信息安全丛书

木马核心技术剖析

Deep Dive into Key Technologies of Trojans

孙钦东 著

科学出版社

木马核心技术剖析

孙钦东　著

科学出版社

北　京

内 容 简 介

木马在网络安全中扮演极其重要的角色，对其进行深入透彻的分析是网络攻防中的重要内容。本书从 Windows 系统启动、存储、文件管理、模块管理、通信管理等核心机制入手，深度分析木马核心技术所依托的系统底层基础，详细剖析木马隐藏、驱动加载与启动、Windows 系统的安全机制绕过、防火墙穿透通信、反病毒软件免杀、反分析、Windows 64 位系统兼容等核心技术的原理与实现机制，给出了一个综合运用各项核心技术、高度可配置的木马框架。

本书是木马技术分析与检测方面的参考书，可作为从事网络安全等研究领域科研人员的参考书，也可作为高等院校网络安全专业本科生与研究生的参考教材。

图书在版编目（CIP）数据

木马核心技术剖析/孙钦东著. —北京：科学出版社，2016

ISBN 978-7-03-049932-5

I. ①木… II. ①孙… III. ①计算机病毒-防治 IV. ①TP309.5

中国版本图书馆 CIP 数据核字（2016）第 222230 号

责任编辑：赵丽欣 / 责任校对：王万红

责任印制：吕春珉 / 封面设计：东方人华平面设计部

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

三河市骏杰印刷有限公司印刷

科学出版社发行 各地新华书店经销

*

2016 年 11 月第 一 版 开本：787×1092 1/16

2016 年 11 月第一次印刷 印张：12 1/2

字数：318 000

定价：60.00 元

（如有印装质量问题，我社负责调换〈骏杰〉）

销售部电话 010-62136230 编辑部电话 010-62134021

前　言

木马在不被察觉的情况下植入并常驻目标主机，时刻监控目标主机的活动，窃取目标主机上的文件、邮件、浏览器记录等各类目标信息。木马因其隐蔽性强、功能范围广等特点，成为网络攻击中重要的一环。小至窃取信息，大至攻击基础网络环境和设备设施，木马在网络安全中扮演越来越重要的角色。在一些场合下，木马的作用已经上升至类似战略武器的高度。

道高一尺，魔高一丈。网络攻击与网络防御，一直是网络安全领域的主旋律。对木马技术进行深入透彻的分析，是网络攻防中的重要内容。

本书侧重于木马核心技术的实现原理与机制分析，尝试通过以点带面的方式，给出木马技术分析的思路与方法。本书从 Windows 系统启动、存储、文件管理、模块管理、通信管理等核心机制入手，深度分析木马核心技术所依托的系统底层基础，详细剖析基于 Rootkit 的木马模块、文件和网络通信端口隐藏、基于 Bootkit 的木马驱动加载与启动、Windows 64 位系统安全机制的绕过、边界防火墙穿透、反病毒软件免杀、如何干扰反病毒分析的过程、Windows 64 位系统兼容等核心技术的原理与实现机制，给出了一个综合运用各项核心技术、高度可配置的木马框架。

全书共分 9 章。第 1 章阐述木马的基本概念、发展阶段、类型及相关核心技术的概况。第 2 章分析了基于 Rootkit 的木马文件、模块及端口等隐藏技术。第 3 章分析了木马利用 Bootkit 绕过 Windows 64 位安全机制，加载驱动的技术。第 4 章分析针对静态查杀和动态查杀的木马免杀技术，及更具隐蔽性的代码注入、傀儡进程、DLL 劫持、脚本化等免杀方法。第 5 章主要分析木马反分析的技术思想及实现。第 6 章阐述木马如何实现 Windows 64 位系统的兼容。第 7 章阐述木马的通信过程、防火墙穿透方法。第 8 章给出了高度可配置的木马整体架构及各个组件之间的设计机制。第 9 章介绍了木马的一些发展趋势和新技术。

本书是木马技术分析与检测方面的参考书，可作为从事反病毒分析、恶意代码检测、安全软件设计与实现、调查取证等研究领域科研人员的参考书，也可作为高等院校信息安全及相关专业大学生与研究生的教材。

