分类号 _____ 密级 _____ 1814192

UDC[注1] _____

学 位 论 文

基于 PE 文件的启发式特征码自动提取的研究

（题名和副题名）

陈晋福

（作者姓名）

指导教师姓名 耿 技 教 授

电子科技大学 成 都

（职务、职称、学位、单位名称及地址）

申请专业学位级别 硕士 专业名称 软 件 工 程

论文提交日期 2010.04 论文答辩日期 2010.05

学位授予单位和日期 电子科技大学

答辩委员会主席 _____

评阅人 _____

年 月 日

注 1：注明《国际十进分类法 UDC》的类号。

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，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作了明确的说明并表示谢意。

签名：陈晋福　　日期：2010 年 5 月 18 日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

签名：陈晋福　　导师签名：

日期：2010 年 5 月 18 日

摘 要

随着互联网的飞速发展，计算机病毒技术也在迅速发展，逐渐融合了网络蠕虫、木马、拒绝服务攻击等各种攻击手段。

造成的损失也由最初的数据丢失，发展到现在的信息泄密，甚至互联网的瘫痪，破坏力越来越大。但不幸的是病毒检测技术还不能完全跟上病毒编写者的速度。特别在特征码提取这一块，现在仍旧停留在人工分析提取特征码的阶段，这使得一个新病毒从发现到检测需要一个很长的过程，而现在的新型病毒的传播速度越来越快，破坏程度也越来越严重。基于上述分析，特征码的提取方式和提取速度必须跟上病毒的传播速度。

本文针对特征码提取技术做了如下工作：

(1)提出了基于 PE 文件启发式特征码自动提取技术。该技术是建立在对病毒感染策略详细分析的基础上，通过对病毒样本的 PE 结构扫描，利用病毒复制和传播的特性，寻找病毒在 PE 结构中区别于正常文件的启发性标志。该启发性标志的选择主要依据病毒的感染策略，从 PE 文件的 PE 头部，可选头部，入口点地址，节表等病毒容易感染区段进行选取，然后通过启发性标志提炼出病毒的特征序列，最后选出最能代表该病毒特性的特征序列组合成特征码。

(2)设计并实现特征码自动提取系统。讨论系统的总体设计和各模块的之间的关系，详细介绍了系统核心模块的设计流程和算法。该系统是根据基于 PE 文件启发式扫描的思想设计的并在该系统中融入了对病毒样本的特征码实现无人工干预的自动提取技术。接着我们通过该系统对大量恶意样本进行特征码提取，并在积累大量病毒样本的特征码基础上对基于PE文件启发式特征码技术和特征码的质量进行验证。对特征码质量的验证主要为误报率，漏报率，检查变种病毒的能力，和特征码提取效率等方面的验证。对启发性标志的验证主要为对特征码标志设置是否合理和寻找更优启发性标志等方面。最后根据实验结果对基于 PE 文件启发性特征码自动提取技术进行总结并提出不足和需要改进之处。

关键词：特征码，PE 文件，特征序列，Win32 病毒

ABSTRACT

With the skyroketing development of the Internet, the computer virus technology experiences a rapid advancement too, for it mixes various attack means ,such as Internet worm, Trojan virus,and denial of service.

The loss it causes worsens from the data loss initially to the information disclosure,even to a paralyzed Internet presently , thus the destructive power grows stronger and stronger. But unfortunately, the advance of virus detection technology has not caught up with the pace of virus writers yet. Especially at the aspect of signature extraction, we still stay at the stage of manuel analysis and collection, thus making it a very long procedure from the discovery of a new virus to its detection, in spite of the increasing spreading speed and the worsening destruction of new virus.Based on the above analysis, the advance of the way and the speed of signature colletion must catch up with the spreading pace of virus.

The dissertation reveals the following points about the technology of signature extraction:

The dissertation puts forward the automatic signature extraction technology based on PE file heuristic analysis. Based on a detailed ananlsis of virus infection strategy, by scanning the PE structure of virus samples, making use of the virus' features of duplication and tansmission, this technology can be used to find the virus' heuristic sign that differs from the normal documents' sign in the PE structure.The choice of heuristic sign mainly accords to the virus' infection strategy, by choosing the vulnerable sections,such as the PE head of PE file, optional head, the entry point address, and then extracting the virus' sign sequence, finally we pick out the one that best represents the virus' features and assemble it into a signature.

The dissertation designs and factualizes the automatic extraction system of signature.The dissertation also presents a discussion about the overall design,the relation of different modules, and introduces the design procedure and algorithm of system's core module in detail. The system is desiged on the idea of PE file heuristic analysis scanning,and blended in the automatic collection system without manuel

intervention to virus' samples.Then through collecting signatures of various malicious samples, and on the base of enormous accumulation of virus samples' signatures,we validate the quality of signatures and its technology based on PE file heuristic analysis. The validation of feature codes' quality is mainly about validating the rate of false alarms and missing reports, checking the ability of varietal virus, and testing the efficiency of signatures.The validation to heuristic sign is mainly about such aspects as verifying the rationality of its setting,and seeking for a better activation sign. Finally,the dissertation makes conclusions and puts forward the defects about automatic PE file heuristic analysis signature extraction technology on the base of experimental results.

Key words: signatures, PE file, signatures sequence,Win32 virus

目录

第一章 绪论

随着计算机技术的迅速发展，微型计算机日益得到广泛应用，互联网在世界上迅速普及。现在互联网已经愈来愈显示出了无可替代的作用。当前，在世界范围内，一个以微电子技术，计算机技术和通信技术为先导的，以信息产业和信息技术为中心的信息革命方兴未艾。随着信息技术和信息产业的发展，国民经济的发展，国家经济信息化已经起着举足轻重的作用，并成为衡量一个国家发展水平的重要标志。因此，实现国家信息化，已经成为世界各国所追求的共同目标。

但是，计算机病毒的蔓延使计算机的安全性受到了严重的威胁[1]。从 1983 年计算机病毒首次被确认以来，并没有引起人们的重视。直到 1987 年计算机病毒才开始受到世界范围内的普遍重视。从某种意义上说，刚刚开始的二十一世纪是计算机病毒与反病毒大斗法的时代，“红色代码”、“齿轮先生”、“尼姆达”病毒的粉墨登场似乎已经证明了这一点，就像当时在 DOS 环境下病毒的发展一样，视窗操作系统病毒的发展也是从简单到复杂，现在的视窗操作系统下的病毒已经非常完善了，它们通过高级语言编程实现，利用了视窗操作系统的种种漏洞，使用先进的加密和隐藏算法，甚至可以直接对杀毒软件进行攻击。例，CIH 病毒在全球先后造成损失据估计超过 10 亿美元，而受 2000 年爱虫病毒的影响，全球的损失预计更是高达 100 亿美元[2]-[4]。据报道 2000 年病毒造成的直接损失在美国就达 140 亿美元。而随着因特网时代的到来，电脑病毒似乎开始了新一轮的进化，脚本语言病毒从最早的充满错误，没有任何隐藏措施发展到今天和传统病毒紧密结合，包含了复杂的加密/解密算法，未来的电脑病毒只会越来越复杂，越来越隐蔽，病毒技术的发展对杀毒软件提出了巨大的挑战。

现在主流的特征码提取方法还停留在人工提取的阶段，虽然提取的特征码准确，误报率低，但效率问题是制约人工特征码提取的一个重要因数。由于变形病毒的出现，病毒的数量与日俱增，使人工提取特征码的方法已经不能跟上病毒出现的数量。

所以我们需要研究和开发出恶意代码特征码自动提取系统，并且能够通过特征码检测出变种的恶意代码。

1.1 特征码提取技术现状

国内知名反病毒软件公司，如金山毒霸，瑞星，江民等，在病毒查杀方面仍主要采用传统的特征码扫描技术，再附加一些主动防御的功能[5]。金山毒霸对病毒和木马的查杀采取病毒库+主动防御+互联网可信认证技术的三重防护。使用数据流、脱壳等一系列先进查杀技术，打造强大的病毒、木马、恶意软件查杀功能，将藏身于系统中的病毒、木马、恶意软件等威胁一网打尽，保障用户系统的安全。瑞星杀毒使用的"碎甲"技术通过对 Windows 驱动程序加载点进行拦截，当发现 Rootkits 时自动使其保护功能失效，就象穿甲弹击碎盔甲一样。目前，此技术可以有效对付 600 余种 Rootkits，并且当有新的 Rootkits 出现时能够迅速地进行升级处理[6]。"江民未知病毒检测"采用病毒行为主动防御技术，可以自动判别目前系统进程的安全状况，用户可以根据进程可疑概率的高低判别是否感染未知病毒并进行相应的处量。江民主动防御系统将主动防御技术与病毒特征码技术完美结合。

Norton,McAfee,Dr.Web,Trend 以及 Kaspersky 等世界领先级的防病毒软件公司都投入了大量的人力和物力进行变形病毒以及未知病毒的检测研究。如现在备受推崇的 KAV（AVP）就是依靠其高效率的虚拟机和启发机制，在病毒检测方面独树一帜。卡巴斯基引擎中的虚拟机技术相当先进，并且采用了单一形式的规则判断，遇见病毒时的启动非常快。McAfee 防止应用程序溢出技术，不考虑硬件平台的情况下将虚拟机技术和实时监控技术的完美结合，Dr.Web 可以对各类 Word 病毒做出快速反应，并进行隔离和清除。脱壳能力超强[7]-[8]。它采用新型的启发式扫描方式，提供多层次的防护方式，紧紧的和操作系统融合一起，拒绝接纳任何包含恶意的代码进入电脑，比如病毒、蠕虫、特洛伊木马以及广告软件、间谍软件等等。新型的基因式扫描杀毒软件。可以预防并清除 22000 种以上的病毒及特洛伊木马，其中包括各种高复杂多变异型的病毒。ESET NOD32 拥有先进的 ThreatSense（专利申请中），可通过对恶意代码进行分析，实时侦测未知的病毒，时刻走在病毒编写者的前面[9]。病毒程序的防护必须要在其对计算机造成影响前实时地进行。那些时刻等待着病毒特征库更新的防毒软件会给攻击打开一扇窗，稍不留神就有可能给您造成灾难性的后果。ESET NOD32 则凭借其 ThreatSense 技术，将会关闭这扇窗，而不像大部分依靠特征库更新的防毒软件。从单个控制台提供高级的、企业范围的病毒防护和监视。

特征代码检测法是使用最为普遍的恶意代码检测方法，国外专家认为特征代

码检测法是检测已知恶意代码的最简单、开销最小的方法。检测时，打开被检测文件，在文件中搜索，用恶意代码数据库中的特征码去匹配文件中的字符串，如果匹配成功，则报告发现恶意代码。如果发现恶意代码特征代码，由于特征代码与恶意代码一一对应，便可以断定，被查文件中患有何种恶意代码。

采用恶意代码特征代码法的检测工具，必须不断更新恶意代码数据库，否则检测工具便会老化，逐渐失去实用价值。特征代码检测法对从未见过的新恶意代码，无法检测。

现在商用的恶意代码检测软件采用的都是“特征码”检测技术，即当发现一种新的恶意代码后，采集其样本，分析其代码，提取其特征码，然后加入到恶意代码特征库中去。

(1)简单特征码

80 年代末期，基于个人电脑病毒的诞生，随即就有了清除病毒的工具——反病毒软件。这一时期，病毒所使用的技术还比较简单，从而检测相对容易，最广泛使用的就是特征码匹配的方法[11]。

特征码是什么呢？比如说，“如果在第 1034 字节处是下面的内容：0xec , 0x99, 0x80,0x99，就表示是大麻病毒。”这就是特征码，一串表明病毒自身特征的十六进制的字串。特征码一般都选得很长，有时可达数十字节，一般也会选取多个，以保证正确判断。杀毒软件通过利用特征串，可以非常容易的查出病毒。

(2)广谱特征

为了躲避杀毒软件的查杀，电脑病毒开始进化。病毒为了躲避杀毒软件的查杀，逐渐演变为变形的形式，每感染一次，就对自身变一次形，通过对自身的变形来躲避查杀。这样一来，同一种病毒的变种病毒大量增加，甚至可以到达天文数字的量级。大量的变形病毒不同形态之间甚至可以做到没有超过三个连续字节是相同的。

为了对付这种情况，首先特征码的获取不可能再是简单的取出一段代码来，而是分段的，中间可以包含任意的内容（也就是增加了一些不参加比较的“掩码字节”，在出现“掩码字节”的地方，出现什么内容都不参加比较）。这就是曾经提出的广谱特征码的概念。这个技术在一段时间内，对于处理某些变形的病毒提供了一种方法，但是也使误报率大大增加，所以采用广谱特征码的技术目前已不能有效的对新病毒进行查杀，并且还可能把正规程序当作病毒误报给用户[12]。

(3)行为判定

针对变形病毒、未知病毒等复杂的病毒情况，采用了虚拟机技术从行为判断

提取特征码，它实际上是一种可控的，由软件模拟出来的程序虚拟运行环境。特征码提取方法：通过代码仿真，让病毒在虚拟环境中运行，记录其对系统破坏特征，提取出对应特征码。

1.2 本文研究的意义和目的

随着计算机技术的迅速发展，微型计算机日益得到广泛应用，互联网在世界上迅速普及。现在互联网已经愈来愈显示出了无可替代的作用。当前，在世界范围内，一个以微电子技术，计算机技术和通信技术为先导的，以信息产业和信息技术为中心的信息革命方兴未艾。信息技术和信息产业的发展，以国民经济的发展，国家经济信息化起着举足轻重的作用，并已经成为衡量一个国家发展水平的重要标志。因此，实现国家信息化，已经成为世界各国所追求的共同目标。

但是，计算机病毒的蔓延使计算机的安全性受到了严重持久战。从 1983 年计算机病毒首次被确认以来，并没有引起人们的重视。直到 1987 年计算机病毒才开始受到世界范围内的普遍重视。从某种意义上说，刚刚开始的二十一世纪是计算机病毒与反病毒大斗法的时代，“红色代码”、“齿轮先生”、“尼姆达”病毒的粉墨登场似乎已经证明了这一点，就像当时在 DOS 环境下病毒的发展一样，视窗操作系统病毒的发展也是从简单到复杂，现在的视窗操作系统下的病毒已经非常完善了，他们使用高级语言编写，利用了视窗操作系统的种种漏洞，使用先进的加密和隐藏算法，甚至可以直接对杀毒软件进行攻击。例，CIH 病毒在全球先后造成的损失据估计超过 10 亿美元，而受 2000 年爱虫病毒的影响，全球的损失预计更是高达 100 亿美元。据报道 2000 年病毒造成的直接损失在美国就达 140 亿美元。而随着因特网时代的到来，电脑病毒似乎开始了新一轮的进化，脚本语言病毒从最早的充满错误，没有任何隐藏措施发展到今天和传统病毒紧密结合，包含了复杂的加密/解密算法，未来的电脑病毒只会越来越复杂，越来越隐蔽，病毒技术的发展对杀毒软件提出了巨大的挑战。

现在主流的特征码提取方法还停留在人工提取的阶段，虽然提取的特征码准确，误报率低，但效率问题是制约人工特征码提取的一个重要因数。由于变形病毒的出现，病毒的数量与日俱增，使人工提取特征码的方法已经不能跟上病毒出现的数量。

所以我们需要研究和开发出恶意代码特征码自动提取系统，并且能够通过特征码检测出变种的恶意代码。经过大量文献调研和对病毒感染技术的研究我们最

终选取了基于 PE 文件的病毒特征码自动提取技术作为本文的研究内容。

1.3 研究工作

随着互联网的飞速发展，计算机病毒技术也飞速发展，逐渐融合了网络蠕虫、木马、拒绝服务攻击等各种攻击手段。造成的损失也由最初的数据丢失，发展到现在的信息泄密，甚至互联网的瘫痪，破坏力越来越大。

但不幸的是病毒检测技术还不能完全跟上病毒编写者的速度。特别在特征码提取这一块，现在仍旧停留在人工分析提取特征码的阶段，这使得一个新病毒从发现到检测需要一个很长的过程，而现在的新型病毒的传播速度越来越快，破坏程度也越来越严重。基于上述分析，特征码的提取方式和提取速度必须跟上病毒的传播速度。

本文首先从病毒的感染技术入手，分析病毒的感染方式并对病毒的加壳方法进行讨论。然后介绍了目前比较流行的特征码提取方法和壳检测以及特征码提取方法。提出了基于 PE 文件启发式特征码提取的技术，最后设计并实现了基于 PE 文件启发式特征码自动提取平台，并通过大量实验验证算法的可行性。具体工作如下：

(1)提出了基于启发式特征码自动提取技术，该技术通过对病毒样本的 PE 结构扫描，利用病毒复制和传播的特性，寻找病毒在 PE 结构中区别与正常文件的启发性标志，然后通过启发性标志提炼出病毒的特征序列，最后选出最能代表病毒特性的特征序列组合成特征码。

(2)对病毒样本的特征码实现无人工干预的自动化提取。

(3)设计并开发特征码自动提取平台，实现（1)、(2）中的思想。并自动提取病毒特征码。最后通过大量实验验证（2)、(3）思想的可行性和提取出的特征码质量，最后提出算法的不足和需要改进之处。

1.4 本文主要类容及组织结构

本课题围绕提出和实现一种新的恶意代码特征码自动提取方式：基于 PE 文件启发式特征码自动提取算法研究进行撰写。主要内容安排如下：

第一章 分析特征码自动提取技术的现状，介绍了课题的研究意义，提出了所要解决的问题。

第二章 从病毒的感染策略入手简要介绍了几种常见的病毒感染方式，深入分析 Win32 和 Win64 病毒与各 Windows 版本平台的内部依赖关系，最后对壳进行讨论，分析及介绍了目前常见的壳。

第三章 首先从一个病毒编写者的角度介绍了 PE 格式的概念然后深入分析了 Win32 病毒与 PE 结构的关系。在分析比较各种特征码提取方法利弊的基础上，提出了基于 PE 文件的启发式特征码提取算法。

第四章 在 Windows 平台下，设计并实现一套基于 PE 文件启发式特征码自动提取的平台。

第五章 对该平台进行了实例测试和分析。

第六章 总结目前的研究工作并对后续的工作进行展望。

第二章 恶意代码的行为分析

本章首先介绍病毒代码研究现状，第二节从病毒的感染技术入手，分析各种病毒的感染策略，以及 Win32 和 Win64 病毒与 Wiondows 系统的依赖关系。第三节介绍壳的概念以及其加载过程，另外还介绍了现在流行的一些壳。最后进行总结。

2.1 恶意代码研究现状及发展趋势

在网络安全事件中，危害性极大的恶意代码事件增长迅速。我国因此而造成的直接经济损失每年高达数 10 亿，而且正以每年 30%至 40%的速度递增。

2007 年网页恶意代码事件超出 2006 年总数 2.6 倍，而 2007 年我国大陆地区被植入恶意代码的主机 IP 增长数量就是 2006 年的 22 倍[13]。

2008 年中，全国的计算机网络虽然没有出现大规模网络拥塞和系统瘫痪事件。但是，“AV 终结者”、“机器狗”和“磁碟机”等影响较大的恶意代码事件相继发生。

在“2008 反病毒技术发展趋势研讨会”上，金山毒霸技术总监陈睿表示，2008 年，新增木马数量将突破 100 万。

根据杀毒软件服务商 360 安全中心的报告，2008 年 1 到 11 月间，360 安全中心共截获了 8772714 种恶意软件，相较于 2007 年这个数字仅为 731849，一年内暴涨了近 12 倍。

根据瑞星公司的报告，2008 年 1 月至 10 月期间瑞星共截获新病毒样本 9306985 个，是 2007 年同期的 12.16 倍。其中木马病毒 5903695 个，后门程序 1863722 个，两者之和超过 776 万，占总体恶意代码数量的 83.4%[14]。这些病毒都以窃取用户网银、网游账号等虚拟财产为主，带有明显的经济利益特征。

而由于经济利益的驱使，2008 年的恶意代码中，能产生利益的木马等部分占据了相当大的部分，而纯技术少利益的病毒等则日渐式弱。

根据瑞星公司的《2008 年中国大陆地区电脑病毒疫情&互联网安全报告》，2008 年 1 月至 10 月，全国约有 8100 多万台电脑终端曾经被恶意代码感染，其中通过网页挂马方式被感染的超过 90%[15]。2008 年 10 月份，瑞星对 1 万台上网电脑的

抽样调查标明，这些电脑每天遇到的挂马网站，高峰期达到8428个，最低也有1689个，去除单台电脑访问多个挂马网站的情况，每天平均有30%的网民访问过挂马网站，中国大陆地区已经成为全球盗号木马最猖獗的地区之一。

开展高效、准确的恶意代码应急响应相关技术研究，不仅有利于提升反恶意代码和防恶意代码的能力，有利于遏制恶意代码在网络中的传播，也有助于维护中国互联网的健康环境和营造一个和谐的网络社会。

2.2 恶意代码感染策略

如果按照感染策略来划分病毒，可将病毒划分为引导区病毒和文件感染型病毒其中文件感染型病毒又可分为重写病毒、随机重写病毒、追加病毒、压缩型病毒等等，本节将重点分析采用文件感染技术的重写病毒、压缩病毒、变形感染计算，入口点隐藏病毒的机理为后面启发式特征码提取算法做好铺垫。

2.2.1 文件感染技术

2.2.1.1 重写病毒

有些病毒只是从磁盘上找到一个文件，然后简单地用自己的拷贝改写该文件。当然这是一种非常初级的技术，不过确实是最为简单的方法。如果这种简单的病毒重写磁盘上所有文件的话，可能造成很大破坏。

重写病毒是不能从系统中直接地删除掉的，只能把被感染的文件删除掉，然后再从备份介质中恢复[16]-[18]。如图2-1表示了重写病毒攻击宿主文件内容的变化。

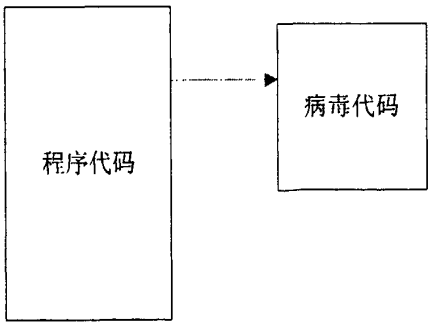

图2-1 重写病毒感染改变了宿主文件的大小

一般来说，重写病毒不是非常成功的威胁，因为病毒感染造成的明显影响太容易被发现了，然而，这种病毒如果和基于网络的传播技术结合起来，可能产生很大威胁。比如，VBS/Loveletter.A@mm 病毒通过群发邮件把病毒送到其它的系

统里，当该病毒执行时，它会用自己的拷贝重写本地所有带有下面扩展名的文件：

.vbs,.vbe,.js,.jse,.css,.wsh,.sct,.hta,.jpg,.jpeg,.wav,.txt,.gif,.doc,.htm,.html,.xls,.ini,.bat,.com,.avi,.qt,.mpg,.mpeg,.cpp,.c,.h,.swd,.psd,.wri,.mp3,and .mp2.

