第 24 卷 第 3 期
2009 年 6 月

天津科技大学学报
Journal of Tianjin University of Science & Technology

Vol.24 No. 3
Jun. 2009

嵌入式病毒传播原理和主动防御技术

李纪扣，程 艳
（天津科技大学计算机科学与信息工程学院，天津 300222）

摘 要：依据嵌入式病毒的传播理论，结合 Windows 操作系统的系统调用机制，提出了一种基于 Hook 系统服务的嵌入式病毒防御系统. 该系统以注册表和进程为主要监控点，结合访问控制规则，实时监控进程操作行为. 实验表明：该系统不仅可以有效地防御已知病毒，还可以防御未知病毒.

关键词：嵌入式；病毒；系统服务；监控；
中图分类号：TP309.5 **文献标志码**：A **文章编号**：1672-6510(2009)03-0071-04

Embedded Boot Virus Principle and Active Defense Technology

LI Ji-kou，CHENG Yan
(College of Computer Science and Information Engineering，Tianjin University of Science & Technology，
Tianjin 300222，China)

Abstract：Based on embedded virus's dissemination principle and the Windows operating system's call mechanism，an embedded virus defense system based on the Hook system service was proposed. Using registry and advancement as the main monitoring point，and considering access control rule，real-time advancement operation behaviors were monitored. Experiments show that this system can defense not only the known virus but also the unknown virus.

Keywords：embedded；virus；system service；monitoring；

随着互联网技术的迅速发展，网络已经深入到人们的社会生活之中，但同时由于病毒泛滥造成的巨大破坏也日益引起人们的重视[1]. 随着杀毒软件检测技术不断提高，采用传统用户态技术的病毒的生存空间越来越小，因此，病毒技术的研究已经开始转向了插入内核的动态嵌入技术[2]而实现其在目标机的深度隐藏，隐蔽地进行信息窃取和各种破坏活动.

传统的防御多采用分层防御结构，它包括硬件防火墙、入侵检测系统、入侵防御系统、个人防火墙软件、病毒防御软件和其他安全装置等. 这种防御结构的优势在于，攻击者实行攻击时需要突破层层的防御，即使攻击者能够成功的绕过外层的防御，还要面对内部的防御. 但这种防御也存在着一些不足：第一、被动的防御模式. 每一层的防护措施都是采用被动的防御模式，必须经常更新来面对新形式的功击；第二、过分依赖“人”的作用. 更新配置和防护软件，仲裁什么样的代码可以在系统中运行，都需要依靠人来完成，如果用户没有足够的安全知识，很难做出正确的判断；第三、各层防御对来自内部的攻击防御不够，如果直接在终端进行修改磁盘等毁灭性的破坏，这些都不会被察觉. 针对这些不足，就需要一种能够从系统内部实施保护的防御措施.

基于上述防御方法的缺点，本文给了一种挂接系统服务调用的防御嵌入式病毒的方法，通过监控注册表和进程来实现对此类病毒的检测. 这种方法从系统内核中进行监控，既可以防御已知病毒，又可以防御未知的病毒.

1 嵌入式病毒传播原理

嵌入式病毒是一种危害极大的病毒程序，它以 DLL 文件的形式存在. DLL 文件[3]是 Windows 中包

收稿日期：2008-12-31；**修回日期**：2009-02-25
作者简介：李纪扣（1963—），男，天津人，副教授，lijikou@tust.edu.cn.

含函数和数据的模块集合,当应用程序调用它的功能函数时,得到系统加载. 嵌入式病毒就是利用了这一原理加载到系统进程中的. 病毒体一般采用的技术是:(1)动态嵌入技术[3-4]. 该技术是通过访问目标进程私有空间将执行代码连接到进程中,作为一个进程的非核心线程运行,成为目标进程的一部分,因此难以被发现和终止. 嵌入式病毒就是利用这一技术将 DLL 文件加载到目标进程中的. (2)安装后门技术. 嵌入式病毒本身不具有自启动的功能,病毒会因为电脑关机或者系统发生异常导致结束运行而彻底失去作用,为了长期控制入侵主机并发挥作用,一般病毒会修改系统的注册表以便在系统重新启动以后,自动启动病毒程序.

