使用主动方法检测未知的大量邮件病毒

Ruiqi Hu 和 Aloysius K. Mok

计算机科学系，
德克萨斯大学奥斯汀分校，Austin，Texas 78712，USA
{hurq,mok}@cs.utexas.edu

抽象的。未知病毒的检测超出了许多现有病毒检测方法的能力。在本文中，我们展示了如何使用主动定制系统行为来提高未知恶意可执行文件的检测率。介绍了两种通用的主动方法，行为倾斜和封锁，以及它们在检测未知大量邮件病毒的原型系统 BESIDES 中的应用。

关键词：病毒检测；恶意可执行检测；入侵检测。

1 介绍

入侵检测系统（IDS）采用的两种主要方法是基于误用的检测和基于异常的检测。因为基于误用的检测依赖于已知的入侵签名，所以基于误用的 IDS 通常无法针对签名尚未编目的新型入侵提供保护。另一方面，由于基于异常的 IDS 依赖于识别与受保护系统行为的“正常”配置文件的偏差，因此除非受保护系统的正常配置文件得到很好的表征，否则它们很容易报告误报。

在任何一种情况下，都必须在入侵发生后立即执行检测，并且一定要在入侵者造成伤害和/或隐藏自己的踪迹之前执行。在本文中，我们介绍了 PAIDS（主动入侵检测系统）的范例，并描述了 PAIDS 的应用，它可以在不知道先验特征的情况下检测某些类型的入侵，并且误报率非常低。 PAIDS 还提供了一种动态权衡检测入侵所需时间和入侵者可能造成的损害的方法，因此可以在一定程度上容忍入侵。为了实现这些优势，PAIDS 采用了两种通用技术：行为偏差和警戒线。

传统上，计算机系统的安全性由一组安全策略捕获。一个完整的安全策略应该将系统的每个行为分类为合法或非法。然而，在实践中，安全策略的规范通常无法扩展 [1] 并且可能不完整。给定一个安全策略，我们可以将系统所有行为的集合划分为三个子集：(S1) 合法行为，(S2) 非法行为和 (S3) 未指定行为，对应于行为是一致的、不一致的还是独立的的

E. Jonsson 等人。 （编辑）：RAID 2004，LNCS 3224，第 82 – 101 页，2004 年。
2004年柏林海德堡QC施普林格出版社

安全政策。 （或者，将安全策略视为建立系统行为合法性的理论公理。）在这种情况下，行为倾斜是指将安全策略 P 修改为 P’，使得合法行为的子集保持不变在 P’ 下，但在 P 下的子集（S3）中的某些行为在 P’ 下是非法的。通过实施可以识别扩大子集（S2）的检测器，行为倾斜可以捕获在未修改的策略 P 下不会被注意到的入侵者。检测速度取决于将其与合法行为区分开来的非法行为前缀的长度行为。为了遏制入侵造成的损害，我们寻求隔离从事非法行为的系统组件的方法，希望直到我们能够将其与合法行为区分开来。通过虚拟化从事非法行为的系统组件的外部环境， 警戒线可防止系统受到永久性损坏，直到可以识别非法行为为止。

在本文中，我们通过 BESIDES 说明了 PAIDS 范例的应用，BESIDES 是一种用于检测大量邮件病毒的工具。我们已经将行为倾斜和封锁技术应用于基于 NT 的 Windows 操作系统（Windows 2000 和 Windows XP）。由于行为倾斜和警戒线是通用技术，我们将它们专门用于特定类型的系统行为以实现高效实施。具体来说，我们使用行为倾斜来定制安全策略，以在受保护系统上使用某些信息项。信息项可以是任何携带信息的逻辑实体，例如文件名、电子邮件地址或二进制文件。行为倾斜是通过定制控制信息项访问的访问控制机制来实现的。同样，封锁适用于必须确保其完整性以维持系统可操作性的关键系统资源。具体来说， 我们提供动态隔离恶意进程与系统资源之间交互的机制，以便它们之间的任何未来交互都不会影响其他合法进程与资源的交互。这是通过在进程第一次访问系统资源时用虚拟资源替换原始资源来实现的。警戒线机制为每个合法进程保证其对关键系统资源的更新最终提交到实际系统资源，如果检测到恶意可执行文件，则为恶意可执行文件动态创建由虚拟系统资源组成的执行环境（警戒线）及其受害者。他们之前对实际系统资源的更新可以通过对这些资源执行恢复操作来撤销，同时他们未来的活动可以被监控和审计。根据系统资源的性质， 封锁可以通过时间封锁和空间封锁等方法实现。

本文的其余部分组织如下：第 2 节是对相关工作的简要讨论。第 3 节详细介绍了行为偏差和警戒线的工作原理。第 4 节讨论了 BESIDES 的实现，这是我们为检测大量邮件病毒而实现的原型系统。第 5 节介绍了我们用 BESIDES 进行的实验以及实验结果的分析。第 6 节讨论了一些未来的方向。

2 相关工作

病毒检测与更普遍的恶意可执行检测主题密切相关 [2、3]。传统的恶意可执行检测解决方案使用基于签名的方法，其中来自已知恶意可执行文件的签名用于识别来自它们的攻击 [4]。病毒扫描程序等安全产品就是此类应用程序的示例。基于签名的方法研究的重点之一是自动生成高质量的签名。沿着这一工作路线，Kephart 和 Arnold 开发了一种统计方法来自动提取病毒签名 [5]。在 [6] 中研究了能够从一组已知病毒中生成签名的启发式分类器。最近，舒尔茨等人。检查了数据挖掘方法如何应用于签名生成
[7] 并构建了一个可以与电子邮件服务器集成的二进制过滤器 [8]。虽然证明在检测已知恶意可执行文件方面非常有效，但基于签名的方法无法检测未知恶意可执行文件。 PAIDS 探讨了用一组不同的方法（即主动方法）解决后者的可能性。在 PAIDS 中，恶意行为的签名在行为倾斜阶段隐式生成，随后在行为监控阶段使用它们来检测执行此类行为的恶意可执行文件。

验证程序是否符合安全属性的静态分析技术也已被提议用于恶意可执行检测。其中一些专注于可疑症状的检测：Biship 和 Dilger 展示了如何动态检测文件访问竞争条件 [9]。 Tesauro 等人。使用神经网络检测引导扇区病毒 [10]。另一种方法是验证是否遵循安全编程实践。罗等。建议使用“tell-tale signs”和“program slicing”来检测恶意代码[11]。这些方法主要用作预防机制，PAIDS 中使用的方法更侧重于恶意可执行文件的检测和容忍。