在本书成稿过程中，张景鹏在系统调试、分析验证、代码整理等方面做了大量工作，王楠、乔益民、申思、朱腾绩、陈顺卿、高凯轩、段惊园、王倩等人在环境搭建、插图处理、文献整理方面给予了大力协助，曹晗、张雪、齐雯静、郭晓伟等参与了本书的校对工作，在此表示衷心的感谢。

本书在分析过程中，使用了一些已经公开或者半公开的方法与例子，也引用了不少相关论文与著作，在此对有关作者表示感谢。

感谢国家自然科学基金对本书出版的资助（No.：61571360）。

木马技术的发展非常迅速，新的技术与思路日新月异。由于作者知识水平所限，书中疏漏欠妥之处在所难免，恳请读者批评指正。

著　者

2016 年 8 月于西安

目　录

第1章 绪 论

1.1 木马的基本概念

木马（Trojan）是一类特殊的计算机程序，其形式多种多样，但本质目的都是类似的，即非法获得目标主机上的信息，或对目标主机进行控制。木马一词来源于古希腊《荷马史诗》中特洛伊木马的故事。在古希腊时代，希腊军队围困特洛伊城，久攻不下。于是希腊军队假装撤退，留下一具巨大的木马，木马腹中藏匿了许多希腊士兵。特洛伊守军不知是计，把木马作为战利品运进城内。夜里木马腹内的士兵与城外里应外合，一举攻破了特洛伊城。从典故可以看出，木马程序具有一定的隐蔽性、欺骗性，对网络的正常运行和信息的安全有严重的威胁。

木马一般主要由控制端与被控端两部分组成。控制端运行在控制者的机器上，分为网络服务和 GUI 交互两部分[1]。控制者使用 GUI 接口与被控端进行交互，对被控端进行控制；控制端通过网络服务建立监听端口，接受被控端的连入和数据，并维护木马两端的连接，使控制者可以向被控端发送数据和控制命令。

被控端运行在被控制的目标主机上，是木马程序的核心。被控端执行控制者下达的各种功能指令，如远程屏幕监控、远程文件操作、远程命令行等。同时，也要能完成一些信息收集的工作，如目标主机的系统信息、浏览器记录、收发的邮件等。被控端需要和控制端之间建立稳定的通信连接，通过该连接能将收集的数据发送到控制端，并能通过该连接获取到控制端的控制指令，确保控制者对该目标主机的控制。

木马与病毒既有关联，也有区别。二者都是恶意性程序，但通常病毒具有感染性，能够自我复制，而木马一般不具备这个特点。木马一般具有通信功能和可控性，攻击者通过木马程序可以获取和控制目标主机，而病毒则不具备这一特性。随着木马技术的发展，木马实现也开始大量借鉴病毒的技术，而木马在吸收了病毒的技术（如感染、变形等）后大大提升了木马被控端的存活能力。

木马与单纯的远程协助或者控制软件，如远程桌面、TeamViewer、QQ 的远程协助等，也不尽相同。远程协助或者控制软件的目的是为了使得互联网上不相干的机器之间建立远程连接，从而保证一方对另一方提供协助或者支持，实施一些必需的网络管理或者系统使用限制等。远程协助或者控制软件是在目标主机用户授权下进行的，而木马则是在目标主机用户不知情的情况下植入，并运行各种后台的功能，通常具备较强的隐蔽性，而这是远程协助或者控制软件所不需要具备的[2]。

木马一般都包含一些默认的、通用的核心功能，如开机自启动、通信隐藏、绕过杀软和反分析等，这些核心功能体现了一个木马的整体性能水平，决定了一个木马是否能够深度隐藏自身，能否突破杀软、反 Rookit 工具的查杀存活下去，能否顺利连回控制端

完成信息回传。这些核心功能主要包括：

1）自启动的能力。木马可以随着目标主机系统的启动而启动，甚至早于系统启动，以对抗反木马软件的拦截，达到监控系统运行的目的。

2）绕过安全软件检测的功能。木马一般能绕过反病毒软件的拦截和查杀，顺利植入到目标主机，并长期驻留而不被反病毒软件检测与查杀。

3）深度隐藏的能力。木马在目标主机上驻留时，能隐藏自身的文件、模块和通信端口等，尽可能隐藏木马的活动迹象，使用户无法察觉到木马的存在。

4）回连穿透防火墙的能力。目标主机所处的网络环境通常是比较复杂的，所处网络中通常会有防火墙等安全防护设备，这就要求木马能在可控的情况下，与控制端保持通信连接，并能顺利穿透各种内网边界上的防火墙和代理服务器。

