基于良性益虫的对等网络蠕虫防御技术

周世杰 秦志光 刘乐源 邓昳轶

(电子科技大学计算机科学与工程学院 成都 610054)

摘 要 对等网络蠕虫利用对等网络的固有特征(如本地路由表、应用层路由等),不仅复制快,而且提供了更好的隐蔽性和传播性,因而其危害大,防御困难。从分析互联网蠕虫及其传播机制入手,对对等网络上的蠕虫(即P2P蠕虫)及其特殊性进行了综合分析。在此基础之上,提出了基于良性益虫的被动激活主动传播防御策略(PAIFDP),并对该策略的技术原理和响应防御系统的功能模块等进行了详细设计。以Peersim仿真平台为基础,对各种不同网络参数下的防御效果和资源消耗情况进行了实验分析。结果表明,基于良性益虫的P2P蠕虫防御技术具有收敛时间快、网络资源消耗少、适应性强等特点。

关键词 蠕虫,对等网络蠕虫,对等网络,良性益虫,防御策略

中图法分类号 TP309 **文献标识码** A

Anti-worm Based Defensive Scheme for P2P Worm

ZHOU Shi-jie QIN Zhi-guang LIU Le-yuan DENG Yi-yi

(School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China)

Abstract P2P worms employ the distinctive features of P2P network, such as the local routing table, application routing mechanism and so on, to quickly distribute them into the network while holding the covert characteristic. Contrarily, the common internet worms generally rely on detecting the victims' IP address to spread. Therefore, the lack of hidden feature and feasible promulgating paths make that it is easier to detect and defense the ordinary internet worms than P2P worms. Consequently, the P2P worm can do more damage to the network if lacking the effective defensive scheme. In this paper, the P2P worm, especially its transmission mechanism was analyzed synthetically. Then, an anti-worm based scheme for the defensive of P2P worm was presented. The principle and functional modules of this new scheme were addressed as well. By using the Peersim P2P simulator, the performance of our novel scheme was evaluated experimentally in various system parameters. The primary experimental results indicated that our anti-worm based defensive scheme for P2P worm has the features of fast convergence, low overload of networking resource (including communication traffic and computing power), and high adaptability.

Keywords Worm, P2P worm, P2P network, Anti-worm, Defensive policy

1 引言

恶意代码是互联网的主要安全威胁之一,其中又以网络蠕虫的危害性最大。近年来,对等网络(P2P: Peer-to-Peer Network)在为人们网络生活带来巨大便利的同时,也成为了蠕虫传播的新温床。以Gnutella为例,一条查询消息大约经过约7跳后即可覆盖约95%的网络节点。同样,利用P2P网络进行传播的蠕虫(即P2P蠕虫)也仅需经过7代复制,即可保证足够的传播深度和广度,进而对对等网络带来致命性的破坏。传统蠕虫传播速度最大的瓶颈在于隐蔽性和散播性的协调,而P2P网络繁密的正常信息流量,正好为蠕虫提供了良好的传播途径和伪装环境,使其传播性和隐蔽性得到完美结合。此外,由于目前P2P流量已经成为Internet的主流,因此P2P蠕虫一旦在P2P网络中广泛传播,也将对Internet带来巨大的危害。

本文针对P2P蠕虫的高传播性和高隐蔽性的新特点,提出了一个基于良性益虫的被动激活主动传播的防御策略(Passive Activating Initiative Flooding Defense Policy: PAIFDP)。该防御策略充分结合了被动防御策略低资源消耗性、蜜罐技术的高拦截性、良性益虫对网络拓扑结构及蠕虫防御反复性方面的高适应性,在保证较好防御效果的基础上,最大限度地节约网络带宽,使节点在有效对抗蠕虫时无须付出高昂代价。

本文第2节介绍Internet蠕虫及P2P蠕虫的发展趋势以及已有的一些防御策略;第3节对本文提出的PAIFDP策略进行详细讨论,并作了相关理论分析;第4节对PAIFDP策略

到稿日期:2010-04-23 返修日期:2010-07-31 本文受自然科学基金(60973119)及教育部博士点基金(新教师基金)(20070614035)资助。

周世杰(1970—),男,博士,副教授,主要研究方向为分布式计算、RFID以及网络安全等,E-mail:sjzhou@uestc.edu.cn;**秦志光**(1956—),男,博士,教授,博士生导师,主要研究方向为信息安全理论、网络安全等;**刘乐源** 男,博士生,主要研究方向为对等计算、网络安全等;**邓昳轶** 男,硕士生,主要研究方向为对等计算、网络安全等。

进行仿真分析；最后总结全文，对未来的研究方向也做了进一步的设想。

2 相关研究工作

蠕虫在本质上是一种黑客行为的自动化工具，从搜索目标，找寻漏洞，到利用漏洞攻击系统，继而传播自身副本，全部主动完成，这点是与传统病毒的本质区别之一。文献[1]从传播途径上将蠕虫分为了三类：Email 蠕虫、文件共享蠕虫和传统蠕虫。在文献[2]中，Nicholas Weaver 等人基于目标巡航机制(Target Discovery)、载体和分发机制(Carrier and Distribution Mechanisms)、激活方式(Activation)、荷载(Payloads)以及攻击者(Attacker)几个方面对蠕虫进行了更精确的划分。通过对已有蠕虫的分析，可以发现目标巡航机制、载体和分发机制、激活方式三个方面在最大程度上决定了蠕虫的传播性能。以 Nimda[3]为代表的混合传播途径的蠕虫，能够使用多种传播途径和传播策略使自身获得更大的传播广度。以 Ramen[4]为代表的多漏洞蠕虫，由于能利用多种不同漏洞进行传播，而可获得较长的活跃时间。以 Slammer[5]和 Witty[6]为代表的超微型蠕虫，虽然其目标巡航机制采用了容易被检查出来的扫描方式，但由于其载荷仅有数百个字节且利用 UDP 协议进行传播，因而仍能快速传播副本，进而产生巨大的网络流量异常。以 Santy[7]为代表的智能传播蠕虫，采用了基于搜索引擎的返回结果来构造攻击目标列表的巡航方式，因而在传播方式上具有了一定的智能性。随着网络及黑客技术的不断发展，蠕虫的发展也趋于多样化、混合化，很难对单一蠕虫做出明确的分类界定。目前，蠕虫已经成为最大的安全问题之一。

与传统蠕虫不同，最早的 P2P 蠕虫 VBS. Gnutella[8]出现在 2000 年 5 月。该蠕虫将自身的副本更名伪装后放置在 Gnutella 应用的共享文件夹中，该病毒文件一旦被下载，即可感染目标主机。2001 年 10 月底出现的 Fizzer 蠕虫[9]，结合了邮件蠕虫的特点，可利用邮件用户地址薄中的信息进行邮件群发，也可将自身副本放置于 KaZaA[10]的共享文件夹中进行传播。由于这两款蠕虫带有传统病毒的特点，需要目标机主动下载带有蠕虫病毒副本的资源，且伪装时也很难保证将文件名更名为近期较热门的资源，因此其在 P2P 网络上传播的效率并不高。隐蔽性较大的 P2P 蠕虫病毒是 2001 年 2 月出现的 W32. Gnuman. Worm 病毒[11]。该蠕虫以伪装后的自身副本作为查询响应，因此传播策略更加高明。其副本大小保持在 8192 字节，需要手工激活。同时，该蠕虫不带任何的荷载，即使被执行也仅仅是传染更多节点，而并不会产生较大的网络流量负担。Zhou Lidong 等人[12]对 P2P 蠕虫的未来发展做了预测。本文认为 P2P 蠕虫可以采用更加主动的方式进行传播，即如同 Internet 蠕虫利用路由表和 DNS 服务构建攻击列表一样，P2P 蠕虫可以利用 P2P 节点主机缓存列表(Host Cache)中的邻居节点来构建攻击列表，以实现准确的目标定位。