沙箱等动态监控技术代表了另一种有助于检测恶意可执行文件的方法。他们实质上实施替代参考监控机制，观察软件执行并在检测到违规时强制执行额外的安全策略 [12]。在 [13-15] 中研究了可通过监视执行的安全策略的范围，在 [1] 中可以找到关于安全策略使用的更一般性讨论。 PAIDS中使用的行为监控机制采用了类似的方法。沙盒是一种通过在虚拟环境中执行进程来实施安全策略的通用机制（例如，Tron [16]、Janus [17]、Consh [18]、Mapbox [19]、SubDomain [20] 和 Systrace [21]） . Cordoning 类似于一种轻量级的沙箱机制，其覆盖范围和时间参数是可定制的。但是，cordoning 强调的是单个系统资源的虚拟化， 而传统的沙盒机制侧重于单个进程的整个执行环境（例如内存空间）的虚拟化。更重要的是，沙箱通常对入侵的容忍度很小，而封锁可以容忍关键系统资源的滥用，如第 3.2 节所述。

长期以来，欺骗工具一直被用作击败恶意入侵者的有效方式。其中，Honeypot（以及后来的 Honeynet）是一种有意部署以引诱入侵者注意的易受攻击的系统（或网络）。它们对于研究入侵者的技术和评估系统安全设置的有效性很有用 [22, 23]。传统蜜罐是专用系统，其配置方式与生产系统相同（或不如生产系统安全，具体取决于它们的使用方式），因此入侵者无法直接分辨两者之间的区别。在蜜罐的上下文中，不会对生产系统进行任何修改。虚拟蜜罐 [24, 25] 等在网络级别模拟物理蜜罐的最新进展在这方面仍然保持不变。最近，Honeypots 的概念被概括为 Honeytokens——“一种信息系统资源，其价值在于未经授权或非法使用该资源”[26]。迄今为止， 很少有实施或实验结果被报道（其中，一个实施 Honeytoken 文件的 Linux 补丁在 2004 年初可用 [27]）。Honeytoken 概念最接近 PAIDS。但是，PAIDS探索的主动方法，例如行为倾斜和警戒线，比Honeytokens更加全面和系统。我们注意到，当 Honeytoken 概念于 2003 年首次出现时，我们的 IDS 工具 BESIDES 的实施进展顺利。

最近，系统行为修改越来越受到关注。 Somayaji 和 Forrest 对异常系统调用序列应用“良性反应”[28]。入侵防御工具 LaBrea 能够通过延迟他们的通信尝试来诱捕已知入侵者 [29]。由 Williamson 等人构建的病毒节流阀。[30-32] 利用在正常电子邮件流量中发现的时间局部性，并且能够在大量邮件病毒进行大量连接尝试时减慢并识别它们。他们在入侵预防和容忍方面的成功证实了基于行为修改的方法的有效性。然而，在所有这些方法中执行的修改都不会尝试修改可能合法的行为，因为这可能会导致误报，而 PAIDS 中的行为倾斜方法更进一步，将一些合法但无关的行为转换为非法行为。在这种情况下不会引起误报。

入侵检测是一个历史悠久的研究领域 [33, 34]。传统上，IDS 分为以下几类：基于滥用的 IDS 寻找已知入侵的签名，例如允许未经授权的用户获得管理员权限的已知操作序列 [35、36]。这与本节前面讨论的基于签名的方法非常相似。基于异常的 IDS 通过识别系统和网络活动中与正常配置文件的偏差来检测入侵。这些正常配置文件可以从合法系统执行期间收集的历史审计数据中研究 [34、37-39]。基于规范的 IDS 寻找源自系统规范 [40] 的已知合法活动的签名。这些 IDS 与基于滥用的 IDS 的不同之处在于，与任何签名都不匹配的活动在前者中被视为入侵，但被视为合法活动

在后者。混合 IDS 将不同类型的 IDS 集成为子系统 [41]。可以弥补不同子系统的局限性，提高整体质量。

IDS 中设计的许多技术适用于恶意可执行检测，反之亦然。例如，为了将基于规范的方法应用于恶意可执行检测，Giffin 等人。展示了如何检测源自移动代码的恶意操作 [42]。 Christodorescu 和 Jha 提出了一种架构，即使在存在混淆的情况下也能检测恶意二进制可执行文件 [43]。所有 IDS（尤其是基于异常的 IDS）的一个共同目标是使用系统的显式或隐式属性生成配置文件，以有效区分入侵行为和正常行为。文献中研究了各种各样的系统属性，从低级网络流量统计和系统调用特征到高级 Web 访问模式和系统资源使用。 McHugh 和 Gates 报告了在网络流量中观察到的一组丰富的位置，并指出这些位置可用于区分异常行为和正常行为 [44]。他们都没有考虑为入侵检测目的修改系统安全设置的可能性，这在 PAIDS 中进行了探索。 PAIDS 中提出的主动方法通过在对入侵者可能行为的预期中扭曲安全策略来修改正常行为的定义。通过这样做，PAIDS 隐式地生成误报率低、易于理解和配置并且范围广泛的配置文件。

3 方法

3.1 行为倾斜

如图 1 所示，行为是系统活动的高级抽象，指定了它们的意图和效果，但忽略了它们执行的实际操作的许多实现细节。我们将明确指定合法（或非法）行为的安全分配部分称为合法行为集（LBS）（或非法行为集（IBS））。未有意或无意明确指定的行为的安全分配表示为未指定的行为集 (UBS)。 UBS 包括 1）用户认为与其系统安全无关的行为，或 2）用户不知道并且无意中未能指定其安全分配；默认的安全策略将应用于设置 UBS 中的行为。 行为倾斜是指对安全策略的操纵，该策略自定义 UBS 集合中行为的安全分配，这些行为默认隐式分配为合法。行为倾斜的目标是在同质系统之间的合法行为上产生足够的差异，以便使恶意可执行文件易于检测。

具体来说，行为倾斜定制了关于系统中某些信息项的使用的安全策略。行为倾斜创建了自己的访问控制机制，该机制描述了信息项的使用，因为许多

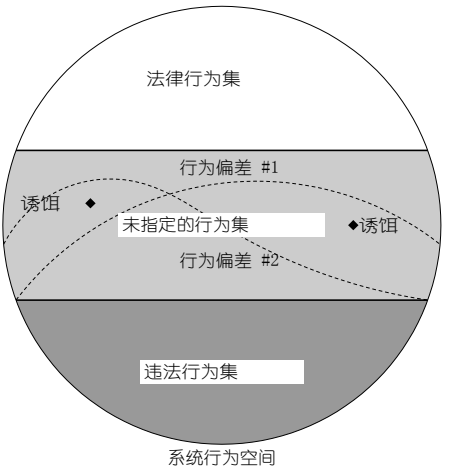


图 1. 系统状态和行为偏差

的信息项不是系统资源，因此不受本机安全机制的保护。定制是在信息域上执行的，信息域是句法相同但语义不同的信息项集。例如，目标系统中的所有文本文件构成一个文本文件的信息域。具体来说，行为倾斜减少了对默认访问权限指定的现有项的访问权限，并在访问权限减少的信息域中创建新的信息项。

图 1 显示了两种可能的行为偏差实例。我们强调，虽然不同的倾斜机制产生不同的安全设置，但它们都保留相同的 LBS。由行为倾斜产生的修改后的安全策略称为倾斜安全策略（或简称倾斜策略）。默认安全策略对用户是透明的，用户不能依赖它来指定她的意图。否则，可能需要额外的冲突解决机制。