5）反分析功能。木马在实现时通常会集成一定的反分析功能，以阻滞、误导对木马的逆向分析，如反汇编、动态调试，也能阻止一些自动化分析工具的分析，如沙盒cuckoo、Thread Expert 等工具对木马行为的自动分析等。

除上述这些核心功能外，木马一般还具备一些应用性功能，这些功能带有明确的应用性目的。常见的应用性功能有[3]：

1）远程命令行。在目标主机上执行系统命令，类似在目标主机本地执行 cmd.exe 的功能。

2）远程文件管理。木马能够查看、删除和移动被控制主机上的文件，就如同在本地使用文件管理器管理文件一样。还能将被控制主机的文件下载到本地，同时支持将本地文件上传到被控制主机。

3）信息窃取。能获取一些常用软件或关键的信息，如聊天软件 Skype 的密码和聊天记录；浏览器的历史记录和通过浏览器登录、访问的账号密码信息；其他一些软件如 KeePass 的访问密码和通过 KeePass 管理的密码信息等。

4）键盘记录。能记录被控制机的一切键盘按键，有部分木马通过 IME 挂钩甚至可以做到记录通过输入法输入的自然语言。

5）进程、服务管理。能够获取目标主机上的进程、服务和服务的状态，可以启动、停止进程和服务。

6）远程桌面。能实时查看远程桌面的变化，并可以截图、录屏等，有的木马可以通过黑屏来锁定远程桌面。

7）摄像头监控。能远程打开被控制端机器连接的摄像头，并能对摄像头实现控制和录影。

上述功能是木马的部分核心或常用的功能，伴随木马技术及应用的发展，木马的功能也呈现多样化、定制化、精细化的特点。尤其是随着互联网业务的增多，木马的目标更精确，类别就更繁多。另外，也出现了使用插件化、脚本化的木马，木马执行控制者下发的插件或者脚本，这些插件可以是一小段独立的简单功能，也可以是一个非常复杂的任务，这使得木马的功能根据植入者的目的，可以任意扩展或者改变，木马的功能不再具有普遍性，而是具有定制化的特点，这种定制化在不同目标和不同环境，甚至在不同时间都是不同的。

木马在用户毫无察觉的情况潜入目标主机，并在目标主机上长期潜伏。这就要求木

马在植入、常驻目标系统时，需要具备较强的隐蔽性，使目标主机用户无法通过查看进程、端口等常规的检测方法发现潜伏的木马，达到长期潜伏的目的。

木马要实现深度的隐藏通常要使用 Rootkit 技术。Rootkit 源于 Unix 系统，原意是指一系列有用的小型程序组成的工具包，使得攻击者成为拥有目标主机最高权限的用户“root”[4]。Rootkit 的功能是在目标主机上隐藏自身及指定的文件、进程和网络链接等信息，其不仅存在于 Windows 系统中，也广泛存在于其他系统，如 OS X、Linux 和 Android 等。

根据实现的机制，Rootkit 可分为用户模式和内核模式。内核模式 Rootkit 在 Windows 内核层实现，通常大量使用未文档化的数据和系统组件代码，使得入侵者能够更长期地占有目标主机系统的底层控制权，并且不容易被杀毒软件等发现[5]。用户模式 Rootkit 在 Windows 操作系统的用户模式下实现，容易被安全软件所检测到。

Windows 下的 Rootkit 依赖于 Windows 内核的组件和机制，尤其是内核 Rootkit。虽然微软提供了 WRK（Windows Research Kernel）供研究人员对 Windows 内核进行研究，但对于规模庞大的 Windows 系统来讲，还只是杯水车薪。目前内核 Rootkit 研究上通常还是依靠对内核的逆向工程，这使得高级的内核 Rootkit 技术实现难度较大。同时，大量更底层的隐藏方法和技术未被揭示或者公开，基于 Rootkit 的深度隐藏实现的难度也较大。