在蠕虫病毒的传播性和隐蔽性不断增强的同时，蠕虫病毒(尤其是 P2P 蠕虫)的防御却进展缓慢。现行的蠕虫防御技术大部分都是建立在对防病毒技术和黑客对抗技术的改进基础上。文献[13]对已有的防御策略进行了总结分析，并认为蠕虫的常规防治策略应包括及时修补漏洞、分析特定蠕虫特征、破坏蠕虫运行环境、建立防火墙及路由控制、采用 IDS 技术、全局防御以及紧急告警等。现有的绝大部分防御技术采用了上面提到的一种或几种防御策略，可分为局部防御和全局防御。

局部防御技术包括特征码检测、内存溢出保护、流量抑制和连接抑制技术等。基于特征码机制的防御机制建立在对特定蠕虫特征分析的基础上，是传统病毒防御机制的延续，也是被广泛接受并使用的防御机制。由于蠕虫技术的复杂性和进化性，传统技术很难抑止其大规模传播。且该机制比较被动，对防毒工具的灵敏和准确性以及用户的安全意识都有较高要求。在“百分之八十的漏洞都是由于内存溢出引起”的结论基础上，基于内存溢出保护的防御方法破坏了蠕虫的运行环境，通过对漏洞的有效控制可控制蠕虫的传播。但由于触发内存溢出的方式多种多样，很难做到全面的保护。基于流量抑制的防御是一种路由控制防御技术，其出发点是检测本机的流量异常，采取某种机制阻塞自己对外的连接，防止进一步感染网络上更多的机器。如果有足够多的主机安装运行这套机制就可以在流量上限制蠕虫的扩散，以控制感染主机数量的进一步增多。惠普公司在其高端路由中加入了连接抑制技术，其防御机制建立在恶意代码的行为与正常代码之间存在差异的基础之上[14]。这种机制是以一定程度上降低本机性能为代价，换取全局的正常运行。同时，这种机制只对会尝试建多个异常连接的扫描蠕虫有效，对伪装在正常通信中以及不主动尝试连接其它计算机蠕虫则无法防御。

基于全局的防御技术包括黑名单、蜜罐技术等。黑名单防御机制通常建立在一些常用网络服务提供者上，不给蠕虫传播提供可靠途径，从而降低蠕虫找到有效目标的几率。2004 年 Santy 蠕虫泛滥时，Google 公司将一些关键字列入黑名单，当某机器试图在 Google 搜索一些敏感信息作为目标时，则给其返回无效的信息，在很大程度上降低了 Santy 的危害[15]。该机制对利用网络便捷进行智能搜索的蠕虫有很好的防御效果，且能对一些重要的网络节点提供良好的保护，但在面对自主攻击的蠕虫时防御效果较差。文献[16]和文献[17]提出的利用 DNS 服务来抑制蠕虫传播的方法也是一种黑名单防御技术。由于大部分网络应用都要利用 DNS 来寻址，该机制利用 IDS 来侦测更新黑名单列表，一旦有黑名单内的计算机试图获得 DNS 服务则返回给伪造的 DNS 响应，使其无法找到攻击目标。另一种全局防御的策略是基于蜜罐技术的防御，其最初用于防范网络黑客攻击。虚拟的蜜罐可以检测和阻断网络蠕虫攻击，通过捕获网络蠕虫扫描攻击的数据，利用特征匹配来判断是否有网络蠕虫攻击[18]。此外，蜜罐能够阻断网络蠕虫的传播，以转移蠕虫的攻击目标，降低蠕虫的攻击效果，且自身还具有良好的隐蔽性。但蜜罐布置方案的优劣决定了其侦测阻断效果，且不能在蠕虫泛滥初期起效，因而需与其他的防御机制配合使用才能获得较好的防御效果。

使用良性益虫对抗蠕虫是一个新颖的思考方向，已出现的良性益虫都是在互联网环境中的。Cheese[19]蠕虫利用 Lion 蠕虫留下的后门控制被感染的主机，清理 Lion 蠕虫留下的后门，修补系统的漏洞。W32. Nachi. Worm[20]则利用 W32. Blaster 所使用的系统的漏洞对其进行对抗。良性益虫 CRClean[21]专门针对 CodeRedII 蠕虫。它通过捕获 CodeRe-

dII 的漏洞扫描信息，对攻击主机发起反攻击，并在反击成功后删除攻击主机上的 CodeRedII 代码，进而将自己驻留到该主机上。这种被动传播策略，虽然可以很准确地找到已感染主机，但是如果一旦攻击主机伪造 IP 地址，CRClean 将无法定位到攻击主机。在文献[22]中，通过建立 WAW(worm-anti-worm)模型对利用良性益虫对抗蠕虫的防御效果进行了理论分析。文献[23]则认为现行的蠕虫防御技术过于被动，使用良性益虫对抗蠕虫可以取得更主动的效果，并进而提出了一种通过捕获蠕虫、转化并释放的方式生成良性益虫的过程。该方法在对抗 3 种已有的蠕虫(CodeRed，Slammer，Blaster)能取得良好效果。

综上所述，局部防御技术大多依赖于特征代码检测和流量异常检测，对于非扫描型蠕虫(特别是智能蠕虫和路由蠕虫)效果甚微，但该方法具有低资源消耗的优点。全局防御技术则通过对网络的全局状态进行监控和分析，来达到抑止蠕虫传播的目的。虽然现有的全局防御技术在蠕虫防御方面效果欠佳，但该方法(尤其是基于蜜罐的防御技术)不依赖于蠕虫检测，具有适应性强和拦截性高的优点。良性益虫对抗技术是一种新型的防御方法，对于抑止蠕虫的反复发作具有良好前景，但已有的良性蠕虫都存在于 Internet 中，且大多采用了扫描方式的目标巡航技术都会对网络流量产生不良影响。文献[23]所述的主动良性益虫防御技术也存在同样的问题，该防御机制能自动捕获恶性蠕虫而生成新的良性益虫，但由于它只是重载恶性蠕虫的荷载部分，如果目标蠕虫是一款扫描类的蠕虫，虽然也能起到防御作用，但对网络流量而言无疑是雪上加霜。总之，对于蠕虫(包括 P2P 蠕虫)，较好的防御方法是利用多种机制的协调合作，尽可能减小蠕虫的破坏。本文针对 P2P 蠕虫的特点，充分借鉴了已有良性益虫在抑止蠕虫传播上的优势和不足，并结合非结构化 P2P 网络的路由特性，提出了基于良性益虫的 P2P 蠕虫病毒防御方法。该防御策略充分结合了被动防御策略低资源消耗、蜜罐技术的高拦截性、良性益虫对网络拓扑结构及蠕虫反复性方面的高适应性，在保证较好防御效果的基础上，最大限度地节约网络带宽，并使得节点在有效对抗蠕虫时无须付出高昂代价。

