Rhea：利用云中的格式感知验证检测权限提升型规避勒索软件攻击

Beom Heyn Kim*
beomheynkim@hanyang.ac.kr 汉阳大学ERICA校区 安山，京畿道，韩国

Seok Min Hong
hsmint@hanyang.ac.kr 汉阳大学ERICA校区 安山，京畿道，韩国

Mohammad Mannan
m.mannan@concordia.ca 康考迪亚大学 蒙特利尔，魁北克，加拿大

arXiv:2601.18216v1 [cs.CR] 26 Jan 2026

摘要

勒索软件变种越来越多地将权限提升与复杂的规避策略相结合，例如间歇性加密、低熵加密和模仿攻击。这类强大的勒索软件变种，即特权升级型规避勒索软件（PEER），能够通过篡改或混淆输入/输出轨迹，击败现有的、依赖输入/输出模式分析的解决方案。与此同时，随着加密数据量减少，传统的基于统计内容的检测会因采样噪声而变得不可靠。我们提出*Rhea*，一种云卸载式勒索软件防御系统，它分析复制的数据快照，即所谓的变异快照。*Rhea*引入格式感知验证，通过验证文件格式在语法和语义上的正确性，而非依赖基于统计或熵的指标。通过将文件格式规范用作检测不变量，*Rhea*即使在攻击者提升后的权限下，也能可靠地识别细粒度且隐蔽的加密。评估结果表明，*Rhea*的性能显著优于现有方法，证明了其在应对现代勒索软件威胁方面的实际有效性。

关键词

勒索软件检测、规避型攻击、文件格式验证、云安全

预印本——评审中（*2026*年*1*月）

1 引言

勒索软件日益在各个领域造成严重的运营和财务损失。2024年，Ascension是美国最大的医疗保健提供商之一，遭遇了一次勒索软件攻击，导致临床运营中断，并使560万名患者和员工的记录遭到泄露 [54]。Medusa等勒索软件即服务 (RaaS) 团伙的兴起进一步加剧了这一趋势 [38]。到2025年，Medusa已被用于入侵300多名受害者，涉及医疗保健、教育、保险及其他关键领域。更糟的是，现代勒索软件行动越来越依赖特权执行技术，例如自带易受攻击驱动程序 (BYOVD) 攻击 [47, 52]。攻击者一旦获得根权限，便不难关闭安全监控程序。与此同时，高级勒索软件攻击往往会采用复杂的规避策略，包括间歇性加密 [49, 62], 、低熵加密和模仿攻击 [63]。这些趋势共同催生了一类权限提升型规避性勒索软件 ***(PEER)***。

遗憾的是，现有的勒索软件防御措施对PEER攻击者的防护能力有限。以往的方案往往严重依赖用户级、内核级、虚拟机监控器级或硬件级的I/O模式分析 [5, 8, 12, 18, 20, 22, 26, 32, 37, 46, 51, 62]。然而，在PEER攻击下，I/O模式分析基本不再可靠，因为特权攻击者可以关闭监控程序，也可以利用混淆的低级轨迹使其失效。

一些系统采用基于内容的检查 [18, 32, 37, 62]。然而，要将这些机制作为终端用户设备上的唯一一线防御措施，颇具挑战。基于内容的检测器会将文件划分为数据块，并使用熵、χ^2统计量或机器学习等统计方法分别分析每个数据块。然而，这些检测器存在一个根本性的统计局限：随着检查窗口变小，经验字节分布会受到采样噪声的强烈干扰。在采样噪声的影响下，明文、压缩数据和密文的可观测统计特征会迅速趋同，因而很难可靠地区分。攻击者利用统计方法的这一固有局限，只加密文件中散布的极短片段，即可规避基于内容的检测 [34], ，我们将此称为细粒度部分加密。

为克服以往基于内容的检测方法的局限性，我们提出格式感知验证 （FAV）。它分析的不是原始字节统计特征，而是底层文件格式的句法和语义不变量。FAV基于一项基本观察：文件格式可以作为有效数据状态转换的正确性规范。更具体地说，文本、ZIP、OOXML、PDF文件等高价值用户文件的良性变异，始终应符合其文件格式丰富的句法和语义结构。即使是很小的加密片段，通常也会违反这些不变量，产生格式错误的文本、损坏的标记或毫无意义的内容。因此，FAV能够可靠地识别细粒度部分加密，而以往依赖统计方法的基于内容的检测方法无法发现这种加密。

在本研究中，我们开发了*Rhea*，这是一种勒索软件防御架构，可将检测和恢复工作从终端设备转移到云端。云端卸载具有多项优势：(1) 与遭到入侵的主机隔离，包括权限提升攻击；(2) 拥有充足的存储空间，可长期保留数据快照；(3) 具备强大的计算能力，可进行细粒度的内容检查；(4) 减轻终端设备的性能负担和管理负担。*Rhea*将PEER限制在虚拟机内，并将虚拟机用作沙箱，防止勒索软件逃离客户机环境。*Rhea*定期执行称为计划检查点的流程，在此期间，它会在指定时间（例如夜间验证时段或预期空闲时段）将这些虚拟机中的块级数据快照（称为变异快照）复制到云端。

*通讯作者

Rhea 随后仅使用格式感知验证（FAV）分析变异快照中捕获的内容，而不依赖短暂且可能不可信的 I/O 流或粗粒度的基于内容的统计启发式方法。

实现与结果。 我们通过结合受控的部分加密模拟和涵盖多种文件格式的真实世界数据集，对 *Rhea* 原型实现进行了评估。在广泛的细粒度加密模式下——包括 Skip-Step 和微粒度攻击，直至单字节修改——*Rhea* 对 TXT、PDF、OOXML 和 ZIP 等结构化文件格式的检测准确率接近 100%。在这些格式中，即使刻意将统计信号降至最低，格式感知验证（FAV）仍能持续检测到加密导致的违规。总体而言，在细粒度部分加密情况下，*Rhea* 的表现大幅优于基于熵和 χ^2 的检测器，表明规范级正确性比传统统计方法提供的信号更强、更可靠。

贡献。 据我们所知，*Rhea* 是首个将文件格式规范视为安全不变量，并将加密检测为对这些不变量的违反，而非统计异常的勒索软件检测系统。具体而言，本文作出以下贡献：

(1) 我们指出了现有勒索软件防御措施在 PEER 威胁模型下的根本局限，尤其是在面对细粒度部分加密时。(2) 我们提出了一种将勒索软件防御任务卸载到云端的架构，能够进行丰富的内容分析，同时不会增加设备端开销。(3) 我们设计并实现了一种新颖的基于 FAV 的检测流水线，能够抵御细粒度部分加密。(4) 我们通过模拟勒索软件和真实世界勒索软件对 *Rhea* 进行了实证评估，证明其检测准确且实用。

我们在第 2 节讨论相关工作，在第 3 节介绍威胁模型，在第 4 节阐述系统设计，在第 5 节总结实现细节，并在第 6 节评估原型。第 7 节概述局限性和未来工作，第 8 节作出总结。

2 相关工作

表 1 总结并比较了先前提出的各种勒索软件攻击对策。请注意，现有解决方案均无法抵御细粒度部分加密，而 *Rhea* 能够通过 FAV 检测到这种加密。本节其余部分将更详细地介绍相关工作。

操作系统级解决方案。 用于防御勒索软件的操作系统级解决方案包括用户级和内核级解决方案。现有检测方法侧重于通过 I/O 模式分析或基于内容的检测来识别勒索软件活动。例如，UNVEIL [22],RWGuard [28] 和 Peeler [5] 通过分析 I/O模式来检测勒索软件。另一方面，CryptoDrop [46],采用基于内容的检测方法，保护用户数据免受加密型勒索软件侵害。与此同时，Redemption [20], ShieldFS [12], 和 Pay-Break [23] 提供了集检测、恢复和预防于一体的解决方案。

近期，已有若干以检测为中心的操作系统级解决方案被提出。ERW-Radar [62] 指出，利用少量预定义模板的模仿攻击会导致重复的 I/O模式。随后，该研究提出了一种模仿攻击检测方法，通过在内核级监控并分析 I/O 事件来实现检测。CANCAL [57] 和 MS-IDS [55] 探讨了如何优化主机系统所承受的检测开销。CANCAL 为工业环境引入了一种精简的基于欺骗的检测方法，在这些环境中，实时检测至关重要。为吸引攻击者，该方法有策略地将诱饵文件部署在包含丰富用户数据的路径中。只有识别出诱饵文件被修改后，才会启动更繁重、深入的勒索软件筛查流程。与此同时，MS-IDS 根据可疑程度对用户级进程进行优先级排序，并为每个进程采用最具成本效益的监控策略。因此，良性进程只受到轻度监控，性能开销降低约十倍。

与操作系统级解决方案的区别： 所有操作系统级解决方案的一个主要局限是，它们容易受到权限提升攻击，因为拥有提升的权限的攻击者可以轻易绕过或禁用用户级或内核级的检测与防御机制 [40–43] 相比之下，*Rhea* 对权限提升攻击具有更强的抵御能力，因为其防御机制会将拥有特权的攻击者限制在虚拟机中。

硬件级解决方案。 以SSD为中心的解决方案将勒索软件检测机制直接集成到固态硬盘的固件级。这些解决方案比操作系统级解决方案更能抵御权限提升攻击。AMOEBA [32] 通过基于内容的分析检测勒索软件攻击。此外，SSD-Insider [8] 不仅提供检测机制，还提供固态硬盘内的恢复机制。RansomBlocker [37] 提出了一种低开销的勒索软件防护方案，而 LAST [64] 提出启用感知保留策略的版本控制，以防止过早删除。同时，RSSD [44] 专注于用于攻击后分析的硬件隔离网络存储。FlashGuard [18] 利用固态硬盘的固有属性来防御针对特权用户的勒索软件攻击。

还有一些并非以SSD为中心的硬件级解决方案。一个典型例子是 Inuksuk [61]，它利用可信执行环境（TEE）保护用户数据免受篡改攻击。借助硬件信任根，此类解决方案即使面对权限提升攻击，也能保证数据完整性。

