分类号 _____ 密级 _____ 1814045

UDC[注1] _____

学 位 论 文

基于异常用户行为的蠕虫检测与特征码自动提取技术研究

（题名和副题名）

陈大鹏

（作者姓名）

指导教师姓名 张小松 副教授

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 计算机应用技术

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 _____

2010 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：陈大鹏　　日期：2010 年 5 月21 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：陈大鹏　　导师签名：

日期：2010 年5 月21 日

摘 要

当今对计算机网络安全提出最大挑战的是起源于上世纪80年代的网络蠕虫。蠕虫因其传播隐蔽，感染速度快，造成损失大的特点使得对蠕虫的研究越来越受到人们的重视。现阶段，蠕虫的检测手段仍然集中在基于特征码的误用检测和基于网络行为的异常检测上，但在检测手段的创新上遇到了一定的瓶颈。另外，传统的基于人工的蠕虫特征码提取方式所具有的滞后性极大的影响着蠕虫的检测和蠕虫的遏制；而处于起步阶段的早期蠕虫特征码自动提取技术则在面对变形蠕虫和反特征码自动提取技术时显得力不从心。因此，提出一种新的蠕虫检测方法和蠕虫特征码自动提取方法具有重要意义。

本文综述了计算机蠕虫的相关原理、关键技术、检测方法以及蠕虫特征码提取的相关技术，并提出了一套新的蠕虫检测方法和蠕虫特征码自动提取方法，在一定程度上弥补了现有技术的不足。

本文的主要工作和贡献包括：

详细的概述了蠕虫的原理和现有的蠕虫检测技术、蠕虫特征码提取技术并分析了现有蠕虫检测技术和蠕虫特征码提取技术的优点和不足。同时对未来蠕虫的发展和蠕虫检测于特征码提取技术的发展做出了预测。

目前的蠕虫检测主要由基于特征码的误用检测和基于蠕虫网络行为的异常检测组成。其中异常检测主要通过利用蠕虫在扫描阶段产生的大量失败连接为依据从而分析出网络中是否存在蠕虫，而不是结合用户网络行为进行蠕虫检测。本文另辟蹊径，结合现有技术的优点，提出了基于异常用户行为的蠕虫检测方式，有效避免了非蠕虫行为干扰的问题，提高了蠕虫检测的准确率。

本文还针对蠕虫特征码人工提取和早期特征码自动提取技术的不足，提出了抗攻击蠕虫特征码自动提取技术，有效的弥补了人工提取的滞后性和早期特征码自动提取技术在变形蠕虫特征码提取和抗反特征码自动提取攻击能力差的不足。这也是对蠕虫特征码自动提取技术的有益尝试。

关键词：蠕虫，异常用户行为，特征码自动提取，抗攻击

ABSTRACT

The computer worm which originated in the 80s of last century is seen as the biggest challenge in the computer network security area at present. Because of the worm's imperceptibility, fast infection and enormous damage, the worm research becomes more and more important. At present stage, the Misuse-based technology and Anomaly-based technology are still the main methods of worm detection meet a bottleneck. On the other hand, the hysteresis of manual worm signature generation may bring some negative effects on worm detection and restraint. The auto-signature generation technology which in the development stage may show its weakness when meets the polymorphic worm or the anti-auto-signature generation technology. Therefore, it is important to propose a new worm detection method and a new worm auto-signature generation method.

This thesis surveys the principles, key techniques and detection approaches of worm detection and the worm signature generation approaches. Then we propose a new worm detection method and a new worm auto-signature generation method to cover the shortage of former technology.

We have the following three contributions:

1. We describe the principles, worm detection approaches and the worm signature generation approaches. Then we analyze the advantage and the weakness of current worm detection approaches and the worm signature generation approaches. At last, we propose a forecast about the trend of development of worm detection and worm signature generation technologies.

2. The existing worm detection technology is based on the Misuse-based technology and Anomaly-based technology. The Anomaly-based technology detect the worm often by analyze the multiple false-connections which generate by the worm in its scan stage, but not by the user's behaviors. To create some new work, combining with the advantages of existing technology, we propose an anomaly-based technology based on the abnormal user-behavior which can avoid the similar worm behaviors disturbance and enhance the accuracy rate of worm detection.

3. This thesis proposes a self-immune signature generation technology to solve the deficiencies of methods of manual-signature generation and the former auto-signature generation. It makes up the hysteresis of manual worm signature generation and the weak ability to resist the anti-auto-signature generation technology. This work is an initial attempt for worm signature generation technology.

Keywords: Computer worm, Abnormal user-behavior, Automated signature generation, Self-immune

目 录

第一章 绪论

1.1 研究目的与意义

在21世纪的今天，世界信息化程度日趋加深，计算机和互联网的利用已从过去的科研和国防领域拓展到现在的日常生活领域，人们对计算机以及互联网的依赖程度大大加强。计算机和互联网的普及，极大地促进了世界政治、经济和社会的发展，提高了人们的生活水平。但是各种计算机病毒，特别是充斥在网络中的计算机蠕虫病毒给计算机和互联网安全带来了极大的挑战。从1982计算机蠕虫病毒概念的提出[1]到1988年世界上首个蠕虫病毒爆发，再到2001年CodeRed[2]蠕虫的爆发，在短短的9小时之内就使得25万台计算机处于瘫痪状态，造成的损失超过了20亿美元。而近几年出现的震荡波，熊猫烧香等恶性蠕虫更是造成了难以估量的损失。

由CERT（计算机应急响应小组）的统计，自1988年以来，统计到的Internet安全事件每年基本接近指数增长，如图1-1所示。另据国家互联网应急中心（CNCERT）统计显示，2009年1月至2009年2月，CNCERT/CC累计捕获恶意代码144805次，平均每日捕获更是达到了2454次。蠕虫作为恶意代码的一种，其所占比例也在与日俱增。如何应对疯狂增长的计算机蠕虫病毒，是世界信息安全领域面对的一个重大课题。

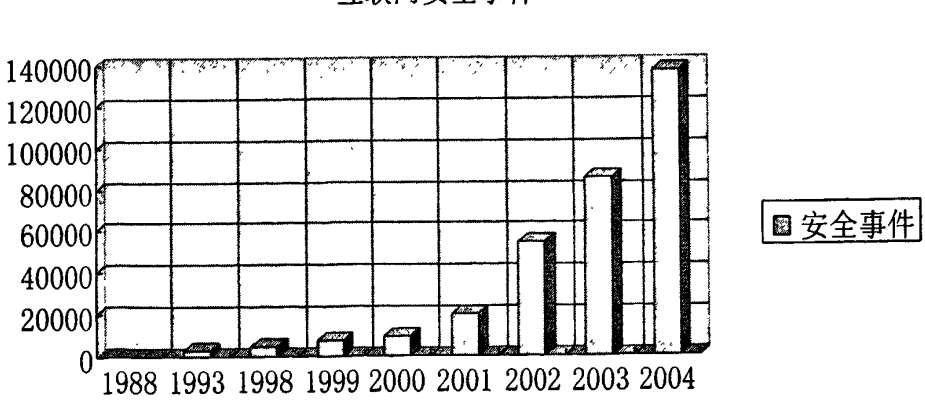

图1-1 互联网安全事件统计图

在这一背景下，如何研究出高效、可靠的蠕虫检测技术，研究出快速、准确的特征码自动提取技术将具有重要的意义。高效、可靠的蠕虫检测技术就是指在蠕虫检测上能够快速的分析识别出目前网络中是否存在异常从而发现是否有蠕虫的存在；可靠是指在检测过程中，能够避免受到类似蠕虫的正常程序，如：迅雷、电驴等 P2P 软件的影响，能够防止在面对慢速扫描蠕虫时出现漏检的情况，并且能够尽可能的检测出未知蠕虫。蠕虫检测技术能否高效、可靠的检测蠕虫直接关系到蠕虫的进一步防范和遏制，关系到能否尽可能的减少蠕虫带来的各项损失。另一方面，研究出快速、准确的特征码自动提取技术能够为蠕虫的高效、可靠的检测提供强有力的支持。特征码检测属于误用检测技术，其特点就是准确度高，速度快，但特征码技术不能用于对未知蠕虫的检测。因此，只有将两者有机的结合起来才能够保证蠕虫的有效检测。

1.2 研究背景

1.2.1 蠕虫的原始定义

“蠕虫”本是一个生物学的名词，最早出现在一本 1975 年出版的名为《Shockwave Rider》的科幻小说中，后来由 Xerox PaloAlto Research Center (PARC) 于 1980 年引入到计算机领域之中。但在当时，引入“蠕虫”这个定义到计算机领域是为了解决分布式计算的问题，而不是用以进行计算机破坏和网络攻击。1982 年时，Xerox PARC 的 John F.Shoch[3]等人描绘出了蠕虫病毒的两个最基本的特征：第一个是“可以从一台计算机移动到另一台计算机”，第二个则是“可以自我复制”。计算机网络蠕虫又有别于通常的计算机病毒，他是恶意代码中的一种，通常意义上的计算机病毒是 “一段代码，能把自身加到其他程序包括操作系统上，它不能独立运行，需要由它的宿主程序运行来激活它”，这一定义是由 Fred Cohen 在 1984 年从技术角度出发给出的[4]。二者都具有一定的相似性，如都是一段具有隐蔽性的代码或程序、都具有复制功能、都能够给计算机安全造成危害，但他们也有明显的区别，如存在形式、攻击方式、感染形式、触发方式、危害要点等。为了更好的弄清计算机网络蠕虫和计算机病毒的区别，下表对计算机网络蠕虫和计算机病毒的比较：

表 1-1 计算机网络蠕虫和计算机病毒的比较

类别	计算机网络蠕虫	计算机病毒
形式	单独程序	依附于其他程序中
直接攻击方式	利用计算机漏洞攻击	利用自身程序达到目的
感染形式	独立完成自我复制	借助寄主完成复制
触发方式	自动运行	宿主程序运行
危害要点	计算机网络和主机	计算机主机文件系统

1.2.2 蠕虫的完整定义及特点

根据 Eugene H. Spafford 的关于蠕虫的最早定义[5]：“网络蠕虫是通过网络传播，无须用户干预能够独立地或者依赖文件共享主动攻击的恶意代码。”而 Kienzle 和 Elder 分别从破坏性、网络传播方式、主动攻击性和独立性 4 个方面对蠕虫进行了定义[6]，即“网络蠕虫是无需计算机使用者干预即可运行的独立程序，它通过不停地获得网络中存在漏洞的计算机上的部分或全部控制权来进行传播”。归纳上述两种定义，目前蠕虫主要具有表 1-2 所列的共同特性：

表 1-2 蠕虫具有的共同特性

特 性	解 释
自我复制性	蠕虫能够自动的以自身为模板复制自身
传播性	蠕虫应具有在网路中传播的能力
独立运行性	蠕虫能够独立运行，无需用户活动的支持，这是蠕虫区别于普通计算机病毒的重要特点
利用漏洞传播	蠕虫利用漏洞进行攻击和传播，正是由于各种软件必然存在漏洞的特点，让蠕虫具有了广泛的生存空间

1.2.3 蠕虫的分类

根据蠕虫的传播策略，可以将蠕虫主要区分为 4 类：Email 蠕虫、文件共享蠕虫、传统蠕虫和 P2P 蠕虫。Email 蠕虫，一般是通过在电子邮件中附带含有蠕虫的附件，通过能够吸引用户关注的电子邮件内容诱引用户打开附件激活蠕虫。文件共享蠕虫通过将蠕虫放置在网络里，修改文件名为用户常用的软件或资料，

引诱用户下载并运行。传统蠕虫则是利用漏洞进行攻击，并达到感染用户计算机和网络的目的。而 P2P 蠕虫则是目前并没真实出现，但已经具有技术可行性的一种新型概念性蠕虫。这种蠕虫通过感染 P2P 软件，从而获得对等机用户列表并进行攻击和感染的蠕虫。这种蠕虫具有高效，感染准确，失败连接少的特点，并且它将混杂在正常的流量中，检测难度极大。

另外，根据蠕虫扫描速度受限类型来分，又能把蠕虫分为“扫描延迟限制型”和“带宽限制型”[6]，但是，P2P 蠕虫作为一类特殊的蠕虫，不能包括在内。其中，“扫描延迟限制型”蠕虫先发出扫描探测包，然后等待连接延迟到达，将产生突发的扫描流量，其扫描速度受限于连接延迟，典型代表是 CodeRed 蠕虫。因此可以通过探测网络中突发流量来检测蠕虫。而“带宽限制型”蠕虫则通过连续发送扫描包，其扫描速度受限于链路带宽，典型的代表是 SQL Slammer 蠕虫[7]。

1.2.4 蠕虫的功能及工作机制

在本文进行蠕虫检测和蠕虫特征码提取的讨论之前，先就蠕虫的功能以及工作机制做一个简单的描述，以便后续展开进一步的讨论。

根据蠕虫的定义和蠕虫的特点，已经蠕虫的基本技术构成，我们可以将蠕虫按功能结构大致分为主功能模块和辅功能模块两个部分[8][9]。具体划分见图 1-2：

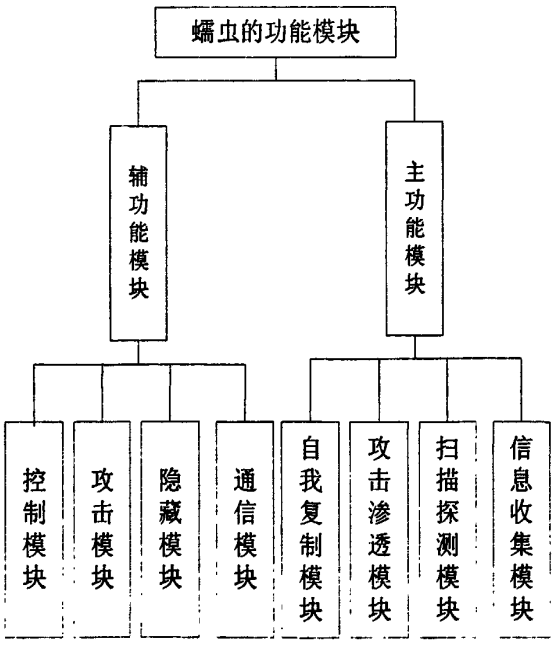

图 1-2 蠕虫的基本功能模块

由图可知主功能模块主要分为四个功能模块：1）信息收集模块，该模块主要负责收集目标网络中的各种信息，用来确定攻击目标；2）扫描探测模块，该模块主要负责对确定目标进行进一步的扫描探测，通过其具有的漏洞，采取相应的攻击；3）攻击渗透模块，该模块主要利用扫描探测模块获得的信息，进行攻击并建立信息传播途径；4）自我复制模块，该模块主要负责以自身为副本进行自动的自我复制，并利用攻击渗透模块建立的信息转播途径完成蠕虫副本的传递。

辅功能模块主要分为四个子功能模块：1）通信模块，该模块主要负责宿主蠕虫与客户端以及蠕虫之间的通信联系，包括信息的传递，控制命令的传递，以及其他指令的传输。2）隐藏模块，该模块主要是为了提高蠕虫的生存能力，包括防检测，防查杀能力，通过对蠕虫自身实体部分进行伪装，隐藏，变形，加密等方式，达到反检测的目的。3）攻击模块，该模块主要负责破坏被感染主机的敏感数据和防御软件等，达到破坏计算机的正常工作环境的目的，为下一步进攻做好铺垫。4）控制模块，该模块主要负责对蠕虫行为的控制，和对被感染主机的控制。

计算机网络蠕虫的工作机制与生物学中的蠕虫感染机制有相似之处，它的主要攻击过程分为以下几步：探测、感染、执行负载[10]、自我复制、再传播，下面本文将部分结合图示分别描述各个过程。

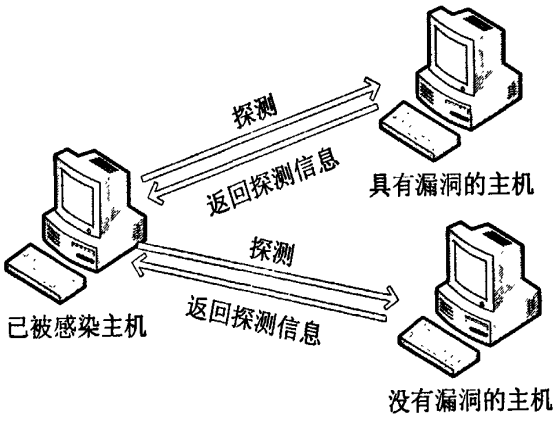

图 1-3 蠕虫的探测过程

如图 1-3 所示，蠕虫的探测过程，也就是蠕虫的扫描阶段，主要是探测网络中是否存在具有漏洞的主机，同时返回相关信息，如果发现具有漏洞的主机，则可以发起下一步的感染过程。但这个过程是常规蠕虫才具有的一步，其他蠕虫如：E-mail 类蠕虫则不具备此过程。

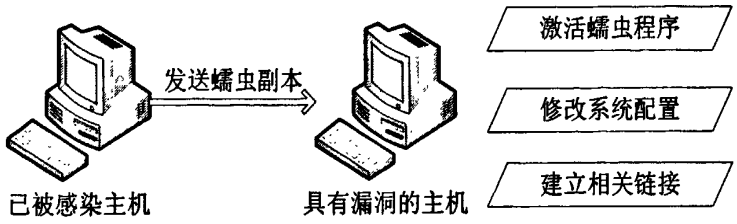

图 1-4 蠕虫的感染过程

如图 1-4 所示，蠕虫在发现可以利用的主机时，便利用主机漏洞将其副本安置在此主机中，同时激活蠕虫程序。蠕虫程序被激活后，将修改系统相关配置，清除系统的各种安全防护措施，同时建立相关链接为下一步做准备。

接下来，具有负载的蠕虫将执行负载。负载是能够完成黑客攻击意图的代码，通过执行负载，蠕虫便能完成黑客赋予的相应的攻击任务。

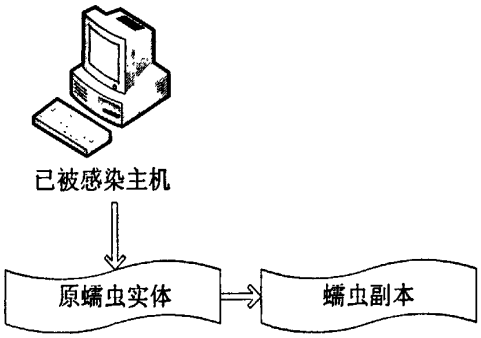

图 1-5 蠕虫的自我复制过程

如图 1-5 所示，在已感染的主机中原蠕虫实体，通过自我复制模块，以自身为副本进行自我复制。将复制出的副本用于下一次蠕虫的传播。

最后，蠕虫通过重复前续步骤到达寻找下一个具有漏洞的主机，并传播副本进行感染的目的。

以上过程循环往复，就将感染大量的主机，对计算机安全和网络安全带来严重的威胁，同时也会造成巨大的损失。

1.3 国内外研究现状

面对计算机网络蠕虫对计算机安全和网络安全造成的巨大威胁和在蠕虫爆发后给用户以及社会带来的巨大损失，各个科研机构投入了相当的力量对蠕虫相关技术进行了研究，并取得了一定的研究成果。下面，本文将就目前国内外蠕虫相

关技术做一个简单的概括总结。

1.3.1 蠕虫检测技术现状

蠕虫的检测，目前已经有多种方法，从大体上将，可以将这所有的方法分为两类：基于特征码（误用）的检测方法和基于异常的检测方法[11][12]。

基于特征码的蠕虫检测方法是一种传统的检测方法[12]，并广泛应用于各种IDS 中，这种检测方法通常是用来检测已知蠕虫。蠕虫特征码有不同的定义，但目前几乎所有的 IDS 所采用的蠕虫特征码都属于一类：内容特征码，这种特征码是出现在蠕虫数据包中的字符串。

不严谨的说，由于基于特征码的蠕虫检测方法的滞后性催生了基于异常的蠕虫检测方法。所有快速传播的蠕虫都会产生大量的网络流量，大多数蠕虫采用的盲目扫描方式选择目标计算机，这样又会产生许多失败连接[13][14][15]。但又因为正常情况下，网络状况在长时间内趋于稳定，上述蠕虫行为必然会造成异常的网络现象[16]。另外，基于异常的蠕虫检测方法还可以通过对蠕虫数据包的数据异常检测，由于正常的网络通信中的数据包包含的基本是数据，而不是代码。但是蠕虫数据包中必须包含相当数量的可执行代码，这使得蠕虫数据包和正常数据包负载有明显不同，文章[17][18][19]就分别叙述了基于返回地址的蠕虫检测技术、基于数据包统计的蠕虫检测技术、基于数据包负载可执行特性的蠕虫检测技术。基于异常的蠕虫检测方法重点就在于如何刻画这些异常现象来体现蠕虫与正常网络程序的不同。

1.3.2 蠕虫特征码自动提取技术现状

当蠕虫被通过异常检测方法检测出以后，我们还是需要对蠕虫进行特征码提取，以供后期的检测，因为基于特征码的检测方法仍然是较为快速，低误报的方法。但是蠕虫的特征码提取早期仍是又人工进行分析提取，效率很低。如果遇到大规模蠕虫爆发，人工提取的速度是远远不够的，因此特征码的自动提取就被提上了研究日程。蠕虫特征码的自动提取又主要分为以下两种：

单字符串型特征码自动提取方法是最早提出来的，它的基本思想是整个内容特征码自动提取的基础。对于这种提取方法的研究，Honeycomb[20]，Autograph[21]和 EarlyBird[22]已经研究得比较深入。这种提取方法是基于一个共同的假设：同一个蠕虫的不同副本中，必然存在着一个共同的字符串，此字符串在其他蠕虫副本

和正常网络流量中不会出现，并且这个字符串的长度不会太短（16 个字节以上）。而蠕虫特征码就正好是这个字符串。

另外一种特征码提取方法就是多 token 型特征码，有一种不严谨的说法，多 token 型特征码是为了应对未来高度变形的蠕虫的特征码自动提取需求而产生的，这里所提到变形既包括了 polymorphic 蠕虫也包括了 metamorphic 蠕虫。高度变形的蠕虫就是指变形程度很高，相同蠕虫的任意两个副本从外形来看相似度非常低，这种蠕虫在现实中非常少见，目前只是在学术上进行研究。多 token 型特征码自动提取技术重点在于 token 的提取和 token 的筛选，文献[23] 提出了各自的方法。

