**恶意代码分析与防治技术实验报告**

**Lab11**

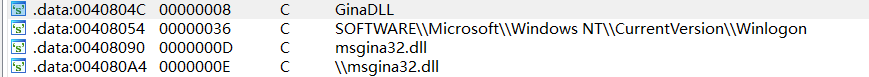
**学号：2013018 姓名：许健 专业：信息安全**

1. **实验内容**

完成课本Lab11的实验内容，编写Yara规则，并尝试IDA Python的自动化分析。

1. **问题解答**
2. Lab11-1

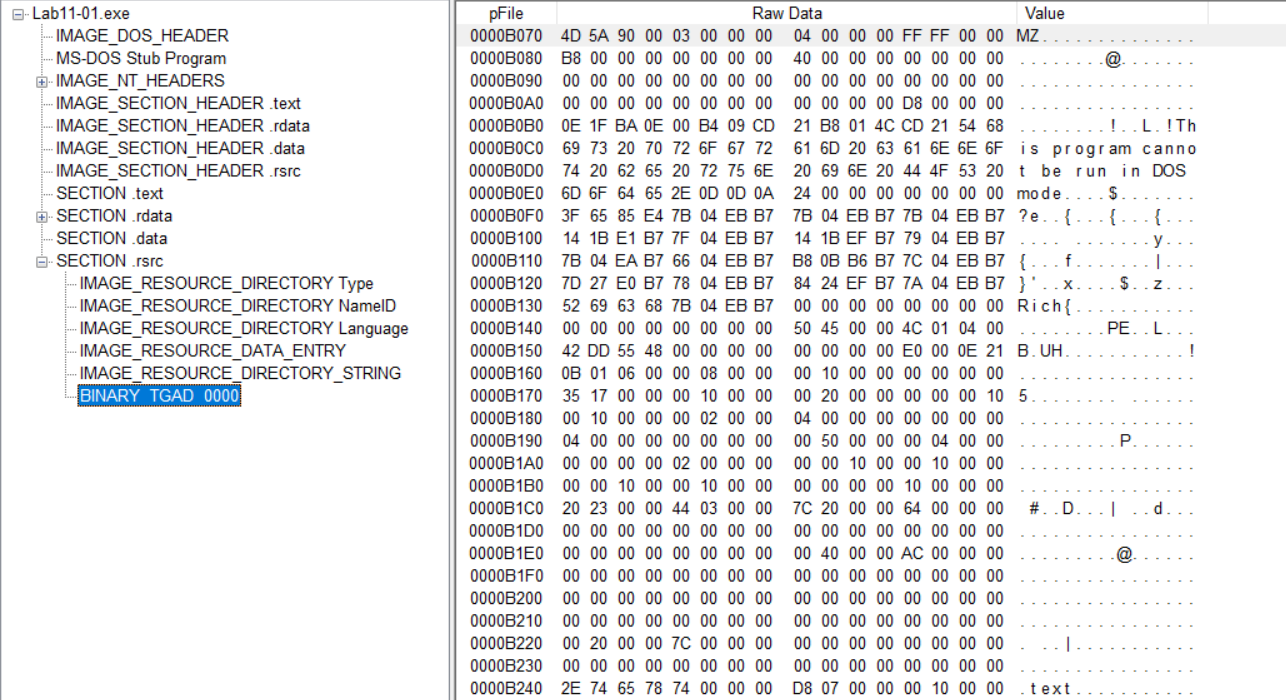
**基础静态分析**



猜测恶意代码要拦截GINA

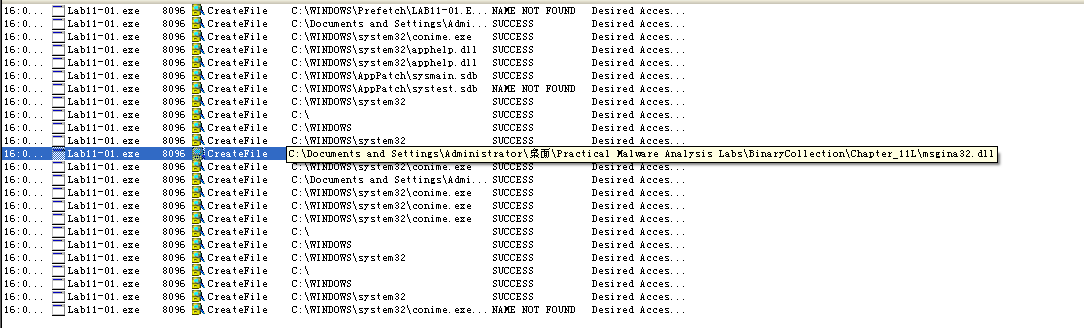


操纵注册表和提取资源节的导入函数



名为TGAD的资源节包含一个内嵌的PE文件

**基础动态分析**

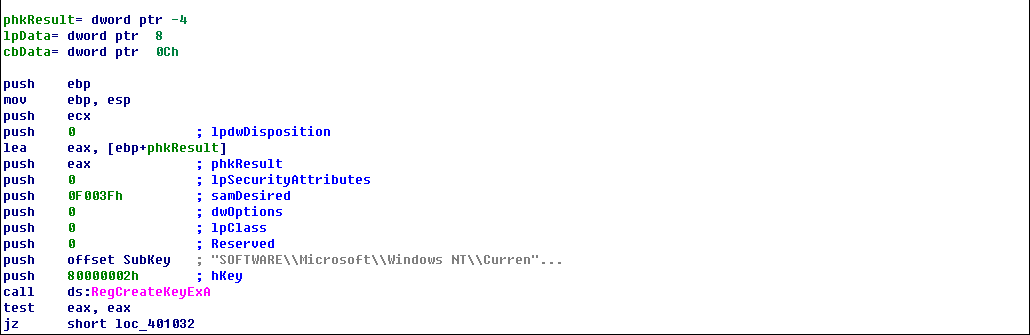