2　基于系统调用的防御技术

2.1　Windows 系统服务调用机制

系统服务是由操作系统提供功能,应用程序通过 API 函数直接或间接地使用系统功能. 在操作系统中存在两种工作模式:用户和内核模式. 应用程序一般运行在用户模式下,当应用程序调用 Win32 API 函数时,该函数对传入的参数进行检查确认,并转换成 Unicode 形式. 然后,调用基于 Ntdll.dll 的本地系统服务并触发 INT2EH 指令,然后进入内核模式. 系统调用 KiSystemService 函数,内核根据所传入系统服务号查找所需的服务函数入口地址,这个指针指向内核模式中的 Ntoskrnl.exe 提供的系统服务函数,最终来完成这个服务程序. 图 1 为 Windows 操作系统调用结构层次图.

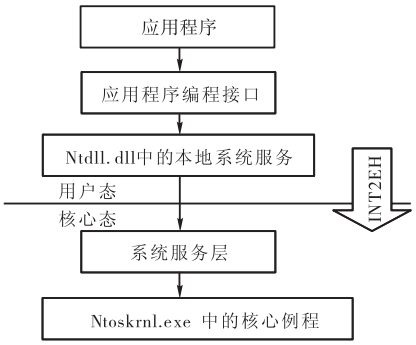

图 1　Windows 操作系统调用结构层次图

Fig.1　Windows operating system transfer structure level chart

2.2　截获系统服务的技术

核心态系统服务截获技术[4]是通过截获执行系统服务分配表 SSDT 的查找操作,改变 SSDT 中某系统服务号对应的系统服务函数指针,将指针指到开发者插入的其他函数入口点. 首先,定位系统 SSDT 是实现劫持的关键. 在 Windows2000 中有一个未公开的 Ntoskrnl.exe 输出表,可以通过它的一个结构数组 KeServiceDescriptorTable 来完成对 SSDT 的访问与修改,其数据结构如下所示[5]:

```
Typedef struct ServiceDescriptorTable{
PVOID *ServiceTableBase;
//在 SSDT 中每个服务调用次数
PULONG ServiceCounterTableBase;
//ServiceTableBase 描述的服务个数
ULONG NomberOfServices;
//SSPT 的基地址
PUCHAR PajamTableBase;
}ServiceDescriptorTable
```

参数 ServiceTableBase 指向系统服务程序地址列表首地址,通过它来获取系统服务表基地址,根据请求的系统服务的 ServiceID 找到对应系统服务函数地址. 用自定义的系统服务函数替代原始函数,同时保存原始函数入口地址. 当调用该系统服务时,就会执行自定义的系统服务,如果允许其调用原服务,则恢复其入口地址调用原始服务函数;如果不允许其调用,则在执行完自定义的服务函数后直接返回.

2.3　截获系统服务的实现

以创建远程线程函数 CreateRemoteThread 为例,它对应的系统服务调度表函数为 ZwCreateThread,只要通过 Hook 函数就可以实现对该函数调用的控制. 下面给出挂接程序的主要程序代码.

```
#define SYSCALL(_api)
KeServiceDescriptorTable.ServieeDescriptor[0].KIServiceTable[*((PULONG)((PCHAR)(_api)+1))]
```

通过这个宏[5],就可以实现 HOOK 系统服务分配表中的服务,HOOK ZwCreateThread 实例如下:

```
OrgZwCreateThread=SYSCALL(ZwCreateThread);
SYSCALL(ZwCreateThread)=MyZwCreateThread;
```

OrgZwCreateThread 用来保存原始系统服务的地址,MyZwCreateThread 自定义函数用来替换原始系统服务. 通过以上面的方法,就可以实现 HOOK 系统服务的任务,然后在替换函数中来修改系统行为.

3　实现对病毒的主动防御

采用一个底层的防御手段,将传统被动防御方式

转变为主动的防御方式，以安全操作系统的访问调用机制和访问控制思想为基础，实现了一个更底层的针对嵌入式病毒的自我防御方法.

3.1　访问控制规则的建立

建立访问控制规则是整个防御系统的核心，有助于控制操作系统中企图运行的应用程序，防止恶意程序的运行.