但随着蠕虫的不断发展，目前的一些检测办法和特征码提取方法已经不能较好的检测和进行蠕虫特征码提取，因此提出一种更好的蠕虫检测方法和特征码提取方法将是本论文研究的重点。

1.4 本文的主要研究工作

总的来说，虽然蠕虫检测技术和蠕虫特征码技术已经成为一个研究热门，但是蠕虫的检测根本思想没有大的改变，除基于误用检测以外，基于异常行为的蠕虫检测主要存在着以下问题：(1)主要依靠失败连接检测,即：大量的检测方法都是依靠蠕虫传播时期产生的大量失败连接进行的，通过对失败连接的分布进行统计分析，产生出了一大批检测方法，这些检测方法中不乏有效果较好，误报漏报都较低的。但是，万变不离其宗，仅仅依靠失败连接这一单一的蠕虫行为进行检测将受到蠕虫技术更新和其他正常软件干扰检测的挑战。(2)检测方式中大量采用机器学习的方式:在目前的大多数基于行为的蠕虫检测方式中，采用机器学习进行检测的技术也占了相当的比例，不论是对正常用户的失败连接分布学习，还是对蠕虫本身的行为进行归类学习，都离不开构建知识库。当然机器学习能够带来一定的好处，即可以检测一些未知蠕虫，但是机器学习最主要的问题还是几种在学习集合的选择上，如果学习集合选择不够合理，那学习效果就不佳，带来的将是检测中高的误报和漏报率。(3)蠕虫检测技术相对独立：目前的蠕虫检测技术大部分都是在网络中进行流量分析从而进行蠕虫检测，也有部分是通过主机异常行为进行蠕虫检测，但是两者并没有形成较好的联动。检测相对独立。这也为蠕虫的定位和遏制带来了不便。因此我们可以尝试通过将网络检测和主机检测联动起来以提升检测效率，降低漏报和误报率。(4) P2P 蠕虫带来的挑战：P2P 蠕虫传播隐蔽，检测困难，已经向传统的蠕虫检测提出了新的挑战。如何检测 P2P 蠕虫将成为亟

待解决的问题。在蠕虫特征码提取方式上，基于 token 的蠕虫特征码提取还需要改进，特别是目前反自动提取方法的进步，我们更需要改进这些算法来防止恶意的攻击。同时分流算法的选择和改进也是有待考虑和研究的问题。

针对以上问题，本文以蠕虫的检测和蠕虫特征码自动提取为核心，紧扣检测与特征码自动提取两大主题，深入剖析了蠕虫的相关原理、关键技术与目前国内外研究前沿成果。本文的主要贡献有以下几点：

1. 详细的概述了蠕虫的原理和现有的蠕虫检测技术、蠕虫特征码提取技术并分析了现有蠕虫检测技术和蠕虫特征码提取技术的优点和不足。同时对未来蠕虫的发展和蠕虫检测于特征码提取技术的发展做出了预测。

2. 目前的蠕虫检测主要由基于特征码的误用检测和基于蠕虫网络行为的异常检测组成。其中异常检测主要通过利用蠕虫在扫描阶段产生的大量失败连接为依据从而分析出网络中是否存在蠕虫，但如果网络行为出现类蠕虫行为干扰如：P2P 软件产生的网络行为，则这种异常检测方法将产生很多误报。本文针对上述不足，结合现有技术的优点，提出了基于异常用户行为的蠕虫检测方式，有效避免了非蠕虫行为干扰的问题，提高了蠕虫检测的准确率。

3. 本文还针对蠕虫特征码人工提取和早期特征码自动提取技术的不足，提出了抗攻击蠕虫特征码自动提取技术，有效的弥补了人工提取的滞后性和早期特征码自动提取技术在变形蠕虫特征码提取和抗反特征码自动提取攻击能力差的不足。这也是对蠕虫特征码自动提取技术的有益尝试。

1.5 论文组织结构

论文分为六章，内容组织如下：

第一章：“绪论”，介绍论文的研究背景，国内外研究现状简介，主要研究工作和论文结构。

第二章：“蠕虫检测相关技术研究”，分别介绍了蠕虫的相关原理、关键技术及检测方法。

第三章：“蠕虫特征码自动提取相关技术研究”，介绍了蠕虫特征码自动提取的基本思路，和目前蠕虫特征码自动提取的基本技术和面临的困难。

第四章：“基于异常用户行为的蠕虫检测技术”，提出利用机器学习和异常检测的思想对用户主机行为进行分析，从而发现异常并以此来判断是否存在蠕虫攻击，同时也通过实验对技术进行相关分析。

第五章：“抗攻击蠕虫特征码自动提取技术”。提出一种新的抗攻击蠕虫特征码自动提取技术，通过引入抗攻击技术，增强了蠕虫特征码自动提取过程中防范特征码自动提取攻击的抵抗能力，同时提高了特征码提取的准确性。

第六章：总结了本文的研究成果与创新点，提出了下一步的研究方向。

第二章 蠕虫检测相关技术研究

本章围绕蠕虫的原理、关键技术及检测方法展开讨论。重点概括了现有的检测机制及其不足。它们为本文对蠕虫的检测研究工作作了铺垫。

2.1 蠕虫的基本技术

本小节将简略的介绍蠕虫的基本技术，从而为后面介绍蠕虫的检测技术做好铺垫。

2.1.1 蠕虫的工作流程

根据蠕虫工作的一般模式，蠕虫工作流程主要可以概括为：漏洞扫描、攻击目标、感染目标、后续处理、自我复制。如图 2-1（蠕虫的基本工作流程）所示。

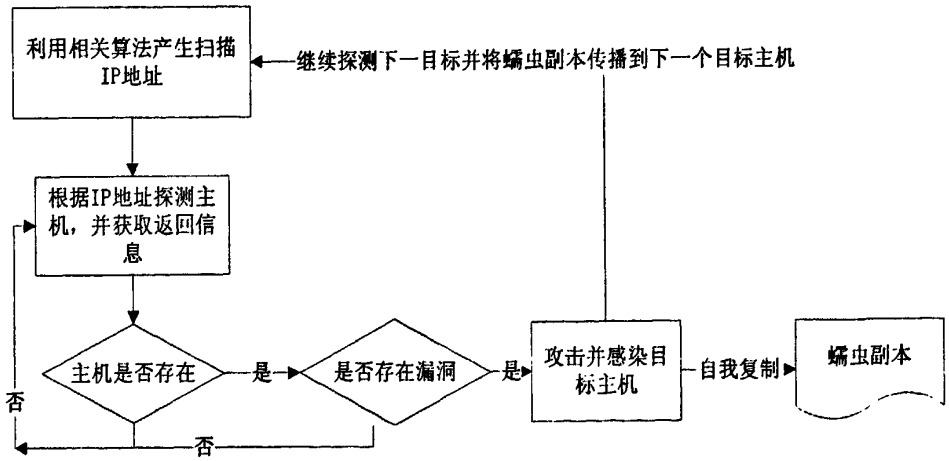

图 2-1 蠕虫的基本工作流程

蠕虫首先利用相关扫描算法形成需要扫描的 IP 地址，然后以上一步获得的 IP 地址作为探测地址，探测这些 IP 地址是否有主机存在；当发现有主机存在时，蠕虫便开始探测该主机是否具有可以利用的漏洞，若无，则继续探测下一主机，若有，则攻击并感染该主机；接下来，蠕虫执行自我复制，同时返回第一步继续扫

描网络，并将自我复制的副本传播到下一个可感染的主机。如此往复，蠕虫便可以在网络中迅速的传播。

2.1.2 蠕虫采用的各种策略

蠕虫要完成从扫描到攻击感染到再传播这个过程需要相应的策略支持，而这些策略正是蠕虫具有的独有特点，既是蠕虫检测的突破口，又是蠕虫检测的难点所在。因此，接下来本文将简述蠕虫具有的各种策略。

2.1.2.1 蠕虫的扫描策略

蠕虫首先是需要一个扫描策略来完成对网络的探测扫描，以此来完成蠕虫工作流程的第一步。

常见的扫描策略有如下几种：

1） 随机扫描

随机扫描是对整个地址空间的 IP 地址进行扫描，这个扫描策略是尽可能的覆盖整个网络，同时忽视那些可能并不存在的地址，因此，该总扫描方式将会产生大量的失败连接数据包。该种扫描具有算法简单，易实现的特点。

2） 本地优先扫描

本地优先扫描是在随机扫描的基础之上改进而来，它的扫描策略引入了本地网段优先扫描的原则，即优先扫描蠕虫所在的网络内的 IP 地址。若蠕虫扫描的目标地址 IP 为 192.168.1.45，则下一个扫描的地址可能是 192.168.1.46。这种扫描方式能够尽可能快的感染蠕虫所在网络的主机，能够短时间对某区域照成巨大的破坏。但该策略却可能对同一台主机进行重复扫描，而引起网络堵塞。

3） 列表扫描

所谓列表扫描，是一种理想的扫描策略，它是在扫描易感染蠕虫的目标之前就预先生成了一份可能感染的目标列表，然后依据列表进行攻击和传播。列表扫描实现难度较大，但由于具有确定性，所以其不会像随机扫描那样产生大量的失败连接，因此，这种蠕虫检测难度较大。

4） 路由扫描

路由扫描是指网络蠕虫根据网络中的路由信息，对 IP 地址空间进行扫描的一种策略。这种扫描策略能够避开那些为分配的地址空间，帮助蠕虫更快、更有效的进行传播。

5） DNS 扫描

所谓DNS 扫描则是指网络蠕虫通过从DNS 服务器中获取 IP 地址来建立攻击地址列表。这种扫描策略的有点在于，其所获取的地址列表具有较强的针对性，并且不容易造成失败连接，即具有较强的可用性。但他也有一些不足：难以得到有 DNS 记录的地址完整列表；蠕虫代码需要携带非常大的地址库，传播速度慢：目标地址列表中地址数受公共域名主机的限制。

6） 协同扫描

协同扫描是蠕虫相互协同，共同完成扫描任务的一种策略，蠕虫之间通过相互通信，确定各自的扫描地址列表，相互互补地址空间，提高了扫描效率。但这种策略容易因某一个主机死机或者蠕虫被查杀造成扫描盲点。

7） 慢扫描

这种扫描通常是较常规扫描慢的一种扫描方法，另外还有通过用户触发而不是主动寻找主机的被动式扫描，这种扫描特点是不会在短时间内出现大量的数据包，因此较难发现。

8） P2P 蠕虫扫描策略

P2P 蠕虫还只是一个停留在理论上的蠕虫病毒，它的扫描策略和常规的蠕虫有所区别，它主要通过感染 P2P 软件从而获得 P2P 软件的对等机列表，从而直接获得主机的 IP 地址。该方法是一种理想的最准确的扫描策略。

以上几点就是蠕虫在扫描阶段所采用的一些策略，研究人员通过掌握并分析这些策略，形成了一些蠕虫检测技术，在后续的篇幅中，本文将进一步介绍。

2.1.2.2 蠕虫的自我防护策略

蠕虫在感染主机以后，为了增强自身的安全性，避免被检测，因此蠕虫通过一系列自我防护策略来达到目的。

1） 变形策略

为了躲避误用检测，即特征码检测技术，蠕虫采取了变形技术，使得每一个蠕虫副本具有不同的字节序列，但功能却保持不变。我们可以把采用了变形技术的蠕虫称为多态蠕虫。

多态蠕虫采用的变形方式有：

a) 加密变形：通过使用一个加密算法，对蠕虫的主体部分进行加密，同时生成相应的解密指令。由于每次使用的密钥不同，所以得到的蠕虫副本形态也不相同，从而达到了变形的目的。

b) 将 NOP 部分进行变形[24]：一般来说，比较简单的办法是插入与 NOP 等价的一字节指令，也可以利用多字节指令实现变形。

c) 指令替换：即使用不同的指令实现相同的功能，也叫指令的等价替换。由此来实现蠕虫的变形。

2） 行为保护策略

行为保护策略，是通过掩盖蠕虫在寄宿主机的过程中，或者在感染，传播的过程中表现出来的异常行为，从而达到逃避检测的目的。目前较为常见的行为保护策略有：

a) 进程隐藏：如果不采用隐藏的策略，蠕虫在后台驻留时很容易被有经验的用户，或者进程检测软件发现。从而暴露蠕虫主体，因此目前的多数蠕虫病毒都采用了进程隐藏策略。为了达到隐藏进程的目的，可以通过 HOOK 查看进程信息的函数，或者修改进程链表来实现。

b) 通信隐藏：蠕虫的检测也可以通过检测异常的网络通信活动来实现，因此采用通信隐藏策略将是蠕虫绕过检测软件的有效手段。

c) 注册表隐藏：蠕虫在攻击和感染的过程中，往往会修改注册表，这个过程容易被检测软件或者用户所察觉，因此通过注册表隐藏策略，实现对注册表修改或者访问操作的透明化，达到逃避检测的目的。

d) 文件隐藏：蠕虫在感染主机后，其本体文件必须进行隐藏操作，否则很难躲避检测软件和用户的检测。文件隐藏可以分为两种，一种是相对隐藏，即并不真正的隐藏文件，而是通过修改文件名使其与正常文件混在一起，达到蒙蔽的效果。第二种则是绝对隐藏，这种隐藏是通过 HOOK 相关 API 函数，达到文件隐藏，即用户不能通过普通的访问方式查看到文件信息。

2.1.2.2 蠕虫的反特征码自动提取策略

这里，对该策略只做一个简要的介绍，在蠕虫特征码自动提取技术研究一章中，本文将做详细的叙述。

由于手动提取蠕虫特征码的严重滞后性，催生了蠕虫特征码的自动提取技术，但是由于蠕虫特征码自动提取技术还在研究之中，还不够成熟，因此针对蠕虫特征码的自动提取技术的攻击方式也开始出现。

例如，通过在蠕虫中包含大量的正常网络数据，这样特征码的自动提取将受到干扰，而将正常的数据段作为特征码，从而导致了大量的误报。这种反特征码自动提取的策略，属于一种高级反检测策略，实现难度较大。

蠕虫所采取的各种策略，也是蠕虫具有的各种特性，针对这些特性，出现了各种蠕虫检测技术。在下一节中，本文将详细介绍现有的叫成熟的蠕虫检测技术，并进行详细的分析。

2.2 蠕虫的检测技术

在计算机网络蠕虫问世之后，对蠕虫的检测技术研究也随即开始，至今已经形成了一些较为成熟的蠕虫检测技术和检测思路。接下来本文将分门别类的对现有的蠕虫检测技术进行分析。

2.2.1 误用检测

按照入侵检测的分类法[25]，可以将蠕虫的检测技术分为误用检测和异常检测。目前两类检测方法都较为常用，但各有区别。误用检测技术通常是采用模式匹配的思想，也即我们常用的特征码匹配技术。该方法一般只能检测已知蠕虫，而不能检测未知蠕虫，因此，它需要在蠕虫爆发后才能通过人工或机器来获取相关的蠕虫特征码，再利用这个特征码进行检测。这种检测技术具有一定的滞后性，但它也具有明显的优势，误用检测技术能够准确的、低误报率的、快速的检测出是否有蠕虫病毒存在。误用检测主要有以下几种检测技术：

2.2.1.1 常规特征码检测技术

所谓常规特征码技术就是将蠕虫在网络传播中数据包所含的特殊字符或字符串提取出来，这些特殊字符或字符串是区别于其他蠕虫和正常网络流量的。一般来说，这些特殊的字符或字符串主要是蠕虫的协议框架，或者核心部分。否则一旦蠕虫进行了变形操作，者很有可能该特征码只能检测出其中一个蠕虫，而不能检测出该种蠕虫。所以提取操作一般又人工完成，以保证它的准确性。只要提取出了蠕虫的特征码，一旦在网络中发现了含有这一特殊的字符或字符串的数据包，就可以检测出蠕虫。另外，也可以按照病毒的特征码提取方式对蠕虫进行特征码提取。提取时应当区分该蠕虫是否具有感染性，同时还要对蠕虫是否加壳或者是否采用了入口点模糊技术。提取过程主要如图 2-2（病毒特征码提取方式）所示。当提取出蠕虫主体的特征码后，检测软件可以扫描主机的各个文件，当找到含有该特征码的文件时，即可发出蠕虫报警，并作出进一步处理。这一种方式就是本机蠕虫误用检测办法。

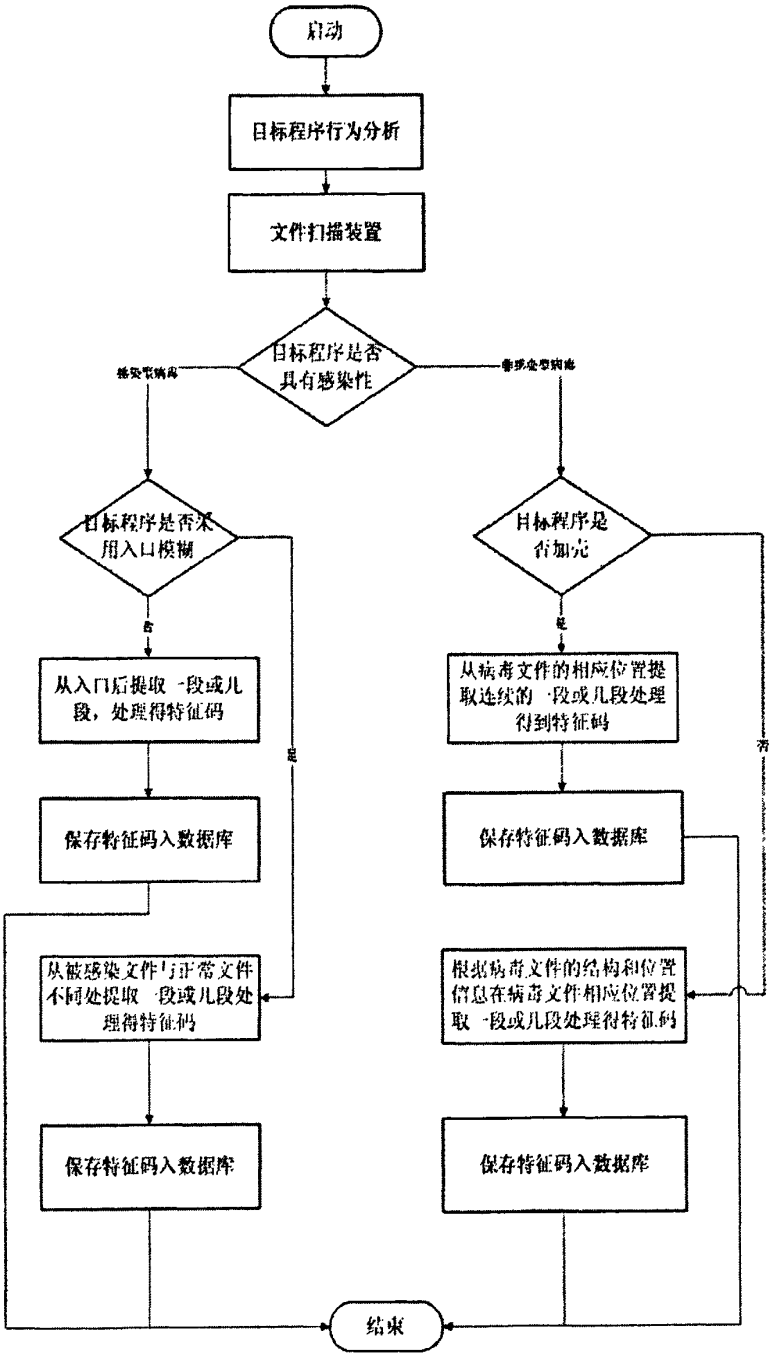

图 2-2 病毒特征码提取过程

2.2.1.2 基于漏洞的特征码检测技术

漏洞特征码[26][27]通常是指对漏洞的描述，一般是一些字节序列，或者是一种数据包格式，它能够准确的描述漏洞的特点。对于利用漏洞进行攻击的蠕虫来说，漏洞特征码技术能够起到很好的遏制作用，有别于传统的蠕虫特征码误用检测方

式的是，漏洞特征码技术可以有效的针对蠕虫的变形技术，只要该蠕虫利用了相应的漏洞，不管它如何变形，都将被检测到。因此，它不仅能检测到已知的利用相关漏洞的蠕虫，还能检测出未知的利用同样漏洞的蠕虫。

2.2.2 异常检测

在这一节里，本文将详细的介绍现有的具有代表性的蠕虫异常检测技术的基本原理和检测方式，并分析其优点和缺点。为本为提出的基于异常用户行为的蠕虫检测技术做铺垫。

2.2.2.1 基于网络流量特征的蠕虫检测

在目前，发展得最为成熟，应用得最广的异常检测方法便是基于网络流量特征的蠕虫检测。这种检测方式是建立在大多数蠕虫共有的一些特征之上，表 2-1（常规蠕虫的共有特征）详细的列举出了这些共同点。

表 2-1 常规蠕虫的共有特征

特征	特征描述
通信量	常规蠕虫爆发初期会产生大量的扫描通信数据，扫描通信量占用了大量的网络带宽，导致通信量出现突增的情况
感染规模	蠕虫感染规模将随蠕虫的扩散成指数增长的趋势
扫描方式	一般蠕虫都具有快速的随机扫描特征，它能保证蠕虫在尽可能短的时间内感染尽可能多的主机

这类蠕虫的典型代表为 CodeRed 蠕虫，通过图 2-3（CodeRed 蠕虫的实际传播曲线）可以看出该类蠕虫的基本特征。根据这一些特征出现了一系列的蠕虫检测方法来识别蠕虫。下面就将介绍和分析一些具有代表性的基于流量特征的蠕虫检测方法。