使用procmon监视恶意代码，发现创建文件msgina32.dll在硬盘的启动目录，恶意代码插入msgina32.dll的路径到注册表，系统重启时WinLogon会加载这个DLL

HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL

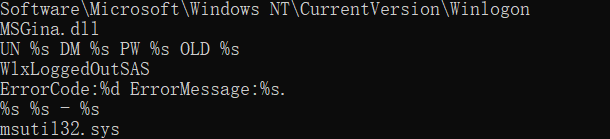
经过比较，msgina32.dll中的内容与TGAD资源节相同

**高级静态分析**

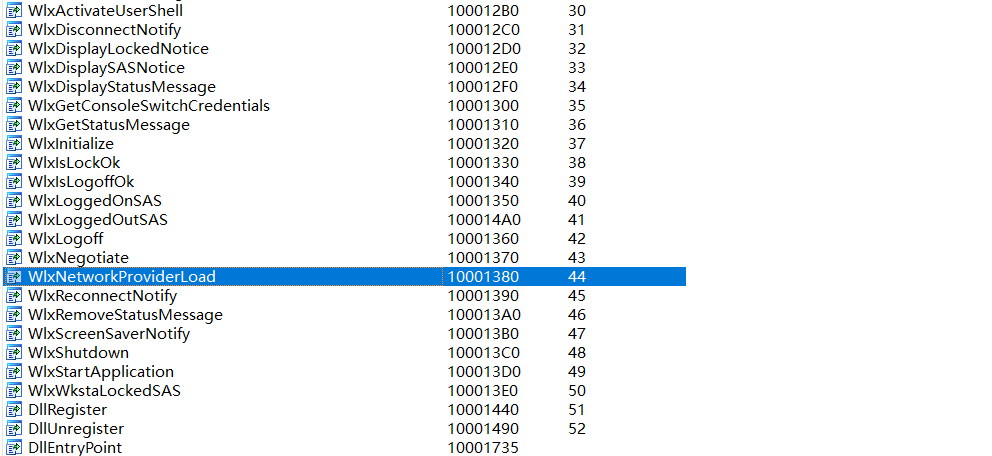


main函数调用了sub\_401080(提取TGAD资源节到msgina32.dll)和sub\_401000(设置GINA注册表的值)，Lab11-01.exe是针对msgina32.dll的安装器，它在系统启动期间由Winlogon加载。