同时，反病毒软件也会参考大量已经公开的 Rootkit 隐藏技术，研究专门的 Anti-Rootkit 工具，有针对性地在系统更底层实现拦截或者进行信息获取，从而查杀基于普通 Rookit 技术进行隐藏的木马。道高一尺，魔高一丈。反病毒软件及 Anti-Rootkit 工具，也不是无懈可击的，内核 Rootkit、反病毒工具和 Anti-Rootkit 的核心模块都工作在 Windows 内核中，这就意味着谁实现的更底层，谁更早启动，谁就能掌握系统的控制权[6]。如果木马的 Rootkit 实现比查杀软件的实现更底层、启动更早，就能拦截 Anti-Rootkit 的功能或者检测对方的存在，以能进行针对性的操作，甚至能监控对方的启动过程，从而禁止对方的某些功能，实现反查杀而存活下来。

为了能更早地启动，木马会使用 Bootkit 这一古老的技术[7]。Bootkit 利用基本输入输出系统 BIOS 固件接口在系统启动之前，先后执行 MBR（Master Boot Record，主引导记录）和 VBR（Volume Boot Record，卷引导记录）的特点，修改 MBR 或者 VBR 的代码，使得系统代码执行之前，先启动执行木马的代码。Bootkit 结合挂钩技术（Hook）能够控制 Windows 系统的启动过程，这不仅可以使得木马早于 Windows 系统启动，从而屏蔽或者绕过 Windows 系统的一些安全策略，如 PatchGuard 和驱动签名等，还能早于安全软件启动，如反病毒软件、Anti-Rootkit 工具等，做好对抗安全软件检测的准备工作。近年来，基于 MBR 的 Bootkit 木马有 Caphaw/Geth、Carberp、TDL4、MBRLock/Bootlock 和 Whistler 等，基于 VBR 的 Bootkit 木马有 Rovnix、Gapz、MaxSS 等。这些木马的扩散范围都很大，产生了很大的影响。

通常，安全软件会采用多种技术拦截、检测木马程序。木马不仅需要对用户、安全工具透明，还需要能够绕过安全软件（杀毒软件、HIPS 等）在系统中安装的常驻组件的检测，执行信息窃取等操作。木马若要成功地在目标系统中存活、常驻，就需要绕过安全软件的查杀，这一过程常称之为免杀。另外，木马在自身的程序中还会加入一些反

调试、反反汇编、反沙盒、反虚拟机检测等功能，这样能够迟滞安全从业人员或者自动化病毒检测环境对木马行为的检测，延长木马的生存周期。

目标主机所处的网络环境通常是比较复杂的，在一些大的机构、公司的重要网络出口会部署边界防火墙、网络代理服务器等安全设备，限制内部用户网络访问，内部用户需要通过代理才能访问网络，这样实现了用户网络访问的行为监督，增强了内部网络的安全性，也增加了木马被控制端回连传输信息的难度。这种内外网边界上的防火墙，如微软的 Forefront TMG，集代理服务、网络路由、防火墙、VPN 服务、Web 缓存和反病毒等于一身，它们提供给用户有限的网络访问方式，即代理服务，其中最常使用的代理服务是 HTTP 代理，且需要进行认证。常用的 HTTP 代理的认证方式主要有 Basic 认证与 Negoiate 认证（NTLM 或者 kerberos 等）。在使用代理认证的网络环境中，木马需要先获取到认证所需要的用户身份信息，探测并确认代理认证方式，然后通过获取到的代理认证方式，与代理服务器交互进行代理认证后，才能进行木马自身的通信，而木马本身通信的协议被限制为只能使用 HTTP/HTTPS。这一过程通常称为木马代理或防火墙穿透[8]。

近年来，伴着多次重大安全事件的发生，如 Google 极光攻击、夜龙攻击、RAS SecurID 窃取攻击等，一种高水平、持续时间长、攻击目的明确的网络攻击逐渐浮出水面，被称为 APT（Advanced Persistent Threat）攻击[9]。APT 攻击针对特定的对象，进行有组织的、长期的、有计划的监视行为，以达到数据窃取的目的。与普通的网络攻击相比，APT 主要以重要目标、重大利益驱动为前提，使用木马等工具进行信息窃取和系统破坏[10]。APT 中使用的木马，制作周期长、技术难度大，通常具备了很多高级特性，如 2010 年攻击伊朗核设施的“震网”、后来的 Duqu、被认为有史以来最复杂的木马程序“火焰”（脚本化程度较高）、针对 EMC RSA 窃取 SECUR ID 令牌攻击等。APT 木马大量使用各种高级技术手段，如跨系统的感染、利用 Rootkit 或者 Bootkit、自定义文件系统等。另外，APT 木马为了提供成功率，经常使用已经公开的或者未公开的漏洞作为木马的植入手段，如使用 Microsoft Office 系列的漏洞、第三方软件的漏洞等。