A） 简单流量特征统计法

简单流量特征统计法主要是直接将连接相关指标，包括连接数量、首次连接失败数量、连接失败总数、ICMP 消息数、连接端口等，进行简单的统计计数。当上述统计数量超过一个阈值时则发出报警。这种简单流量特征统计法能够应对常规的蠕虫，能够起到一定的作用。但其算法简单，对阈值的选择过于依靠经验，不能很好的解决在复杂网络环境下的蠕虫检测。

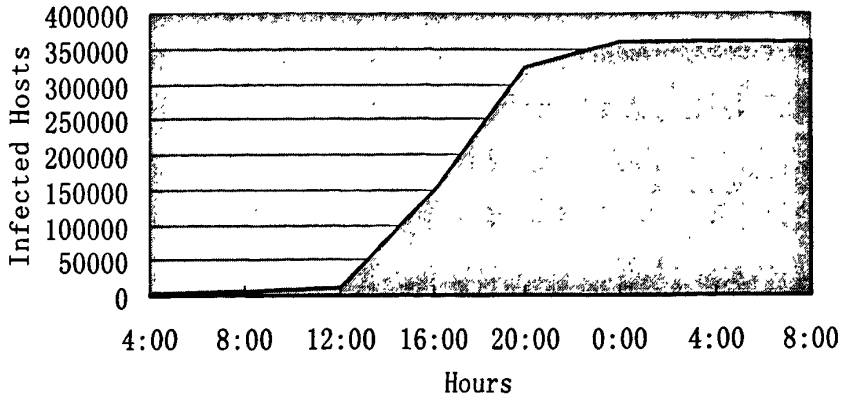

图 2-3 CodeRed 蠕虫的实际传播曲线

B） 基于失败连接的蠕虫检测方法

1. 反向假设检验（RSHT）法[28][29]

反向假设检验是一种具有代表性的基于首次失败连接数的检测方法，它利用蠕虫是一种通过网络进行传播，并且不需要人为干预的一种恶意软件。蠕虫在进行传播之前通常要进行扫描，确定哪些主机有漏洞，扫描的 IP 地址的选择通常具有随机性，扫描的方式可以通过 TCP 协议和 UDP 协议。目前一般蠕虫采用的扫描方法，使得只有少部分的主机会给与应答。

直观的讲，利用假设检验算法，我们设 H1 为出现蠕虫的假设，H0 为正常。则当出现首次连接失败的情况，折线就会向接受假设 H1 的方向逼近，如果首次连接成功，折线就会向接受假设 H0 的方向逼近。

初一看，上述的假设测试方法就已经可以用于蠕虫检测了，实际上假设测试还不能满足要求。举个例子，如果开始的时候主机没有感染蠕虫，那么假设检验的结果很快就会小于下限，从而判断主机没有感染蠕虫，而不再进行检测，如果此后，主机被感染，假设检验将不再起作用。一个简单的改进是，在假设检验结果小于下限时，从新开始假设检验，这可能带来新的问题。一个很好的解决方法是引入反向假设检验。

反向假设检验的算法和假设检验是一样的，不过推倒方向正好相反，假设检验是按照时间先后顺序往后计算，而反向假设检验则是往前计算。在理论上可以证明，反向假设检验用于蠕虫检测的效果更好，是因为前者可以去掉主机未感染蠕虫之前的首次连接的影响。

反向假设检验很好的利用了蠕虫扫描过程中的产生的首次失败连接数量统计信息，能够快速的检测出快速扫描型蠕虫，并且具有较低的误报率。但这种方法也有一定的缺陷，即不能够很好的解决慢速扫描型蠕虫，另外当用户使用 P2P 软件时，该检测方式容易造成误报。

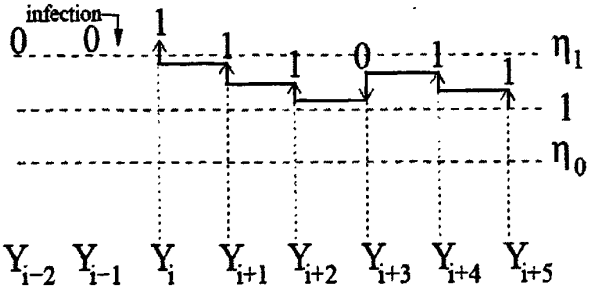

图 2-4 利用反向假设检验检测蠕虫

2. 基于失败连接流量偏离度的检测法(FCFD)[30]

基于失败连接流量偏离度的检测法是另一种依赖于失败连接统计数据的一种检测方法。它区别于反向假设检验法的是，它主要是通过建立失败连接集，并计算失败连接的偏离度，然后获得时间序列，最后利用小波奇异分析出异常突变，并发现蠕虫的一种方法。图 2-5（失败连接数量）显示了一个 5 分钟的流量统计，中间 CodeRed 开始爆发。

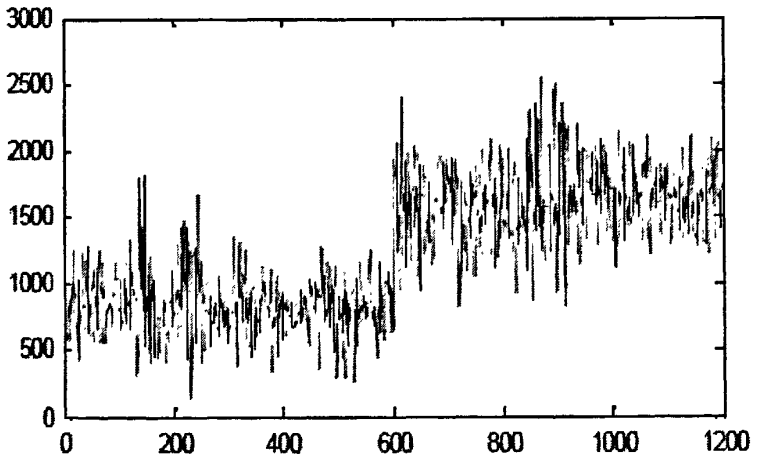

图 2-5 失败连接数量

由图可以看出，在蠕虫攻击发生之初，网络流量几乎没有任何变化，但是在蠕虫爆发时，失败连接流量分布明显发生了突变。

这一种检测方法具有以下几个方面的优点：1、检测速度快，能够在常规蠕虫爆发后立即检测出网络异常；2、能够准确的定位蠕虫攻击的主机；3、想对于简

单的流量特征统计法有了明显的进步。

但这一种检测法也有一定的缺陷:1、同反向假设检验法一样，该方法不能对慢速扫描的蠕虫进行很好的检测。2、不容易区分蠕虫和 P2P 软件，容易出现误报。

2.2.2.2 基于包内容的检测技术

基于包内容的检测技术[31][32][33][34]是一种对正常网络流量的字符出现频率进行统计，得到正常网络流量的频率统计图。检测的时候，对网络中的数据流进行字符频率统计，再与正常的统计图进行比对，当数据流的频率信息产生的异常足够大时，则检测出蠕虫攻击。此外，还可以通过比较数据包中字符频率与正常的统计图比较来判断是否为出现异常。这两种方法都是对数据包的单个字节频率进行统计，可以通过填充字节绕过该方法。一种改进的办法是对数据包的内容进行n-gram，然后用机器学习的方法获得恶意代码的一系列特征，再将网络数据包中的n-gram与之进行匹配，从而判断是否为异常数据包。

2.2.2.3 基于可执行代码的蠕虫检测技术

一般来说，蠕虫主体会含有可以被执行的代码，因此，如果在网络数据包中检测出可执行代码，就可以为蠕虫的提前预警打下基础。同时，也可以利用该技术对网络流量进行分流后，在利用其他技术进行协助分析，从而检测出蠕虫的存在。下面将介绍对于基于可执行代码的检测技术的代表技术。

1. 基于程序光谱分析的可执行代码蠕虫检测技术

I. Kim 提出了一种在网络数据包中检测可执行代码的方法 [35] 。该方法将指令分为12种类型，每种类型用一种不同的颜色表示，如图2-6（不同类型文件的程序光谱图）。通过统计分析可以看出不同类型的文件将会有不同的色谱。尤其，可执行文件的色谱与其他类型将有较大的区别。因而，文献中引用离散马尔科夫链，即将来的指令仅依赖于当前的指令，而与之前的指令无关，并据此获得一个指令转移概率矩阵。对任一可执行文件，根据该指令转移概率矩阵，计算出其指令光谱期望值，如果该值超过一定阈值，则认为该可执行文件为恶意程序。

2. 基于反汇编的可执行代码检测技术

R. Chinchani [36]中提出首先对数据包进行反汇编，获得汇编后的控制流图，并对控制流图进行分析。如果控制流图中含有系统调用指令，中断指令，解密圈等，则认为该数据包为攻击数据包。

T. Toth etc. [37]也提出了一种利用网络数据包的反汇编后连续有效指令的数量

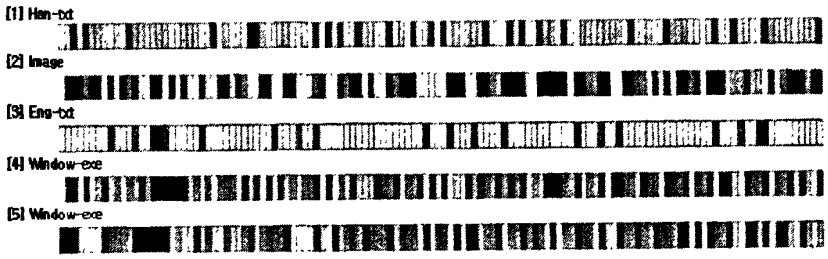

图 2-6 不同类型文件的程序光谱图

来判断数据包是否可疑的方法。该方法的主要依据是正常数据包的连续有效指令的数量很少，而网络攻击数据包的连续有效指令数量很大。尤其对于在攻击数据包中加入了较多的NOP SLED的攻击数据包而言，该方法是比较有效的。但P. Akritidis[38]在其文献中指出，虽然文献[37]中提出的方法可以检测出一定的攻击数据包，但是经过特定构造的数据包却有可能绕过该种检测方式。因而提出了一种改进的检测的检测办法，即：检测从任意位置开始执行后，最终都会达到恶意代码的位置。

3. 基于仿真的可执行代码检测技术

M. Polychronakis [39] 提出了用CPU虚拟机来模拟仿真执行数据包反汇编后的指令的执行来检测异常数据包。该方法主要依据是：恶意数据包中会有很多可执行代码，即使是采用了多态技术、变形技术后的数据包，其解密部分也是可执行的，并且整个解密过程将会不断地从内存的某部分区域读出数据进行解密，然后再将这些数据写回内存中。因而，可以通过判断在虚拟CPU中执行时的读操作的数量以及是否有一个循环的解密体来判断是否为恶意数据包。但对于大规模的网络而言，由于该方法会对所有数据包仿真，可能会存在一个处理瓶颈问题。因而，Qinghua Zhang[40]提出可以通过对静态反汇编后得到的控制流图进行分析，找出解

密体，然后再对可疑数据包进行仿真。该方法可以在一定程序上提高处理效率，缓解之前的处理方式中的瓶颈问题。另外，M. Polychronakis[41]还提出了采用写操作来判断是否为恶意数据包。该方法主要是判断是否在虚拟CPU一次执行中是否向多个不同的内存位置写入了数据。文献[42]中描述了软件仿真是蠕虫传播特性研究的重要手段之一,也日益引起研究人员的兴趣。这种方法可以充分考虑网络带宽、网络结构、正常网络流量等重要因素对蠕虫传播行为的影响，并具有比较好的扩展性和灵活性，能够分析多种具有不同特征的蠕虫。鉴于软件仿真方法的诸多优点，它被认为是蠕虫传播特性研究方面最有效的方法，引起了国外研究机构的兴趣和重视。

2.3 小结

本章介绍了蠕虫相关工作流程和采用的关键策略，重点介绍了现阶段蠕虫的检测方法，总结了目前存在的蠕虫检测方法存在的一些问题。普遍的，目前的蠕虫检测技术还是着重利用对蠕虫失败连接进行分析而得出结论。如：对其失败连接进行统计，试图发现失败连接的瞬间增多或者突变，但这种目前面临着严重的挑战。例如，P2P 软件的出现和大量应用，同样会出现大量的失败连接从而造成误报。这些问题也是本文主要研究的内容，本文将提出新的解决办法来解决蠕虫检测方式单一的状况，同时解决蠕虫检测侧重网络检测而忽视主机检测的问题。本章节对现有蠕虫检测技术的介绍和总结为本文第四章的基于异常用户行为的蠕虫检测技术作了铺垫。

第三章 蠕虫特征码自动提取相关技术研究

本章讨论主要介绍了蠕虫特征码自动提取技术（后文可简称为 ASG）的基本概念，详细分析了现有的蠕虫特征码自动提取技术的优缺点，同时还详细的介绍了目前针对蠕虫特征码自动提取技术的攻击技术。为后文介绍本文提出的抗攻击蠕虫特征码自动提取技术做好铺垫。

3.1 蠕虫特征码自动提取技术的基本概念

3.1.1 蠕虫特征码自动提取的基本背景和基本概念

近几年蠕虫爆发出现了指数增长趋势，蠕虫对计算机以及网络的危害越来越严重。为此，网络领域的研究人员提出了入侵检测系统的概念（IDSes）来抵御蠕虫的攻击[60]，这种技术通过比对负载或者是比对正侧表达式。但是传统的蠕虫特征码提取方式主要依靠人工提取，但人工提取蠕虫特征码却不能满足蠕虫高速的增长趋势，更不能满足现在蠕虫的快速传播、变形的特点。如何找到一种能够自动的提取蠕虫特征码，缩短蠕虫爆发到遏制的时间，减小蠕虫造成的各种损失，是目前蠕虫特征码提取技术面临的难题。在这个背景之下，蠕虫特征码自动提取技术开始出现，并且受到了安全领域相关人员的高度重视。

所谓的蠕虫特征码自动提取技术就是通过一定的算法自动的从蠕虫中提取出有别于正常数据流和其他蠕虫数据流的字符串，并将其作为特征码用于误用检测的一种技术。

3.1.2 蠕虫特征码自动提取技术的基本特点

1）有效性：

有效性是指所提取出的特征码必须是有效的特征码，即能够用该特征码通过误用检测技术检测出该蠕虫。如果所提取出的特征码不能检测出该蠕虫，那么该特征码即是无效的。

2）正确性：

正确性是指所提取的特征码必须是能正确的检测出对应的蠕虫，而不会将正常的数据包误报成蠕虫，否则将导致误报。让蠕虫特征码自动提取技术失去正确性是反蠕虫特征码自动提取技术的一种重要手段。

3）高效性：

高效性是评价蠕虫特征码自动提取技术的重要指标，蠕虫特征码自动提取技术存在的意义就是可以快速的，准确的提取蠕虫特征码。

3.1.3 基于 ASG 技术的系统正常运行的基本前提

要实现 ASG 技术，就要构造基于 ASG 技术的蠕虫特征码自动提取系统，蠕虫特征码自动提取系统需要一些基本的前提条件，才能保证特征码自动提取需要满足的有效性、正确性、高效性。蠕虫特征码自动提取系统的最基本前提则是 ASG 系统需要运行在一个无毒的和不会受到蠕虫或病毒攻击的环境中。另外，还需要一些基本的前提它们分别是：

前提一：蠕虫在网络中传播的实体数据包总是含有一定的有别于正常数据流的字符串。这个前提条件说明了，在蠕虫实体中一定存在一定的有别于正常数据流的字符串，而这个字符串就是我们需要提取出的特征码。

前提二：从所有网络数据流中筛选出来的用来进行特征码自动提取的数据流被认为是可疑数据流。这个前提条件主要是为了保证被提取出的特征码满足正确性。即，从被筛选出来的可疑数据流中提取出的特征码一定是一个和相关蠕虫对应的特征码，而不会是正常数据的特征码。

前提三：攻击者不能控制蠕虫样本是否经过 ASG 系统。这个前提是一个必然成立的前提，因为如果蠕虫能够被攻击者所控制，并选择是否经过 ASG 系统，那么 ASG 系统将失效。因此，提出这个前提是理所当然的。

前提四：攻击者不能控制哪些蠕虫数据流会被 ASG 系统筛选为可疑流。这个前提被提出的原因和上一个前提相似。

前提五：攻击者不能改变正常流中的子串出现频率。如果攻击者能够改变正常流中的子串出现频率，则可以通过增加某子串出现率导致 ASG 系统提取出错误的子串作为特征码，而导致出现错误的特征码。

前提六：攻击者不能控制正常的数据流被 ASG 系统筛选到可疑数据流中。这

个前提主要说明了攻击者不能使得正常的数据流被筛选到可疑数据流中，否则ASG系统将错误的把正常的数据流作为蠕虫数据流并提取出他的特征码。这样被提取出来的码将是一个错误特征码。

前提七：被用来做为参照的被收集来的正常数据流基本能够代表整个网络中的正常数据流。这主要是保证作为参照的数据流的普遍性，否则它将不具有代表性。

以上几个前提是保证蠕虫自动提取技术正常有效的基本前提，也是一个通用前提。

3.1.4 蠕虫特征码自动提取技术的分类

经过近段时期的研究，一些蠕虫特征码自动提取（ASG）技术相继被提出。例如早期的Hoenycomb[44],Autograph[45]以及EarlyBird[46]。这三种技术都认为能够在蠕虫负载中找到一条连续的字符串,并且该字符串能够作为对应蠕虫的特征码。但是这种技术很快就被变形蠕虫给破解，因为变形蠕虫的负载不会出现连续不变的字符串，因此这种提取方式会很快失效。

后来，研究人员开始针对变形蠕虫进行研究，并提出了如Polygraph[47]，Hamsa[48]以及Lisabeth[49]等ASG技术。这三种技术将一些子串的集合（即多字符串）作为一个特征码而不是一段连续的字符串。而这些子串则是在蠕虫负载中出现频率较高而在正常流中几乎不会出现的字符串组成。Polygraph提出了三种基于多字符串的特征。包括：联合性、令牌序列性、 Bayes特性。因此，ASG技术分类，一般是按照蠕虫特征码的构成进行划分的。

下面将详细对蠕虫特征码技术按特征码构成进行分类，并作出详细的解释说明。

1）单一字符串特征码

总的来说，这种技术主要是在可疑流中寻找出一条出现频率最为频繁的字符串作为特征码。它会对网络中收集的可疑流量进行内容负载分割，然后对得到的字符串在可疑流中出现的次数进行统计，并将出现频率最高的字符串放到特征池中，并将包含该字符串的可疑流量从可以流量池中删除，重复这个步骤，就能得到能够覆盖一定比例的可疑流的字符串集合，并把它作为蠕虫的特征码。

2）联合字符串特征码

单一字符串型特征码自动提取技术虽然从一定程度上解决了蠕虫特征码的自动提取问题，但是在面对多态蠕虫时，上述技术则显得无力[57][58][59]。而多态蠕虫是蠕虫发展的一个新的趋势，它通过各种变形技术变更蠕虫负载中的字符串位置，或者利用加密技术进行变形。

因此，在这种前提下就很难找出一个足够长的公共部分作为特征码。但是，多态蠕虫也有自身的弱点，多态蠕虫的变形只能是相对的，对于那些核心的部分，如协议框架，跳转地址等，多态蠕虫是不会进行变形的。因此，就出现了联合字符串型的蠕虫特征码提取技术。所以，联合字符串特征码技术技术则是将可疑流所含的公共子串一起共同构成一条特征码的技术。其中，一条公共子串也可以称为一个令牌或者记号。

3）令牌序列特征码

令牌序列特征码是联合字符串型的一个改进型，它仍然是基于公共子串的。只是不同的是，令牌序列型特征码是由一定顺序的令牌组成。

4）贝叶斯特征码

贝叶斯特征码技术具有更高的抗干扰度，它不苛求精确的匹配，因此它能够在网络流具有噪音的环境下以及多种蠕虫数据流被筛选为可疑流的情况下进行特征提取。

贝叶斯特征码技术的算法较为复杂，首先必须假定在被筛选出的可疑流量中，各个令牌出现的概率是相互独立的，因此可以计算令牌在可疑流和正常流中出现的经验概率，然后利用贝叶斯公式，计算出字符串中出现该令牌这一事件的置信度，也就是该字符串就是蠕虫字符串的概率支持度。

3.2 反蠕虫特征自动提取技术

所谓反蠕虫特征码自动提取技术（Anti-ASG）就是指针对蠕虫特征码自动提取技术的攻击技术。任何事物都有其两面性，蠕虫特征码技术也具有这样的特性，它不是一个完美的技术，因此也具有自动的弱点。它在帮助我们自动的获取蠕虫特征码的同时，还有可能被攻击者利用其自身的弱点反而达到攻击我们的目的。

目前，随着人们对蠕虫特征码技术的深入研究，针对蠕虫特征码自动提取技术的攻击技术[50][51][52][53]也陆续被提了出来。如何解决好这个难题，将是本文后面提出的抗攻击蠕虫特征码自动特征码提取技术所面对的。