行为倾斜完成后，信息项的使用由检测违反倾斜策略的行为监控机制监控。任何违反倾斜策略的行为都会触发入侵警报。应该注意的是，监控机制并不强制执行偏斜策略，而只是将违反偏斜策略的行为报告给负责处理所报告违规行为的更高级别的实体（例如，用户或 IDS）。

3.2 警戒线

尽管行为倾斜和监控使恶意可执行文件更容易被检测到，但在恶意可执行文件有机会做出不当行为之前，无法进行实际检测。因此，需要额外的

保护机制，以应对恶意可执行文件在最终被发现之前可能造成的任何损害。其中，系统状态的恢复是一个明显的问题。在本节中，我们将说明另一种主动方法，即警戒线，可用于恢复所选系统资源的状态。现有的系统恢复解决方案，例如从备份介质或可恢复文件系统恢复，通常执行批量恢复操作，其中包含用户最近工作的单个系统资源的最新更新可能在恢复后丢失。封锁通过单独执行恢复操作来解决此问题。

通常，封锁是一种允许系统中进程的执行环境动态部分虚拟化的机制。它在关键系统资源（CSR）上执行，系统中的对象的安全性被认为对系统的完整性和可用性至关重要（例如，可执行文件、网络服务和数据文件等）。封锁机制将 CSR（也称为实际 CSR）转换为可恢复 CSR（或封锁 CSR），可以通过动态创建虚拟 CSR（称为当前 CSR）并将其授予进程来恢复到已知安全状态打算与实际的 CSR 进行交互。当前 CSR 提供与实际 CSR 相同的界面。底层警戒机制确保在正常系统执行期间对当前 CSR 的所有更新最终应用于实际 CSR。

当行为监视器检测到恶意可执行文件时，它对所有封锁 CSR 的更新可以通过对封锁 CSR 执行相应的恢复操作来恢复，这些恢复操作将实际 CSR 恢复到已知安全状态（称为恢复 CSR）。恶意可执行文件及其受害者——它的子进程，以及访问已更新系统资源的进程
通过恶意可执行文件[1] - 继续与当前 CSR 交互，
他们在检测时使用的同一组 CSR。然而，他们的未来
对这组当前 CSR 的更新不会应用于实际的 CSR，而是要接受审计和其他入侵调查机制。未受影响的进程——除恶意可执行文件及其受害者之外的现有进程，以及所有新启动的进程——将在它们未来的交互中使用恢复的 CSR。因此，恶意可执行文件及其受害者被动态创建的执行环境（称为警戒线）封锁，该执行环境由这些当前 CSR 组成。因此，恶意可执行文件及其受害者在 CSR 上执行的操作与未受影响进程执行的操作隔离开来。

CSR根据其性质可分为可逆CSR、可延迟CSR和可替代CSR。两种封锁机制：时间封锁和空间封锁可以应用于这些 CSR，如下所述：

可恢复的 CSR 是可以撤销其更新的 CSR。例如，如果在病毒感染期间发现日志文件系统中的文件已损坏，则可以将其恢复到以前的安全状态。可恢复 CSR 的封锁可以是

[1]我们注意到，准确识别受害者可能与识别隐蔽通道一样困难。我们在这里使用的简单标准是一个合理的近似值，尽管可以设计出更准确的算法。

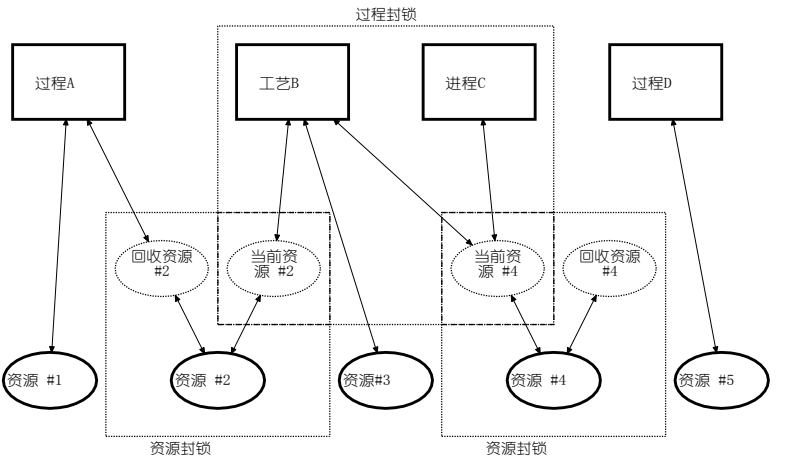


图 2. 封锁示例

通过生成一个包含所有已提交更新的撤销操作的撤销列表来实现。可恢复资源的恢复可以通过在导致 CSR 安全状态的撤销列表中携带撤销操作来执行。

及时封锁缓冲 CSR 上的操作请求并延迟它们的承诺，直到可以达到资源的安全状态。因此，它也被称为延迟承诺。 Cordoning-in-time 可以应用于可延迟的 CSR——CSR 在被访问时可以容忍一定的延迟。例如，如果事务不是实时事务（例如 SMTP 服务器），则服务于网络事务的网络资源是可延迟的。请求的延迟可以任意长，除非它超过某个事务特定的超时值。超时限制延迟的最大长度。这种约束以及其他约束（例如，由于资源可用性造成的约束）可能会使可延迟的 CSR 成为部分可恢复的 CSR；后者只能在有限的时间间隔内恢复。可延迟资源的恢复可以通过丢弃缓冲的请求直到达到安全状态来执行。 如果在用尽所有缓冲请求后找不到这样的安全状态，则 CSR 可能无法安全恢复，除非它也是可恢复的。

Cordoning-in-space 应用于可替代的 CSR——一种可以透明地由相同类型的另一个 CSR（称为其替代品）替代的 CSR。例如，文件是可替代的 CSR，因为对文件执行的任何操作都可以重定向到该文件的副本。 Cordoning-in-space 将操作请求从一个流程重定向到一个可替代的 CSR 到它的替代品。实际的 CSR 在系统执行期间保持在安全状态，并且仅当后者处于安全状态时才通过复制其替代内容来更新。可替换的 CSR 可以通过将其替换为处于安全状态时保存的实际 CSR 的副本来恢复。如果 CSR 存在多个替代品（例如，多个编写器进程），则需要进一步解决冲突，但

