恶意软件分类方法研究*

卢 浩[1,2]，胡华平[1]，刘 波[1]

(1.国防科学技术大学 计算机学院 网络与信息安全研究所，湖南 长沙 410073；2.空军雷达学院 训练部 教育技术中心，湖北 武汉 430019)

摘 要：在详细论述恶意软件的工作原理、运行机制的基础上，对各类恶意软件进行系统的分类，阐述各类恶意软件间的关系，归纳恶意软件的发展趋势并给出阻断恶意软件的方法。

关键词：信息安全；恶意软件；防火墙；入侵检测系统；混合型攻击

中图法分类号：TP393.03 文献标识码：A 文章编号：1001-3695(2006)09-0004-04

Research on Classification of Malware

LU Hao[1,2], HU Hua-ping[1], LIU Bo[1]

(1. *Institute of Network & Information Security, College of Computer Science, National University of Defense Technology, Changsha Hunan* 410073, *China*; 2. *Centre of Education Technology, Dept. of Training, Air Force Radar Academy, Wuhan Hubei* 430019, *China*)

Abstract: Basing on the analyse on operating mechanism of malware, various malware were classified clearly. After expanding on the relationship between these malware, the trends in evolvement of the malware were concluded. Finally the method to block malware was proposed.

Key words: Information Security; Malware; Firewall; IDS; Blended Attack

随着网络的广泛普及和应用，网络环境下多样化的传播途径和复杂的应用环境给恶意软件(Malware)的传播带来巨大便利，从而对网络系统及网络上主机的安全构成巨大威胁。

恶意软件是非用户期望运行的、怀有恶意目的或完成恶意功能的软件的统称[1,2]。恶意代码(Malicious Code)含义与恶意软件相近，区别在于所描述的粒度不同。恶意代码用于描述完成特定恶意功能的代码片段，而恶意软件则指完成恶意功能的完整的程序集合。

1 恶意软件的分类、原理

恶意软件的种类较多，如 Virus(病毒)，Worm(蠕虫)，Bot，Trojan Horse(特洛伊木马)，Exploit(漏洞利用程序)，Backdoor(后门)，Rootkit，Spyware(间谍软件)，Spamware(垃圾信息发送软件)，Adware(垃圾广告软件)等[1,3]。在对恶意软件的名称引用上存在一些混用、指代不明的情况，如将所有导致计算机工作不正常的恶意软件都称为病毒，将未授权隐藏在计算机内运行的恶意软件都称为(特洛伊)木马等。本文对恶意软件进行系统的分类，以明确、规范各类恶意软件的概念，区分其间的差异。

入侵目标系统并达到特定目的的完整过程包含三个阶段：①获取对目标系统的远程控制权；②维持对目标系统的远程控制权；③通过远程控制在目标系统上完成特定业务逻辑。这里所说的目标系统可以是目标主机系统，也可以是目标网络设备。

根据不同恶意软件所完成的功能在完整的入侵过程中所处阶段的不同，我们将恶意软件分为三种类型。

1.1 获取目标系统远程控制权类

获取目标系统远程控制权类恶意软件的基本特征是在未授权的条件下，利用各种手段获取对目标系统的远程控制权的功能，即具有完整入侵过程中第一阶段的功能。

(1) Exploit(漏洞利用程序)。它是第一类恶意软件的基本形式。Exploit 利用操作系统或应用程序中存在的缺陷(Bug)，可达到以非授权的方式远程控制目标系统或提升本地用户权限的目的[4,5]。具体过程为：构造特定的输入数据并提交给存在缺陷的操作系统程序或应用程序，使这些有缺陷的程序在所构造的输入数据下，正常的程序流程发生改变，从而导致未授权用户可以远程控制目标系统，或使本地用户获得更高的权限。

使用 Exploit 获取目标系统的远程控制权后，入侵过程中后续的维持控制权、完成特定业务逻辑的工作均与 Exploit 无关。典型的 Exploit 类恶意软件有 Buffer Overflow Exploit(缓冲区溢出漏洞利用工具)，SQL Injection Exploit(SQL 注入工具)等。当操作系统或应用程序的源代码行数达到一定规模后，程序中存在缺陷是不可避免的，所以各种漏洞层出不穷，Exploit 类恶意软件也日益增多。因此可以为操作系统或应用程序打上相应缺陷的 Patch(补丁)，恶意软件则无法利用 Exploit 对目标系统进行远程控制。

