基于主机事件的攻击发现技术研究综述

袁军翼[1], 杨志鹏[2], 刘代东[2], 魏松杰[1,2]

[1]南京理工大学 计算机科学与工程学院 南京 中国 210094
[2]南京理工大学 网络空间安全学院 南京 中国 210094

摘要 勒索软件攻击以其瞬间的数据劫持速度、众多变体和高度伪装能力, 对网络安全构成了巨大挑战, 使得依赖长期行为监测的传统检测方法在实时监测能力和准确性上应对乏力。本文提出了一种新型的勒索软件检测方法, 以硬件性能计数器(Hardware Performance Counters, HPC)为特征, 融合无监督学习与有监督学习方法, 提高了针对勒索软件的检测准确性和实时性。该方法从程序执行的初始阶段开始收集硬件性能计数器的事件数据, 特别关注实际加密行为发生前的数据特征。硬件性能计数器可以区分多种程序行为事件, 为分析攻击软件加密行为发生前的系统状态, 提供了丰富全面的监测信息。为了区分无关事件, 本文通过 Boruta 算法, 进一步从计数器数据中筛选出最能识别勒索软件行为的核心硬件事件。使用 Transformer 网络架构, 结合序列预测和对比学习这两种无监督学习任务, 在分析未标记的计数器时间序列数据中, 实现数据表征和分类。此外, 通过在有标记的数据集上进行微调和迭代, 检测模型进一步提高了准确性。本文方法在 27 类勒索软件家族数据集上进行了广泛的实验测试, 实验结果能有效检测勒索软件的动态行为事件, 在 10 s 的检测窗口中检测准确率高达 98.2%, 性能超过同类检测方法。

关键词 勒索软件; 恶意软件检测; 网络安全; 硬件性能计数器; 动态检测
中图法分类号 TP309.5 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2025.11.10

Detecting Ransomware with Hardware Performance Counters

YUAN Junyi[1], YANG Zhipeng[2], LIU Daidong[2], WEI Songjie[1,2]

[1] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
[2] School of Cyber Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China

Abstract Ransomware attacks are increasingly recognized as a formidable threat to cybersecurity, characterized by their rapid data hijacking capabilities, a multitude of variants, and sophisticated levels of disguise. These malicious threats continuously evolve, posing significant challenges to traditional detection methodologies that predominantly rely on long-term behavioral monitoring. Such conventional approaches often struggle with limitations in real-time monitoring capabilities and fail to deliver the necessary accuracy. In response, this paper introduces a novel ransomware detection method that utilizes hardware performance counters (HPC) as distinctive features, ingeniously combining unsupervised and supervised learning techniques to significantly enhance detection efficiency and accuracy. This method starts by meticulously collecting event data from hardware performance counters at the onset of program execution, focusing specifically on data characteristics before the actual encryption activities. Hardware performance counters can monitor hundreds of system events, providing a rich and comprehensive dataset that details the pre-encryption state of the system. However, many events may be irrelevant to detecting malicious activities. To address this challenge, the Boruta algorithm is employed to intelligently filter out the most indicative hardware events associated with ransomware behaviors, refining the data for further analysis. Building on this filtered dataset, the model leverages a sophisticated Transformer network architecture that integrates sequence prediction and contrastive learning—two powerful types of unsupervised learning tasks. This integration enables the model to develop robust data representation capabilities, effectively analyzing and interpreting unlabeled time-series data. The approach's robustness lies in its ability to capture and learn from the subtle nuances and patterns distinguishing normal operations from potential threats. Further enhancing the model's efficacy, it undergoes fine-tuning on a carefully curated labeled dataset, transitioning into the supervised learning phase to refine predictive accuracy. The practical application and effectiveness of this method have been rigorously tested on a dataset that includes 27 distinct ransomware families. The results from these extensive tests are highly encouraging. They demonstrate that the proposed method not only effectively detects ransomware but does so with remarkable precision, significantly outperforming traditional detection methods. The method achieved a detection accuracy rate of up to 98.2% within a 10-second detection window. The performance exceeds that of similar detection methods.

通信作者: 魏松杰, 博士, 副教授, Email: swei@njust.edu.cn。
本课题得到了国家重点研发计划项目(No. 2020YFB1804604), 国家自然科学基金项目(No. 61802186)资助。
收稿日期: 2024-02-06; 修改日期: 2024-04-18; 定稿日期: 2025-10-14

Key words ransomware; malware detection; cybersecurity; hardware performance counters; dynamic detection

1 引言

随着数字经济的高速发展, 勒索软件通过加密方式来劫持用户数据并阻止访问, 对全球网络安全构成了严重威胁。特别是勒索软件即服务(RaaS)模式的出现, 大幅降低了发起类似攻击的技术门槛, 使得勒索软件数量与日俱增[1]。