**msgina32.dll分析**



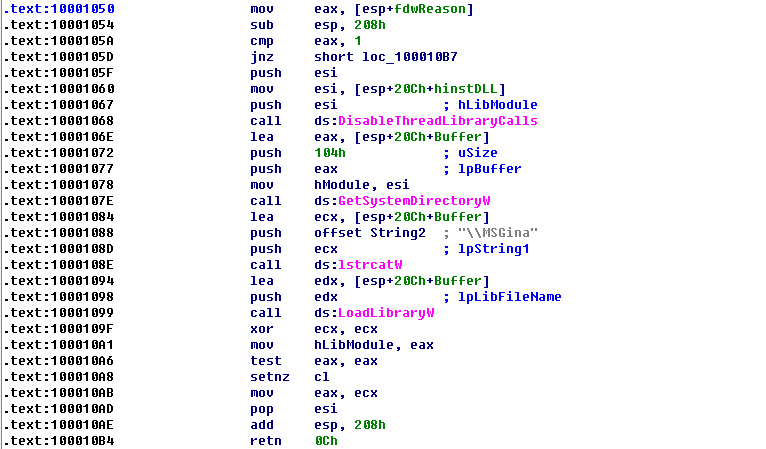
记录用户登录凭证的日志信息



检查msgina32.dll的导出函数表，包含很多以前缀Wlx开始的函数，GINA需要Wlx开始的函数，因此GINA拦截的恶意代码必须包含这些函数。

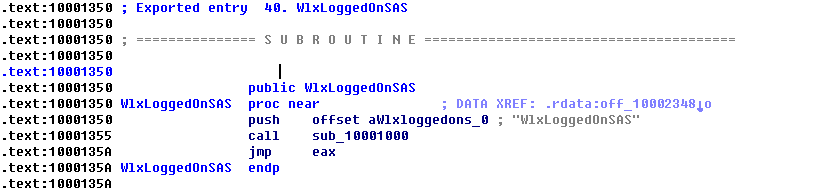
**高级静态分析**

分析msgina32.dll函数的DLLmain，它会检查参数fdwReason，该参数表示DLL入口函数被调用的原因。恶意代码检查传入参数是否是DLL\_PROCESS\_ATTACH，在进程启动或者使用LoadLibrary函数加载DLL时，带有这一参数的入口函数将会被调用。如果这个特殊的DLL在DLL\_PROCESS\_ATTACH期间被调用，那么恶意代码将通过调用LoadLibraryW函数获取Windows系统目录中msgina.dll的句柄。这里被命名为hLibModule。



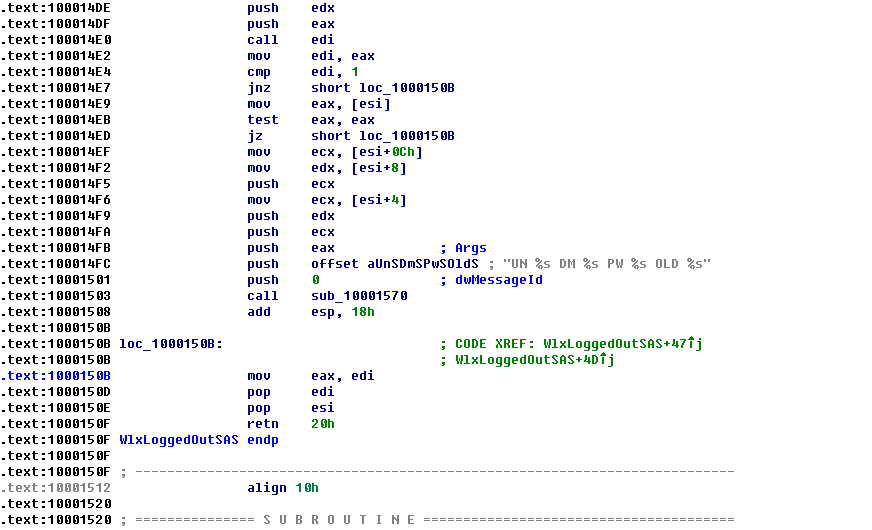


使用hLiibModule可以让这个DLL的导入函数恰当地调用Windows DLL msgina.dll中的函数。既然msgina32.dll拦截Winlogon与msgina.dll之间的通信，则它必须恰当地调用msgina.dll中的函数，从而使系统操作继续正常运行。

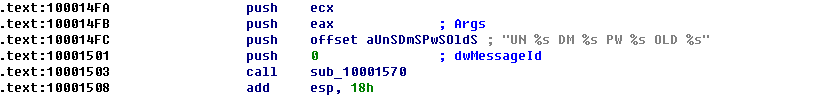


分析导出函数WlxLoggedOnSAS，sub\_10001000只是简单地传递给msgina.dll中真正的WlxLoggedOnSAS函数地址，然后跳转到此处执行。