3 P2P 环境下基于良性益虫的 PAIFDP 防御策略

3.1 概述

已有的 P2P 蠕虫传播对用户的特定行为依赖性很大，大多采用共享文件方式进行传播，造成的冲击都比较有限。但是，P2P 蠕虫也完全可以采用更加主动的方式进行传播(如利用本地节点逻辑拓扑结构中的邻居节点信息来识别新的可染主机)。传统扫描蠕虫并不知道被染主机在网络所处的位置，所以在选择目标上是趋错的(error-prone)。它的性能取决于可染主机在整个 IP 地址空间中的密度，以及节点快速感染其他不同节点的能力。这些特征都被作为抑制蠕虫机制的设计出发点。已有的机制在快速侦测以及成功抑制扫描蠕虫上显示了良好的前景，但在对付 P2P 蠕虫时则显得能力有限。其主要原因是作为一种拓扑蠕虫，P2P 蠕虫并不像扫描蠕虫一样在网络行为上有明显的可查性。P2P 的拓扑结构为蠕虫提供了一个精确的途径来找到更多的可染主机而不须进行随机猜测，在极大地提高识别可染主机准确度的同时也消除了与大量不同主机进行高强度通信的必要性。此外，攻击信息也可以轻松地混入普通的 P2P 信息中。

通过对已有良性益虫的分析，大部分的良性益虫的失败原因在于采用了扫描式巡航目标策略，这种策略不仅带来了巨大网络通信开销，而且在打补丁的环节对上下文考虑不周，不能保证有效地给目标机打上补丁，因此难以消除恶意蠕虫的反复性威胁。CRClean 是一个有借鉴价值的案例，即只通过被动的方式激活，而不产生任何网络负担，同时如果能在防伪造地址的攻击方面做些改进，则可以得到非常好的效果。本文的 PAIFDP 策略即是以此为基础提出来的。

3.2 PAIFDP 中良性益虫的基本原理

基于良性益虫的被动激活主动传播防御策略(Passive Activating Initiative Flooding Defense Policy：PAIFDP)是一套综合型的防御策略。在考虑到网络全局的基础上以良性益虫作为防御功能的主要载体，将防御的责任分发给每个普通用户。其中被动激活指的是益虫不会主动地进行激活，而是响应特定攻击行为激活。主动传播指的是激活后的益虫会对一跳以内的所有邻居节点做浅泛洪式的散布。全局上看，该防御策略具有神经元网络的某些特征，外部的刺激(即针对某个漏洞的攻击)会引起防御策略的反应，扩大自身的影响范围，并减少外部攻击者的活动区域，直至蔓延并消灭攻击源头。而在长时间没有外部刺激的情况下它则会自动消亡。局部上看，各个初始防御节点上的良性益虫就像是“母虫”，随着攻击的发生而繁殖下一代的“母虫”至其他节点。它并不判断攻击的来源，只关心自己身边的节点的安全，在让所有邻居节点拥有自己的下一代“母虫”后，则不再理会攻击。而一种“母虫”只针对一种漏洞攻击，不会产生误判。它们之间不需要通讯联系，由攻击者的攻击行为作为引导，自动地向攻击的方向靠拢，在保证完全阻断攻击者的基础上，不去进行不必要的繁殖过程。同时，新的良性益虫升级只需要修改新的漏洞和对应的补丁即可，编写周期比新编写恶意蠕虫要短(甚至可以与厂商合作，在公布漏洞之前就编写好新的良性益虫)。图 1 显示了 P2P 网络中 PAIFDP 良性益虫在受攻击后的扩散情况。

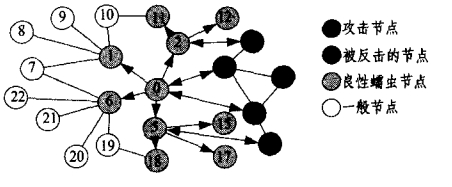

图 1 PAIFDP 良性蠕虫在受到攻击后的散布情况

由于良性益虫是防御方提供的，具有生成的及时性和散布渠道的可靠性等先天优势，同时，将防御功能以蠕虫形式驻留，内存资源开销很小，因此能以低代价对抗恶意蠕虫。益虫技术如同以低开销方式在网络上随机散布带有反击功能的蜜罐，从而维护局部乃至全局的网络安全。以下内容将从益虫功能、活动流程、驻留阶段选择与散步策略等方面进一步分析本文提出的 PAIFDP 防御方案。

3.2.1 良性益虫功能描述

P2P 网络中节点上维护的节点信息可以非常轻易地被恶性蠕虫所利用，同样地，良性益虫也可以利用这些信息来构建自己的“攻击列表”，以此来避免任何网络流量负担。由于 P2P 恶性蠕虫大多不会去进行目标扫描，不可能用本地机流量异常来发现被攻击或感染，因此需要一种针对指定漏洞的侦测机制来发现攻击行为。同时为了避免伪造 IP 地址而产

生网络流量负担增加的问题，良性益虫应在确保自身及其所有邻居节点对指定蠕虫免疫后自杀。另外良性益虫的分布也须在全局上具有不可测性，让恶意蠕虫的编写者无法绕开这类具有防御功能的节点。最后良性益虫应该在全局上具有时间可控性，在指定期限内自杀，避免常驻内存给主机资源造成负担。

综上所述，本文提出的PAIFDP方案中，一个良性益虫的主要功能模块包括：

(1)信息搜集模块：当该蠕虫驻留主机的内存时，根据host_cache中邻居节点信息，构建攻击列表，当自身副本被传播后，在构建攻击列表时剔除副本源头的主机IP，避免重复感染。

(2)扫描探测模块：该蠕虫不使用主动搜索功能，而是采用被动激活的方式，等待侦测到针对某特定漏洞的攻击后，给自身打上补丁，并将自身副本传播到所有邻居节点，同时反击攻击源，如果成功则为攻击源打上补丁，最后传播自身副本至攻击源的所有邻居节点。

(3)自我推进模块：可以利用P2P应用的文件传输通道来实现。将该蠕虫做成守护进程的形式，只有良性益虫可以利用特定P2P应用的文件传输通道在一定程度上降低良性益虫的代码量，同时保证良性益虫在传播中的完整可靠性。

(4)攻击渗透及漏洞侦测模块：为了提高其通用性，降低良性益虫的开发难度和开发周期，可将其他模块做成toolkit形式，而攻击渗透模块则应根据每次漏洞的不同，编写新的攻击渗透模块。这种做法已经在恶意蠕虫中有先例，例如：sclapper[24]蠕虫将其代码中的攻击渗透模块改成一个最近公布的漏洞的攻击渗透模块而形成了最后传播更广的slapper[25]蠕虫。同时攻击渗透模块还附带针对特定漏洞的侦测功能，以便准确快速地对特定漏洞攻击做出反应。

(5)辅助功能模块：辅助功能模块中加入对应特定漏洞的打补丁功能模块即可。

该蠕虫的功能模块结构如图2所示。

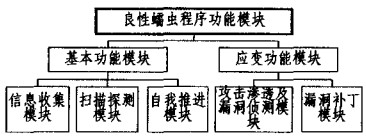

图2 良性益虫功能模块图

3.2.2 良性益虫的活动流程

该良性益虫在节点上获得驻留后，会获取该节点的host_cache信息，以此为基础构建其hit-list列表，并进入驻留计时阶段。在驻留期间，一旦探测到有针对某个漏洞的入侵活动，则立刻进入激活阶段。进入激活阶段的良性益虫根据其hit-list列表，传播自身副本至其它的邻居节点进行驻留，完成后给节点打上补丁，然后自杀并释放占用的资源。如果在驻留期间没有探测到任何入侵活动，在超过驻留期限后，给本地打上补丁，然后自杀并释放所占用的资源。PAIFDP采用如下算法对蠕虫传播进行抑止：