另一个重写病毒传染方法用于所谓的 tiny 病毒，DOS 平台的 Trivial 病毒家族就是一种典型的 tiny 病毒[19]。许多病毒作者试图写出最短的病毒，于是出现了 Trivial 的很多变种。最短小的病毒只有 22 个字节（Trivial.22）。

这种病毒的算法非常简单：

1.在当前目录下寻找任何(*.*)新的宿主文件。

2.以写方式打开文件。

3.把病毒代码写入宿主文件的顶端。

这些最短的病毒通常只能感染当前目录下的一个宿主文件，因为找到下一个宿主文件需要很多字节的代码。这种病毒也不能感染标记为 read-only(只读)的文件，因为那也需要许多额外的指令。

操作系统在载入程序时会初始化某些寄存器，利用这些寄存器的内容，病毒代码还可以优化，因此，病毒作者还可以用这些条件进一步缩短他们的病毒代码。

但是，如果那个平台没有用病毒所期望的方式对寄存器进行初始化，这些优化可能导致病毒在另外一个平台上执行出错。

有些重写病毒还利用 BOIS 的写磁盘功能而不是 DOS 的文件功能感染新文件。这种病毒最初级的形式只有 15 个字节。病毒用自己的代码重写每一个扇区。显然，这种病毒太快地杀死了自己的宿主系统，系统的崩溃阻碍了病毒的进一步传播。

图 2-2 展示了重写病毒重写了宿主文件的顶部，而没有改变文件的大小。

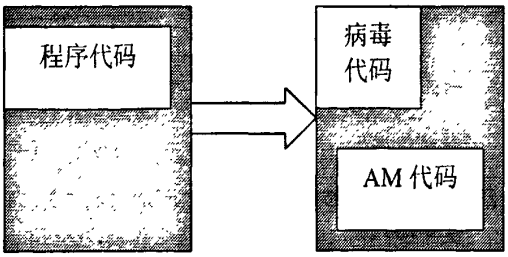

图 2-2 重写病毒没有改变宿主文件的大小

2.2.1.2 压缩病毒

压缩宿主程序是一种特殊的感染技术[20]-[22]。这种技术有时用来隐藏宿主程序长度的增长：采用一个二进制的压缩算法，对宿主程序进行充分的压缩。压缩型病毒有时被认为是“有益的”，因为他们会把宿主程序压缩很多，从而节省了磁盘

空间。(运行时的二进制压缩工具，如 PKLITE,LZEXE,UPX 或 ASPACK，是非常流行的程序，它们中好多都被攻击者自发地用来压缩木马，病毒或电脑蠕虫的代码，以增加迷惑性，同时减少病毒长度)。

DOS 病毒 Cruncher 是最早使用压缩技术的病毒[23]。用到这个技术 32 位 Windows 病毒包括 Vecna 编写的 W32/HybirsF(HybirsF 蠕虫的一个文件感染插件)。另外一个声名狼藉的病毒是 W32/Aldebera，该病毒结合了压缩技术和多态技术。Aldebera 在压缩宿主程序时，试图令压缩结果和原始文件长度。。

Jacky Qwerty 编写的 W32/Redemption 病毒同样利用了压缩技术来感染 Windows 中的 32 位 PE 文件。图 2-3 显示了压缩型病毒如何感染一个文件。

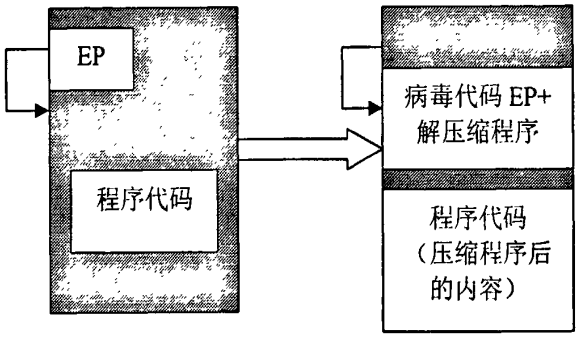

图 2-3 一个压缩型病毒

2.2.1.3 变形感染技术

变形虫感染技术是现在比较常见的一种技术[24]-[33]。这种技术把宿主程序嵌入到病毒体中：具体是把病毒头部放到文件之前，而病毒尾部追加到宿主之后。病毒头部可以访问尾部，然后被转载人。病毒在硬盘上生成一个包含原始宿主内容的新文件，以便于它将来可以正确运行。例如病毒作者 Alcopaul 编写的 W32/Sand.12300 就是运用了这个技巧来感染 Windows 系统中的 PE 文件。该病毒是用 Visual Basic 编写的。

图 2-4 显示了宿主程序被采用了变形虫技术的病毒感染前后的情况。病毒程序在感染正常文件时首先将病毒头加入到原始程序的头部，再将病毒尾部添加到正常文件的尾部。

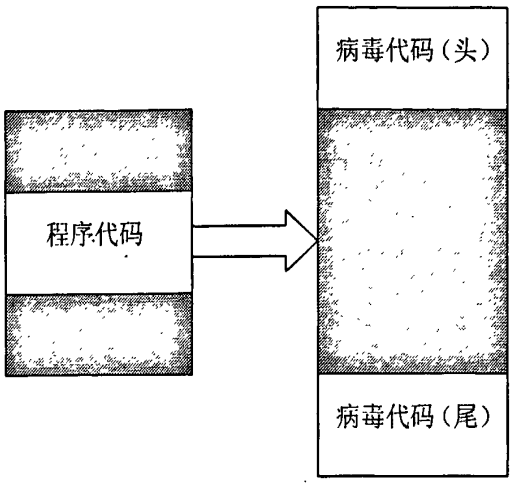

图 2-4 变形虫感染技术

2.2.2Win32 和 Win64 病毒与 Microsoft Windows

微软的策略很清楚。一个软件产品要想获得“针对 Microsoft Windows 设计”的标识，最重要的要求就是：该产品中每个应用程序都必须是用 32 位编译器生成的 PE 格式的 Microsoft Win32 可执行程序。因此毫不奇怪，第三方开发的 Win32 程序的数量在过去几年中增长的特别快。人们相互交换和下载的PE程序越来越多。

Windows95 及 Win32 病毒很长一段时间都未引起大问题的主要原因是病毒编写者必须学习大量的知识才能“支持”新的系统。年轻的病毒作者们都熟悉微软的广告词“Windows everywhere”病毒作者们的回答是“Windows viruses everywhere!”这些病毒作者不会再为 DOS 病毒浪费时间，而是会一直不断地专研 Win32 和 Win64 平台。

攻击者继续编写 DOS 病毒已经不再有任何意义。因为，现在病毒扫描器的主要弱项是如何一般性地或启发性地检测 Windows 病毒，另外 Windows 病毒的清除也不是那么容易。反病毒厂商还必须学习和理解新的 64 位文件格式，并且投入足够的时间来研究和设计新的扫描技术。

由于 Windows95 和 NT 比 DOS 复杂，因此 Windows 95/NT 中的第一代病毒的开发自然就花费了更多时间。然而，Win32 病毒的数量在与日俱增。根据已知数据，DOS 病毒达到 10000 个变种用了 10 年，但 Win32 病毒只用了 9 年。这表明：随着新平台的出现，尽管病毒编写速度暂时缓慢，但最终任何类型病毒的增长率都将是指数级的[34]。

同一个程序，Windows95 的装入程序认为它是好的，而 Windows NT 的装入程序认为它是坏的。这个问题的出现是因为 PE 文件的原因。PE 文件将会在下一章详细的进行介绍。

PE 文件格式是微软为其所有 Win32 操作系统（Windows NT/2000/XP/2003，Windows 95/98/ME,Win32s 和 Windows CE）设计的。这就是为什么所有 Win32 系统中的装入程序都必须了解这种可执行文件格式的原因。然而，装入程序在不同系统中的不同的具体实现。Windows NT 的装入程序在执行一个 PE 文件前对其所做的检查比 Windows 95 的载入程序多。因此 Windows NT 发现被 Boza（第一列 Windows 95 病毒）感染的文件很可疑。具体原因是（病毒在宿主节表中插入的）.vlad 节头部中的一个域被病毒代码算错了。如果计算正确，正确的节和节头部可以被毫无问题的加入到 PE 文件中。因此，Windows NT 的装入程序并非像某些人猜想的那样包含什么优秀的病毒检测技术。

如果 Boza 修正了这个问题，那么它甚至可以在 Windows NT 平台上启动宿主程序。但是病毒在 Windows NT 上仍然将无法复制，因为存在另一个兼容性的问题，该问题影响了所有早期的 Windows 95 病毒[35]。所有 Windows 95 病毒都必须解决一个特殊的问题。即如何调用 GetModuleHandle()和 GetProcAddress()这两个 Win32 内核 API，由于这两个 API 位于 KERNEL32.DLL 中，因此 Windows 95 病毒可以通过直接 KERNEL32.DLL 访问这些函数。多数 Windows 95 病毒都是用将 GetModuleHandle() 和 GetProcAddress() 地址进行硬编码的指针。通过 GetProcAddress()病毒可以访问自己想要调用的任何 API。

当链接程序生成一个可执行文件时，它认为文件将被映射到内存中的一个特定位置。PE 文件头部的 ImageBase 域保存了这个地址值。对可执行文件来说，这个地址通常缺省为 0x400000。在 Windows 95 中 KERNEL32.DLL 的 ImageBase 的地址为 0Xbff700000。因此，GetModuleHandle()和 GetProcAddress()的地址在 KERNEL32.DLL 的同一发布中应该是固定的，但在 KERNEL32.DLL 的新版本中这个地址可能不同，这使得 Windows 95 病毒甚至不能兼容其他的 Window 95 发布。Windows NT 中的 Imagebase 域的缺省值为 0x77F000000，因此那些只能使用 Windows95 中的基地值的病毒就无法感染 Windows NT[36]-[38]。

导致不兼容的第三个原因是显然的：Windows NT 不支持 VxD。因此像 Memorial 这样的基于 VxD 的病毒就不能在 Windows NT 上运行。它们本应该该具有 Windows NT 和 Windows 95 的不同而采用不同的驱动级感染算法，但这样做会令病毒变得很复杂。

2.3 恶意代码的加壳技术

2.3.1 壳的概念

在自然界中，植物用壳来保护种子，动物用壳来保护身体。同样，在一些计算机软件里也有一段专门负责保护软件不被非法修改或者反编译的程序。他们一般都是先于程序执行，拿到控制权，然后完成它们保护软件的任务。由于这段程序和自然界的壳在功能上有很多相同的地方，基于命名的规则，就把这样的程序称为“壳”了，也就是软件壳[39]。如图 2-5。

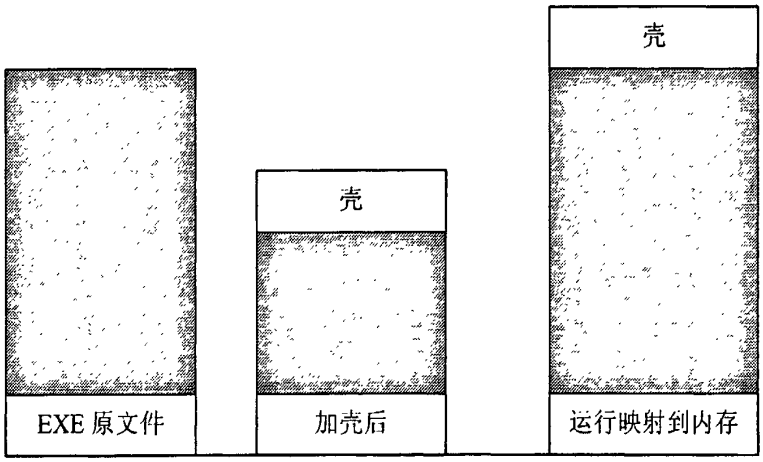

图 2-5 描述壳的示意图

软件壳是和软件加密和保护分不开的。最早提出“壳”这个概念的，是当年推出脱壳软件 RCOPY3 的作者熊焰先生[40]。在 DOS 时代，“壳”一般都是指磁盘加密软件的段加密程序。由于那个时候的加密软件还刚刚起步，所以大多数的加密软件(加壳软件)所生成的“成品”在“壳”和需要加密的程序之间总有一条明显的“分界线”。有经验的人可以在跟踪软件的运行以后找出这条分界线来。随着软件加密技术的发展，保护 EXE 文件不被动态跟踪和静态反编译就显得非常重要了。所以专门实现这样功能的加壳程序便诞生了。MESS、CRACKSTOP、HACKSTOP、TRAP、UPS、ASpack、UPX、PECompact、ASProtect、tElock、DBPE 等都是比较有名气的本类软件代表。

由于 Microsoft 保留了 Windows 95 的很多技术上的秘密，即便 Windows 95 已

经推出三年多了的时间，也没见过在其上面运行的“壳”软件。直到1998年中期，这样的软件才迟迟出现，而这个时候Windows 98也发布了一段时间。这类的软件不发表尚可，一发布就大批地涌现出来。显示加壳类的软件，如BJFNT,PELOCKNT等，他们的出现，使暴露三年多的Windows下的PE格式EXE文件得到很好的保护。接着出现的就是压缩壳（Packers），因为Windows下运行的EXE文件“体积”一般都比较大，所以它的实用价值比起DOS下的压缩软件要大的多，这类软件也很多，UPX,ASPack，PEECryptor等。随着加壳技术发展，这两类软件之间的界限越来越模糊，很多加壳软件除了具有较强的压缩性能，同时也有了较强的保护性能。

加壳软件一般都有良好的操作界面，使用也比较简单。除了一些商业壳，还有一些个人开发的壳，种类较多。壳对软件提供了良好保护的同时，也带来了兼容性的问题，选这一款壳包含软件后，要在不同硬件和系统上多测试。由于壳能包含自己代码，因此许多木马和病毒都喜欢用壳来包含和隐藏自己。对于一些流行的壳，杀毒引擎能对目标软件脱壳，再进行病毒检查。而大多数私人壳，杀毒软件不会专门开发解压引擎，而是直接把壳当成病毒或木马处理。

有加壳就一定会有脱壳。一般的脱壳软件多是专门针对某加壳软件编写的，虽然针对性强，效果好，但是收集麻烦。因此掌握手动脱壳技术十分必要。

软件壳和病毒在某些方面是比较类似的，都需要比原程序代码更早地获得控制权。壳修改了原程序的执行文件的组织结构，从而能够比原程序的代码提前获得控制权，并且不会影响原程序的正常运行。

2.3.2 壳的加载过程

利用软件壳，使用者可以对被加壳程序进行变化，改变特征码或者移动其位置，使之看上去成为了另外的应用程序，但功能并不改变。从这点上来说，加壳就是对目标程序的形态上的一种变形。

我们以Windows系统下PE文件为例，简单说说一般壳的装载过程。

(1).获取壳自己所需要使用的API地址[41]。

如果用PE编辑工具查看加壳后的文件，会发现未加壳的文件和加壳后的文件的输入表(IAT:ImPort API Table)不一样，加壳后的输入表一般所引入的DLL和API函数很少，甚至只有kernel32.dll以及GetProcAddress这个API函数。

壳实际上还需要其他的API函数来完成它的工作，为了隐藏这些API，它一

般只在壳的代码中用显式连接的方式动态加载这些函数。

(2)解密原程序的各个区块(Section)的数据

壳出于保护原程序代码和数据的目的，一般都会加密原程序文件的各个区块。在程序执行时，壳将会对这些区块数据解密，以让程序能正常运行。壳一般是按区块加密的，那么在解密时也是按区块解密的，并且把解密的区块数据按照区块的定义放在合法的内存位置。

如果加壳时用到了压缩的技术，那么在解密之前还有一道工序，当然是解压缩。这也是一些壳的特色之一，比如说原来的程序鹰爪未加壳时为 1MB~2MB 大小，加壳后反而只有几百 KB。

(3)重定位

文件在执行时将会被映射到指定的内存地址中，这个初始内存的地址被称为基址(ImageBase)。当然这只是程序文件中声明的，程序运行时能保证系统一定满足要求吗?

对于 EXE 文件来说，Windows 系统会尽量满足。例如某 EXE 文件的基地址为 0x400000，而运行时 Windows 系统提供给程序的基地址也同样是 0x400000。在这种情况下就不需要进行地址“重定位”了。由于不需要对 EXE 文件进行“重定位，’，所以加壳软件一般会把原程序文件中用于保存重定位信息的区块干删除，这样使得加壳后的文件更加小巧。有些工具提供“WipeReloc”的功能，其实就是这个作用。

不过对于 DLL 的动态链接库文件来说，Windows 系统没有办法保证每一次 DLL 运行时提供相同的基地址。这样的“重定位”就很重要了，此时的壳中也需要提供进行“重定位”的代码，否则原程序中的代码是无法正常运行起来的。从这点来说，加壳的 DLL 比加壳的 EXE 更难修正。

(4)HOOK--API

程序文件中的输入表的作用是让 Windows 系统在程序运行时提供 API 的实际地址给程序使用，在程序的第一行代码执行之前 Windows 系统完成了这个工作。

壳一般都修改了原程序文件的输入表，然后自己模仿 Windows 系统的工作填充输入表中相关的数据。在填充过程序中，壳就可填充 Hook--API 的代码地址，这样就可间接地获取程序的控制权[43]-[46]。

(5)跳转到程序原入口点(OEP)

从这个时候起壳就把控制权交还给原程序了，一般的壳在这里会有明显的一个“分界线”。

2.3.3 常见加壳方式与常见壳

不同的壳所侧重的方面也不一样，有的侧重于压缩，有多侧重于加密。压缩壳的特点就是减少软件体积的大小，加密保护不是重点。目前兼容性和稳定性比较好的压缩壳有 UPX,ASPack，PECompackt 等。

2.3.2.1 压缩壳

UPX 是一个以命令行操作的可执行文件免费压缩程序，兼容性和稳定性很好[47]。UPX 包含 DOS,Linux 和 Windows 等版本，并且开源。官方主页：http://upx.sourceforge.net.

UPX 的命令格式为：upx [-123456789dlthVL][-qvfk][-o file] file.

