CHAPTER 5

第5章 恶意代码攻击机理分析

5.1 恶意代码概述

5.1.1 恶意代码的定义

计算机病毒是早期主要形式的恶意代码。在20世纪80年代，计算机病毒诞生，它是早期恶意代码的主要内容，由Adleman命名、Cohen设计出的一种在运行过程中具有复制自身功能的破坏性程序。在这之后，病毒被Adleman定义为一个具有相同性质的程序集合，只要程序具有破坏、传染或模仿的特点，就可认为是计算机病毒。这种定义使得病毒内涵被赋予了扩大化的倾向，将任意带有破坏性的程序都认为是病毒，掩盖了病毒潜伏、传染等其他重要特征。20世纪90年代末，随着计算机网络技术的发展进步，恶意代码(Malicious Code)的定义也被逐渐扩充并丰富起来，恶意代码被Grimes定义为从一台计算机系统到另外一台计算机系统未经授权认证，经过存储介质和网络进行传播的破坏计算机系统完整性的程序或代码。计算机病毒(Computer Virus)、蠕虫(Worms)、特洛伊木马(Trojan Horse)、逻辑炸弹(Logic Bombs)、病菌(Bacteria)、用户级RootKit、核心级RootKit、脚本恶意代码(Malicious Scripts)和恶意Active X控件等都属于恶意代码。

因此可以总结出恶意代码两个显著的特点：非授权性和破坏性。表5-1列举了恶意代码的几个主要类型及关于其自身的定义说明和特点。

5.1.2 恶意代码的危害

恶意代码问题不但造成了企业和众多用户巨大的经济损失，还对国家的安全产生了严重的威胁。目前国际上一些发达国家都投入大量资金和人力对恶意代码领域的问题进行了长期深度的研究，同时在某种程度上取得了显著的技术成果。1991年的海湾战争，据报道，伊拉克从国外购买的打印机被美国植入了可远程控制的恶意代码，这使得战争还没打响之前就造成了伊拉克整个计算机网络管理的雷达预警系统全部瘫痪。这是美国第一次公开在

表 5-1　主要恶意代码的相关定义

恶意代码类型	相关定义说明	特　　点
计算机病毒	指编制或在计算机程序中插入的破坏计算机功能或毁坏数据、影响计算机使用，并能自我复制的一组计算机指令或程序代码	潜伏、传染和破坏
计算机蠕虫	指通过网络自我复制、消耗系统和网络资源的程序	扫描、攻击和扩散
特洛伊木马	指一种与连接远程计算机通过网络控制本地计算机的程序	欺骗、隐蔽和信息窃取
逻辑炸弹	指一段嵌入计算机系统程序的、通过特殊的数据或时间作为条件完成破坏功能的程序	潜伏和破坏
病菌	指不依赖于系统软件自我复制和传播，以消耗系统资源为目的的程序	传染和拒绝服务
用户级 RootKit	指通过替代或修改被执行的程序进入系统，从而实现隐藏和创建后门的程序	隐蔽、潜伏
核心级 RootKit	指嵌入操作系统内核进行隐藏和创建后门的程序	隐蔽、潜伏

实战中利用恶意代码攻击技术获得的重大军事利益。在 Internet 安全事件中，由恶意代码造成的经济损失居第一位。

5.1.3　恶意代码存在原因

计算机技术飞速发展的同时并未使系统的安全性得到增强。计算机技术的进步带来的安全增强能力最多只能对由应用环境的复杂性带来的安全威胁的增长程度进行一定的弥补。除此之外，计算机新技术的出现或许会让计算机系统的安全性降低。AT&T 实验室的 S. Bellovin 曾经对美国 CERT 提供的安全报告进行过分析，分析结果表明，大约 50％的计算机网络安全问题是由软件工程中产生的安全缺陷引起的，其中，很多问题的根源都来自操作系统的安全脆弱性。

互联网的飞速发展成为恶意代码广泛传播成长的温室。互联网自身的开放性，缺乏中心控制性和全局视图能力的特点，使得网络主机处在一个受不到统一保护的环境中。Bellovin 等认为计算机和网络系统自身存在的设计缺陷，会导致安全隐患的发生。

针对性是恶意代码的主要特征之一，即特定的脆弱点的针对性，这充分说明了恶意代码实现其恶意目的正是建立在软件的脆弱性基础上的。历史上产生广泛影响的 1988 年 Morris 蠕虫事件，入侵的最初突破点就是利用的邮件系统的脆弱性。

虽然人们做了诸多努力来保证系统和网络基础设施的安全，但令人遗憾的是，系统的脆弱性仍然无法避免。各种安全措施只能减少，但不能杜绝系统的脆弱性；而测试手段也只能证明系统存在脆弱性，却无法证明系统不存在脆弱性。更何况，为了实际需求的满足，信息系统规模的逐渐扩大，会使安全脆弱性的问题越来越明显。随着逐步发现这些脆弱性，针对这些脆弱性的新的恶意代码将会层出不穷。

总体来讲，许多不可避免的安全问题和安全脆弱性存在于信息系统的各个层次结构中，包括从底层的操作系统到上层的网络应用在内的各个层次。这些安全脆弱性的不可避免，也是恶意代码必然存在的原因。

5.1.4　恶意代码传播与发作

如今的信息社会，信息共享是大势所趋，而恶意代码入侵最常见的途径则是信息共享引

起的信息流动。恶意代码的入侵途径各种各样，如从 Internet 上下载的程序自身也许就带有恶意代码、接收已经感染恶意代码的电子邮件、从光盘或软盘向系统上安装携带恶意代码的软件、黑客或攻击者故意将恶意代码植入系统等。

通过用户执行该恶意代码或已被恶意感染的可执行代码，从而使得恶意代码得以执行，进而将自身或自身的变体植入其他可执行程序中，就造成了恶意代码感染。被执行的恶意代码在完成自身传播、具有足够的权限并满足某些条件时，就会发作同时进行破坏活动，造成信息丢失或泄密等。

恶意代码的入侵和发作需要在盗用系统或应用进程的合法权限基础上才可以实现。随着 Internet 开放程度越来越高，信息共享和交流也越来越强，恶意代码编写水平也越来越高，可被恶意代码利用的系统和网络的脆弱性也越来越多，从而使恶意代码越来越具有欺骗性和隐蔽性。重要的是，新的恶意代码总是在恶意代码的检测技术前出现，这也就是由 Cohen 和 Adelman 提出的“恶意代码通用检测方法的不可判定性”的结论。首先人们很难将正常代码和恶意代码区别开，其次对许多信息系统没有必要的保护措施。所以，人们经常被恶意代码所蒙蔽，从而在无意中执行了恶意代码。在 CERT 统计数据中，其中因被欺骗或误用从而发生的恶意代码事件达到所有恶意代码事件的 90%。只要某些条件被满足，恶意代码就会发作，甚至大规模传播。

虽然越来越多的恶意代码安全事件通过 Internet 发生，但是恶意代码却早就已经出现。多年来，攻击者致力于对具有更强攻击能力和生存能力的恶意代码的研究。

5.1.5 恶意代码攻击模型

恶意代码行为各不相同，破坏程度也不尽相同，但它们的作用机制基本大致相同，作用过程可大概分为 6 个部分。

① 侵入系统。恶意代码实现其恶意目的的第一步必然是入侵系统。恶意代码侵入系统有许多途径，如从互联网下载的程序其自身也许就带有恶意代码、接收了已被恶意感染的电子邮件、通过光盘或软盘在系统上安装的软件、攻击者故意植入系统的恶意代码等。