在介绍反蠕虫特征码自动提取技术之前，我们有必要先了解多态蠕虫的概念。因为这个概念涉及到反蠕虫特征码技术的基本内容，也是部分反蠕虫特征码自动提取技术的技术基础。

3.2.1 多态蠕虫的基本构成

下面，将参照图3-1（多态蠕虫结构模型）大致介绍一下多态蠕虫的基本构成，以为后作好铺垫。

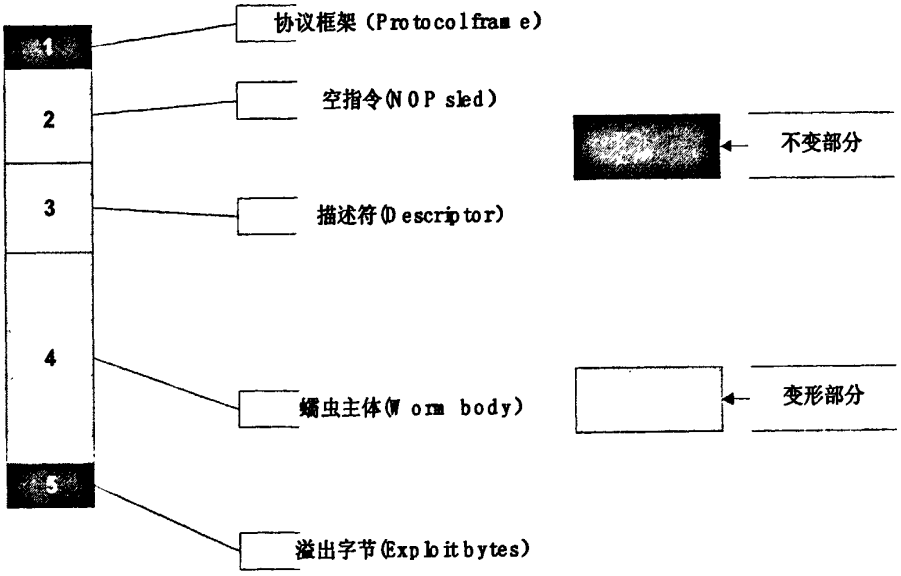

图3-1 多态蠕虫结构模型

1）协议框架（Protocol framework）

协议框架是蠕虫的一个重要组成部分，它是蠕虫攻击漏洞的重要基础。普遍的讲，网络蠕虫的攻击和传播是依靠攻击相应的漏洞从而达到占领目标的目的，这些漏洞可能是一些特殊的程序编码缺陷或者是一定的代码执行顺序。为了达到攻击漏洞的目的，蠕虫必须包含一段和漏洞相关特征同样的代码，即协议框架。重要的是该部分是基本不会因蠕虫进行变形操作而发生改变的，因此它为蠕虫特征码的提取打下了重要的基础。

2）溢出字节（Exploit bytes）

所谓溢出字节就是专门用来利用漏洞的缓冲区溢出将程序本来跳转到正常指令的地址修改为蠕虫恶意代码段的地址上来。它仅仅是一个重定向地址，因此溢出字节不会太长。和协议框架一样，溢出字节也基本不会因为蠕虫进行变形操作而发生改变，它也是蠕虫特征码提取的重要依据之一。

3）描述符（Descriptor）

描述符的作用是为了解释蠕虫的加密部分，并且将加密的控制流转换成明文。但是描述符却不同于以上两个部分，它是可以通过一定手段进行变形操作的。典型的集中技术是[55][56]：

a) 混入一些不会影响描述符执行的一些指令

b) 通过等价的指令进行指令替换，且同样能达到之前的执行结果

c) 对每一个变形版本的描述符使用不同的寄存器

d) 使用不同的加密方式

因此描述符部分是基本不能作为特征码自动提取的参照的，否则将会出现特征码不能应变变形蠕虫的情况。

4）空指令（NOP sled）

这段字节段主要是用来填充那些希望被溢出的堆栈，它是没有执行意义的一段代码，甚至这段代码是可有可无的。但恰恰是这段无意义的代码却成为了变形蠕虫进行变形的主要手段之一。

5）蠕虫主体（Worm body）

蠕虫主体是最好进行变形的部分，它可以通过各种加密手段变形成不同版本的蠕虫主体，使得每个版本的蠕虫主体都可能呈现不一样的数据字段构成。从而达到防止自动提取特征码的目的。

通过以上内容，我们知道蠕虫特征码自动提取的重点和关键点是在如何从蠕虫的两个不变部分提取出有用的特征字符串，并将它们组合成蠕虫的特征码。

3.2.2 几种主要的反蠕虫特征码自动提取技术（Anti-ASG）

3.2.2.1 Red herring attack [51]

Red herring attack 的主要思想是利用在蠕虫中故意混入干扰的字符串，使得蠕虫特征码提取被这些故意混入的干扰字符串所影响，从而将这些字符串作为特征码提取出来，造成在后续的误用检测中出现大量的误报或漏报情况。图 3-2（Red herring attack 攻击示意图）

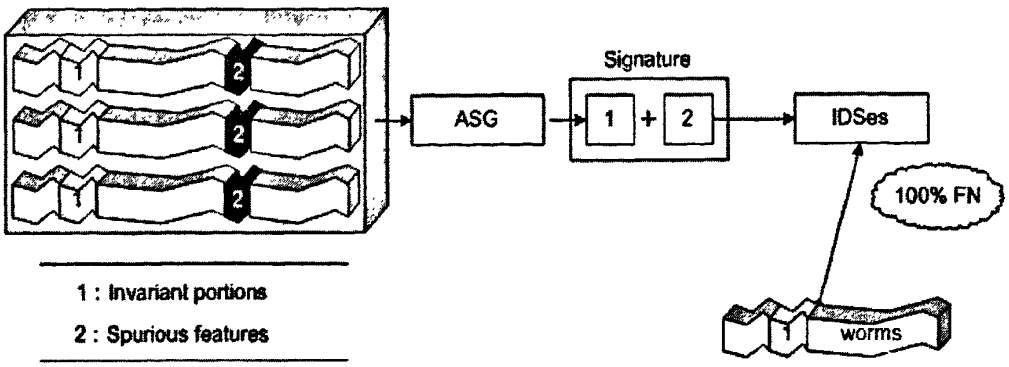

图 3-2 Red herring attack 攻击示意图

如图所示，1：为蠕虫本身的特征字符，2：为攻击者伪造的特征码，在攻击者进行正式攻击前，攻击者将 2 部分添加到蠕虫中，并且第 2 部分可能会比 1 部分在蠕虫中有更高的占有率，这样当蠕虫流进过 ASG 系统时，ASG 系统会将 1 部分和 2 部分，甚至是只有第 2 部分提取出来作为该蠕虫的特征码。特征码被提取出来被放入 IDS 检测系统之后，攻击者在将没有含有 2 部分的蠕虫放出，这样 IDS 就无法检测到蠕虫，造成 100%的漏报，如果被添加到蠕虫的 2 部分的是在正常流中经常出现的字符串的话，就会造成 100%的误报，而导致正常网络流不能通过 IDS。因此，在本文提出的蠕虫特征码自动提取技术中，我们通过引入筛选特征码的同时还要判断该特征码的误报率和漏报率来应对这种攻击，进过实验，我们的方法是完全有效的。

3.2.2.2 Suspicious pool posisoning attack [52]

这是一种利用基于贝叶斯算法进行特征码提取的漏洞进行的针对可疑数据池的污染攻击。攻击开始时，攻击者利用向假蠕虫中有意的添加一些特殊构造的数据串，从而使在这些网络数据流被分流到可疑池后，这些被有意添加的数据串将导致真正的蠕虫特征串的贝叶斯分数明显减少。因为真正的蠕虫特征串相对于那些特殊构造的数据串来说是绝对稀少的。而后，ASG 将提取出错误的特征码。而本文所设计的蠕虫特征自动提取技术引入了一个阈值，只有那些包含特征码串的

流超过整个可疑池的流一定数值时，才能被提取。

3.2.2.3 Innocuous pool posisoning attack [52]

常见的正常池污染攻击方式是通过增加蠕虫不变串在正常流中出现的几率，或者其他办法，尝试将这些蠕虫不变串添加到正常池中，从而导致贝叶斯算法失效，因为如果将蠕虫的不变串作为特征码提取出来的话，它在正常池中的误报率将非常的高，因此 ASG 会认为这些特征码是不可取的。从而让攻击者达到误导 ASG 提取错误的串作为特征码的目的。

在本文提出的蠕虫特征码自动提取技术中，我们将按一定的时间刷新正常池，使得该种攻击失效。具体的办法，将在后文进行详细的讨论，并给出相关的实验结果。

3.2.2.4 Allergy attack [53]

所谓的 Allergy attack 实质是一种特征码 DoS 攻击，它的主要目的是通过一定的手段让特征码自动提取系统提取出正常流中才含有的字符串来作为特征码，从而导致 IDS 对正常流进行阻断或者采取相关的措施。这种攻击的攻击串必须具备能够在 ASG 系统分流时被分流道可疑流中，从而才能被进行特征码提取；但是在被提取之后，得到的特征码却是在正常流中出现的字符串。其导致的结果便是上面提到的，正常流将被阻断。这一攻击对于目前的 ASG 系统来说是最为致命的，因为目前的 ASG 技术都建立在“被提取的特征码是在正常流中基本不出现而在可疑流中或者蠕虫中才出现的字符串”这个前提之上。因此，一旦能伪造出这样一个错误的字符串，则可以导致目前的 ASG 系统失效。因此本文提出了采用区域分布式的特征码自动提取技术来解决这个问题。这一理论是建立在“虽然在某一个网络数据流中一些字符串很少出现，但在另一个网络流中也许他会经常出现”这个前提上，因此通过区域分布式的方式，让蠕虫特征码自动提取不会因一个点被污染而提取出错误的特征码。

3.3 小结

本章主要总结了蠕虫特征自动提取的基本原理、特征码技术分类、已经目前的针对特征码自动提取技术的攻击技术的概述，让读者能够大致掌握目前特征码自动提取技术的基本情况，同时对蠕虫特征码自动提取技术的相关研究工作也为后续本为提出的抗攻击蠕虫特征码自动提取技术打下了良好的基础。

第四章 基于异常用户行为的蠕虫检测技术

本章研究并实现了一种新的蠕虫检测技术：基于异常用户行为的蠕虫检测技术。它打破了以往以分析网络流量特征异常为主的蠕虫异常检测手段，而是通过分析用户产生的网络流量特征来发现异常流量特征与用户正常网络流量特征的偏离度来发现蠕虫的一种检测技术。该算法不仅可以部署在每一个用户节点的主机上并结合一些主机检测手段对蠕虫进行检测，还可以直接部署在网络中，对本地网络蠕虫进行检测。同时，当将该检测技术部署到网络中时，可以作为后面本文将介绍的抗攻击蠕虫特征码自动提取技术的可疑流量分流器。

4.1 设计思想与目标

4.1.1 设计思想

我们知道对于每一个独立的用户个体来说，他在使用计算机网络时，总会存在一定的固定模式，这个模式我们称为用户行为，而且这个模式在一定时期内不会突然改变，整个用户行为的改变模式是一个平滑的改变过程，而不是一种极端的跳跃改变过程。因此我们可以通过分析用户的网络使用行为，来进行蠕虫的早期检测。本文通过对蠕虫的行为特征进行研究发现，蠕虫的行为特征能够在短时间内改变用户网络行为特征参数，显示出不同的网络行为特征。因此，基于这个前提条件，我们同学对用户的历史网络行为特征进行学习，建立起用户网络行为特征集，从而通过对比用户网络行为特征集，计算出用户网络行为偏离度，从而发现出异常的用户网络行为特征，及时检测出蠕虫。

4.1.2 设计目标

通过前序的分析，可以看出，蠕虫的检测需要满足准确高效，低误报率和低漏报率。本文提出的基于异常用户行为的蠕虫检测技术将努力实现以下目标：

1：低误报率和漏报率。误报率和漏报率是评价一个异常检测技术好坏的重要指标，它不仅能反映该检测技术的准确性，还能反映出该检测技术的容错性。误报率过高将会影响正常的流量通行，严重的可能还会因此失去正常的关键数据造成难以估量的损失。漏报率过高，将造成更为严重的后果，它会让蠕虫在网络中

迅速传播，给社会和人们的日常生活带来巨大的不变和造成巨大的损失。因此，如何能够提高蠕虫检测技术的精确性，降低误报率和漏报率，是蠕虫检测技术不断完善的根本动力。因此，本文将通过转换检测思路，另辟蹊径，希望能开辟出另一条检测方式。

2：较强的自适应能力：针对本技术具有机器学习的特点，我们必须让整个技术具有将强的自适应能力。即，能够较快的适应用户网络行为的渐进改变，能够不断修正产考参数，让用户网络特征模型能够和用户当前的网络行为相匹配。从而减少因用户行为的不断改变而造成的参考数据偏离，进而防止系统误报率和漏报率因时间的推移不断升高最终导致系统失效的情况发生。

3：能够对已知和未知蠕虫进行检测：本技术将误用检测和异常检测结合起来形成一个完整的蠕虫检测体系，以误用检测为辅，基于异常用户行为的蠕虫异常检测为主，对蠕虫实施有效的检测。

4：便于实施和扩展：本技术能够很好的实例化，形成基于异常用户行为的蠕虫检测系统，同时，该技术要具备一定的扩展能力。即能够与后续的蠕虫特征码自动提取系统结合，成为一个有效的分流系统。另外，本技术还能与一些基于主机的异常检测结合起来，形成联动体制。这一部分的内容，将在后续章节中详细讲述。

4.2 技术框架

本章主要叙述了本技术的基本技术原理，技术设计框架，和基本设计流程。下面我们将首先介绍本技术的基本技术原理。

4.2.1 基本技术原理

在本小节中，我们将分别阐述，本技术的基本支持前提、技术的基本原理、以及技术的基本框架。

4.2.1.1 基本支持前提

在开始深入讨论本技术的算法和详细设计时，我们需要引入相应的支持前提来作为对本技术的理论支持。其中，本技术依赖的最基本前提是：蠕虫的网络行为特征是异于正常的网络行为特征的。接下来，本技术还将依赖于以下几个基本前提：

前提一：蠕虫的网络行为具有一致性，不会因网络环境的不同而发生适应性

改变。这个前提说明了，蠕虫从其扫描阶段到攻击阶段再到再传播阶段所表现出来的网络特性是较为固定的，不会因为转变了网络环境，而出现与网络环境较为一致的表现。如，不会因为用户经常和某几个 IP 地址发生网络联系，而蠕虫也会调整策略，只和这几个 IP 地址发生网络联系。

前提二:用户的网络行为特征在连续的时间内呈现线性变化的稳定变化特征，而不会出现跳跃性的变化。这个前提主要说明了，用户的网络活动具有规律性，并可以对这短时间的访问规律进行学习建模。如果用户的网络活动特征出现跳跃性变化特征，那么本技术将可能失效。

前提三：蠕虫在攻击网络主机时一定会表现出相关的网络特征，而不是没有网络特征。这一点是一个必然成立的前提，如果蠕虫在进行网络攻击时，没有任何网络的特征的话，那么它就失去了蠕虫的基本特点，而退化成主机病毒。

以上一个基本前提和三个主要前提共同作为本技术的理论前提支撑，本技术将在上述三个前提之上进行进一步的探讨和研究。下一节，我们将描述本技术的基本原理。

4.2.1.2 基本技术原理

本技术是在上述几个前提条件的基础上发展出来的以用户行为为参照的蠕虫检测技术。在前序章节中，我们提到了蠕虫的工作流程，特别是其在扫描期间运用各种扫描策略进行扫描时，将会在短时间内产生大量的数据包，并且将随机访问大量的非用户习惯访问的目的地址。因此，由于蠕虫的这一特性大大的增加了节点主机的陌生访问地址数，而这一特性是与用户的习惯网络特性是相异的。在一定时期内产生的大量无关链接导致了异常的用户行为特征的产生，从而表明了蠕虫存在的可能。但如何刻画好这一特性，将是本文研究的重点。

为了更好的说明这一种差别，我们将对用户的网络行为特征进行建模，以便后续进行研究。

4.2.1.3 用户网络行为特征模型（UNB-MODEL）

为了验证用户的网络特性在一定的时间内将逐渐趋于固定，并且只会具有微小的变化，我们对用户的网络行为特征进行了建模。同时，为了避开 P2P 软件的影响，一下模型都不考虑 P2P 软件的情况。至于 P2P 软件干扰下的用户异常行为蠕虫检测，我们将在后续章节中给出相应的解决办法。下面，我们介绍第一个定义：

定义 1：用户活跃度（Degree of user activity-DUA）

若用户 *a* 在过去的时间 *t* 内产生了一系列的访问地址，我们将这一系列的访问地址记录下来并建立历史访问集合 $H_a(t, addr)$，同理我们将用户 a 在目前时间 t 内产生的一系列访问地址记录下来，同样形成访问集合 $Na(t,addr)$,则有用户活跃度（*DUA*）为：

$$DUA = \int_0^t H_a(t,addr) * N_a(t,addr)dt \qquad (4\text{-}1)$$

其中运算符*为：

$$H_{addr} * N_{addr} = \begin{cases} 1 & N_{addr} \in H_{addr} \\ 0 & N_{addr} \notin H_{addr} \end{cases} \qquad (4\text{-}2)$$

DUA 用来描述用户在一段时间内访问目的地的变化，*DUA* 的大小反应了用户在时间 *t* 内的新目的地址产生数量。从另一方面来说，它就代表了用户的网络活跃度。因此，只喜欢浏览固定网页新闻的用户 *DUA* 值将远远小于喜欢在网络中无目的冲浪的用户。但这里的无目的只是短时间内，当时间 *t* 取值足够大时，无目的冲浪用户也将产生一定的规律分布。因此，我们引入下一个描述函数：活跃度分布（distribution of DUA）。

定义 2：活跃度分布（Distribution of DUA-DDUA）

T 时刻活跃度分布是在单位时间窗口Δt 内，T+Δt 与上一时刻的 *DUA* 差值。则 *DDUA* 为:

$$DDUA_T = | DUA_{T+\Delta t} - DUA_T |, \qquad T > 0 \qquad (4\text{-}3)$$

由此可以知道，DDUA 值将随着用户的习惯渐进改变而出现波动。但波动将随着时间的推移逐渐稳定，实验数据证实了这一点。图 4-1（正常用户活跃度分布图） 显示了这一特性。

从图中，我们可以看出，随着时间的推移，活跃度分布波动逐渐减弱，并且趋于稳定，而出现的细小波动，是用户的网络行为渐进改变的反应。但是，如果出现蠕虫攻击，那么此时的活跃度分布将出现异常。图 4-2（异常情况下用户活跃度分布图)充分的说明了这一点。我们可以从图中看到，在两参考线之间，DDUA 曲线出现了突然的跃升，这个跃升就是蠕虫在攻击阶段发起扫描行为而产生的。这一现象与正常用户活动下的 DDUA 曲线区别明显。

这只是单个节点的情况，如果将该模型放大到具有 N 个节点的网络中时，是

否还能出现上述规律呢？我们建立一个 N 节点的用户活跃度 $NDUA$ 函数：

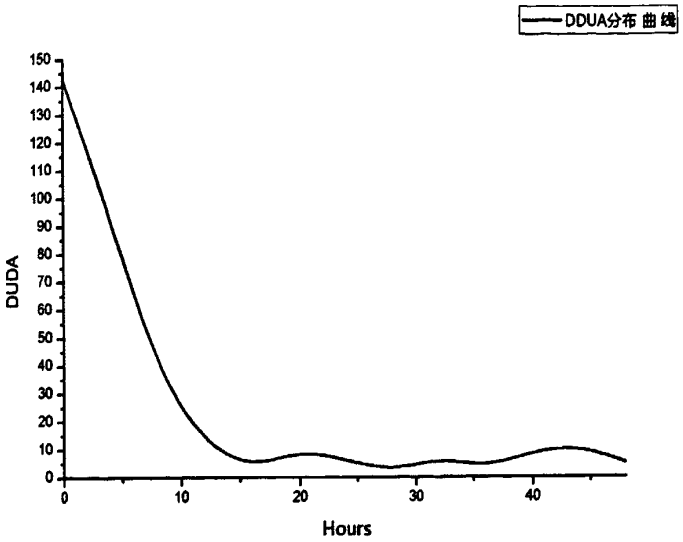

图 4-1 正常用户活跃度分布图

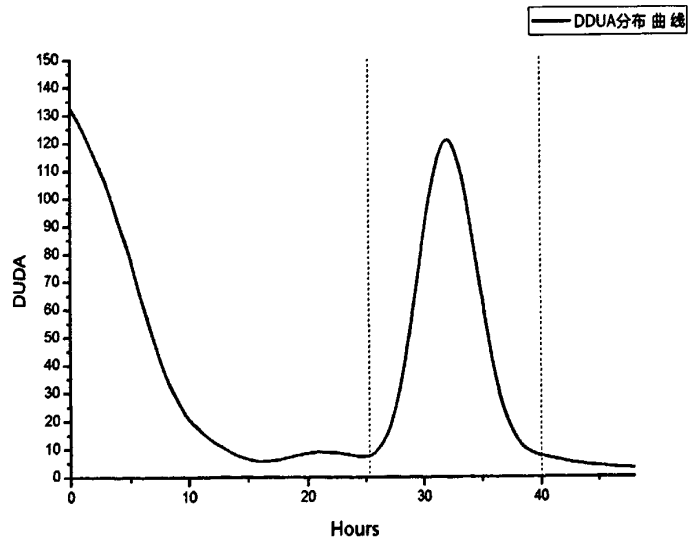

图 4-2 异常情况下用户活跃度分布图

定义 3：N 用户活跃度（NDUA）：

它反映了 N 个用户的网络中，整体网络的活跃度，其函数为：

$$NDUA = \sum_{i=0}^{n} DUA(i) \circ x \tag{4-4}$$

其中运算符 $\circ$ 为：

$$DUA(i) \circ x = \begin{cases} 1 & DUA(i) = x \\ 0 & DUA(i) \neq x \end{cases} \tag{4-5}$$

同理我们可以建立起 N 个用户的活跃度分布（D-NDUA）

定义 4：N 用户活跃度分布（D-NDUA）

T 时刻活跃度分布是在单位时间窗口 Δt 内，T+Δt 与上一时刻的 *NDUA* 差值。则 *D-NDUA* 为:

$$D-NDUA_T = | NDUA_{T+\Delta t} - NDUA_T |, \quad T > 0 \tag{4-6}$$

由此，我们可以知道，在 N 个用户的网络中，同样存在着一定的用户网络行

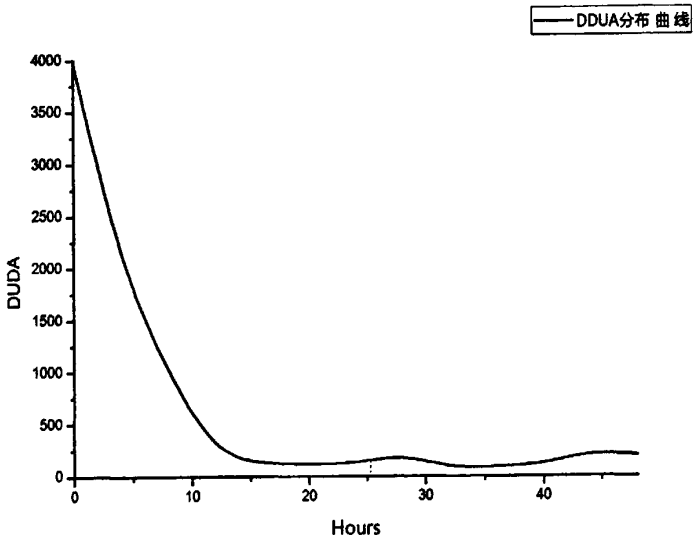

图 4-3　N 个正常用户活跃度分布情况

为规律，此规律同单用户节点的规律类似。图 4-3 （N 个正常用户活跃度分布情况）反映了这一规律。

从图中，我们仍然可以看出，随着时间的推移活跃度分布波动逐渐减弱，并且趋于稳定，而出现的细小波动，是用户的网络行为渐进改变的反应。但是，如果出现蠕虫攻击，那么此时的活跃度分布将出现异常。图 4-4（N 个用户异常情况下用户活跃度分布图）充分的说明了这一点。