3.1.1　建立访问控制规则信息数据结构

通过分析嵌入式病毒的传播原理，本文提出要对注册表和进程进行保护，把它们的访问控制信息都存储在相应的链表 pKRuleList 和 pPRuleList 中. 在链表中增加新规则时，要判断链表中是否已经存在该规则，如果存在则覆盖原规则；否则，将其添加到链表的尾部.

访问注册表控制规则的数据结构为

```
Typedef struct reg_rul{
REG_RUL *pNext;
PCHAR RegName;
PCHAR RegUser;
ACCESS_PLACE AcPlace;
PCHAR ProcName;
ACCESS_TYPE AcRights;
}REG_RUL,*PREG_RUL;
```

pNext 是指向 REG_RUL 结构的指针；RegName 为要保护的注册表项；RegUser 是允许访问的客体；AcPlace 用来指明允许访问的地点，包括本地或者网络访问；ProcName 是允许访问注册表项的进程名；AcRights 用来表明进程 ProcName 对应的访问权限.

进程访问控制规则数据结构为

```
Typedef struct p_protected{
P_PROTECTED *pNext;
PCHAR FName;
PCHAR PrName;
} P_PROTECTED,*PP_PROTECTED;
```

其中，pNext 是指向结构 P_PROTECTED 的指针，以构成一个链表；FName 是保护的进程文件名；PrName 是允许在该进程空间中创建线程的应用进程名.

3.1.2　关键访问要素的提取

关键访问要素的提取包括访问地点和访问进程的提取：

(1) 访问地点的提取. 通过函数 ZwOpenProcessToken 来获取获取登陆用户的访问令牌句柄，然后通过 ZwQueryInformationToken 函数来获得是哪个模块发布的令牌，如果是由 User32 模块发布的，就说明该访问为本地访问，否则为网络访问.

(2) 访问进程的提取. 下面给出获取访问进程信息部分代码：

```
PEPROCESS pPROC;
pCHAR ProcName;
pPROC=PsGetCurrentProcess();
ProcName=(PCHAR)PROC+PROCESSNAME_
OFFSET;
```

3.1.3　访问控制规则的匹配

第一步，根据提取的访问信息来检查访问控制规则链表，如果该链表中不存在该访问规则，则直接调用原来的系统服务；如果存在的话，则转入第二步.

第二步，根据访问控制规则来判断当前访问者的用户是否为授权用户，如果为授权用户则转入第三步；否则拒绝访问.

第三步，获取访问者使用的访问进程，获取其访问权限. 根据访问控制规则来判断该进程是否有访问权限，如果有权限则转入第四步；否则拒绝访问.

第四步，根据访问控制规则来判断获取的访问地点是否合法，如果合法，则允许调用原始系统服务；否则，返回STATUS_ACCESS_DENIED 指示，拒绝访问.

3.2　注册表保护模块

通过上文分析，嵌入式病毒就是利用注册表具有启动的功能，把嵌入式病毒的启动程序写入注册表的启动项中. 为了防止病毒被启动，可以通过截获注册表系统服务调用，实现对注册表的访问控制.

病毒对注册表的修改一般由 RegCreateKey 函数完成，因此只要监测系统对此函数的调用，并通过修改信息判断是否调用了病毒函数，就可以实现防止修改行为.

(1) 首先通过 PathFromHandle 函数来获取将要访问的注册表键值的全名.

(2) 利用 CheckFileByName 函数检查 pKRuleList 链表，判断是否存在该规则. 如果该规则存在，则执行(3)；否则，不进行访问检查，直接调用 ZwCreateKey 函数.

(3) 根据提取的访问信息与访问规则库进行匹配，如果符合则允许其调用原系统服务函数；否则，拒绝访问.

3.3　进程保护模块

通过分析可知，嵌入式病毒采用线程注入技术把自己的病毒代码加载到操作系统的关键进程中的. 因此，可以通过截获相关的系统调用，判断用户是否有修改进程的权限.

要实现对目标进程的注入,首先要利用 OpenProcess 函数打开目标进程获取目标进程的句柄,然后把目标进程的句柄传递给 CreateRemoteThread,其对应的系统服务函数为 ZwCreateThread,通过 HOOK 系统服务 ZwCreateThread 便可以实现对注入目标进程的保护.