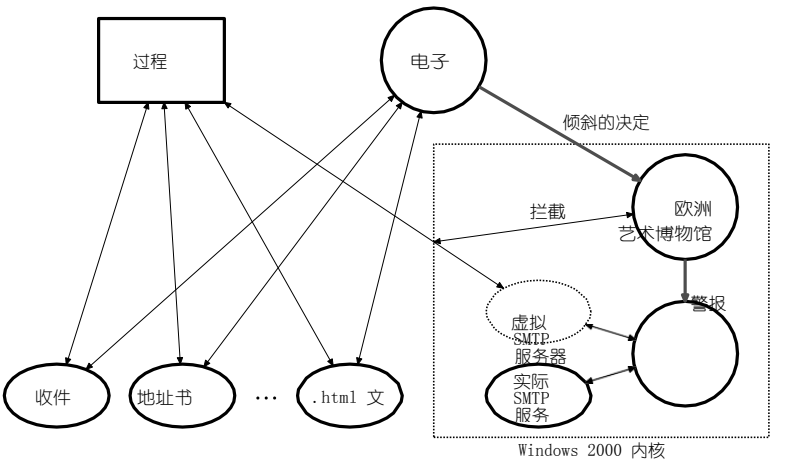


图 3. BESIDES 架构

本文不做赘述。我们在这里注意到，CSR 的封锁和恢复是可以单独执行的独立机制。

图 2 显示了对两个 CSR 进行封锁的示例：资源 #2 和资源 #4。进程 B 和进程 C 被确定为病毒感染的受害者。它们被放置在由 Current Resource #2 和 Current Resource #4 组成的警戒线中。封锁机制执行恢复操作，其中创建恢复资源 #2 和恢复资源 #4。与受害者共享 CSR 的未受影响进程（例如进程 A）继续使用恢复的资源 #2。新进程（例如进程 D）在将来请求访问资源 #4 时将获得恢复的资源 #4。剩余的系统资源，例如资源 #1、#3 和 #5，未被封锁。

4 执行

BESIDES是一个原型系统，它使用行为倾斜、行为监控和警戒线来检测未知的海量邮件病毒。如图 3 所示，BESIDES 由一个电子邮件地址域串（EADS）、一个电子邮件地址使用监视器（EAUM）和一个 SMTP 服务器 Cordoner（SSC）组成。EADS 是用户模式应用程序，EAUM 和SSC 封装在内核模式驱动程序中。BESIDES 在 Windows 2000 上实现，也可以在 Windows XP 上运行。我们选择构建基于 Windows 的系统的主要原因是 Windows 是最受攻击的系统之一，尤其是病毒攻击在那里经常出现。在 Windows 上开发和试验 BESIDES 使我们能够说明我们的方法并表明它可以在商业操作系统上实现。

4.1 电子邮件地址域串

安装 BESIDES 后，用户需要使用 EADS 来歪曲电子邮件地址域的行为，这是一个被攻击者积极利用的信息域

恶意可执行文件，例如大量邮件病毒。倾斜是通过修改电子邮件地址的使用策略来执行的。典型的 Windows 系统不限制电子邮件地址的使用。特别是，所有电子邮件地址都可以用作电子邮件发件人或收件人。 EADS 通过使某些电子邮件地址在任何本地撰写的电子邮件中不可用来更改此默认使用策略。受影响的电子邮件地址称为诱饵地址，或简称为诱饵。除非用户能够确定其中哪些是可倾斜的，否则现有电子邮件地址不会倾斜。默认情况下，EADS 允许所有受试者完全使用（即“全部允许”）除诱饵之外的任何电子邮件地址。它认为任何诱饵的使用都违反了倾斜的电子邮件地址使用政策（即“全部拒绝”）EAUM 将在检测到在本地发送的任何电子邮件消息中使用诱饵时发出入侵警报。此外，EADS 设置了对某些电子邮件域（例如 foo.com in alice@foo.com）在诱饵中“拒绝所有”。
这使得倾斜更加通用，甚至病毒操纵电子邮件
使用它们之前的地址（例如，Bugbear）也可以被检测到。

电子邮件地址域的倾斜需要在域中创建足够的不可预测性，这样病毒就不可能通过简单地查看电子邮件地址本身来判断电子邮件地址是否合法。 EADS 在其倾斜过程中同时使用启发式方法和随机化方法，即，它根据用户指定的一组诱饵或随机生成诱饵地址。

具体来说，EADS 在以下文件类型中创建电子邮件诱饵：电子邮箱、
例如.eml 文件；基于文本的文件，例如.HTM、.TXT 和.ASP 文件等；和二进制文件，例如 .MP3、.JPG 和 .PDF 文件等。通过导入使用诱饵地址作为发件人的诱饵电子邮件信息来扭曲电子邮箱。基于文本的文件被新创建的包含诱饵的语法有效文件所扭曲。二进制文件与基于文本的文件倾斜相同的文本文件倾斜，但具有
修改后的扩展名。此类诱饵二进制文件格式无效，无法由对这些文件进行操作的合法应用程序进行处理。这不会造成问题，因为这些诱饵本来就不应该被他们使用。然而，许多大型邮件病毒仍然能够在这些文件中发现诱饵，因为它们通常通过“暴力”访问它们并忽略它们的格式，例如，通过使用简单的字符串匹配算法搜索电子邮件地址。默认情况下，所有这些诱饵都放置在常用的系统目录中，例如“C:\”和“我的文档”。用户是
允许选择她喜欢的其他目录。

4.2 电子邮件地址使用监视器

BESIDES 使用系统调用插入来实现 EAUM（以及 SSC）。系统调用插入是一种允许拦截系统调用调用的通用技术，已在许多地方广泛使用[45、46、20、47、19、17、48]。 Windows 2000 有两组系统调用 [49]：Win32 应用程序编程接口（API），Windows 应用程序的标准 API，作为用户模式动态链接库（DLL）实现。本机 API（或本机系统调用）在内核中实现，并且是

通过 ntdll.dll 中的虚拟用户模式函数 thunk 导出到用户模式模块。用户模式系统 DLL 使用本机 API 来实现 Win32 API。 Win32 API 要么在用户模式 DLL 中实现，要么映射到一个
（或一系列）原生 API。 Windows 2000 在内核中实现了 TCP/IP 协议栈 [50]。在本机系统调用接口上，应用程序级协议行为是直接可观察的，因此可以被有效地监控。此接口不存在传输层协议特定数据（例如，TCP/UDP 标头）。这节省了解析、分析它们然后重建协议数据流状态所需的额外时间，这在典型的基于网络的拦截器中是不可避免的。

EAUM 在内核模式下运行并监控 SMTP 会话中电子邮件地址的使用。 SMTP 会话由进程和 SMTP 服务器之间的所有 SMTP 流量组成。它以“HELO”（或“EHLO”）命令开始，通常以“QUIT”命令结束。 EAUM 在系统启动时的 BESIDES 初始化期间向系统调用插入模块（也称为 BESIDES 引擎）注册一组系统调用过滤器。它使用从 SMTP 协议 [51] 派生的 SMTP 自动机来模拟两者的进展
本地 SMTP 客户端和远程 SMTP 服务器。 BESIDES 中的 SMTP 自动机在所有 SMTP 会话之间共享。 BESIDES 引擎拦截传输网络数据的本机系统调用，并调用 EAUM 注册的系统调用过滤器。这些过滤器从网络流量中提取 SMTP 数据，解析它们以生成 SMTP 令牌，然后在监视相应 SMTP 会话的 SMTP 自动机中执行状态转换。每个 SMTP 会话在 EAUM 中由 SMTP 上下文表示，每次特定 SMTP 会话取得进展时，它作为参数传递给 SMTP 自动机。当 SMTP 自动机进入相应状态时，可以监视 SMTP 会话中电子邮件地址的使用。具体来说，EAUM 寻找明确使用电子邮件地址的 SMTP 命令（即“MAIL FROM: ”和“RCPT TO: ”）并根据偏斜验证这些用法
EADS 指定的电子邮件地址使用政策。如果发现任何违规行为，
EAUM 通过入侵警报通知 SSC，因为任何诱饵地址都不应用作收件人或发件人。使用合法的电子邮件地址不会触发任何警报，因为 EADS 允许使用它们。这种监控方法的一个优点是，携带自己的 SMTP 客户端的病毒会受到检测，而更高级别的干预（例如 Win32 API 包装器）是可以绕过的。 SMTP 服务器包装形式的误用检测机制未被使用，因为病毒可能会选择非本地管理的开放 SMTP 中继。