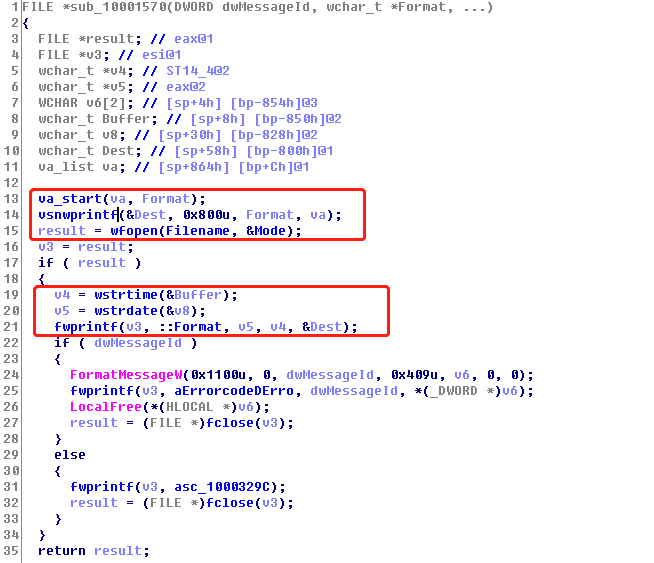




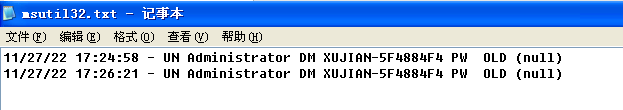
WlxLoggedOutSAS函数包含一些额外的代码（当系统注销时调用WlxLoggedOutSAS）



sub\_10001570是一个登录凭证记录函数，恶意代码打开C:\Windows\System32\msutil32.sys，记录日期、时间、登陆凭证信息等，因此msutil32.sys是一个用来存储记录的日志文件。



重启机器，登陆系统并注销系统，可以看到msutil32.dll中记录用户登录凭证



1. 这个恶意代码向磁盘释放了什么？

这个恶意代码会从名叫TGAD的资源节提取一个文件并命名为msgina32.dll，释放到磁盘

1. 这个恶意代码如何进行驻留？

这个恶意代码会在注册表中添加一个键值来安装这个DLL，系统重启之后，依旧会加载这个DLL

1. 这个恶意代码如何窃取用户登录凭证？

恶意代码用GINA机制来拦截用户的登录凭证，msgina32.dll会拦截所有提交到系统认证的用户登录凭证

1. 这个恶意代码对窃取的证书做了什么处理？

这个恶意代码会将凭证保存在C:\WINDOWS\system32\msutil32.dll中

1. 如何在你的测试环境让这个恶意代码获得用户登录凭证？

重启系统

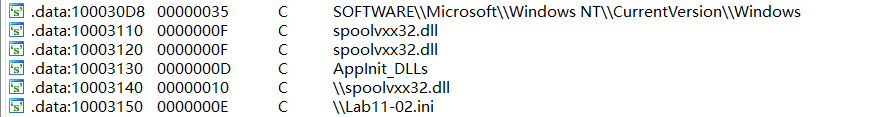
1. Lab11-2

**对Lab11-02.dll做基础的静态分析**

包含一个导出函数installer，导入函数包括操作注册表、修改文件系统、搜索一个进程或者线程列表(CreateToolhelp32Snapshot)



恶意代码使用AppInit\_DLLs来永久安装自身，\\Lab11-02.ini表明恶意代码使用本实验提供的INI文件



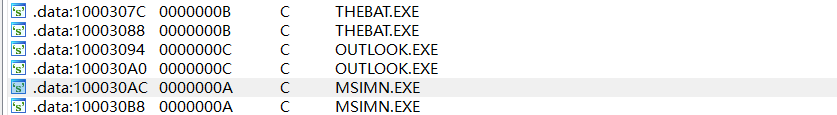
检查INI文件的内容，发现它被编码或加密

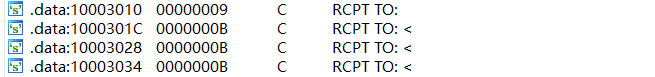


恶意代码可能使用了网络功能，目前我们不清楚执行了什么行为



进程名(THEBAT.EXE、OUTLOOK.EXE、MSIMN.EXE)是邮件客户端，猜测恶意代码对邮件做了某些处理。RCPT是一个SMTP命令，用来创建一个电子邮件的收件人。





**基本动态分析**

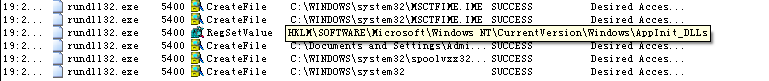
使用installer安装恶意代码



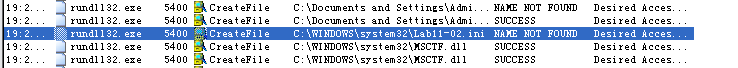
procmon设置过滤器，可以看到恶意代码在windows系统目录下创建了一个名为spoolvxx32.dll的文件，该文件与Lab11-02.dll相同。