算法1 PAIFDP中益虫工作过程

```
01 getHostCache() /*驻留后获取本地 hostCache()信息*/
02 buildHitList() /*以此构造 hitlist 列表*/
03 while(timeMark<=lifeCycle) /*当时间戳没超过益虫生命周期
   时进行如下循环*/
04    If(attackDetect()=TRUE) /*检测是否遭到特定攻击*/
05      activatingAntiworm() /*如果发现攻击则激活益虫*/
06      propagate(m_hitList) /*根据 hitList 传播自身副本至所有
                              邻居节点*/
07      patchLocalHost() /*为本地节点打上补丁*/
08      suicide() /*在本地自杀，释放所占用的资源*/
09    else
10      timeMark++ //如果在生命周期内没有发现任何攻击行
                   为
11    end if
12 end while
13 patchLocalHost() /*为本地节点打上补丁*/
14 suicide();/*在本地自杀，释放所占用的资源*/
```

3.3 初期驻留点的选择及散布策略

3.3.1 对于P2P网络的关键节点部署良性益虫

在PAIFDP中，我们选择P2P网络中的一些关键节点(Key Nodes)进行益虫的防御性部署。这里的关键节点包括拓扑结构中度数较大的点以及在扩展P2P应用中资源较多、易被访问的超级节点(super Nodes)。当一个漏洞被公布出来后，网络管理员从补丁提供者那里下载的将不再是单纯的补丁，而是这种带补丁程序的良性益虫。将这种良性益虫放置在这些关键节点的理由包括：

(1)及时保护这些需要安全保障的节点，使恶意蠕虫不会对它们的正常运行产生影响。

(2)由于这些节点流量或度数较大，因此其截获到漏洞查询信息的几率也就越大，这有利于良性益虫的散播，也就是有利于补丁的散布，以及时降低恶意蠕虫的活动空间。

(3)这些关键节点，有很大几率在局部网络间的连接处，这样就可以将蠕虫的活动区域限制在局部网络中而不会扩散至整个网络。

3.3.2 普通节点的分布式下载渠道

同样，普通节点下载补丁也是下载该良性益虫。不同的是，当普通节点试图到补丁提供商处下载补丁时，后者根据其IP地址段，反馈就近的下载过该补丁的节点的地址信息，然后普通节点再就近下载补丁(即迭代查询方式)。使用该方式可以有效地避免对补丁提供商的DDOS攻击，同时也使得普通节点能更快地下载到蠕虫补丁。补丁散布的基本原理如图3所示。

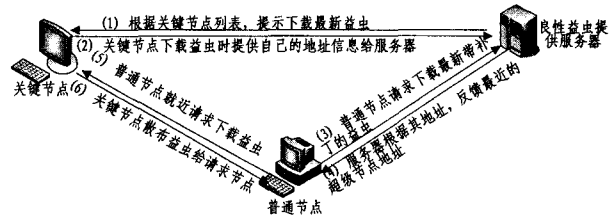

图3 迭代方式实现普通节点益虫散布原理图

3.4 PAIFDP策略优势

3.4.1 良性益虫的功能优势

1)利用蠕虫本身携带补丁的形式，一方面能最小化占用本地资源，另一方面打上补丁的方式使节点免疫，可以避免恶意蠕虫的反复性。

2)良性益虫的激活是被动的。在实际情况下，某些漏洞并不致命，或者这些漏洞并没有被恶意蠕虫作者作为攻击目标。被动激活的方式，可以避免“反应过度”，不会引起无谓的

网络流量及本地资源负担的增加。

3)良性益虫的传播是仅以邻居节点为目标的浅泛洪方式。这样可以避免主动搜索感染主机而给网络带来负担,同时也可以避免IP地址伪造攻击所带来的威胁。例如有A,B,C,D 4个节点,节点B已被感染,并试图伪造IP地址来攻击A,节点A不能通过准确定位反攻节点B,但由于节点A在节点B的邻居节点列表中,说明两节点近期有通讯,那么节点B很有可能就在节点A的邻居节点列表中,即使不在,节点A对其所有邻居包括节点C,D进行传播良性益虫,也可以使节点B试图攻击其邻居列表中C,D节点的企图落空,并遭到反击(C,D在A节点邻居节点列表,就极可能也在节点B的邻居节点列表中),以此在全局上对指定蠕虫实现抑制和扑杀。

3.4.2 良性益虫驻留节点的随机性选择优势

普通节点在下载这种蠕虫补丁后,就充当了"民兵"的角色,类似于蜜罐技术中蜜罐的作用,而这些民兵的出现完全取决于用户个人行为,这是不可预测的。以前的防御策略,对防御者来说,网络状况是不可知的,而攻击者往往隐蔽在这种不可知性下对网络进行攻击。而这种新防御策略下,所有普通节点都可能拥有反击的"武器",对攻击者来说,下一个攻击目标是否会反击是不可预测的,即蠕虫的编写者会同样面临网络的不可预测性,将不可能在新编制的蠕虫中设立机制来绕开这种具有反击能力的节点,而如果不做判断地肆意攻击,就会受到更剧烈的反击,结果是更多的节点更快地打上补丁,最终扼杀了自己的活动空间,使其在尚未广泛散播之前遭到扑杀,同时这种反击以不牺牲网络资源为前提。

4 PAIFDP策略的仿真实验

由于引入了良性益虫,防御效果主要取决于良性益虫的反应速度和良性益虫的传播速度,另外以下的问题也不能被忽视:

(1)由于现实中用户安全意识并不能保证,因此在仿真中我们要假设恶性蠕虫的发起点的个数超过了初始良性益虫驻留的节点个数。在这种最差情况下,少量的良性益虫能否有效地抑制恶性蠕虫的传播?

(2)由于考虑到普通用户的心理承受能力及计算机处理能力,在一段时期中一台主机内存中驻留的良性益虫不能过多。那么,在减少驻留期限的情况下,恶性蠕虫能否得到有效的抑制?

4.1 仿真环境简介

本文采用PEERSIM P2P仿真软件作为实验分析工具。网络结构采用BRITE网络拓扑生成器生成符合power-law结构特点的网络拓扑作为实验的基础[26]。本文仅考虑针对采用Gnutella0.4协议所构建的对等网络。在Gnutella网络中,探询消息(Ping Message)和查询消息(Query Message)均采用泛洪路由方式,而探询应答(Pong Message)和查询命中消息(Query Hit Message)则是单步回溯的方式。从Gnutella对等网络的拓扑结构特性来看,在这种环境下P2P蠕虫可获得较大的散播性和很好的隐蔽性。本防御机制如果能对这种状况下的恶意蠕虫取得很好的防御效果,那么在其他的P2P应用中也能获得良好的效果。

实验中,我们改进了PEERSIM的初始化模块,该模块中可以输入不同的P2P拓扑结构和网络规模。每次运行,仿真工具都依据一定概率在拓扑图中随机选择一些节点,分别初始化为恶意蠕虫传播节点和良性益虫驻留节点,通过实验来模拟蠕虫繁殖和恶意蠕虫散布的过程。整个实验由节点的状态信息驱动,不模拟节点间的信息传递,因此可仿真大规模对等网络中蠕虫传播及其防御的情况。

4.2 实验参数及评价指标

显然,系统规模(即节点总数)N、初始良性益虫所占比例K_a^0等系统参数会直接影响防御效果。考虑到蠕虫病毒爆发的突发性,系统中初始恶意蠕虫感染节点数N_w^0和初始良性益虫数N_a^0以及二者的比值$K_{a,w}^0=\frac{N_a^0}{N_w^0}$对防御效果也有直接影响。由于蠕虫与具体的网络拓扑结构有密切关系,因此网络拓扑结构也是实验中的一个重要参数。本文主要考虑分布式的P2P网络拓扑和具有超级节点(Super Node)的P2P网络拓扑。实验参数及其含义如表1所列。

