基于行为分析的防代码注入技术探讨

吴会松

（福建江夏学院 网络实验办公室，福建 福州 350001）

摘 要：随着网络的普及，木马给我们的带来的威胁越来越大，通过对木马的行为特征的分析，在总结其特征的基础上动态的拦截 API 函数，防止代码的注入技术探讨.

关键词：木马；行为；拦截；注入

中图分类号：TP393.08 文献标识码：A 文章编号：1673-260X（2011）08-0045-03

1 问题的提出

随着电子计算机技术的发展和网络的普及，网络在我们在日常的生活工作中越来越显得重要，比如可以足不出户购物，网上转账等，在给我们带来便利的同时，各种木马病毒给我们工作生活带来的财产、隐私的威胁也越来越大，据《金山安全 2010 木马发展趋势报告》称每年木马产生的经济利益超过 10 亿元.

在 2009 年通过问卷调查，37.89%的病毒通过网页浏览和网页下载进行传播，排在病毒传播的首位，和 2008 年相比，提高了 11.48%，延续了从 2007 年以来的大幅度增长趋势，进一步说明了目前网页“挂马”仍然是最受恶意攻击者青睐的病毒散播方式①.在对用户求救和网络监测的情况分析后，发现大量的网络非法牟利的人通过“挂马”的方式来实现其目的.挂马者主要利用微软以及其它应用普遍的第三方软件（如 realplay，adobe flash、暴风影音等）漏洞进行攻击.“挂马”是指在网页中嵌入恶意代码，当存在安全漏洞的用户访问这些网页时，木马会侵入用户系统，然后盗取用户敏感信息或者进行攻击、破坏.这种通过浏览页面方式进行攻击的方法具有较强的隐蔽性，用户难于发现，因此，潜在的危害性更大.我们必须持续重视浏览器和各种流行应用软件的安全性，提高对”挂马”攻击方式的防范能力[1,2].

2 代码注入行为分析

在 windows 系统中，每个进程都有自己的私有内存地址空间，在写时保护的机制下，使用指针访问内存时，一个进程想进入另一个进程的内存空间进行访问是不允许的. 为了保证程序的稳定性，就好比 MSN 无法利用直接读取内存的方式读取 QQ 内存中存放的图片数据. 但是仍然可以通过一些方法进入并操作进程的私有空间，并将自己的代码嵌入在运行的进程中，要想注入 DLL，现在常用的木马注入技术就是采用运程线程技术 Createmotethread 将木马病毒嵌入在一个正常运行的系统进程中.

寻找特定的进程是现在主流杀毒软件常用的方法，com 和 exe 文件是 Windows 系统下的可执行文件，它们在运行时会产生独立的进程，如果一个木马在运行时就被检测软件查出进程或端口，比如 FPort 软件，是常用的检测端口的软件，一旦木马病毒被发现，那么它们的隐藏也就宣告失败了.由于在 NT 构架的 windows 系统中，管理员可以通过 PSAPI，ToolHelp API 及 PDH（performance Data Helper）等方法来查看进程，所以木马病毒会通过利用 DLL 来实现木马的隐藏，DLL 就是 Dynamic Link Library 的缩写，它是动态链接库的意思，在 windows 操作系统中，API 函数都是在动态链接库中实现的，它有多个功能函数构成，没有程序逻辑，不能独立运行，一般要有进程加载并调用[3].

RUNDLL32 是 32 位操作系统自带的动态链接库工具，它是运行 Dll 文件最简单的方法，在命令行下可以执行 DLL 中的某个函数，比如使用：

Rundll32.exe Exdll GoFunc

如果在 ExDll.dll 的 GoFunc 函数中实现了木马功能，在系统的任务管理器中增加的是 rundll32.exe，而不是相应的木马文件，这样就是实现了木马病毒简单的隐藏.

要检测一个进程是否是木马可以通过判断一个新进程的父进程是否为浏览器进程来判断，在浏览时如果有新的并且父进程是浏览器进程的新进程产生的话，可以认为这个进程就是木马进程，在浏览的网页中含有木马程序[3].

3 代码拦截追踪技术

我们知道一些新的进程会产生，在木马病毒被

激活后,本文所述检测木马的核心思想是 :在进程监视的前提下，由于被映射进进程的私有空间的 DLL 页面,有 COPY ON WRITE (COW) 写时拷贝保护属性[4],可以在核心态下对 API 调用实时监视,结合 detour 拦截函数调用的优势,在浏览有潜在的木马病毒的网页时,要监视浏览器进程的函数调度情况,可以在核心态下监视,这时,函数调用的所有参数都会原原本本地传给 CreateProcesslnternalW,同时 CreateProcessW 会被浏览器调用启动,会多传入两个 0,通过 detour 技术的 API 函数调用拦截技术,可以拦截到这个函数,浏览器在创建新进程时被截获并告知使用者，这样阻止了利用 IE 漏洞在无提示的情况下执行木马程序的代码注入行为.

