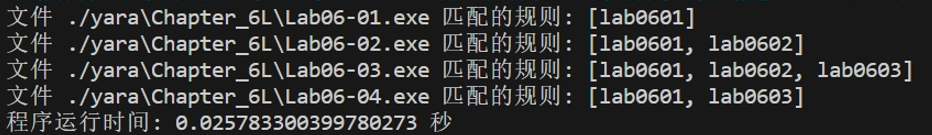
condition:

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)

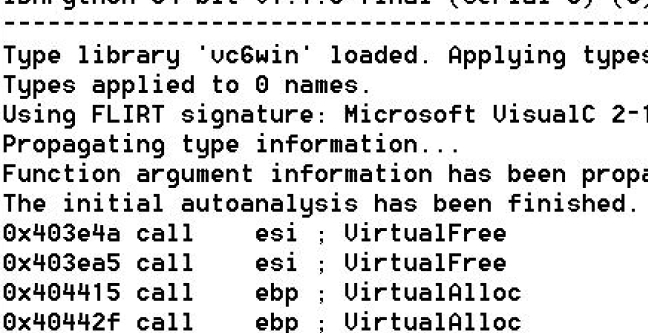
if m == 'call' or m == 'jmp':

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. **对程序使用IDA的静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。**