(2) Trojan Horse(特洛伊木马)，简称为 Trojan(木马)。它是伪装成合法程序以欺骗用户执行的一类恶意软件[1,6,7]。使用 Trojan 入侵目标系统的具体过程为：Trojan 首先通过网络或各种存储介质传播到用户处，因其具有伪装、欺骗性，常被经验不足或防范意识较差的用户执行后，释放出其携带的 Backdoor

收稿日期：2005-07-29；修返日期：2005-09-23

基金项目：国家“863”计划资助项目(2003AA142010)

(后门, 概念见第 1.2 节(1)) 以实现对目标主机的远程控制[9], 在必要时还可释放出其携带的 Exploit 以提升用户权限。

从使用 Trojan 实施入侵的过程可以看出, 与 Exploit 仅包含获取远程控制权的功能不同, 除包含用于欺骗用户在目标系统上执行的 Trojan Header(木马头部) 之外, Trojan 还包含 Backdoor, Exploit 等 Payload (载荷), 这些 Payload 在 Trojan 执行后被释放出来, 以维持对目标系统的远程控制并完成特定业务逻辑[7]。除通过可执行文件复制、执行来传播的 Trojan 外, 更具欺骗性的 Trojan 类型有邮件附件 Trojan、网页恶意代码 Trojan、宏病毒 Trojan, 这几种 Trojan 往往在用户毫不知情的情况下就被执行了, 危害非常大。

(3) Worm(蠕虫)。它具有自我繁殖能力, 无需用户干预便可自动在网络环境中传播的一类恶意软件[1,6]。Worm 利用目标系统的 Weak Password(弱口令) 或目标系统中存在的缺陷获得对目标系统的远程控制权, 并搜集目标系统内的相关信息从而将 Worm 自身传染至与目标系统有网络联系的其他系统。Worm 自身的存在有两种形式, 即可执行文件的形式和内存中进程/线程的形式。当以进程/线程的形式存在时, 传播过程中 Worm 在目标系统内不涉及文件操作, 具有更强的隐蔽性。与 Trojan 类似, Worm 也包含 Payload 部分, 以便在成功入侵目标系统后完成特定的业务逻辑[8], 如 CodeRed Worm, 其 Payload 部分的功能是对白宫 Web 服务器进行 DoS(Denial of Service, 拒绝服务) 攻击。

(4) Bot。Bot 的概念与 Worm 相近。若某 Worm 的 Payload 部分包含一个 Backdoor, 则称该 Worm 为 Bot, 或者说可远程控制的 Worm 即为 Bot[9,10]。

Bot 比 Worm 更进一步, Worm 虽然具有自主繁殖、传播的能力, 但其传播出去之后就不再受控, 不能根据 Worm 发布者的意图调整行为方式; 而 Bot 在传播出去之后依然可以对其进行控制和调整。由此可见, Bot 的危害性比 Worm 更大, 制作精巧的 Bot 甚至可以根据发布者的意图对其 Payload 部分进行升级以适应新的环境或完成新的功能。

被 Bot 成功入侵后的受控主机即为 Zombie(傀儡主机), 一组 Zombie 则称为 Botnet[10]。当前 Internet 上的主要安全威胁之一的 DDoS(Distributed Denial of Service, 分布式拒绝服务) 攻击就是通过向 Botnet 发出针对特定目标的 DoS 攻击命令来完成的。

(5) Virus(病毒)。它是依附于宿主文件, 在宿主文件被执行的条件下跟随宿主文件四处传播并完成特定业务功能的一类恶意软件[1,6]。Virus 的结构与 Worm 相似, 除包含用于传播自身的 Virus Header(病毒头部) 外, 还包含用于完成特定业务逻辑的 Payload 部分。

Virus 和 Worm 是容易混淆的两个概念。两者主要区别在于: ①Worm 的传播无需宿主文件, 可通过网络直接将 Worm 自身传播到目标系统; 而病毒传播需要宿主文件, 病毒只能寄生在宿主文件中。②Worm 的繁殖、传播无需人工干预, 由 Worm 自主、自动完成; 病毒的传播需要人工干预。首先, 病毒的传播依赖于宿主文件的位置改变, 宿主文件到哪里, 病毒才可能传播到哪里。其次, 若宿主文件未被执行, 则寄生其中的病毒便不会感染目标系统[11]。