② 维持或提升已有的特权。恶意代码的传播与破坏需要建立在盗用用户或进程的权限合法的基础之上。

③ 隐蔽策略。为了隐蔽已经入侵的恶意代码，可能会对恶意代码进行改名、删除源文件或修改系统的安全策略。

④ 潜伏。恶意代码侵入系统后，在具有足够的权限并满足某些条件时，就会发作同时进行破坏活动。

⑤ 破坏。恶意代码具有破坏性的本质，为的是造成信息丢失、泄密，系统完整性被破坏等。

⑥ 重复①～⑤对新的目标实施攻击过程。恶意代码的攻击模型如图 5-1 所示。

恶意代码的攻击过程可以存在于恶意代码攻击模型中的部分或全部。例如，①④⑤⑥存在于计算机病毒行为，①②⑤⑥存在于网络蠕虫，①②③⑤存在于特洛伊木马，①④⑤存在于逻辑炸弹，①⑤⑥存在于病菌，①②③⑤存在于用户级 RootKit，①③⑤存在于核心级 RootKit，其他的恶意代码行为也可以映射到模型中的相应部分，其中①和⑤是必不可少的。

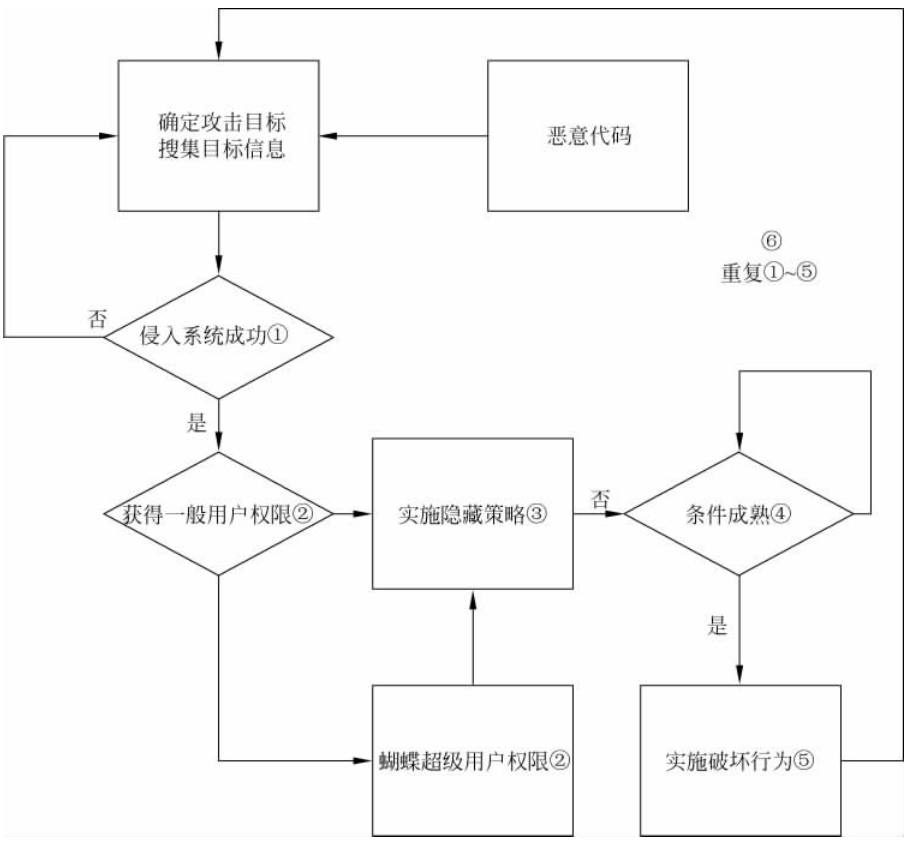

图 5-1　恶意代码的攻击模型

5.2　恶意代码生存技术

1. 反跟踪技术

恶意代码靠采用反跟踪技术来提高自身的伪装能力和防破译能力，使检测与清除恶意代码的难度大大增加。反跟踪技术大致可以分为两大类：反动态跟踪技术和反静态分析技术。

1）反动态跟踪技术

（1）禁止跟踪中断。恶意代码通过修改程序的入口地址对调试分析工具运行系统的单步中断与断点中断服务程序来实现其反跟踪的目的。

（2）封锁键盘输入和屏幕显示，使跟踪调试工具运行的必需环境被破坏。

（3）检测跟踪法。根据检测跟踪调试和正常执行二者的运行环境、中断入口和时间不同，各自采取相应的措施实现其反跟踪目的。

（4）其他技术，如指令流队列法和逆指令流法等。

2）反静态分析技术

（1）对程序代码分块加密执行。为了不让程序代码通过反汇编进行静态分析，将以分块的程序代码以密文形式装入内存，由解密程序在执行时进行译码，立即清除执行完毕后的

代码，力求分析者在任何时候都无法从内存中获得执行代码完整形式。

(2) 伪指令法。伪指令法是指将“废指令”插入指令流中，让静态反汇编得不到全部正常的指令，进而不能进行有效的静态分析。例如，Apparition 是一种基于编译器变形的 Win32 平台的病毒，每次新的病毒体可执行代码被编译器编译出时要被插入一定数量的伪指令，不仅使其变形，而且实现了反跟踪的目的。不仅如此，该技术还在宏病毒与脚本恶意代码中应用广泛。

2. 加密技术

加密技术是恶意代码进行自我保护的手段之一，再配合反跟踪技术的使用，让分析者不能正常分析调试恶意代码，无法获得工作原理，自然也不能抽取特征串。从加密的内容上划分，加密手段有 3 种，即信息加密、数据加密和程序代码加密。大部分恶意代码对程序体本身加密，但还是有少数恶意代码对被感染的文件加密。例如，Cascade 是 DOS 环境下采用加密技术的恶意代码的第一例。解密器稳定，能够使内存中加密的程序体被解密。Mad 和 Zombie 是 Cascade 延伸了加密技术，让恶意代码加密技术扩展到 32 位的操作系统平台。不仅如此，“中国炸弹”和“幽灵病毒”也是这一类恶意代码。

3. 模糊变换技术

在感染一个客体对象时，恶意代码利用模糊变换技术生成的潜入代码都不尽相同。尽管是同一种恶意代码但仍会具有很多不相同的样本，几乎不存在稳定代码，仅仅是基于特征的检测工具通常不能有效识别它们。随着恶意代码的不断发展，病毒检测和防御软件的代码编写越来越难，反病毒软件的误报率也随之增加。

目前，模糊变换技术主要分为以下几种。

(1) 指令替换技术。模糊变换引擎(Mutation Engine)对恶意代码的二进制代码反汇编，解码并计算指令长度，再对其同义变换。例如，指令“XOR REG，REG”被变换为“SUB REG，REG”；寄存器 REG1 和寄存器 REG2 互换；JMP 指令和 CALL 指令变换等。例如，“Regswap”使用了寄存器互换这一变形技术。

(2) 指令压缩技术。经恶意代码反汇编后的全部指令由模糊变换器检测，对可压缩的指令同义压缩。压缩技术要想使病毒体代码的长度发生改变，必须对病毒体内的跳转指令重定位。例如，指令“MOV REG，12345678 / PUSH REG”变换为指令“PUSH 12345678”等。

(3) 指令扩展技术。扩展技术对汇编指令同义扩展，所有经过压缩技术变换的指令都能够使用扩展技术来进行逆变换。扩展技术远比压缩技术可变换的空间要大，甚至指令能够进行几十或上百种的扩展变换。扩展技术也需要对恶意代码的长度进行改变，进行恶意代码中跳转指令的重定位。

(4) 伪指令技术。伪指令技术主要是将无效指令插入到恶意代码程序体，如空指令。

(5) 重编译技术。使用重编译技术，利用自带编译器或操作系统提供的编译器，将恶意代码重新编译为代码形态不同的新恶意代码，这种技术不仅实现了变形目的，而且为跨平台的实现提供了条件。这表现在 UNIX/Linux 操作系统，系统默认配置有标准 C 的编译器。宏病毒和脚本恶意代码是采用这类技术变形的代表性恶意代码。