与硬件级解决方案的区别：硬件级解决方案依赖专用硬件，因此难以部署到各种不同的机器配置中。以SSD为中心的解决方案仅适用于配备专用SSD和定制固件的设备，而基于TEE的解决方案则要求配备可信计算硬件。此外，诸如*trim* 命令之类的SSD接口 [44] 可能会暴露给权限提升型勒索软件，使攻击者有可能绕过内置安全措施，并在开始加密前删除数据。此外，基于TEE的解决方案只提供不可变备份，却没有明确的检测方法 [44, 61]。*Rhea* 不受此类硬件要求的限制，不会向客户虚拟机暴露此类接口，并引入了一种新的检测方法，作为不可变备份基础设施的补充。

虚拟机监控器级解决方案。RansomSpector [51]和 Ransom-Tag [26] 采用虚拟机内省（VMI）以及

表 1：勒索软件对策相关方法的比较。每列表示是否支持相应功能。√ 表示支持该功能，而 ✗ 表示不支持该功能。

级别	方法	防篡改-能力	接口缩减	纯软件解决方案	基于云的检测	格式感知验证
OS	Redemption [20],ShieldFS [12], UNVEIL [22], Peeler [5], CryptoDrop [46], RWGuard [28], PayBreak [23], CANCAL [57], MS-IDS [55], ERW-Radar [62]	✗	✗	✓	✗	✗
硬件	SSD-Insider [8],FlashGuard [18], AMOEBA [32],RansomBlocker [37], LAST [64]	✓	✗	✗	✗	✗
	Inuksuk [61],RSSD [44]	✓	✓	✗	✗	✗
虚拟机监控器	RansomSpector [51], RansomTag [26], Rocky [21],Time Machine [19]	✓	✓	✓	✗	✗
	Rhea（本研究）	✓	✓	✓	✓	✓

I/O 模式分析，以获取虚拟机的内部信息，从而实现更准确的勒索软件检测和恢复。其他虚拟机监控器级解决方案仅专注于提供备份基础设施 [19, 21]，通过拦截虚拟 I/O 并将备份复制到云来实现。

与虚拟机监控器级解决方案的区别：RansomTag [26]和 RansomSpector [51] 依赖 I/O 模式分析来检测勒索软件，因此本质上容易受到模仿攻击的影响。与此同时，Rocky [21] 和 Time Machine [19] 并未纳入可卸载至云端的有效检测机制。与以往的虚拟机监控器级解决方案不同，*Rhea* 引入了一种新颖的格式感知验证方法，无需依赖 I/O 行为。

其他。基于机器学习的异常检测也已被用于检测恶意行为，包括勒索软件。诸如针对系统日志的深度学习 [15],、基于自编码器的网络入侵检测 [33], 以及数据流异常检测 [10] 等技术，已被用于识别异常的系统行为。一些研究提出使用机器学习对加密流量进行分类，并将其与其他高熵数据格式区分开来。例如，统计学和机器学习技术已被用于识别加密数据 [11, 13]。

与其他解决方案的区别：基于机器学习的异常检测方法无法有效应对 PEER，因为 PEER 可以模仿正常的 I/O模式，或间歇性运行以躲避检测。仅通过区分密文和明文的基于机器学习的分类器，无法提供全面的勒索软件防护。

3 威胁模型与假设

Rhea 假设处于企业 VDI 环境中，虚拟机参与计划检查点。检查点通常在用户活动较少的时段执行，例如夜间维护时段或没有用户交互的空闲时间段，以最大限度减少干扰。在每个计划检查点，应用程序应完成对受保护用户数据的持续更新，包括任何特定文件格式的元数据定稿，然后进入静止状态，在此状态下不再发出写入操作。随后，虚拟机会强制执行刷新屏障（例如 sync，并在必要时进行短暂的文件系统或卷静止），之后 *Rhea* 再创建快照。此过程可确保每个检查点快照仅捕获已提交且格式一致的文件版本，而不会捕获未完成更新过程中的瞬态状态；对于 ZIP 和 OOXML 等结构化格式而言，这些瞬态状态是误报的主要来源。在连续检查点之间，*Rhea* 还可以记录块级 I/O 活动，以支持检测后分析和恢复，例如定位受影响区域并重建最后一个已知的良好状态。

我们的目标系统包含一台运行操作系统环境的用户设备，该设备位于虚拟机（VM）中。经常使用 VDI 的企业环境是主要目标环境 [7],，但目标环境并不局限于此。任何能够运行轻量级虚拟化层并通过网络连接云服务器的计算设备，都可以成为目标环境。正如前文所述，此类设备包括个人电脑 [36], 、移动设备 [6],和边缘服务器 [45]。

我们的解决方案并未与特定的云部署模型紧密耦合，但要求用户信任云服务器，并假设攻击者无法攻陷这些服务器。如果云服务器受到可信计算硬件的保护，且用户能够信任硬件提供商，那么就可以使用公有云 [24, 25]。否则，企业可以在本地私有云上运行云服务器。也可以采用混合云：将加密块快照备份保存在公有云上，同时在私有云上分析块快照。

另一方面，攻击者可以诱骗用户点击电子邮件中的可疑链接或访问钓鱼网站，从而入侵用户的操作系统环境，导致功能强大的恶意软件被安装到用户的虚拟机中。Rootkit（根套件）尤其危险，因为它们能够利用操作系统内核中的漏洞提升权限。这类特权恶意软件可以绕过基于主机的防病毒软件或云防病毒服务的代理。

不过，我们假设攻击者无法逃逸虚拟机，因为轻量级虚拟化层的可信计算基（TCB）要小得多。值得注意的是，并非所有用户设备都配备了专门设计的固态硬盘或可信计算硬件，而虚拟机监控器可以运行在种类广泛得多的用户设备上，甚至包括智能手机 [14, 59]。

我们针对利用标准加密算法加密用户数据、从而产生高熵密文的勒索软件。也就是说，我们不考虑采用自定义或非标准加密算法的勒索软件变种，因为这类算法可能会生成低熵密文 [9, 17, 27]。这类方案的密码学强度通常较弱，并且常常已通过分析工作被破解或完全解密（例如 NoMoreRansom [3]）；与之相反，现代高影响力勒索软件家族使用的是高级加密标准级别的加密。此外，我们不考虑仅限制对系统资源的访问、但不采用加密或删除手段的勒索软件，例如屏幕锁定勒索软件 [16, 60]。此外，窃取数据并威胁公开数据的数据勒索软件/泄露勒索软件 [58] 也不在本文讨论范围内。

请注意，如果磁盘使用全盘加密软件（例如 Windows BitLocker [30]）进行加密，那么在没有加密密钥的情况下，所提出的方法将无法工作。如果块由此类软件加密，*Rhea* 便无法区分这些加密块究竟是由勒索软件造成的，还是由合法的磁盘加密造成的。用户有两种选择：(1) 在来宾系统内禁用磁盘加密，同时让底层虚拟机监控器对静态数据进行加密；(2) 通过云服务（例如 Azure 磁盘加密 [31] ）或本地部署的 Configuration Manager [29] 向 *Rhea* 提供恢复密钥。

4 *Rhea*设计

4.1 检测 PEER 的挑战

在 PEER 攻击下，I/O 模式分析变得极不可靠。由于 PEER 能够绕过内核级防御，在内核中收集的任何文件 I/O 跟踪都不可信。这使得可信遥测数据仅限于在虚拟化层或硬件层捕获的块级 I/O。然而，在这一层检测 PEER 面临若干挑战。采用间歇式加密时，相关的块 I/O 范围会随时间不断扩大，因为加密写入分散在整个时间线上。分析如此长的轨迹是一项艰巨任务，因为需要存储和处理的块 I/O 事件数量可能极其庞大。

此外，PEER 还可以通过任意交错插入恶意块 I/O 与良性 I/O 来掩饰恶意块 I/O。PEER 还可以通过模仿攻击进一步提升其隐蔽性，这会使关联与攻击相关的 I/O 事件变得异常复杂。

基于内容的检测听起来可能是文件 I/O 分析的一种有前景的替代方案，但面对若干挑战时并不可靠。首先，细粒度部分加密攻击正日益普遍。例如，Black Basta、BlackCat、Play 和 LockBit 等当代规避型勒索软件变种无需加密整个文件；它们会灵活调整加密任务的偏移量和大小。例如，Black Basta 已知会：（1）对于小于 5000 B 的文件，加密整个文件；（2）对于大于 5000 B 且小于 1 GB 的文件，反复加密 64 B 并跳过 128B；以及（3）对于至少大于 1 GB 的文件，先加密前 5000 B，随后反复加密 64 B 并跳过 6336 B [34]。其次，规避型变种可能会有意在加密数据旁插入低随机性的填充

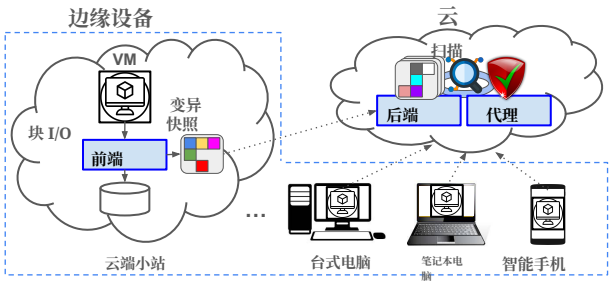


图 1：*Rhea* 架构。*Rhea* 前端 会定期生成变异快照。随后，*Rhea* 后端 将一系列变异快照存储为长期不可变备份。借助这一机制，*Rhea* 代理会进行更深入的分析，以检测勒索软件攻击。

数据。因此，许多写入缓冲区包含密文和明文的混合内容，导致其测得的熵保持在足够低的水平，从而避免被检测到。第三，从更根本的层面看，通常用于测量熵的统计方法（例如香农熵或 χ^2 tests）在较小粒度（如 64B）下，无法可靠地正确分类明文和密文数据。简而言之，当规避型攻击每次仅加密小数据块时，仅凭字节级统计信息极难区分密文与压缩明文。为克服这些挑战，Rhea 在其仅基于内容的检测流水线中引入了一种新颖技术 FAV。