4.3 SMTP 服务器 Cordoner

除了检测大量邮件病毒外，BESIDES 还尝试保护 CSR（此处为 SMTP 服务器）免受它们可能的滥用。 SMTP 服务器是可延迟的 CSR，因为电子邮件不被视为实时通信机制，并且电子邮件用户可以容忍一定程度的延迟。它也是弱可恢复的（我们指的是传递消息的损坏

包含病毒可以通过在稍后检测到病毒时向收件人发送后续警告消息来缓和）。每当进程启动 SMTP 会话时，SSC 都会识别它请求的 SMTP 服务器，并为其分配相应的虚拟 SMTP 服务器（当前 SMTP 服务器）。 Delayedcommitment 然后用于缓冲进程发送到虚拟 SMTP 服务器的 SMTP 消息。 SSC 也在内核模式下运行，并与 EAUM 共享相同的 SMTP 自动机。

具体来说，SSC 拦截 SMTP 命令并在内部对其进行缓冲。然后它会组成一个肯定的回复，并让 BESIDES 引擎将其转发给 SMTP 客户端，表明命令成功。收到这样的回复后，SMTP 客户端会认为上一个命令成功，并继续执行下一个 SMTP 命令。 SSC 实质上创建了一个虚拟 SMTP 服务器供进程交互。 SMTP 消息可以延迟的最长时间由封锁期决定——用户指定的超时值小于平均用户可容忍的延迟，以及用户指定的最大延迟消息数阈值。当 SMTP 消息延迟超过封锁期或延迟消息数超过消息数阈值时，SMTP 消息将被传递（提交）到实际的 SMTP 服务器。发送 SMTP 消息后， SSC 在 SMTP 服务器特定日志中创建相应的日志条目（撤销记录），其中包含时间、主题和传递消息的收件人。当收到入侵警报时，SSC 将执行恶意活动的进程识别为恶意可执行文件。然后，它根据 CSR 访问历史和进程层次结构确定受害者集合，即访问由该进程更新的 CSR 的所有进程及其所有子进程都被标记为受害者。在此之后，SSC 在他们更新的所有封锁 CSR 上启动恢复操作。如果拥有 SMTP 会话的进程是受害者之一或恶意可执行文件本身，则不会提交来自该 SMTP 会话的缓冲消息；相反，他们都被隔离了。 之前提交的所有消息都被视为可疑消息，并且 SSC 使用保存在传递日志条目中的信息向其收件人发送警告消息作为弱恢复机制。由于发送到 SMTP 服务器的 SMTP 消息是独立的，因此接收它们的顺序不会影响 SMTP 服务器的正确操作 [52]。因此，即使在恢复操作期间丢失了某些邮件，实际的 SMTP 服务器也可以保持安全状态。同时，未受影响的进程不知道这种恢复，并且可以继续进行，就好像没有发生入侵一样。

5 实验结果

我们用在野外收集的病毒在 BESIDES 上进行了一系列实验。这些实验是在由两台机器组成的封闭局域网（LAN）上执行的：服务器和客户端。 BESIDES 安装在客户端上，其 EADS 在所有实验中的设置方式相同。服务器通过向客户端提供必要的网络环境来模拟一个典型的网络环境。

表 1. BESIDES 的有效性（BESIDES SSC 配置为在这些实验期间拦截最多 10 条 SMTP 消息，并且每条 SMTP 消息最多拦截 60 秒。Outlook Express 手册是手动倾斜的）

病毒		怪物	欢乐时光	克莱兹	我的末日
客户	检测到？	是的	是的	是的	是的
	检测时使用的诱饵	地址节	。.htm	。.html	。.htm
	延迟消息被隔离？	是的	是的	是的	是的
服务器	检测到？（通过杀毒软件）	是的	不	是的	不
	收到 SMTP 消息？	是的	不	不	不

工作服务，例如 DNS、路由、SMTP、POP3 和远程访问服务（RAS）。服务器还安装了杀毒软件（如Symantec Antivirus）和网络取证工具（如Ethereal、TcpDump等）。可以从这些工具的输出以及服务日志中收集病毒传播的证据。

本节的其余部分介绍了两组实验结果。首先，我们展示了使用几种实际病毒对 BESIDES 进行实验时的结果。这些结果证明了行为偏差和警戒线在现实环境中应用时的有效性。第二组结果显示了在正常系统应用程序的正常系统执行期间观察到的性能开销，包括在本机系统调用界面和命令行界面观察到的延迟。正如我们预期的那样，即使我们没有在 BESIDES 中执行任何优化，开销也在合理范围内。

5.1 有效性实验

我们对四种实际病毒——BugBear [53]、Haptime [54]、Klez [55] 和 MyDoom [56]——进行的实验结果如表 1 所示。在所有实验中，BESIDES 能够检测到被感染的病毒实验过。尽管所有这些病毒都试图在客户端计算机上收集电子邮件地址，但它们的方法各不相同，而且当 BESIDES 检测到它们时它们实际使用的诱饵地址也各不相同。在所有实验中，BESIDES SSC 拦截了多条病毒发送的 SMTP 消息并成功隔离。但是，在使用 Bugbear 进行实验期间，发现一些携带病毒的消息在检测到病毒之前就已发送。从 Bugbear 的 SMTP 服务器收到的电子邮件消息中，我们发现 Bugbear 实际上在发送之前操纵了发件人或收件人地址。具体来说， 它通过组合一个电子邮件地址的用户字段和另一个电子邮件地址的域字段来形成一个新的电子邮件地址。 BESIDES 最初无法检测到这些消息，因为它只将用户名字段和域字段的匹配视为可接受的
匹配一个诱饵。然后它将这些消息提交给 SMTP 服务器[2]。和