表1 系统参数

参数	含义
N	系统规模,即仿真中总体的网络节点数目
K_a^0	初始良性益虫在总体规模中所占比例
N_w^0	初始恶意蠕虫感染节点数
N_a^0	初始良性益虫数
$K_{a,w}^0$	$K_{a,w}^0=\frac{N_a^0}{N_w^0}$初始良性益虫与初始恶意蠕虫比
P_w	恶意蠕虫的感染能力
P_a	良性益虫的传播能力
$P_{a,w}$	$P_{a,w}=\frac{P_a}{P_w}$良性益虫与恶意蠕虫的感染能力比
T_a	良性益虫生命周期

恶意蠕虫的感染能力P_w和良性益虫的传播能力P_a也是系统的一个重要参数。恶性蠕虫感染能力值P_w模拟蠕虫在单位时间里尝试攻击的次数(即探测邻居节点的状态),良性益虫的传播能力值P_a表示良性益虫在单位时间内可以通过散发补丁等方式所免疫的节点数。两个值的高低可反映两种蠕虫在单位时间里的感染能力的强弱。对于良性益虫,还有一个所谓驻留周期(AntiWormLife时间片)或生命期(Life时间片)的问题,它反映了良性益虫在系统中的驻留时间长短,用T_a来表示。以良性驻留时为起点,只要在此期间受到恶性蠕虫的探测企图则都可以引起良性益虫的激活与反击。

实验中每个节点的状态包括感染(Infected)、易染(Vulnerable)、免疫(Immune)、良性益虫驻留(Anti-Worm-Resident)、良性益虫激活(Anti-Worm-Active)5种状态。当整个系统趋于平衡时,系统中既没有新的感染节点产生,也没有新的免疫节点产生。通过统计和计算稳定状态下感染节点数I在总节点数N(不包括初始化时驻留了良性益虫的节点数N_a^0)中所占的比率,可以对防御效果进行评估。本文将该数值定义为感染区(Infected-fraction)$F_i=\frac{I}{N-N_a^0}$,并以此作为防御策略效果的判断标准。显然,感染区F_i值越小说明防御效果越好,反之防御效果则差。

蠕虫引起的网络资源消耗通常包括3个部分:寻找可路由主机的探测消息引起的网络资源消耗;找到可路由主机后查看主机是否可染引起的网络资源消耗;以及实际进行副本传播时引起的资源消耗。对于P2P上的蠕虫来说,由于邻居列表中的主机都是可路由的,故不会带来探测消息的资源消

耗。为了简单起见，将查看消耗和传播消耗等效，即查看邻居的节点状态记录为一次资源消耗，实际传播自身副本也记录为一次资源消耗。良性益虫引起的资源消耗定义为 C_a，恶性蠕虫引起的资源消耗定义为 C_w。系统总体资源消耗 C_t 为两者之和：$C_t=C_a+C_w$。此外，良性与恶性蠕虫的资源消耗比定义 C_r 也可以作为资源消耗的评价指标：$C_r=\frac{C_a}{C_w}$。防御有效性评价指标如表 2 所列。

表 2　防御性能评价指标

参数	含义
F_i	$F_i=\frac{I}{N-N_a^0}$ 感染区值，即感染节点在初始可感染节点中的比例
I	系统稳定后，总体感染的节点数目
C_a	良性益虫引起的资源消耗
C_w	恶意蠕虫引起的资源消耗
C_t	$C_t=C_a+C_w$ 系统总体资源消耗
C_r	$C_r=\frac{C_a}{C_w}$ 良性与恶性蠕虫的资源消耗比

4.3　网络规模的影响

不同的网路规模 N 下，整体的防御效果可能会有所不同。将初始感染节点数 N_w^0 设定为整个网络规模的 1%，初始良性益虫节点数 N_a^0 设定为整个网络规模的 1%，良性益虫和恶性蠕虫的感染能力 P_a 和 P_w，均设为 3。良性益虫生命周期 T_a 设定为到实验终止。分别将网络规模 N 设为 1000，5000，10000，30000，50000，观察并记录感染区 F_i 的最大值和最小值、良性和恶性蠕虫的资源消耗 C_a，C_w，C_a 与 C_w 的比值 C_r，以及消耗总值 C_t。实验结果如表 3 所列。

表 3　不同网络规模下的防御效果

网络规模	F_i 最大值	F_i 最小值	C_a	C_w	C_r	C_t
1000	83.64%	0.10%	4330	15111	0.29	19444
5000	80.51%	0.00%	21111	66035	0.32	87146
10000	79.63%	0.01%	42331	128581	0.33	170912
30000	78.45%	0.02%	126997	382056	0.33	509053
50000	80.36%	0.02%	211845	681131	0.31	892946

表 3 中的数据表明，在此防御策略下，网络结构的规模对整体的防御效果没有明显影响，说明此防御策略能适应各种网络规模的 P2P 应用网络。需要注意的是，真实的 P2P 网络具有百万级的节点，而恶性蠕虫和良性益虫的感染能力都不可能如本试验所示那样强大，在此条件下其最大的感染区大小应比本试验显示的要低得多。

4.4　引入良性益虫节点的影响

首先我们考虑引入良性益虫后对防御结果的影响。在实验中，将初始感染节点数 N_w^0 设定为整个网络规模的 1%，网络规模 N 设定为 50000，良性益虫和恶性蠕虫的感染能 P_a 和 P_w 均设定为 3。在此条件下，考虑良性益虫数 N_a 占整个节点数的比例为 1%（即 $N_a/N=0.1$）的情况下其感染区大小和资源消耗的变化情况。通过表 4、图 4 及图 5 可以看到，在网络中不存在良性益虫的情况下，恶性蠕虫只需要 4 个时间片就可以感染整个 P2P 网络的 90%以上，即是说在 P2P 网络中，一款蠕虫只要在一个节点上对所有邻居节点复制自身，那么只需要经过繁殖 4 代，就可以达到 90%以上的覆盖率，如果不加防范，要用一款蠕虫击跨一个 P2P 网络是非常容易的。

表 4　加入良性益虫对防御效果的影响

时间片	F_i（无良性益虫）	F_i（有良性益虫）	C_t（无良性益虫）	C_t（有良性益虫）
0	1%	1.01%	0	0
1	9.61%	9.11%	10079	9191
2	43.43%	40.34%	63782	60331
3	79.50%	71.99%	176208	167377
4	94.07%	79.15%	315931	296033
5	98.18%	72.81%	462813	423672
6	99.32%	58.92%	611775	541979
7	99.72%	40.17%	761305	646667
8	99.85%	20.59%	911056	735058
9	99.93%	7.21%	1060939	802042
10	99.95%	1.78%	1360826	841172

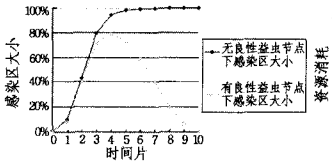

图 4　加入良性益虫后感染区变化对比曲线图

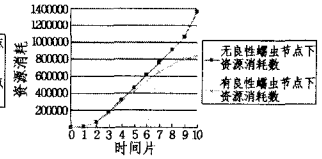

图 5　加入良性益虫后资源消耗数变化对比曲线图

在加入良性益虫后，感染区在蠕虫繁殖最活跃的第三代和第四代时能进行有效的反击，在第五代以后就能明显地收敛感染区，最终将特定蠕虫在 P2P 网络中彻底清除。而资源消耗也并没有因为反击而增大，相反，由于对恶性蠕虫的有效扑杀，资源消耗比没有防御的情况下要小得多。同时需要注意，本实验仿真的资源消耗都是必要的资源消耗，而实际情况下，恶意蠕虫还会带来一些攻击性的资源消耗，而良性益虫在单个节点上产生的资源消耗是可控的，因此实际条件下良性益虫的存在可以在更大程度上减少网络资源的消耗。