4.2 架构概述

图 1 展示了*Rhea*的架构概述。*Rhea* 前端运行于边缘设备上，而*Rhea* 后端则作为后端服务器运行于云端，为所有已连接的边缘设备管理共享数据存储。边缘设备的范围很广，既包括台式电脑、笔记本电脑、平板电脑和智能手机等个人计算设备，也包括部署在云端小站的服务器；云端小站本质上是位于用户设备与云服务器之间的边缘数据中心。*Rhea* 前端包含一个虚拟块设备，可为虚拟机 (VM) 实现无缝存储集成。当虚拟机运行客户操作系统时，在客户操作系统中发起的所有文件输入/输出 (I/O) 操作都会转换为块 I/O 操作，随后由*Rhea* 前端提供的底层直通块设备进行处理。

为维护数据一致性和可靠性，*Rhea* 前端会在计划检查点将边缘设备上的数据状态转换复制到*Rhea* 后端。在块层中，块是 I/O 的基本单位，由虚拟机发出的写入请求会应用于边缘设备上的底层块设备。在每个检查点间隔（时期）内，*Rhea* 会跟踪块级修改，并保留每个已修改块的最新版本，作为该时期的块快照 。在时期结束时，系统执行刷新和静默屏障，确保所有待处理更新均已完成；随后，这些块快照会汇总为一个变异快照，用于记录该时期内发生的完整数据状态转换。接着，生成的变异快照会复制到*Rhea* 后端，并存储在 DynamoDB [50] 等分布式存储系统中。由于变异快照表示已提交的块状态

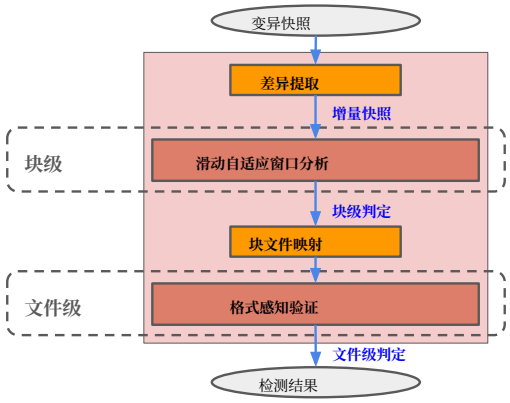


图 2：*Rhea* 的格式感知的纯内容检测流水线。*Rhea* 的双层检测流水线以变异快照为输入，并生成检测结果作为输出。

在检查点边界，按顺序重放变异快照即可使后端在云中忠实地重建块设备快照。

云中的*Rhea* 代理使用变异快照维护块设备快照，并检测勒索软件留下的加密痕迹。图 2 展示了其双层检测流水线，该流水线由四个阶段组成。前两个阶段采用启发式方法来高效筛选候选项，而格式感知验证（*FAV*）则作为权威阶段，作出最终判定。

(1) **差异提取：**代理从变异快照和定期块设备快照中提取增量快照，这些快照表示每个时期写入块设备的确切前向差异。这些增量定义了后续阶段的分析范围。(2) **滑动自适应窗口分析（SAWA）：**SAWA 作为早期预过滤器，扫描增量快照，以识别统计检验（例如 χ^2）无法可靠区分明文与密文的区段。当加密粒度足够粗时，SAWA 可能已经能够将某些块标记为明显可疑。不过，SAWA 不会直接做出决定，而是将含糊或处于临界状态的块保守地转交给下一阶段，从而确保较高的召回率。(3) **块—文件映射：**SAWA 标记的块会根据块设备快照中的元数据映射回相应文件，并转换为文件偏移范围。这些范围可定位可能受影响的文件区域，并传递给最终阶段。(4) **格式感知验证（FAV）：**FAV 是整个流程中的决定性阶段。它在文件级检查可疑文件，重点关注前一阶段识别出的文件偏移范围，并确定观察到的数据状态是否无效。FAV 不依赖统计异常，而是检查数据是否违反文件格式规范规定的句法或语义不变量。因此，FAV 能够准确检测出可逃避基于熵或 χ^2 的方法的细粒度加密和部分加密。

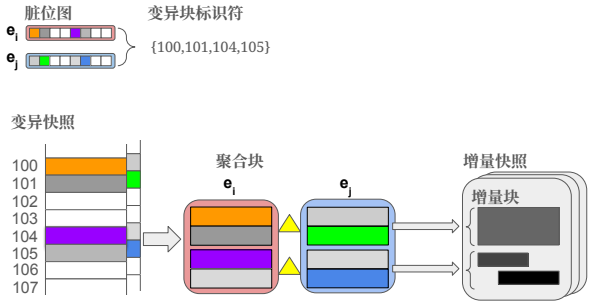


图 3：差异提取。*Rhea* 计算变异块的前向差分，以构建增量快照。

4.3 差异提取

差异提取阶段将原始变异快照转换为紧凑的增量块，这些增量块共同构成单个增量快照，如图 3 所示。该阶段接收两个时期：e_i 和e_j，其中 $e_i \le e_j$，并输出按区段分组的增量块序列；区段是文件系统中连续块分配的近似单位。每个增量块记录一段连续的已修改字节，即表示对于相应区段，在 e_i 之前获取的块设备快照与在 e_j 之后获取的块设备快照之间的前向差异。这种表示可直接供下一阶段使用。

该阶段首先识别在 e_i 至 e_j（含）之间的纪元中发生变异的脏块所对应的物理块标识符集合。每个按纪元划分的变异快照都附有一个脏位图，用于记录给定纪元内更新了哪些块。为确定纪元窗口 $[e_i, e_j]$ 内所有被修改块的完整集合，*Rhea* 会对该范围内的所有脏位图执行逻辑或归约，从而计算出所有变异块的并集。

接下来，收集到的块标识符会按其所属区段分组，其中区段大小固定为 4,096 B（即对文件级区段的近似），块大小为 512 B（即对磁盘级扇区的近似）。由于收集到的块标识符仅对应脏块，因此每个区段只记录该区段内实际更新的块。*Rhea* 会进一步将每个区段内连续运行的已修改块合并为聚合块。这种合并方式能够最大限度地减少块边界处的短残余跨度，并使后续的按字节差分阶段能够更高效地处理较长的连续跨度。

随后，针对每个聚合块计算前向差分。对于每个聚合块，*Rhea* 会获取两个字节数组：一个是时期 e_i 之前的最新内容，另一个是时期 e_j之后块快照中的最新内容。为计算前向差分，我们线性扫描配对的字节数组，以定位发生变异的区域，并生成与这些区域相对应的增量块 。为减少较大写入中夹杂微小未变岛状区域所导致的碎片化，我们将长度短于 min_gap_length（默认值为 16B）的匹配序列视为周围差异的一部分。这种“间隙抑制”可避免因新旧内容中偶然相同的短序列而意外拆分单个高熵区域。最后，将所有增量块汇编成一个增量快照。

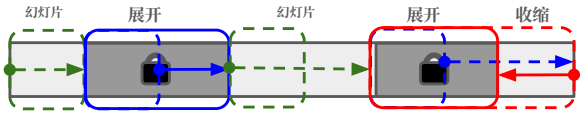


图 4：SAWA 流程示意图。

该设计具有确定性，且易于并行处理：区段彼此独立；区段内的聚合块序列可以按顺序处理，内存开销较低（每次流式处理一个序列），也可以在资源允许时并行处理。最后，生成的增量块为后续阶段提供了规范的衔接。由于每个增量块都对应给定时间窗口内的一次精确变异，后续阶段可以将分析范围限定在已修改的区域。

4.4 滑动自适应窗口分析

图 4 展示了 SAWA 机制在我们的检测流水线中的运行方式。SAWA 从固定的初始窗口宽度开始，测量数据区域的均匀性。为了量化均匀性，SAWA 使用卡方均匀性检验，以衡量观测到的字节频率与完全均匀分布之间的偏差：

$$\chi^2 = \sum_{b=0}^{255} \frac{(c_b - m/256)^2}{m/256},$$

其中，c_b是长度为 m 的窗口中字节值 b 的观测计数。较低的 χ^2 值对应更均匀、熵更高的数据。当 $\chi^2 \le \tau$ 时，窗口会被判定为可疑，其中 τ 是可配置的阈值。

SAWA 通过重复执行滑动、测量、扩展和收缩的自适应循环，使检测窗口与任意大小的加密区域对齐。它首先在每个增量块上执行滑动搜索，以定位观测到的 χ^2 统计量首次低于预定义均匀性阈值的位置，这表明该区域可能已加密（接近均匀分布）。检测到这样的区域后，滑动阶段暂停，SAWA 转入扩展阶段。在此阶段，窗口大小逐次翻倍，以覆盖整个可疑区域。如果 χ^2 统计量升至阈值以上（即不再足够均匀），SAWA 就进入收缩阶段。在收缩阶段，SAWA 使用二分搜索来寻找最佳边界，在保持 χ^2 统计量低于阈值的同时，尽可能增大窗口长度。此过程结束时，扩展后的窗口即告确定。随后，滑动阶段从该最终窗口的末尾继续；遇到另一个可疑区域时，则重复扩展阶段。

算法 1 展示了整体工作流程。该过程从使用宽度为 w（例如 16B ）、步长为 s （例如 4B ）的滑动基准窗口进行正向扫描开始。当窗口通过均匀性测试时，SAWA 会锚定该位置并进入自适应扩展阶段，只要均匀性仍然成立，就反复将窗口大小加倍。这种几何增长可避免将单个较长的高熵区域分割成许多小片段，从而提高均匀性测量的可靠性。随后可进入收缩阶段

算法 1滑动自适应窗口分析

要求： DeltaBlocks 列表 D、窗口宽度 w、步长 s、均匀性阈值 τ、设备块大小 B
确保： 可疑块范围集合 R

R $\leftarrow \emptyset$
对于所有 DeltaBlock $d \in \mathcal{D}$ **执行**
  $x \leftarrow$ 字节载荷 of d
  $p \leftarrow 0$
  当 $p + w \le |x|$ **时执行**
    $L \leftarrow w$
    $chi \leftarrow \chi^2(x[p{:}p{+}L])$
    如果 $chi \le \tau$ **则**
      当 $chi \le \tau$ **且** $p + 2L \le |x|$ **时执行**
        $L \leftarrow 2 \times L$ ▷ 几何扩展
        $chi \leftarrow \chi^2(x[p{:}p{+}L])$
      结束 while
      最大化窗口长度 L **满足以下条件** $chi \le \tau$
      $range \leftarrow$ 获取 $[p{:}p{+}L$ 的块范围)
      R $\leftarrow$ R $\cup\{range\}$
      $p \leftarrow p + L$
    else
      $p \leftarrow p + s$
    结束 if
  结束 while 循环
结束 for 循环
返回$\mathcal{R}$