3.1 用户态下对 API 函数的拦截技术

在用户态下对 API 函数的拦截可以通过系统所含的结构化异常处理机制 (SEH) 来构造单步运行调试器，通过记录每一步 32 位机的指令寄存器的 EIP 值来对 API 函数进行跟踪拦截. 在 SEH 机制中，通过设定多个异常处理程序给每一个线程,也可以设置不同的异常处理程序给每一个线程,以方便处理链的形成.在 NT 系统构架中,不管是在用户态还是在核心态，寄存器 FS 都保存有当前线程的 TIB (信息块结构) .在 TIB 中,系统如果发生异常,通过回调 handle 函数地址,通过指向 EXCEPTION_REGISTRATION 的第一项结构将系统处理控制转回到这个结构注册的回调中.通过在标志寄存器的陷阱标志位第八位模式下的单步模式运行,每被中央处理器执行一条指令，都会产生异常,随后会自动复位陷阱标志位,如果在每一个单步处理过程中都重新设置陷阱标志就可以保持单步跟踪状态.

3.2 核心态下 API 函数调用拦截技术

在系统中,API 函数在执行时通常要用到系统空间的函数,而在用户态下,由于权限有限,这时须转入到核心态下来对 API 函数的调用进行监视,而要进入核心态,可以通过调用门 CallGate 来实现从用户态 ring3 进入核心态 ring0,如果要从核心态转回到用户态，可以通过段间返加指令 RETF 来实现.在对 API 函数进行拦截的过程中,确定了 32 位机的指令寄存器 EIP 的位置也就截获了 API 函数,在 WIN2000 系统中,在 IDT 中断向量表的 2E 号中断的描述符内有转移的目标地址,通过调用中断门 CallGate 来实现,而在 WIN XP 系统中,是通过 sysenter 来实现的.

在核心态下,要对 API 函数进行拦截,可以通过调用 InterlockedExchange () 或用 MDL 来锁住 SSDT 后进行修改,在核心态下,工作的重点就是通过 EAX 值来索引 SSDT 提供的调用函数地址,其他的工作还有将用户空间的参数拷贝到核心空间、系统初始化堆栈等工作,可以用 MDL 来锁住 SSDT 后再修改,或者调用 InterlockedExchange () 来修改,之后就可以将相应的控制转到函数中去了.通过停止 COW 机制后修改 CR0 可以有一些系统的 DLL 函数安装 hook 钩子,要钩挂系统 API 函数,一般情况下有两种方式，一是找到 API 函数入口点通过 detour 开发库来将最前面的 5 个字节做出修改[6] 并将语句跳转到新的代码上,然后再返回原函数的代码中,另一种方式是通过模块的引出或引入表来对 API 函数的入口地址调整到新的大代码段中[4].

找到了入口地址的位置才能对 API 函数的入口地址做出修改，在 kernel32. dll 中有一个 API 函数 CreateRemoteThread,并且动态链接库 DLL 是经过一组输出函数地址表来向系统提供相应的服务. 所以要实现钩挂函数就必须清楚 DLL 的结构. WindowsNT 系统中, 动态链接库文件是作为 PE 文件格式进行存储的.如图 1 所示:

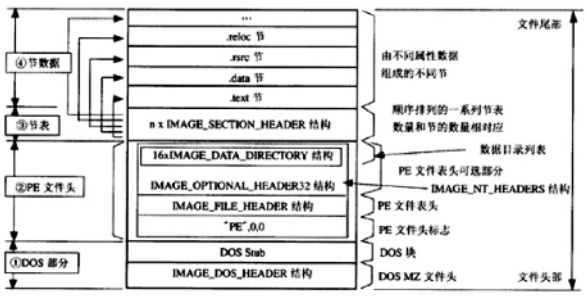

图 1 PE 文件基本结构

如上图所示 PE 首部是一个 IMAGE_NT_HEADERS 的结构,PE DWORD 标志后的是结构 IMAGE_FILE_HEADER.这个域只包含这个文件最基本的信息.这个结构表现为并未从它的原始 COFF 实现更改过.除了是 PE 首部的一部分,它还表现在微软 Win32 编译器生成的 COFF OBJ 文件的最开始部分.PE 首部的第三个组成部分是一个 IMAGE_OPTIONAL_HEADER 型的结构.图 1 中所示的 IMAGE_OPTIONAL_HEADER 结构的最后一个 DataDirectory 域是 IMAGE_DATA_DIRECTORY 结构的结构数组，它由 16 个成员组成.而 IMAGE_DATA_DIRECTORY 结构是由两个成员组成的,第一个是指示特定数据的 RVA(相对虚拟地址),第二个是这个特定数据的大小信息.在这个结构数组中第一个成员指向的是此模块的函数引出表.