Tequtla 是第一例在全球范围传播破坏的变形病毒，从其出现到研发出可以有效检测该病毒的软件，研究人员一共用了 9 个月。

模糊变换技术是恶意代码更好生存的有效方法之一，是如今研究恶意代码的关注点。

4. 自动生产技术

恶意代码自动生产技术基于人工分析技术。即使在这方面零基础的人也能利用“计算机病毒生成器”组合出算法、功能各不相同的计算机病毒。普通病毒能够利用“多态性发生器”编译成具有多态性的病毒。多态变换引擎能够让程序代码本身产生改变，但却可以保持原有功能。例如，保加利亚的 Dark Avenger，变换引擎每产生一个恶意代码，其程序体都会发生变化。反恶意代码软件若只是采用基于特征的扫描技术，则无法检测和清除这种恶意代码。

5. 变形技术

在恶意代码的查杀过程中，多数杀毒厂商通过提取恶意代码特征值的方式对恶意代码进行分辨。需要一个特征代码库是基于特征码的病毒查杀技术的致命缺点，同时这个库中的代码要具有固定性。病毒设计者利用这一漏洞，设计出具体同一功能不同特征码的恶意代码。将这种变换恶意代码特征码的技术称为变形技术。常见的恶意代码变形技术包括以下几种。

(1) 重汇编技术：变形引擎对病毒体的二进制代码进行反汇编，解码每一条指令，并对指令进行同义变换。例如，Regswap 就是采用简单的寄存器互换的变形。

(2) 压缩技术：变形器检测病毒体反汇编后的全部指令，对可进行压缩的一段指令进行同义压缩。

(3) 膨胀技术：压缩技术的逆变换就是对汇编指令同义膨胀。

(4) 伪指令技术：伪指令技术主要是对病毒体插入废指令，如空指令、跳转到下一指令和压弹栈等。

(5) 重编译技术：病毒体携带病毒体的源码，需要自带编译器或利用操作系统提供的编译器进行重新编译。这为跨平台的恶意代码出现打下了基础。

6. 三线程技术

恶意代码中应用三线程技术是为了防止恶意代码被外部操作停止运行。当一个恶意代码进程同时开启了 3 个线程，其中一个为负责远程控制的工作的主线程，另外两个为用来监视线程负责检查恶意代码程序是否被删除或被停止自启动的监视线程和守护线程，这是它的工作原理。注入其他可执行文件内的守护线程，同步于恶意代码进程。只要进程被停止，它就会重新启动该进程，同时向主线程提供必要的数据，这样就使得恶意代码可以持续运行。“中国黑客”就是采用这种技术的恶意代码。

7. 进程注入技术

在系统启动时，操作系统的系统服务和网络服务一般能够自动加载。恶意代码程序为了实现隐藏和启动的目的，把自身嵌入到与有关这些服务的进程中。这类恶意代码只需要安装一次，就能被服务器加载到系统中运行，并且可以一直处于活跃状态。Windows 下的大部分关键服务程序能够被 WinEggDropShell 注入。

8. 通信隐藏技术

实现恶意代码的通信隐藏技术一般有三类实现方法：端口定制技术、通信加密技术和

隐蔽通道技术。

(1) 端口定制技术。旧木马几乎都存在预设固定的监听端口,但是新木马一般都有定制端口的功能。

优点:木马检测工具的一种检测方法就是检测默认端口,定制端口可以避过此方法的检测。

端口复用技术利用系统网络打开的端口(如 25 和 139 等)传送数据。木马 Executor 用 80 端口传递控制信息和数据; Blade Runner、Doly Trojan、Fore、FTP trojan、Larva、ebEx、inCrash 等木马复用 21 端口; Shtrilitz Stealth、Terminator、WinPC、WinSpy 等木马复用 25 端口。使用端口复用技术的木马在保证端口默认服务正常工作的条件下复用,具有很强的欺骗性,可欺骗防火墙等安全设备,可避过 IDS 和安全扫描系统等安全工具。

(2) 通信加密技术,即将恶意代码的通信内容加密发送。通信加密技术胜在能够使得通信内容隐藏,弊端是无法隐藏通信状态。

(3) 隐蔽通道技术能有效隐藏通信内容和通信状态,目前常见的能提供隐蔽通道方式进行通信的后门有 BO2K、Code Red Ⅱ、Nimida 和 Covert TCP 等。但恶意代码编写者需要耗费大量时间以便找寻隐蔽通道。

9. 内核级隐藏技术

(1) LKM 隐藏。LKM,译为可加载内核模块,用来扩展 Linux 内核功能。LKM 能够在不用重新编译内核的情况下把动态加载到内存中。基于这个优点,LKM 技术经常使用在系统设备的驱动程序和 Rootkit 中。

LKM Rootkit 技术能通过系统提供的接口,将外部代码加载到内核空间,即将恶意程序转化为内核的某一部分,再通过 hook 系统调用的方式实现隐藏功能。

(2) 内存映射隐藏。内存映射是指由一个文件到一块内存的映射。文件映射使得可以将硬盘上的内容映射至内存中,用户可以通过内存指令读写文件。使用内存映射避免了多次调用 I/O 操作的行为,减少了不必要的资源浪费。

5.3 恶意代码攻击技术

1. 进程注入技术

当系统启动时,系统服务和网络服务在操作系统中被自动加载。进程注入技术就是将这些与服务相关的嵌入了恶意代码程序本身的可执行代码作为载体,实现自身隐藏和启动的目的。这类恶意代码只需要安装一次,就能被服务加载到系统中运行,并且可以一直处于活跃状态。

2. 超级管理技术

部分恶意代码采用超级管理技术对反恶意代码软件系统进行拒绝服务攻击,阻碍反恶意代码软件正常运行。例如,国产特洛伊木马“广外女生”,就是用这个技术成功攻击“金山毒霸”和“天网防火墙”的。

3. 端口反向连接技术

防火墙对于外网进入内部的数据流有严格的访问控制策略,但对于从内到外的数据并

没有严格控制。指令恶意代码攻击的服务端(被控制端)主动连接客户端(控制端)端口为反向连接技术。最早实现这项技术的木马程序是国外的 Boinet,它可以通过 ICO、IRC、HTTP 和反向主动连接这 4 种方式联系客户端。“网络神偷”是我国最早实现端口反向连接技术的恶意代码。“灰鸽子”则是这项技术的集大成者,它内置 FTP、域名、服务端主动连接这 3 种服务端在线通知功能。

4. 缓冲区溢出攻击技术

80%的远程网络攻击为缓冲区溢出漏洞攻击,能使 Internet 一个匿名用户有机会获得另一台主机的部分或全部的控制权。恶意代码利用系统和网络服务的安全漏洞植入并且执行攻击代码,攻击代码以一定的权限运行有缓冲区溢出漏洞的程序来获得被攻击主机的控制权。缓冲区溢出攻击成为恶意代码从被动式传播转为主动式传播的主要途径之一。例如,“红色代码”利用 IIS Server 上 Indexing Service 的缓冲区溢出漏洞完成攻击、传播和破坏等恶意目的。

5.4　恶意代码的分析技术方法

5.4.1　恶意代码分析技术方法概况

如图 5-2 所示,恶意代码的分析方法由静态分析方法和动态分析方法两部分构成。其中静态分析方法有反恶意代码软件的检测和分析、字符串分析和静态反编译分析等;动态分析包括文件监测、进程监测、注册表监测和动态反汇编分析等。

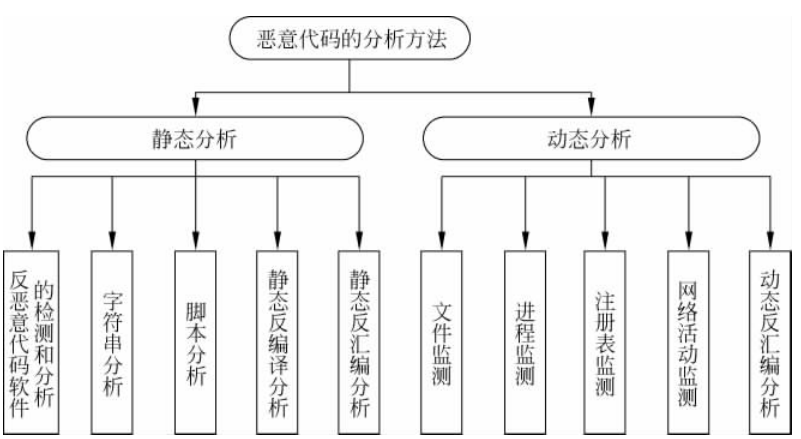

图 5-2　恶意代码的分析方法

5.4.2　静态分析技术方法

恶意代码的静态分析主要有以下几种方法。

(1) 反恶意代码软件的检测和分析。反恶意代码软件检测出恶意代码方法有特征代码法、校验和法、行为监测法、软件模拟法等。根据恶意代码的信息去搜寻更多的资料,若该恶意代码的分析数据已被反恶意代码软件收录,那就可以直接利用它们的分析结果。