恶意代码将spoolvxx32.dll添加到AppInit\_DLLs列表中，导致恶意代码被加载到所有装载User32.dll的进程中。

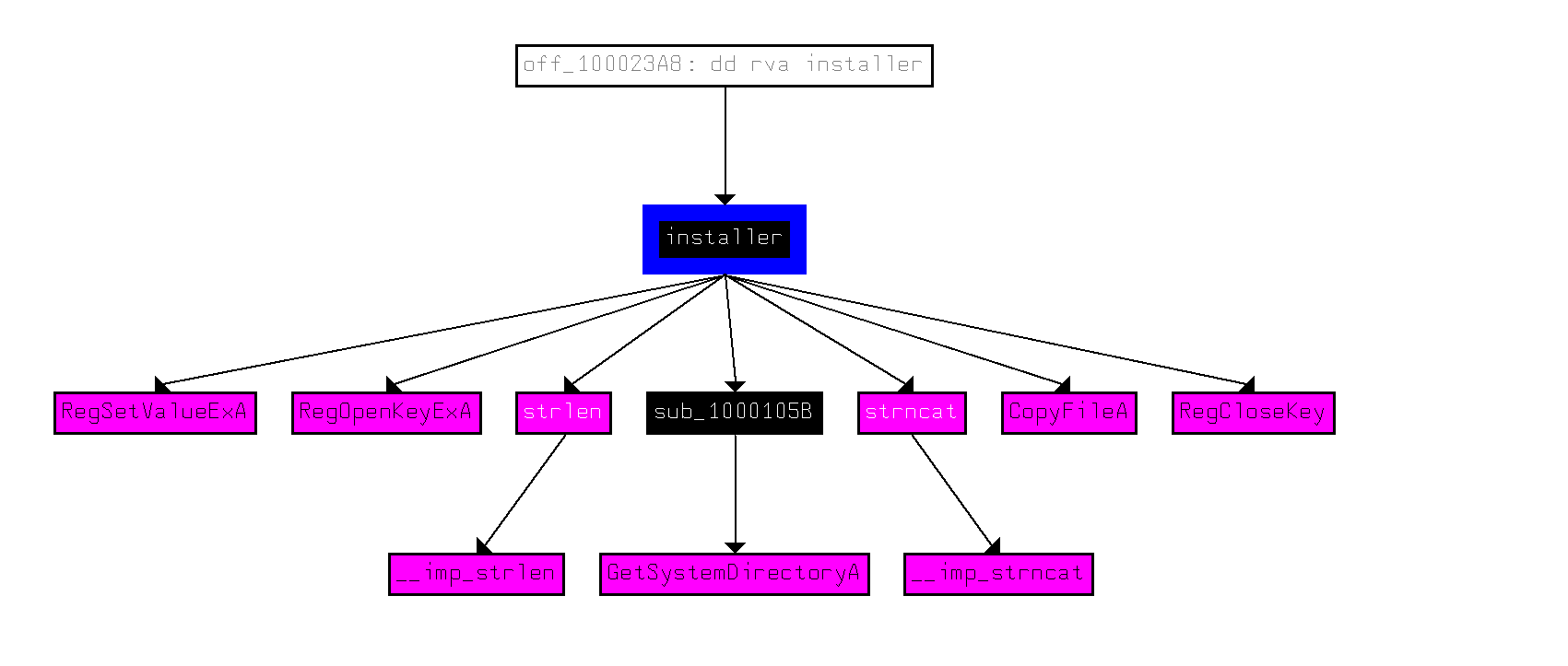


恶意代码从windows系统目录中试图打开Lab11-02.ini文件，为了恶意代码可以访问该文件，我们应该将其放入C:\Windows\System32目录下。