UPX 早期版本压缩引擎是自己实现的，3.x 版本也支持 LZMA 第三方压缩引擎。UPX 除了对目标程序进行压缩外，也可解压缩。UPX 的开发近乎完美，它不包含任何反调试或保护策略。另外，UPX 保护工具 UPXRP，UPX-Scrambler 等可修改 UPX 加壳标志，使 UPX 自解压缩功能失效。

ASPack 是一款 Win32 可执行文件压缩软件，可压缩 Wind32 位可执行文件 EXE,DLL,OCX,具有很好的兼容性和稳定性。官方主页：http://www.aspack.com。

2.3.2.2 加密壳

加密壳种类比较多，不同的壳有不同的壳有不同的侧重点。一些壳单纯保护程序，另一些壳还提供额外的功能，如提供注册机制，使用次数，时间限制等等。加密壳还有一个特点，越是有名的加密壳研究的人越多，其被脱壳或破解的可能性也很大，所以不要太依赖壳的包含。加密壳在强度与兼容性上做的好的并不多。

ASProtect 是一款非常强的 Win32 位保护工具，这款壳开创了壳的新时代。它拥有压缩，加密，反跟踪代码，CRC 校验和花指令等保护措施。它使用 Blowfish、Twofish、TEA 等强劲的加密算法，还用 RSA1024 作为注册作为注册密钥生成器。它还通过 API 钩子与加壳的程序进行通信，并且 ASProtect 为软件开发人员提供 SDK，实现加密程序内外结合。SDK 支持 VC、VB、Delphi 等。

ASProtect 的创建者俄国人 Alexey Solodovnikov ,其将 ASPack 中的一些开发经验运用到 ASProtect 中。该壳的编写简单而精妙，是款经典之作。ASProtect 是注重兼容性和稳定性，因此没有采用过多的反调试策略。ASProtect 加壳过程中也可挂接用户自己写的 DLL 文件。

Armadillo 也称穿山甲，是一款应用面较广的商业保护软件。可以运用各种手段来包含你的软件。同时也可以为软件加上种种限制，包括时间、次数，启动画面等。官方主页为：http://www.siliconrealms.com。Armadillo 对外发行时有 Public、Custom 两个版本。Public 是公开演示的版本，Custom 是注册用户拿到的版本。只有 Custom 才有完整的功能，如强大的 Nanomites 保护。Public 版有功能限制，没有什么强度不建议采用。

Armadillo 有如下保护功能：Nanoites、Import Table Elimination、Strategic Code Splicing、Memory-Patching Protections 等。其中 Nanomites 功能最为强大，使用时，需要在程序里加入 Nanomites 标签，如下面这段 VC 编译器里定义的标签：

#define NANOBEGIN _asm-emit 0xEB _asm_emit 0x03 _asm_emit 0xD6 _asm_emit 0xD7 _asm_emit 0x01

#define NANOEND _asm-emit 0xEB _asm_emit 0x03 _asm_emit 0xD6 _asm_emit 0xD7 _asm_emit 0x00

用 NANOBEGIN 和 NANOEND 标签将需要保护的代码括住，Armadillo 加壳时，会扫描程序，处理标签的跳转指令，将所有跳转指令换成 INT 3 指令，其机器码是 CC。此时 Armadillo 运行时，是双进程，子进程遇到 CC 异常，由父进程截获这个 INT 3 异常，计算出跳转指令的目标地址并反馈给子进程，子进程继续运行。由于 INT3 机器码是 CC，因此也称这种保护是 CC 保护。

EXECryptor 是一款商业保护软件，官方主页：http://www.strongbit.com。其可以为目标软件加上注册机制，时间限制，使用次数限制等附加功能。这款壳的特点是 Anti-Debug 比较强大，同时做的比较隐蔽，另外就是采用了虚拟机保护一些关键代码。要使这款壳有强大的保护，必须合理使用 SDK 功能，将关键功能代码用虚拟机保护起来。

Themida 是 Oreans 的一款商业保护软件，官方链接:www.oreans.com。Themida1.1 以前的版本带驱动，稳定性有些影响。Themida 最大的特点就是其虚拟机保护技术，因此在程序中擅用 SDK，将其关键代码让 Themida 用虚拟机保护起来。Themida 最大的特点就是生成的软件有些大。WinLicense 这款壳和 Themida 的同一个公司的一个系列产品，WinLicense 主要多了一个协议，可以设定使用时间，运行次数功能，两者核心保护是一样的。

2.4 小结

本章首先对计算机病毒感染文件做了大量介绍，然后分析了 Win32 和 Win64 病毒对 Winodws 平台的依赖关系，最后介绍了病毒加壳的过程和压缩壳与加密壳，为第三章核心算法理论知识做了良好的铺垫。

第三章 基于 PE 文件的启发式特征码自动提取技术

本章基于第二章中介绍的相关内容做铺垫，首先对特征码提取技术问题进行分析。其中包括问题是什么，难点是什么，以及解决问题的思路。接着从 Win32 病毒感染策略的角度探讨启发式分析，以及如何根据感染策略寻找启发性标志。提取基于 PE 文件启发性特征码提取技术。最后探讨如何将启发性标志中提取的特征序列组合成特征码。

3.1 特征码提取技术问题分析

3.1.1 特征码提取和其原则

恶意代码特征码提取的依据是通过逆向工程分析恶意代码，只有该种恶意代码才会具有的特殊代码。

逆向工程主要是采用反汇编的技术，将新的软件进行分析，有可疑行为。从中提取出一段以判断其是否具识别该可疑文件与其他文件的差异代码。

恶意代码特征码的提取是特征码检测法的关键，特征码不能任意选取，而要求能够准确无误报的实现检测，选择代码串的规则是:

(l)短小的恶意代码只有一百多个字节，长的恶意代码有上 10KB 字节的。如果随意从恶意代码体中选一段作为代表该恶意代码的特征代码串，可能在不同的环境中，这个特征串并不真正具有代表性，不能使将该串所对应的恶意代码被检查出来。选这种串作为恶意代码库的特征串便是不合适的。

(2)代码串不应含有恶意代码的数据区，因为数据区是经常变化的。

(3)在保持唯一性的前提下，应尽量使特征代码长度短些，以减少时间和空间开销。如果一种恶意代码的特征代码增长一个字节，要检测 3000 种恶意代码，增加的空间就是 3000 字节。

(4)一定要在仔细分析了程序之后才能选出最具代表性的代码串，使之成为将该恶意代码区别于其他恶意代码和该恶意代码的其他变种的代码串。

选定好的特征代码串是不容易的，是恶意代码检测程序的精华所在。一般情况下，代码串是连续的若干个字节组成的串，但是有些检测软件采用的是可变长

串，即在串中包含有一个到几个“模糊”字节。检测软件遇到这种串时，只要除“模糊”字节之外的字串都能完好匹配，则也能判别出恶意代码。例如给定特征串:“Eg7C0010?37CB”则“Eg7C00102737CB”和“Eg7C0010gC37CB”都能被识别出来。又例如:“Eg7C37CB”可以匹配“Eg7C0037CB”,“Eg7C001137CB”，和“Eg7C00112237CB”，但不匹配“Eg7C001122334437CB”因为 7C 和 37 之间的子串已超过 4 个字节。

(5)特征串必须能将恶意代码与正常的非恶意代码程序区分开。如果非恶意代码程序当成恶意代码报告给用户便是误报。误报越多，说明检测程序特征码提取存在问题。

3.1.2 特征码提取技术的难点

病毒编写者与防病毒人员之间的博弈是不断发展和演化的，当通过特征码检查未知病毒技术作为各杀毒厂商目前主流的病毒检测方法时，病毒编写学者们也在不断研究新的技术，以躲过特征码的检查。由此出现了多态病毒和变形病毒，这些病毒在复制自身的同时能变换自身的代码结构，或通过加花指令，废操作的形式绕过特征码的检测。在这种情况下对病毒特征码的提取形成了一下难点：

1.如何提取有效的特征信息作为病毒的特征码成为一个难点问题。提取的特征信息既要保证能代表此种病毒的特性，又要保证该特征信息在病毒属于不会被改变的部分。

2.如何自动提取病毒的特征码是存在于解决病毒特征码提取效率方面的另一个难点。现在主流的特征码提取法仍停留在人工提取的阶段。虽然工提取的特征码性能上很好。但提取病毒特征码需要病毒专业人员进行手工分析，且特征码提取的效率低下，耗费的成本很高。

3.特征码的质量是特征码提交技术的又一个难点。提取的特征码必须具有低误报率，漏报率。并且能够有效通过提取的特征码检测出该病毒的以及其变种病毒。

基于上述病毒特征码提取的难点，本章接下来首先介绍现有特征码提取技术，在分析它们的优缺点后，提出基于 PE 文件启发式特征码提取技术的思路。

3.1.3 常见特征码提取方法以及其优缺点

3.1.3.1 常见特征码提取方法

1 字符串扫描法

字符串扫描时提取计算机病毒特征码最简单的方法。它使用从特停病毒体中提取出的特征字序列，即 16 进制字符串进行检测，这些字符串不太可能出现在无毒的程序代码中。这些字符串从病毒中提出出来后，被存入数据库，供病毒扫描引擎用来和扫描允许的有限时间按部就班地对文件中的预订区域和系统区域进行扫描。事实上，防毒扫描引擎中最具挑战性的任务之一就是如何聪明的利用这段有限的时间,通常一个文件不超过秒数来成功地找出病毒。

请看下面中利用 IDA（交互式反汇编器）反汇编得到的引导型病毒 Stoned 的一个变种的代码片段

```
Seg000:7C40   BE 04 00          mov si,4      ;try it 4 times
Seg000:7C40                                                  ;
Seg000:7C43
Seg000:7C43             next;              ; CODE XREF:sub_7c3A+27
Seg000:7C43   BB 01 02          mov ax,201h       ;read one sector
Seg000:7C46   0E                push cs
Seg000:7C47   07                push es
Seg000:7C48                     assume es:seg000
Seg000:7C48   BB 00 02          move bx,200h      ;to here
Seg000:7C48   33 C9             xor cx,cx
Seg000:7C4D   8B 01             mov dx,cx
Seg000:7C4F   41                inc  cx
Seg000:7C50   9C                pushf
Seg000:7C52   2E FF 1E 09 00    call dword ptr cs:9    ;int 13h
Seg000:7C56   73 0E             jnb short Fine
Seg000:7C58   33 C0             xor ax, ax
Seg000:7C5A   9C                pushf
Seg000:7C58   2E FF 1E 09 00    call dword ptr cs:9    ;int 13h
Seg000:7C60   4E                dec si
Seg000:7C61   75 E0             jnz short next
Seg000:7C63   EB 35             jmp short giveup
```

该代码尝试读取软盘引导扇区最多达 4 次，在每两次尝试之间对磁盘控制器的硬件进行复位，如上代码第一次调用 int 13h 时，AH=02，即读取扇区；第二次调用 int 13h 时，AH=0，即复位磁盘控制器。

这是一段典型的病毒代码。读取软盘初始扇区的 4 次尝试是必要的，因为老式软盘驱动器速度太慢会跟不上。病毒使用 PUSH CS 和 POP ES 这对指令把磁盘缓冲区指向病毒代码段。

注意这段代码似乎想对设置 CX 和 DX 寄存器的指令进行优化，但其优化方式并不成功。CS 和 DX 寄存器都是磁盘中断处理程序的参数。

因此从Stoned病毒中提取出的一种特征模式就可能是如下的16个字节，《Virus Bulletin》杂志也曾刊登过这一特征码字符串：

0400 B801 020E 07BB 0002 33C9 8BD1 419C

通常，16 个字节病毒特征码对于检查 16-恶意代码已经足够可靠了，不会引起虚警。因此毫不奇怪，像《Virus Bulletin》这样的计算机病毒研究期刊过去发布的一般都是 16 字节的特征序列，这个长度足以检查引导区病毒。但要想可靠地检测 32 位病毒，需要用更长的字符串。上述代码片段也可能出现在 Stoned 病毒的其他变种中。实际上，该病毒时几个近似变种就可以用上述特征码字符串来检查。另外，这个字符串也许还能检测出一些属于其他家族但与 Stone 关系密切的病毒。一方面，这是一个优势，因此扫描器用该字符串能检测出更多的病毒；但另一方面，对用户来说这也可能成为严重的缺点，因为扫描器也许把一个完全不同病毒误判成了 Stoned 病毒。因而，用户可能就会认为该病毒的危害比较小，但其实这个被误判的病毒危害性可能比预期的要大的多。

识别上出问题，可能导致数据受损。当扫描器检测到有病毒，但未识别出是什么样的病毒时，如果尝试去清除，就容易出现这个情况。

2 通配符提取法

一般的特征码提取器常常支持通配符。通配符一般用于跳过某些字节或字节范围，有些扫描器还允许正则表达式。

0400 B801 020E 07BB ??02 %3 33C9 8BD1 419C

上述字符串被解释为：

尝试匹配 04，如果找到则继续；

尝试匹配 00，如果找到则继续

尝试匹配 B0，如果找到则继续

尝试匹配 01，如果找到则继续

尝试匹配 02，如果找到则继续

尝试匹配 03，如果找到则继续

尝试匹配 07，如果找到则继续

尝试匹配 BB，如果找到则继续

忽略此字节，尝试匹配 02，如果找到则继续

在接下来的 3 个位置（字节）中尝试匹配 33，如果找到则继续

尝试匹配 C9，如果找到则继续

尝试匹配 8B，如果找到则继续

尝试匹配 D1，如果找到则继续

尝试匹配 41，如果找到则继续

尝试匹配 9C，如果找到则报告发现病毒。

通配符常常支持半字节匹配即匹配一个十六进制位。这样可以更精确地匹配指令组。一些早期的加密病毒甚至到现在的多态病毒用带通配符的字符串很容易检测出来。

显然，就算单独使用 Boyer-Moore 算法，对于字符串扫描工具来说效率也是不够的。这种算法是为了字符串的快速搜索而开发的，它利用了反向字符串匹配。而如果从字符串结尾开始比较，则第一次比较就会发现不匹配，这显著减少了所需比较的次数，因此很多匹配的位置可以被自动略过。

3 不匹配字节数提取法

字符串中的“不匹配字节数”是为 IBM Antivirs 开发的一种特征码技术。其原理是允许字符串中有 N 个字符为任意数值，而不论其在字符串中的什么位置。如果字符串 01 02 03 04 05 07 08 09 的“不匹配字节数”如果取值为 2，则可以匹配图 3-1 中任何一个模式：

```
01  02  AA  04  05  06  BB  08  09  0A  0B  0C  0D  0E
0F  10  01  02  03  CC  DD  06  07  08  09  0A  0B  0C
0D  0E  0F  10  01  EE  03  04  05  06  07  FF  09  0A  0B
                0C  0D  0E  0F  10
```

图 3-1 一组“不匹配字节数”为 2 的字符串

“不匹配字节数”法特别有助于为同一家族的计算机病毒开发出更好的通用检测方法，但缺点是其扫描速度相当慢。

4 散列

散列是一个常见术语，指能加速特征码搜索算法的一些技术。散列可以运用在扫描特征码字符串的首字节或第一个 16/32 位字进行计算。这种允许字符串中后

面的字节包含通配符。病毒研究人员通过精心选择字符串的前几个字节内容，就可以更好的控制散列。例如，“避免采用常规文件开头的那些常见字节，如一些零字节”就是一个很好的方法。如果研究人员再下点功夫，还可以选择开头部分有一些相同字节的字符串，从而减少所需的比较次数。

为获得最快的检索速度，有些字符串扫描器不支持任何通配符。例如，澳大利亚的 VET 防病毒软件就用了 Roger Riordan 的一项发明，该发明使用 16 字节的扫描字符串、一个 64KB 的散列表和一个 8 位移位寄存器。该算法把字符串中每个 16 位字用作散列表索引来进行检索。

Frans Veldman 为其 TbScan 扫描器开发了一种有效的散列技术。该算法允许字符串中有通配符，但采用了两个散列表和一个相应的字符串链表。第一个散列表中包含第二个散列表的索引位，该算法使用了扫描字符串中的四个不含通配符的 32 位字常量。

5 入口点和固定点提取法

入口点和固定点扫描程序是进一步提升了防毒工具的特征码提取和检测性能。此类特征码提取对比程序利用了一些对象（如那些通过可执行文件头部能访问到的对象）的入口点。在无结构的二进制可执行文件中，此类扫描程序将跟踪转移控制权的各种指令，并从这些指令指向的位置开始扫描。

由于这种位置是计算机病毒常见的攻击目标，所以此类特征码提取比对程序有很大优势，其他的方法必须用扫描特征码串与被扫描区域的每个扫描位置进行匹配，而入口点特征码提取对比程序通常只需要扫描字符串在一个位置进行匹配，这个位置就是入口点本身。

试想一个 1KB 长的缓冲区，称它为 B。在 B 中可以从（1024-S）个起始位置开始做字符串匹配，其中 S 是待匹配的最短字符串长度。即使扫描程序的散列算法效率很高，使得只有 1%的情况下扫描器需要在给定位置执行完整的字符串扫描操作，计算量也可能随着字符串数量增长而很快增长。例如，如果有 1000 个字符串，扫描器可能需要对每个位置执行 10 次完整匹配。因此（1024-S）去掉。结果与原来的计算量差别很大的。

如果入口点的字符串不够好，则可以用固定点特征码提取和扫描法来帮忙，固定点特征码扫描法对每个字符串值在一个位置进行匹配，因此就可以设置一个其实位置 M,例如文件的主入口点，然后在距此入口点 M+X 字节的位置匹配每个字符串。由于 X 通常是 0，因此所需计算次数再次减少。此类方法还有一个额外的好处：可大幅度降低磁盘的 I/0 次数。

由于每个字符串的长度是固定的，此算法也可以检测字符串的尾字节是否与当前被扫描文件中对应位置匹配。只有当字符串尾字节匹配时，才进行完整的字符串匹配操作，但这种情况在实际中很少发生。该技术类似于把 Boyer-Moore 算法和简单的散列法结合起来。

3.1.3.2 常见特征码提取法的优缺点

字符串扫描法可能导致识别上的问题，可能将两个不同类型的病毒识别为同一种病毒，这样会造成病毒清除时损坏正常文件。

散列法会带来数据的比对次数太多造成比对程序的低效率

入口点和固定点提取特征码，虽然在效率上有所改进但对变形病毒和多态病毒的检测效率低下。

加上现在主流的特征码提取方法还停留在人工提取的阶段，虽然提取的特征码准确，误报率低，但效率问题是制约人工特征码提取的一个重要因素。由于变形病毒的出现，病毒的数量与日俱增，使人工提取特征码的方法已经不能跟上病毒出现的数量。

3.1.4 基于启发式特征码提取的思路

综合各种常见特征码提技术的优缺点，我们提出了基于 PE 文件启发式特征码自动提取算法。该算法的具体思路为：

1.通过深入研究病毒感染策略的基础上分析 Win32 病毒如何感染 PE 文件结构。

2.通过病毒的感染策略和大量病毒分析探索性的设置 PE 结构中的启发性标志。

3.在对一个病毒样本扫描时，逐个寻找其 PE 结构中的启发性标志。如果找到启发性标志，则针对该标志提取相应的特征码序列，

4.选取最能代表该病毒特征的特征序列进行特征码序列组合，形成特征码。

5.通过大量实验验证启发性标志的健壮性，修改并继续寻找新的启发性标志。

具体的特征序列提取方法和特征码序列组合方法将在 3.3 节和下一章系统设计与实现中详细阐明。

3.2 从 Win32 病毒感染策略到启发式分析

3.2.1 Win32 API 及其支持平台

1995 年，微软推出了 Windows 95 作为其重要的新一代操作系统平台。尽管 Windows 强烈的依赖 Windows 3.x 和 DOS 技术，但正是它令 Win32 这个术语具有了实际的意义。

Win32 就是一套 API 的名字。从一个 32 位 Windows 应用程序中可以调用的各种系统函数包含在 Win32 API 中。实现 Win32 API 的平台有几种。其中包括 Windows NT 这个最为重要的 Win32 平台。除了 Dos 程序外 Windows NT 程序还能执行 16 位 Windows 程序、OS/2 1.x 字符模式程序，另外在一些扩展支持下，甚至可以有限地运行基于 Presentation Manager 1.3 的程序。此外 Windows NT 引入了可以运行 Win32 的应用程序（即调用 Win32 API 函数的应用程序）的新的文件格式，可移植执行程序。该格式就算不是基于 UNIX 的 COFF 格式，也与它非常的类似。正如可移植一词表明的那样，PE 格式是一种容易移植的文件格式。如今这种格式的确是 Windows NT 系统上最常见和最重要的可执行文件格式。

其他的平台也能运行 Win32 应用程序。实际上，一种名为 Win32s 的平台比 Windows NT 推出的还早。任何曾尝试给 Win32s 开发过软件的人都知道它是一种很不稳定的解决方案。

Windows NT 是一种鲁棒性很好的系统，但需要很强的硬件来支持，因此 Win32 技术并未迅速获得微软期望的市场地位，这个过程最后导致 Windows 95 的开发，Windows 95 缺省支持这种新的 PE 格式。因此，它就支持一个特殊的 Win32 API 集。Windows 95 比起 Win32s 来说，是一种好的多的 Win32 API 实现。但是，Windows 95 包含的 Win32 API 实现并不如 Windows NT 中的完整。

在 Windows NT 获得更多的发展动力之前，Windows 9x 一直是微软 win32 平台。在 Windows NT 之后，Windows 2000 和 Windows 98/Me 获得普及，后来 Windows XP 和更安全的 Windows 2003 服务器版有取代了它们，现在 Windows 7 是它们的后继者。所有这些系统都将支持 Win32 API 的表单，即，多数情况下可以保证各种系统上的 Win32 API 二进制的兼容性。

最后 Windows CE 也支持 Win32 API 和 PE 格式。Windows CE 主要支持手持 PC。

Windows NT 和 Windows CE 都能在采用不同 CPU 的计算机上运行。PE 文件格式也可以用在不同机器上，只是实际运行的代码是为目标处理器编译得到的二进制代码，而 PE 头部包含了目标处理器的类型信息。所有这些平台都包含 Win32 函数的不同实现，各种实现中都有大部分的 Win32 函数。因此无论在什么平台上运行，应用程序都可以调用这些函数。Win32 API 各种实现间的差别大多是与操作系统的实际能力和可用硬件资源相关的。例如，在 Win32s 中那个调用 GreateThread() 就简单地返回 NULL；Windows CE 实现的 API 集包含了几百万个函数，但它对一些不重要的函数如 GetWindowsDirectory()根本不提供支持，因此在设计上 Windows CE 内核是放置在手持 PC 的 ROM 中的。由于手持 PC 的硬件上的严重限制，微软被迫开发了一个比 Windows NT 和 Windows 95 规模小的新操作系统。

尽管 Win32 API 的几种实现中，部分函数存在差别，或者有的函数根本没有实现，但总体来讲还是可以做到编写一个程序，然后在各种支持 Win32 API 的系统上都能用。病毒编写者们已经非常熟悉这个事实了。他们最早开发的那些病毒专门攻击 Windows 95，但渐渐地，病毒作者们也在改进攻击 PE 文件格式的技术，使得被感染文件仍然和 Windows NT/2000/XP 兼容，并能在这些系统上正确的运行。

多数 Windows NT 病毒都依赖 Windows NT 系统的行为和功能，比如与虚拟设备驱动程序及虚拟机管理器相关的那些特性，但这些病毒中有些只包含少数的缺陷，而且稍微修改就能运行在其它 Win32 平台上。

这类病毒的检测和清除不是很容易。特别是清除病毒可能很难实现。原因在于迄今为止，PE 格式比使用的任何其它可执行文件格式都要复杂的多。但是，PE 格式在设计上也确实比其他格式要好的多。