(2) 字符串分析。字符串分析的目的是寻找文件中使用的 ASCII 或其他方法编码的连

续字符串。一些有用的信息可以通过在恶意代码样本中搜寻字符串得到，例如：

① 恶意代码的名称；

② 帮助和命令行选项；

③ 用户对话框，可以通过它分析恶意代码的目的；

④ 后门密码；

⑤ 恶意代码相关的网址；

⑥ 恶意代码作者或攻击者的 E-mail 地址；

⑦ 恶意代码用到的库，函数调用，以及其他的可执行文件；

⑧ 其他的有用的信息。

(3) 脚本分析。恶意代码是用 JavaScript、Perl 或 Shell 等脚本语言编写，那么恶意代码本身就可能带有源代码。通过文本编辑器将脚本打开查看源代码。脚本分析能帮助分析者用较短时间识别出大量流行的脚本类型，表 5-2 列出了常用的脚本语言。

表 5-2　常用脚本语言

脚 本 语 言	在文件中识别其特征	文件通常后缀
Bourne Shell	以！#/bin/sh 开始	.sh
Perl	以！#/usr/bin/perl 开始	.pl、.perl
JavaScript	以< Script language = "JavaScript">形式出现	.js、.html、.htm
VBScript	包含单词 VBScript 或在文件中散布着字符 vb	.vbs、.html、.htm

(4) 静态反编译分析。可以采用反编译工具来查看携带解释器的恶意源代码。在编译时，代码会被编译器优化，组成部分被重写来使得程序更适合解释和执行，这个特性使得编译的代码不适合逆向编译。因此，逆向编译是将对计算机优化的代码重新转化为源代码，这使得程序结构和流程分离开来，同时变量的名称由计算机自动生成，这使得逆向编译的代码有着较差的可读性。表 5-3 列出了一些反编译工具，它们能够生成被编译程序的 C 或 Java 语言的源代码。

表 5-3　反编译工具

工　　具	平　　台	概　　述	下 载 地 址
Reverse Engineering Compiler（REC）by Giampiero Caprino	SunOS，Linux，Windows	在 Windows、Linux、Bsd、Sunos 多种平台下将面向 x86、SPARC、68k、PowerPC 和 MIPS 等多种体协结构的处理器的代码逆向编译成 C 代码	www.backerstreet.com/rec/rec.htm
Dcc，by Cristina Cifuentes	运行于 UNIX 平台上，但是分析 Windows 的 exe 可执行文件	将 Windows 的面向 x86 体协结构写的程序逆向编译成 C 代码	www.itee.uq.edu.au/～cristina/dcc.html
JreversePro	用 Java 编写，这个工具可以在任何有 Java 虚拟机的系统上运行	将 Java 字节代码逆向编译成 Java 源代码	jrevpro.sourceforge.net/
HomeBrew Decompiler	UNIX 系统	逆向编译 Java 字节代码	www.pdr.cx/projects/hbd/

(5) 静态反汇编分析。有有线性遍历和递归遍历两种方法。GNU 程序 OBJDUMP 和一些链接优化工具使用线性遍历算法从输入程序的入口点开始反汇编,简单地遍历程序的整个代码区,反汇编它所遇到的每一条指令。虽然方法简单,但存在不能够处理嵌入到指令流中的数据问题,如跳转表。递归遍历算法试图用反汇编出来的控制流指令来指导反汇编过程,以此解决上面线性遍历所存在的问题。直观地说,无论何时反汇编器遇到一个分支或 CALL 指令,反汇编都从那条指令的可能的后继指令继续执行。很多的二进制传输和优化系统采用这种方法。正确判定间接控制转移的可能目标难度很大是存在的缺点。恶意代码被反汇编后,就可用控制流来分析构造它的流程图,该图又可以被许多的数据流分析工具所使用。由于控制流程图是大多数静态分析的基础,因此不正确的流程图反过来会使整个静态分析过程得到错误的结果。

5.4.3　动态分析技术方法

恶意代码的动态分析主要有以下几种方法。

(1) 文件监测。恶意代码在传播和破坏的过程中需要依赖读写文件系统,但存在极少数只是单纯依赖内存却没有与文件系统进行交互。恶意代码执行后,在目标主机上可能进行读写文件、修改程序、添加文件,甚至把代码嵌入其他文件。由此对文件系统必须要进行监测。FileMon 是常用的文件监测程序,能够记录与文件相关的动作,如打开、读取、写入、关闭、删除和存储时间戳等。另外,还有文件完整性监测工具,如 Trip wire、AIDE 等。

(2) 进程监测。恶意代码要入侵甚至传播,必须要有新的进程生成或盗用系统进程的合法权限,主机上所有被植入进程的细节都能为分析恶意代码提供重要参考信息。常用的进程监测工具是 Process Explorer,它将机器上的每一个执行中的程序显示出来,将每一个进程的工作详细展示出来。虽然 Windows 系统自己内嵌了一个进程展示工具,但是只显示了进程的名称和 CPU 占用率,这不足以了解进程的详细活动情况。而 Process Explorer 比任何的内嵌工具更有用,它可以看见文件、注册表键值和进程装载的全部动态链接库的情况。并且对每一个运行的进程,该工具还显示了进程的属主、优先级和环境变量。

(3) 网络活动监测。恶意代码从早期单一的传染形式到变成依赖网络传染的多种传染方式。因此分析恶意代码还要监测恶意代码的网络行为。使用网络嗅探器检测恶意代码传播的内容,当恶意代码在网络上发送包时,嗅探器就会将它们捕获。如果局域网是基于共享 Hub 的,那么可以连接嗅探计算机到任意端口开始嗅探;如果是基于交换机的,那么就要配置一个嗅探端口,将全部局域网的分组包引导到嗅探端口上去。常用的网络监测工具如表 5-4 所示。

表 5-4　网络监测工具

工具	平台	概　　述	下 载 地 址
TCPview	Linux、Windows	查看端口和线程	www. sysinternal. com
Fport	Windows	查看本机开放端口,以及端口和进程对应关系	www. foundstone. com
Nmap	Linux、Windows	开源的扫描工具,用于系统管理员查看一个大型的网络有哪些主机,以及其上运行哪种服务,支持多种协议的扫描	www. insercure. org
Nessus	Linux、Windows	Nessus 是一款经典的安全评估软件,功能强大且更新快,采用 C-S 模式,服务器端负责进行安全检查,客户端用来配置管理服务器端	

(4) 注册表监测。Windows 操作系统的注册表是包含了操作系统和大多数应用程序配置的层次数据库,恶意代码运行时一般要修改 Windows 操作系统的配置来改变 Windows 操作系统的行为,实现入侵目的。常用的监测软件是 Regmon,它能够实时显示读写注册表项的全部动作。

(5) 动态反汇编分析。动态反汇编是指在恶意代码的执行过程中对其进行监测和分析。其基本思想是将恶意代码运行的控制权交给动态调试工具。该监测过程从代码的入口点处开始,控制权在程序代码与调试工具之间来回传递,直到程序执行完为止。这种技术能得到正确的反汇编代码,但只能对程序中那些实际执行的部分有效。

目前主要动态反汇编分析方法有以下两种。

① 同内存调试。这种方法使调试工具与被分析的恶意代码程序加载到相同的地址空间里。该方法的优点是实现代价相对较低,控制权转交到调试工具或从调试工具转回恶意代码程序的实现相对来说比较简单;缺点是需要改变被分析程序的地址。