由此，我们可以得出，用户的网络行为在一段时间内具有稳定的规律性，并且出现渐进的波动。而该波动与蠕虫爆发时产生的波动相比，其波动程度具有极大的差异。这个差异就是本文的理论基础，也是研究的切入点。接下来，我们将简单介绍整个技术的基本框架。

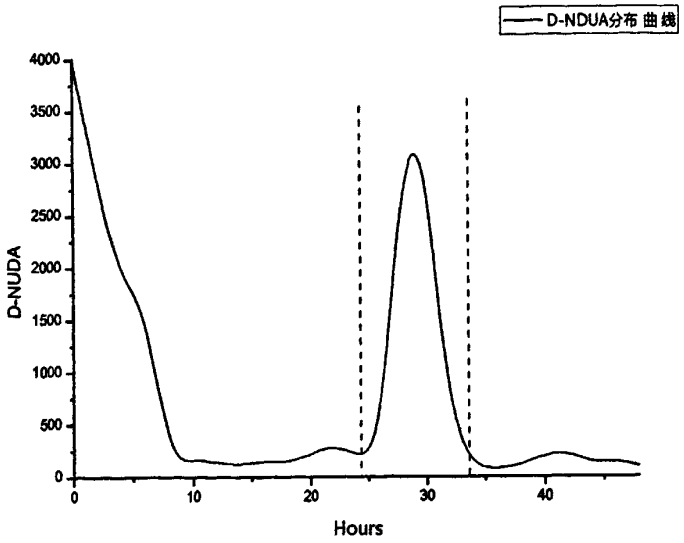

图 4-4 N个用户异常情况下用户活跃度分布图

4.2.2 基本技术框架

通过上一节对正常用户行为和有蠕虫参与的异常用户行为进行分析，我们可以发现利用用户行为之间的差异是可以用来进行蠕虫检测的。但是，前面的模型只是一个验证的简单模型，如何才能更好的刻画这种差别，并且能够将这种检测技术实例化，将是本文后续工作需要研究的。接下来，本文将介绍基于异常用户行为的蠕虫检测技术的基本技术框架。

4.2.2.1 渐进式的用户行为学习

本技术是基于机器学习的异常检测技术，因此它会有一个机器学习阶段，整个学习的基本流程框架如图 4-5 （用户行为学习流程框架）

我们将收集来的网络数据流首先通过误用检测系统，尽量保证在学习阶段所学习的参数的正确性。然后进入到我们的用户行为学习器中，这时学习器将对网络流量按照源 IP 地址，目的 IP 地址，协议，源端口号，目的端口号进行重组。得到网络连接集，然后进入我们的参数提取器，在这里将提取出相关的参数，这些参数是为后面的检测做铺垫的。同时，随着时间的推移，用户习惯的改变，我们也可以通过随时更新学习数据，学习新的用户行为来达到渐进成长的目的。最后将连接集合和相关参数合并形成用户行为连接集合。具体的算法将在下一节详细讲述，这里不再叙述。下面本文将叙述蠕虫检测阶段的基本技术框架。

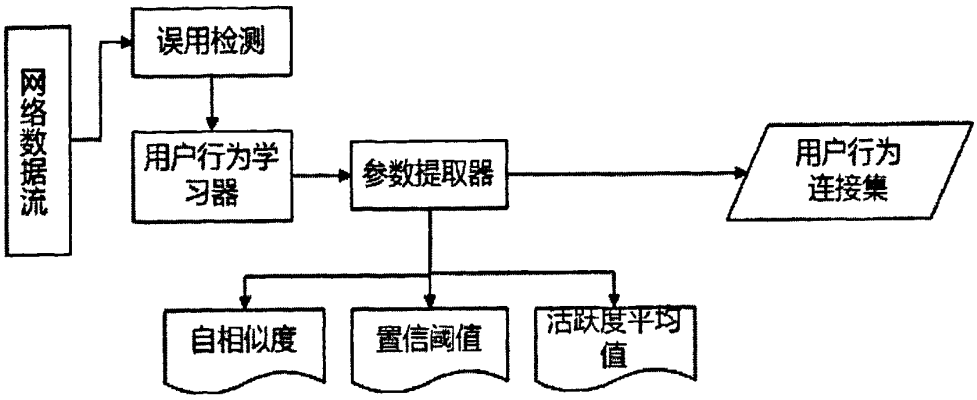

图 4-5 用户行为学习流程框架

4.2.2.2 基于异常用户行为的检测框架

下图 4-6 （异常用户行为的检测框架）描述了在完成学习过程后，进行的检测进程的技术框架。

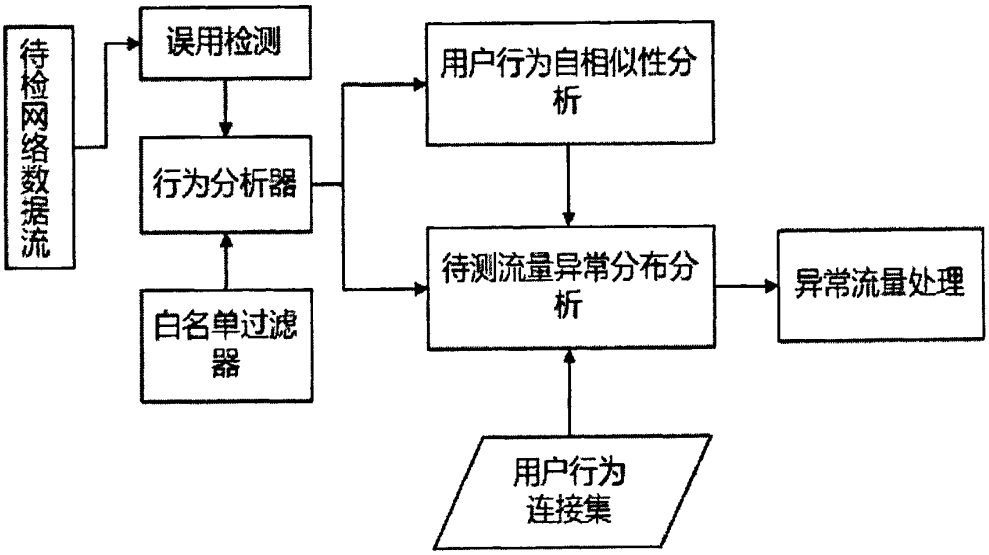

图 4-6 异常用户行为的检测框架

当通过行为学习获取到相应的学习参数后，我们就可以进行蠕虫检测，首先我们将待测网络数据流经过误用检测系统和通过白名单过滤器，筛选掉已知的蠕虫数据流和正常的干扰数据流，然后将数据流通过行为分析器，通过本文提出的基于异常用户行为的蠕虫检测算法计算出相关的参数，最后通过检测算法分流处异常流量，并作出处理。这里，异常流量的处理可以进行阻塞、丢弃、我们还可以把分流出来的异常流量经过本文后面将提出的蠕虫特征码自动提取系统进行特征码自动提取。上面就是本技术的粗略技术框架，接下来我们将详细的介绍基于

异常用户行为的蠕虫检测技术的核心部分，即基于异常用户行为的蠕虫检测技术的核心算法。

4.3 基于异常用户行为的蠕虫检测算法

在前述章节中，我们通过分析正常用户网络行为特征和异常情况下的用户网络行为特征之间的差异，展示了通过分析待检网络行为特征与正常用户网络行为特征之间的偏离程度，能够检测出蠕虫的存在。但是，如何描述正常用户的网络行为特征，如何刻画异常网络行为与正常网络行为之间的差别是整个基于异常用户行为的蠕虫检测技术的核心。因此，我们必须有一个较好的算法予以支持。下面，本文将着重介绍本算法。

4.3.1 算法的基本定义

4.3.1.1 用户网络行为连接集

一个用户网络行为连接可以表示为：

$$u\_con = (l_1, l_2 ... l_k) \tag{4-7}$$

其中l_k表示一个连接因子，它具有一些基本属性，包括协议、源 IP 地址、目的 IP 地址、源端口、目的端口，我们将其表示为l_k={proto, Sip, Dip, Spt, Dpt}。因此我们可以将一个网络连接行为集表示为：

$$U\_con = (u\_con_1, u\_con_2 ... u\_con_i) \tag{4-8}$$

4.3.1.2 用户网络行为连接自相似度

为了分析用户网络行为的重复度，即同一网络行为占总体网络行为的比例，我们引入了这一概念，我们将用户网络行为连接集的自相似度表示为：

$$self\_sim(U\_con) = \begin{cases} 1, & if \sum_{i=1}^{k} self\_sim(l_i) > T_{sim} \\ 0 & \end{cases} \tag{4-9}$$

其中$self_sim(l_i)$为：

$$self\_sim(l_i)=\begin{cases}1, & if\ \dfrac{m}{N}>T_i\\ 0\end{cases} \tag{4-10}$$

其中 T_i 为经验阈值，通过长期测试，我们设定该值为 0.2，m 表示具有相同属性的连接因子 l_i 的连接 *u-con* 的最大数量，N 为连接集的连接个数总数。

4.3.1.3 活跃度平均值和置信阈值

活跃度平均值 $\overline{m_i}$ 表示连接因子 l_k 取值为 V_i 的用户连接数量的平均值。而置信阈值为进行异常分析的参数。它可以表示为：

$$u_i=k*(\overline{m_i}+Z_{a/2}\frac{S_N}{\sqrt{N_i}}) \tag{4-11}$$

其中

$$S_N=\sqrt{\frac{1}{N_i-1}(\sum_{p=1}^{Ni}m_{ip}^2-N_i\overline{m_i}^2)} \tag{4-12}$$

$\sum_{p=1}^{Ni}m_{ip}^2$ 为 m_i 样本的平方和。且上述 *Za/2* 取值为（0-1），其取值越接近 1，表示检测精确度越高，但漏报也越高；但是，精确度和漏报不是简单的反比关系，经过精密测试，该值为 0.96 时，效果最为理想。如果 v_i 在正常状态下从未出现，则 $\overline{m_i}$ 和 u_i 设置为一个固定值，该固定值可以根据网络规模的大小来设定，例如可以为 $\overline{m_i}=\frac{1}{n}*s$，$u_i=\frac{1}{n}*s$；其中，n 表示连接因子 l_i 的所有连接的数量，s 表示一预定值，例如可取为 1。k 为一经验参数，如果该值设置较低，那么当用户短时间连续重复访问某一以访问的地址时，将会导致系统报警；如果该值设置较高，当恶意程序出现对某一个用户常访问的地址发起了 DDOS 攻击时，将出现漏报。因此，该值经过测试，设定在 9.57 较为妥当。

上述几个公式就是在用户行为学习阶段的参数提取器进行特征提取时，所用到的公式。它的目的就是计算出自相似度、活跃度平均值、置信阈值。下面，我们将引入在异常用户行为检测时所需要的公式。

4.3.1.4 异常用户连接行为偏离度

该参数反映了待测单个连接与正常用户连接集的偏离程度。它可以表示为：

$$dis(l_i,v_j,r)=\begin{cases}(\dfrac{m_j}{\overline{m_j}}-1)self\_sim(U\_con_{v_j}), if\ m_j>u_j\\ 0\end{cases} \tag{4-13}$$

4.3.1.5 异常用户行为偏离度

对于当前时刻 t 采集待测连接流量，对每个连接因子 l_i 取其值 v_j，计算 l_i 取值为 v_j 的用户网络连接行为的自相似度 $self\_sim(U\_con_{v_j})$ 和偏离度 $dis(l_i,v_j,r)$，然后得出连接因子 li 的偏离度；重复上述步骤，直至计算出每一个连接因子的偏离度，进而得出异常用户行为分布和正常状态的偏离度 Dis（t），其中 Dis（t）可以表示为；

$$Dis(t)=\sum_{i=1}^{K}\sum_{j=1}^{M}dis(l_i,v_j,r) \tag{4-14}$$

当计算出的 $Dis\ (t)$ 值超过一定的阈值时，则可以发出蠕虫报警。

4.3.2 基于异常用户行为的蠕虫检测算法

在经过用户行为学习过程并计算出相关参数后，边可以开始进行蠕虫的检测，蠕虫的检测归根到底是计算异常用户行为分布和正常状态的偏离度 Dis（t）是否超过阈值。下面就具体的检测算法描述如下：算法 4-15 （基于异常用户行为的蠕虫检测算法）

算法 4-15 基于异常用户行为的蠕虫检测算法

INPUT *S_con, u_i,N_con*
OUTPUT *Dis (t)*

Begin
Input S_con at time=t
for *a_i (i=0, i<total,i++)*
for *every v_j of l_i*
if *l_i is in N-con*
{
count *self_sim(S-con)*
count *$dis(l_i,v_j,t)$ on v_j for each l_i*
}
else
count *$dis(l_i,v_j,t)$ with u=1*
end for
count *$dis(l_i,v_j,t)$*
end for
count *Dis (t)*
if *Dis (t) >Th_{dis}*
give an alarm and redirect traffic to A-ASG system

```
        else
          update the U-con and recount ui
        end if
      end for
    end begin
  return Dis（t）
```

其中，S_con 为待检网络数据流，N_con 为用户行为连接集。在计算偏离都时要分两种情况：1）如果待测连接存在于用户连接集中时，则按算法进行标准的运算，如果计算结果超过阈值，则用户行为异常，如可能因感染病毒出现了 DDOS 攻击，则可能在短时间内重复连续的访问某一地址；2）如果待测连接不存在于连接集中时，则将参数 u 设置为一固定参数，这里我们取 1，并进行计算。最后将总的偏离度与阈值进行比较，如果超过，则应发起警报。当判断出蠕虫流后，我们可以将数据流分流到蠕虫特征码自动提取系统，即本文下一章节将谈到的 A-ASG 系统。如果没有检测到，则可以更新当前的用户行为连接集，同时更新相关参数。

本章，主要详细的描述了基于异常用户行为的蠕虫检测算法，介绍了蠕虫检测的理论过程。下面，我们将以本技术的测试为主线附带构建一个较为完整的检测系统，同时对本技术进行相关测试，对本技术进行评估。

4.4 基于异常用户行为的蠕虫检测系统构建与测试

在进行测试之前，我们将简要的叙述基于异常用户行为的蠕虫检测系统的构建，其中，为了为本文后续所研究的蠕虫特征码自动提取技术结合，本系统将直接嵌套入整个蠕虫检测系统之中，但测试，只针对本技术所涉及的模块进行。下面，本文将简要介绍系统的整体构架。

4.4.1 基于异常用户行为的蠕虫检测系统的整体构架

图 4-7 （基于异常用户行为的蠕虫检测系统整体构架）展示了整个系统的主要模块。本系统利用了目前成熟的 Snort 异常检测系统作为平台，将本系统以插件的形式挂接于 Snort 系统中，这样可以减少在数据包处理上的开发时间，可以让我们把精力集中在算法和技术验证的研究之上。这里对插件的安装做一个简要的介绍。

Snort 的插件技术可以让开发者灵活的加入自己开发的检测插件，并扩展其功

能。因此，我们将本系统作为一个预处理插件安装在 Snort 上。预处理插件的安装需要：1）将插件的头文件包含在文件 Plugbase.c 中；2）将插件的 Setup 函数添加到 Plugbase.c 的 InitPreprocessors()函数中；3）将对应的插件名和参数信息添加到配置文件 Snort.conf 中。

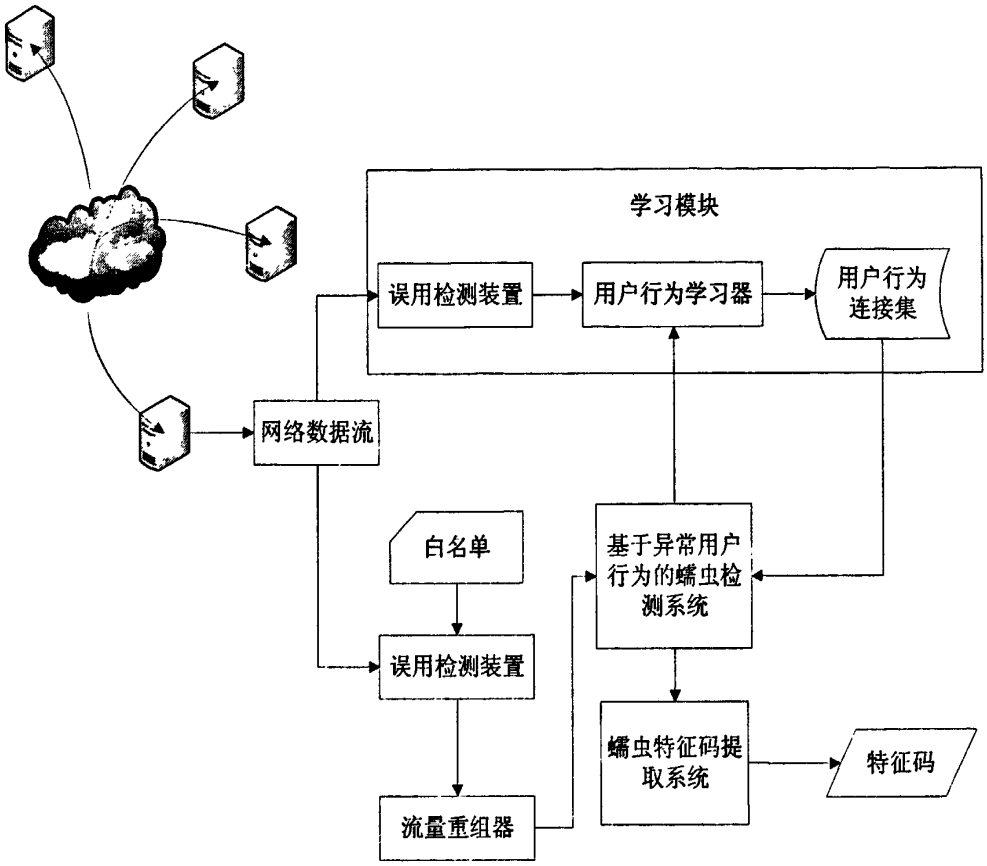

图 4-7　基于异常用户行为的蠕虫检测系统整体构架

另文，本系统的所采用误用检测装置是基于特征码匹配检测的蠕虫检测技术。匹配方法可以直接利用 Snort 的特征码匹配技术，只需将特征码按标准格式记录即可，这里不再累述。接下来，通过实际测试介绍用户行为学习器的作用和实际效果。测试均在表 4-1 （基于异常用户行为的蠕虫检测系统测试环境）中进行

表 4-1 基于异常用户行为的蠕虫检测系统测试环境

操作系统	Windows vista
编译环境	VC2008
CPU	PentiumIV 2.8G
内存	1G

4.4.2 用户行为学习模块测试

理论上正常连接的机器学习将会得到正确的用户连接集和相关参数结果.整个学习过程应该在用户活跃度分布图趋于稳定是停止，图 4-8 则是学习 1 小时、10 小时、40 小时时的用户活跃分布图。我们可以看到在 40 小时时，分布图基本趋于稳定。此时得到的相关参数较为可靠。

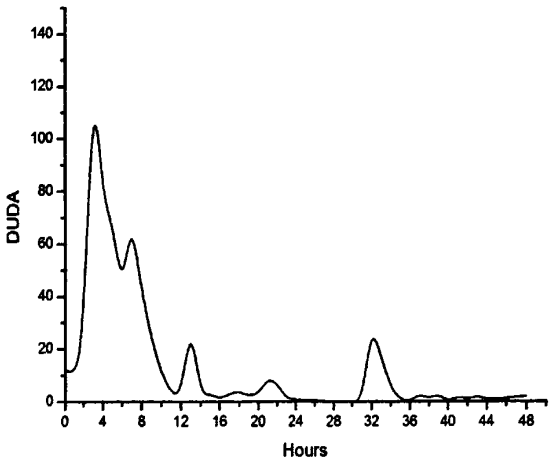

图 4-8 学习过程中的用户活跃度

图 4-8（学习过程中的用户活跃度）也反映了用户 2 天内的网络活动情况，因此在晚间时刻活动逐渐减弱，在凌晨之后甚至将为 0，但第二天开始，用户活跃度明显小于第一天才开始统计时，因此，我们可以进行 48 小时的初期学习。后期的测试将依赖于该学习结果。

4.4.3 正常用户行为测试

下面，我们在用户正常进行网络活动时，进行网络检测，测试显示在正常情况下，用户异常偏离度大大低于阈值。测试结果如下：（测试时间：15 分钟）

ip_src:
ip_dst:
port_dst:
distributionCounteip_src:0.000
distributionCounteip_des:0.000

distributionCounteport:0.000
distributionCountetotal:0.000

ip_src:
ip_dst:
port_dst:
distributionCounteip_src:0.000
distributionCounteip_des:0.000
distributionCounteport:0.000
distributionCountetotal:0.000

ip_src:
ip_dst: 192.168.211.142:1.000||
port_dst:1900:2.003||
distributionCounteip_src:0.000
distributionCounteip_des:1.000
distributionCounteport:2.003
distributionCountetotal:3.003