1.2　木马的发展历程

在早期的 DOS 系统时代，大多数恶意软件主要以炫耀技术或者恶意破坏为主，没有木马的明显特征。随着 Windows 9.x 系统时代的到来，虽然出现了个别优秀的木马，如 Netbus、Bo2k、Sub7 等，但受限于当时互联网的普及程度较低及互联网应用的贫乏，并没有形成非常重大的安全影响。

直到进入 Windows NT 5.0 时代(Windows XP、Windows Server 2003 等)，木马技术得到了快速的发展，也出现了一批功能强大的木马程序，如 gh0st、Beast、冰河、Bifrost Trojan 等。随着 Windows NT 6.0 系统的出现，尤其是 Windows Vista 系统出现之后，微软在操作系统中加入了一系列的安全策略，其中最为突出的一项就是 UAC（User Account Control，用户账户控制），它限制了应用程序对管理员权限的使用，导致很多 Windows NT 5.0 时代的木马不能在 Windows Vista 及以后的一系列系统中正常植入和运行。另外，

Windows 系统内核组件结构的变化，也对木马的植入、启动方式、木马程序的运行权限和隐蔽性等提出了更高的要求，甚至是木马架构上也提出了更高的要求，这就要求木马需要兼容 Windows NT 5.0 和 Windows NT 6.0 的一系列系统环境。

随着计算机硬件及软件需求的发展，Windows 64 位系统逐渐大范围使用，对木马的结构和相关技术又有了新的要求——木马需要专门提供针对 Windows 64 系统的组件。

由此可见，Windows 系统版本的重大更新，会导致木马的结构及相关技术产生新的变化。这种变化所导致的木马技术变化，可以大致作为木马发展阶段的标志。

1. Windows NT 5.0 时代的木马

这个阶段的木马主要运行在 Windows 2000、Windows XP 和 Windows Sever 2003 等系统上，功能相对比较简单，主要以窃取数据为主，有时没有明确的控制端，木马窃取的数据通常都是发送到指定的邮箱等。同时，带有控制功能的木马也逐渐出现，并成为主流。

在通信方式上，这个阶段的木马是以正向连接为主，被植入到目标主机上的木马组件打开监听端口，等待控制方主动连接，然后进行木马的相关控制操作。这种方式需要控制端和被控端处在同一内网，或者被控端植入的目标主机具有公网地址。进而，出现了反向连接的木马，这是木马技术发展的一个重要节点。木马的控制端作为服务器，建立侦听端口，等待被植入木马组件程序的连接，在连接建立后，通过这个连接与木马完成通信和控制功能。至此，木马通信技术基本发展成熟，以后很长时间里，甚至直到现在，木马主要都是使用这两种结构进行通信。

在隐蔽性上，早期的木马基本是基于进程的，在系统中有一个明显的进程，部分木马为了能有一定的隐藏作用，在进程命名上仿造系统进程，如 svch0st.exe、services.exe 等。随着不断发展，也慢慢地出现了一些利用进程注入、DLL 劫持、傀儡进程等，将木马进程注入其他系统进程中进行隐藏的方式。随着 Rootkit 的发展和利用，木马的隐藏技术也发展很快。在这一阶段，基本确定了木马隐藏的具体方向，如文件隐藏、通信端口隐藏、注册表隐藏、模块隐藏及驱动隐藏等[11]。

在启动方式上，木马的自启动主要有注册表启动、系统服务、DLL 劫持等方式。其中，svchost.exe 的服务和 DLL 劫持是最具代表性的两种启动方式。svchost.exe 的服务指的是 svchost.exe 本身作为服务宿主，并不实现任何具体功能，而真正的服务功能以 DLL 形式封装，当系统以服务权限启动 svchost.exe 时，包含服务功能的 DLL 被加载运行。通过将木马功能封装在该 DLL 中，使其被 svchost.exe 加载到自己的进程空间执行，从而使得木马模块具有和系统服务一致的权限[12]。著名的开源木马 gh0st 就是采用这种方式进行自启动的。DLL 劫持是通过替换系统 DLL，使木马的 DLL 加载进正常进程空间的一种方式，由于不需要操作系统敏感区域，这一方式曾被长期、大量使用。