② 仿真调试,即虚拟调试。这种方法是让调试工具与分析的恶意代码程序处于不同的地址空间,可绕过很多传统动态反跟踪类技术。这种方法的优点是不用修改目标程序中的地址,但在进程中控制权的转移上要付出较高的代价。

表 5-5 列出了 Windows 平台常用的调试工具。

表 5-5 Windows 平台常用的调试工具

调试器工具	概　　述	下 载 地 址
Ollydbg	免费调试器,图形界面,功能强大	http://home.t-online.de/home/Ollydbg
IDA Pro	一个主要的调试器和代码分析工具,简化版本可以免费得到	http://www.datarescue.com
SoftICE	商业软件,提供优秀的调试功能和 GUI 界面。支持源代码和二进制调试	www.compuware.com/products/devpartner/softice.htm

5.4.4 两种分析技术比较

恶意代码程序是静态分析的中心,而行为是动态分析的中心。静态分析无关于恶意代码的行为,只通过恶意代码自身来判断恶意代码想要实现的目标。动态分析是依赖于恶意代码的运行环境和不同的监测目标。不同的环境和不同的目标可能得到不同的动态分析结果。动态分析对环境和目标的依赖导致动态分析的不完全,但同时也将恶意代码的运行环境、监测目标和程序行为紧密地联系起来。通过动态分析可以看到,运行环境的变化直接引起了恶意代码的内部行为和监测结果的变化。

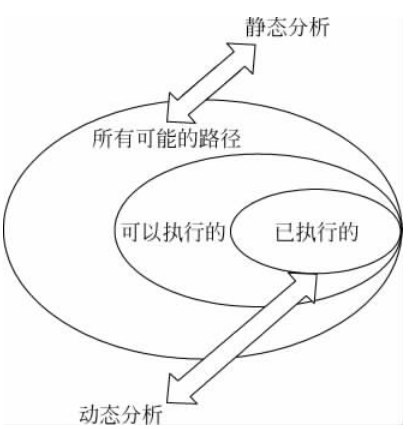

图 5-3 动静态分析范围示意图

如图 5-3 所示,静态分析根据代码内容推出执行的所有特性,而动态分析是恶意代码一次执行或多次执行得到的特性。动态分析不能证明代码一定满足某个特定的属性,但是可以检测到异常的属性,还可

以提供关于恶意代码程序行为的有用信息。静态分析或许会得到很多冗余信息，分析结果会被冗余信息所迷惑。动态分析可以确切地、有针对性地得到分析所需要的具体数据。动态分析和静态分析的关系如图 5-3 所示。

由图 5-3 中可以看出，静态分析和动态分析其实是对恶意代码所有可能执行的子集的不同选择。一个准确的静态分析要考虑到恶意代码的每一种可能的执行情况，即每一次执行时恶意代码的全部可能状态。当恶意代码的代码量增加到一定程度时，静态分析要对所有状态进行分析就不可行了，因为分析代码的可能执行状态的代价已不能接受。静态分析和动态分析不是对立的技术。虽然静态分析和动态分析有很大差别，应用于同一恶意代码时代价不同，结果也不同，但静态分析和动态分析可以使用相同的分析手段。静态分析和动态分析使用在不同阶段，能够互相补充支持。恶意代码的程序和行为之间有着强烈的依赖关系，决定了动态分析和静态分析相结合能达到更好的分析效果。不同的分析可以收集到不同的有用信息，对恶意代码分析应先执行静态分析后再执行动态分析，比单独地执行任一种分析更有效。静态分析能够规避动态分析收集的信息充分性差的缺点，而动态分析能够收集针对性强的少量信息。

5.5　典型恶意代码攻击与防范

5.5.1　典型计算机病毒攻击与防范

1. 计算机病毒攻击

长期以来，人们设计计算机忽略了安全问题。计算机系统的脆弱性，为计算机病毒的产生和传播提供了可能。万维网使地球一体化，为计算机病毒的传播提供了可能的空间。新的计算机技术不断发展应用，为计算机病毒提供了一定的条件。国外专家认为，分布式数字处理、可重编程嵌入计算机、网络化通信的信息格式等都为计算机病毒侵入提供了可能。

计算机病毒入侵技术和病毒有效注入方法研究的内容相近。从目前现状来看，病毒注入方法主要有以下几种。

(1) 无线电方式。主要是将病毒码通过无线电的方式发射到对方电子系统。这是计算机病毒注入的最佳方式，但是技术实现最难。

(2) 固化式方法。即把事先存放有病毒的硬件或软件直接或间接交给对方，该病毒直接将对方的电子系统传染，再在需要时将其激活来进行攻击。这种攻击方法过于隐蔽，即使彻查也不能完全排除是否有其他特殊功能。目前，我国还需要从国外进口很多计算机组件，故芯片攻击不能完全避免。

(3) 后门攻击方式。后门是由软件设计师或维护人员设计发明的，是一个允许知道后门存在的人绕过正常安全防护措施进入系统的漏洞。计算机入侵者常常通过后门来采取攻击，如 Windows98 就存在后门。

(4) 数据控制链侵入方式。随着网络的发展，计算机病毒可以通过数据控制链侵入计算机系统。利用远程修改技术可以很直接方便地改变数据控制链的正常路径。

2. 对计算机病毒攻击的防范对策和方法

(1) 建立有效的计算机病毒防护体系。有效的计算机病毒防护体系应包括多个防护

层：①访问控制层；②病毒检测层；③病毒遏制层；④病毒清除层；⑤系统恢复层；⑥应急计划层。

（2）严把硬件安全关。涉及国家机密系统的设备及产品，应该尽量使用国产；对引进的计算机系统要在进行安全性检查后才能使用，避免和控制计算机病毒伺机入侵。

（3）防止电磁辐射和电磁泄露。使用电磁屏蔽的方法阻断电磁波辐射，同时还能够有效避免“电磁辐射式”病毒的攻击。

（4）加强计算机应急反应分队建设。成立自动化系统安全支援分队，以解决计算机防御性的有关问题。

3. ARP 病毒攻击与防范

（1）ARP 协议工作原理。数据包在以太网中传输的形式是根据其首部的 MAC 地址来进行寻址的以太包。发送方必须知道目的主机的 MAC 地址才能向其发送数据。ARP 协议把 32b 的 IP 逻辑地址转换为 48b 的以太网物理地址。

因为每个主机上都有一个 ARP 高速缓存，所以 ARP 才能够高效运行。最近 IP 地址到 MAC 地址之间的映射记录被存放在这个高速缓存中，存放时间一般为 20 分钟。

人们可以用 ARP 命令来检查 ARP 高速缓存。参数-a 的意思是显示高速缓存中所有的内容，如图 5-4 所示。

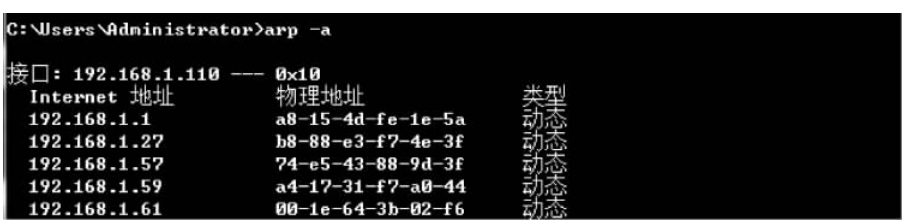

图 5-4　用 ARP 命令来检查 ARP 高速缓存

在每台安装有 TCP/IP 协议的计算机中都有一个 IP 地址与 MAC 地址对应的 ARP 缓存表。

（2）ARP 协议安全漏洞。ARP 是建立在相互信任的各个主机之间的局域网协议，是存在安全漏洞的。

① 主机地址映射表基于高速缓存及动态更新。恶意用户如果在下次交换前 MAC 地址刷新时限内成功地修改了被欺骗计算机上的地址缓存，就有可能发生假冒或拒绝服务攻击。

② ARP 协议是一个无状态的协议。接收到 ARP 应答帧，主机就会更新本地的 ARP 缓存，不要求主机必须先发送 ARP 请求后才能接收 ARP 应答，而是直接把应答帧中的 IP 地址和 MAC 地址存储在 ARP 高速缓存中。