可以看到最后的 distributionCountetotal，都没有超过阈值 50 。表明为正常。

4.4.4 尼姆达蠕虫测试

理论上可以检测到尼姆达蠕虫，并进行报警，测试结果表明系统通过该项测试，测试结果如下：（测试时间 5 分钟，蠕虫爆发在第 4 分钟）

ip_src:192.168.1.103:39.000||
ip_dst:192.17.59.16:3.000||214.226.222.55:3.000||192.126.213.212:1.000||192.175.136.113:1.000||177.108.173.243:1.000||129.25.253.237:1.000||192.191.164.122:1.000||202.24.3.248:1.000||192.29.129.203:1.000||192.191.179.116:1.000||192.168.181.200:1.000||122.8.73.94:1.000||192.168.157.80:1.000||192.86.6.137:1.000||192.168.42.12:1.000||192.109.120.179:1.000||34.138.146.161:1.000||192.168.88.251:1.000||
port_dst:445:27.000||80:11.000||
distributionCounteip_src:39.000
distributionCounteip_des:22.000
distributionCounteport:38.000
distributionCountetatal:99.000

从结果中，我们可以看到，蠕虫在短时间内访问了大量的陌生地址，从而导

致用户行为瞬间改变，形成了与正常用户行为的偏离， 即出现了异常用户行为。此时，其偏离度已经高达 99，因此，我们可以发出报警。

4.4.5 Busan 蠕虫测试

理论上可以检测到 Busan 蠕虫，并进行报警，测试结果表明系统通过该项测试，测试结果如下：（测试时间 5 分钟，蠕虫在开始测试时爆发）：

ip_src:192.168.1.103:145.000||
ip_dst:192.168.65.132:1.000||192.175.136.113:1.000||129.25.253.237:1.000||145.13.233.112:3.000||49.225.40.77:3.000||192.168.204.64:1.000||192.210.193.140:5.000||192.168.63.172:1.000||177.108.173.243:1.000||192.126.213.212:1.000||192.191.179.116:1.000||192.168.202.104:1.000||192.5.6.192:5.000||192.168.225.223:1.000||192.191.164.122:1.000||147.40.17.127:5.000||192.168.109.156:1.000||192.29.129.203:1.000||202.24.3.248:1.000||192.168.248.88:1.000||122.8.73.94:1.000||192.168.132.20:1.000||192.109.120.179:1.000||192.168.16.207:1.000||185.30.107.246:5.000||192.168.39.72:1.000||192.216.14.226:5.000||192.168.62.192:1.000||192.168.201.124:1.000||192.86.6.137:1.000||229.81.77.108:1.000||192.111.130.147:5.000||192.168.224.243:1.000||34.138.146.161:1.000||192.168.108.176:1.000||118.84.146.251:3.000||192.168.179.239:1.000||192.168.156.120:1.000||192.168.155.140:1.000||192.168.177.44:1.000||192.168.86.36:1.000||34.243.244.131:3.000||192.172.204.167:3.000||81.17.229.72:3.000||192.168.178.4:1.000||192.168.40.52:1.000||192.168.85.56:1.000||192.131.56.72:3.000||192.168.37.132:1.000||192.168.61.231:1.000||192.168.14.12:1.000||192.113.236.196:3.000||192.168.84.96:1.000|
port_dst:80:91.000||
distributionCounteip_src:145.000
distributionCounteip_des:93.000
distributionCounteport:91.000
distributionCountetatal:329.000

可以看到偏离度明显超过阈值，达到 329。

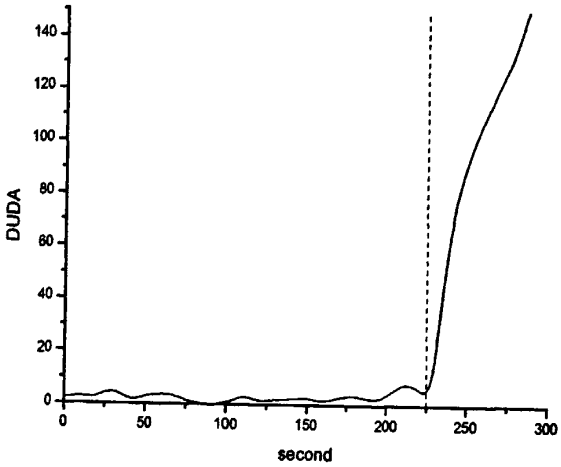

图 4-9 在尼姆达蠕虫爆发时的用户活跃度变化

4.4.6 灵敏性测试

在进行蠕虫测试时，我们对用户活跃度进行了分析同时对比本技术的偏离度值，从而测试系统的灵敏度。测试结果反映，系统能够正确的反应用户活跃度的变化，正确反映出异常的用户行为并出现相应的偏离。如图 4-9 （在尼姆达蠕虫爆发时的用户活跃度变化）与 4-10（尼姆达蠕虫爆发时异常用户行为偏离度变化），我们可以看出两者在蠕虫爆发后的变化具有一致性，并且，异常偏离度反应延迟较小，在蠕虫爆发后 1 分钟左右便宜度超过阈值，即可发出报警。

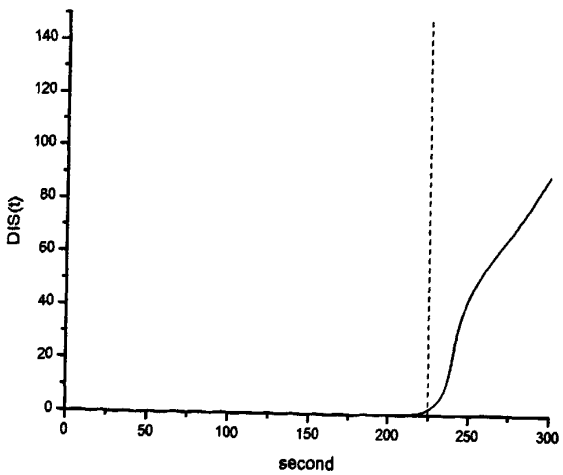

图 4-10 尼姆达蠕虫爆发时异常用户行为偏离度变化

4.5 基于异常用户行为的蠕虫检测系统的改进

通过对系统进行的测试表明，基于异常用户行为的蠕虫检测技术能够有效的检测出蠕虫，并且具有较高的灵敏性。但是如果存在 P2P 软件干扰的情况下，或者用户大量下载软件时将出现较高的误报率。因为，P2P 软件也会在短时间内发起大量的陌生连接。因此，本文通过将主机检测系统引入到系统中，对检测系统进行改进。

具体的来说，就是通过主机检测系统检测系统进程和该进程发送的数据包，如果该进程是存在于白名单中的软件或程序，则它发送的数据包则可以不被检测。通过这种办法，大大的降低了误报率。

4.6 小结

本节介绍了基于异常用户行为的蠕虫检测技术，通过对用户网络行为的分析，分析出异常行为与正常行为的偏离度，从而达到检测蠕虫的目的。虽然本技术取得了一定的检测效果，但是仍然存在许多不足。我们将在后续的研究作出完善和改进，也希望本文的思想能够为蠕虫检测带来新的思考方向。

第五章 抗攻击蠕虫特征码自动提取技术

上一章主要讨论了基于用户异常行为的蠕虫检测技术。本章将结合上述思想，提出一种新的抗攻击蠕虫特征码自动提取技术，它能够有效的抗击目前针对蠕虫特征码自动提取技术的攻击，并且能高效、准确地提取出蠕虫的特征码。本技术也是对蠕虫特征码自动提取技术的又一有益尝试。

5.1 设计思想与目标

本文所提出的抗攻击蠕虫特征码自动提取技术（Anti-attack ASG 下文可简称 A-ASG）是在总结了前人的研究工作，详细的考虑了目前的针对蠕虫特征码自动提取技术攻击技术的基础上提出的一种新的针对多态蠕虫的抗攻击蠕虫特征码自动提取技术。本技术通过引入区域分布式特征码提取思想，使得本技术不仅能够提取出有效、正确的特征码，还能有效的抵御相关的攻击。在本章末尾，我们将对本技术给出详细的测试报告以证明本方法的正确有效性。

5.2 A-ASG 总体设计

5.2.1 整体设计概览

本文提出的 A-ASG 系统与其他几种 ASG 系统不同的是，本文的 A-ASG 只依赖于 ASG 技术基本前提，即“蠕虫在网络中传播的实体数据包总是含有一定的有别于正常数据流的字符串”。图 5-1（A-ASG 的整体结构）显示了 A-ASG 的详细整体设计。首先我们需要对捕获的网络数据包按照协议和目标端口号进行重组，对每一个端口和协议数据流对，我们利用误用检测或者异常检测对其进行筛选，将分离出的可疑流量汇入到可疑池（MPool）中，将无害流汇入正常候选池（NCPool）中。这一步，主要是为了防止 Innocuous pool poisoning attack，而此时，NCPool 并不会立即参与特征码提取，而是会在下一个时刻 T 后，才会被采用。只有到 T 时刻后，MPool 和 NCPool 才会协同参与进行特征码提取。特征码被提取后，将放在区域分布式系统中进行评测和验证，直到验证通过以后，才会被正

式的加入到误用检测系统中。

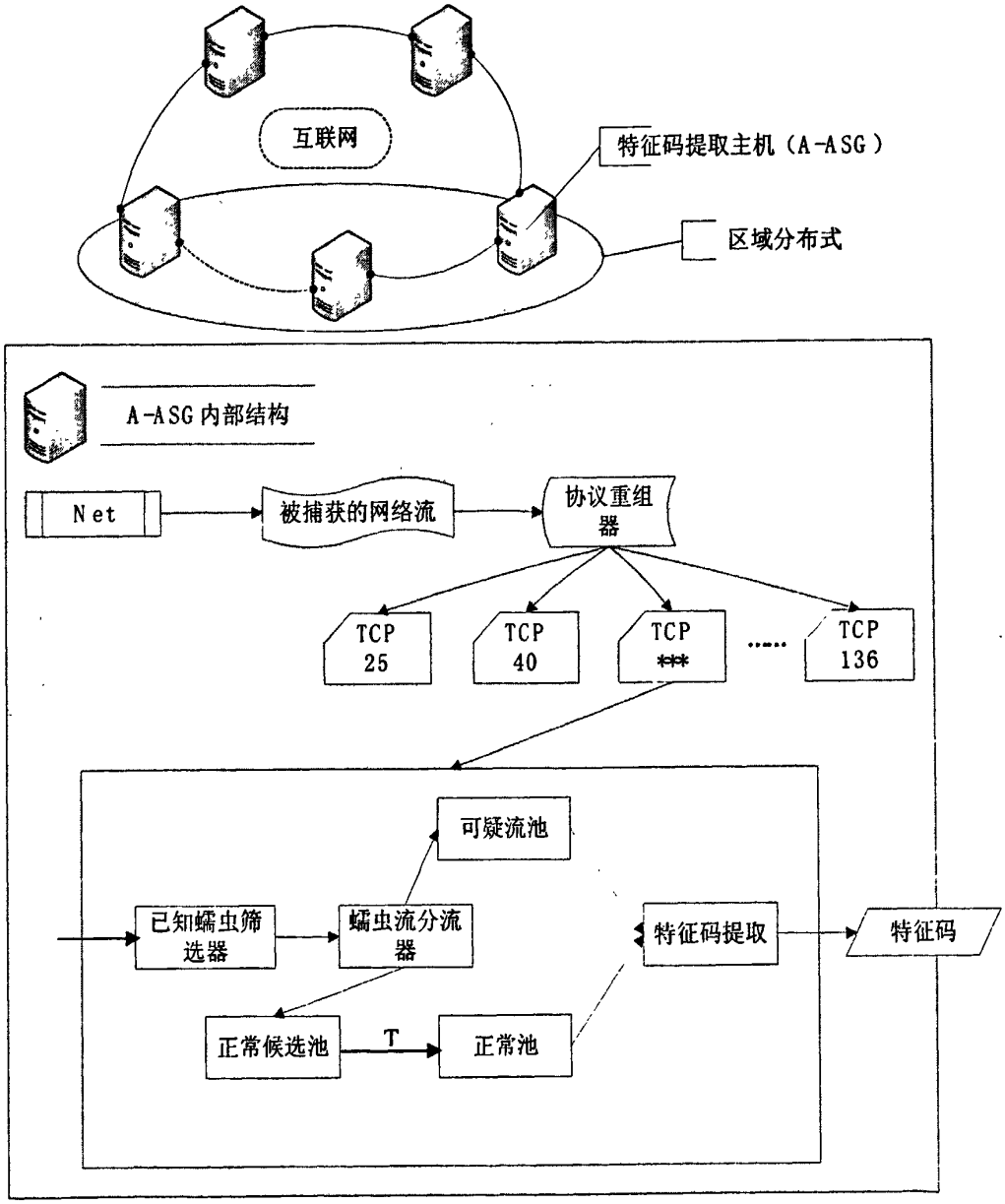

图 5-1 A-ASG 的整体结构

5.2.1.1 特征码提取主机（Signature generator host）

特征码提取主机是安装有 A-ASG 系统的处于网络中的主机，每一台主机都是整个 A-ASG 系统的组成单元。它们共同构成整个区域分布式特征码提取架构并完成特征码的提取。

5.2.1.2 协议重组器（Protocol classifier）

重组器主要是对捕获的网络数据包按照协议和目标端口号进行重组，对每一个端口和协议数据流进行配对，配对的目的是因为有的蠕虫数据不可能在一个数据包中完成，因此需要对同一个协议和端口等的数据包进行配对处理，将他们形成一条数据流。以供后续处理。

5.2.1.3 已知蠕虫筛选器（Known worm filter）

这里主要运用了误用检测系统，通过对网络数据流进行特征码匹配检测，如果发现已知蠕虫数据流，则将这部分流剔除。将其他的数据流汇入蠕虫流分流器中，等待蠕虫流分流器进行下一步的分流。

5.2.1.4 蠕虫流分流器（Worm flow Claaifier）

蠕虫分流器的作用是将收到的数据流进行分流，将其分流成可疑流和无害流，并分别将它们汇入可疑流池和正常流池。

5.2.1.5 可疑流池 Mpool（Suspicious pool）

被存放在可疑流池的数据流是作为特征码提取的提取源，即 A-ASG 系统将从这个池里进行特征码提取。

5.2.1.6 正常候选池 NCpool（Innocuous candidate pool）

被存放在正常候选池的数据流是作为 T 时刻后的特征码提取参考数据，而不是立即进入特征码提取进程中。这样做的好处是可疑防止 Innocuous pool poisoning attack。

5.2.1.7 正常池（Innocuous pool）

存放在该部分的数据流就可以参加特征码的提取进程，并作为特征码提取的产考。

5.2.1.8 特征码提取（Signature generator）

本部分主要是执行特征码提取算法，利用可疑流池的数据作为提取源，正常池的数据作为参考源，提取出正确、有效的蠕虫特征码。

5.2.2 关键技术概览

在本文所提出的 A-ASG 技术中，包含了一些关键的思想，它们是整个 A-ASG

的重要技术思想支撑，分别包括：网络数据包重组思想、蠕虫流分流思想、正常池收集和更新思想、基于区域分布式结构的特征码认证机制

5.2.2.1 网络数据包重组思想

我们知道，蠕虫所发出的数据一般是不能仅用一个网络数据包就能包含的，因此，我们需要将同协议和端口的数据包进行连接重组，形成一条数据流。只有这样，我们才能正常的对数据流进行后续的分流。

5.2.2.2 蠕虫流分流思想

如果不引入蠕虫流分流的思想，那么蠕虫特征码提取将无源可依，因为在大量的数据流中，将不仅包含蠕虫的数据包还会包含正常的数据包。去掉蠕虫流分流将会造成提取的特征码失去有效性和正确性。同时，在引入蠕虫流分流的思想，并构建出蠕虫流分流器以后，蠕虫流分流的质量好坏将决定后续特征码提取的质量。如果蠕虫流分流时将正常的流量分入了可疑池中，那么就可能会造成提取出错误的特征码，从而造成大量的误报。但是，为了避免这一问题的出现，我们在后面的区域分布式特征码认证思想中引入了误报率的检查措施，可以从这一方面进行一定的控制。相反的，如果本属于可疑流的数据流被错误的划分到了正常候选池中，就会有可能造成丢失部分蠕虫特征码，从而造成一定的漏报。蠕虫分流思想是特征码提取过程中最为关键的一步，因此选用恰当的蠕虫流分流技术来构建蠕虫流分流器也是相当重要的。

这里，蠕虫流分流器作为一个接口，它能够用一些蠕虫异常检测的方法来进行分流。在其他人的研究中，分流技术有很多，如基于 TaintCheck[52]和基于执行监测的 Vigilante[54]方法，这两种方法都能精确的对蠕虫进行分流但却效率较低。而基于错误探测的分流法虽然具有较高的效率，却同时拥有了较高的误分率。因此本文将引入前述的基于异常用户行为的蠕虫检测技术作为一种分流方法。这样不仅可以有效的将可疑流和无害流分开，而且还能在特征码提取之前就做出蠕虫预警。

5.2.2.3 正常池收集和思想

值得一提的是，这里我们引入了两个正常池，一个是正常候选池、一个是正常池。下面将分别讨论这两个池的作用。

首先是正常候选池，正常候选池是每次通过分流器得到的无害数据流就进入正常候选池，而不是直接进入正常池。这样做的作用是防止针对正常池的反特征

码自动提取攻击。因为，在正常候选池的数据进入正常池之前，我们会通过上一次提取的最新特征码对这个池的数据进行扫描，这样就可以防止正常池被污染。在一个间隔时间 T 后，我们才将收集到得数据汇入正常池中。当然，在第一次将数据灌入正常候选池时，必须经过精心的过滤和挑选，因为此时正常候选池是“无防备”的，也就是说，第一次注入到正常候选池的内容必须保证是无害的。

其次是正常池，正常池是存储由正常候选池汇入进来的数据内容，并将参与到蠕虫特征码的提取进程之中。为了保证特征码提取以后进行特征码认证，必须每隔一段时间就要对正常池进行更新。否则，随着区域网络内用户习惯的逐步改变，网路数据流也会在宏观上有一些变化。而在特征码认证阶段，我们会将提取出的备选特征码放入到正常池中进行评测，即找出他的误报率。如果正常池数据过时，很有可能造成误报率异常降低，而提取出错误的特征码。

因此，本文提出的两池思想是能够很好的解决针对正常池的反特征码自动提取攻击的方法，具有很强的健壮性。

5.2.2.4 基于区域分布式结构的特征码认证机制

这里所指的区域分布式有别于传统的分布式。传统的分布式较为宏观，是整体的分布式结构，而区域分布式存在两级。分别是网段内微观分布式，网段间宏观分布式。如图 5-2 （区域分布式结构示意图）

从图中可以看出，整个区域分布式结构分为两层，一层为网段内的分布式，一层为网段间的分布式。为了说明这样做的目的，下面先要介绍采用分布式的目的。

我们知道，网络中每个节点的网络数据流是不一样的，这和该节点的用户行为有较大的关系。比如，某用户喜欢浏览网页，因此该网络数据流中，正常流将会以 HTTP 协议为主。而某用户喜欢用迅雷看电影，那么该节点的正常流中将出现以 P2P 协议为主的数据流。那么，数据流的不同，被分离到正常池的数据流就也不会相同。这里就会出现一个问题，例如：在节点 A 进行特征码验证时，攻击者有意的在蠕虫中加入了正常流常有的网页浏览所产生的字符串，而由于用户 A 喜欢用 P2P 软件，因此正常池里边很少含有相关的字符串，这是，被提取出的候选特征码通过正常池进行误报比对时，发现误报率低于阈值，因此通过次备用特征码的认证。这里，我们的 ASG 系统就受到了一次成功的反特征码自动提取攻击，这样被提取的特征码会匹配用户以后正常的网页浏览行为，将其作为蠕虫行为。因此，我们需要引入一个分布式的概念来解决这个问题。此时，当备选特征码被

提取出以后，如果在本机验证误报率低于阈值时，我们还会将该备选特征码发送到其他节点上进行认证。这样，当它被发送到用户 B 的节点上时，由于用户 B 有较多的网页浏览行为，因此该备选特征码就会出现较高的误报率，从而被丢弃掉。因此，这样就能有效的阻止攻击者针对蠕虫特征码自动提取的行为。

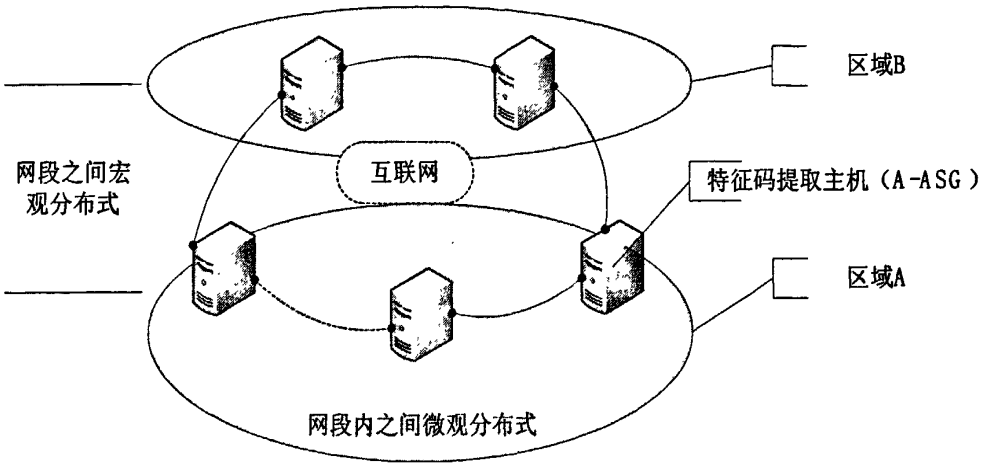

图 5-2 区域分布式结构示意图

但是，如果每一次提取都将特征码放到整个网络的每一个节点去进行认证的话，将会极大的降低效率。因此，我们引入了区域分布式的机制。即，备用特征码首先放入网段内的区域节点中进行认证，如果认证未通过则直接丢弃，如果认证通过再随机的发送到下一个区域分布式中进行认证。这样，就很好的提高了特征码认证阶段的效率。