以微调窗口边界的末端，旨在找到仍低于阈值的近乎最大跨度。

SAWA 阶段旨在从每个增量块中隔离出字节频率近乎均匀的局部字节范围。由于诸如 χ^2 测试之类的统计方法在窗口宽度较小时并不可靠，因此该阶段作为一种启发式方法，用于识别压缩或加密通常产生的高熵潜在迹象。SAWA 会为每个增量块输出可疑区域记录，并精确保留每个区域的物理溯源信息，包括所覆盖的设备块标识符及其在相应块内的偏移量。借助这些信息，后续的格式感知分析器只需检查相关字节跨度，即可最终判定是否发生了 PEER 攻击。SAWA 的输出有意保持保守——SAWA 充当后续阶段的高召回率、本地性保留型预过滤器。

4.5 块到文件映射

块到文件映射阶段会将 SAWA 在物理块级别识别出的可疑区域转换为文件路径和字节偏移量。*Rhea* 会扫描文件系统元数据，重建每个文件数据区域的布局，并将其表示为物理块地址空间中一组互不重叠的区间。块标识符到文件路径及偏移量的映射保存在内存中的基于区间树的索引中，以支持高效查询。

给定一组可疑块标识符，映射器查询区间索引，以获取相应的文件标识符和规范路径。由于文件碎片化很常见，多个不相邻的

区域可能属于同一个文件；映射器会对这些区域去重，生成一个最小的可疑文件列表，从而覆盖所有标记块。这样，系统便能以最小开销，将分析定位到与 SAWA 报告的可疑区域相关的精确文件路径–偏移量对。

可信清单与白名单机制。为减少误报，检测流水线会应用白名单过滤器，根据精选白名单验证媒体及其他静态文件的完整性。其核心假设是，这类文件（例如图像、视频及其他已发布媒体资源）在分发后很少被修改，因此只要其完整性保持不变，就可以安全地信任它们。因此，检测流水线会将 .jpg、.png、.mp3 和 .mp4 等文件格式的验证工作交由白名单过滤器处理。

块到文件映射将可疑块解析到特定文件后，流水线会将每个候选文件与可信内容清单进行交叉核验。该清单存储了可信来源此前发布的文件的密码学哈希值（SHA-256）。对于每个可疑文件，流水线都会计算其当前哈希值，并与清单中的条目进行比较。哈希值与可信记录匹配的文件会被标记为已验证，随后不再纳入进一步分析。

该机制确保不会反复将良性但经常访问的内容——例如缓存媒体或捆绑应用资源——标记为可疑。过滤器通过验证完整性，而非仅依赖文件名或路径，能够有力防范意外和恶意造成的误报。最终得到的未列入白名单的可疑文件集合会被传递给下游的格式感知过滤器，以进行深入检查。

4.6 格式感知验证

识别出候选可疑字节范围后——这些范围可能由 SAWA 识别（适用于大小达到或超过最小 X^2 窗口大小的块），也可能是直接转发无法可靠评分的较小块——处理流水线会执行格式感知验证，利用特定格式语法和语义来完善这些初步结果。验证器不会将文件视为不透明字节序列，而是重建其逻辑内容——包括解压缩或其他解码步骤——并评估解码结构是否符合所声明文件类型的句法和语义不变量。此阶段大幅减少了误报，同时保留违反格式预期的加密或损坏区域。每个包含候选区域的文件都会根据其魔数和扩展名（例如 .txt、.zip、.xlsx、.pptx、.docx、.pdf）分派给相应的专用验证器。验证器会运用深入的格式感知推理——解析容器元数据、解压缩压缩流、检测加密成员，并解读特定内容结构——以确定被标记的区域究竟代表密文、良性的高熵数据（例如图像、字体、媒体），还是较小的明文片段。格式感知验证器通过分析解码并解读后的内容，而非原始字节，确保最终仅有确实异常、在语义上不合常理或无法验证的区域会被升级处理。

文本验证器。文本验证器针对预期包含人类可读内容的格式，例如日志、配置文件和源代码。其核心理念是，文本数据采用高度结构化的编码方案进行编码，例如 UTF-8，而密文表现得如同随机字节，几乎从不符合此类语法约束。因此，文本验证器会对每个可疑片段执行严格的 UTF-8 解码。真正的明文必须能够被正常解码，而密文或损坏的字节极有可能导致解码失败。成功解码的片段会被判定为良性，但解码失败会使该区域被升级标记为可疑密文。

原则上，攻击者可以生成满足 UTF-8 语法约束的密文，但根据标准密码学假设，生成既是密文又能在自然语言中呈现语义连贯性的内容，在计算上不可行，因为密文的设计目标是与随机数据无法区分。也就是说，即使随机密文偶然被解码为 UTF-8，它几乎也不会在自然语言中呈现语义连贯性，而通过结合自然语言处理（NLP）的分析即可轻易检测出这一点。

ZIP 验证器。ZIP 验证器通过解析归档文件的容器结构，而非将其视为扁平的字节序列，来区分合法的压缩内容与真正的密文。它首先确定 中央目录结束记录（*EOCD*）、中央目录（*CD*）和本地文件头的边界，然后将文件划分为三个逻辑区域：(i) 元数据（EOCD/CD），(ii) 成员载荷，以及 (iii) 未占用间隙。

对元数据区域进行结构验证，以确保文件头和记录正确无误。验证器会检查 EOCD 记录、CD 条目和本地文件头之间的字段一致性，核验偏移量、标志、长度和签名模式。这些检查可以区分合法的结构字节与异常的、呈均匀分布或格式错误的片段，并将后者升级处理。

成员载荷根据其压缩方法进行解析。DEFLATE 成员会被解压，而 STORE 成员则按原样读取。ZIP 验证器会递归遍历嵌套容器（例如，ZIP 中嵌套 ZIP），直至到达解码后的叶节点，或深度与字节预算耗尽。所有分类均基于解码字节进行，以防将压缩的高熵数据误认为密文。加密成员和媒体叶节点（例如 JPEG、PNG、MP3、MP4）会跳过基于解码的分析，转而与可信清单进行核验。加密成员使用其加密载荷哈希进行验证，而媒体叶节点则使用基于解码字节或原始字节计算的内容哈希进行验证。哈希匹配的项目会列入白名单。类文本叶节点则使用前文所述的文本验证器功能进行明文检查。

未占用间隙可能由填充、对齐、未完整写入的归档文件或格式错误的 ZIP 文件造成。由于攻击者可能会在这类空闲空间中隐藏密文，验证器会采用结合大小与熵的防护措施：容许较小且低熵的填充；对于异常庞大、高熵或结构上不合理的间隙，则采取保守策略，将其升级为潜在的注入密文或隐藏恶意内容。

OOXML 验证器。OOXML 格式（.docx、.xlsx、.pptx）是基于 ZIP 的容器，会将内容组织成结构化部件，例如用于存储文档文本、幻灯片或工作表的 XML 文件、嵌入的媒体文件，以及用于描述各部件之间连接方式的元数据文件（例如样式、关系、主题）。OOXML 验证器

采用与 ZIP 验证器相同的先成员、后解码再判定方法：它通过本地文件头枚举所有成员，并针对每个可疑的 SAWA 字节区段，依据与其重叠的成员得出分类判定，而非依据数据流中途的字节。

成员载荷通过解码底层容器（DEFLATE 解压或 STORE 直通）获得，确保所有分类都针对解码后的内容，而非原始压缩流。媒体部件（例如 word/media/、ppt/media/、xl/media/ 和缩略图）通过可信清单进行验证：匹配的 SHA-256 摘要会列入白名单，而哈希不匹配或无法读取的媒体则会升级为可疑项。

文本类部件，包括 word/document.xml、幻灯片 XML、工作表 XML 和共享字符串表，会通过严格的 UTF-8 解码和轻量级 XML 合理性检查进行明文验证。格式错误的 XML、混合二进制/XML 区域或不合理的文本结构会被升级处理，而语义一致的类文本内容则会判定为良性。

与任何成员均不重叠的可疑片段会被解释为元数据区域（本地文件头（LFH）范围、数据描述符尾部或较小的 LFH 前导区）。这些区域会接受保守的结构验证；只有与 OOXML 容器布局不符、呈现均匀特征或异常大的间隙才会被升级处理，从而避免针对预期结构字节的误报。

PDF 验证器。 PDF 验证器具备流感知能力：它会将字节范围映射到具体的 PDF 流及其相邻的结构元数据（例如文件头、xref、尾部和交叉引用起始位置），从而能够针对每个流作出决策，而不是采用粗粒度的全文件启发式方法。轻量级字典解析会提取 /Filter 和 /DecodeParms 条目，使验证器能够对受支持的滤波器管线（Flate、ASCII85、ASCIIHex、RunLength，以及用于 Flate 的 PNG 预测器）进行解码。只要能够进行解码，验证就会基于流的解码后的表示执行，确保决策反映其逻辑结构，而非其压缩或编码形式。

某些流类型，尤其是图像流和字体流，包含不透明二进制载荷，其内部结构仅受到 PDF 规范的宽松约束。对于此类流，验证器会应用基于白名单的完整性检查：对流载荷的规范化表示计算密码学哈希（SHA-256），并将其与 PDF 发布者提供的可信清单进行比对。其他组件类型，例如 ICC 配置文件、内容流、对象流、嵌入文件和 XMP 元数据，则通过检查其解码字节是否呈现该对象类别所预期的最低结构特征来进行验证（例如，可识别的字典前缀、有效的流头或格式正确的元数据标记）。对于解码后的表示存在格式错误、与其声明类型不一致或呈现不自然均一性的组件，会将其升级标记为潜在密文。任何流之外的区域，包括 PDF 文件头、对象字典、交叉引用表、尾部，以及由对齐或增量更新引入的小间隙，则使用简单的结构检查而非内容分析进行验证。这些区域通常由带有可预测标记的短元数据片段组成（例如“%PDF-1.x”、“xref”、“trailer”、对象编号和字典分隔符）。