③ 在通信中，ARP 缓存的优先级最高。

上述缺陷很容易被利用伪造 IP 地址进行 ARP 欺骗。

（3）ARP 欺骗病毒。假定在一个局域网中有 3 台计算机，它们的 IP 地址和 MAC 地址分别如下：

主机 A 的 IP 地址为 192.168.1.10，网卡地址为 01-0e-2d-73-65-17；

主机 B 的 IP 地址为 192.168.1.20，网卡地址为 00-1f-6d-c3-32-04；

主机 C 的 IP 地址为 192.168.1.30，网卡地址为 03-cc-4e-d5-1a-27。

如果主机 C 想窃听主机 A 与主机 B 通信，可利用 ARP 协议的漏洞，冒用主机 B 的名义与主机 A 通信，同时冒用主机 A 的名义与主机 B 通信。

主机 C 向主机 A 发送 ARP 应答报文，在应答报文中将主机 B 的 IP 地址 192.168.0.20 与主机 C 的网卡地址对应，当主机 A 收到 ARP 应答报文时，更新 ARP 高速缓存，增加“192.168.0.20 < - > 03-cc-4e-d5-1a-27”项。同时，主机 C 向主机 B 发送 ARP 应答报文，在响应报文中将主机 A 的 IP 地址 192.168.0.10 与主机 C 的网卡地址对应，当主机 B 收到 ARP 应答报文时，更新 ARP 高速缓存，增加“192.168.0.10 < - > 03-cc-4e-d5-1a-27”项。这样，当主机 A 再与主机 B 通信时，其数据包会发送到网卡地址为 03-cc-4e-d5-1a-27 的主机 C 上；当主机 B 与主机 A 通信时，数据包也会发送到网卡地址为 03-cc-4e-d5-1a-27 的主机 C 上。如果主机 C 能够实现自动路由转发，就可以在不影响主机 A 与主机 B 之间通信的前提下进行数据窃听了。

但如果前面主机 C 发给主机 A 的 ARP 更新包中的 MAC 地址不是自己的，而是伪造的根本不存在 MAC 地址，那么这时主机 A 和主机 B 之间就不可能再正常通信了，这就是 ARP 病毒对 PC 在网络间通信造成数据被窃听或网络不通的严重后果。

局域网通信发生在主机与主机之间，要与外网的主机通信，需要网关或路由器（一般局域网的路由器可直接充当网关的角色）。如果局域网内的某台主机 A 想要与外网的主机通信，那么在封装数据包时，目标 MAC 地址需要写成网关的 MAC 地址，网关再进行转发，发到网外去。如果这台主机 A 使用 ARP 数据包请求网关的 MAC 地址时，出现一台另有目的的主机向主机 A 回应了一个 ARP 应答报文，数据包就将这台病毒主机的 MAC 地址或根本不存在的 MAC 地址告诉主机 A，这时主机 A 发给远程网络的数据由于经过另有目的的主机造成数据被窃听或错误的 MAC 地址，而最终没有网关对数据进行转发，导致与外网不能正常通信。所以 ARP 病毒也能影响局域网与外网的通信。

（4）ARP 病毒防范。

① 使用静态 ARP 表。停止使用地址动态绑定和 ARP 高速缓存定期更新的策略。在 ARP 高速缓存中保存永久的 IP 地址与硬件地址映射表，允许由系统管理人员进行人工修改。

② 受托主机的永久条目放置于路由器的 ARP 高速缓存中，能有效地减少 ARP 欺骗。

③ 会话加密。不应把网络安全信任关系建立在 IP 地址或硬件 MAC 地址的基础上，而是应该对所传输的重要数据事先进行加密，再开始传输。

④ 使用 ARP 服务器。在确保这台 ARP 服务器不被黑客攻击的情况下通过该服务器查找 ARP 转换表来响应其他机器的 ARP 广播。

5.5.2　典型网络蠕虫攻击与防范

网络蠕虫暴发后，造成了巨大经济损失。通过研究分析所利用的系统漏洞和攻击手段，进行特征分析，及时补救，消除主机上的蠕虫体，为未来检测防范该蠕虫提供有效信息。

1. 网络蠕虫攻击模式

（1）对电子邮件的攻击。这种内部含有自动搜索邮件服务器和地址的机制蠕虫程序，

利用本地邮件客户端的漏洞，将寄发病毒邮件给搜索到的邮件地址。这类蠕虫有很多，其中的代表有：Netsky.C，D，Q，P（网络天空及其变种）；Beagle.B，C，F，I，H，M，N，Q，U，X，Z（恶鹰蠕虫及其变种）等。

（2）对于操作系统漏洞的攻击。系统漏洞蠕虫一般具备一个小型的溢出系统，它随机产生IP并尝试溢出，之后复制自身。被感染的系统性能速度会快速降低，甚至崩溃。一般此类蠕虫都是针对微软的系统漏洞发起攻击，若最新补丁没有及时安装，便会感染。

（3）对于文件共享服务的攻击。对于目前流行的P2P系统，网络用户之间可以分享彼此计算机中的文件，此类蠕虫就是利用这一服务，将自己隐藏在共享目录下，通过伪装成一个常用软件使其他用户下载并执行。

2. 冲击波蠕虫、震荡波蠕虫

冲击波蠕虫和震荡波蠕虫使用的都是常用的扫描策略对目标主机特定端口进行大量的连接尝试，连接成功后蠕虫程序会在目标主机和被感染主机之间建立连接，传送蠕虫副本。

（1）蠕虫特征。为了便于分析，冲击波蠕虫和震荡波蠕虫的特征如表5-6所示。

表5-6　冲击波蠕虫和震荡波蠕虫的特征

蠕虫名称	扫描策略	攻击手段	扫描端口
冲击波蠕虫	顺序扫描	RPC缓冲区溢出	TCP135端口
震荡波蠕虫	随机扫描	LSASS缓冲区溢出	445端口

（2）防治策略。冲击波蠕虫和震荡波蠕虫的扫描策略比较简单，在扫描网络时没有对目标IP地址做限定，常常会扫描一些尚未被分配的IP地址空间，被称为网络黑洞。可以通过对与网络黑洞相关的数据包进行监控和分析及时发现攻击行为。

只凭借对网络黑洞的扫描很难立即判定此行为就是网络蠕虫发起的攻击行为。由于每一种网络蠕虫都有各自明显的特征，在发现可疑行为后，不应急于对它定性，而应进一步进行特征匹配操作。一般网络蠕虫在攻击成功后，会向被攻击的计算机传送自身的可执行文件。基于特征匹配为这些文件建立特征，可以及时发现计算机中是否存在相应的网络蠕虫个体。

3. “熊猫烧香”蠕虫

（1）蠕虫特征。“熊猫烧香”蠕虫一个重要特征就是在感染主机的时，会在磁盘中产生大量的destop_.ini系统只读文件。“熊猫烧香”蠕虫还会窃取目标主机经常使用的邮箱地址和密码，用来发送包含蠕虫代码的邮件以此进行传播。它还可以通过用户在站点下载感染文件来传播。

（2）防治策略。利用特征匹配和网络黑洞等手段来制定监测防范策略。

获得系统控制权是蠕虫入侵的前提。“熊猫烧香”蠕虫在局域网内可以通过弱口令漏洞进行传播，一旦成功就会控制目标主机开始新一轮攻击。因此，计算机用户应该尽量避免用户名默认或密码为空，以增强主机的安全性。同时，如果能够合理控制程序访问系统客体的操作，则程序对系统的危害也将被限制。通过安全操作系统的强制存取控制机制可以将计算机系统划分为系统管理空间、用户空间和保护空间。强制存取控制机制将系统用户划分为普通用户和系统管理员。系统管理空间不可以被普通用户读写，用户空间的应用程序和