5.2.3 蠕虫特征码自动提取基本流程概述

本小节主要介绍蠕虫特征码自动提取的基本流程，但其中不包括特征码提取进程中的特征码提取具体算法。这是因为该算法是蠕虫特征码自动提取技术的核心所在，所以将在下一节中进行详细的叙述。所以这里将整个提取算法部分作为特征码提取进程来处理。图 5-3（蠕虫特征码提取流程）详细的介绍了整个提取过程。

第一步：网路数据流进入协议重组器，所谓重组，实际上是将由同一个蠕虫发送的不同数据包进行整合，整合成一条完整的数据流。这个整合过程将按照数据包的协议、端口号进行重组。并且通过顺序链进行存储。每一条流将分配一个顺序链，因此重组之后可能存在几条流的情况。

第二步：当数据包被重组完成之后将进行已知蠕虫的检测。这样才能保证进

入分流器的数据流不含有已知的蠕虫，从而尽可能的减少正常池被污染的可能。

第三部：蠕虫分流器收到数据流后，对其进行分析。如果数据流为异常的则自动分向可疑池，否则分向正常候选池。正常候选池在等待时刻 T 后，将数据发往正常池中。

第四步：当可疑池收到数据后，提取进程将利用可疑池作为源，正常池作为参考进行特征码提取。整个过程是一个 TOKEN 的筛选和重组的过程。只有当被提取出来的特征码通过特征码认证后，才能被定为正式的可用特征码而输出到相关的误用检测系统中。

接下来，本文将着重讲解蠕虫特征码提取的具体算法，和其抗攻击能力的详细分析。

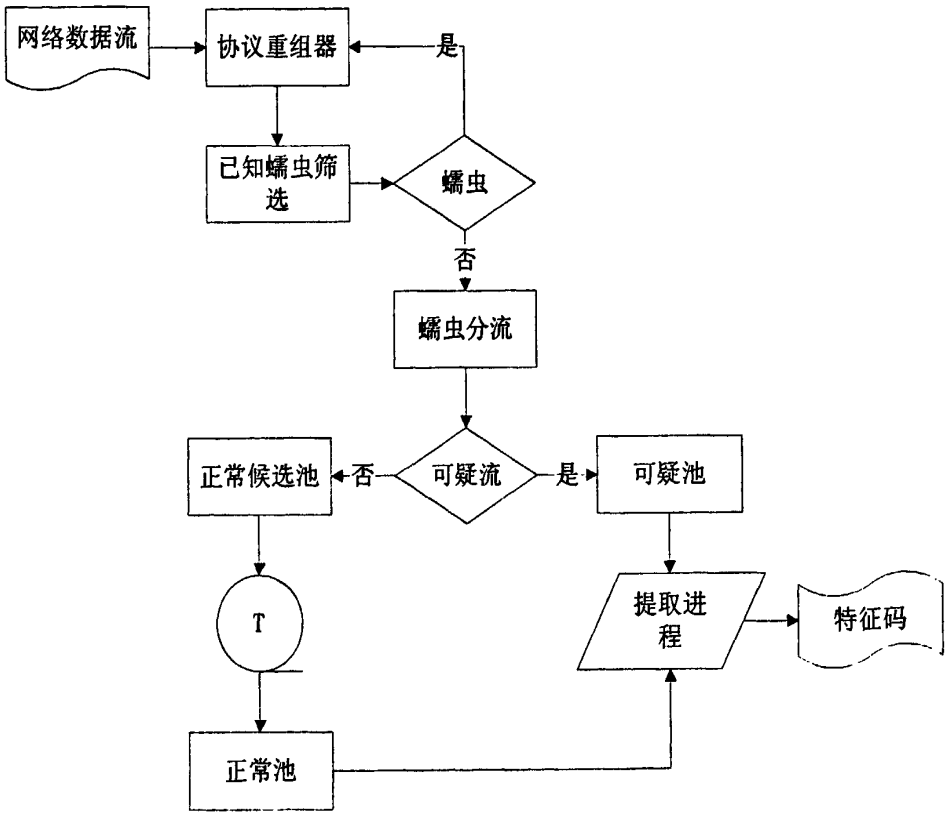

图 5-3 蠕虫特征码提取流程

5.3 蠕虫特征码提取算法

本部分将详细介绍蠕虫特征码的具体算法，包括提取进程的具体流程和工作

方式。在介绍之前，首先将对算法中涉及到得数学变量进行定义。定义如表 5-1（蠕虫特征码提取算法数学变量定义表）所示。

表 5-1 蠕虫特征码提取算法数学变量定义表

蠕虫特征码提取算法数学变量定义表	
SP	可疑池 Suspicious Pool
I-CP	正常候选池 Innocuous Candidate Pool
I-P	正常池 Innocuous Pool
L_{min}	一个整数参数，参数 $STR_i \geq L_{min}$
TS	令牌集 Token Set
SP_k	可疑池中的第 k 条流
t_i	令牌 t_i，其中 $t_i \in TS$
STR_i	令牌 t_i 的字符串集
tf_i	令牌 t_i 的属性
S	特征码集合
s_i	特征码 s_i，这里 $s_i \in S$
SC	候选特征码集
sc_i	候选特征码 $sc_i \in SC$
FP_{si}	特征码 s_i 的误报率
FP_{acpt}	误报率接受阈值，FP_{si} 不能大于 FP_{acpt}，否则不予接受
COV_{si}	s_i 在可疑池中的覆盖率
COV_{acpt}	特征码在可疑池中的覆盖率阈值，COV_{si} 不得小于 COV_{acpt}，否则不予接受
M_{si}	包含特征码 s_i 的可疑流
COV_{min}	$\|M_{si}\|/\|M\|$ 不得小于 COV_{min}

接下来，本文将以提取进程的工作流程为线索，如图 5-4（蠕虫特征码提取进程）所示，逐步介绍本文提出的抗攻击蠕虫特征码自动提取技术的关键算法。

5.3.1 令牌提取算法

特征码提取进程的第一步为令牌提取。所谓令牌实质上是一串二进制比特序列。我们将可疑池都看做二进制字节流(或字符串)，令牌提取的本质是从这些字节流中提取出满足以下要求的的子串：（1）子串长度不得小于 L_{min}；（2）子串的覆盖率至少满足 COV_{acpt}。这里有必要对覆盖率做一个解释，所谓覆盖率就是指该子串在一条流中出现的比率，只有字符串的覆盖率高于 COV_{acpt} 才能被作为一个令牌。这一步骤可以借助现有的一些令牌提取算法完成，如 Hamsa 提出的算法，只是要注意，本文添加了一个约束条件，也就是上面的条件（2）。只有满足这一条件的才能被选为令牌。

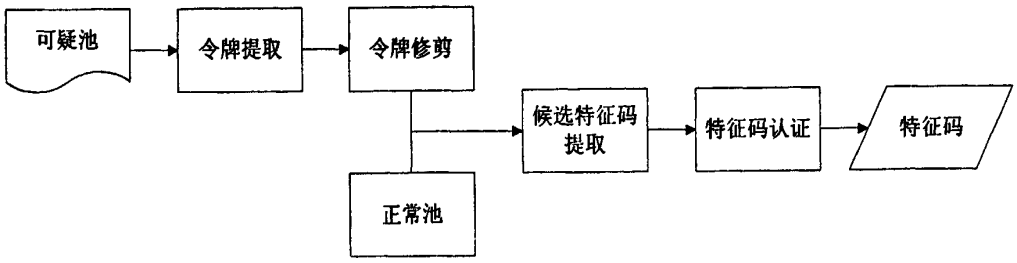

图 5-4 蠕虫特征码提取进程

5.3.2 令牌修剪算法

一个令牌必然出现在可疑池中的某一个流里，但并不意味着他只出现在一个流里，它还可能出现在其他流中。因此，我们不能用一个单一的变量符号来表示这个令牌，因为我们还需要标识出它还存在于其他那些流中。因此我们定义了一个二元组 $t_i = \{STR_i, tf_i\}$ 来表示一个令牌。其中 tf_i 是 STR_i 的令牌信息。如，目前在可疑池中有 3 条数据流，字符串“%http://1a%.”在流 A 中出现了 2 次，在流 B 中出现了 3 次，而在流 C 中没有出现。因此，这个令牌“%http://1a%.”的 tf={2，3，0}。为了保证被提取的令牌不会成为其他令牌的子令牌，即出现包含关系，我们就需要对令牌进行修剪。例如我们得到两个个令牌 t_i=“%http://1a%”，tf={2，3，1}；t_j=“%http://1a%abc”，tf={2，2，0}，我们就需要将 tf={2，3，1}修改为 tf={0，3，1}，因为 tf_i 是 tf_j 的子令牌。令牌修剪算法如下：算法 5-1 （令牌修剪算法）

算法 5-1 令牌修剪算法

INPUT: TS, $flow\_num$
OUTPUT: TS without subtokens
  for all $t_i \in TS$ **do**
    for all $t_j \in TS \,\&\&\, t_i \neq t_j$
      if $STR_i \subset STR_j$ **then**
        for $k = 1$ **to** $flow\_num$ **do**
          if $tf_i[k] == tf_j[k]$ **then**
            $tf_i[k] = 0$
          end if
        end for
        if $tf_i[k] = 0, \forall tf_i[k] \in tf_i$ **then**
          $TS.delete(t_i)$
          break
        end if
      end if
    end for
  end for
return TS

5.3.3 候选特征码提取算法

候选特征码由被提取出来的令牌所组成，但是要将令牌归入候选特征码必须满足一下几个条件：(1)令牌的 COV 值必须达到规定的 COVacpt 阈值；其中 COV 值的计算如算法 5-2（令牌覆盖率算法）所示；（2）令牌必须满足规定的 FP_{acpt} 阈值。一旦令牌中集（TS）有符合这两个条件的令牌 t_i，将被发送到候选特征码中（SC），同时从令牌集（TS）中删除 t_i。而剩下的令牌集将按照算法 5-3 （令牌合并算法）进行令牌重组，形成新的令牌，同时自动删去原有的令牌。例如：

目前有两个令牌，t_a= “ab”，tf_a=(2,3,0)；t_b= “cd”，tf_b=(1,2,2)，则新构成的t_c= “ab” + “cd”，tf_c=(1,2,0)。

算法 5-2 令牌覆盖率算法

INPUT t_i, *flow_num*
OUTPUT amount of suspicious flows which including t_i
  num = 0
  for *i* = 1 **to** *flow_num* **do**
    if $tf_i[i] \neq 0$ **then**
      num = *num* +1
    endif
  end for
  return *num*

算法 5-3 令牌合并算法

INPUT t_i, t_j, *flow_num*
OUTPUT a new token if exists
  $t_u = \varnothing$
  $STR_u = STR_i \cup STR_j$
  for $k = 1$ **to** *flow_num* **do**
    $tf_u[k] = min(tf_i[k], tf_j[k])$
end for
if $tf_u[k] == 0, \forall tf_u[k] \in tf_u$ **then**
    return $\varnothing$
else
return t_u
    end if

5.3.4 特征码认证算法

特征码认证是特征码提取进程的最后一步，它是将候选特征码放入到区域分布式系统中进行认证。认证通过的条件为：对于每一个分布式节点候选特征码必须满足

$$s_i \in S \Leftrightarrow s_i \in SC \&\& FP_{s_i} \leq FP_{acpt}, \quad (5\text{-}4)$$

算法 5-6 特征码认证算法

INPUT: SP, ICP, FP_{acpt} and COV_{acpt}

OUTPUT: Signature set S for worms in M

$TS = SC = S = \varnothing$

$TMPS = \varnothing$

$TS = SP.getTokenList()$

$flow\_num = SP.getFlowNum()$

$delSubToken(TS, flow\_num)$

for all $t_i \in TS$ **do**

  if $calCov(t_i, flow\_num) < COV_{acpt}$ **then**

    $TS.delete(t_i)$

  end if

end for

while $TS.IsnotEmpty()$ **do**

  for all $t_i \in TS$ **do**

    if $calcFP(t_i, N) < FP_{acpt}$ **then**

      $SC.append(t_i)$

      $TS.delete(t_i)$

    end if

  end for

  for all $t_i \in TS$ **do**

    for all $t_k \in TS \&\& t_i \neq t_k$ **do**

      $newToken = merge(t_i, t_k, flow\_num)$

      if $newToken \neq \varnothing$ $calCov\ newToken\ flow\ num\ COV_{acpt}$ **then**

        $TMPS.append(newToken)$

```
            end if
        end for
        TS.delete(t_i)
    end for
    TS = TMPS
   TMPS =∅
end while
for all t_i ∈ SC do
    if ValidateSC(t_i) then
        S.append(t_i)
    end if
end for
return S
```

算法 5-6 （特征码认证算法）详细说明了整个认证过程。这个过程实际上是一个排除误用特征码的过程。值得注意的是，这个认证过程首先是在区域节点内进行认证，而不是直接就将特征码散发到整个分布式节点中进行认证。只有待区域节点认证通过后，才再次随机散发到另一个区域节点集中进行认证。

以上内容就是特征码提取进程的核心算法，也是本文提出的蠕虫特征码自动提取技术的核心所在。令牌提取、令牌修剪、候选特征码提取、特征码认真四个步骤，提取出有效的特征码，同时能够抵抗一些针对特征码自动提取技术的攻击。接下来，我们将通过实验测试分析本技术的抗攻击能力。

5.4 技术实现与测试

本部分主要通过构建蠕虫特征码提取平台，将前述功能模块实例化，最后通过反特征码自动提取攻击测试来横向比对现有的特征码自动提取技术，同时验证本技术方案的可行性。

5.4.1 技术实现平台

本次测试在表 5-2（测试平台环境）实验环境中进行：

表 5-2 测试平台环境

	实验 1
操作系统	Windows vista
编译环境	VC2008
CPU	PentiumIV 2.8G
内存	1G
变形引擎	Metasploit

5.4.2 协议重组器和蠕虫分流器的实现

这里我们将协议重组器和蠕虫分流器利用基于异常用户行为的蠕虫检测系统中的检测模块进行实现。该检测模块主要是利用 Snort 进行流量捕获，在利用我们自行开发的插件，首先将端口和协议相同的流进行链接，然后通过检测模块的检测算法进行分流。首相将数据流通过误用检测，将那些被误用检测查出的数据流删除，然后将剩下的数据流通过异常检测模块，将使检测模块发出报警的流发向可疑池；其他的发向正常候选池。

5.4.3 特征码认证的实现

特征码认证主要利用了基于异常用户行为的蠕虫检测平台，同时加入分布式模块，将每一个蠕虫检测平台统领起来。这里要加入一个控制端，控制端负责接收由节点提取的候选特征码，并通过控制端进行特征码认证发放。此部分在前述章节关于基于异常用户行为的蠕虫检测系统实现部分已经介绍，这里不再累述。

5.4.4 蠕虫变形的实现

我们主要使用 Metasploit 来模拟蠕虫对漏洞主机的攻击。Metasploit 是一个非常好的蠕虫负载（payload）变形工具，他可以通过多种算法对蠕虫进行变形。

5.4.5 无攻击状态下特征码提取测试

首先，为了验证本技术的可用性，我们先不进行反特征码自动提取攻击，而直接提取正常的多态蠕虫特征码。在本次测试中，我们将数据流通过前述模块保证了在可疑池中只有蠕虫流而在正常池中只有正常数据流。同时，本次测试的各个参数设置为：L_{min}=2，COV_{acpt}=4，FP_{acpt}=0.005。本次测试中，我们主要横向比对了之前的 Polygraph，Hamsa，Lisabeth 三种特征码提取技术，同时还比较了最初的最长子串提取技术。我们使用了几种软件漏洞进行蠕虫攻击测试，同时将对应漏洞的蠕虫特征码提取出来，其测试结果见表 5-3 至 5-5。

表 5-3 蠕虫攻击 PeerCast<=0.1216 URL 处理代码缓冲溢出漏洞的特征码

Class	FN	FP	Signature
Longest substring	0%	92.7%	‘ HTTP/1.0\r\n\r\n’
Polygraph	0%	0%	‘GET ’, ‘ HTTP/1.0\r\n\r\n’, 0x77dc12b8
Hamsa	0%	0%	‘GET ’, ‘ HTTP/1.0\r\n\r\n’, 0x77dc12b8
Lisabeth	0%	0.0021%	0x77dc12b8
A-ASG	0%	0.0021%	0x77dc12b8

表 5-4 蠕虫攻击 Ipswitch WS_FTP Server 5.0.5 XMD5 溢出漏洞的特征码

Class	FN	FP	Signature
Longest substring	0%	0.43%	‘XMD5 ’
Polygraph	0%	0.0003%	‘XMD5 ’, ‘\r\n’, 0x7c2ec663
Hamsa	0%	0.0003%	‘XMD5 ’, ‘\r\n’, 0x7c2ec663

Lisabeth	0%	0.0037%	0x7c2ec663
A-ASG	0%	0.0037%	0x7c2ec663

表 5-5 蠕虫攻击 MailEnable IMAPD 2.35 登录请求溢出漏洞的特征码

Class	FN	FP	Signature
Longest substring	0%	0.62%	' LOGIN '
Polygraph	0%	0%	' LOGIN ', '\r\n\r\n', 0x10049abb
Hamsa	0%	0%	' LOGIN ', '\r\n', '\r\n\r\n', 0x10049abb
Lisabeth	0%	0.0006%	0x10049abb
A-ASG	0%	0.0006%	0x10049abb

从测试结果看。由 Polygraph 和 Hamasa 提取出的特征码拥有最低的误报率和漏报率，而 Lisabeth 和本文提出的 A-ASG 拥有的误报率和漏报率稍高，但差别基本可以忽略。这一差别是由于前两者是提取的最长子串，因此误报率和漏报率才相对较低。在后面的测试中，我们可以看到，当面对反特征码攻击技术时，现有的几种特征码提取技术便开始暴露出缺陷。

5.4.6 Red herring 攻击状态下特征码提取测试

前面，我们进行了在无攻击状态下的特征码提取测试，清晰的反应了本文提出的特征码自动提取技术的可用性，虽然它存在与另外两种特征码自动提取技术在 FP 值上的极其细微的差别，而且这个差别是可以忽略的。在此次的测试中我们将引入 Red herring 攻击，我们在蠕虫数据包中有意的添加了伪造的特征字符串，用以混淆，该伪造的特征字符串至少满足最低的 L_{min} 值，因此它将被包含到令牌集合中。同时本次测试的参数如下：L_{min}=2，COV_{acpt}=4，FP_{acpt}=0.005。测试结果见表 5-5 至 5-7。

表 5-5 在 Red herring 攻击下的基于 PeerCast<=0.1216 URL 处理代码缓冲溢出漏洞的蠕虫特征码

Class	FN	FP	Signature
Longest substring	100%	0%	0xd09763a68f37

Polygraph	100%	0%	'GET ', ' HTTP/1.0\r\n\r\n', 0x77dc12b8, 0xd09763a68f37
Hamsa	100%	0%	'GET ', ' HTTP/1.0\r\n\r\n', 0x77dc12b8, 0xd09763a68f37
Lisabeth	0%	0.0021%	0x77dc12b8
	100%	0%	0xd09763a68f37
A-ASG	0%	0.0021%	0x77dc12b8
	100%	0%	0xd09763a68f37

表 5-6 在 Red herring 攻击下的基于 Ipswitch WS_FTP Server 5.0.5 XMD5 溢出漏洞的蠕虫特征码

Class	FN	FP	Signature
Longest substring	100%	0%	0x6a259fe48b0a6b
Polygraph	100%	0%	'XMD5 ', '\r\n', 0x7c2ec663, 0x6a259fe48b0a6b
Hamsa	100%	0%	'XMD5 ', '\r\n', 0x7c2ec663, 0x6a259fe48b0a6b
Lisabeth	0%	0.0037%	0x7c2ec663
	100%	0%	0x6a259fe48b0a6b
A-ASG	0%	0.0037%	0x7c2ec663
	100%	0%	0x6a259fe48b0a6b

表 5-7 在 Red herring 攻击下的基于 MailEnable IMAPD 2.35 登录请求溢出漏洞的蠕虫特征码

Class	FN	FP	Signature
Longest substring	100%	0%	0x3e079a5f825b
Polygraph	100%	0%	' LOGIN ', '\r\n\r\n', 0x10049abb, 0x3e079a5f825b
Hamsa	100%	0%	' LOGIN ', '\r\n', '\r\n\r\n',

			0x10049abb, 0x3e079a5f825b
Lisabeth	0%	0.0006%	0x10049abb
	100%	0%	0x3e079a5f825b
A-ASG	0%	0.0006%	0x10049abb
	100%	0%	0x3e079a5f825b

从测试结果可以看到，所有的基于最长子串的特征码提取技术出现了 100%的漏报率。他们提取出了被故意添加的伪造字符串。而 Lisabeth 和 A-ASG 却能够提取出正确的特征码。因此从这一方面来说，Lisabeth 和 A-ASG 是能够抵御 Red herring 攻击的。

5.4.7 Correlate outlier 攻击状态下特征码提取测试

在测试 Correlate outlier 攻击状态下的特征码提取效果前，我们需要构造一个 Correlate outlier 攻击。为此，我们从网络数据流中选择出了一个特别的数据子串，该子串在正常池中出现的概率大于 5%而小于 10%，并用此作为一个特殊的协议框架加入到我们伪造的特征子串中用于 Correlate ourlier 攻击。本次测试的参数设置如下：伪造特征数为 3，贝叶斯阈值 1.5，A-ASG 参数为：L_{min}=2，COV_{acpt}=4，FP_{acpt}=0.005。测试结果见表 5-8 至 5-10。

表 5-8 在 Correlated Outlier Attack 攻击下的基于 PeerCast <= 0.1216 URL 漏洞的蠕虫特征

Class	FN	FP	Signature
Longest substring	100%	4.3%	'Microsoft-IIS/6.0\r\n'
Polygraph	0%	9.7%	'GET ': 0.0035, ' HTTP/1.0\r\n\r\n': 0.0013, 0x77dc12b8: 5.7329, 'image/gif\r\n': 2.4393, 'Microsoft-IIS/6.0\r\n': 1.6385, 'GIF89a': 2.1744
Hamsa	100%	0%	'GET ', ' HTTP/1.0\r\n\r\n', 0x77dc12b8,
Lisabeth	0%	0.0021%	0x77dc12b8
A-ASG	0%	0.0021%	0x77dc12b8