在这个函数引出表结构中,同最终找到具体的引出函数相关的域有三个.第一个是位于引出表相对偏移 20H 处的 AddressOfNames 域，这个域中的

数组时 DWORD 类型的，每一个 DWORD 表示一个 ASCII 字符串的相对虚拟地址，此模块提供函数的函数名就是这个字符串，想要得到这个由这个模块引出的所有函数名可以通过遍历这个数组来实现. 第二个是 AddressOfNameOrdinals，它是位于引出相对偏移 24H 处的域，在此域中，数组也是 DWORD 类型的，这个 DWORD 数组的每项和引出表减去 Base 域的值，结果就是一个函数的索引值.第三个域是 AddressOfFunctions，它处在引出表相对偏移 1CH 处，它的作用和上述其他的域相似，需要声明的是在实际使用中，用 GetModuleHandleA 函数可以得到模块的基地址，而以上阐述的很多域都使用了 RVA.

4 相关测试

要达到隐藏的目的，可以通过拦截 RegQueryValue 函数来过滤注册表信息，这样往往可以躲避杀毒软件的监视控制.也就是说侦测病毒程序的有效方法之一，通过追踪 RegQueryValue 函数的执行过程，并记录其执行过程所经历的模块信息可以实现.在 XP 系统中，利用 detour 开发库的 JMP 法设计注入了一个动态链接库模块到目标程序空间来实现对该函数的拦截，并动态跟踪了该函数的执行过程[7].

表一显示了 RegQueryValue 函数在被 hook 钩挂后在用户空间所执行经历过相应模块信息的分析图表，在这个表中，列 1 是模块经历的顺序，列 2 是经历模块的名称，列 3 是经历模块跳转的入口地址，其中第 3、17 行是对注入的拦截模块的显示.

表 1 跟踪结果

序号	经过模块	模块入口地址
1	c:\windows\sysytem32\advapi32.dll	76DCB775
2	e:\hookjmp.dll	00DE213E
3	c:\windows\sysytem32\advapi32.dll	76DCB775
4	c:\windows\sysytem32\ntdll.dll	6DA22100
5	c:\windows\sysytem32\advapi32.dll	76DA6CF7
	……	
16	c:\windows\sysytem32\advapi32.dll	76DD9DD5
17	e:\hookjmp.dll	00DE2258

由于在用户态下运行调试时要做好函数执行的每步指令寄存器的 EIP 值的记录，所以它会影响到函数的执行速度.为解决以上不足，在函数执行完参照枚举的模块信息后再对每个所记录得到的指令寄存器 EIP 值所属模块的分析，定位 API 函数并获取内核调用地址是截获流程所要做的工作，而且在追踪时进行钩挂，通过定点定位拦截截获到指定 API 函数在系统空间中调用系统服务的过程，这样可以大大提高执行效率，对整个系统运行的影响比较有限，在系统空间也不会有太大的工作量要做.通过本实验，RegQueryValue 函数转入到系统空间执行的参数记录如下 转入系统空间的起始地址是 0x90631500；拦截点的地址是 0x906315E0 ；SSDT/SHADOW SSDT 通过查表基地址得到是 0x90631528 ；查表索引值得到是 0xB5 ；系统服务函数地址是 0xba3ec652.

模块 hookhelp.sys 的范围为 0xba3e8000~0xba3ef000，ntoskrnl.exe 的范围是 0x904d8000—0x906e3000，由此可知，这个函数在内核执行过程中，调用 SSDT 表中所输出的内核函数的地址 0xba3ec652 落在 hookhelp.sys 模块中. 而正常情况下，该内核函数应为地址应落在 ntoskrnl.exe 模块中的 NtQueryValueKey. 这说明 ntoskrnl.exe 模块函数被 hookhelp.sys 模块拦截了.

5 结束语

现在木马病毒对人们的工作生活的危害越来越大，通过本文所述的采用结构化异常处理机制（SEH）及基于内核定点定位拦截 API 函数的技术可以有效突破目前普遍被采用，通过对 API 函数整个执行过程的跟踪，并对照相应的模块信息记录它们运行过程所经历过的模块信息，这对于提高信息的安全，防止代码的注入有一定的借鉴意义.

参考文献：

〔1〕张健.我国计算机病毒现状和发展趋势[R].天津：国家计算机病毒应急处理中心，2008.

〔2〕国家计算机病毒应急处理中心.2009 年中国计算机病毒疫情调查技术分析报告[EB/OL]. http://www.antivirus-china.org.cn/head/diaocha2009/report2009.doc.

〔3〕商海波.木马的行为分析及新型反木马策略的研究[D].浙江工业大学，2005.

〔4〕曹俊.基于网络传播的计算机病毒机理及防御策略研究[D].中南大学，2004.

〔5〕杨新柱.可执行文件格式分析与应用[D].北京邮电大学，2009.

〔6〕朱若磊. 基于动态追踪技术侦测 API 钩挂[J]. 计算机工程与设计，2010,31(15).