数据用户可以进行读写，普通用户对保护空间的程序和数据只可读不可写。从而限制了“网络蠕虫”的传播。

5.5.3　典型特洛伊木马攻击与防范

1. 木马攻击技术

在设计木马时，必须考虑的因素有以下几点，首先隐蔽性要好，其次要顺利实现客户端与服务器端的通信，最后还要有其他需要的功能。综上所述，木马的设计者重点会采用以下技术。

(1) 隐藏技术。木马的服务器为了防止发现端要进行隐藏。早期隐藏技术相对简单。从在任务栏目里隐藏程序到现在采用了内核插入式的嵌入方式，利用远程插入线程技术，嵌入 DLL 线程，或者挂接 PSAPI 等隐藏技术，实现木马的隐藏，甚至在 Windwos NT/2000 环境下，都能达到良好的隐藏效果。

木马有伪隐藏和真隐藏两种实现隐藏的方式。伪隐藏是指让仍然存在的程序进程消失在进程列表中，把木马服务器端的程序注册为一个服务就可以奏效。但是只适用 Windows 9x 系统。在 Windows NT/2000 中，则可以采用 API 的拦截技术，建立一个后台的系统钩子，拦截 PSAPI 的 EnumProcessM 等相关函数从而来实现对进程和服务的遍历调用控制，当检测到 PID 为木马的服务器端进程的时候进行直接跳过，实现隐藏。真隐藏是指木马的服务器程序运行后，不产生新的进程和服务，而是完全融进内核。真隐藏的方法一般采用以下方法。

① 远程线程插入技术：将要实现的功能程序做成一个的线程，在运行时自动插入到进程中。它使得程序不以进程或服务的方式工作进而彻底消失。

② 动态链接库注入技术(DLL 注入技术)：一个动态链接库文件的“木马”程序，通过使用远程插入技术，将其做成加载语句插入到目标进程中去，并将调用动态链接库函数的语句插入到目标进程。

③ HokoingAPI 技术：通过修改 API 函数的入口地址的方法来欺骗试图列举本地所有进程的程序。

(2) 自加载技术。木马首先要隐藏，其次就是启动。木马的设计者希望木马随着被植入的计算机每次启动时都能自动运行，故使用各种方法来实现自加载运行。

木马自运行的常见方法有：加载程序到启动组；将程序启动路径写到注册表的 HKEY_LOCAL_MACHINE/SOFTlrARE/Microsoft/Windows/CurrentVersions/Run 子键(以及 RunOnce、RunService、RunOnceService 等)；修改 Boot. ini；通过注册表中的输入法键值直接挂接启动；修改 Explorer. exe 启动参数及在 win. ini 和 systenL . ini 中的 load 节中添加启动项；在 Autoexec. bat 中添加程序等；或者采用文件关联实现木马的启动(冰河木马)；也可利用 DLL 木马替换系统原有的动态链接库，使系统在装载这些链接库时启动木马(GINA 木马)。

(3) 反向连接技术。反向连接和正向连接在本质上的区别并不大。在正向连接的情况下，服务器端就是被控制端，在编程实现的时候是采用服务器端的编程方法的，而控制端是采用客户端的编程方法。当采用反向连接技术编程时，就是服务器端采用客户端的编程方法，而将客户端变成了采用服务器端的编程方法。

防火墙一般对于连入的链接会严格过滤，但对于连出的却不会严格防范。反弹端口型木马采用反向连接技术，服务器端（被控制端）用主动端口，客户端（控制端）用被动端口。被植入反弹木马服务器端的计算机定时监测客户端的存在，发现客户端上线立即弹出端口主动连接客户端打开的主动端口。

（4）端口复用技术。在 Winsock 的实现中，对于服务器的绑定是可以多重绑定的，原则上是谁的指定最明确则将包递交给谁，且没有权限之分。

① 一个木马绑定到一个合法存在的端口上进行隐藏，通过特定的包格式判断如果是自己的包则处理，否则通过 127.0.0.1 的地址交给真正的服务器应用进行处理。

② 一个木马在低权限用户上绑定高权限的服务应用的端口进行嗅探。

（5）数据传输技术。木马常用 TCP、UDP 协议进行传递数据，但隐蔽性比较差，容易查到。但是可以采用以下方法躲避这种侦察。

一种方法是将木马通过连接绑定了通信的通用端口上发送信息。缺点是木马在等待和运行的过程中始终由一个和外界联系的端口打开。

另一种办法是使用 ICMP 协议。ICMP 报文由系统内核或进程直接处理而不通过端口，防火墙一般不会对 ICMP_ECHOREPLY 报文进行过滤，否则主机无法对外进行 ping 等路由诊断操作。

2. 木马防范技术

目前防范木马的手段有两种，依靠杀毒软件和网络防火墙。杀毒软件主要依靠木马特征和修改行为特征来识别木马，而防火墙软件主要通过对网络通信的控制实现对木马通信的封锁。目前我国市场的杀毒软件主要为瑞星、江民、金山。木马同病毒一样具备隐蔽性、非授权性及危害性等特征，因此，常常把木马的简单检测和清除等功能集成到系统中。

（1）杀毒软件技术特点。

① 未知病毒防治技术纵深发展。反病毒企业采取智能行为判断技术和启发式查毒技术研制出了未知病毒查杀技术。

② 病毒防护体系日趋完备。病毒防护体系就是通常提到的实时监控系统，目前病毒防护体系为脚本、内存、邮件、文件多种监控协同工作，大大增强预防病毒的能力。

③ 立体防毒成为病毒防护新标准。单一的病毒防治手段已不能满足用户的防毒需求，因此出现了立体防病毒体系，将计算机的使用过程进行逐层分解，对每一层进行分别控制和管理，从而达到病毒整体防护的效果。

（2）防火墙软件技术特点。防火墙具有较强的抗攻击能力。防火墙有以下几个阶段。

① 包过滤技术。包过滤防火墙工作在网络层对数据包的源及目的 IP 具有识别和控制作用，在传输层识别数据包是 TCP 还是 UDP 及所用的端口信息。目前的路由器、一些交换机和操作系统都已经具有包过滤控制的能力。

由于只是分析数据包的 IP 地址、TCP/UDP 协议、端口，包过滤防火墙的处理速度相对较快，并且易于配置。但对反向连接型木马的阻断则没有什么效果。

② 应用代理网关技术。应用代理网关防火墙彻底将内网与外网的直接通信隔断，内网用户对外网的访问转换为防火墙对外网的访问，结果由外网传到防火墙再转发给内网用户。所有通信都必须经应用层代理软件转发。

③ 状态监测技术。数据包并不是独立的，而是前后之间有着密切的状态联系，故产生

了状态监测技术。

状态监测防火墙摒弃了包过滤防火墙仅考查数据包的 IP 地址等几个参数，而不关心数据包连接状态变化的缺点，在防火墙的核心部分建立状态连接表，并将进出网络的数据当成一个个的会话，利用状态表跟踪每一个会话状态。状态监测对每一个包的检查不仅根据规则表，还包括了数据包是否符合会话所处的状态，因此提供完整的对传输层的控制能力。

④ 个人防火墙。个人防火墙采用了一种面向应用程序的过滤技术，这种技术对采用反弹端口技术的木马有一定的效用。但随着线程注入式木马的出现，访问网络时防火墙认为是宿主程序访问网络，一般不予阻拦。

(3) “系统级深度防护和立体联动防毒”技术。江民公司推出新的防杀病毒的技术，即“系统级深度防护和立体联动防毒”技术，由三大技术来应对，即“驱动级编程技术”“系统级深度防护技术”“立体联动防杀技术”，用户在杀毒软件上可以自动识别是否安装了江民黑客防火墙，并可控制防火墙的开启、关闭和设置安全级别，实现立体联动防毒功能，在遇到混合型病毒后，防火墙规则库、杀毒软件病毒库同步升级，彻底防范类似“冲击波”“震荡波”等病毒的攻击。