(6) 第一类恶意软件对比如表 1 所示。

表 1　第一类恶意软件对比

病　毒　种　类	Exploit	Trojan	Worm	Bot	Virus
获取目标系统控制权的能力	√	√	√	√	√
能否包含 Payload		√	√	√	√
感染目标系统模式	主动	被动	主动	主动	被动

注: 被动感染目标系统模式, 即需要目标系统用户执行外来程序才能获取对目标系统的远程控制权; 主动感染目标系统模式, 即获取对目标系统的远程控制权, 无需目标系统用户的任何动作。

1.2　维持远程控制权类

获取对目标系统的远程控制权后, 在目标系统中运行此类恶意软件以维持对目标系统的远程控制权。此类恶意软件用于完整入侵过程中的第二阶段。

(1) Backdoor(后门)。它是一类运行在目标系统中, 用以提供对目标系统未经授权的远程控制服务[1,7] 的恶意软件。需要注意的是, Backdoor 与第一类恶意软件不同, Backdoor 的作用是通过其运行以提供对目标系统未经授权的远程控制的服务; 而第一类恶意软件需要利用各种手段来达到此目的, 即 Backdoor 必须在目标系统上运行才能提供相应的服务, 因此必须先使用第一类恶意软件以获得在目标系统上执行程序的权限。第一类恶意软件是 Backdoor 发挥作用的前提和基础, Backdoor 运行后, 后续对目标系统的远程控制均通过 Backdoor 提供的服务来完成。

(2) Rootkit。其概念起源于 UNIX/Linux 操作系统, 最初是指 UNIX/Linux 系统中一组用于获取并维持 Root 权限的工具集。发展至今日, 被广为接受的 Rootkit 概念是指用于帮助入侵者在获取目标主机管理员权限后, 尽可能长久地维持这种管理员权限的工具[4]。在当前的 Rootkit 概念中, 获取管理员权限的过程不由 Rootkit 来完成, 即 Rootkit 的使用基于已经获得了管理员权限的假设。

由 Backdoor 的概念可知, Backdoor 仅提供了一条非授权访问、控制目标系统的 "通道", 但并不涉及对这条通道的保护, 因此这条通道很容易被目标系统上的管理员或网络安全设备察觉或检测到。Rootkit 的作用是要尽可能长久地维持对目标系统的远程控制, 故其基本任务就是要隐藏 Backdoor 所提供的通道, 尽可能使得目标系统上的管理员或安全设备不能察觉、检测到该通道的存在。

当前主流操作系统平台下的 Rootkit 已经比较成熟, 内核级的 Rootkit 能做到对操作系统中的进程、线程、网络连接、网络数据、网络通信目的地的深度隐藏[4], 以保护目标系统中运行的恶意软件不被检测到。在实际应用中, Rootkit 通常直接包含了 Backdoor 的功能。因此, 可将 Rootkit 理解为带隐藏功能的 Backdoor。

(3) 第二类恶意软件对比如表 2 所示。

表 2　第二类恶意软件对比

恶　意　软　件　种　类	Backdoor	Rootkit
提供非授权的远程访问控制通道	√	√
隐藏自身的能力		√
隐藏远程访问控制通道的能力		√
隐藏目标系统中运行的其他恶意软件		√

1.3　完成特定业务逻辑类

第三类恶意软件用于完成入侵目标系统最终所要进行的操作, 如窃取情报、破坏系统、发动攻击、中转数据等。第一类

和第二类恶意软件的作用在于为第三类恶意软件提供一个安全、便捷的运行平台。

(1) Spyware(间谍软件)[1,3]。它是典型的第三类恶意软件,用于从目标系统中收集各种情报、信息,如商业、军事情报,用户信用卡号、个人隐私信息/文档,各种网站/邮箱用户名、口令等信息。收集到这些信息后,Spyware 通过网络将其发送给入侵者。在 Rootkit 的保护下,Spyware 本身以及 Spyware 所产生的网络通信都被隐藏,使得 Spyware 可在目标系统中安全地存活下来。