另外 Win64 除了拥有 64 位 Windows 体系外，基本上与 Win32 一样。为解决平台间的差异问题，相当于 Win32，Win64 做了少量修改。

3.2.2 从 32 位 Windows 感染技术到启发式分析

32 为 Windows 感染技术是启发性分析的基础，启发性标志是通过深入剖析 Windows 感染技术的基础上设定的。只有深入理解了 Windows 感染技术的机理才能设计出优秀的启发性标志。所以本节在对 32 位 Windows 感染技术进行探讨的同时提出设置启发性标志的思路。

1 头部感染型病毒

在 Windows NT 病毒的 PE 头部的末尾（节表之后）和第一个节开始之前插入病毒代码。它把 PE 头部的 AdreesOfEntryPoint 域改为指向病毒的入口点。已知的第一个使用此技术的病毒是 Win95/Murkery。

Windows NT 中的头部感染型病毒必须非常短。由于节的起始点必须位于 FileAlignment 域取值的整数倍位置，因此病毒在宿主文件中可以覆盖掉的区域小于 FileAlignment。当一个应用程序包含的节太多但其 FileAlignment 值只有 512 字节时，病毒代码就没有可用的空间了。AdreesOfEntryPoint 域是一个相对虚拟地址（RVA），然后此类病毒的代码并不位于任何节中，因此该 RVA 就是（病毒代码入口点）在文件中的真实物理偏移量，病毒一定会把这个偏移量写入头部的 AdreesOfEntryPoint 域。尽管入口点并未指向任何代码节，但 Windows NT 的装入过程仍会很乐意地执行被感染程序。

扫描器如果只用第一代头部感染型病毒的样本测试过，就有可能不能检测出第二代此类病毒。第一代样本中，AdreesOfEntryPoint 域指向的是一个有效的节。扫描器在寻找程序的入口点时，必须检查所有的节头部以及 AdreesOfEntryPoint 是否指向这些节头部中的哪一个。第二代病毒中入口点可以不指向任何的节，如果扫描器未实现处理这种情况的能力，就会跳过染毒文件，而不是从真正的入口点对其进行扫描，因此无法检测出第二代病毒造成的感染。

对于头部感染型病毒可以通过检查 AdreesOfEntryPoint 域是否指向 PE 头部来设定启发性标志。

2 前置病毒

感染 PE 文件最简单的方法就是重改其开头部分。因为前置病毒要处理复杂的 PE 格式。这类病毒通常使用高级语言编写，如 C 甚至 Delphi 编写的。前置感染就是把病毒代码放置到 PE 文件之前。被感染程序从病毒的 EXE 头部开始执行。当病毒想把控制权传给原始程序代码时，它会把原始程序从染毒文件中提取出来，写入一个临时文件，然后执行它

这类病毒很容易清除，因为原始的头部信息位于感染程序的最后面，而且未加密。

对于前置型病毒，我们可以通过检查 PE 文件是否具有多个 PE 头部来设定启发性标志。

3 不增加新的节头部的追加病毒

不增加新的节头部的追加病毒的感染机制类似于 Boza，它不会在节表末尾增加新的节头部，而是修改最后一个节头部，然后把自己安置到该节中。这样该病

毒很容易就可以感染所有PE可执行文件，而如果采用新的节头部的话，则要担心是否能将其加入到节表中。病毒通过修改VirtualSize和SIzeOfRawData域，就可以把自己的代码放到可执行文件的最后，这样就不需要修改PE头部的NumberOfSection域了。病毒还把AdressOfEntryPoint域改为指向病毒体，并重新计算SizeOfImage域以反映出程序被感染后的大小。

最后一个节头部的Characteristics域被加入了“可写/可执行”属性。任何一个节头部只要有了“可写”属性，就能运行该节中的自修改代码，但是很多病毒作者刚开始没有意识到这一点。

像W32/Zelly这样的病毒采用了两种或更多中的感染策略。Zelly的基本感染模式是在宿主程序末尾。这就把病毒体和宿主程序更紧密的结合到一起了。

对于追加型病毒，我们可以通过检查节表的Characteristics属性和AdressOfEntryPoint判断等来设定启发性标志。

4 不修改入口点的追加病毒

有些Windows NT病毒不修改被感染程序的AdressOfEntryPoint域。这些病毒把自身的代码追加到PE文件的后面，但其获取控制权的方式却比较复杂。它计算AdressOfEntryPoint域指向什么位置，然后再该位置放置一条指向病毒体的跳转指令。幸运的是，要编写这种病毒难度很大。

这是因为这种病毒必须处理指向宿主代码中被改写区域的重定向条目，W32/Cabanas病毒把指向那个区域的重定向体条目屏蔽掉。Marburg病毒如果发现这种重定向条目，则不会在入口点放置JMP指令，而是修改AdressOfEntryPoint域。JMP指令不应该是程序中第一条指令。Marburg的如下做法表明了这一点：当AdressOfEntryPoint代码的前256个字节没有重定向条目时，该病毒把JMP指令放置到一个随机的无用指令代码块末尾。这样，对扫描器和完整性检查工具来说找出病毒代码的入口点就不那么容易。

对于不修改入口点地址的追加病毒，我们通过先定位程序的真实代码段，然后查找代码段前后是否有跨段JMP指令来设定启发性标志。

5 感染KERNEL32.DLL的病毒

多数Windows NT病毒攻击的都是PE格式文件。感染KERNEL32.DLL的病毒不会攻击该库文件的入口，而是用别的办法获得控制权。PE文件中很多别的入口点可以被病毒利用，尤其DLL本质上就是一些导出API(它们的入口点)。因此，攻击KERNEL32.DLL的最简单办法就是修改其中的某个API的导出RVA以指向位于DLL映射末尾的病毒代码Lorez用的就是这个方法，这种病毒可以很容易地

常驻内存。系统的初始化期间会把被感染的 KERNEL32.DLL 装入内存，当程序中调用的某个 API 连接到病毒代码时，病毒就获得了控制权。

链接程序会在所有系统 DLL 文件的 PE 头部放置一个预先计算好的校验和。Windows NT 在加载 DLL 前会重新计算这个校验和。如果得到的值和 DLL 头部中保存的值不一样，则系统装入程序就会在蓝屏启动阶段停止运行并显示一条错误信息。但是，这并不意味着此类病毒不能在 Windows NT 上实现，他们仅仅是稍许增加了实现的复杂性。尽管微软并未公布校验和的算法，但在 IMAGEHLP.DLL 中有计算校验和的 API（如 CheckSumMappedFile()）,病毒在感染完成后，用来这些函数完全可以重新计算出一个正确的校验和。但对于 Windows NT 的装入程序来说，仅有这一点还不够，还有其它步骤需要完成，但是毫无疑问，病毒作者们很快就能解决这些问题。病毒扫描器应该重新计算 PEtoubu 的校验和来检查 KERNEL32.DLL 的一致性，特别是当扫描器本身就是一个 Win32 程序而且连接到已感染的 KERNEL32.DLL 时，更需要这样做。

对于感染 KERNEL32.DLL 的病毒可以通过分析导入表来设定启发性标志。

6 Dll 加载插入技术

这种特殊的感染技术的通过修改 PE 文件，以便当宿主程序被加载时，还会被额外加载了名为 INITX.DAT 的 DLL。这个额外的 LoadLibrary()代码被插入的宿主代码节的空闲空间中，病毒修改宿主入口点以指向这个新插入的代码。当执行宿主程序时，只要系统中有 INITX.DAT 文件，该文件中的病毒代码就会在宿主程序代码之前执行。此后，控制权才会装给原始的宿主入口点代码。

对与采用 DLL 加载插入技术的病毒，我们可以通过检查入口地址和导入表等方法来设置启发性标志。

3.3 基于 PE 文件启发式特征码提取技术

启发式指的“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。”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 PE 文件结构分析和扫描器，通过前面对病毒感染策略的分析，提出了病毒可能修改和控制的 PE 结构的启发性标志，当然我们的对象是已知的病毒样本，这些病毒样本与正常文件的 PE 结构一定会有不同的区别，这是由病毒复制自身和传播的特性决定的，找到这些容易被病毒修改的 PE 结构，将其设置为启发性标志，在提取每个病毒样本时，依次分析这些病毒样本是否具有启发性标志，通过启发性标志生

成特征序列，然后对特征序列进行筛选，用那些最能代表病毒特征的特征序列组成特征码。

3.3.1 启发式特征码提取法应用范围

启发式特征码提取法经过试验证明是能够成功提取新病毒的一种手段。基于PE文件启发式特征码提取方法面临的挑战也显而易见，经常导致虚警是其面临的最大挑战。虚警会浪费用户和分析人员的时间和精力。所以要尽量使启发式特征码提取的算法最大程度的接近病毒的感染手段和特征（病毒的感染特征在2.2节和3.1节都有详细的分析）。

例如用启发式特征码提取法来提取宏病毒的特征码，会得到非常好的效果。同时用启发式特征码提取法来提取二进制病毒效果也会非常好，但在检查二进制病毒时启发式特征码提取法引起的虚警往往比用启发式特征码提取法提取宏病毒要高。

所以解决虚警的问题，必须对启发式特征码提取法的启发能力和启发性标志的设定进行大量实验并控制其在一个能够接受的范围内。启发式特征码提取法不是一种孤立的方法，它必须与是否很好地理解了病毒的感染手段密切相关。对不同类型的病毒，需要用完全不同的规则来构建启发式特征码提取法的判断逻辑。

显然，用于捕获Linux或DOS病毒或引导性病毒的启发式特征码提取法不能用于提取Win32病毒。本节将介绍Windows病毒启发式特征码提取法的主要算法。

启发式特征码提取法从PE文件分析入手，然后寻找可疑的代码组合，下一小节将着重讨论在PE文件中寻找启发性标志，它们大部分不是基于代码仿真，但反映出32位编译器（如Mircrosoft、Borland或Watcom的编辑器）编译得到的PE文件中不可能出现的特定的结构问题。这种方法能检测出像Marbur或HPS这样的多态病毒。

如果一个程序是病毒，那么它一定有与正常文件不同的特征形态，记录它们之间的显著区别，并作为提取特征码的一种方法。

3.3.2 从病毒感染策略分析PE文件

对于可预见的未来，微软各种操作系统中的PE文件格式都将扮演一个关键的角色。众所周知，Windows NT有VAX VMS和UNIX的血统。如前所闻所述，PE格式非常类似于通用对象文件格式（common object file format,COFF），只是对

COFF 做了一些更新。其名称为可移植是因为多种平台都使用该文件格式。

关于 PE 格式，需要知道的最重要的就是：这种可执行代码在磁盘上时和 Windows 将其装入内存准备执行时的结构非常相似。这使系统装入程序需要完成的工作变的很简单。在 16 位 Windows 中，代码执行前必须由装入程序花很长时间进行准备。这是因为 16 为 Windows 应用程序中，所有调用了外界动态链接库的函数都必须被重定位。有些很大的应用程序可能包含成千上万的 API 调用重定位信息，系统装入程序时必须在分段读取文件内容并未逐一分配内存时，修改这些重定位信息。PE 应用程序不再需要为函数库调用进行重定位。系统装入程序使用 PE 文件中的导入地址表这个特殊区域来完成那个功能。IAT 在 Win32 病毒采用的感染技术中起到了关键作用。

Win32 程序的代码、数据、资源、导入表和导出表使用的是内存中一块连续的线性空间。可执行文件只须知道装入程序把自己映射到内存中什么位置即可。知道了基地值后，就可以根据文件印象中的存储的指针轻松地找到文件的各个部分。

应该熟悉的另一个概念是相对虚拟地址。PE 文件中的很多域都是用 RVA 来指明的。RVA 就是表示该域到文件的内存映像起始位置的偏移量。例如，Windows 装入程序可能会把 PE 文件映像中的某个域 0x401234 地址开始，则该域的 RVA 就是 0x1234。

研究 PE 文件及感染这种文件的病毒时，还有一个要熟悉的概念是节。PE 文件的一个节大体等同于 16 位 NE 文件中的一个段。节中可能包含代码，也可能包含数据。有些节包含了完成程序实际功能所需要的代码或数据，而其他的数据节则包含了对操作系统来说重要的信息。在深入讲述 PE 文件的重要细节前，请参看图 3-2，它显示了 PE 文件的整体结构。

1.PE 头部

PE 格式中的第一个重点是 PE 头部。与微软所有其他可执行文件格式一样，PE 文件也有一个头部区域，其中包括了一组位置明确的域。PE 头部记录了对可移植执行文件映像来说必不可少的信息。PE 头部并不位于文件的最开头，因为那个位置存放的是老的 DOS 存根程序。

DOS 存根程序就是一个很小的 DOS EXE 程序，它会显示一条错误信息（通常是“This program cannot be run in Dos mode”）。由于这个 DOS 头部位于 PE 文件开始，因此有些 DOS 病毒通过感染这个 DOS 头部也可以正确感染 PE 文件。但是 Windows NT 的装入程序需要把 PE 文件作为 32 位程序才能正确执行，因此，DOS 存根程序对 16 位 Windows 系来说，仍然存在着兼容性问题。

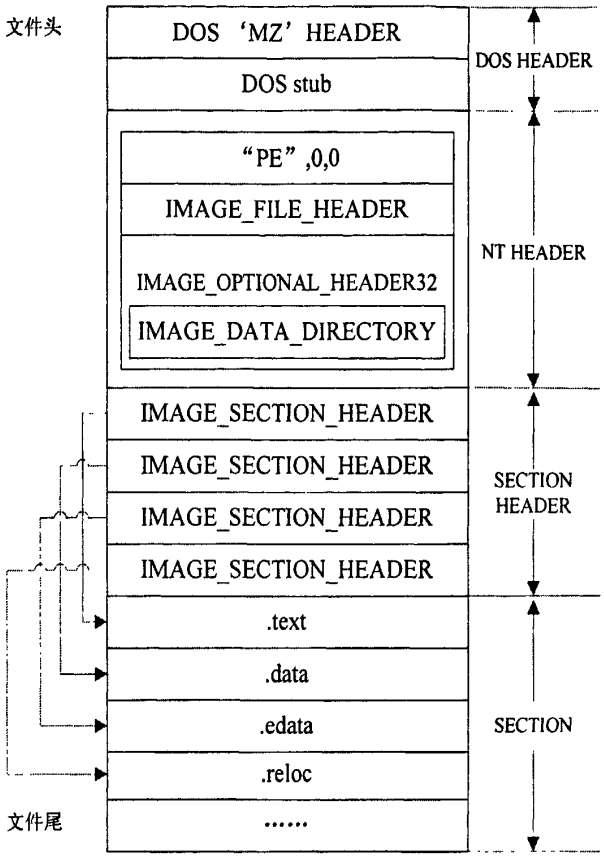

图 3-2 PE 文件结构

装入程序从 DOS 头部的 Ifanew 域取出 PE 头部的文件地址。PE 头部以一个重要的魔数 PE\0\0 开始，后面是文件头部和可选头部。

下面将只讲述 PE 头部中与 Win32 病毒病毒相关的重要的域。这些域是顺序排列的，但本文将重点讨论最常见的取值，因此有几个域就不讨论了。图 3-2 显示了 PE 文件映像的简要的结构图。

下面列出了 PE 文件头的几个重要的域。

Machine 域（字，WORD）

说明本程序意欲在什么 CPU 上运行。很多 Win32 病毒在实际感染文件前，检查文件的这个域中是否有对应于 Intel i386 的魔数。但是，有些劣质的病毒并不检查机器的类型，感染了本来是要在其他平台上运行的 PE 文件，结果当病毒代码在错误的平台上执行时，这些文件就崩溃了，未来将可能出现支持多种处理器的病毒，例如一个病毒可以攻击 ARM，也可以攻击 IA64 和常规的 X86PE 文件。

NumberOfSections 域

说明 EXE 或 DLL 文件中有多少个节。病毒使用这个域的原因千差万别。比如有的病毒在 PE 映像中添加了一个新节用于存放病毒体，这种病毒就会增加 NumberOfSections 域的取值。基于 Windows NT 的系统在一个 PE 文件中可以接受的节的最大数量为 96。

Characteristics 域（字，WORD）

关于文件信息的一些标志。多数病毒都会检查这些标志以确保这个可执行映像不是一个 DLL 而是一个程序。通常病毒不会修改这个域。

下面是可选头部的开始是一个魔数域。有些病毒会检查此域的取值，以确保当前普通的可执行文件而不是 ROM 映像或别的什么。

SizeOfCode 域（双字，DWORD）

记录了全部可执行代码大小的总和。当前的病毒在向宿主程序添加的代码节时，通常不会修改这个值，但未来的病毒可能会这样做。

AddressOfEntryPoint 域（双字，DWORD）

这个入口点地址域说明映像从哪里开始执行。其取值通常是一个指向.text 节的 RVA。这个对多数 Win32 病毒时至关重要。多数已知的病毒感染技术都会修改这个域，令其指向病毒代码的实际入口点。

ImageBase（双字，DWORD）

当连接程序生成一个 PE 可执行文件时，它认为该文件映像将被映射到一个特定的内存地址。那个地址就存储在 ImageBase 域中。如果文件映像能被加载到指定地址（当前对微软可执行文件来说 0x400000），就不需要装入程序来修成重定位信息。多数病毒在感染 PE 文件前都要用这个域的值来计算某些内容的实际地址，但它们一般不会修改这个域的值。

SectionAlignment 域（双字，DWORD）

当可执行文件映射到内存中时，每个节起始的虚拟地址都必须是本域取值的整数倍。这个域最小为 0x1000，但 Borland 公司的连接程序使用的缺省值大的多。如 0x10000.多数 Win32 病毒都用这个域来计算病毒体的正确位置，但不会改变这个域。

FileAlignment 域（双字，DWORD）

PE 文件中，原始数据起始的地址必须是本域取值的整数倍。病毒不会修改此域，但是用这个域的方式与使用 SectionAlignment 域类似。

SizeOfImage 域（双字，DWORD）

当连接程序生成 PE 文件时，它会计算程序需要装入到内存的文件映像各部分

大小的总和。这包括了从映像基地值开始一直到最后一个节末尾的大小。最后一节的末尾地址对齐到最接近的SectionAlignment整数倍的位置。几乎所有的PE感染技术都使用和修改SizeOfImage域的值。

CheckSum域（双字，DWORD）

这是文件校验和。大多数可执行文件的此域值是0，但所有的DLL和驱动程序都必须有一个校验和。Win32的装入程序的装载DLL前，简单地忽略这个域而不作检查，这使得有些Win32病毒能够很容易地感染KERNEL32.DLL。有些病毒用这个域来作为文件是否已被感染的标记，以避免二次感染。还有些病毒为了更好地演示宿主已被感染的事实而重新计算值。

2.节表

节表位于PE头部和各节原始数据之间，包含了关于PE文件映像中各节的信息。

节的作用主要是把不同的功能模块划分开来，如可执行代码、数据、全局数据、调试信息、重定位信息等。病毒要在PE文件中增加新的病毒代码或把病毒代码插入一个现有节时，很重要的一个任务就是修改节表。文件中的每个节都在节表中有一个节头部。这些头部描述了各节的名字、各节装入内存后的虚拟地址，即该节在内存中起始位置的RVA。各节在文件中的有效尺寸、各节在文件中偏移量以及各节的原始数据尺寸（SizeOfRawData）。第一代病毒，如Boza，会在节表中插入一个新的节头部。

有时候，在文件中找不到插入新的节头部的空间，要想修改不是很容易。因此，当今的病毒会攻击线有的最后一个节头部，并修改其中各域的值以便把病毒代码插入该节。这使得病毒的代码节不容易被发现，这种感染方法的风险也减小了。

图3-3列举了Windows计算器的节表为例。

节的名字可以任意取，甚至可以只包括零值，因为装入程序似乎并不关心它。但一般而言，节的名字描述了节的实际功能。这里可能会感到混淆，因为PE文件的实际代码是放在一个.text 节中的。这个名字是传统遗留下来的，与过去COFF格式中的名字一样。连接程序把各种OBJ文件中所有的.text节都集中起来，放入PE文件中的一个大.text节中，并把这个节的头部放在节表的第一个位置。后文将提到，.text节不仅包含代码，而且也包含了一个为DLL库函数调用服务的额为跳转表。Borland公司的连接程序用的是名为“CODE”的.text节，这不是一个传统的名字。

```
01.text VirtSize: 000096B0 VirtAddr: 00001000
raw data offs: 00000400 raw data size: 00009800
relocation offs: 00000000 relocations: 00000000
line # offs: 00000000 line #'s: 00000000
characteristics: 60000020 CODE MEM_EXECUTE MEM_READ

02 .bss VirtSize: 0000094C VirtAddr: 0000B000
raw data offs: 00000000 raw data size: 00000000
relocation offs: 00000000 relocations: 00000000
line # offs: 00000000 line #'s: 00000000
characteristics: C0000080 UNINITIALIZED_DATA MEM_READ MEM_WRIT

03 .data VirtSize: 00001700 VirtAddr: 0000C000
raw data offs: 00009C00 raw data size: 00001800
relocation offs: 00000000 relocations: 00000000
line # offs: 00000000 line #'s: 00000000
characteristics: C0000040 INITIALIZED_DATA MEM_READ MEM_WRITE

04 .idata VirtSize: 00000B64 VirtAddr: 0000E000
raw data offs: 0000B400 raw data size: 00000C00
relocation offs: 00000000 relocations: 00000000
line # offs: 00000000 line #'s: 00000000
characteristics: 40000040 INITIALIZED_DATA MEM_READ

05 .rsrc VirtSize: 000015CC VirtAddr: 0000F000
raw data offs: 0000C000 raw data size: 00001600
relocation offs: 00000000 relocations: 00000000
line # offs: 00000000 line #'s: 00000000
characteristics: 40000040 INITIALIZED_DATA MEM_READ

06 .reloc VirtSize: 00001040 VirtAddr: 00011000
raw data offs: 0000D600 raw data size: 00001200
relocation offs: 00000000 relocations: 00000000
line # offs: 00000000 line #'s: 00000000
characteristics: 42000040 INITIALIZED_DATA MEM_DISCARDABLE MEM_READ
```