5 实现细节

Rhea* 前端和后端**。 *Rhea* 构建于*Rocky* [21], 之上；Rocky 是一种开源分布式复制块设备，旨在为边缘云环境提供具备故障和篡改抵抗能力的存储。*Rocky*实现了第 4.2 节前文所述的 ***Rhea 前端 和***Rhea*** 后端 设计。为方便起见，***Rocky*** 端点 和连接器云端 分别重命名为***Rhea*** 前端 和***Rhea*** 后端。相关实现细节将在原论文 [21] 中讨论。

Rhea* 代理**。 我们使用 Python 实现了*Rhea* 代理的原型，并基于 *Apache Spark* 构建了主要检测流水线，以支持对大型快照数据集进行可扩展的容错处理。整个实现包含约 1 万行 Python 代码，组织为三个组件 [1]：预处理器（292 行代码）、检测器（7,341 行）和后处理器（655 行代码）。预处理器从 ***Rhea 后端获取变异快照和位图，将其持久化到本地，并为每个时期构建块设备快照。检测器运行前文所述的检测流水线，其中包含与检测逻辑阶段相对应的独立子系统：差异提取、SAWA、块到文件映射和 FAV。在检测器内部，FAV（detector/fileaware/）包含 4,527 行代码，其余检测器逻辑（差异提取、SAWA、块到文件映射和编排）包含 2,814 行代码。总的来说，FAV 占检测器的大部分代码（占检测器代码的 ≈62%）。检测器采用模块化设计，具有很强的可扩展性，因此可以针对不受支持的文件格式添加新的格式感知验证器，也可以添加功能更强大的格式感知过滤器。

6 评估

我们的评估旨在回答以下研究问题：

(1) *Rhea* 在各种部分加密策略下检测 PEER攻击的效果如何，整个检测流水线的性能概况如何？(2) 与此前基于统计分析提出的基于内容的检测方法相比，*Rhea* 能否利用细粒度的部分加密策略更有效地检测 PEER攻击？(3) *Rhea* 能否检测现实世界中的规避型勒索软件变种发起的已知攻击？

6.1 各种规避型攻击的检测

我们通过实施模拟攻击，并使用前文所述的完整检测流水线对这些攻击进行检测，评估*Rhea* 检测各种规避型攻击的有效性和效率。

环境设置。 我们的实验在实验室的一组由五台台式电脑组成的集群上进行。每台机器均配备 Intel i5 处理器、16 GB 内存和 512 GB NVMe 固态硬盘，并运行 Ubuntu 22.04 作为主机操作系统。这五台机器被配置为一个 Spark 集群。*Rhea* 的前端、后端和代理共同部署在集群中的一台机器上。工作负载生成、预处理和检测作为独立阶段执行。由于这些阶段独立运行，

[1]我们使用 cloc 报告三个核心流水线组件的代码行数（LoC）。

表 2：用于评估的模拟 PEER 攻击模式

模式（参数）	定义
Fast（N）	仅加密前 N 字节（前缀）文件。
SkipStep (N，S)	重复条带模式：加密 N 字节，然后跳过 S 字节，持续至文件末尾。
Animagus (F)	随机加密文件的大约 F% 块。

将各组件部署在同一台机器上不会影响检测效果或性能测量。

为评估*Rhea*的检测能力，我们为十种文件格式分别收集了一个具有代表性的文件，总大小约为 16.4 MB。该文件集涵盖常见的台式电脑文件格式和容器化文档：一个 PDF（105,116 B）；OOXML 文档，包括 PowerPoint（59,960 B）、Excel（12,817 B）和 Word（75,973 B）；图像文件，包括 JPEG（142,294 B）和 PNG（71,291 B）；一个 ZIP 压缩包（64,829 B）；音频和视频媒体，分别为 MP3（480,967 B）和 MP4（16,172,912 B）；以及一个纯文本文件（28,324B）。

在检测方面，我们将设备块大小固定为 512 B，将区段大小设为 4 KB，并在所有运行中使用同一组 SAWA 参数。具体而言，我们将窗口宽度设为 w=16B，步长设为 s=8 B，从而使相邻窗口测量结果之间产生 50% 的重叠。对于统计检验，根据第 4.4 节的描述，我们使用 χ^2 检验和 256 箱字节均匀模型（df $= 255$）对每个窗口进行评分。我们采用 χ^2 阈值 τ=350，以区分高熵片段与良性内容。所选参数组合（w=16B、s=8B、τ=350）来自初步扫描，旨在平衡误报控制与 PEER 攻击模式下的较高召回率，后文将对此进行介绍。

攻击场景。我们的 PEER 模拟器会克隆原始文件，并按照表 2 所述的三种攻击模式，使用 256 位 AES 分组密码的 CTR 模式对其进行加密：*Fast*、*Skip-Step* 和 *Animagus*。这些攻击模式取材于现有规避型勒索软件的行为，例如采用部分加密的 Black Basta [34],BlackCat [35],和 Play [48] [2]，以及实施模仿攻击的研究原型勒索软件 [63]。*Fast* 加密针对文件头和索引，以最短运行时间破坏文件的可用性。*Skip-Step* 交替生成加密区域和明文区域，造成部分损坏，从而绕过采用统计分析的朴素检测器。*Animagus* 随机选择一定比例的文件块进行加密，并在每个写入缓冲区中混合加密块与明文块，降低每次写入的熵，同时仍使文件无法恢复。

对于每个模拟的 PEER 攻击活动，我们执行相应的攻击模式，并运行*Rhea* 代理来测量检测准确率。首先，我们使用 mkfs.ntfs 命令格式化*Rhea* 前端公开的块设备，以安装 NTFS 文件系统。然后，我们挂载该块设备，并将所有原始文件复制到挂载点。随后，PEER 模拟器会根据本次运行配置的特定攻击模式克隆并加密文件。对于*Fast* 和*Skip-Step* 模式，我们将密文块大小设为 4 KB。对于*Skip-Step*，模拟器会跳过 4 KB，然后再加密下一个块。对于*Animagus*，我们将模拟器配置为加密 25% 的文件块。

检测结果。 表3展示了所述实验的结果。我们在块级别计算混淆矩阵，以评估基于 SAWA 的分析性能。随后，我们在文件级计算混淆矩阵，以评估基于文件级 FAV 的分析性能。在块级别，对于大多数攻击模式，召回率几乎达到完美（FN $= 0$），而精确率则因变异策略而有很大差异。*Fast*、*Skip-Step* 和 *Animagus* 等规避型攻击模式会生成大量类密文区域，尽管召回率很高，却拉低了块级精确率。尽管如此，在文件级，*Rhea* 会按文件汇总可疑块，并应用文件感知验证器，从而在所有设置下消除误报。因此，文件级分析的准确率、精确率、召回率和 F1分数均达到 100%。

性能概况。表 4 汇总了三种攻击模式下的端到端运行时间。对于每种模式，我们分别运行攻击和检测流水线三次，并报告平均运行时间及标准差（std.）。总体平均运行时间为 29.45 s，标准差为 3.01 s。运行时间与加密的块数成正比。最快的配置是 *Fast*（25.26 s），因为它加密的块数最少——约 80 个块，总计约 40 KB。*Animagus* 模式耗时 27.43 s，加密约 8 K 个块（约 4.1 MB）。最慢的配置是 *Skip-Step*，耗时 28.46 s，加密约 17 K 个块（约 8.2 MB）。

按逻辑阶段分解总运行时间后，平均而言，主要耗时来自 SAWA （ 12.28s ）和 FAV （ 7.03 s ）。由于数据集中包含许多良性的高熵文件，无论攻击模式如何，对这些文件进行块级处理所需的时间都相近，因此 SAWA 的开销在所有模式下都相对稳定。*Fast*和*Animagus* 模式的 FAV 开销相近，因为大多数加密区域与压缩区域重叠。因此，无论是否发生加密，都必须对这些区域进行解压缩，导致处理开销相近。然而，当加密越来越多地影响非压缩区域时（如*Skip-Step* 模式），FAV 开销便会上升。具体而言，FAV 运行时间增加了约 47%，因为 *Skip-Step* 加密的区域数量约为*Animagus*的两倍。

6.2 细粒度攻击的检测

为证明 *Rhea* 比以往依赖统计方法的基于内容的检测器更有效地检测细粒度部分加密攻击，我们实现了基于香农熵和 χ^2 检验的纯文件级检测器，以及不含 SAWA 的 FAV，并比较它们的检测率。

环境设置。 我们在实验室的一台个人电脑上进行了细粒度攻击检测实验，该电脑的配置已在第 6.1 节中介绍。为了评估*Rhea* 对细粒度文件加密的检测能力，我们收集了 140 个良性文本文件（321 MB）、45 个良性 ZIP 文件（442 MB）、353 个 DOCX 文件（325 MB）和 765 个 PDF 文件（651 MB）。对于基于熵的

[2]我们并不声称这些模式涵盖了所有情况。

表3：样本数据中每种模拟的 PEER 攻击模式的块级和文件级结果。

攻击模式	块级				文件级				性能指标			
	TP	TN	FP	FN	TP	TN	FP	FN	准确率	精确率	召回率	F1 分数
Fast	80	34379	32957	0	10	12	0	0				
Skip	16837	34027	16552	0	10	12	0	0	100%	100%	100%	100%
Animagus	8394	34328	24694	0	10	12	0	0				

表 4：每种攻击模式的性能开销（时间单位为秒），分别基于明文/加密数据。

攻击模式	时间（秒）					总计（均值 ± 标准差）	写入块	加密	
	增量	SAWA	映射	FAV	输出			块	字节
Fast	5.486	10.082	1.193	6.101	2.395	25.257 ± 0.430	67416	80	40960
Skip	5.727	10.318	1.203	8.840	2.372	28.460 ± 0.657	67416	16837	8619239
Animagus	6.077	11.643	1.155	6.020	2.532	27.427 ± 0.570	67416	8394	4297728

和基于 χ^2 的检测器，调整阈值，使其在未应用加密时达到不会产生误报的最高灵敏度。

攻击场景。 我们模拟细粒度部分加密攻击，以压力测试我们的基于 FAV 的检测能力，超越现有真实世界勒索软件攻击的范围。我们使用 PEER 模拟器克隆文件，并应用 *Skip-Step* 加密模式。每次写入的加密字节数从 64 B 到 512 B 不等，而跳过距离固定为 128B，这遵循了在 Black Basta 勒索软件变种中观察到的攻击模式 [34]。