Spyware 在目标系统中的运行途径主要有三种:①将 Spyware 放在 Worm/Bot/Virus/Trojan 的 Payload 中,在 Worm/Bot/Virus/Trojan 执行时被释放出来并执行;②利用 Exploit 获取对目标系统远程控制权后,将 Spyware 通过网络传输到目标主机并执行;③在目标系统上安装并运行 Rootkit/Backdoor 后,通过其提供的远程控制服务将 Spyware 传输到目标主机并执行。

(2) Spamware(垃圾信息发送软件)。为了避免被追查,Spam(非期望的垃圾信息,如垃圾邮件)的发送者通常不会直接使用自己的主机发送垃圾信息。为了加大垃圾信息的发送范围,仅仅用一台主机发送垃圾信息是不够的。Spamware(垃圾信息发送软件)[1,3]就是运行在大量被入侵主机中用于发送垃圾信息的恶意软件,Spamware 在目标系统中的运行途径与 Spyware 类似。

(3) Adware(垃圾广告软件)[1,3]。它运行在被入侵主机中,用于以各种方式显示垃圾广告的恶意软件。Adware 在目标系统中的运行途径与 Spyware 类似。

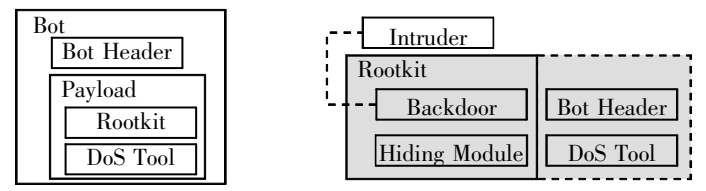
(4) 其他第三类恶意软件。其种类很多,根据具体应用需求不同而不同,如对目标系统进行攻击/破坏的恶意软件有多种不同的类型,但目前尚无统一规范的命名。

2 各类恶意软件间的关系与运行机制

第一类恶意软件和第二类恶意软件是第三类恶意软件发挥作用的前提,它们为第三类恶意软件提供并维持对目标系统的未授权远程控制权,入侵最终要实现的功能由第三类恶意软件完成。

第一类恶意软件中,除 Exploit 外,Trojan, Worm, Bot, Virus 均可包含 Payload 部分。故除实现获取对目标系统的远程控制权的功能外,Trojan, Worm, Bot, Virus 还可将第二类恶意软件及第三类恶意软件包含在 Payload 中,在执行后释放出来。

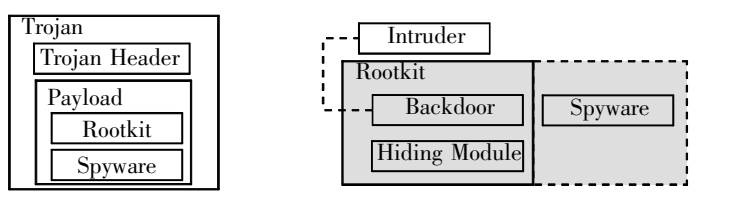
一个 Payload 部分包含 Rootkit 和 DoS 工具的典型 Bot 的静态结构以及入侵目标系统后在内存中的结构,如图 1 所示。图中 Bot 的静态结构表示 Bot 在执行前的结构组成。由图 1(a)可见,静态时 Bot 由 Bot Header 和 Payload 组成,其中 Payload 由 Rootkit 和 DoS Tool 组成。Bot 成功入侵目标系统后,入侵者通过 Rootkit 内包含的 Backdoor 来远程控制目标系统,Bot Header 继续在目标系统内运行以便将 Bot 传播到其他系统中;Rootkit 内包含的 Hiding Module(隐藏模块)将 Rootkit 自身及 Bot Header, DoS Tool 都隐藏起来以躲避检测,DoS Tool 在入侵者的远程控制下实施对特定目标的攻击。图 1(b)中深色背景部分表示目标系统中被 Rootkit 隐藏起来,从而检测不到的内容。