5.5.4　典型 Rootkit 攻击与防范

1. Rootkit 的定义

Rootkit 的概念出现于 20 世纪 90 年代初，是用来保持系统最高权限的工具集。

在安装 Rootkit 之前，必须先获取目标系统的超级用户权限。Rootkit 并不能帮助攻击者攻破系统，而是使攻击者能重新获取系统超级用户权限的技术。目前存在监听网络数据、破解密码、窃取管理员密码等超级用户访问权限的方法。

Rootkit 提供的主要功能如下。

(1) 保持对系统的访问权限：通过预留后门来保持访问。

(2) 攻击其他系统：出现了用于攻击其他系统的本地攻击工具和远程攻击工具两种。本地攻击工具是重新获取主机的管理员权限，代表有本地密码嗅探器和解密器；远程攻击工具是将目标主机作为一个跳板，攻击网络上的其他主机。

(3) 隐藏攻击痕迹：Rootkit 应具备隐藏攻击信息的功能，能修改或删除日志文件、隐藏相关文件、进程、通信链接等。

2. Rootkit 的攻击过程

为了对 Rootkit 的攻击过程有更深入的了解，下面将描述 Rootkit 攻击系统的几个关键步骤。

(1) 收集目标主机的信息。首先，对目标主机进行自动扫描，收集信息。例如，分析主机安装的系统、能否匿名登录、能否用 Telnet 连接等，从而发现漏洞。

(2) 获取目标主机的超级权限。利用漏洞，采取缓冲区溢出等手段获取超级用户访问权限。

(3) 在目标主机上安装 Rootkit。攻击者获取 Root 权限后，可通过隐蔽的网络端口将 Rootkit 工具包加载到目标主机中。还可不断将计算机病毒、键盘记录器等工具经由同样端口上传至目标主机。最后攻击者只需要执行 Rootkit 安装脚本，所有工具的安装就能完成。

(4) 清除 Rootkit 痕迹。清除 Rootkit 痕迹是 Rootkit 任务的关键,完成 Rootkit 安装后,通过删除文件、日志等方式清除入侵信息。

(5) 操纵目标主机。攻击者在目标主机上成功安装 Rootkit,就能在管理员毫无察觉的情况下长期控制系统,可以操纵这台目标主机执行病毒传播、拒绝服务攻击等非法活动。

3. Rootkit 的分类

根据对操作系统攻击对象所处位置的不同,可将 Rootkit 分为以下几类,如图 5-5 所示。

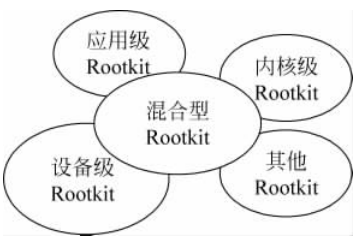

图 5-5 Rootkit 分类示意图

(1) 应用级 Rootkit 又称为传统 Rootkit,是最早出现的一类 Rootkit。

入侵者在目标主机上安装了恶意软件后,必定会在目标主机上预留网络端口或启动非法进程,将用户层的系统工具,如/bin/ls、/bin/ps、/bin/netstat 替换成恶意的程序,将与 Rootkit 相关的所有信息过滤后再显示。

典型的应用级 Rootkit 有 TOrn、WOOtkit、lion 蠕虫、lrk rootkit 等,应用级 Rootkit 技术已逐渐不被采用。

(2) 内核级 Rootkit 是指入侵到操作系统内核层的的恶意软件。操作系统内核处于整个系统的最底层,被认为是系统中最基本的部分,像文件系统、进程调度、存储管理、系统调用等任务都是在系统内核中实现的。内核级 Rootkit 通常修改或替换内核中的数据结构或函数(如中断处理函数、系统调用、文件系统等)。

目前最普遍的攻击目标是系统调用,按照实现攻击手段,可以把系统调用的 Rootkit 分为以下几类:重定向系统调用表的 Rootkit、修改系统调用表的 Rootkit、修改系统调用的 Rootkit。

典型的内核级 Rootkit 有 knark、adore、sucKIT、zk 等,虽同属于内核层的 Rootkit,但这些 Rootkit 的篡改技术并不尽相同。

(3) 设备级 Rootkit 是一种以计算机设备为目标的攻击技术,这些设备包括 BIOS、网卡、声卡、硬盘控制器等。所有这些设备都能通过程序与系统进行交互,因此一旦这些设备安装了 Rootkit,入侵者就很有可能通过它们干扰系统的正常工作。

(4) 另外存在一些 Rootkit,如安装在虚拟机监控器下的 Rootkit,其攻击对象不属于应用层、内核层或设备层。

(5) 同一个 Rootkit 的攻击目标有可能存在于多个层。美国乔治理工大学搭建的蜜罐就曾捕获到一种名为 r.tgz 的混合型 Rootkit,它包含了应用层和内核层 Rootkit。

4. Rootkit 防范

在 Linux 下防范 Rootkit 最有效的方法是定期对重要系统文件的完整性进行核查,常用的核查工具有 Zeppoo、Rootkit Hunter 和 Chkrootkit。

其中,Zeppoo 可以检测隐藏的系统任务、模块、syscalls、恶意符号和隐藏的连接。让 Linux 系统管理员根据 Zeppoo 发现的隐藏的、非法的程序来及时判断是否 Linux 系统中存在 Rootkits。

Rootkit Hunter 工具主要执行以下测试。

(1) MD5 校验测试,检测是否存在改动过的文件。

(2) 对 Rootkits 使用的二进制和系统工具文件进行检测。

(3) 对特洛伊程序的特征码进行检测。

(4) 对大多常用程序的文件异常属性进行检测。

(5) 对系统进行相关的测试。

(6) 对混杂模式下的接口和后门程序常用的端口进行扫描。

(7) 对配置、日志文件及隐藏文件等进行检测。

(8) 对常用端口应用程序进行版本测试。

完成上面的检测后,屏幕即可显示扫描结果:可能被感染的文件、不正确的 MD5 校验文件和易被感染的应用程序。

除了与 Rootkit Hunter 相同的测试外,Chkrootkit 还对一些重要的二进制文件进行检测,如搜索入侵者已更改的日志文件的特征信息等。

5.6　本章小结

恶意代码是信息系统安全的重要威胁之一。本章首先对恶意代码的定义、恶意代码存在的原因、传播途径等进行了分析,同时对恶意代码的攻击行为进行了刻画,给出了恶意代码的攻击模型。然后总结分析了目前恶意代码实现的关键技术,如生存技术、攻击技术、模糊变化技术、隐蔽技术等。接下来本章给出了恶意代码的分析技术方法体系,对静态分析和动态分析两种方法进行详细介绍。最后描述了典型恶意代码的攻击与防范技术。

习题 5

1. 关于网页中的恶意代码,下列说法错误的是(　　)。
   A. 网页中的恶意代码只能通过 IE 浏览器发挥作用
   B. 网页中的恶意代码可以修改系统注册表
   C. 网页中的恶意代码可以修改系统文件
   D. 网页中的恶意代码可以窃取用户的机密性文件
2. 以下对木马阐述不正确的是(　　)。
   A. 木马可以自我复制和传播
   B. 有些木马可以查看目标主机的屏幕
   C. 有些木马可以对目标主机上的文件进行任意操作
   D. 木马是一种恶意程序,它们在宿主主机上运行,在用户毫无察觉的情况下,让攻击者获得了远程访问和控制系统的权限
3. 计算机病毒是(　　)。
   A. 被损坏的程序　　B. 硬件故障
   C. 一段特制的程序　　D. 芯片霉变
4. 木马程序的最大危害在于它(　　)。
   A. 记录键盘信息　　B. 窃取用户信息
   C. 破坏软硬件系统　　D. 阻塞网络

5. 计算机病毒的特点是(　　)。

A. 传播性、潜伏性、破坏性　　B. 传播性、破坏性、易读性

C. 潜伏性、破坏性、易读性　　D. 传播性、潜伏性、安全性

6. 通过 Java Script、Applet、ActiveX(三者选一)编辑的脚本程序修改 IE 浏览器：默认主页被修改；IE 标题栏被添加非法信息。

7. 编写一个脚本病毒，扫描是否存在 U 盘，如果存在，将病毒写到 U 盘上。