在功能上，本阶段木马的功能基本趋于完善，包含了信息窃取、远程文件操作、远程命令行、远程桌面、键盘记录、远程设置代理等功能。

2. Windows NT 6.0 时代的木马

这一阶段的木马由于 Windows 系统安全机制的完善，出现了一些新的安全特性，最

为突出的就是 UAC。木马程序经常需要使用系统管理员权限进行各种操作，典型的包括启动项的写入、驱动的加载、在系统目录中释放文件等。在 Windows NT 5.0 系统中适用的木马，无法在 Windows 7 和 Windows Server 2008 等系统中正常运行。能不能突破 UAC，是这一阶段木马的一个重要技术指标。

另外，在系统的隐藏性上，由于 Windows 内核的部分底层架构变化，原本适用的一些 Rootkit 方法，在这些系统上不再能发挥作用。例如，在 Windows XP 系统上通过挂钩 tcpip.sys 中的 TCPDispatchDeviceControl 函数，实现了网络通信端口隐藏，但在 Windows 7 系统中因为内核网络组件发生变化，该端口隐藏方法不再有效。

3. Windows X64 系统兼容的木马

从 Windows XP 以来，微软就推出了 64 位版本的系统。但根据有关数据显示，Windows 64 位系统从 Windows 7 开始才被大范围使用。Windows 64 位操作系统不仅可以使用更大的内存容量，而且还具有优越的计算性能。这两大优势，使 Windows 64 位操作系统越来越受欢迎，使用越来越广。因此，64 位 Windows 操作系统也成为了互联网环境中一个不可忽视的重要组成部分[13]。

Windows 64 位对 32 位程序的采用 Wow64（Win32 emulation on Windows 64-bit）子系统，它为已有的 32 位程序提供了 32 位的模拟环境，使得大多数 32 位程序不需要修改，就可以运行在 Windows 64 位系统上，该模拟层处在用户模式下[14]。由于木马的特殊性，往往需要和系统进行紧密的交互，而 Wow64 只是一个轻量级的模拟层，通过它无法访问到真正的系统相关数据结构和信息，这就需要木马本身还要针对 Windows 64 位系统提供 64 位版本的组件。木马提供 64 位版本的组件，不仅仅是因为机器地址长度和可执行文件格式的区别，还牵扯到突破 Wow64 对运行在其上的 32 位程序的限制。Wow64 子系统限制了 32 位程序对系统的部分访问操作，如 Wow64 子系统会重定向 32 位程序对注册表、文件和文件目录的访问操作，使其实际访问 Wow64 对应的注册表项和文件目录；限制 32 位进程对 64 位进程的访问等。一般的，兼容 64 位系统的木马，会同时提供 32 位和 64 位的组件，根据系统的类别，释放相应版本的组件。

Windows 64 位系统的另外一个特点就是安全性进一步加强，PatchGuard 和驱动签名是其中最重要的两个特性。PatchGuard 使得木马的驱动模块不能随意修改系统内核组件，而驱动签名阻止未经签名的驱动被系统加载。这些机制对木马利用 Rootkit 进行了严厉的限制。道高一尺，魔高一丈。在这一阶段，还是出现了很多优秀的木马程序，如 TDSS 系列、Gapz 系列木马，它们利用 Bootkit 技术绕过了 Windows 64 位系统的这些安全限制，甚至使用了自定义的文件系统来达到木马文件隐藏的目的。

4. APT 高级木马

APT 攻击由于其隐蔽性、长期性，且是针对重要目标的攻击行为，这经常会给被攻击目标造成较大的损失。APT 攻击中数据窃取、监控等重要环节基本都是使用木马程序完成。因此，在 APT 攻击中，木马的作用已经上升到一种战略武器的高度，APT 木马也因此具有较高的复杂性和全新的特性。APT 高级木马大量使用未公开的漏洞，以增强