图 3-3 用 PEDUMP 查看计算器.exe 的节表

另一个常见的的节名字为.data，该节包含了初始化后的数据。.bss 节包含的是未初始化的静态和全局变量。.rsrc 节存储了应用程序的资源。

.idata 节包含了导入表，这是 PE 格式中对病毒而言非常重要的一部分，但节只是从逻辑上对文件映像进行分隔。由于并未做强制规定，.idata 节的内容可能会被归入到其他任何节中，或者根本就不出现。

.edata 节对病毒也非常重要，因为它列出了当前程序导出给其他可执行文件的所有 API。

.reloc 节存储了基地值重定位表。有些病毒特别关注可执行文件的重定位条目，

但多数来自微软的可执行文件似乎都没有.reloc 节。.reloc 节在早期的 PE 格式设计中存在问题：实际的可执行程序在 DLL 前就被加载了，并在其自己的虚拟地址空间中执行。

最后还有一个名为.debug 的常见节，它包含了可执行文件的调试信息。这对病毒来说并不重要，尽管有些可执行文件缺省包含了各种各样的特殊节名。

对大多数病毒而言，节表头部有三个非常重要的域：VirtualSize、SizeOfRawData 和 Characteristics 域。

Characteristics 域包含了一组标志，用于指示节的属性。代码节有一个“可执行”的标志，但不需要有“可写”标记，因此代码和数据是分开的。对追加型病毒代码，由于它必须把数据区域放置在代码中，所以情况有所不同。因此，病毒必须检查和修改存储其代码的节的 Characteristcs 域。

所有这些都表明，32 位病毒的感染技术相比不同的 DOS EXE 病毒的技术更加复杂。多数的感染实现起来都不容易。

3 PE 文件导入：DLL 与可执行文件如何连接

多数 Windows NT 病毒的开发都严重依赖域作者对导入表的理解。导入表是 PE 文件结构的一个非常重要的部分。Win32 应用程序是通过 PE 结构中的导入表连接到它们调用的那些 DLL 的。导入表包含了被导入的 DLL 名字以及从这些 DLL 导入的函数的名字。看看下面的例子：

ADVAPI32.DLL

Ordn Name

285 RegCreateKeyW

279 RegCloseKey

KERNEL32.DLL

Ordn Name

292 GetProfileStringW

415 LocalSize

254 GetModuleHandleA

52 CreateFileW

278 GetProcAddress

171 GetCommandLineW

659 lstrcatW

126 FindClose

133 FindFirstFileW

470 ReadFile

635 WriteFile

24 CloseHandle

79 DeleteFileW

可执行代码位于 PE 文件的.text 节。当应用程序调用 DLL 中的一个函数时，实际的 CALL 指令不直接调用 DLL，而是首先使用.text 节中的恶意个跳转指令（JMP DWORD PTR [XXXXXXXX]）。

该跳转指令的目标地址存储在.idata 节中，也被称为导入地址表中的恶意个条目。跳转指令把控制权传递到该 IAT 条目所指向的地址，即目标地址。因此.idata 节中的双字包含的函数入口点地址的真实地址，如下面的转储结果所示。图 3-4 中，一个应用程序调用 KERNEL32.DLL 中的 FindfirstFileA()函数。

```
.text (CODE)
0041008E E85A370000 CALL 004137ED ; KERNEL32!FindFirstFileA


004137E7 FF2568004300 JMP [KERNEL32!GetProcAddress] ; 00430068
004137ED FF256C004300 JMP [KERNEL32!FindFirstFileA] ; 0043006C
004137F3 FF2570004300 JMP [KERNEL32!ExitProcess] ; 00430070
004137F9 FF2574004300 JMP [KERNEL32!GetVersion] ; 00430074


.idata (00430000)
.
00430068 1E3CF177 ;-> 77F13C1E Entry of KERNEL32!GetProcAddress
0043006C DBC3F077 ;-> 77F0C3DB Entry of KERNEL32!FindFirstFileA
00430070 6995F177 ;-> 77F19569 Entry of KERNEL32!ExitProcess
00430074 9C3CF177 ;-> 77F13C9C Entry of KERNEL32!GetVersion
```

图 3-4 函数导入表

以这种方式实现函数调用可以简化并加速装入程序。当调用了某个给定 DLL 函数的所有指令都转化为指向同一个地址时，装入程序就不再需要修改每个调用指令了。装入程序需要做的只是在.idata 节中为每个被导入的函数修改其双字的地

址值。

导入表对目前 32 位 Windows 病毒来说非常有用。由于系统装入程序必须修改导入表调用的 Win32 程序使用的所有 API 地址，所以病毒只需要查看宿主程序的导入表，就能很容易获得自己所需要调用的 API 地址。

有些应用程序必须克服这个问题。例如，如果一个 Win32 程序希望能够把 Windows NT 平台下当前运行的所有进程按名字列出，它可以使用 LoadLibrary()函数加载必须的 DLL，使用 GetProcAddres()获取 API 的地址。实际的应用程序可以从其导入表中得到 LoadLibrary()和 GetProcAddres()这两个 API 函数。这就解决了“先有鸡还是先有蛋”的问题：即当需要调用一个 API 时，如果不知道它的地址，该如何调用的问题。

4 PE 文件导出

域导入一个函数对应的是导出一个函数，以便域它可以被 EXE 文件或其他 DLL 调用。PE 文件在.edata 节中保存了它的导出的函数信息。考虑如下对 KERNEL32.DLL 的转储结果，其中列出了该 DLL 导出的几个函数

Entry Pt Ordn Name

000079CA 1 AddAtomA

0000EE2B 38 CopyFileA

0000C3DB 131 FindFirstFileA

00013C1E 279 GetProcAddress

KERNEL32.DLL 的导出表由一个 Image_Export_directory 目录构成，其中包含了指向三个不同列表的指针：函数地址表、函数名表、和函数序数表。现在的 Win32 病毒会在函数名表中查找“GetProcAddres”字符串，以便于获取该 API 函数的入口点值。

当把这个值和 ImageBase 域的值相加时，就得出 DLL 中的该函数的 32 位地址。实际上，这个算法基本上就是真正的 GetProcAddress()函数在 KERNEL32.DLL 内部所遵循的算法。该函数对于那些希望与多种 Win32 系统兼容的 Windows 病毒来说是最重要的函数之一。当有了 GetProcAddress()的地址，病毒就可以获得它希望调用的任何 API 的地址。

3.3.3 PE 文件中的启发性标志

1.代码从最后一节开始执行

PE 格式有一个很重要的优点：不同的功能区域，如代码区，数据区在逻辑上分为不同的节。大多数 Win32 病毒都是应用 2.2 节和 3.1 节介绍的感染技术进行文件感染，并且其中大部分都会将应用程序的入口点修改并指向程序的最后一节，而不是.text 节。缺省情况下，连接程序会把所有的目标代码都放入.text 节，创建多个代码节也有可能，但编译器缺省情况下不这样做，因此大多数 Win32 应用程序永远不会出现这样的结构。如果 PE 映像的入口点不是指向代码节，我们可以选取其指向的内容中的一部分做为一段特征码。

2.节的头部可疑的属性

所有的节都有一个用于描述某些属性的 Characteristics 域，该域包含一组标志用于指示节的属性。代码节要有一个“可执行”的标志，但不需要有“可写”的属性，因为数据跟代码是分开的。很多时候，病毒所在的节没有“可执行标志”但却有“可写”标志，或同时又“可执行”和“可写”标志。这两种情况都必须视为是病毒的一种特殊行为，我们采用定义一个特殊的十六进制字节来代表其这种属性。有些病毒未能成功的设置 Characteristics 域，结果该域为零值，这也需要定义专门的十六进制字节来表示。如图 3-5 为一个 Win32 病毒的最后区段的内容。

```
------------------------------------------------------------------
Section3        Name[.dswlab]
------------------------------------------------------------------
物理地址：(Misc.PhysicalAddress)
00001000  [4096]
真实地址：(Misc.VirtualSize)
00001000  [4096]
节RVA：(VirtualAddress)
00007000  [28672]
节物理长度：(SizeOfRawData)
0000043E  [1086]
节物理偏移：(PointerToRawData)
00005000  [20480]
重定位偏移：(PointerToRelocations)
00000000  [0]
行号表偏移：(PointerToLinenumbers)
00000000  [0]
重定位项目个数：(NumberOfRelocations)
    0000  [0]
行号表行号个数：(NumberOfLinenumbers)
    0000  [0]
节属性：(Characteristics)
C0000040  [3221225536]
        ------------------------------------------------------
        CNT_INITIALIZED_DATA    $00000040(含初始数据)
        MEM_READ                $40000000(可读)
        MEM_WRITE               $80000000(可写)
        ------------------------------------------------------
```

图 3-5 Win32 病毒的最后区段的内容

3.可疑的代码重定向

有些病毒修改的入口点域（指 PE 文件壳选头部的 AderssOfEntryPoint 域）它们的做法是：在入口点放置一个跳转指令，将程序指向其它的节。如果发现代码执行流程在靠近程序入口点的位置从主代码节跳到了别的节，那么可以将此跳转后的一段代码作为特征码进行提取。如图 3-6 为入口点跳转的流程

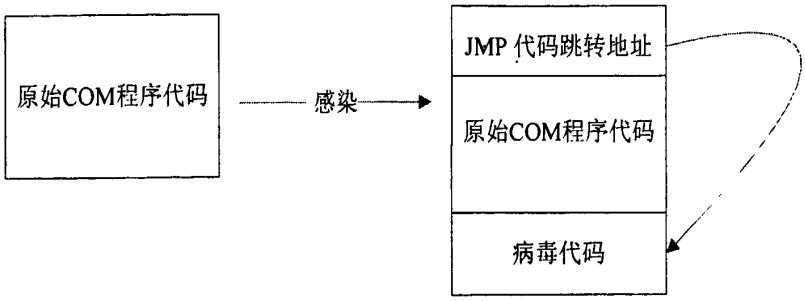

图 3-6 为入口点跳转的流程

4.可疑的代码节名称

如果一个正常情况下不包含代码的节（如.reloc、.debug 等）获取了对程序的控制权，那么就可以根据类似信息定义相应的的特征码与上述行为进行匹配。

5.可能的头部感染

如果 PE 程序的入口点并不指向任何节，而是指向 PE 头部之后和第一节的原始数据之前的区域，那么该 PE 文件可能是感染了一个头部感染型病毒。可以根据以上信息定位到头部感染型病毒的代码，在其中提取一段做为特征码。该启发性标志对检测 CIH 风格的病毒及被病毒破坏的可执行文件极为有用。

6.来自 KERNER32.DLL 的基于序号的可疑导入表项

有些 Win32 病毒会修改所感染程序的导入表，在其中加入基于序号的导入表项。如果有来自 KERNEL32.DLL 的基于序号的导入表项，则可视为是可疑的。一个程序如果从系统 DLL 中按序号导入 API 的话，无法保证它一定能在 Windows 的其他版本中正常运行。所以此类信息也有可能是程序员误操作所致，但不管怎样，如果 GetProcAdress()或 GetModuleHandleA()函数是用序号导入的话，那就应该在该处提取特征序列。

7.导入地址表被修改

如果应用程序的导入表包含 GetProcAdress()和 GetModuleHandleA()两个 API 的导入表项，同时又是用序号导入它们的，则导入表肯定被修改过，此时应该在该处提取对应的特征码序列。

8.多个 PE 头部

当一个 PE 程序有不止一个 PE 头部时，必须视之为病毒特性。因为 PE 头部

包含了很多未被使用的或取值固定的域。当 lfanew 域指向程序的后半部分时，就是这种情况。这时，在文件起点附近可能会找到另一个 PE 头部。如图 3-7 病毒为修改 lfanew 的情况

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	C8	1C	00	00
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68

图 3-7 病毒为修改 lfanew 域

注意该图的 0x3Ch 位置开始的两个字节为 1CC8，为 PE 头的地址，此地址明显已定义到整个程序的后部去了，故此 PE 头是伪造的，还有一个真正的 PE 头被病毒所掩盖了。

9.多个 Windows 程序头部和可疑的 KERNEL32.DLL 导入表

结构分析法通过搜索多个新型可执行文件头部如 32 位的 PE 头部，可以检测出前置病毒。要做到这一点，可以检查实际映像尺寸是否超过文件头部给出的代码尺寸。只要病毒不在程序的尾部加密存储原始的头部信息，就能检测到多个 Windows 程序头部。另外，必须检查导入表中是否有一些特定 API 的导入表项。如果有来自 KERNEL32.DLL 的导入表项：GetMoudleHandle()、Sleep()、FindFirstFile()、FindNextFile()、MoveFile()、GetWindowsDirectory()、WinExec()、DeleteFile()、WriteFile()、GreateFile()、MoveFile()和 CreateProcess()，则可以根据上述相应特征，提取特征向量。

10.可疑的重定位信息

这个标志与代码相关。如果代码中包含能用于确定病毒实际起始地址的指令，就应该对这段代码提取相应的特征向量。例如,调用下一个可能偏移位置的 CALL 指令就是有问题的。很多 Win32 病毒使用 E80000（即下一个 CALL 地址）形式的 32 为等效形式 E800000000。

11.内核查询

如果代码中包含有硬编码的指针指向了某些系统区域，如 KERNEL32.DLL 或 VMM 的内存区域，则该特征可作为特征序列进行提取。这种病毒常常同时在其代码中寻找 PE\0\0 标记，这个行为也应该被提取为特征向量。

当一个应用程序在仿真执行时，如果它访问了一系列内存区域，则应该对其提取特征序列。例如，在计算机病毒和漏洞利用代码中都常常有直接实现

GetProcAderss()功能的代码序列。在计算机病毒启动代码中，直接访问属于KERNEL32.DLL头部区的一些内存区域是很常见的但在正常程序中却是反常的行为。

12.内核的完整性

KERNEL32.DLL文件的完整性可以通过IMAGEHLP.DLL中的一个API，如CheckSumMappedFile()或MapFileAndCheckSum()来检查。

这样，感染KERNEL32.DLL而不重新计算其Checksum域的病毒就很容易露出尾巴，如果发现上述行为可以将其作为特征序列进行提取。

13.把节装入到VMM的地址空间

不幸的是，在Windows的一些系统中可以把一个节装入到ring 0内存区域。虚拟机管理器的内存区域从地址0xC0001000处开始。在0Xc0000000地址有一个未用页面，有几种病毒占据了这个未用页面。病毒的宿主程序的节表中增加了一个新的节头部。这个新的节头部中的虚拟机地址域指向内容地址0Xc0000000.系统装入程序在染毒程序执行时自动地分配该页面。然后，病毒代码就在内核模式下运行。系统装入程序本来很容易就可以做到拒绝分配该页面。但有些Win32系统未实现这个功能。因此，当任何节头部的虚拟地址域指向VMM区域时，应提取对应的特征序列。

14.可选头部的SizeofCode域取值不正确

多数病毒在向PE程序中添加新的代码节时，都不会去碰可选头部的SizeOfCode域。如果新计算所有代码节的尺寸，得到的结果与SizeOfCode域的值不同的话，我们认为该PE文件是被病毒添加的新节。再次基础上提取相应的特征序列。

15.含有多个可以可疑

具有上述启发式标志的2个，或2个以上的启发信息，那么可以依次提取对应启发式标志的特征序列。这样能有效的降低算法的误报率和漏报率。

3.3.4 加壳病毒的特征码提取

病毒加壳一直是病毒躲过杀毒软件，和病毒分析的一条主要途径。对于加壳后的病毒样本，其本身的PE文件结构已被打乱，对于加壳文件的特征码提取不能采用启发式分析法。虽然加壳病毒的PE结构被打乱但加壳程序每次对病毒的加壳方式都是一致的，加壳后的病毒结构也是稳定的，所以我们通过提取加壳病毒入

口点处的一段代码来作为加壳病毒的特征码，其本质是对壳进行特征码提取。加壳病毒特征码提取方法的对象为通过第三方软件不能脱壳的病毒。其具体提取方法如下：

1.分析加壳文件总体结构。

2.定位到程序入口点处。

3.在程序入口点位置附近提取特征码

4.结束。

3.3.5 未加壳特征码的提取与构成

3.3.5.1 特征码的提取

特征码的提取分为两个部分，第一部分为依次查找 PE 文件中上节中提到的启发式标志，如果找到对应的启发性标志。则根据对应启发性标志提取特征码。现列举前 3 个启发性标志进行特征码提取说明

如果检测到启发性标志 1 代码从最后一节开始，则首先定位到病毒代码开始的最后一节，然后在该节中选取一部分适合的代码做特征向量。

如果检测到启发性标志 2 节的头部可疑的属性，由于节名称和节属性在大多数 PE 文件中都具有共性所以不能直接提取节名称和属性作为特征向量，应该将其节名称和节属性加上部分节内容做 Hash 转换，然后将转换后的 16 进制数据选取一部分作为特征向量

如果检测到启发性标志 3 可疑的代码重定向，则类似于启发性标志 1，先跳转到病毒代码入口处，然后再在病毒代码部分选取一部分适合的代码作为特征向量。

所有检测到的启发性标志都按上面三例进行特征码的提取。

3.3.5.2 特征码的构成

由于特征码是根据多个启发性标志解析出的特征序列构成的，所以在构成上应该是分段组合而成的。每个启发性标志的特征向量是固定的。特征码的总长度为 32 个十六进制字符串组成。如果同时检测到多个特征向量，并且特征向量的组合超过 32 个字符串，则选取最具有代表性的特征向量作为特征码。一般以病毒代码处提取的特征向量为主，其他启发性标志提取的特征向量为辅的原则。如果该病毒很不幸没有检测到一个启发性标志，则在病毒程序代码入口点后的代码部分提取 32 位字符串作为该病毒的特征码。

经过测试有 PE 文件启发式方法提取出的特征码具有较好的误报率和漏报率，

可以作为专业病毒分析师提取病毒特征码的重要参考，也可以直接运用提取出的特征码对可疑文件做特征码比对。

3.4 小节

本章首先对特征码提取技术问题进行分析。其中包括问题是什么，难点是什么，以及解决问题的思路。接着从 Win32 病毒感染策略的角度探讨启发式分析，以及如何根据感染策略寻找启发性标志。然后提出基于 PE 文件启发性特征码提取技术以及其关键技术点。最后探讨如何将启发性标志中提取的特征序列组合成特征码。

第四章 基于 PE 文件启发式特征码自动提取系统的设计与实现

本章根据第三章中所提出的 PE 文件启发式特征码提取法，设计并实现一个特征码自动提取的系统。该系统的主要功能为首先对病毒样本进行筛选，将病毒样本分为加壳病毒样本和无壳病毒样本，对加壳病毒样本采用文件入口点提取特征码。而对于无壳病毒样本才用基于 PE 文件的启发式特征码提取法。

在这一章中，首先进行系统总体设计。然后对该系统中的核心模块进行分类介绍。

4.1 总体设计

4.1.1 总体目标

该平台运行在 Windows 平台下，根据第三章所提出的病毒特征码提取算法。并假设需要提取特征码的 PE 文件全文病毒样本。该特征码自动提取平台的总体目标为：

(1)根据目标路径自动找到所有待提取特征码文件。

所有病毒样本均放入一个文件夹中，作为特征码自动提取系统的输入。特征码自动提取平台能自动提取该文件夹中的所有下所有恶意样本作为分析对象。

(2)能自动分析所有该路径下所有文件是否可以提取特征码。

对每个病毒样本分析其结构，判断是否是 Win32 下的 PE 文件，如果不是则放弃提取，如果为 PE 文件则进行下一步的分析。

(3)分析 PE 文件找到代码入口点地址。

对每个病毒样本分析它 PE 文件中的可选 PE 头部，接着找到可选 PE 头部中的 AdressOfEntryPoint 域。AdressOfEntryPoint 域是其入口点地址，接着判断入口点地址是否为一个有效地址，有些病毒或加壳文件会将入口点地址修改为一个无效值。

(4)分析 PE 文件寻找启发性标志。

根据 3.3 节中提到的启发性标志，按 PE 格式寻找待分析的恶意代码是否具有启发性标志。

(5)分段提取PE文件的特征码向量。

在寻找病毒样本启发性标志时，如果找到相应的启发性标志，则在该标志处提取特征向量。如果遍历完整个PE文件都没有发现启发性标志，就在代码入口点后面提取一段作为特征码。

(6)对特征向量进行组合形成特征码。

对在各启发性标志处提取的特征码向量进行组合，选取最能代表病毒样本特征的32位特征向量形成特征码。

(7)将提取的病毒特征码入库。

将提取的特征码与病毒特征码库中的特征码比较，如果为新的特征码，则将该特征码入库。

4.1.2 总体结构

该系统包括10个模块分别为：

(1)映射文件模块

将病毒样本映射入内存。

(2)检查文件是否为PE模块

通过PE文件的DOS头和PE头特征标志判断文件是否为PE文件。

(3)检查文件是否加壳模块

PE文件壳的目的不只是为了迷惑反病毒软件，而且还在于隐藏程序设计的思路，这也是现在很多商业软件都普遍加壳的原因。所以，在PE文件加壳过程中要隐藏导入表的信息，使得在最终加过壳的PE文件的导入表中包含尽可能少的信息。因而，通过判断PE文件从Kernel32.dll中导入的函数的个数比正常文件少很多就可以判断该文件加过壳。

(4)对加壳文件提取特征码模块

对于加壳病毒样本提取其程序入口处的一段 32 位十六进制的内容作为特征码。

(5)寻找病毒入口点地址模块

分析病毒样本的结构，定位到其PE文件中的可选头部，接着找到可选头部中的AdressOfEntryPoint域。AdressOfEntryPoint域的内容为其入口点地址。

(6)寻找启发式标志模块

根据3.3节中提到的启发性标志，按PE格式寻找待分析的恶意代码是否具有

启发性标志。具体在 4.2.2 中详细介绍。

(7)特征向量提取模块

在寻找病毒样本启发性标志时，如果找到相应的启发性标志，则在该标志处提取特征向量。

(8)特征向量组合模块

对在各启发性标志处提取的特征码向量进行组合，选取最能代表病毒样本特征的 32 位特征向量形成特征码。如果为加壳病毒样本，则将对加壳文件提取特征码模块中提取的特征向量直接作为该病毒样本的特征码

(9)特征码入库模块

将提取的特征码与病毒特征码库中的特征码比较，如果为新的特征码，则将该特征码入库。

(10)获得其他信息模块

提取该病毒样本的其它信息以便为病毒分析人员分析病毒时提供辅助信息。该模块主要提取病毒样本的编译器信息、链接器信息等。

其中映射文件模块，对加壳文件提取特征码模块，寻找病毒入口点地址模块，寻找启发式标志模块，特征向量提取模块，特征向量组合模块为本系统核心模块下面章节将详细阐述各核心模块的设计思路与设计流程。图 4-1 显示了基于 PE 文件启发式特征码提取系统结构图。

系统将病毒样本分为加壳和未加壳两种类型，对于加壳病毒样本，首先用第三方软件进行脱壳，如果脱壳未成功则采用壳的特征码提取法进行特征码的提取，如果脱壳成功，则采用基于 PE 文件的启发式特征码提取法，对病毒样本的 PE 结构进行分析，寻找 PE 结构中我们设定的启发性标志，如果找到启发性标志，则通过启发性标志规定的相应动作进行特征序列的提取。通过对各特征序列进行选取和组合形成最终的特征码，上述过程通过前面的 10 个模块的组合运行得以实现，其中通过启发性标志提取病毒样本特征码的对象是未加壳的病毒样本，或加壳后通过第三方软件脱壳后的病毒样本。.