检测结果。 图 5 展示了检测结果。在所有文件类型中，即使加密稀疏且分散，基于 FAV 的检测仍保持近乎完美的准确率。相比之下，基于熵和 χ^2 的检测器依赖整体统计偏差，因此在加密应用于小块、非连续区域时，性能会显著下降。这些结果表明，在文件格式规范方面的正确性，比字节级统计异常更能提供强大且可靠的信号，用于检测细粒度部分加密攻击。

我们进一步通过在相同的 *Skip-Step* 攻击模式下，将加密粒度降低至 64 B 以下，证明基于 FAV 的检测能够抵御极细粒度的部分加密。图 6 显示，即使在这种极端情况下——加密粒度低至单字节——FAV 对 PDF、DOCX 和 ZIP 文件的检测率仍达到 100%。纯文本（TXT）是唯一的例外，检测率略有下降，在 1 B 时为 95.0%，在 2 B 时为 97.86%。这一现象表明，极小的扰动有时会因统计上的巧合而绕过 UTF-8 有效性检查，即加密字节恰好落入有效的 UTF-8 编码范围。

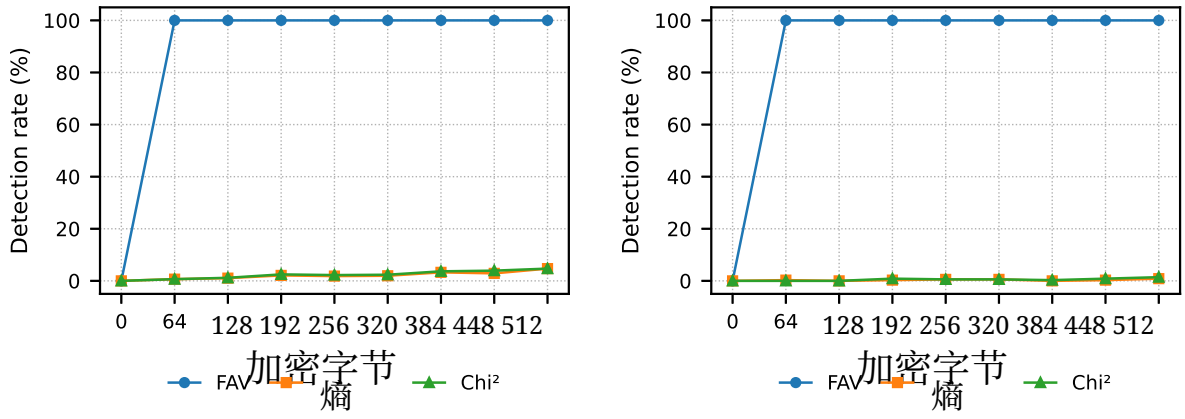
6.3 真实世界攻击的检测

环境设置。用于检测真实世界攻击的实验在前文所述的同一硬件平台上进行。我们使用了一个包含 140 个文本文件的数据集，每个勒索软件样本都会对这些文件进行加密。真实世界勒索软件样本在 VirtualBox 7.2.4 中运行，每个勒索软件完成加密过程后，系统状态都会回滚到干净快照。

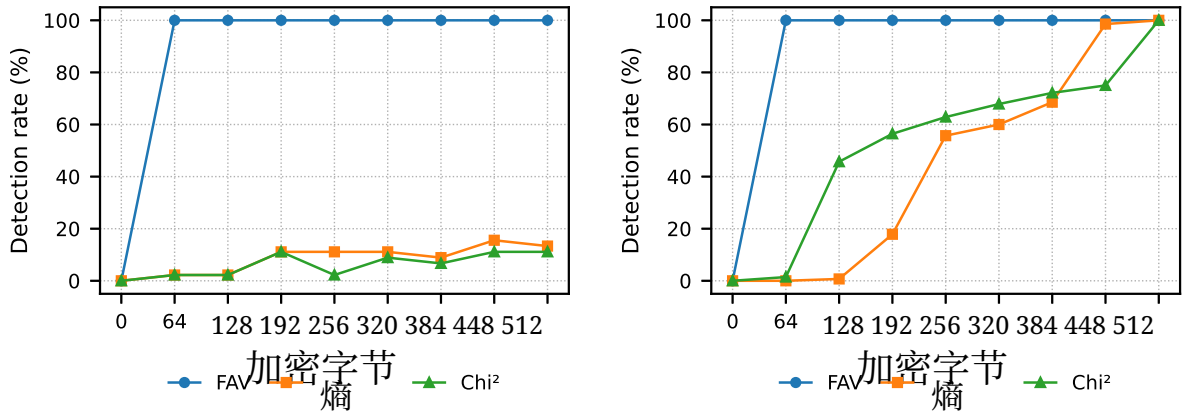
表5：对 78 个真实世界勒索软件样本进行评估。

家族	样本（%）	家族	样本（%）
Play	1（1.28）	LockFile	2（2.56）
BlackBasta	1（1.28）	Cryptor	1 (1.28)
BlackMatter	2 (2.56)	Chaos	1 (1.28)
TeslaCrypt	1 (1.28)	GarrantDecrypt	1 (1.28)
Hive	1 (1.28)	Conti	1 (1.28)
Maze	1 (1.28)	Mole	1 (1.28)
Akira	5 (6.41)	潘多拉	1 (1.28)
Medusa	3 (3.85)	Vigorf	1 (1.28)
Rhysida	1（1.28）	Darkbit	1（1.28）
AvosLocker	2（2.56）	Saturn	1 (1.28)
Virlock	1 (1.28)	RagnarLocker	1 (1.28)
NightSky	1 (1.28)	Nemty	1 (1.28)
Sodinokibi	2 (2.56)	诸神黄昏	1 (1.28)
Babuk	25 (32.05)	WastedLocker	1 (1.28)
福波斯	1 (1.28)	孤岛危机	1 (1.28)
HelloKitty	1 (1.28)	Yurei	1（1.28）
VoidCrypt	1（1.28）	Lynx	1（1.28）
Nitrogen	2（2.56）	其他	10（12.82）

攻击场景。 为获取真实世界勒索软件样本，我们从 MarauderMap 数据集 [53] 和 MalwareBazaar [4] 中收集了 78 个活跃的规避型勒索软件二进制文件。对于每个样本，我们根据 VirusTotal [56] 上多个检测器最常推断出的标签，为其分配了一个勒索软件家族。每个样本在被执行于虚拟机内部之前，都会先单独进行验证。确认无误后，允许勒索软件运行，直到其加密阶段完成。执行结束后，我们使用 scp 命令通过网络将加密文件复制到主机，并且



(a) PDF 文件。 **(b) DOCX 文件（OOXML 文件）。**



(c) ZIP 文件。 **(d) TXT 文件。**

图 5：细粒度部分加密情况下与基于内容的检测的比较。在四种文件类型中，固定跳过距离（skip=128）时，检测率随每次写入的加密字节数变化。FAV（*Rhea*）在稀疏加密情况下仍保持稳健，而随着加密变得稀少且分散，统计检测器的性能会下降。

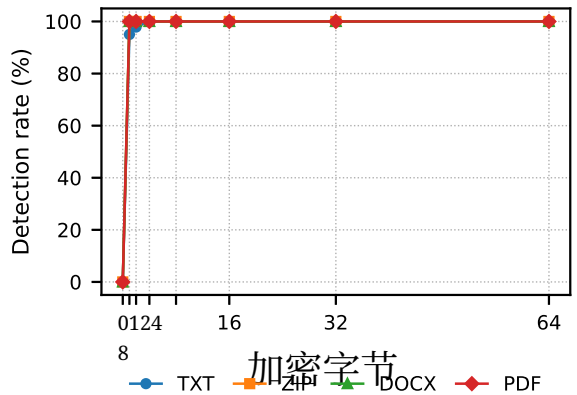


图 6：在细粒度部分加密（字节 < 64）情况下测试 FAV 的极限。固定跳过距离（skip=128）时，每次写入的加密字节数变化对应的检测率。加密字节数大于 4 B 时，所有格式均可被完美检测；TXT 是唯一在 1–2B 时检测性能略有下降的情况。

将虚拟机回滚到初始的干净状态，然后运行下一个样本。随后，我们使用复制的加密文件在主机上执行基于 FAV 的检测，并使用检测器输出对结果进行分析。

检测结果。 我们评估*Rhea* 检测真实世界的勒索软件攻击的效果；结果汇总于表 5。我们将样本映射到 35 个真实世界的勒索软件家族，而无法明确映射到已确立勒索软件家族的样本则标记为其他。在所有评估的勒索软件样本中，*Rhea* 正确识别了所有加密文件，且假阳性和假阴性均为零，准确率、精确率、召回率和 F1 分数均达到 100%。这些结果表明，在所评估的条件下，*Rhea* 能够以完美的灵敏度和精确率可靠地检测真实世界的勒索软件活动。

7 讨论

一些广泛使用的文件格式会有意允许存在一些区域，其语法或语义仅受到较弱约束，并且依法可以包含任意字节序列。例如，PDF 文件中的对象之间的非流式

间隙、对齐填充、增量更新部分和辅助元数据，以及 ZIP 压缩档案中活动条目之外的前缀数据、填充或未占用间隙。尽管这种宽松性支持灵活的创建和编辑工作流，但也为隐藏少量加密或恶意数据提供了机会，使其在语法上保持有效，同时难以与良性的格式化伪影区分开来。

这种局限性是 PDF [1] 和 ZIP [39], 等格式基础规范固有的，因为这些规范并不要求有效文件中的所有字节都得到语义上的解释，也不要求通过内在完整性机制加以保护。因此，攻击者可以在这些宽松区域中嵌入密文，而不违反格式规范。例如，PDF 规范允许在某些注释行或未使用的间隙中存放任意二进制数据；而 ZIP 压缩档案中可能包含未被中央目录引用或未受条目级完整性检查覆盖的字节。仅依据基础规范运行时，这些区域仍会形成残余攻击面。