4.5　超级节点的影响

P2P 网络中的超级节点拥有大量丰富资源和高质量连接。参考不同节点的能力差异，通常将网络中度数较大、连通性较高且拥有大量索引信息的节点升级为超级节点。为了简单起见，实验不考虑超级节点的资源分布情况，仅将高度数的点设定为超级节点。经统计，生成的 50000 规模拓扑中度数大于 100 的节点共有 148 个，将这 148 个节点定义为超级节点，观察在 10 个时间片内，随机选择初始节点、良性益虫优先占领和恶性蠕虫优先占领的情况下感染区大小和资源消耗数的变化情况。同时为了体现超级节点的连通性优势，将两种蠕虫的感染能力都设定为 5，两类蠕虫的初始个数分别为 200。实验结果如表 5 所列。

表 5　超级节点选择性对防御效果的影响

时间片	F_i			C_t		
	恶性蠕虫优先	良性益虫优先	随机选择初始节点	恶性蠕虫优先	良性益虫优先	随机选择初始节点
0	0.40%	0.40%	0.40%	0	0	0
1	12.42%	8.26%	11.66%	15549	10985	14358
2	68.47%	55.51%	67.24%	142061	112674	136777
3	87.55%	72.42%	88.83%	365507	315089	360015
4	76.68%	48.05%	79.39%	589346	511142	587245
5	49.94%	15.37%	55.30%	786858	668696	789561
6	17.44%	1.68%	22.19%	944117	776161	954978
7	2.10%	0.10%	3.30%	1057278	817926	1075096
8	0.08%	0.00%	0.17%	1103318	823301	1132658
9	0.00%	0.00%	0.01%	1109991	823652	1143282
10	0.00%	0.00%	0.00%	1110297	823659	1143895

通过表 5、图 6 及图 7 可以看到，将良性益虫布置在超级

节点上能有效地控制最大感染区的大小，加快感染区的收敛速度，说明超级节点的高连通性可以为快速激活良性益虫提供优势。

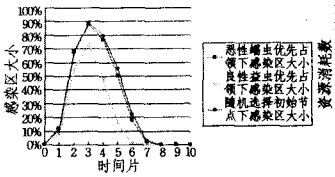

图 6　超级节点不同占先情况下的感染区大小

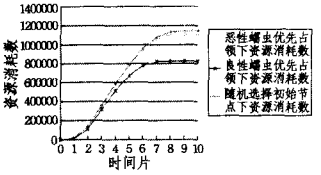

图 7　超级节点不同占先情况下的资源消耗情况

需要注意的是，将恶性节点布置到超级节点的防御效果比随机布置恶性节点略好。该现象可能是因为恶意节点优先占领超级节点，使恶意节点在初期有更多的邻居节点，有更大的几率在早期激活良性益虫所致。此现象也说明在本防御策略下，就算恶性蠕虫在占领超级节点上有优势，也不会明显降低防御效果，并有可能带来防御效果的提升。如果一款蠕虫针对该策略，试图避免传染超级节点，则其传染深度和广度都将受到极大的限制，最终为网络整体免疫提供了足够的缓冲时间。

4.6　驻留时间的影响

益虫驻留时间长短 T_a 有可能会影响整个防御效果。将初始感染节点数 N_w^0 设定为整个网络规模的 1%，初始良性益虫节点数 N_a^0 设定为整个网络规模的 1%，网络规模 N 设定为 50000，良性益虫和恶性蠕虫的感染能力设定为 3（即一个时间片感染 3 个邻居节点）。将驻留时间 T_a 设定为 1，4，8，15 以及到实验终止，观察感染区的最大最小值、良性和恶性蠕虫的资源消耗和消耗总值及比率大小变化情况。实验结果如表 6 所列。其中 t 表示驻留时间。

表 6　驻留时间对防御效果的影响

t	F_i 最大值	F_i 最小值	C_a	C_w	C_r	C_t
1	81.41%	0.39%	212554	740596	0.29	953150
4	80.84%	0.03%	211710	684475	0.31	896185
8	79.14%	0.01%	211305	646245	0.32	857550
15	79.49%	0.02%	212656	652103	0.33	864759
实验终点	79.80%	0.01%	211435	667091	0.32	878526

通过表 6 可以看出驻留时间对防御效果的影响极小，这是由于本防御策略的驻留时间不同于一般蠕虫的驻留时间，其截止日期不是一个统一的现实或系统时间，而是每个蠕虫副本有一个自己的时间记数，以驻留开始为记数起点，只要恶性蠕虫足够活跃，则可以保证长期有良性益虫处于驻留期内。考虑到恶性蠕虫可能在早期避免攻击，等所有良性益虫驻留期过后再行攻击，可以让良性益虫的驻留期设置稍微长些，该期限长度取决于网络达到整体免疫的时间。

4.7　初始节点比率的影响

系统初始时已经被感染蠕虫病毒的节点数（可以称之为攻击节点）与初始驻留了良性益虫的节点数的比率（即 $K_{a,w}^0 = N_a^0/N_w^0$），也可能影响全局的资源消耗率和防御效果。为此，将网络规模 N 设定为 50000，良性益虫和恶性蠕虫的感染能力设定为 3，即一个时间片感染 3 个邻居节点。良性益虫生命周期设定为到实验终止为止。初始良性益虫节点设定为 1000，将恶性蠕虫分别设定为 10，50，100，500，1000，2000，即两者比率为 100∶1，20∶1，10∶1，2∶1，1∶1，1∶2。观察并记录感染区的最大最小值、良性和恶性蠕虫的资源消耗和消耗总值及比率大小。实验结果如表 7 所列。其中，R 表示初始良性益虫与初始恶性蠕虫之比：

$$R=\frac{\text{初始良性益虫数量}}{\text{初始恶意蠕虫数量}}$$

表 7　改变初始恶性蠕虫比率对防御效果的影响

R	F_i 最大值	F_i 最小值	C_a	C_w	C_r	C_t
100∶1	68.59%	0.01%	207870	471523	0.44	680393
20∶1	67.10%	0.01%	208803	467465	0.45	676268
10∶1	68.11%	0.02%	208432	471360	0.44	679792
2∶1	68.82%	0.02%	209062	487975	0.43	697037
1∶1	69.93%	0.02%	209010	490851	0.41	699861
1∶2	72.17%	0.02%	210007	513902	0.41	723909

当初始恶性蠕虫节点设定为 1000，初始良性益虫数 N_a^0 分别设定为 10，50，100，500，1000，2000，即两者比率为 100∶1，20∶1，10∶1，2∶1，1∶1，1∶2 时，分别观察感染区的最大值和最小值、良性和恶性蠕虫的资源消耗和消耗总值及比率大小。实验结果如表 8 所列。其中，R^{-1} 表示初始恶意蠕虫与初始两性益虫之比：

$$R^{-1}=\frac{\text{初始恶意蠕虫数量}}{\text{初始良性益虫数量}}$$

表 8　改变初始良性益虫比率对防御效果的影响

R^{-1}	F_i 最大值	F_i 最小值	C_a	C_w	C_r	C_t
100∶1	98.68%	0.01%	213610	2262832	0.09	2476442
20∶1	95.94%	0.01%	213070	1576571	0.13	1789641
10∶1	93.79%	0.01%	214032	1288243	0.17	1502275
2∶1	78.51%	0.01%	211967	656286	0.32	868353
1∶1	62.84%	0.02%	210102	510189	0.41	720291
1∶2	56.30%	0.02%	201076	334863	0.60	535939