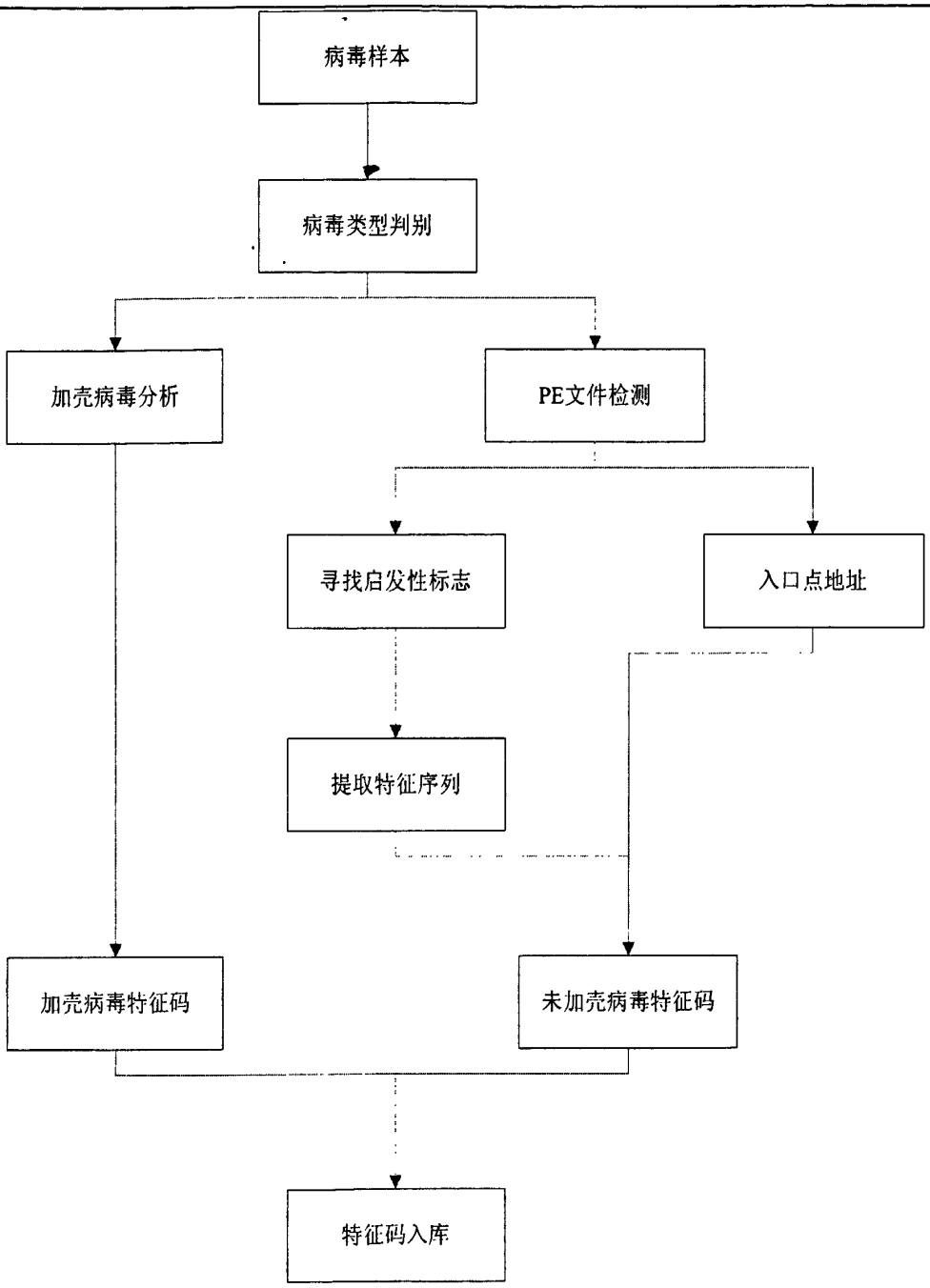

图 4-1 基于 PE 文件启发式特征码提取系统结构图

如图 4-1 所示系统的输入是待提取的病毒文件，经过分析和特征码的提取，系统的输出为病毒的特征码。

系统内部各模块的联系与合作运行请参看图 4-2 PE 文件启发式特征码提取系统的总体流程图。图 4-2 从系统流程的角度展现了模块之间的关联和运作方式。下面的部分将通过该流程图详细的阐述基于PE文件启发式特征码自动提取平台的

设计思路。

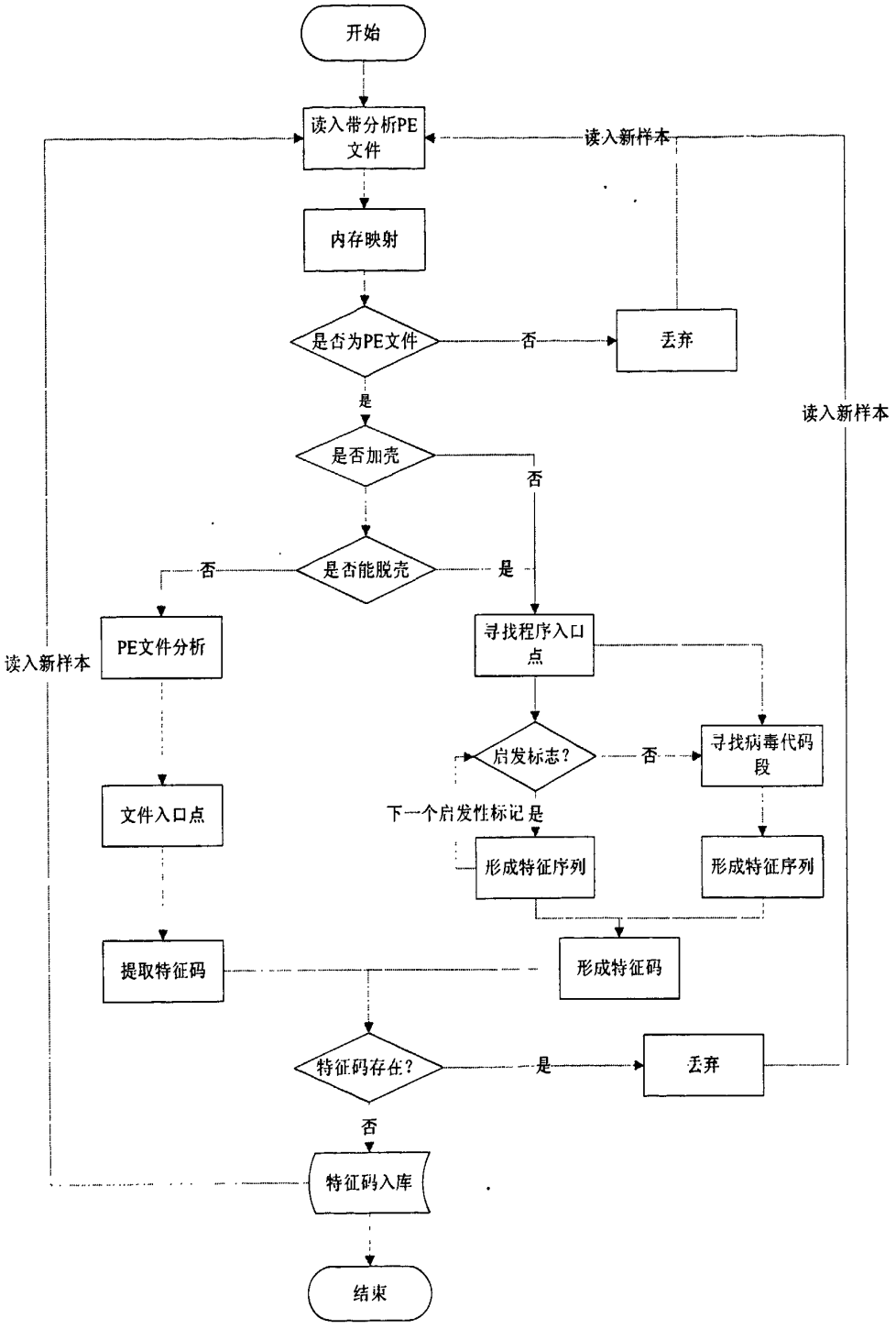

图 4-2 PE 文件启发式特征码提取系统的总体流程图

如图 4-2 所示基于 PE 文件启发式特征码提取系统首先从待分析病毒中提取一个病毒样本，接着将该病毒样被加载到内存中，接着判断该病毒样本是否是 Win32

平台下的 PE 文件。如果不是一个正常的 PE 文件，那么该病毒样本的破坏行为不是针对 Windows 平台的，此时丢弃该病毒样本继续从病毒库中提取下一个病毒样本进行分析。

接着判断文件是否为一个加壳文件。如果文件加壳那么采用第三方通用脱壳程序对其进行脱壳，如果第三方通过脱壳程序不能对该病毒样本脱壳，则采用 3.4 节中提到的加壳病毒特征码提取方法对该带壳病毒直接提取特征码。

如果第三方脱壳工具能成功脱去该病毒样本的壳，或该病毒样本没有加壳，则通过 PE 文件分析首先找到该病毒样本的 AddressOfEntryPoint。因为如果接下来的启发式标志没有找到或者根据启发式标志提取的特征序列不能够成完整的特征码，需要从该病毒样本 AddressOfEntryPoint 后的病毒代码中提取一部分 16 进制字节作为特征序列。接着通过 PE 文件分析依次寻找 3.7.2 中提到的启发式标志，如果找到对应的启发式标志则通过 3.7.3.1 节提到的特征序列提取方式提取该病毒样本的特征序列。直到寻找完所有启发式标志，然后判断已经提取的特征向量是否能构成一个完整的特征码，如果不能构成一个完整的特征码，则通过 3.7.3.2 小节提到的方法进行特征序列的组合

最后形成完整的特征码，然后判断该病毒样本的特征码是否已经存在于病毒特征码库当中，如果存在则丢弃该特征码。继续提取新的病毒样本进行分析。如果该特征码不在当前特征库中则将该特征码入库。然后继续提取新的病毒样本进行分析。

4.2 特征码自动提取系统的设计与实现

下面的章节将对各模块的具体实现和设计思路进行分析。其中包括映射文件模块的实现，寻找启发性标志模块，按格式建立特征码模块。

4.2.1 映射文件模块的实现

映射文件模块负责将带提取特征码的文件打开并将其映射到内存中。

4.2.1.1 设计流程

设计流程如图 4-3 映射文件模块设计流程图

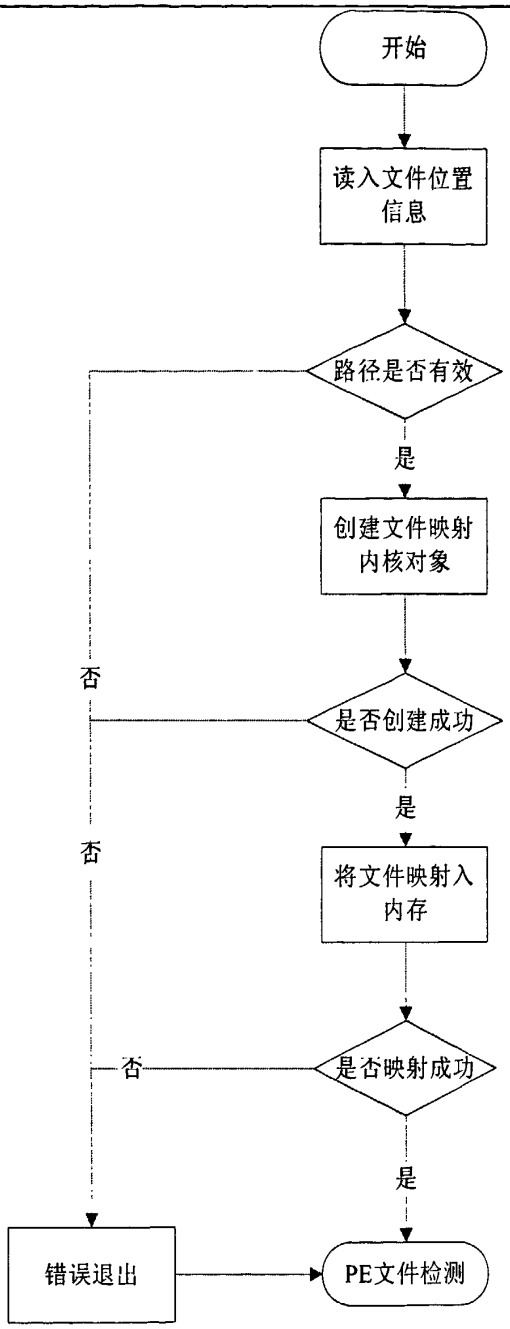

图 4-3 映射文件模块设计流程图

内存映射文件与虚拟内存有些类似，通过内存映射文件可以保留一个地址空间的区域，同时将物理存储器提交给此区域，只是内存文件映射的物理存储器来自一个已经存在于磁盘上的文件，而非系统的页文件，而且在对该文件进行操作之前必须首先对文件进行映射，就如同将整个文件从磁盘加载到内存。由此可以看出，使用内存映射文件处理存储于磁盘上的文件时，将不必再对文件执行 I/O 操

作，这意味着在对文件进行处理时将不必再为文件申请并分配缓存，所有的文件缓存操作均由系统直接管理，由于取消了将文件数据加载到内存、数据从内存到文件的回写以及释放内存块等步骤，使得内存映射文件在处理大数据量的文件时能起到相当重要的作用。另外，实际工程中的系统往往需要在多个进程之间共享数据，如果数据量小，处理方法是灵活多变的，如果共享数据容量巨大，那么就需要借助于内存映射文件来进行。实际上，内存映射文件正是解决本地多个进程间数据共享的最有效方法。

内存映射文件并不是简单的文件 I/O 操作，实际用到了 Windows 的核心编程技术--内存管理。所以，如果想对内存映射文件有更深刻的认识，必须对 Windows 操作系统的内存管理机制有清楚的认识，内存管理的相关知识非常复杂，超出了本文的讨论范畴，在此就不再赘述，感兴趣的读者可以参阅其他相关书籍。下面给出使用内存映射文件的一般方法：

首先要通过 CreateFile()函数来创建或打开一个文件内核对象，这个对象标识了磁盘上将要用作内存映射文件的文件。在用 CreateFile()将文件映像在物理存储器的位置通告给操作系统后，只指定了映像文件的路径，映像的长度还没有指定。为了指定文件映射对象需要多大的物理存储空间还需要通过 CreateFileMapping()函数来创建一个文件映射内核对象以告诉系统文件的尺寸以及访问文件的方式。在创建了文件映射对象后，还必须为文件数据保留一个地址空间区域，并把文件数据作为映射到该区域的物理存储器进行提交。由 MapViewOfFile()函数负责通过系统的管理而将文件映射对象的全部或部分映射到进程地址空间。此时，对内存映射文件的使用和处理同通常加载到内存中的文件数据的处理方式基本一样，在完成了对内存映射文件的使用时，还要通过一系列的操作完成对其的清除和使用过资源的释放。这部分相对比较简单，可以通过 UnmapViewOfFile()完成从进程的地址空间撤消文件数据的映像、通过 CloseHandle()关闭前面创建的文件映射对象和文件对象。

4.2.1.2 函数声明

在文件映射中主要用到几个数据结构，其声明如下：

```
HANDLE CreateFile(LPCTSTR lpFileName,DWORD dwDesiredAccess,DWORD dwShareMode,LPSECURITY_ATTRIBUTES pSecurityAttributes,DWORD dwCreationDisposition,DWORD dwFlagsAndAttributes, HANDLE hTemplateFile);
```

HANDLE CreateFileMapping(HANDLE hFile,LPSECURITY_ATTRIBUTES lpFileMappingAttributes,DWORD flProtect,DWORD dwMaximumSizeHigh,DWORD dwMaximumSizeLow,LPCTSTR lpName);

LPVOID MapViewOfFile(HANDLE hFileMappingObject,DWORD dwDesiredAccess,DWORD dwFileOffsetHigh,DWORD dwFileOffsetLow,DWORD dwNumberOfBytesToMap);

4.2.2 寻找启发性标志模块

寻找启发性标志模块为特征码自动提取平台的一个核心模块，该模块的主要功能是通过 PE 文件分析寻找病毒样本中的启发性标志，根据不同的启发性标志提取相应的特征码序列。

4.2.2.1 设计流程

寻找启发性标志模块设计流程请参看图 4-4 寻找启发性标志流程图

如图 4-4，该模块首先对该病毒样本进行 PE 文件分析，由 3.3.2 节的介绍的 PE 文件结构可知，在对整个 PE 文件进行扫描时，可以根据 PE 文件结构逐个寻找启发性标志。

首先找到 DOS HEADER 头并提取其 0x3Ch 处的内容判断，是否该病毒样本的 PE 头部偏移量是否在整个 PE 文件样本的中间，如果 PE 头部的偏移量不在正常位置，可以往回寻找是否还有其他 PE 头部。如果找到了一个启发性标志，因为 PE 头部包含了很多未被使用的或取值固定的域。当 lfanew 域指向程序的后半部分时，就是这种情况。这时，在文件起点附近可能会找到另一个 PE 头部。该头部才是真正正常文件的 PE 头部，病毒通过加载一个自己的 PE 头部以实现得到程序控制权的目的。所以当找到此启发性标志时可以提取相应内容作为特征码。

通过 DOS HEADER 和 PE HEADER 的大小定位到 PE 可选头部，在 PE 可选头部我们首先提取 SizeOfCode 域的大小，以比对是否符合实际代码节的大小，因为多数病毒在向 PE 程序中添加新的代码节时，都不会去碰可选头部的 SizeOfCode 域。如果新计算所有代码节的尺寸，得到的结果与 SizeOfCode 域的值不同的话，我们认为该 PE 文件是被病毒添加的新节。因此可以根据此启发式标志提取相应的特征码序列。