(a) Bot 静态结构 (b) Bot 入侵目标系统后在内存中的结构

图 1 Bot 静态结构及入侵目标系统后在内存中的结构示意图

Payload 部分包含 Rootkit 和 Spyware 的一个典型 Trojan 的静态结构及入侵目标系统后在内存中的结构,如图 2 所示。由图可知,Trojan 欺骗用户执行从而入侵目标系统后,其 Payload 部分被释放出来并运行于目标系统中。因 Trojan 对目标系统的感染采用的是被动模式,故 Trojan Header 在欺骗用户执行后即已完成其任务,并不存在于目标系统内存中。常被混用的两个概念是木马(Trojan)和 Rootkit。根据前文分析可知,Trojan 的主要特征在于其伪装性、欺骗性,通过 Trojan Header 欺骗用户执行;而 Rootkit 的主要特征在于隐藏。



(a) Trojan 静态结构 (b) Trojan 入侵目标系统后在内存中的结构

图 2 Trojan 静态结构及入侵目标系统后在内存中的结构示意图

当前威胁非常大的一类被称为 Blended Attck(混合型攻击)[12]的攻击方式,它综合运用了各类恶意软件的相关技术,使得攻击效能最大化。①在获取目标系统远程控制权阶段,将 Trojan, Worm, Bot, Virus 的入侵手段结合到一个执行程序中,实现主动入侵和被动入侵的结合,从而显著增强其对目标系统的入侵能力。②在维持对目标系统的远程控制权阶段,充分运用各项成熟的 Rootkit 相关技术,使得 Rootkit 及其保护下的其他恶意软件能可靠、长久地存活于目标系统中。③在完成特定业务逻辑阶段,集多种第三类恶意软件的功能于一身,这样既可以搜集情报,又可以实施破坏,或将目标系统作为防止反向网络追踪的数据中转站。

本文提出的对恶意软件的分类方法是恶意软件的基本组成单元,在具体应用中,各类恶意软件的合理、综合运用,使得当前恶意软件呈现出形式多样、功能日益强大的状况。

3 恶意软件发展趋势

随着各类恶意软件相关技术的日益成熟,当前恶意软件的发展出现了一些值得关注的新趋势,了解和把握这些趋势将有利于有针对性地制定相应的防护措施。

3.1 由任意传播转向定量传播

在 2004 年之前的数年间,Worm 和 Bot 在传播时采用的主要是任意传播策略,对于其搜索到的任何可利用、入侵的目标系统都进行感染,将自身传播到目标系统中[13~16]。根据多家安全服务公司及防病毒软/硬件厂商的监控数据[16,17],自 2004 年 5 月至今,已极少监测到类似前几年的 CodeRed Worm, Slammer Worm 等大规模传播的 Worm/Bot,而恶意软件的发展势头在近几年特别是近一年中更加迅猛,因恶意软件导致的各种安全问题发生频率越来越高。综合防病毒软/硬件厂商对相

关数据的分析结果表明,当前的 Worm/Bot 传播策略发生了明显的调整,已由任意传播策略转向定量传播策略,即 Worm 和 Bot 在传播过程中,仅感染满足特定条件的目标系统,且当成功感染目标系统的数量达到一定值后,则不再继续传播。根据 Kaspersky Labs 的统计分析,这一临界数量在 5 000 ～10 000[17] 之间。

任意传播策略的特征是传播速度快、被感染的目标系统数量大。这两个特征导致了以任意传播策略传播的 Worm/Bot 很容易就被防病毒软/硬件厂商采集到 Worm/Bot 的样本,进而提取其 Signature(特征码)并更新其病毒库,从而使得普通用户能及时、有效地清除这些 Worm/Bot 并阻止其在网络上的继续传播。

定量传播策略的特征是传播速度快,但被感染的目标系统数量相对而言非常少。因此,基于此策略传播的 Worm/Bot 很难被防病毒软/硬件厂商采集到样本,安装了防病毒软/硬件系统的普通用户即使及时更新病毒库,也无法查出系统中已经存在的 Worm/Bot,无法过滤进入系统中的 Worm/Bot。

3.2　存活能力显著提高

(1)在传播过程中,通过使用 Polymophism(变形)[18,19] 技术,经处理后的恶意软件不再具有固定的 Signature,因此可有效对抗当前防病毒软/硬件的过滤和查杀。精心编写的恶意软件甚至可以将 Polymophism 引擎包含在其自身内部,使得其每次传播都具有不同的 Signature。

(2)利用 Rootkit 实现对进程、线程、通信连接、通信数据、通信目的地的隐藏,同时结合 Firewall/IDS Bypass(防火墙旁路)、Firewall/IDS Disable(禁用防火墙/入侵检测系统)、降低系统安全等级设置等技术,可有效防止在被入侵的目标系统内运行着的恶意软件被检测到。