表 7 显示了改变初始良性益虫和恶性蠕虫比率对防御效果几乎没有影响，只有当初始恶性蠕虫超过初始良性益虫两倍以后才会引起防御效果的下降。而表 8 则显示改变良性益虫在总体规模的比率会带来非常明显的防御效果改变。当初始良性益虫驻留节点占整个规模的 1%以后，防御效果会达到一个比较理想的值，能把最大感染区大小控制到可以接受的数值，极大地减小总体资源消耗，并能较快地收敛感染区，其后再增加良性蠕虫的比率带来的效果提升则不会非常明显，然而只要良性益虫不是少到无法被激活，均可以有效地进行收敛。

4.8　蠕虫感染能力的影响

良性益虫和恶性蠕虫的感染能力的不同也会影响防御效果。将网络规模 N 设定为 50000，将初始感染节点数设定为总节点数的 1%，初始良性益虫节点数设定为 1%，良性益虫生命周期设定为实验终止为止。良性益虫感染能力和恶性蠕虫的感染能力比（$P_{a,w}$）分别为：3∶3，5∶5，10∶10，5∶10，10∶5。观察并记录感染区的最大最小值、良性和恶性蠕虫的资源消耗和消耗总值及比率大小。实验结果如表 9 所列。

表 9　改变感染能力比率对防御效果的影响

$P_{a,w}$	F_i 最大值	F_i 最小值	C_a	C_w	C_r	C_t
3∶3	80.05%	0.01%	211513	671061	0.32	882574
5∶5	75.31%	0.00%	278271	634944	0.44	913215
10∶10	75.18%	0.00%	353895	575084	0.62	932979
5∶10	85.57%	0.00%	285713	861397	0.33	1147110
10∶5	69.67%	0.00%	343951	407436	0.84	751387

从表 9 可以看出，对本防御策略来说，两种蠕虫的绝对传

播速度越快，越有利于控制最大感染区的大小，但由于网络中度数不超过10度的节点占绝大多数，故当感染能力提升到一定程度时，就不会带来实质性的效果提升。另外，无论感染能力如何提升，只要两者的感染能力对比不变并不会引起明显的资源消耗改变。而感染能力对比变化，则可能引起最大感染区大小、资源消耗数和系统平衡时间的明显改变。改变方向取决于哪种蠕虫的感染能力更强，如果良性益虫更强，则防御效果更好，反之，则防御效果变差。在实际网络中为了优化防御效果，我们可以通过某些特定服务，加强良性益虫的感染能力。

4.9 多样性的影响

以上的试验是建立在所有普通节点都是可染的假设上的，实际上在大部分P2P应用中，用户可以使用遵循相同协议的不同工具，将其运行在不同的软硬件环境中。由于其多样性(diversity)，在某种工具软件或者操作系统上存在的漏洞不可能影响所有用户。我们称这种不可能被蠕虫病毒感染的节点为初始免疫节点。初始免疫节点的存在对我们进行防御是有益的，它们能阻止恶意蠕虫的传播。实际的防御效果需要通过模拟仿真进行观察，因为多样性在没有特别的服务支持时同样会阻止良性益虫的散布。将网络规模设定为50000，初始感染节点数设定为1%，初始良性益虫节点数设定为1%，良性益虫生命周期设定为实验终止为止，良性益虫和恶性蠕虫的感染能力设定为3，即一个时间片感染3个邻居节点，初始免疫节点分别设定为整个网络规模的10%，20%，30%，40%，50%。观察并记录感染区的最大最小值以及良性和恶性蠕虫的资源消耗和消耗总值及比率大小。需要说明的是，此时的感染区大小是指被感染节点在初始非免疫节点(不包括初始免疫和初始良性节点)中的百分比。实验结果如表10所列。其中，λ_0 表示初始免疫节点比例。

表10 改变初始免疫节点比率对防御效果的影响

λ_0	F_i 最大值	F_i 最小值	C_a	C_w	C_r	C_t
10%	76.24%	0.01%	199429	541346	0.37	740775
20%	69.93%	0.01%	185054	425895	0.43	610949
30%	61.80%	0.03%	167103	336213	0.50	503316
40%	57.14%	0.03%	145688	269161	0.54	414849
50%	46.49%	0.06%	117265	195770	0.59	313035

表10所示的实验结果表明在初始免疫节点比例增加时，感染区最大值和总体资源消耗数都有明显的下降，说明在多样性条件下该防御策略同样具有比较理想的防御效果。

4.10 实验总结

通过上面的实验，可以发现主要有以下的几个因素影响本策略的防御效果：

(1)初始良性节点占整体网络规模的百分比

初始良性益虫节点如果占整个网络规模的1%以上，可保证最大感染区和总体资源消耗均控制在一个较小范围。同时，即便不能达到1%，如果良性益虫在驻留期间能被激活，感染区也可收敛在一个较小值。后者的收敛时间较前者大。

(2)良性益虫感染能力的绝对强度和对比强度

良性益虫在感染能力上要比恶性蠕虫有优势，该优势可以通过P2P应用的服务来实现。例如为良性益虫提供通道，让良性益虫可以利用P2P应用的传输服务来实现副本的繁殖传递，加快其传播速度并保证副本传播的完整性。

(3)P2P协议本身的多样性

P2P应用协议的设计应偏向多样化，对每个节点来说其邻居节点中免疫节点的比率应增加，这种增加会极大抑制恶性蠕虫的传播，对良性益虫的传播效果影响较小，最终在总体上控制了感染区的扩大。

(4)初始良性益虫的分布机制

初始良性益虫节点应较早地布置到超级节点上，之前的实验也证明了良性益虫节点的位置对整个防御效果的重要性。

实验结果表明，PAIFDP防御策略具有网络规模无关性，可以适应任何规模的P2P应用。良性益虫产生的资源开销远远低于恶性蠕虫带来的资源开销，并能极大地减小恶性蠕虫带来的资源消耗。同时该策略能对不同的攻击强度做出自动的响应，恶性蠕虫传播强度越强则可以带来越强的良性益虫反击，传播速度越快的蠕虫越容易在短时间被扑灭，而应对攻击强度不高的恶性蠕虫时，则不会反应过度，不会引起额外的资源开销。最后在初始节点不能保证占领高度数节点的情况下，该策略仍可以有很好的性能表现，必要时可以采取完全随机的方式布置良性节点以增加超级节点的隐蔽性，提高网络安全性。

结束语 实验结果表明，本文所提出的基于良性益虫的被动激活主动传播防御策略(PAIFDP)具有收敛时间快、网络资源消耗少、适应性强等特点。如果能在P2P网络中布置基于PAIFDP的蠕虫防御机制，则可有效地减少P2P蠕虫带来的破坏。本文所提出的PAIFDP防御策略在超级节点的保密和利用陈旧漏洞攻击的蠕虫仍有可能在小范围内泛滥等问题上加以改进。未来研究工作包括编码实现具有本防御策略功能的良性益虫，进行大规模的更接近真实环境的对抗实验，以此作为其有效性的进一步验证。

参 考 文 献

[1] Kizenzle D M, Elder M C. Recent worms: A survey and trends [C]// Staniford S, ed. Proc. of the ACM CSS Worksh◌p on Rapid Malcode(WORM2003). Washington, 2003

[2] Weaver N, Paxson V. Stuart Staniford Robert Cunningham. A Taxonomy of Computer Worms[C]// Staniford S, ed. Proc. of the ACM CSS Workshop on Rapid Malcode (WORM2003). Washington, 2003

[3] CERT. CERT Advisory CA-2001-26 Nimda Worm [OL]. http://www.cert.org/advisories/ca-2001-26.html