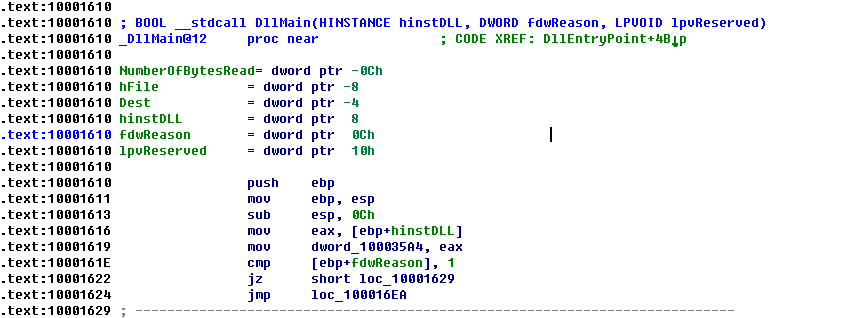
**使用IDA Pro高级静态分析**

查看installer导出函数的交叉引用图，installer在注册表中设置了一个值，并且将一个文件复制到Windows的系统目录，目的就是将恶意代码复制到spoolvxx32.dll，并将其设置为一个APPInit\_DLLs值。

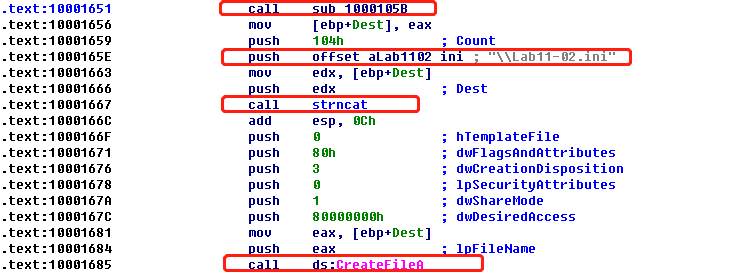


查看DLLmain函数，它首先检查DLL\_PROCESS\_ATTACH

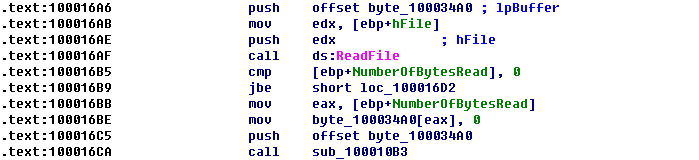
恶意代码仅在DLL\_PROCESS\_ATTACH状态下运行，如果不是则直接返回



恶意代码试图打开C:\Windows\System32\Lab11-02.ini



恶意代码将文件读入到一个全局变量缓冲区中，ReadFile调用后，恶意代码检查确保文件的大小大于0，然后将包含文件内容的缓冲区传递给sub\_100010B3，猜测该程序是解密程序



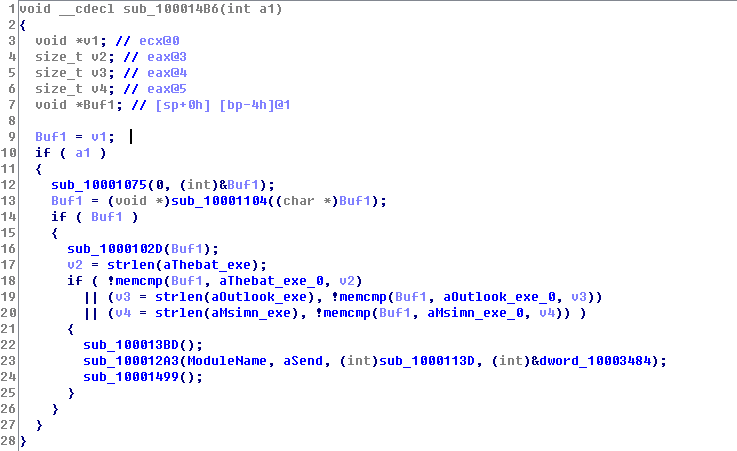
使用ollydbg动态分析，在100016CA处下断点，step over得到如下结果



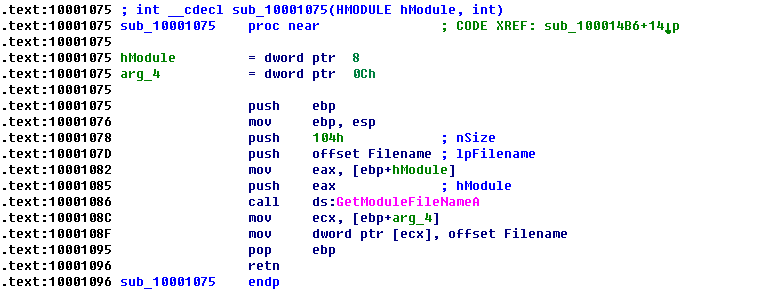
[解密内容为一个邮箱地址](mailto:解密内容为一个邮箱地址billy@malwareanalysisbook.com)billy@malwareanalysisbook.com