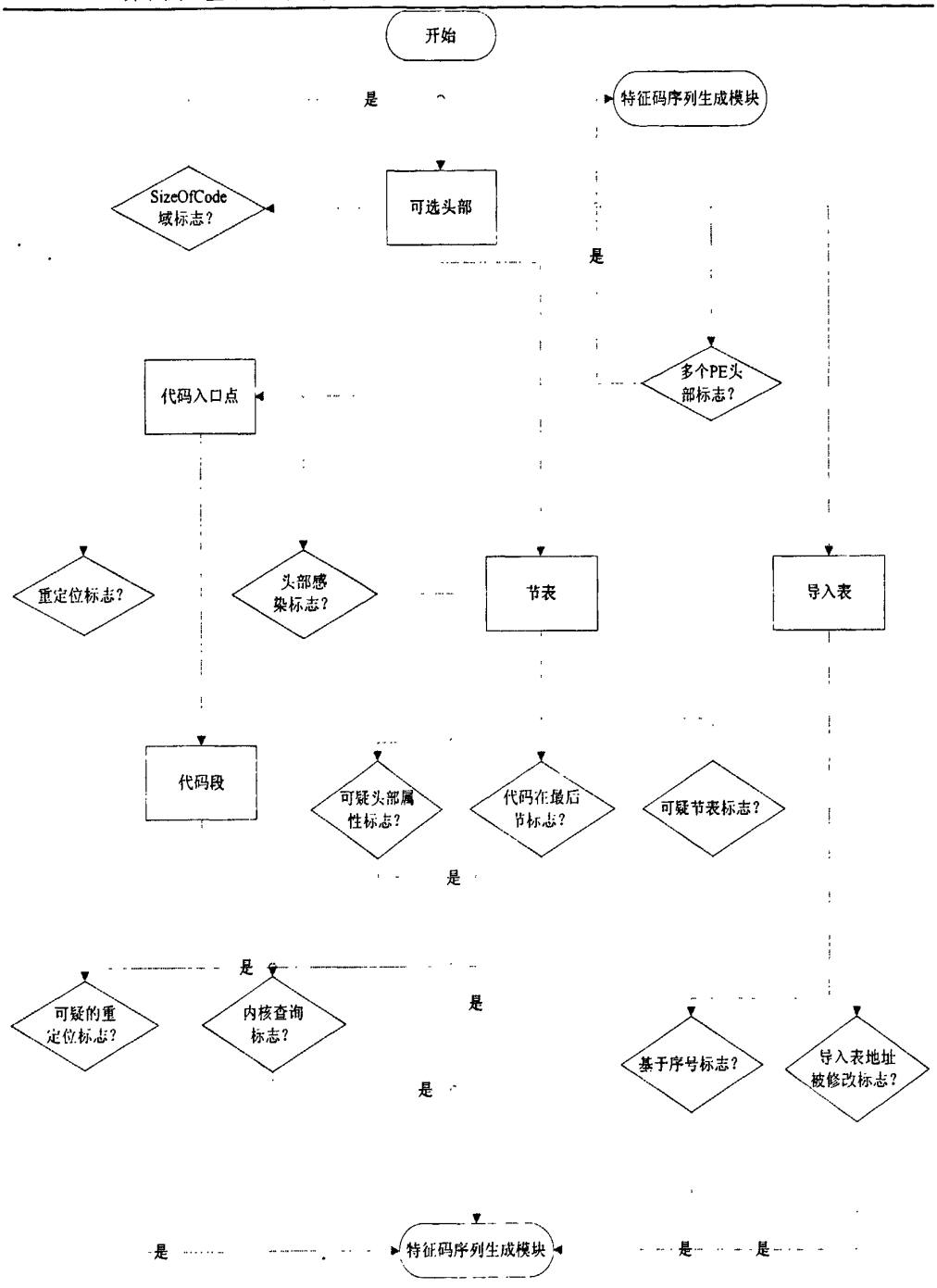

图 4-4 寻找启发性标志流程图

接着继续提取可选头入口点地址域的值，判断该值是否执行紧接着的代码段，如果该值指向了其它节，那么可以认定在这个域处找到了启发性标志。因为大部分病毒都会将应用程序的入口点修改并指向程序的最后一节，而不是.text 节。缺省情况下，连接程序会把所有的目标代码都放入.text 节。可以根据此启发性标志

提取相应的特征码序列。

根据可选头部的代码入口点域定位到程序代码部分，继续寻找其它启发性标志。首先检查代码段的起始位置附近是否有跳转指令，接着判断跳转指令是否将跳转到病毒样本的其他节。如果跳转到其它节可以判定代码重定向启发性标志找到。我们根据跳转指令进入跳转到相应的地址空间。在此空间内提取一段代码作为特征码向量。采用这种提取的原因是有些病毒修改的入口点域在入口点放置一个跳转指令，将程序指向其它的节。然后再开始执行自己的感染代码。

接着继续查找代码段如果代码中包含能用于确定病毒实际起始地址的指令，例如,调用下一个可能偏移位置的 CALL 指令就是有问题的。很多 Win32 病毒使用 E80000（即下一个 CALL 地址）形式的 32 为等效形式 E800000000。如果找到如上述例子的指令就可以判定找到可疑重定位信息标志。根据指令的跳转位置提取相应代码作为特征码向量。

如果入口点地址不指向任何节，而在 PE 头或在 DOS 头和节表中，那么可以判定找到了头部感染启发性标志。因为正常文件不在代码入口点一般都会指向代码节，并且绝不可能出现在 DOS 头部，PE 头部等位置。根据代码入口点所指向的位置定位病毒样本代码，然后提取相应的特征向量。

到此基于代码入口点域以及代码段的启发性标志特征以寻找完毕，通过 PE 可选头部信息定位到该病毒文件的节表。

分析每个节表的 Characteristics 域，判断其属性值，Characteristics 的属性值定义如下：

0x00000020 这个节包含代码。通常与可执行标志（0x80000000）一起设定。

0x00000040 这个节包含已初始化的数据。除了可执行的节和.bss 节之外，几乎所有的节都设定了这个标志。

0x00000080 这个节包含未初始化的数据（例如.bss 节）。

0x00000200 这个节包含备注或其它类型的信息。典型的使用这个标志的节是由编译器生成的.drectv 节，它包含编译器传递给链接器的命令。

0x00000800 这个节的内容并不放入最后的 EXE 文件中。这些节是编译器或汇编程序用来给链接器传递信息的。

0x02000000 这个节可以被丢弃，因为一旦它被加载之后，进程就不再需要它了。最常见的可以被丢弃的节是基址重定位节（.reloc）。

0x10000000 这个节是共享的。当用于 DLL 时，这个节中的数据在所有使用这个 DLL 的进程中是共享的。数据节默认是不共享的，这意味着使用某个 DLL

的所有进程都有这个节中的数据的私有副本。说得更专业一点就是，共享节告诉内存管理器对这个节的页面映射进行一些额外设置以便使用这个 DLL 的所有进程都使用同一块物理内存。要使一个节变成共享的，可以在链接时使用 SHARED 属性。例如“LINK /SECTION:MYDATA,RWS ...”告诉链接器这个叫做 MYDATA 的节应该被设置成可读、可写和共享的。

0x20000000 这个节是可执行的。只要设置了“包含代码”标志（0x00000020），通常也会设置这个标志。

0x40000000 这个节是可读的。EXE 文件中的节总是设置这个标志。

0x80000000 这个节是可写的。如果 EXE 文件的节没有设置这个标志，加载器会把映射的页面都标记成只读和只执行的。通常.data 和.bss 节被设置这个属性。有趣的是.idata 节也设置了这个标志。

如果发现该节不是.text 或.data 节但又具有 0x20000000、0x80000000、A0000000 属性则判定找到了节的头部可疑的属性启发性标志，因为代码节要有一个“可执行”的标志，但不需要有“可写”的属性，因为数据跟代码是分开的。很多时候，病毒所在的节没有“可执行标志”但却有“可写”标志，或同时又“可执行”和“可写”标志。这两种情况都必须视为是病毒的一种特殊行为，我们采用定义一个特殊的十六进制字节来代表其这种属性。有些病毒未能成功的设置 Characteristics 域，结果该域为零值，这也需要定义专门的十六进制字节来表示。

然后跳转到导入表，判断导入表的 KERNER32.DLL 是否基于序号形式标识，如果找到则可以认定找到来自 KERNER32.DLL 的基于序号的可疑导入表项启发性标志。一些 Win32 病毒会修改所感染程序的导入表，在其中加入基于序号的导入表项。如果有来自 KERNEL32.DLL 的基于序号的导入表项，则可视为是可疑的。

接着判断应用程序的导入表包含 GetProcAdress()和 GetModuleHandleA()连个 API 的导入表项，同时又是用序号导入它们的，则导入表肯定被修改过，可以认定找到导入地址表被修改启发性标志，此时应该在该处提取对应的特征码序列。

4.2.2.2 关键代码

ObtainCharacteristicsSignature（）函数为判断病毒样本是否具有节表属性可疑启发性标志，输入为内存映像文件，经过 PE 分析把 DOS 头赋值给变量 pdh，通过 pdh 定位 PE 头，再定位可选头，最后定位到节表，依次查找每个节表的 Characteristics 域是否具有可疑属性，如果找到启发性标志则返回可疑节名称，反之则返回 NULL

```
int* __fastcall TForm1:: ObtainCharacteristicsSignature (PVOID buff,int type)
{
    PIMAGE_DOS_HEADER pdh=(PIMAGE_DOS_HEADER)buff;
    PIMAGE_NT_HEADERS
pnh=(PIMAGE_NT_HEADERS)((BYTE*)pdh+pdh->e_lfanew);  //得到PE头部地址
    IMAGE_OPTIONAL_HEADER32  ioh=(IMAGE_OPTIONAL_HEADER32)
(pnh->OptionalHeader);     //得到可选头部地址
    DWORD aoep=ioh.AddressOfEntryPoint;
    WORD numberOfSections=pnh->FileHeader.NumberOfSections;
    PIMAGE_SECTION_HEADER
psh=(PIMAGE_SECTION_HEADER)IMAGE_FIRST_SECTION(pnh);
    int i=0;
    while(i<numberOfSections)      //遍历节表
    {
  if(aoep>=psh->VirtualAddress&&aoep<psh->VirtualAddress+(psh->Misc).Virtual
Size)
          {
  int offset_end=psh->Misc.VirtualSize-(aoep-psh->VirtualAddress)-1;
           int offset_end=psh->Misc.VirtualSize-(aoep-psh->VirtualAddress);
           if(type==INFECTTYPE)
           {
              maxoffset[0]=0;
  maxoffset[1]=offset_end;
  entrypoint=psh->PointerToRawData+aoep-psh->VirtualAddress;
          }
          else if(type==OTHERTYPE)
          {
              maxoffset[0]=aoep-psh->VirtualAddress;     //delta
              maxoffset[1]=offset_end;
  entrypoint=psh->PointerToRawData+aoep-psh->VirtualAddress;
           }
           return psh->name;
```

```
            }
    psh++;
    i++;

    return NULL;
    }
```

4.2.3 按格式建立特征码模块

该模块的功能是收集各启发性标志产生的特征序列。组合为 32 位十六进制的特征码，提取的特征码形式为：

558bec83c4f0b89c233100e8901effffa18033310033d2e8349affffa1803331

4.2.3.1 设计流程

按格式建立特征码模块设计流程请参看图 4-5 按格式建立特征码模块设计流程图

如图 4-5 所示，该模块分别收集寻找启发性标志模块和加壳病毒样本特征码提取模块的特征码序列。然后根据特征序列组合成特征码。特征序列的具体组合方式如下：

1.首先判断特征序列来自哪个模块，如果是加壳病毒样本特征码提取模块的特征序列则特征序列已为一个 32 位十六进制的特征码，不需要再进行组合，直接作为生成的特征码转入 4。如果特征序列是来自于寻找启发性标志模块则转入 2。

2.不同的启发性标志会有不同长度的特征序列，将这些特征序列分别装入一个数组（数组最大长度可容纳所有特征序列），对数组的长度进行分析如果超过 16 位十六进制数。则只取前 16 位特征向量，剩下 16 位十六进制数从恶意样本的代码段提取，将两部分特征序列组合成完整的特征码转入 4。如果启发性标志模块收集的特征序列小于 16 位则转入 3。

3.计算数组的长度，分析还需要补充多少位特征序列，然后从恶意样本的代码部分提的特征序列中提取相应长度的特征序列与启发性标志的特征序列组合形成完整特征码。

4.结束。

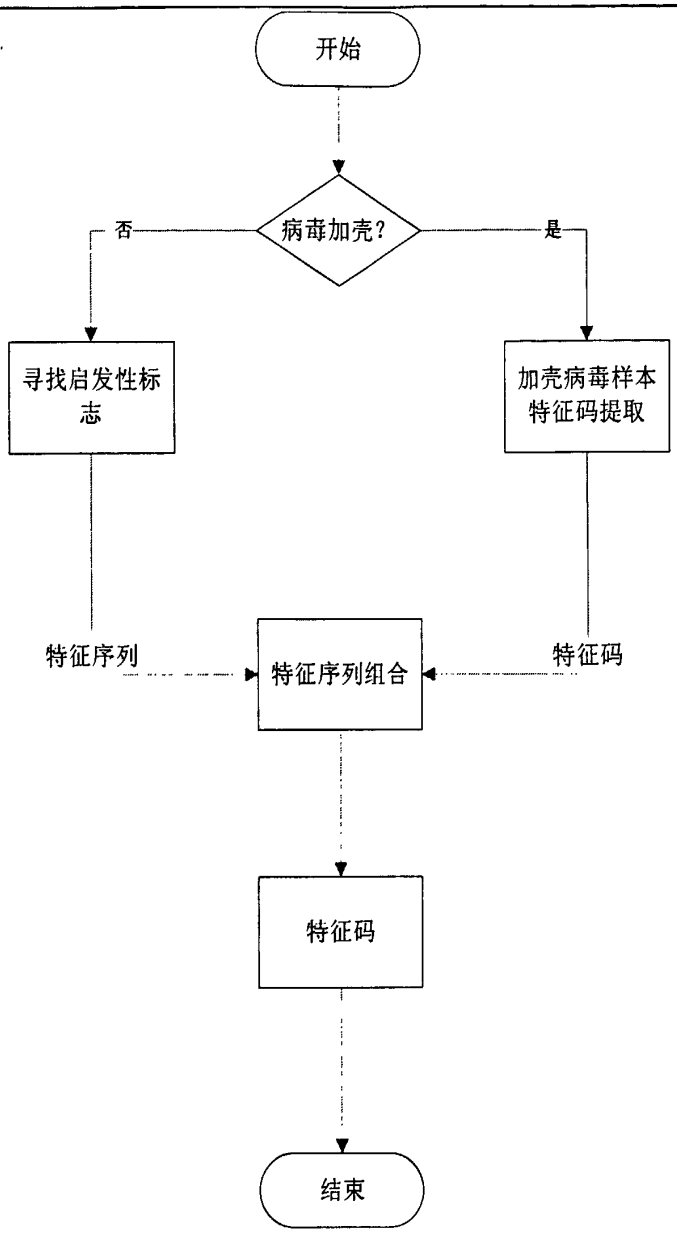

图 4-5 按格式建立特征码模块设计流程图

4.2.3.2 关键代码

```
int __fastcall TForm1::IsentrypointModified(int signature1,int signature2,int signature3, int type)    //type 为特征码组合类型
{
    int array[200];
    int signature[32];
    int length;
```

```
if(type == 3)
{
    for(int i = 0; i < 32; i++)
    {
        signature[i] = signature[3];
        return signature;
    }

}
else
{
    if((length = sizeof(signature1)/4) > 15)    //计算特征码长度
    {
        for(int j = 0; j < 16; j++)          //提取特征向量 1 的部分内容
        {
            signature[j] = signature1[j];
        }
        for(int j = 16; j < 32; j++)         //提取特征向量 2 的部分内容
        {
            signature[j] = signature1[j];
        }
        return(signature);
    }
    else
    {
        for(int k = 0; k < length; k++)
        {
            signature[k] = signature1[k];
        }
        for(int k = (32 -length); k < 32; K++)
        {
```

```
                    signature[k] = signature1[k];
                }
                reutrn(signature);
            }
        }
    }
```

4.4 小结

本章给出了基于PE文件启发式特征码自动提取平台的设计思路和具体实现首先从总体设计入手，分析了系统的总体流程和框架，接着对系统中的重要模块进行详细的阐述和分析。对核心模块给出了算法论证和关键代码。

第五章 系统测试

本章对第三章提出的算法和在第四章中设计并实现的特征码自动提取系统进行测试，首先对根据算法提取出的特征码进行误报率，漏报率，以及算法检查病毒变种的能力进行测试，然后对算法的性能进行测试。最后给出测试结果并提出算法中还存在的一些问题。

本章安排如下：第一节，测试环境的搭建以及介绍测试所用到的病毒样本。第二节，设计测试内容。第三节，给出测试结果。第四节，对测试结果进行总结并提出存在的问题。第五节，本章总结。

5.1 测试环境

5.1.1 PE 文件启发式特征码自动提取平台的硬件环境

一台普通台式机。120G 硬盘，2G 内存，intel core2 Due 处理器。

5.1.2 PE 文件启发式特征码自动提取平台的软件环境

操作系统为 WindowsXP，所需软件为 vc6.0。所有病毒均由卡巴斯基检测分类后的结果，每一类病毒和其变体被分在同一个文件夹下。（不同杀毒软件对病毒和其变体的分类有所不同。）

5.1.3 测试对象

待分析恶意样本为卡巴斯基检测为恶意病毒的 3797 个病毒文件，所有病毒样本均放在 virus_and_classfied_by_kap/目录下

比对的正常文件为 Windows XP 操作系统目录下 C:\WINDOWS 下所有 exe 和 dll 文件，共 1685 个文件。

在一个目录下所有的文件均为由卡巴斯基检测出的一种类型病毒和其病毒变体如：

在 virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\ 目录下有 3 个病毒样本，它们均为同一病毒的 3 个变种。

5.2 测试内容

5.2.1 特征码分析比对测试

1.已知恶意病毒文件夹 virus_and_classfied_by_kap\目录下的所有病毒样本进行分析，提取所有病毒的特征码。

2.提取 C:\WINDOWS 下所有 exe 和 dll 文件的特征码。

3.测试所有 1 中特征码在与 2 中的特征码是否有重合的现象。

（测试结果见 5.3 节，表 1 到表 6）

5.2.2 漏报率比对测试

对已知恶意病毒文件夹 virus_and_classfied_by_kap\目录下的所有病毒样本进行分析，提取所有病毒的特征码，测试一个病毒和其病毒变体是否能具有相同的特征码。（测试结果见 5.3 节，表 6）

漏报率计算方式为：漏报率 = 未能通过病毒特征码检测到的病毒变种数/参与检测的病毒总数。

5.2.3 误报率比对测试

提取 C:\WINDOWS 下的所有 exe 和 dll 的特征码，与 virus_and_classfied_by_kap/目录下的 3979 个病毒样本特征码重合的误报率。（测试结果见 5.3 节，表 7）

误报率计算方式为：误报率 = 正常文件被检测为病毒的个数/正常文件总数。

5.2.4 特征码提取性能测试

测试提取 3979 个病毒样本的所需要的时间（测试结果见 5.3 节，表 8）

5.3 测试结果

表 5-1 特征码分析比对测试一

用例名称	特征码分析比对测试一
测试内容	对已知恶意病毒文件夹进行分析，分析特征码检测同一文件夹下其他文件的准确率。(在一个目录下所有的文件均为由卡巴斯基检测出的一种类型病毒和其病毒变体)
测试条件	主机 1 台
测试过程	1.在目录 virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\下有 74 个病毒 2 随机提取一个病毒其特征码为： 558bec6aff68f841400068b02f400064a1000000005064892500000000083ec68 3 测试该目录下其他病毒是否具有相同特征码 4.测试在 C:\WINDOWS 目录下是否有与该特征码重合的正常文件。 5.测试在 virus_and_classfied_by_kap\根目录下所有病毒特征码与该特征码重合的个数（根目录下共有 3979 个病毒）
测试结果	1. virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\目录下 74 个病毒特征码均一致。 2.C:\WINDOWS\ 目录下与特征码重合的文件数为 0。 3 在 virus_and_classfied_by_kap\根目录下 特征码重合个数为 74.(包括 virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\ 下的 74 个病毒)

表 5-2 特征码分析比对测试二

用例名称	特征码分析比对测试二
测试内容	对已知恶意病毒文件夹进行分析，分析特征码检测同一文件夹下其它文件的准确率。（在一个目录下所有的文件均为由卡巴斯基检测出的一种类型病毒和其病毒变体）
测试条件	主机 1 台
测试过程	1.在目录 virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\ 下有 3 个病毒 2 随机提取一个病毒其特征码为： 558bec83c4f0b89c233100e8901effffa18033310033d2e8349affffa1803331 3 测试该目录下其它病毒是否具有相同特征码 4.测试在 C:\WINDOWS 目录下是否有与该特征码重合的正常文件。 5.测试在 virus_and_classfied_by_kap\根目录下所有病毒特征码与该特征码重合的个数（根目录下共有 3979 个病毒）
测试结果	1. virus _and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\目录下 3 个病毒特征码均一致。 2.C:\WINDOWS\ 目录下与特征码重合的文件数为 0。 3 在 virus_and_classfied_by_kap\根目录下 特征码重合个数为 3.(包括 virus _and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\ 下的 3 个病毒)

表 5-3 特征码分析比对测试三

用例名称	特征码分析比对测试三
测试内容	对已知恶意病毒文件夹进行分析，分析特征码检测同一文件夹下其它文件的准确率。（在一个目录下所有的文件均为由卡巴斯基检测出的一种类型病毒和其病毒变体）
测试条件	主机 1 台
测试过程	1.在目录 virus _and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\ 下有 42 个病毒 2 随机提取一个病毒其特征码为： 558bec6aff68e831400068d024400064a1000000005064892500000000083ec68 3 测试该目录下其它病毒是否具有相同特征码 4.测试在 C:\WINDOWS 目录下是否有与该特征码重合的正常文件。 5.测试在 virus_and_classfied_by_kap\根目录下所有病毒特征码与该特征码重合的个数（根目录下共有 3979 个病毒）
测试结果	1. virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\目录下 3 个病毒特征码均一致。 2.C:\WINDOWS\ 目录下与特征码重合的文件数为 0。 3 在 virus_and_classfied_by_kap\根目录下 特征码重合个数为 51.(包括 virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\ 下的 42 个病毒) 3.1 在 virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\ando 目录下有 3 个病毒的特征码与该特征码重合。（该目录下一共 3 个病毒） 3.2 在\virus_and_classfied_by_kap\Trojan-Dropper\Win32\Small\ceh 目录下有 6 个病毒的特征码与该特征码重合（该目录下一共 6 个病毒）