[4] Mihai Moldovanu. Ramen Worm Analysis [OL]. http://tfm.profm.ro/index.html, 2005.8.5

[5] Moore D, Paxson V, Savage S, et al. Slammer Worm Dissection: Inside the Slammer Worm[J]. IEEE Seurity & Privacy, 2003, 1(4)

[6] Kumar A, Paxson V, Weaver N. An Analysis of the Witty Outbreak: Exploiting Underlying Structure for Detailed Reconstruction of an Internet-scale Event[C]// Proc. of the ACM CSS Workshop on Rapid Malcode(WORM2005). Washington, 2005

[7] 郑辉. Santy蠕虫分析报告[R]. 中国教育和科研计算机网紧急响应组, 2004

[8] Symantec Inc, VBS. Gnutella[OL]. http://securityresponse.symantec.com/avcenter/venc/data/vbs.gnutella.html

(下转第79页)

的组合运算。最后是检验故障的步骤，又分为两种情况，一种是采用条件检验是否有故障发生，如 Shamir 的防御算法，假如发生则不输出签名；另一种是产生随机数，假如有故障产生则用随机数替代签名，否则输出正确的签名。攻击者若能够在前面的操作步骤注入故障，且跳过检验步骤或检验步骤出错，则仍然可以实施故障攻击。新的攻击算法也证明了原有的防御算法并不能有效地抵御故障分析，主要原因是原有的防御措施都是在计算最终签名结果之后才进行错误检验，最终都可以通过注入故障和计算 $f()$ 得到如下的形式：

$f(\tilde{S})=\delta+q\Delta \bmod n$ 或 $f(\tilde{S})=\delta+p\Delta \bmod n$

再通过其它的组合计算 $F()$ 获得 $\gcd(F(f(\tilde{S})),n)=p$ 或 q，从而因式分解 n。

本文提出一种可行的防御措施建议：核心算法是让最终签名结果的计算在故障检验步骤之后进行，具体为：首先在组合步骤之前产生随机数，计算出一个中间结果值 Sig^*，然后执行故障检验步骤，假如有故障则不输出结果，否则再通过计算消除随机数，并输出正确的签名 Sig。

结束语 本文提出了针对 RSA-CRT 的故障攻击算法，介绍了基于模幂运算故障的攻击算法以及对应的防御算法，其中以 BOS 算法为分析对象，建立了新的基于错误检验故障的 RSA 故障攻击模型。与 Wagner 的攻击算法不同，本文给出了永久故障与暂时故障相结合的故障攻击算法，并进行了理论推导和仿真实验，通过实验对比了本文攻击算法与原有攻击算法的复杂度和故障利用率，实验结果验证了本文攻击算法的可行性，且比 Wagner 的攻击算法具有更高的执行效率，执行攻击所需的故障签名数更少。为了更全面地讨论攻击算法，应针对其它可能出现的故障情况作进一步分析，最后总结故障攻击的核心思想，分析了原有防御算法的不足，提出了防御建议。

目前本文仅从理论和仿真实验的角度针对基于错误检验故障的 RSA 故障攻击进行研究，研究了故障可能出现的新情况，给出了相应的攻击算法。但实际过程中故障出现的情况是多变的，针对不同情况的故障还需要做不同的分析，另外关于如何在适当的位置和时机注入故障等方面还有待进一步研究，为抵御新的防御算法设计一种有效的防御算法也是将来值得研究和关注的。

参 考 文 献

[1] Kocher P. Timing attack on Implementations of Diffie-Hellman, RSA, DSS, and Other systems[C]// Proceedings of Advances in Cryptology-CRYPTO'96. Springer-verlag, 1996: 104-113

[2] Boneh D, DeMillo R, Lipton R. On the Importance of Checking Cryptographic Protocols for Faults[C]// Fumy W, ed. Advances in Cryptology—Eurocrypt ' 97, LNCS1233. Springer-Verlag, 1997: 37-51

[3] Shamir A. How to Check Modular Exponentiation[C]// EUROCRYPT ' 97. Springer-Verlag, 1997

[4] Blömer J, Otto M, Seifert J-P. A New RSA-CRT Algorithm Secure Against Bellcore Attacks[C]// Sushil Jajodia, Vijayalakshmi Atluri, Trent Jaeger, eds. Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003. Washington, DC, USA, October 2003: 311-320

[5] Lenstra A K. Memo on RSA Signature Generation in the Presence of Faults[EB/OL]. http://cm. bell-labs. com/who/akl/ September 1996

[6] Wagner D. Cryptanalysis of a Provably Secure RSA-CRT Algorithm[C]// Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick Drew McDaniel, eds. Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004. Washington, DC, USA, October 2004: 92-97

[7] Schmidt J-M. Differential Fault Analysis [R]. A report from IAIK Lab in Austria. June 2008. http://www. a-sit. at/pdfs/ DFA-Report. pdf

(上接第 64 页)

[9] Symantec Inc, W32. HLLW. Fizzer@mm[OL]. http://security-response. symantec. com/avcenter/venc/data/w32. hllw. fizzer @mm. html

[10] KaZaA[OL]. http://www. kazaa. com

[11] Symantec Inc. , W32. Gnuman. Worm[OL]. http://securityresponse. symantec. com/avcenter/venc/data/w32. gnuman. worm. html

[12] Zhou Li-dong. A First Look at Peer-to-Peer Worms Threats and Defenses[C] // Miguel Castro, Robbert van Renesse, IPTPS 2005. New York, 2005: 24-35, http://research. microsoft. com/ users/lidongz/p2pworm. pdf

[13] 郑辉. Internet 蠕虫研究[D]. 天津：南开大学信息技术科学学院，2003

[14] HP Inc，基于病毒遏制技术(Virus-Throttling)的连接速度过滤[OL]. http://www. hp. com. cn/network _ pmg/pdfs/virus _ throttling_tech_brief. pdf

[15] 郑辉，吕冠一. Google Hacking 与智能蠕虫防治[J]. 信息安全与通信保密，2005，8：70-74

[16] 郑辉. DNS 抑制蠕虫传播[C]// XCON'2003. 2003

[17] Whyte D, Kranakis D, van Oorschot P C. DNS-based Detection of Scanning Worms in an Enterprise Network[C] // The 12th Annual Network and Distributed System Security Symposium (DNSS'05). San Diego, 2005

[18] Provos N A. virtual honeypot framework[R]. 03-1. Center of Information Technology Integration, University of Michigan, 2003

[19] Barber B. Cheese Worm: Pros and Cons of a ''Friendly "Worm [C]// SANS Conferences 2001. SANS Institute, 2001

[20] Symantec Inc, W32. Welchia. Worm. http://www. symantec. com/avcenter/venc/data/w32. welchia. worm. html

[21] Kern M. Codegreen beta release. 2001[OL]. http://online. securityfocus. com/archive/82/211462

[22] 文伟平，卿斯汉，蒋建春，等. 网络蠕虫研究与进展[J]. 软件学报，2004，15(8)：1208-1218

[23] Castaneda F, Sezer E C, Xu Jun. WORM VS WORM: Preliminary Studay of an Active Counter-Attack Mechanism[C]// Proc. of the ACM CSS Workshop on Rapid Malcode (WORM2004). Washington, 2004

[24] Mituzuas D. FreeBSD Scalper Worm[OL]. http://www. dammit. lt/apache-worm/

[25] F secure Inc. Global slapper worm information center[OL]. http://www. f-secure. com/slapper/

[26] Tian Bu, Towsley D. On distinguishing between Internet power law topology generators[C] // Proc. of the IEEE INFOCOM 2002. New York, IEEE, 2002: 638-647