该邮箱地址存储在byte\_100034A0处

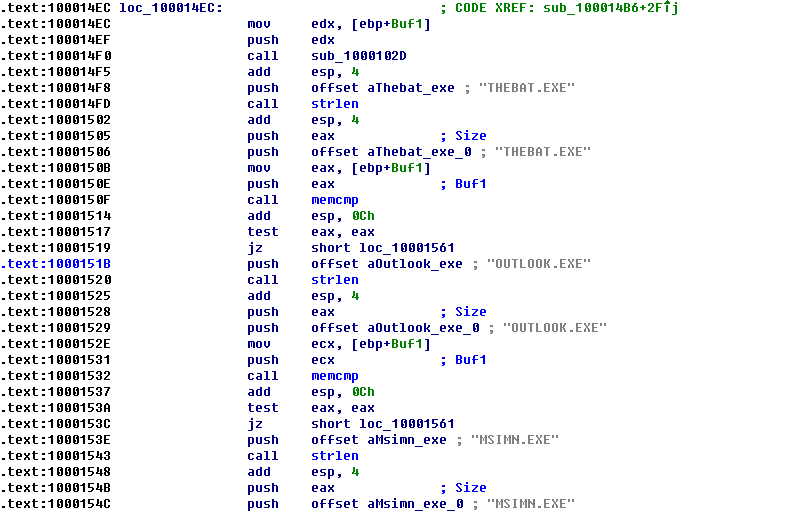
DLLmain中还有函数sub\_100014B6需要分析



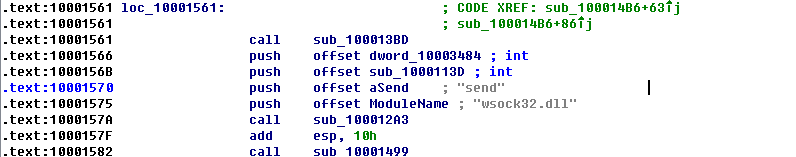
sub\_10001075调用了GetModuleFileNameA函数，参数hModule被设置为0，函数会返回加载这个DLL进程的绝对路劲，接下来恶意代码在arg\_4中返回路径名(字符串指针被传递给函数)，这个字符串被传入至少两个函数，这些函数解析文件名并将文件名中的所有字符转为大写。



接下来，将当前进程大写字母的进程名与THEBAT.EXE、OUTLOOK.EXE、MSIMN.EXE作比较，如果进程名不等于其中的任何一个，恶意代码就会退出



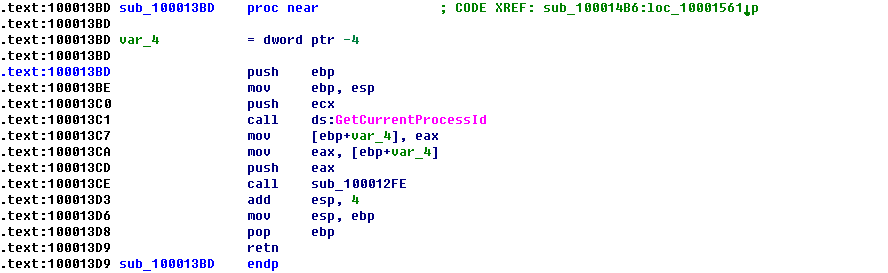
如果恶意代码加载入到这三个进程中的任意一个，就会执行这段代码



函数sub\_100013BD会先调用GetCurrentProcessId函数，然后是sub\_100012FE函数，该函数调用了GetCurrentThreadId，它返回当前运行线程的线程标识符(TID)；

然后调用CreateToolhelp32Snapshot，并且使用结果循环遍历当前进程的所有TID，如果TID不是当前的线程，则用TID调用SuspendThread。

因此函数sub\_100013BD的目的是挂起当前进程的所有运行线程。

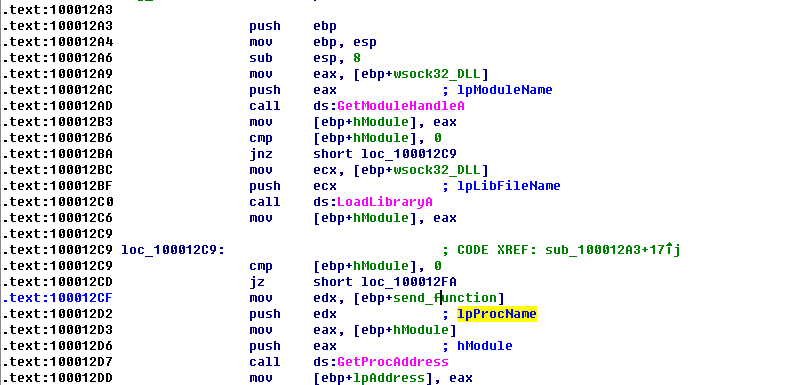




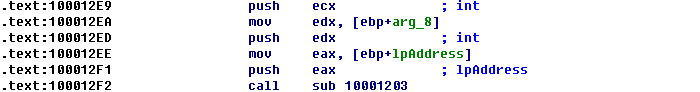
函数sub\_10001499的作用则与之相反，使用ResumeThread来恢复所有的线程