[2]后来通过在电子邮件域上创建额外的电子邮件使用策略并在 EADS 中创建诱饵域来修复此漏洞。有关详细信息，请参阅第 4.1 节。

在我们的实验设置中，平均十分之二的此类邮件被发现提交给 SMTP 服务器。我们注意到实际数字随倾斜操作而变化。在我们的一项实验中，在 BESIDES 检测到病毒之前提交了两条这样的消息。服务器上的防病毒软件检测到 Bugbear 和 Klez，因为它们也通过网络共享传播，这种机制在所有这些实验中都没有被 BESIDES 封锁。当 Haptime 试图感染本地文件并从中收集电子邮件地址时，我们观察到它对硬盘的大量访问。这一切都发生在病毒开始执行大量邮件操作之前。这表明如果 BESIDES 也倾斜文件访问权限，则可以更快地检测到 Haptime。

Mydoom 的爆发晚于实验中使用的 BESIDES 版本完成。因此，BESIDES 在用它进行实验时并不知道该病毒。当 BESIDES 尝试使用放置在 .htm 文件中的诱饵电子邮件地址发送邮件时，它成功地检测到该病毒。这证明了 BESIDES 检测未知病毒的能力。

由于某些病毒使用概率方法（例如 Klez），它们在不同实验中的行为可能不同。因此，此处显示的结果只是一种可能的行为。此外，由于某些病毒使用本地时间来决定是否执行（或不执行）特定操作（例如，如果 MyDoom 执行 DDoS 攻击则不会执行大量邮件，这取决于系统时间），我们手动更改客户端的系统时间以加速病毒激活大量邮件。我们强调这样做是为了加快我们的实验，并且不需要系统时间操作来影响生产中的检测。

5.2 性能实验

总体系统开销。表 2 显示了在正常系统执行期间在三个本机系统调用接口上观察到的系统调用开销的统计结果。前置拦截器和后置拦截器两个拦截器，用于在实际系统调用执行前后进行拦截操作。表中显示的三个本机系统调用代表三个封锁会话阶段。

调用 NtCreateFile() 来创建或打开文件（包括套接字）[49]。 BESIDES 拦截它以便识别 SMTP 会话并创建它们相应的 SMTP 上下文。这是 SMTP 会话的设置阶段。后拦截器用于执行这些操作。尽可能
从表 2 可以看出，开销只是实际系统调用的一小部分。

NtDeviceIoControlFile() 对表示设备的文件对象执行 I/O 控制操作 [49]。网络数据包也使用此系统调用进行传输。 BESIDES 拦截此系统调用，以便它可以检查 SMTP 会话控制和数据消息。这是一个检查阶段
SMTP 会话。在拦截期间，解析 SMTP 数据，生成 SMTP 令牌，并更新 BESIDES EAUM SMTP 自动机的状态。前置拦截器和后置拦截器分别处理发送数据和接收数据。观察到的开销只是交流的一小部分

表 2. 在本机系统调用接口处观察到的 BESIDES 系统调用开销。所有数字均根据直接从客户端计算机（具有 128MB 内存的 Pentium III 730MHz CPU）检索的 CPU 性能计数器值计算得出

本机系统调用	关闭()	NtCreateFile()	NtDeviceIoControlFile()
总执行时间	283076	1997247	16110683
预拦截时间	108520	24033	21748
实际系统调用时间	32960	1471367	15791127
后拦截时间	23588	371826	156350
系统调用-置入时间	118008	130021	141457
开销（%）	758.85%	35.74%	2.02%

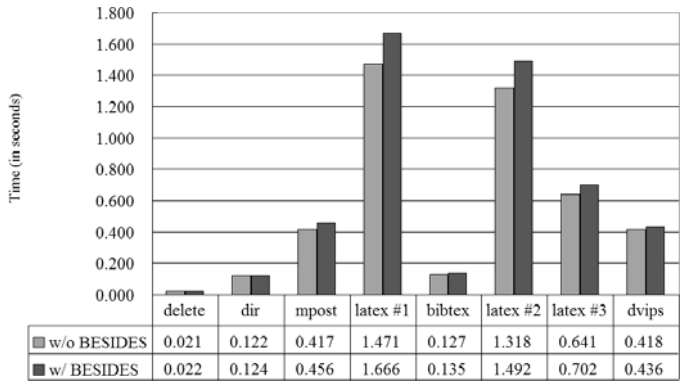
tual 系统调用，因为实际的系统调用会导致昂贵的硬件 I/O 操作。

NtClose() 由用户模式进程调用以关闭对象 [49] 的句柄。 BESIDES 拦截它以终止 SMTP 会话。这称为该 SMTP 会话的终止阶段。预拦截器释放系统

用于监视此 SMTP 会话的资源。实际的系统调用是一个轻量级的系统调用，花费的时间比其他两个要少得多。观察到的开销支配实际的系统调用。然而，由于设置和终止阶段在 SMTP 会话的生命周期中只执行一次，因此它们相对较高的成本可以通过更快的检查阶段来分摊。最后，应该注意的是，在所有系统调用拦截中都存在一种不同类型的开销，即系统调用干预开销。此开销说明了每个被拦截的系统调用必须支付的强制性开销，包括额外的系统调用查找时间和内核堆栈设置时间等。但是，对于那些未被拦截的系统调用，创建了优化的快捷方式

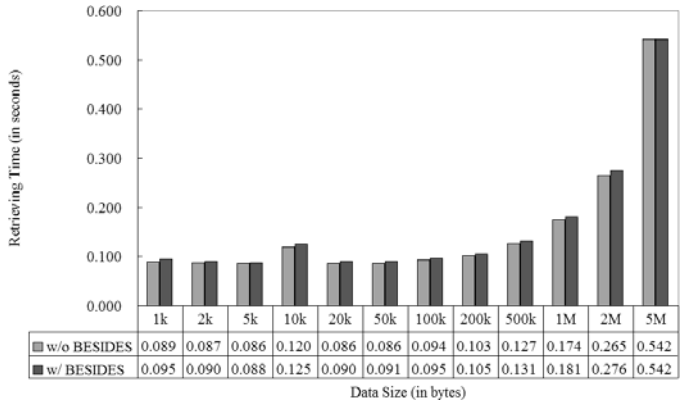
在拦截例程中，以便产生尽可能少的开销。

特定于应用程序的开销。我们还测量了安装 BESIDES 的客户端计算机上几个应用程序的时间开销。图 4 显示了在一系列应用程序中观察到的开销（10 次单独运行的平均值），这些应用程序从其 .tex 源文件编译本文草稿的后记版本。执行的应用程序包括删除（清理
目录），dir（列出目录内容），mpost（构建图形 .eps 文件），latex #1（latex 的第一次运行），bibtex（准备参考书目项目），latex #2（latex 的第二次运行）， latex #3（第三次 latex 运行）和 dvips（将 .dvi 文件转换为 postscript 文件）。 CPU 密集型和
存在 I/O 密集型应用程序，该系列可视为代表
发送不需要网络访问的应用程序。这些应用程序仅受 BESIDES 引擎引入的本机系统调用插入开销的影响。在这些实验中观察到的平均时间开销约为 8%。最高的增长（大约 13%）出现在 latex #1 和 latex #2 中，它们都执行重要的 I/O 操作。最低的增长（大约 1.5% 和 3.3%）分别发生在 dir 和 delete 中。这些是shell命令



	delete	dir	mpost	latex #1	bibtex	latex #2	latex #3	dvips
w/o BESIDES	0.021	0.122	0.417	1.471	0.127	1.318	0.641	0.418
w/ BESIDES	0.022	0.124	0.456	1.666	0.135	1.492	0.702	0.436