表 5-4 特征码分析比对测试四

用例名称	特征码分析比对测试四
测试内容	对已知恶意病毒文件夹进行分析，分析特征码检测同一文件夹下其它文件的准确率。(在一个目录下所有的文件均为由卡巴斯基检测出的一种类型病毒和其病毒变体)
测试条件	主机 1 台
测试过程	1.在目录 virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\ 下有 92 个病毒 2 随机提取一个病毒其特征码为： 558bec6aff68f831400068b025400064a1000000005064892500000000083ec68 3 测试该目录下其它病毒是否具有相同特征码 4.测试在 C:\WINDOWS 目录下是否有与该特征码重合的正常文件。 5.测试在 virus_and_classfied_by_kap\根目录下所有病毒特征码与该特征码重合的个数（根目录下共有 3979 个病毒）
测试结果	1. virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\目录下 92 个病毒特征码均一致。 2.C:\WINDOWS\ 目录下与特征码重合的文件数为 0。 3 在 virus_and_classfied_by_kap\根目录下 特征码重合个数为 261.(包括 virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\ 下的 92 个病毒) 3.1 在\virus_and_classfied_by_kap\Trojan-Dropper\Win32\Agent\zep 目录下有 17 个病毒的特征码与该特征码重合。(该目录下一共 17 个病毒) 3.2 在\virus_and_classfied_by_kap\Trojan-Dropper\Win32\Agent\aahc 目录下有 32 个

表 5-4(续) 特征码分析比对测试四

测试结果	病毒的特征码与该特征码重合（该目录下一共 47 个病毒） 3.3 在\virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\ammj 目录下有 4 个病毒的特征码与该特征码重合（该目录下一共 4 个病毒） 3.4 在\virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akee 目录下有 21 个病毒的特征码与该特征码重合（该目录下一共 21 个病毒） 3.5 在\virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\appe 目录下有 83 个病毒的特征码与该特征码重合（该目录下一共 83 个病毒） 3.6 在\virus_and_classfied_by_kap\Trojan\Win32\Agent\amtx 目录下有 8 个病毒的特征码与该特征码重合（该目录下一共 8 个病毒） 3.7 在 \virus_and_classfied_by_kap\Trojan-GameThief\Win32\OnLineGames\tvck 目录下有 4 个病毒的特征码与该特征码重合（该目录下一共 4 个病毒）

表 5-5 特征码分析比对测试五

用例名称	特征码分析比对测试五
测试内容	对已知恶意病毒文件夹进行分析，分析特征码检测同一文件夹下其他文件的准确率。(在一个目录下所有的文件均为由卡巴斯基检测出的一种类型病毒和其病毒变体)
测试条件	主机 1 台
测试过程	1.在目录 virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\ 下有 52 个病毒 2 随机提取一个病毒其特征码为：

表 5-5.(续) 特征码分析比对测试五

测试过程	558bec6aff68e831400068d024400064a1000000005064892500000000083ec68 3 测试该目录其它病毒是否具有相同特征码 4.测试在 C:\WINDOWS 目录下是否有与该特征码重合的正常文件。 5.测试在 virus_and_classfied_by_kap\根目录下所有病毒特征码与该特征码重合的个数（根目录下共有 3979 个病毒）
测试结果	1. virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\目录下 52 个病毒特征码均一致。 2.C:\WINDOWS\ 目录下与特征码重合的文件数为 0。 3 在 virus_and_classfied_by_kap\根目录下 特征码重合个数为 53.(包括 virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\akms\下的 52 个病毒)。 3.1 在 \virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\amvt\ 目录下有 1 个病毒的特征码与该特征码重合。（该目录下一共 4 个病毒）

表 5-6. 漏报率比对测试

用例名称	漏报率比对测试
测试内容	抽取病毒特征码样本，测试病毒特征码的漏报率
测试条件	主机 1 台
测试过程	在特征码分析比对测试二中 \virus_and_classfied_by_kap\Trojan-Dropper\Win32\Agent\aahc 该目录下一共有 47 个文件其中有 32 个特征码重合。另有 15 个文件为另一个特征码。

表 5-6(续)漏报率比对测试

测试过程	15 个文件的特征码均为： 558bec6aff68f831400068102640006 4a1000000005064892500000000 83ec6 8 在特征码分析比对测试五中 \virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\amvt\ 该目录下一共有 4 个文件其中有 1 个特征码重合。另有 3 个文件为另一个特征码。 3 个文件的特征码均为： 558bec6aff68e831400068202540006 4a1000000005064892500000000 83ec68
测试结果	五个特征码分析比对用例共检测病毒数 442 个（即表 2 到表 6 一共检测的病毒数）。其中 18 个病毒变形（测试 2 中的 15 个加上测试五中的 3 个）没有检测到。 漏报率为 18/442 = 0.0407%

表 5-7.误报率比对测试

用例名称	误报率比对测试
测试内容	抽取正常文件样本，测试在 3979 个病毒特征码中的误报率
测试条件	主机 1 台
测试过程	1.在 c:\windows\ 目录下提取所有.exe 和.dll 文件特征码。一共 1685 个文件。 2.将 1685 个特征码依次与\virus_and_classfied_by_kap\ 目录下病毒特征码比对（该目录下一共 3979 个病毒）。 3.查看是否有正常文件特征码与病毒特征码重合。

表 5-7.(续)误报率比对测试

测试结果

一共有 277 个文件重合。其中 275 个文件为同一特征码,所以特征码重合数为 3

详解结果

1.1

在目录 C:\WINDOWS\assembly\CGA 下 255 个文件（该目录不能被直接访问，在 DOS 环境下才能访问。而且其中 255 个文件的特征码都一样）其特征码都为

ff25002040000000000000000000000000000000000000000000000000000000

与病毒

（1）\virus_and_classfied_by_kap\Trojan-Clicker\MSIL\Xone\bo\002898.exe

（2）\virus_and_classfied_by_kap\Trojan-Downloader\MSIL\Agent\ce\002860.exe

具有相同特征码。

1.2 在目录 C:\WINDOWS 下文件

C:\WINDOWS\assembly\NativeImages_v2.0.50727_32\Microsoft.VisualStu#\15f8922f9b1bc34babf04da06b0c34de\

下文件: Microsoft.VisualStudio.Designer.Interfaces.ni.dll

C:\WINDOWS\system32\cfx32.ocx）

其特征码为（共两个）:

00000000000000000000000000000000000000000000000000000000000000000

与病毒

\virus_and_classfied_by_kap\Trojan-PSW\Win32\Delf\cvg\002692.exe

\virus_and_classfied_by_kap\Trojan-PSW\Win32\Delf\cvg\002802.exe

具有相同特征码。

1.3 在目录 C:\WINDOWS 下文件

C:\WINDOWS\NOTEPAD.EXE 其特征码码为

6a706898180010e8bf10000033db538b3dcc100010ffd76681384d5a751f8b48

与病毒\virus_and_classfied_by_kap\not\002621.exe 具有相同特征码。

2 误报率:

1685 个文件中有 277 个重合。误报率为 227/1685 = 16%

表 5-8.特征码性能测试

用例名称	特征码性能测试
测试内容	测试提取 3979 个病毒文件所需要的时间
测试条件	主机 1 台
测试过程	1. 设置配置文件 设置提取目录为\virus_and_classfied_by_kap\ 2. 启动程序 3. 提取特征码，记录时间。
测试结果	1 提取 3979 个病毒文件特征码共用时间 4303 毫秒 2 平均单个样本提取时间为 1.081 毫秒

5.4 测试总结与存在的问题

\virus_and_classfied_by_kap\下的3797个病毒样本特征码与C:/windows/目录下的1685个正常文件特征码比对均没有发现有特征码重合的现象，该平台对同一病毒的其它变种检测达到了预期效果。

1 漏报率为：0.0407%

2 误报率为：16%（对不同类型病毒样本，提取相同特征码）

3 提取 3979 个病毒文件特征码共用时间 4303 毫秒

4 平均单个样本提取时间为 1.081 毫秒

但在实验过程也发现该系统也有许多不足之处，具体如下：

1.该系统并不能提取出\virus_and_classfied_by_kap\目录下所有的病毒特征码，在提取某些病毒时，抛出入口点地址定位超出程序大小的异常。这些病毒为：

\virus_and_classfied_by_kap\not\002653.exe

\virus_and_classfied_by_kap\not\11\002652.exe

\virus_and_classfied_by_kap\Trojan\Win32\Pakes\jtc\000560.exe

\virus_and_classfied_by_kap\Trojan-GameThief\Win32\Magania\bng\000831.exe

\virus_and_classfied_by_kap\Worm\Win32\AutoRun\ruw\000050.exe

\virus_and_classfied_by_kap\Trojan-PSW\Win32\QQRob\bv\002873.exe

出现此类问题的原因可能是这些病毒并未彻底脱壳。可能还存在某种壳导致分析异常。

2.该系统在\virus_and_classfied_by_kap\目录下提取出的文件有部分特征码为全0，这些病毒为：

\virus_and_classfied_by_kap\Trojan-Downloader\Win32\Agent\bafy\000014.exe

\virus_and_classfied_by_kap\Worm\Win32\Hipak\a\000583.exe

\virus_and_classfied_by_kap\Trojan-PSW\Win32\Delf\cvg\002802.exe

\virus_and_classfied_by_kap\Trojan-PSW\Win32\Delf\cvg\002692.exe

出现此类问题的原因可能是特征码提取地点有误，或也可能为文件并未彻底脱壳，所以未找到真正的程序入口点。

5.5 小结

本节首先介绍了测试需要用到的软硬件环境，然后列举出了待分析的病毒样本和所要进行的测试内容，接着针对每个测试内容进行具体测试并给出测试结果，最后对测试结果进行总结并提出了系统还存在的不足之处。

第六章 总结与展望

6.1 总结

本文从恶意代码国内外的研究现状入手，首先从研究文件感染技术开始分析了恶意代码的感染策略，其中着重分析了重写病毒，压缩病毒和变形感染技术，对 Win32 和 Win64 病毒与 Mircosoft Windows 系列平台的关系进行了总结。然后介绍了壳的概念和壳常见的加壳过程，并例举了目前比较流行的常见压缩壳和加密壳的种类和工作原理。然后深入分析了 Win32 病毒感染策略和 PE 文件的关系。其中介绍了 PE 文件的构成和其中关键的数据结构和 32 位 Windows 感染技术。然后分析了常见的壳检测方式和特征码提取方式，提出了常见特征码提取方式的不足，提出了基于 PE 文见启发式特征码自动提取方法。最后将该提出的新算法实现于基于 PE 文件启发式特征码自动提取平台上，并取得了较好的效果。本文取得了以下的成果。

1.提出了基于 PE 文件启发式特征码提取算法，启发式指的“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。”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 PE 文件结构分析和扫描器，通过前面对病毒感染策略的分析，提出了病毒可能修改和控制的 PE 结构的启发性标志，当然我们的对象是已知的病毒样本，这些病毒样本与正常文件的 PE 结构一定会有不同的区别，这是由病毒复制自身和传播的特性决定的，找到这些容易被病毒修改的 PE 结构，将其设置为启发性标志，在提取每个病毒样本时，依次分析这些病毒样本是否具有启发性标志，通过启发性标志生成特征序列，然后对特征序列进行筛选，用那些最能代表病毒特征的特征序列组成特征码。

2.根据提出的特征码提取的新思路，设计并实现了基于 PE 文件启发式特征码自动提取平台。该平台运用了上述两种提取特征码的思路，并实现了特征码的自动提取。

3.最后通过实验进行了论证，并得到了预期的效果，提取的特征码的误报率，漏报率和检测病毒变种的能力都达到了理想的效果，但从实验结果看出该平台还有很多需要改进的地方和空间。

6.2 展望

虽然基于PE文件启发式特征码自动提取平台对提取特征码的效果达到了预期设计的目标但经过试验发现还有很多不足之处，经过总结发现要从本质上解决这些不足需要从不断的优化算法和提出新的设计思路上下手。具体如下：

1.对病毒启发性标志的选择上应在通过大量实验数据进行总结，优化特征序列生成算法和特征码组合方式，使提取出的特征码更具有代表性，进一步降低特征码的误报率后漏报率，以及加强特征码检查病毒变种的能力。

2.优化特征码的格式，如可以加入其它特征码提取算法，如通配符、标签法或精确定位法，使特征码具有更好的通用性。

3.优化该系统代码和算法的基础上进一步加强该平台提取特征码的效率，使其能接近商用杀毒软件特征码的匹配速度。

4.优化特征码库，使特征码的存储和查找速度更快，并能在收集大量特征码后仍具有高效的查找和存储速度。

致谢

在研究生学习期间，我得到了来自老师和同学的很多关怀和帮助，从他们身上我学到了很多东西，让我懂得了很多做人做事的道理。

首先，要特别感谢我的导师耿技教授。在刚进入研究生学习时，他给我做好了良好的学习和生活规划，给我指明了方向。在学习上，他热心指导我，传授给我好的学习方法，使我在学习上得到了很大的进步；在生活上，他时常了解我的情况，当我遇到困难的时候给我作出指导，帮我渡过了很多难关；在实际项目中，他给我提供了宝贵的机会，让我得到了锻炼。我不仅在实践能力方面得到了提高，同时也增强了我做事的信心。

感谢我的项目指导老师周世杰，在本次论文完成过程中和在实验室学习的两年时间中，周老师给了我悉心的指导和监督，周老师是我见过最认真对学生最负责的老师。在周老师的团队里，我不仅学到了许多理论知识和前沿技术更重要的是我学到了周老师对事严谨和认真的态度，和永不放弃的精神。这将是我以后工作生活中不可多得的财富。

感谢师兄陈振、范成渝和杨睿以及师姐杨静婷在研究中对我的指导和鼓励，每次与师兄师姐讨论交流都能使我获得新的建议和研究思路，不断的在研究上取得进展。感谢同一教研室的同学，蒋小风、王哲、张圳、敬锐、刘乐源、任昭宵、孙正隆、白绪红等，和他们的相处与交流是令人颇为愉快的一件事，他们的朝气蓬勃不仅催我奋进，亦常带给我激励和启发。感谢同一项目组的同学余圣、陈陪和周佩颖。给予我在学习和生活上的帮助。

其次，我要感谢和我一起学习生活的朋友们特别是王帷萍学妹，正是由于你们的支持和帮助，我才能顺利地完成研究生阶段的学习和工作。

最后，我要感谢我的父母和亲人，你们在精神和物质上给了我大力的支持。感谢秦志光院长，感谢计算机学院、软件学院的所有老师，给我提供这么好的学习和生活环境。特别感谢王巧师姐，在我研究生三年中给我的莫大帮助。

参考文献

[1] 秦志光，张凤荔,计算机病毒原理与防范.北京：人民邮电出版社，2007.

[2] 张仁斌，李钢，侯整风,计算机病毒与反病毒技术.北京：清华大学出版社，2006.

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

[4] 韩筱卿，计算机病毒分析与防范大全 北京：电子工业出版社，2008.

[5] 约瑟夫·麦科明克&苏珊·费雪贺区 著，何颖怡 译,第四级病毒.山东：汕头大学出版社，2004.

[6] 慈庆玉.计算机变形病毒技术探讨[J].中国数据通讯,2005,1(1):37-40.

[8] 田畅, 郑少仁. 计算机病毒计算模型的研究. 计算机学报. Vol.24, No.2, 2001：43-49

[9]. P.-C. Lin et al., "Using String Matching for Deep Packet Inspection," Computer, vol. 41,no. 4, Apr. 2008, pp. 23-28.

[10] N. Agrawal, W. Bolosky, J. Douceur, and J. Lorch, "A five-year study of file-system metadata," Proceedings of the 5th USENIX conference on File and Storage Technologies, p.3-3, San Jose, CA, February 2007

[11] W. Yan, "Revealing Self-similarity in NTFS File Operations," poster paper, Proceedings of the 7th USENIX Conference on File and Storage Technologies, San Francisco, CA, February 2009 Authorized

[12] W. Yan, Z. Zhang, and N. Ansari "Revealing packed malware," IEEESecurity and Privacy, vol. 6, no. 5, pp. 65-69, Sep/Oct, 2008

[13] C. Kruegel, W. Robertson, and G. Vigna, "Detecting Kernel-Level RootkitsThrough Binary Analysis", Proceedings of 20th Annual ComputerSecurity Applications Conference, pp. 91-100. Tuscon, AZ, December,2004.

[14] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh. On the effectivenessof address-space randomization. In Proceedings of the ACM Conference on Computer and Communications Security, pages 298–307, 2004.

[15] Linux-NTFS Project, NTFS Documentation, http://www.linux-ntfs.org

[16] R. Nagar, Windows NT File System Internals, O'Reilly, 1997.

[17] W. Leland, M. Taqqu, W. Willinger and D. Wilson, "On the self-similar nature of Ethernet traffic", IEEE/ACM Transactions on Networking, vol.2, no.1 pp. 1-15, 1994.

[18] J. R. Douceur and W. J. Bolosky, "A large-scale study of file-system contents", Proceedings of 1999 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pp. 59–70, Atlanta,Georgia, June, 1999.

[19] SNORT, http://www.snort.org, 2008.

[20] S. Kim, "Pattern Matching Acceleration for Network Intrusion Detection Systems," Proc. Fifth Int'l Workshop Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2005.

[21] D. Kim, S. Kim, L. Choi, and H. Kim, "A High-Throughput System Architecture for Deep Packet Filtering in Network Intrusion Prevention," Proc. 19th Int'l Conf. Architecture of Computing Systems (ARCS), 2006.

[22] H.C. Roan, W.J. Hwang, and C.T. Lo, "Shift-Or Circuit for Efficient Network Intrusion Detection Pattern Matching," Proc. 16th Int'l Conf. Field Programmable Logic and Applications (FPL), 2006.

[23] Kreibich, C., Crowcroft, J. Honeycomb - Creating Intrusion Detection Signatures Using Honeypots. ACM SIGCOMM Computer Communication Review 34 (2004) 51-56.

[24] S.Singh, C.Estan, G.Varghese, and S.Savage. Automated worm fingerprinting. In Proc. OSDI, 2004.

[25] V.Yegneswaran, J.Giffin, P.Barford, and S.Jha. An architecture for generating semantic-aware signatures. In USENIX Security Symposium,2005

[26] K. Wang, G. Cretu, and S. J. Stolfo. Anomalous Payload-based WormDetection and Signature Generation. In Symposium on Recent Advances in Intrusion Detection, 2005.

[27] J. Newsome, B. Karp, and D. Song. Polygraph: Automatically generating signatures for polymorphic worms. In IEEE Security and Privacy Symposium, 2005.

[28] Zhichun Li, Manan Sanghi, Yan Chen, Ming-Yang Kao and Brian Chavez. Hamsa: Fast Signature Generation for Zero-day Polymorphic Worms with Provable Attack Resilience. In IEEE Symposium on Security and Privacy, 2006.

[29] Bob Netherton. DTrace. Solaris 10 Workshop, 2005.

[30] http://nepenthes.mwcollect.org

[31] http://www.few.vu.nl/argos

[32] S.Staniford, V.Paxson and N.Weaver. How to Own the Internet in Your Spare Time. In Proc. of the 11th USENIX Security Symposium, pp.149-167, USENIX Association, 2002.

[33] S.Staniford, D.Moore, V.Paxson and N.Weaver. The Top Speed of Flash Worms. In Proc. of RAID, 2004.

[34] N.Weaver. Potential Strategies for High Speed Active Worms : A Worst case Analysis, 2002.

[35] J.Nazario. Defense and Detection Strategies against Internet Worms.2005.

[36] Wikipedia, http://en.wikipedia.org/wiki/Computer_virus

[37] Peter Szor. The Art of Computer Virus Research and Defense, 2005.

[38] K.-A. Kim and B. Karp. Autograph: Toward Automated Distributed Worm Signature Detection. In Proc. of the USENIX Security Symposium, 2004.

[39] C. T. Smirnov A. Dira: Automatic detection, identification, and repair of control-hijacking attacks. In Proceedings of NDSS05: Network and Distributed System Security Symposium Confrence Proceedings, San Diego, California, February 2005. Internet Society.

[40] snort.org. Snort, an open source network intrusion detection system.

[41] Y. Tang and S. Chen. Defending against internet worms: A signature-based approach. In Proceedings of IEEE INFOCOM'05, March 2005.

[42] T. G. team. Gcc, the gnu compiler collection ,http://gcc.gnu.org/. Online, 2006.

[43] N. T.J.Bailey. The Mathematical Theory of Infectious Diseases and its Applications. Griffin,2nd edition, 1975.

[44] A.Wagner, T. Dübendorfer, B. Plattner, and R. Hiestand. Experiences with worm propagation simulations. In WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode,pages 34–41, New York, NY, USA, 2003. ACM Press.

[45] D. Wagner and D. Dean. Intrusion detection via static analysis. In Proceedings of the 2001 IEEE Symposium on Security and Privacy, 2001.

[46] D. Wagner and P. Soto. Mimicry attacks on hostbased intrusion detection systems. In Proceedings of the 9th ACM Conference on Computer and Communications Security, November 2002.

[47] S. S. Wang K. Anomalous payload-based network intrusion detection. In Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection, Sophia Antipolis,France, September 2004.

[48] D. C. D.Wei Xu and R. Sekar. An efficient and backwardscompatible transformation to ensure memory safety of c programs. In Proceedings of ACM SIGSOFT/FSE, Newport Beach, USA,November 2004.

攻硕期间取得的研究成果

参加的科研项目

[1] 2007.9 — 2007.11 病毒分析报告

[2] 2007.11 — 2008.1 恶意网页搜索引擎

[3] 2008.3 — 2008.4 华为基金蜜罐项目后期建设

[4]2008.7 –2008.12 修复蜜罐项目中的网路数据包捕获缺陷

[5] 2008.12 – 2009.4 病毒特征码自动提取系统

基于PE文件的启发式特征码自动提取的研究

作者：陈晋福
学位授予单位：电子科技大学