而函数sub\_100012A3应该是恶意代码执行一些影响改变当前进程运行行为的动作，比如修改内核或是安装一个inline挂钩。

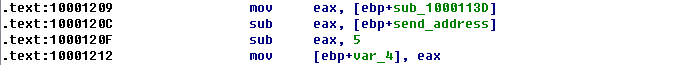
查看函数sub100012A3处的代码，重命名其参数



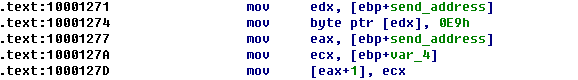
GetModuleHandleA函数获得wsock32.dll的句柄，恶意代码最后传递send函数地址和另外两个参数到sub\_10001203函数(sub\_1000113D和dword\_10003484)



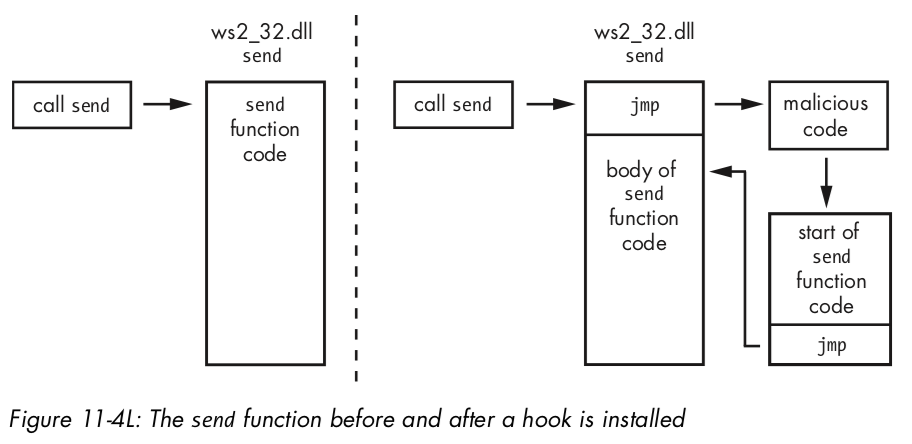
查看sub\_10001203函数，该函数首先计算send函数的内存地址和sub\_1000113D开始的内存地址之间的差，在将差值jmp sub\_1000113D之前，需要从中减去额外的5个字节。



这一部分用来安装inline挂钩，jmp指令0xE9被复制到send函数的开头，之后代码复制var\_4到jmp指令后的内存，作为相对跳转地址，从而让代码跳转到sub\_1000113D处，该函数可以重命名为hook\_function



安装inline挂钩的示意图



1. 这个恶意DDL导出了什么？

Lab11-02.dll包含一个名为installer的导出函数

1. 使用rundll32.exe安装这个恶意代码后，发生了什么？

恶意代码会作为spoolvxx32.dll将自身复制到系统目录中，并且在APPInit\_DLLs键值下永久安装。此外恶意代码尝试从系统目录中打开Lab11-02.ini，但是并没有这个文件。

1. 为了使这个恶意代码正确安装，Lab11-02.ini必须放置在何处？

C:\Windows\System32系统目录下

1. 这个安装的恶意代码如何驻留？
2. 这个恶意代码采用的用户态Rookit技术是什么？
3. 挂钩代码做了什么？
4. 哪个或者哪些进程执行这个恶意攻击，为什么？
5. .ini文件的意义是什么？
6. 你怎样用wireshark动态抓获这个恶意代码的行为？
7. Lab11-3
8. 使用基础的静态分析过程，你可以发现什么有趣的线索？
9. 当运行这个恶意代码时，发生了什么？
10. Lab11-03.exe如何安装Lab11-03.dll使其长期驻留？
11. 这个恶意代码感染Windows系统的哪个文件？
12. Lab11-03.dll做了什么？
13. 这个恶意代码将收集的数据放在何处？
14. **Yara规则**
15. **IDAPython脚本**