自身的命中率。著名震网木马使用了4个系统漏洞，其中3个未公开。此类木马在功能上趋于脚本化，被植入的木马中附带脚本引擎，控制者只要下发编写的功能脚本，即可完成相应的功能。在通信方式上，呈现出多协议，能穿透防火墙、代理认证，能隐藏被控制端，无明显的控制端。这类木马甚至还附带了传播功能，能跨系统的进行复制，如能通过感染移动设备进行传播。

1.3 木马的类型

自木马程序诞生，在每一个发展阶段，都有很多有代表性的木马。这些木马功能、结构也呈现多样性。不同文献中也给出了多种多样的木马分类方法，总体而言没有形成一种严格的、标准的分类方法。本书从木马的整体目的性功能和木马的网络架构两个方面对木马进行分类。

从应用功能上来看，木马可以包含单个功能，也可以包含多个功能，以功能来归类木马显得非常模糊，种类也不一而足。但从木马在目标机器上植入的目的上来看，又显得比较清晰，从这个层面上可以分为以下三个类别。

① *控制类木马* 控制类木马在被植入到目标机器上后，植入者通过该木马的回连，就可以达到控制和利用目标机器的目的。控制者通过被植入的木马，可以对目标系统的文件进行操作，如上传、下载和删除；在远程机器上执行程序，浏览和控制该机器的运行程序，监视目标系统的桌面、剪贴板；控制目标机器关联的设备，如摄像头等；在目标机器上搭建代理等；集中控制大量的计算机组成僵尸网络，进而进行一些网络攻击活动，如DDOS等[15]。

② *信息窃取类木马* 信息窃取类木马只是进行单方面的信息收集，将收集的结果返回给木马控制端。这种木马大多数都是B/S结构，在窃取的信息到达Web服务器后，数据被存入数据库，控制者通过浏览器查看窃取信息的具体内容。当前，信息窃取类的木马收集的信息主要有几个大的方面：专门截取网络游戏账号信息；专门截取聊天软件账号信息和聊天记录；专门截取邮件客户端信息，收集用户收发邮件内容；专门盗取网络银行、在线支付应用和信用卡账户信息，甚至账户内的货币。

③ *下载者类木马* 下载者类木马的功能比较简单，实际上是攻击者入侵的前行者。此类木马在植入目标机器后，回连到指定地址，从指定的地址下载配置好的恶意软件，或者恶意推广的软件，然后在被控制机器上安装和执行下载的恶意软件，以开展进一步的攻击活动。

木马的控制端和被控端之间是通过网络进行数据交换，从这个角度上讲，木马是一类网络通信软件。常见网络结构主要有C/S结构、B/S结构和P2P结构等[16]。以此分类的话，木马也可分为以下几类。

① *C/S结构* 即Client/Server，也就是客户机和服务器结构通常是服务器端，在一台有公网地址的服务器上开启指定端口进行监听，客户端通过网络协议与端口建立连接后进行通信（TCP），或者直接进行通信（UDP）。客户端的主要作用是将用户的请求提交给服务器端，服务器接收客户端的请求并处理，然后返回执行结果给客户端。最主要的一点是，

C/S 结构的客户端和服务器都是专门的软件程序。C/S 结构，对于正向连接的木马来说，被控端是服务器，控制端是客户端，被控端建立端口侦听，而后控制端主动连接被控端进行通信；对于反向木马来说，被控端是客户端，控制端是服务器端，被控端主动向控制端发起连接和请求。虽然有主从之分，但 C/S 木马的通信交互是双向的。

② B/S 结构　即 Browser/Server，也就是浏览器和服务器结构。是一种常见的 Web 网络结构，相比 C/S 模式，浏览器作为客户端程序。通常服务器程序采用 PHP/JSP 等脚本语言开发，浏览器和服务器之间的交互使用 HTTP/HTTPS、WebSocket 等协议进行通信。严格意义上来讲，木马的被控端并没有被包含在这个结构中。控制者通过浏览器将控制命令发往服务器，服务器对该数据进行缓存，木马被控端不断地向服务器请求已缓存的控制命令和数据，被控制端从服务器得到控制者的命令和数据后，进行解析处理。被控端将窃取的数据发送到服务器，控制端通过浏览器查看这些数据。