图 4. Latex 应用系列的时间开销



	1k	2k	5k	10k	20k	50k	100k	200k	500k	1M	2M	5M
w/o BESIDES	0.089	0.087	0.086	0.120	0.086	0.086	0.094	0.103	0.127	0.174	0.265	0.542
w/ BESIDES	0.095	0.090	0.088	0.125	0.090	0.091	0.095	0.105	0.131	0.181	0.276	0.542

图 5. 命令行 Web 客户端的时间开销

执行需要很少系统调用的简单任务。我们注意到 CPU 密集型应用程序（例如 dvips）的开销要小得多（例如，dvips 为 4.3%）。这些结果表明 I/O 密集型应用程序可能比 CPU 密集型应用程序承受更多的开销。它们符合我们的预期，因为基本上所有 I/O 操作都是由本机系统调用执行的，因此会受到拦截，而 CPU 密集型操作包含更高百分比的用户模式代码，几乎不会进行系统调用。

图 5 显示了基于命令行的 Web 客户端从本地 Web 服务器检索数据文件时观察到的开销。本实验中使用的数据文件大小从 1kB 到 5MB 不等。尽管 Web 客户端使用 HTTP 检索这些数据文件，但其网络流量仍受 BESIDES 中 SMTP 过滤器的检查。系统调用干预开销相对较小，因为 Web 客户端在整个过程中执行的其他系统调用很少。观察到的两个最大开销约为 6.3% 和 5%（分别为 1kB 和 50kB）。两个最小的开销是 0% 和 1.8%（分别为 5MB 和 100kB）。平均开销约为 3.4%，接近

到 2.02%，在 NtDeviceIoControlFile() 接口处观察到的开销。这证实了我们之前的推测，即会话设置阶段和会话终止阶段看似高的增长可以通过会话检查阶段的低开销来分摊。

6 结论

本文介绍了一种通用范例，PAIDS，用于通过主动方法进行入侵检测和容忍。我们展示了我们在行为倾斜和封锁方面的工作，这两种主动方法可用于在系统中产生不可预测性，从而更容易检测到未知的恶意可执行文件。这种方法与现有方法的不同之处在于，这种主动系统可以预测来自恶意可执行文件的攻击，并通过修改系统的安全策略提前为它们做好准备。

PAIDS 的优势在于它可以检测到原始中尚未发现的入侵者，而且 PAIDS 的误报率非常低。 BESIDES 是一个使用 PAIDS 方法的概念验证原型，它可以在许多方面得到增强。明显的增强包括用于附加信息域（例如，文件访问串）和系统资源（例如，文件系统 cordoner）的 skewers 和 cordoners。 BESIDES SSC 可以通过更通用的处理和恢复方案来增强，以应对一般的恶意可执行文件。我们也有兴趣设计更主动的方法。总的来说，我们想研究通过封锁系统用于与外部环境交互的所有协议，我们可以在多大程度上系统地封锁部分甚至整个系统。

最后，我们想指出，我们研究的主动方法只是解决检测未知恶意可执行文件的一般问题的一部分。仅配备主动技术的系统仍然容易受到新型恶意可执行文件的攻击，这些可执行文件不会滥用任何倾斜的信息域或以更微妙的方式滥用系统，例如从合法应用程序窃取 CPU 周期。 PAIDS 不是万能药，因为它仅对传播感染途径或造成损害的机制已明确表征的病毒有效。由不同领域的技术组成的综合解决方案显然更有效，因为每种技术的弱点都可以通过其他技术的优势来弥补。我们想探索如何将主动方法与此类混合解决方案相结合。

参考文献

1. Blakley, B.：皇帝的旧盔甲。载于：ACM 新安全范式研讨会论文集，加利福尼亚州箭头湖（1996 年）
2. Cohen, F.：计算机病毒：理论与实验。计算机和安全 **6**(1987) 22–35
3. Chess, D.M., White, S.R.：一种无法检测到的计算机病毒。载于：2000 年病毒公报国际会议记录，佛罗里达州奥兰多（2000）

4. Gryaznov, D.: 2000 年度扫描仪：启发法。载于：第 5 届病毒公报国际会议论文集，马萨诸塞州波士顿（1999）225–234
5. Kephart, J.O., Arnold, W.C.: 计算机病毒签名的自动提取。载于：第四届病毒公报国际会议论文集，英格兰阿宾登（1994）178–184
6. Arnold, W., Tesauro, G.: 自动生成的 Win32 启发式病毒检测。载于：2000 年病毒公报国际会议记录，佛罗里达州奥兰多（2000）
7. Schultz, M.G.、Eskin, E.、Zadok, E.、Stolfo, S.J.: 用于检测新恶意可执行文件的数据挖掘方法。载于：2001 年 IEEE 安全与隐私研讨会论文集，加利福尼亚州奥克兰（2001）38–49
8. Schultz, M.G.、Eskin, E.、Zadok, E.、Bhattacharyya, M.、Stolfo, S.J.: MEF：恶意电子邮件过滤器 — 一种检测恶意 Windows 可执行文件的 UNIX 邮件过滤器。在：年度 USENIX 技术会议论文集，FREENIX Track，波士顿，马萨诸塞州（2001）245–252
9. Bishop, M.、Dilger, M.: 检查文件访问中的竞争条件。计算系统 9（1996）131–152
10. Tesauro, G.、Kephart, J.、Sorkin, G.: 计算机病毒识别的神经网络。 IEEE 专家 11（1996）5–6
11. Lo, R.W., Levitt, K.N., Olsson, R.A.: MCF：恶意代码过滤器。计算机和安全 14（1995）541–566
12. Anderson, J.P.: 计算机安全技术规划研究。技术报告，ESD-TR-73-51，美国空军电子系统部指挥和管理系统副主任，总部电子系统部（AFSC），马萨诸塞州贝德福德（1972）
13. Viswanathan, M.: 软件系统运行时分析的基础。宾夕法尼亚大学博士论文（2000）
14. Hamlen, K.W.、Morrisett, G.、Schneider, F.B.: 执行机制的可计算性类。技术报告，TR 2003-1908，康奈尔大学计算机科学系（2003）
15. Bauer, L.、Ligatti, J.、Walker, D.: 更可执行的安全策略。载于：计算机安全基础，丹麦哥本哈根（2002）
16. Berman, A.、Bourassa, V.、Selberg, E.: TRON：UNIX 操作系统的进程特定文件保护。载于：1995 年 USENIX 技术会议记录，新奥尔良，路易斯安那（1995）165–175
17. Goldberg, I.、Wagner, D.、Thomas, R.、Brewer, E.A.: 不受信任的助手应用程序的安全环境。载于：第 6 届 USENIX 安全研讨会论文集，加利福尼亚州圣何塞（1996）
18. Alexandrov, A.、Kmiec, P.、Schauser, K.: Consh：互联网计算的受限执行环境。载于：美国路易斯安那州新奥尔良年度 USENIX 技术会议论文集（1998 年）
19. Acharya, A., Raje, M.: MAPbox：使用参数化行为类来限制不受信任的应用程序。载于：第 9 届 USENIX 安全研讨会论文集，科罗拉多州丹佛市（2000）
20. Cowan, C.、Beattie, S.、Kroah-Hartman, G.、Pu, C.、Wagle, P.、Gligor, V.: 子域：Parsimonious Server Security。载于：第 14 届系统管理会议论文集，路易斯安那州新奥尔良（2000）
21. Provos, N.: 使用系统调用策略提高主机安全性。载于：第 12 届 USENIX 安全研讨会论文集，华盛顿特区（2003）257–272
22. 蜜罐、入侵检测、事件响应：http://www.honeypots.net/.