相比之下，由开放式封装约定（OPC） [2],定义的 OOXML 格式（.docx、.xlsx、.pptx）具有更严格的语义闭合性。所有有意义的内容都必须位于显式标注类型的包部件中，通过定义明确的关系进行引用，并且具有可确定的内容类型。因此，OOXML 包不允许存在同类语义不受约束的区域：落在 OPC 结构之外的字节属于结构违规，而非无害的附属数据。因此，PDF 和 ZIP 等宽松格式中存在的隐藏区域，在格式良好的 OOXML 文档中基本不存在。

为缓解 PDF 和 ZIP 等格式中宽松区域带来的风险，我们在安全敏感场景下施加超出基础规范的额外限制。我们通过检测器与符合要求的创作或打包工具之间建立的清洁格式契约来体现这一做法。根据该契约，除非有明确的完整性机制（例如加密校验和、可信清单或白名单）加以保护，否则文件应避免存在不受限制的间隙或不透明区域。任何偏离这些更严格不变量的情况——无论是由于保留了遗留内容、存在不受约束的填充，还是存在未计入的区域——都将被视为契约违规，并保守地标记为潜在的完整性或安全问题。如果无法假定存在此类契约，*Rhea* 便会将这些宽松区域视为规范不严谨的格式固有的残余攻击面。

攻击者还可以通过将密文编码为 ASCII 装甲表示（例如 Base64 或十六进制）来规避 SAWA 中基于一致性的 *Rhea* 检测。这些表示会将任意二进制数据映射到有限的可打印 ASCII 字符集合中。由于这类编码将字节限制在较小的字符集内（例如 A–Z、a–z、0–9、 +//），生成的数据每字节熵较低，看起来可能像无害的 UTF-8 文本，而非均匀随机的密文。为缓解这种规避手法，可以检测包含类似 Base64 或十六进制字符序列的区域，对其解码后再使用熵、幻数或格式感知验证重新评估；若其中存在密文，便可将其还原。还可以使用基于大语言模型的语义分析来评估解码内容是否格式正确且符合上下文，从而帮助区分无害的编码数据与隐蔽嵌入的密文。

此外，现代勒索软件越来越多地采用双重勒索手段：在从被攻陷设备窃取数据后，威胁泄露受害者的私人数据。这一趋势绕过了仅专注于检测本地加密活动的防御措施，包括 *Rhea*。有效的缓解措施需要通过网络出口过滤限制出站数据传输，监控异常压缩或上传行为，并落实最小权限访问控制。*Rhea* 与这类缓解措施相互独立，因此未来扩展其功能以检测数据窃取将十分直接。

Rhea 假定有可用的应用程序，能够在明确的时间点（例如保存、提交或应用级检查点边界）以集中方式执行文件更新。这一假设并不要求新的应用程序语义；相反，它符合企业和消费者环境中的常见部署实践。许多系统已经依赖定期整合或维护阶段，包括每夜备份、自动保存刷新以及计划软件更新。*Rhea* 利用这些现有窗口执行验证，使部署成为对当前工作流的切实扩展，而非采用障碍。

8 结论

本文介绍了*Rhea*，这是一种云辅助勒索软件防御系统，通过基于快照的格式感知内容验证，检测权限提升型和规避型攻击。*Rhea* 将检测建立在规范级正确性之上，而非熵或基于分布的信号，因此能够识别可逃避统计检测器的细粒度部分加密攻击。我们的评估表明，该系统在包括 TXT、PDF、OOXML 和 ZIP 在内的结构化格式中，即使面对微粒度加密，也能达到近乎完美的检测准确率，同时只产生实际可接受的开销。这些结果表明，格式感知、规范驱动的验证为防御现代勒索软件攻击提供了可扩展的基础。

致谢

参考文献

[1] 2008. ISO 32000-1:2008 — 文档管理 — 便携式文档格式 — 第1部分：PDF 1.7。[2] 2021. ISO/IEC 29500-2:2021 — 信息技术 — 文档描述与处理语言 — Office Open XML 文件格式 — 第2部分：开放封装约定。[3] 2025. No More Ransom 项目。https://www.nomoreransom.org/en/index.html。访问日期：2025-11-21。[4] abuse.ch。2025年。Malware Bazaar。https://bazaar.abuse.ch/。访问日期：2025-05-25。[5] Muhammad Ejaz Ahmed、Hyoungshick Kim、Seyit Camtepe 和 Surya Nepal。2021年。Peeler：通过分析内核级事件检测勒索软件。载于计算机安全—*ESORICS 2021*：第*26*届欧洲计算机安全研究研讨会，德国达姆施塔特，*2021*年*10*月*4*—*8*日，论文集，第*I*部分 *26*。施普林格，240–260。[6] Android 开源项目。2025年。*Android* 虚拟化框架 *(AVF)* 概述。https://source.android.com/docs/core/virtualization 访问日期：2025年11月7日。[7] Anunta。2025年。2025年顶级VDI趋势：企业如何保持领先。*Anunta* 科技博客 (2025)。https://www.anuntatech.com/blog/top-vdi-trends-of-2025-what-your-business-needs-to-stay-ahead/ [8] SungHa Baek、Youngdon Jung、Aziz Mohaisen、Sungjin Lee 和 DaeHun Nyang。2018年。SSD-Insider：通过完美数据恢复保护固态硬盘免受勒索软件侵害。载于*2018*年*IEEE*第*38*届分布式计算系统国际会议 *(ICDCS)*。IEEE，875–884。[9] Joonho Bang、Hyunho Kim 和 Sangjin Lee。2024年。勒索软件中的熵共享：绕过基于熵的加密操作检测。*Sensors* 24，5（2024年），1446。doi:10.3390/s24051446