在截获函数 MyZwCreateThread 中,首先根据传递的参数 ProcessHandle 来决定目标进程是自身调用动态连接库文件,还是被其他进程注入线程,如果参数 ProcessHandle 为 NULL,就是自身调用. 否则,通过 ObReferenceObjectByHandle 函数来获取目标进程对象的指针,然后根据访问控制规则来判断目标进程是否允许被注入,如果允许注入,那么就注入目标进程. 否则,拒绝该操作. 进程监控的实现流程如图 2 所示.

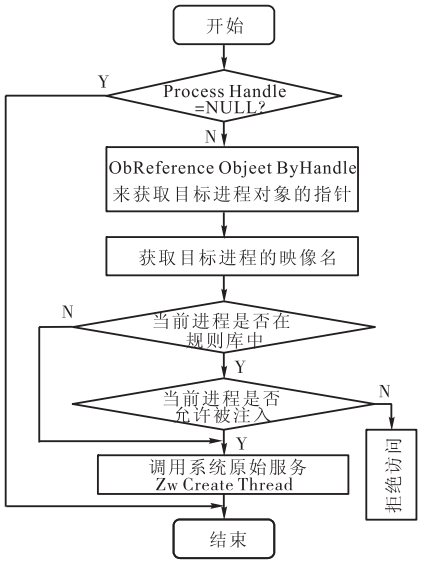

图 2　进程监控实现流程图

Fig.2　Advancement monitoring flow chart

4　实　验

为了验证该防御系统对嵌入式病毒的防御效果,选取了广外男生、灰鸽子、特洛伊木马、Bighorse、GetIp 和 TestVirus 病毒进行实验. TestVirus 是在研究嵌入式病毒的行为特征的基础上,由作者编写的一个病毒. GetIp 属于脚本病毒.

实验环境为 Windows 2000 操作系统. 测试软件包括本文提出的防御系统、天网个人防火墙和诺顿防火墙. 测试结果见表 1.

从实验结果可以看出:本文的防御系统对嵌入式病毒防御成功,对非嵌入式病毒防御不成功;对采用 NdisHook 信道的 Bighorse 和未知病毒 TestVirus,天网个人防火墙和诺顿防火墙防御不成功,而本文提出的针对嵌入式病毒的防御系统却能发现其行为,并依据规则库进行判断.

表 1　测试结果

Tab.1　Test result

测试项目	信道方式	防御系统	天网	诺顿
广外男生	线程插入	√	√	√
灰鸽子	线程插入	√	√	√
特洛伊	线程插入	√	√	√
Bighorse	NdisHook	√	×	×
GetIp	http	×	√	√
TestVirus	线程插入	√	×	×

注:“√”为防御成功;“×”为防御失败.

5　结　语

本文分析了嵌入式病毒的传播原理和传播的主要途径,给出了一种基于挂接系统服务的嵌入式病毒防御系统. 该系统可以实时监测进程和注册表访问行为. 从分析和实验可以看出,传统的防御方法由于是基于对已知病毒的分析和研究之上的,所以在检测时能够更准确,误报率低;但是,对于未知的新病毒,由于知识库缺少该类病毒的特征数据,很可能会产生漏报的后果. 而基于挂接系统服务的嵌入式病毒防御系统则不存在此问题,使这类新病毒不至漏网.

参考文献:

[1] 李伟斌,王华勇,罗平. 通过注册表监控实现木马检测[J]. 计算机工程与设计,2006,27(12):2220–2222.

[2] 郝东白,郭林,黄皓. 基于限定令牌的木马防护系统设计[J]. 计算机工程与应用,2007,43(24):141–146.

[3] 于继江. 动态嵌入式 DLL 木马实现方法[J]. 电脑知识与技术,2005(7):47–49.

[4] 骆力明,符宇同,鲁悦. 利用 Hook 技术实现进程控制[J]. 微计算机信息,2007,24(5-3):240–242.

[5] 张连成. 主机主动入侵防御系统的研究与实现[D]. 河南:解放军信息工程大学,2007.

[6] 王元西,庞国仲. 基于钩子函数通用对话框设计与实现[J]. 微计算机信息,2000,16(2):58–60.

[7] Dorothy Denning. An intrusion detection model[J]. IEEE Transaction on Software Engineering,1987,13(2):222–232.

[8] 唐北平,吴德健. 计算机病毒程序的检测和清除[J]. 计算机时代,2003(6):7–8.