表 5-9 在 Correlated Outlier Attack 攻击下的基于 Ipswitch WS_FTP Server 5.0.5 XMD5 漏洞的蠕虫特征

Class	FN	FP	Signature
Longest substring	100%	4.3%	'Microsoft-IIS/6.0\r\n'
Polygraph	0%	8.5%	'XMD5 ': 0.7340, '\r\n': 0.0000, 0x7c2ec663: 7.3365, 'image/gif\r\n': 2.4393, 'Microsoft-IIS/6.0\r\n': 1.6385, 'GIF89a': 2.1744
Hamsa	0%	0.0003%	'XMD5 ', '\r\n', 0x7c2ec663
Lisabeth	0%	0.0037%	0x7c2ec663
A-ASG	0%	0.0037%	0x7c2ec663

表 5-9 在 Correlated Outlier Attack 攻击下的基于 MailEnable IMAPD 2.35 登录请求漏洞的蠕虫特征

Class	FN	FP	Signature
Longest substring	100%	4.3%	'Microsoft-IIS/6.0\r\n'
Polygraph	0%	7.6%	' LOGIN ': 0.0035, '\r\n\r\n': 0.0000, 0x10049abb: 5.8664, 'image/gif\r\n': 2.4393, 'Microsoft-IIS/6.0\r\n': 1.6385, 'GIF89a': 2.1744
Hamsa	0%	0%	' LOGIN ', '\r\n', '\r\n\r\n', 0x10049abb
Lisabeth	0%	0.0006%	0x10049abb
A-ASG	0%	0.0006%	0x10049abb

从测试结果可以看到，所有的基于最长子串的特征码提取技术出现了 100%的漏报率。而基于贝叶斯算法的特征码提取算法也包含了伪造的特征子串。而其他的如 Hamsa，Lisabeth 和 A-ASG 却没有包含，因为它的误报率已经高于了阈值

FP_{acpt}。由此我们可以看出 A-ASG 能够很好的抵御该种攻击。

5.4.8 Suspicious Pool Poisoning 攻击状态下特征码提取测试

在此测试试验中，我们将上面三种针对上述各个漏洞的蠕虫流同时加到可疑池中，同时将部分正常流也加入到可疑池中。从而测试在可疑池被污染的状态下，各个特征码自动提取算法的健壮性。测试结果如图 5-5（Suspicious Pool Poisoning 攻击状态下特征码提取测试结果）。

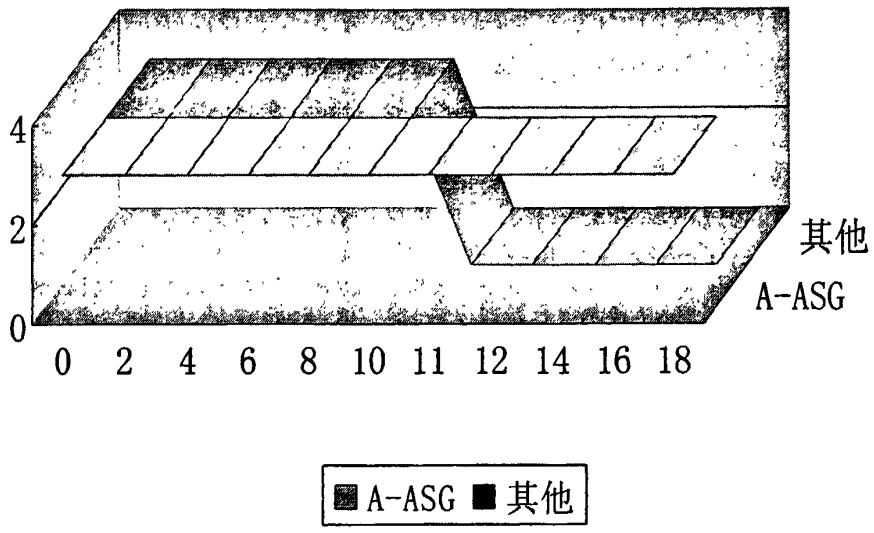

图 5-5 Suspicious Pool Poisoning 攻击状态下特征码提取测试结果

通过测试图，我们可以看到，在污染流小于 10 条时，各个特征码自动提取技术都表现良好，但是当污染流高于 10 条后，除了 A-ASG 外的特征码自动提取技术出现不能提取特征码的情况。这一个测试充分说明了 A-ASG 的健壮性和抗污染攻击能力。

5.5 小结

随着蠕虫特征码自动提取技术的兴起并不断的被完善，ASG 系统将逐步的替代部分人工的特征码提取，这将有利于面对目前飞速增长的蠕虫攻击，特别是多态蠕虫的攻击。但是，在 ASG 技术出现后，针对 ASG 技术的攻击也开始被研究，

并取得了一定的攻击效果。面对这一挑战，本文在总结了前人的研究经验，并充分考虑了目前的反蠕虫特征码自动提取技术，提出了 A-ASG 抗攻击蠕虫特征码自动提取技术。通过 5.4 节的测试，可以看到 A-ASG 技术的抗攻击能力和健壮度。同时为了减小整体分布式的效率损耗，我们做出了区域分布式的改进，使得提取技术的效率有了一定改善。在下一步工作中，我们将继续改进本技术，让它更具实用性，能够更好的满足现实条件的网络环境。

第六章 结束语

蠕虫检测是计算机安全领域的重要内容之一，与此相伴的则是蠕虫的特征码自动提取，两者相辅相成。本文紧紧围绕蠕虫的检测和特征码自动提取进行了深入的分析和研究，总结了目前的研究成果，并提出了自己的检测方法和特征码提取方法，为进一步研究奠定了基础。

6.1 论文的主要成果、创新与不足

本文综述了计算机蠕虫的相关原理、关键技术、检测方法以及蠕虫特征码提取的相关技术，并提出了一套新的蠕虫检测方法和蠕虫特征码自动提取方法，在一定程度上弥补了现有技术的不足。

本文的主要工作和贡献包括：

1：本文详细的概述了蠕虫的原理和现有的蠕虫检测技术、蠕虫特征码提取技术并分析了现有蠕虫检测技术和蠕虫特征码提取技术的优点和不足。同时对未来蠕虫的发展和蠕虫检测于特征码提取技术的发展做出了预测。

2：本文通过对用户网络行为的研究分析，建立了用户行为模型，提出了基于异常用户行为的蠕虫检测算法，另辟蹊径，打破了传统的基于失败连接分析的蠕虫检测模式。我们还采用了渐进式的用户行为学习模式，不断更新用户行为，大大降低了误报率和漏报率。同时，本文还结合了现有的主机检测思想，设置白名单，有效的避免了对 P2P 软件的误报。

3：本文还针对蠕虫特征码人工提取和早期特征码自动提取技术的不足，提出了抗攻击蠕虫特征码自动提取技术，有效的弥补了人工提取的滞后性和早期特征码自动提取技术在变形蠕虫特征码提取和抗反特征码自动提取攻击能力差的不足。这也是对蠕虫特征码自动提取技术的有益尝试。

但论文在许多部分仍存在不足，主要存在以下几个方面：

1. 用户行为模型还较为粗糙，模型建立较为理想化，仍然有一些实际的情况并没有考虑到其中。

2. 在用户行为偏离度的判定上仍有不足，特别是在部分参数的选定上具有经验性，并不能做到根据每一个用户节点的网络环境自动的计算出相关的参数。

3. 对 P2P 软件等非蠕虫行为的感染还不能从网络行为来予以识别，还需要借助主机检测技术来辅助识别。

4. 对 P2P 蠕虫的检测因目前并没有相关蠕虫而不能展开测试。

5. 在蠕虫特征码提取上面，整个系统效率仍然较低，还不能真正的用于实际的特征码提取。

6.2 未来的研究工作

本文的工作仅是对基于异常网络行为的蠕虫检测的初步研究，以后的研究工作可在以下几方面开展开：

1. 进一步优化用户网络行为模型，使得模型更能贴近真实的网络环境，为检测算法的研究提供理论依据。

2. 进一步对用户行为进行研究，不仅是从网络连接的一般属性进行统计判别，还希望能找到更具代表性的相关行为。

3. 改进特征码自动提取算法，提高相关效率。

致 谢

本研究及学位论文是在我的导师张小松老师的亲切关怀和悉心指导下完成的。他严肃的科学态度，严谨的治学精神，精益求精的工作作风，深深地感染和激励着我。从课题的选择到项目的最终完成，张老师都始终给予我细心的指导和不懈的支持。两年多来，张老师不仅在学业上给我以精心指导，同时还在思想、生活上给我以无微不至的关怀，在此谨向张老师致以诚挚的谢意和崇高的敬意。

在此，我还要感谢在一起愉快的度过研究生生活各位同学门，正是由于你们的帮助和支持，我才能克服一个一个的困难和疑惑，直至本文的顺利完成。特别感谢陈厅同学和潘小会同学，他们对本课题做了不少工作，给予我不少的帮助。

在论文即将完成之际，我的心情无法平静，从开始进入课题到论文的顺利完成，有多少可敬的师长、同学、朋友给了我无言的帮助，在这里请接受我诚挚的谢意!最后我还要感谢培养我长大含辛茹苦的父母，谢谢你们!

参考文献

[1] John F.Shoch, Jon A.Hupp. The "worm" programs-early experience with a distributed computation. Communications of the ACM, 1982,25(3):p172-180

[2] 谭毓安. 新型网络病毒一红色代码病毒的分析和防范[J]. 计算机系统应用，2002，2：67-71

[3] Shoch J F'Hupp J A. The "Worm'' Progmms 一--Early Experfonce with a Distributed Computation[J] Communications oftheACM，1982，22(3)：172-180.

[4] Spafford EH. m Intemet worm program：An analysis. Technical Report,CSD-TR-823，West Lafayette：Department ofComputer Science,Purdue University。1988. i 一 29

[5] Eugene H Spaford. The Internet worln program：an analysis[J]. ACM SIGCOM M Computer Communication Review,1989，19 (1)：17—57.

[6] David Moore，Vern Paxson，Stefan Savage，et al. Inside the slammerworm[J]. IEEE Security and Privacy,2003，1(4)：33-39.

[7] 向郑涛，陈宇峰，董亚波，鲁东明，蠕虫检测技术研究进展，计算机工程与设计，2009， 30 (5)

[8] 郑辉. Internet蠕虫研究：[博士学位论文]. 南开大学：2003

[9]文伟平，卿斯汉，蒋建春，王业君. 网络蠕虫研究与进展. 软件学报，2004，15(08)：1208—1219

[10] 何朝辉。网络蠕虫研究及本地化蠕虫预警研究及原型系统设计：[硕士学位论文]。北京：中国科学院软件研究所 2005。

[11] D. G Glazer, "Computer Worms," May 2005, http://www.research.umbc.edu/~dgorin1/is432/worms.htm

[12] R. A. et al., Snort 2.1 Intrusion Detection, 2nd ed., Syngress,O'Reilly, 2004, pp. 490–91.

[13] Liao Mingtao, Zhang Deyun, Hou Lin, Li Jinku. "Network Worms Detecting System Based on Failed Connections Analysis". 微电子与计算机，24 卷第 5 期，2007.

[14] Liao Mingtao, Zhang Deyun, Hou Lin. "A Novel Approach for Early Detection of Worm Based on Failed Connection Flow Dissimilarity". Computer Engineering，August 2006.

[15] Stuart E. Schechter, Jaeyeon Jung, and Arthur W. Berger. "Fast Detection of Scanning Worm Infections".

[16] WANG Ping, FANG Bin-xing, YUN Xiao-chun, PENG Da-wei. User-habit based early warning of worm. Journal on Communications February 2006 Vol.27 No.2.

[17] A. Pasupulati, J . Coit, K. Levitt. S. F. Wu,etc. "Buttercup: On Network-based Detection of Polymorphic Buffer Overflow Vulnerabilities", Network Operations and Management Symposium, 2004. NOMS 2004. Network Operations and Management Symposium, 2004. NOMS 2004 …, 2004 - ieeexplore.ieee.org.

[18] Ke Wang, Salvatore J. Stolfo. "Anomalous Payload-based Network Intrusion Detection". In Symposium on Recent Advances in Intrusion Detection, Sophia Antipolis.", In RAID, Sept 2004.

[19] Thomas Toth and Christopher Kruegel. "Accurate Buffer Overflow Detection via Abstract Payload Execution." RAID International Symposium on Recent Advances in Intrusion Detection 2002.

[20] C. Kreibich and J. Crowcroft. Honeycomb – creating intrusion detection signatures using honeypots. In Proceedings of the Second Workshop on Hot Topics in Networks (HotNets- II), November 2003.

[21] H.-A. Kim and B. Karp. Autograph: toward automated, distributed worm signature detection. In Proceedings of the 13th USENIX Security Symposium, August 2004.

[22] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In Proceedings of the 6th ACM/USENIX Symposium on Operating System Design and Implementation (OSDI), Dec. 2004.

[23] Zhichun Li, Manan Sanghi, Yan Chen, Ming-Yang Kao and Brian Chavez. Hamsa: Fast Signature Generation for Zeroday Polymorphic Worms with Provable Attack Resilience.In Proc. of the IEEE Symposium on Security and Privacy,Oakland, CA, May 2006.

[24] P hkritidis，E P Markatos，etc。Stride：Polymorphic sled detection through instruct ion sequence analysis．In：Proceedings of the 20th IFIP International Information Security Conference(IFIP / SEC)．Makuhari Messe，Chiba，JAPAN：May 2005．375-392

[25] Robert Moskovitch, Nir Nissim, Dima Stopel, Clint Feher,Roman Englert, and Yuval Elovici, Improving the Detection of Unknown Computer Worms Activity Using Active Learning, J. Hertzberg, M. Beetz, and R. Englert (Eds.): KI 2007, LNAI 4667, pp. 489–493, 2007.

[26] David Brumley, James Newsome, and Dawn Song, Hao Wang and Somesh Jha, Towards Automatic Generation of Vulnerability-Based Signatures

[27] J. Crandall, Z. Su, S. F. Wu, and F. Chong. On deriving unknown vulnerabilities from zero-day polymorphic and metamorphic worm exploits. In Proc. 12th ACM Conference on Computer and Communications Security (CCS), 2005.

[28] Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishnan. "Fast Portscan Detection Using Sequential Hypothesis Testing". In Proceedings of the IEEE Symposium on Security and Privacy, May 2004.

[29] S. E. Schechter, J. Jung and A.W. Berger, "Fast Detection of Scanning Worm Infections", In Proceedings of the seventh International Symposium on RAID, September 2004.

[30] 廖明涛，张德运，侯琳，基于失败连接流量偏离度的蠕虫早期检测方法，计算 机 工程 Computer Engineering, 2006 年 8 月，第 32 卷第 15 期

[31] K. Wang, S. J. Stolfo, Anomalous Payload-based Network. Intrusion Detection, RAID 2004.

[32] Like Zhang, Gregory B. White. An Approach to Detect Executable Content for Anomaly Based Network Intrusion Detection. IEEE International Parallel and Distributed

Processing Symposium 2007.

[33] D. Wagner and D. Dean. Intrusion detection via static analysis. In Proceedings of the 2001 IEEE Symposium on Security and Privacy, pages 156–168, May 2001.

[34] Mohammad M. Masud Latifur Khan Bhavani Thuraisingham. A Hybrid Model to Detect Malicious Executables. Communications, 2007. ICC '07. IEEE International Conference.

[35] Ikkyun Kim, Koohong Kang, YangSeo Choi,Daewon Kim, Jintae Oh, and Kijun Han. A Practical Approach for Detecting Executable Codes in Network Traffic.

[36] Ramkumar Chinchani and Eric van den Berg. A Fast Static Analysis Approach To Detect Exploit Code Inside Network Flow. Proceedings of the International Symposium on RAID, 2005 – Springer.

[37] Thomas Toth ,Christopher Kruegel .Accurate Buffer Overflow Detection via Abstract Payload Execution.2002-08-Distributed Systems Group, Technical University of Vienna.

[38] P. Akritidis, E. P. Markatos, M. Polychronakis, and K. Anagnostakis. STRIDE: POLYMORPHIC SLED DETECTION THROUGH INSTRUCTION SEQUENCE ANALYSIS. 20th IFIP International Information Security Conference, 2005 - i2r.a-star.edu.sg.

[39] M. Polychronakis, K. G. Anagnostakis and E. P. Markatos, "Network-Level Polymorphic Shellcode Detection Using Emulation", In Journal in Computer Virology, vol. 2, no. 4, pp. 257-274, February 2007.

[40]Q. Zhang, D. S. Reeves , Peng Ning and S. Purushothaman Iyer, "Analyzing Network Traffic To Detect Self-Decrypting Exploit Code", In Proceedings of 2007 ASIACCS, March 2007.

[41]M. Polychronakis, K. G. Anagnostakis and E. P. Markatos, "Emulation-based Detection of Non-self-contained Polymorphic Shellcode", In Proceedings of the 10th International Symposium on RAID, September 2007.

[42]许长伟 ,向继 ,鲍春杰 ,荆继武. 蠕虫软件仿真技术研究与进展.

[43] Perdisci, R., Dagon, D., Lee, W., Fogla, P. and Sharif, M. (2006), Misleading worm signature generators using deliberate noise injection", Proceedings of the 2006 IEEE Symposium on Security and Privacy, Berkeley, CA, May 21-24.

[44] Kreibich, C. and Crowcroft, J. (2003), "Honeycomb: creating intrusion detection signatures using honeypots", Proceedings of the 2nd Workshop on Hot Topics in Networks (HotNets-II), Cambridge, MA, November.

[45] Kim, H.-A. and Karp, B. (2004), "Autograph: toward automated, distributed worm signature detection", Proceedings of the 13th USENIX Security Symposium, San Diego, CA, August.

[46] Singh, S., Estan, C., Varghese, G. and Savage, S. (2004), "Automated worm fingerprinting", Proceedings of the 6th ACM/USENIX Symposium on Operating System Design and Implementation (OSDI), San Francisco, CA, December.

[47] Newsome, J., Karp, B. and Song, D. (2005), "Polygraph: automatically generating signatures for polymorphic worms", paper presented at the IEEE Security and Privacy Symposium, Oakland, CA.

[48] Li, Z., Sanghi, M., Chen, Y., Kao, M.-Y. and Chavez, B. (2006), "Hamsa: fast signature generation for zero-day polymorphic worms with provable attack resilience", Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, May.

[49] Cavallaro, L., Lanzi, A., Mayer, L. and Monga, M. (2008), "LISABETH: automated content-based signature generator for zero-day polymorphic worms", Proceedings of the 4th International Workshop on Software Engineering for Secure Systems (SESS'08), Leipzig, Germany, May.

[50] Toth, T. and Kruegel, C. (2002), "Accurate buffer overflow detection via abstract payload execution", Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID'02), Zurich October, pp. 274-91.

[51] Perdisci, R., Dagon, D., Lee, W., Fogla, P. and Sharif, M. (2006), "Misleading worm signature generators using deliberate noise injection", Proceedings of the 2006 IEEE Symposium on Security and Privacy, Berkeley, CA, May 21-24.

[52] Newsome, J., Karp, B. and Song, D. (2006), "Paragraph: thwarting signature learning by training aliciously", Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID'06), Hamburg, September.

[53] Chung, S.P. and Mok, A.K. (2006), "Allergy attack against automatic signature generation", Proceedings of 9th International Symposium on Recent Advances in Intrusion Detection (RAID'06), Hamburg, September, pp. 61-80.

[54] Costa, M., Crowcroft, J., Castro, M. and Rowstron, A. (2005), "Vigilante: end-to-end containment of internet worms", Proceedings of the 20th ACM Symposium on Operating Systems Principles, Brighton, UK, pp. 133-47.

[55] Andersson, S., Clark, A. and Mohay, G. (2005), "Detecting network-based obfuscated code injection attacks using sandboxing", Proceedings of the AusCERT Asia Pacific Information Technology Security Conference (AusCERT'05): Refereed R&D Stream, Gold Coast, Australia, University of Queensland, Queensland.

[56] Detristan, T., Ulenspiegel, T., Malcom, Y. and Underduk, M. (2003), "Polymorphic shellcode engine using spectrum analysis", Phrack, Vol. 11 No. 61.

[57] Kolesnikov, O., Dagon, D. and Lee, W. (2004), "Advanced polymorphic worms: evading IDS by blending in with normal traffic", available at: www.cc.gatech.edu/~ok/w/ok_pw.pdf

[58] Eller, R. (2003), "Bypassing msb data filters for buffer overflow exploits on intel platforms", available at: http://community.core-sdi.com/juliano/bypass-msb.txt

[59] van Gundy, M., Balzarotti, D. and Vigna, G. (2007), "Catch me, if you can: evading network signatures with web-based polymorphic Automated worm fingerprinting's", Proceedings of the 1st Conference on First USENIX Workshop on Offensive Technologies, Boston, MA, August 6, p. 7.

[60] Paxson, V. (1999), "Bro: a system for detecting network intruders in real-time", Computer Networks, Vol. 31 Nos 23/24, pp. 2435-63.

攻硕期间取得的成果

1．发表论文情况

Chen Da-peng, Zhang Xiao-song. Internet Anomaly Detection with Weighted Fuzzy Matching over Frequent Episode Rules (ICICA 2008. EI 检索)

2．科研工作情况

华为基金： 基于 Windows 的恶意软件行为蜜罐分析研究

华为基金： 基于蠕虫动态行为和统计特征的检测方法研究与验证

四川省科技攻关项目：基于多层代理的分布式联动入侵检测系统

3．发明专利

申请号 200810044620.X (名称：一种网络攻击检测方法)

申请号 200810044620 (名称：一种基于频繁片段规则的网络入侵检测方法)

基于异常用户行为的蠕虫检测与特征码自动提取技术研究

作者： 陈大鹏
学位授予单位： 电子科技大学