③ P2P 结构　也就是对等计算机网络，是一种在 Peer 之间分配任务和工作的分布式应用架构，是对等计算模型在应用层形成的一种网络形式。P2P 结构的木马具有去中心化、扩展性强等优点，能提高木马通信的隐蔽性，避免了木马控制端和被控端之间直接数据交互导致木马控制端信息暴露的问题。P2P 结构的木马，主要用来组建和控制僵尸网络。现有的 P2P 技术的僵尸网络主要有集中式类型、非结构化类型、结构化类型等[17]。

当然在很多情况下，木马植入者的目的不是那么单一，很多时候都包含了多个目的，如既能完成下载者的功能，又能当作后门，甚至还能完成代理的功能。在一些新的木马样本中，甚至大量使用脚本解释器（如 LUA 解释器），通过动态下发功能，来随时完成控制者期望的功能。同时，木马的网络架构也在适应互联网的快速发展而发生演变。总之，在安全软件不断发展、前进的时候，木马也在快速演化着。

1.4　本书组织结构

在木马不断发展、演化的过程中，木马的隐蔽性和复杂性越来越强，在预防和查杀上也越来越困难，木马相关的技术也在不断演进发展。基于此现状，本书系统阐述剖析木马的核心技术，主要包括文件隐藏、高级启动技术、免杀、通信与防火墙穿透、Windows 64 位系统兼容及模块化的木马结构等方面的原理与技术，具体如图 1-1 所示。

本书主要包含以下章节：

第 1 章绪论。主要阐述木马相关知识，包含木马的基本概念、木马的发展阶段及木马的类型，以及相关核心技术的概况。

第 2 章主要分析木马基于 Rootkit 进行文件隐藏、模块隐藏和端口隐藏的原理与机制。文件隐藏包括 Hook SSDT、HOOK FSD 和基于自定义文件系统的隐藏方法；模块隐藏详解了“摘链”和基于 PE Loader 的隐藏方法；端口隐藏部分分析了 Windows NT 5.0 和 Windows NT6.0 上的端口隐藏实现的原理。

第 3 章主要对硬盘分区中的 MBR 和 VBR 进行深入分析。分析了 Windows NT 5.0 和 Windows NT 系统启动中关键组件的启动顺序和作用。分析了木马在启动、Windows 64

位加载驱动的过程中，如何利用 Bootkit 技术实现相应功能的原理和整个流程。

图 1-1　组织结构图

第 4 章重点分析了针对静态查杀和针对动态查杀的免杀技术，并对这两种免杀策略中使用的一些有代表性的方法进行了详细分析，如字符串变形、基于 C++模板的 API 动态调用等，给出了一些实现示例；分析了木马为绕过反病毒软件拦截和查杀，而使用的更为复杂的方法，如傀儡进程、利用 Explorer.exe 进程的窗口过程执行代码、DLL 劫持等；介绍了脚本在木马实现中的应用。

第 5 章主要分析了反病毒分析工作中常用工具，如调试器、反汇编软件、虚拟机和沙盒的工作原理，进而分析了针对此类工具的对抗方法，如反调试、反反汇编、反虚拟机和反沙盒等技术的原理和实现。

第 6 章主要分析了 Wow64 子系统的实现原理、兼容 32 位程序的过程，分析了 Wow64 控制处理器模式的切换过程，以及对文件、注册表重定向的机制等；分析了木马程序在支持 Windows 64 位系统过程中，如何获取当前运行环境信息，在 Wow64 子系统中执行 64 位的指令及调用 Windows 64 位 API 等的过程。

第 7 章主要分析了木马的通信过程及常用的通信协议，重点分析了木马如何在 TCP、UDP 和 HTTP/HTTPS 等标准协议之上，构建木马自定义的通信控制协议；基于实例分析了木马穿透网络防火墙和代理认证服务器的过程。

第 8 章分析了模块化的木马系统框架，对木马的主要组成部分的功能进行了论述；

分析了木马生成器的工作流程，以及木马被控端中 Dropper 组件的功能、模块的组织形式等；分析了木马常驻模块的通信架构、功能插件化等关键技术；对木马控制端的基本架构、网络服务及 IOCP 技术进行了分析。

第 9 章主要对木马技术的新发展进行了概述，分析了木马通信隐蔽化、木马硬件化的趋势、攻击平台多样化等方面涉及的主要技术与趋势。