[10] Sandeep Bhatkar、Abhishek Chaturvedi 和 R Sekar。2006年。数据流异常检测。载于2006年IEEE安全与隐私研讨会（S&P'06）。IEEE，15–页。[11] Seunghun Cha 和 Hyoungshick Kim。2016年。加密流量检测：一种机器学习方法。载于国际信息安全应用研讨会。施普林格，54–65。[12] Andrea Continella、Alessandro Guagnelli、Giovanni Zingaro、Giulio De Pasquale、Alessandro Barenghi、Stefano Zanero 和 Federico Maggi。2016年。ShieldFS：一种自我修复、具备勒索软件感知能力的文件系统。载于第32届年度计算机安全应用大会论文集（美国加利福尼亚州洛杉矶）（ACSAC '16）。计算机协会，美国纽约州纽约，336–347。doi:10.1145/2991079.2991110[13] Fabio De Gaspari、Dorjan Hitaj、Giulio Pagnotta、Lorenzo De Carli 和 Luigi V Mancini。2022年。压缩数据和加密数据的可靠检测。神经计算与应用 34，22（2022年），20379–20393。[14] UTM开发者。2024。UTM——适用于iOS和macOS的虚拟机。官方网站。https://getutm.app [15] Min Du、Feifei Li、Guineng Zheng 和 Vivek Srikumar。2017年。Deeplog：通过深度学习从系统日志中进行异常检测与诊断。载于2017年ACM SIGSAC计算机与通信安全会议论文集。1285–1298。[16] Farooq。 [未注明日期]。Screenlocker：屏幕锁定型勒索软件模拟。https: //github.com/farooq9/screenlocker。访问日期：2025-03-18。[17] Jae Hyun Han、Zhiqiang Lin 和 Michalis Polychronakis。2020年。基于行为的勒索软件检测的有效性。载于第16届EAI通信网络安全与隐私国际会议论文集（SecureComm）。 120 140 10 1007 978 3 030 63095 9_7 – 。doi: . / - - - - [18] Jian Huang、Jun Xu、Xinyu Xing、Peng Liu 和 Moinuddin K Qureshi。2017年。FlashGuard：利用闪存固有特性防御加密型勒索软件。载于2017年ACM SIGSAC计算机与通信安全会议论文集。2231–2244。[19] Jian-Syue Huang、Tsung-Han Liu、Yi-Hsien Chen、Hsuan-Yu Peng、Tse-Wei Huang、Chin-Laung Lei 和 Chun-Ying Huang。2024年。Time Machine：一种高效且可迁移后端的架构，用于在虚拟机监控器中防御勒索软件。载于2024年云计算安全研讨会论文集。66–79。[20] Amin Kharraz 和 Engin Kirda。2017年。Redemption：在终端主机上实时防护勒索软件。 98 119 10 1007 978 3 319 66332 6_5 – 。doi: . / - - - - Beom Heyn Kim 和 Hyoungshick Kim。未注明年份。Rocky：通过复制块设备构建防篡改、抗故障的边缘虚拟化桌面基础设施。载于第37届年度计算机安全应用大会论文集（线上活动，美国）（ACSAC '21）。计算机协会，美国纽约州纽约， [21] 2021 285 296 10 1145 3485832 3485886 – 。doi: . / . Engin Kirda。未注明年份。UNVEIL：一种大规模自动化勒索软件检测方法（主题演讲）。载于2017年IEEE第24届软件分析、演化与再工程国际会议（SANER）。1–1。doi:10.1109/SANER.2017.7 884603[23] Eugene Kolodenker、William Koch、Gianluca Stringhini 和 Manuel Egele。2017年。Paybreak：防御加密型勒索软件。载于2017年ACM亚洲计算机与通信安全大会论文集。599–611。[24] Klaudia Krawiecka、Arseny Kurnikov、Andrew Paverd、Mohammad Mannan 和 N. Asokan。2018年。SafeKeeper：使用可信执行环境保护网络密码。WWW '18：2018年万维网大会论文集， [22] 2017 349 358 10 1145 3178876 3186101 – 。doi: . / . Arseny Kurnikov、Andrew Paverd、Mohammad Mannan 和 N. Asokan。2018年。云端密钥：对加密凭据进行可审计的多设备访问。载于第13届可用性、可靠性与安全国际会议论文集（德国汉堡）（ARES '18）。计算机协会，美国纽约州纽约，文章40，10页。doi:10.1145/3230833.3234518[26] Boyang Ma、Yilin Yang、Jinku Li、Fengwei Zhang、Wenbo Shen、Yajin Zhou 和 Jianfeng Ma。2023。遍历虚拟机监控器与固态硬盘：一种结合细粒度数据恢复的基于标签方法，用于抵御加密型勒索软件。载于2023年ACM SIGSAC计算机与通信安全会议论文集。341–355。[27] Timothy R. McIntosh、Julian Jang-Jaccard、Paul A. Watters 和 Teo Susnjak。2019年。基于熵的勒索软件检测的不足。载于神经信息处理——第26届国际会议，ICONIP 2019，澳大利亚新南威尔士州悉尼，2019年12月12—15日，论文集，第V部分（计算机与信息科学通讯，第1143卷）。施普林格，181–189。doi:10.1007/978-3-030-36802- 9 20[28] [25] Shagufta Mehnaz、Anand Mudgerikar 和 Elisa Bertino。2018年。RWGuard：一种针对加密型勒索软件的实时检测系统：第21届国际研讨会，RAID 2018，希腊克里特岛伊拉克利翁，2018年9月10—12日，论文集。 114 136 10 1007 978 3 030 00470 5_6 – 。doi: . / - - - - 微软公司。未注明年份。部署BitLocker管理——Configuration Manager。 [29] 2023 https://learn.microsoft.com/en-us/intune/configmgr/protect/deploy-use/bitlocker/deploy-management-agent。最后更新于2023年2月9日。[30] 微软公司。2025。BitLocker驱动器加密。https://learn.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-概述 访问日期：2025-03-13。[31] 微软公司。2025。托管磁盘加密选项概述 – Azure 虚拟机。https://learn.microsoft.com/en-us/azure/virtual-machines/windows/disk-encryption-overview。最后更新于 2025-09-24。 [32] Donghyun Min、Yungwoo Ko、Ryan Walker、Junghee Lee 和 Youngjae Kim。2021。一种基于内容的勒索软件检测与备份固态硬盘，用于防御勒索软件。IEEE 集成电路与系统计算机辅助设计汇刊 41, 7 (2021), 2038–2051。 [33] Yisroel Mirsky、Tomer Doitshman、Yuval Elovici 和 Asaf Shabtai。2018。Kitsune：用于在线网络入侵检测的自编码器集成。arXiv 预印本 arXiv:1802.09089 (2018)。[34] Tobias Müller、Jakob Rieck、Florian Wilkens、Luca Glockow、Nicholas Farnham、Jannes Quer、Matthias Marx 和 Dominik Oepen。2024。Black Basta Buster：无需支付赎金即可解密文件。在线博客文章。https://srlabs .de/blog/black-basta-buster-decrypting-files-without-paying-the-ransom 访问日期：2026年1月27日。 [35] Lindsey O'Donnell-Welch。2023。BlackCat 勒索软件操作者利用恶意驱动程序规避检测。https://duo.com/decipher/blackcat-ransomware-actors-use-malicious-drivers-to-evade-detection（最后访问日期：09/03/2025）。[36] 甲骨文公司。2025。Oracle VM VirtualBox（版本 7.0.12）。可从 https://www.virtualbox.org/ 获取。访问日期：2025-11-07。[37] Jisung Park、Youngdon Jung、Jonghoon Won、Minji Kang、Sungjin Lee 和 Jihong Kim。2019。RansomBlocker：一种低开销的防勒索软件固态硬盘。载于《第 56 届设计自动化会议论文集》（2019）。1–6。 [38] Sarah Parvini。2025。网络安全官员警告 Medusa 勒索软件攻击可能造成高昂损失。https://apnews.com/article/fbi-cisa-gmail-outlook-cyber-security-email-6ed749556967654ff41a629a230973e6。最后访问日期：17/03/2025。[39] PKWARE 公司。2023。APPNOTE.TXT：ZIP 文件格式规范。https: // pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT。版本 6.3.10。 [40] ransomware-cl0p 2025。2025。ESXi 上的勒索软件：Cl0p Linux 变种的机制化。https://thehackernews.com/2025/01/ransomware-on-esxi-mechanization-of.html?utm_source=chatgpt.com 访问日期：2025-03-20。 [41] ransomware-ghost 2025。2025。Ghost 勒索软件活动：指标与缓解策略。https://www.cisa.gov/news-events/cybersecurity- advisories/aa25-050a?utm_source= chatgpt.com 访问日期：2025-03-20。 [42] ransomware-mallox 2024。2024。Mallox 勒索软件变种以拥有特权的 VMware ESXi 环境为目标。https://www.darkreading.com/cloud-security/mal lox-ransomware-variant-targets-privileged-vmware-esxi-environment?utm_ source= chatgpt.com 访问日期：2025-03-20。 [43] ransomware-ransomhub 2024。2024。RansomHub 从不休眠：第 1 集。https: // www.group-ib.com/blog/ransomhub-never-sleeps-episode-1/?utm_source= chatgpt.com 访问日期：2025-03-20。 [44] Benjamin Reidys、Peng Liu 和 Jian Huang。2022。RSSD：通过硬件隔离的网络存储协同设计和攻击后分析抵御勒索软件。载于第 27 届 ACM 编程语言与操作系统架构支持国际会议论文集（瑞士洛桑）（ASPLOS '22）。计算机协会，纽约，美国，726–739。doi:10.1145/3503222.3507773[45] Mahadev Satyanarayanan、Victor Bahl、Ramon Caceres 和 Nigel Davies。2009。论基于虚拟机的云端小站在移动计算中的价值。IEEE 普适计算（2009 年 11 月）。https://www.microsoft.com/en-us/research/publication/ the-case-for-vm-based-cloudlets-in-mobile-computing/ [46] Nolen Scaife、Henry Carter、Patrick Traynor 和 Kevin RB Butler。2016。加密锁（然后丢弃）：阻止针对用户数据的勒索软件攻击。载于《2016 IEEE 第 36 届分布式计算系统国际会议》（ICDCS）。IEEE，303–312。[47] Sead Fadilpašić。 [无日期]。这种狡猾的勒索软件能够劫持你的系统并关闭 Microsoft Defender。https://www.techradar.com/pro/security/this-devious-ransomware-is-able-to-hijack-your-system-to-turn-off-microsoft- defender。发布日期：2025年8月7日；访问日期：2025年8月18日。 [48] SentinelOne。2023。Play 勒索软件：深入分析、检测与缓解。https://www.sentinelone.com/anthology/play/（最后访问日期：06/05/2025）。[49] ShardSecure。2023。间歇性加密勒索软件详解。https: // shardsecure.com/blog/intermittent-encryption-ransomware（最后访问日期：09/03/2025）。[50] Swaminathan Sivasubramanian。2012。Amazon DynamoDB：可无缝扩展的非关系型数据库服务。载于《2012 年 ACM SIGMOD 数据管理国际会议论文集》。729–730。 [51] Fei Tang、Boyang Ma、Jinku Li、Fengwei Zhang、Jipeng Su 和 Jianfeng Ma。2020。RansomSpector：一种基于内省的方法，用于检测加密勒索软件。《计算机安全》。 97, 2020 14 10 1016 2020 101997 C（oct），页码。 [52] doi: 。/j.cose.。 2025 TechRadar Pro。您的防病毒软件正遭到新型“杀手”工具攻击——以下是目前所知信息。https://www.techradar.com/pro/security/your-antivirus-is-under-attack-from-new-killer-tool-heres-what-we-know。访问日期：2025-08- 18。[53] THU-WingTecher。2025。MarauderMap：来自 95 个家族的最新 7,796 个活跃且独特的勒索软件样本。https://github.com/THU-WingTecher/Mar

auderMap。访问日期：2025-05-25。 [54] Alanna Titterington。2024。勒索软件：2024 年最值得关注的攻击。https: // www.kaspersky.com/blog/ransowmare-attacks-in-2024/52949/（最后访问日期：09/03/2025）。[55] Christian van Sloun、Vincent Woeste、Konrad Wolsing、Jan Pennekamp、Klaus Wehrle、Antoine Saillard、Jan Bauer、Eric Wagner、Ina Berenice Fink、Mari Schmidt 等。2025。尽管存在 I/O 开销，仍能检测勒索软件：一种实用的多阶段方法。载于网络与分布式系统安全研讨会，第 978 卷。互联网协会，3543–3545。 [56] VirusTotal。2026。*VirusTotal — 免费在线病毒、恶意软件和 URL 扫描器*。https://www.virustotal.com/gui/home/upload 访问日期：2026-01-15。[57] Shenao Wang 、 Feng Dong 、 Hangfeng Yang 、 Jingheng Xu 和 Haoyu Wang。2024。CANCAL：面向工业环境的实时轻量级勒索软件检测与响应。载于 *2024 年 ACM SIGSAC 计算机与通信安全会议论文集*。2326–2340。 [58] xcitium。2025。勒索软件的不同类型。https://www.xcitium.com/types- of-ransomware/（最后访问日期：11/03/2025）。

[59] Xoureldeen. 2024. Vectras VM - Android虚拟机模拟器。GitHub 仓库。https://github.com/xoureldeen/Vectras-VM-Android [60] Joseph Young. 2025. 从屏幕锁定型勒索软件中恢复。https://www.youngfamilydigital.com/blog/recovering-from-screen-locking-ransomware 访问日期：2025-03-18。[61] Lianying Zhao 和 Mohammad Mannan. 2019. 借助可信执行环境防止特权数据篡改的写保护。*arXiv* 预印本 *arXiv:1905.10723* （2019）。[62] Lingbo Zhao、Yuhui Zhang、Zhilu Wang、Fengkai Yuan 和 Rui Hou。 [n. d.]。ERW-Radar：通过上下文行为检测和细粒度内容分析抵御规避型勒索软件的自适应检测系统。（[n. d.]）。[63] Chijin Zhou、Lihua Guo、Yiwei Hou、Zhenya Ma、Quan Zhang、Mingzhe Wang、Zhe Liu 和 Yu Jiang。2023。基于 I/O 的勒索软件检测的局限：一种模仿型攻击。载于*2023 IEEE安全与隐私研讨会（SP）*。2584–2601。doi:10.1109/SP46215.2023.10179372[64] Weidong Zhu、Carson Stillman、Sara Rampazzi 和 Kevin R.B. Butler。2025。借助保留感知型版本控制SSD实现安全高效的数据丢失防护。载于第*32*届*ACM SIGSAC*计算机与通信安全会议（*CCS '25*）（中国台湾台北）。ACM。