23. 蜜网项目：http://project.honeynet.org/.
24. Provos, N.：虚拟蜜罐框架。载于：第 13 届 USENIX 安全研讨会论文集，加利福尼亚州圣地亚哥（2004）
25. Jiang, X., Xu, D.：Collapsar：一种基于虚拟机的网络攻击拘留中心架构。载于：第 13 届 USENIX 安全研讨会论文集，加利福尼亚州圣地亚哥（2004）
26. Spitzner, L. : Honeytokens : 另一个蜜罐 (2003)http://www.securityfocus.com/infocus/1713.
27. Pontz, B.：蜜标。 CERT 蜜罐档案（2004） http://cert.uni-stuttgart.de/archive/honeypots/2004/01/msg00059.html.
28. Somayaji, A., Forrest, S.：使用系统调用延迟的自动响应。载于：第 9 届 USENIX 安全研讨会论文集，科罗拉多州丹佛市（2000）
29. LaBrea Sentry IPS：下一代入侵防御系统： http://www.labreatechnologies.com/.
30. Williamson, M.M.：节流病毒：限制传播以打败恶意移动代码。载于：第 18 届年度计算机安全应用会议论文集，内华达州拉斯维加斯（2002 年）
31. Twycross, J., Williamson, M.M.：实施和测试病毒控制。载于：第 12 届 USENIX 安全研讨会论文集，华盛顿特区（2003）
32. Williamson, M.M.：电子邮件病毒节流阀的设计、实施和测试。载于：第 19 届年度计算机安全应用会议论文集，内华达州拉斯维加斯（2003 年）
33. Anderson, J.P.：计算机安全威胁监测和监视。技术报告，James P. Anderson 公司，宾夕法尼亚州华盛顿堡（1980 年）
34. Denning, D.E.：入侵检测模型。在：IEEE 软件工程汇刊。 SE-13 卷：(2)。(1987) 222–232
35. Porras, P.A., Kemmerer, R.A.：渗透状态转换分析——一种基于规则的入侵检测方法。在：第 8 届年度计算机安全应用大会，德克萨斯州圣安东尼奥(1992) 220–229
36. Kumar, S.：计算机入侵的分类和检测。普渡大学博士论文（1995）
37. Lunt, T.F.：检测计算机系统中的入侵者。在：审计和计算机技术会议论文集。(1993)
38. Javitz, H.S., Valdes, A.：NIDES 统计组件：描述和论证。技术报告，SRI International，计算机科学实验室，加利福尼亚州门洛帕克（1993 年）
39. Anderson, D.、Lunt, T.F.、Javitz, H.、Tamaru, A.、Valdes, A.：使用 NIDES 的统计组件检测异常程序行为。技术报告，SRI-CSL-95-06，SRI International，计算机科学实验室，加利福尼亚州门洛帕克（1995 年）
40. Wagner, D., Dean, D.：通过静态分析进行入侵检测。载于：2001 年 IEEE 安全与隐私研讨会论文集，加利福尼亚州奥克兰（2001）156–169
41. Neumann, P.G., Porras, P.A.：迄今为止使用 EMERALD 的经验。在：第 1 届 USENIX 入侵检测和网络监控研讨会论文集，加利福尼亚州圣克拉拉（1999）73–80
42. Giffin, J.T.、Jha, S.、Miller, B.P.：检测被操纵的远程呼叫流。载于：第 11 届 USENIX 安全研讨会论文集，加利福尼亚州旧金山（2002）

43. Christodorescu, M., Jha, S.: 检测恶意模式的可执行文件静态分析。载于：第 12 届 USENIX 安全研讨会论文集，华盛顿特区（2003）
44. McHugh, J., Gates, C.: 局部性：思考正常行为和外来威胁的新范式。载于：ACM 新安全范式研讨会论文集，瑞士阿斯科纳（2003）
45. Jain, K., Sekar, R.: 用于系统调用介入的用户级基础设施：入侵检测和限制的平台。在：网络和分布式系统安全研讨会论文集，加利福尼亚州圣地亚哥（2000）19–34
46. Ghormley, D.P.、Petrou, D.、Rodrigues, S.H.、Anderson, T.E.: SLIC：商品操作系统的可扩展系统。载于：年度 USENIX 技术会议论文集，路易斯安那州新奥尔良（1998）39–52
47. Fraser, T.、Badger, L.、Feldman, M.: 使用通用软件包装器强化 COTS 软件。载于：IEEE 安全与隐私研讨会论文集，加利福尼亚州奥克兰（1999）2–16
48. Ko, C.、Fraser, T.、Badger, L.、Kilpatrick, D.: 使用软件包装器检测和对抗系统入侵。载于：第 9 届 USENIX 安全研讨会论文集，科罗拉多州丹佛市（2000）
49. Nebbett, G.: Windows NT/2000 本机 API 参考。第一版。麦克米伦技术出版社（2000）
50. Solomon, D.A., Russinovich, M.E.: Inside Microsoft Windows 2000。第 3 版。微软出版社（2000）
51. Postel, J.B.: 简单邮件传输协议（1982）http://www.ietf.org/rfc/rfc0821.txt.
52. Russell, D.L.: 通信过程系统中的状态恢复。 IEEE 软件工程汇刊 SE6（1980）133–144
53. Liu, Y., Sevcenco, S.: W32.bugbear@mm。赛门铁克安全响应中心（2003）http://securityresponse.symantec.com/avcenter/venc/data/w32.bugbear@mm.html。
54. Sevcenco, S.: Vbs.haptime.a@mm。赛门铁克安全响应中心（2004）http://securityresponse.symantec.com/avcenter/venc/data/vbs.haptime.a@mm.html。
55. Gudmundsson, A., Chien, E.: W32.klez.e@mm。赛门铁克安全响应中心（2003）http://securityresponse.symantec.com/avcenter/venc/data/w32.klez.e@mm.html。
56. Ferrie, P., Lee, T.: W32.mydoom.a@mm。赛门铁克安全响应中心（2004）http://securityresponse.symantec.com/avcenter/venc/data/w32.novarg.a@mm.html。