综合运用以上措施后,当前恶意软件能有效躲避当前广泛应用的、传统的、基于 Border Firewall + DMZ IDS + Personal Firewall, Antivirus Software 结构的 Defense in Depth(深度防御)体系[20] 的检测,存活能力显著提高。

3.3　应对基于系统缺陷的攻击响应时间迅速缩短

Zero-Day Exploit[21] 是指缺陷被发现后立即出现的漏洞利用程序。Zero-Day Exploit(0-Day Exploit)的不断涌现[5] 标志着发现系统缺陷到漏洞利用程序出现的间隔时间比以往大大缩短,从而导致应对基于系统缺陷的攻击的响应时间迅速缩短。当出现 Zero-Day Exploit 时,软件厂商往往尚未发布相应补丁,即使有补丁发布,多数用户也无法保障在补丁出现时立即给系统打上补丁,从而导致大量有缺陷的系统暴露于 Zero-Day Exploit 的威胁之下。

3.4　传播途径多样化

众多 P2P(Peer-to-Peer,点到点)模式以及 IM(Instant Messaging,即时通信)类网络应用程序的迅猛发展,为恶意软件提供了更便捷、更快速的传播途径。

3.5　更易用、功能更强大

当前恶意软件的发展使得入侵者只需很少的专业知识即可掌握其用法。同时,综合运用各种恶意软件相关技术的混合型恶意软件的功能越来越强大,一旦入侵成功,即可实现对目标系统的完全控制。

4　结论与展望

应对当前恶意软件的快速发展与传播对传统的 Defense in Depth 防御体系带来的巨大挑战,本文认为在保留传统的主要基于 Signature 扫描方式的防御体系的前提下,迫切需要采取以下两方面的措施:①从阻断 Exploit, Worm, Bot 这些主动入侵型恶意软件的角度入手,大力加强 IPS(Intrusion Prevention System,入侵阻止系统)的研发,将恶意软件阻挡于系统之外;②从阻断 Trojan, Virus 这些被动入侵型恶意软件的角度入手,堵住恶意软件得到执行权限的源头[22],如安装 Application Firewall(应用程序防火墙)[23],严格控制不明可执行文件的执行动作。综合以上措施,则能有效阻止恶意软件从外、从内进入目标系统。

参考文献:

[1] d Skoudis, Lenny Zeltser. Malware: Fighting Malicious Code[M]. US: Prentice Hall PTR, 2003.16-18.

[2] Gary McGraw, Greg Morrisett. Attacking Malicious Code: A Report to the Infosec Research Council[J]. IEEE Software, 2000, 17(5):1-2.

[3] Spyware, Adware, and Malware[EB/OL]. http://www.cs.umn.edu/help/security/spyware.php, 2005-06.

[4] Greg Hoglund, Gary McGraw. Exploiting Software: How to Break Code[M]. US: Addison Wesley, 2004.60-81.

[5] Jon Erickson. Hacking: The Art of Exploitation[M]. US: No Starch Press, 2003.48-54.

[6] Harold Thimbleby. A Framework for Modelling Trojans and Computer Virus Infection[J]. Computer Journal, 1998, 41(7): 444-458.

[7] Lenny Zeltser. Reverse Engineering Malware[EB/OL]. http://www.zeltser.com/reverse-malware-paper/, 2005-06.

[8] Tom Chen. Trends in Viruses and Worms[J]. Internet Protocol Journal, 2003, 6(3):8-12.

[9] Laurianne McLaughlin. Bot Software Spreads, Causes New Worries[J]. IEEE Distributed Systems Online, 2004, 5(6):3-5.

[10] Bill Mccarty. Botnets: Big and Bigger[J]. IEEE Security and Privacy, 2003, 1(4):87-80.

[11] What is the Difference between Viruses, Worms, and Trojans? [EB/OL]. http://service1.symantec.com/SUPPORT/nav.nsf/docid/1999041209131106, 2005-06.

[12] Eric Chien, Péter Szr. Blended Attacks Exploits, Vulnerabilities and Buffer-Overflow Techniques in Computer Viruses[C]. Virus Bulletin Conference, 2002.2-15.

[13] Michele Garetto, Weibo Gong, Don Towsley. Modeling Malware Spreading Dynamics[C]. IEEE INFOCOM, 2003.1869-1871.

[14] Lawrence A Gordon, Martin P Loeb. 2004 CSI/FBI Computer Crime and Security Survey[EB/OL]. http://www.gocsi.com/, 2005-06.

[15] Nikolai Joukov. Internet Worms as Internet-Wide Threat[EB/OL]. http://www.ecsl.cs.sunysb.edu/tech_reports.html, 2005-06.

[16] Ferdinand Gomes. Internet Security Threat Report: Malicious Code Trends[EB/OL]. http://enterprisesecurity.symantec.com/content.cfm?articleid=1539, 2005-06.

[17] Munir Kotadia. Stealth Virus Warning Sounded Again[EB/OL]. http://news.zdnet.co.uk/internet/0,39020369,39199961,00.htm, 2005-06.

(下转第 12 页)

[C]. The 17th International Joint Conference on Artificial Intelligence, 2001. 891-896.

[23] Knobbe J, Blockeel H, Siebes A, *et al*. Multi-Relational Data Mining[C]. Proceedings of Benelearn '99, 1999.

[24] Knobbe J, Siebes A, Vander Wallen, *et al*. Multi-Relational Decision Tree Induction[C]. Proceedings of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD '99, 1999.

[25] Blockeel H, De Raedt L. Relational Knowledge Discovery in Databases[C]. Proceedings of the 6th International Workshop on Inductive Logic Programming, volume 1314 of Lecture Notes in Artificial Intelligence, Springer-Verlag, 1996. 199-211.

[26] A Inokuchi, T Washio, H Motoda. Complete Mining of Frequent Patterns from Graphs: Mining Graph Data[J]. Machine Learning, 2003, 50(3): 321-354.

[27] R Michalski, I Mozetic, J Hong, *et al*. The Multi-purpose Incremental Learning System AQ15 and Its Testing Application on Three Medical Domains[C]. Proceedings of the 5th National Conference on Artificial Intelligence, San Mateo: Morgan Kaufmann, 1986. 1041-1045.

[28] E Segal, Y Barash, I Simon, *et al*. From Promoter Sequence to Expression: A Probabilistic Framework[C]. Washington, DC: Proceedings of the 6th International Conference on Research in Computational Molecular Biology(RECOMB-02), 2002. 263-273.

[29] E Segal, A Battle, D Koller. Decomposing Gene Expression into Cellular Processes[A]. R B Altman, A K Dunker, L Hunter, *et al*. Proceedings of the Pacific Symposium on Biocomputing[M]. Kauai: World Scientific, 2003. 89-100.

[30] S Muggleton. Inductive Logic Programming[J]. New Generation Computing, 1991, 8(4): 295-318.

[31] S Muggleton. Inductive Logic Programming[M]. London: Academic Press, 1992.

[32] S Muggleton. Inverse Entailment and Progol[J]. New Generation Computing, 1995, 13(3&4): 245-286.

[33] S Muggleton, C Feng. Efficient Induction of Logic Programs[C]. Ohmsha: Proceedings of the 1st Conference on Algorithmic Learning Theory, 1990. 368-381.

[34] C Nedellec, C Rouveirol, H Ade, *et al*. Declarative Bias in Inductive Logic Programming[A]. L De Raedt. Advances in Inductive Logic Programming[M]., Amsterdam: IOS Press, 1996. 82-103.

[35] T Niblett. A Study of Generalization in Logic Programs[C]. Pitman: Proceedings of the 3rd European Working Session on Learning, 1988. 131-138.

[36] S H Nienhuys-Cheng, R de Wolf. Foundations of Inductive Logic Programming[M]. Berlin: Springer, 1997.

[37] G Plotkin. A Note on Inductive Generalization[A]. B Meltzer, D Michie. Machine Intelligence[M]. Edinburgh: Edinburgh University Press, 1969. 153-163.

[38] J R Quinlan. Learning Logical Definitions from Relations[J]. Machine Learning, 1990, 5(3): 239-266.

[39] J R Quinlan. C4.5: Programs for Machine Learning[M]. San Mateo: Morgan Kaufmann, 1993.

[40] R Agrawal, R Srikant. Mining Sequential Patterns[C]. Proceedings of the 11th International Conference on Data Engineering, Los Alamitos: IEEE Computer Society Press, 1995. 3-14.

[41] R Agrawal, H Mannila, R Srikant, *et al*. Fast Discovery of Association Rules[A]. U Fayyad, G Piatetsky-Shapiro, P Smyth, *et al*. Advances in Knowledge Discovery and Data Mining[M]. Menlo Park: AAAI Press, 1996. 307-328.

[42] H Blockeel, L De Raedt. Top-down Induction of First Order Logical Decision Trees[J]. Artificial Intelligence, 1998, 101(1-2): 285-297.

[43] I Bratko. Prolog Programming for Artificial Intelligence(3rd edition)[M]. Harlow: Addison-Wesley, 2001.

[44] L Breiman, J H Friedman, R A Olshen, *et al*. Classification and Regression Trees[M]. Belmont: Wadsworth, 1984.

[45] P Clark, R Boswell. Rule Induction with CN2: Some Recent Improvements[C]. Proceedings of the 5th European Working Session on Learning, Berlin: Springer, 1991. 151-163.

[46] P Clark, T Niblett. The CN2 Induction Algorithm[J]. Machine Learning, 1989, 3(4): 261-283.

[47] L Dehaspe, H Toivonen, R D King. Finding Frequent Substructures in Chemical Compounds[C]. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, Menlo Park: AAAI Press, 1998. 30-36.

[48] L Dehaspe, H Toivonen. Discovery of Frequent Datalog Patterns[J]. Data Mining and Knowledge Discovery, 1999, 3(1): 7-36.

[49] L De Raedt, S Dzeroski. First Order Jk-clausal Theories are PAC-learnable[J]. Artificial Intelligence, 1994, 70(1-2): 375-392.

[50] S Dzeroski, S Muggleton, S Russell. PAC-learn Ability of Determinate Logic Programs[C]. Proceedings of the 5th ACM Workshop on Computational Learning Theory, New York: ACM Press, 1992. 128-135.

[51] S Dzeroski, S Schulze-Kremer, K Heidtke, *et al*. Diterpene Structure Elucidation from 13C NMR Spectra with Inductive Logic Programming[J]. Applied Artificial Intelligence, 1998, 12: 363-383.

[52] S Dzeroski, H Blockeel, B Kompare, *et al*. Experiments in Predicting Biodegradability[C]. Proceedings of the 9th International Workshop on Inductive Logic Programming, Berlin: Springer, 1999. 80-91.

作者简介:

徐光美(1977-),**女,山东人,博士研究生,主要研究方向为数据挖掘;杨炳儒**(1943-),**男,天津人,教授,博导,主要研究方向为知识发现与推理机制、柔性建模与系统集成;张伟**(1974-),**男,山东人,博士研究生,主要研究方向为数据挖掘、数据库;宁淑荣**(1976-),**女,山西人,博士研究生,主要研究方向为人工智能。**

(上接第 7 页)

[18] aniel Wolff. BOTs and Packers: The New Polymorphism[C]. APWG North American Spring Meeting, 2005. 6-15.

[19] David M Chess, Steve R White. An Undetectable Computer Virus[C]. Proceedings of Virus Bulletin Conference, 2000. 2-5.

[20] Stephen Northcutt. Inside Network Perimeter Security[M]. US: New Riders Publishing, 2003. 531-548.

[21] Abhay Joshi. How to Protect Your Company From "Zero-day" Exploits[EB/OL]. http://www.computerworld.com/printthis/2004/0,4814,90447,00.html, 2005-06.

[22] John V Harrison. Enhancing Network Security by Preventing User-Initiated Malware Execution[C]. Proc. of the International Conference on Information Technology: Coding and Computing, 2005. 3-6.

[23] Ryan Permeh. The Use of Application Specific Security Measures in a Modern Computing Environment[EB/OL]. http://www.eeye.com/html/Research/Papers/DS20010322.html, 2005-06.

作者简介:

卢浩(1973-),**男,湖北黄陂人,讲师,硕士研究生,主要研究方向为计算机网络、信息安全;胡华平**(1967-),**男,教授,博导,主要研究方向为信息安全、密码学;刘波**(1973-),**男,助研,博士研究生,主要研究方向为计算机网络、信息安全。**