勒索软件攻击的复杂性不断增加, 其中使用的代码混淆技术和高速数据劫持能力使得它们难以被及时检测和阻止。以 LockBit 3 为例, 其每分钟可以加密超过 25,000 个数据项, 显著缩短了安全响应的可行时间窗口[2]。这种快速劫持速率严重威胁到数据完整性, 极大加剧了受害者的检测和防御压力。

本文通过检测系统被勒索软件劫持后第一个加密文件出现时间, 分析了不同勒索软件家族在受控沙箱环境中的预加密阶段耗时。例如在装有 4 GB RAM 和 2.5GHz 处理器的典型 Windows 系统上, 测试了包含 2321 个文件(总计 27.31 GB)的数据集。如图 1 所示, 超过 74%的勒索软件家族在启动后的前 10 秒内尚未开始加密, 凸显了及时检测勒索软件的挑战和时机。勒索软件检测的即时性是网络安全领域的一个至关重要但极具挑战性的研究任务[3]。

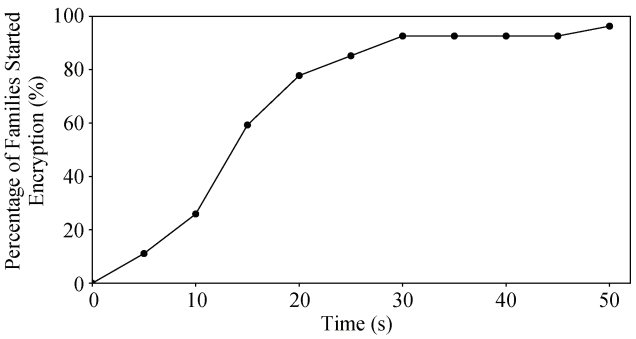

图 1 勒索软件预加密阶段耗时

Figure 1 Ransomware pre-encryption time

随着勒索软件的不断变异, 传统的检测方法, 例如通过反编译提取可执行文件的静态特征或在受控环境中采集大量动态特征, 尽管易用和灵活, 但面临高资源消耗、依赖频繁更新及高误报率的问题[4]。

与上述方法相比, 硬件辅助的恶意软件检测(Hardware-assisted Malware Detection, HMD)技术通过从硬件寄存器直接提取数据, 有效降低了监控资源开销, 并能抵抗检测逃避和代码混淆[5]。近年来, 英特尔便针对勒索软件攻击威胁提出了威胁检测技术(Threat Detection Technology, TDT), 其利用 CPU 的遥测数据来提供更深入的威胁分析检测能力, 扩大可见性以提高系统的整体安全性能[6]。

在 HMD 研究中, 广泛使用的硬件性能计数器(Hardware Performance Counters, HPC)能监控诸如 CPU 周期、缓存缺失率等底层硬件系统事件信息, 这些是恶意软件制作者难以干扰的。特别是在监测勒索软件的早期文件扫描和加密行为方面显得尤为有效。然而, HPC 的敏感性和测量值对硬件配置和系统负载的反应性提出了高效利用的挑战[7]。

针对这一问题, 本文提出了一种基于 HPC 的勒索软件检测方法。该方法以来自 HPC 的整数流为分析对象, 通过深度模型捕捉勒索软件在运行初期与正常软件运行时的显著差异, 实现对勒索软件和良性软件的有效区分。本文的主要贡献为。

(1) 对勒索软件 HPC 数据进行了分析建模, 通过特征提取和表达, 对 HPC 事件进行重要性分析;

(2) 通过融合对比学习任务和时序重构任务这两种训练策略, 从整体时间序列和单点时间序列两个关键维度分别提取特征分类特征和时间戳特征;

(3) 构造了包含真实勒索软件样本和正常软件的测试环境和样本集, 实现了实时自动的检测流程, 结果具有高准确度和实时性。

2 相关工作

2.1 硬件性能计数器

硬件性能计数器(HPC)是现代处理器中的专用寄存器, 用于监控各种系统活动, 提供对软件和硬件交互的深入洞察。在现代的硬件架构中, 每个处理器核心都配备了多个性能监控单元(Performance Monitoring Units, PMUs), 每个 PMU 都有独立的资源并能够支持一系列不同的监测事件。如表 1 所示, HPC 能够记录多种事件, 例如指令执行、分支预测及缓存访问, 这些都是分析软件行为的关键数据。

表 1 HPC 事件

Table 1 HPC event

事件	描述
Instruction Retired	计数指令最后一个微操作执行完毕的情况
Branches	计数程序执行过程中的分支指令数量
Brancher-misses	计数预测错误的分支指令数量
Cache-references	CPU 访问缓存内存的总次数
Cache-misses	CPU 尝试从缓存内存中检索数据失败数量

自从 Malone 等人提出以来, HPC 已经证明其在

网络安全领域内作为可靠的硬件特征的价值, 尤其在最小化性能损耗的同时, 能有效监控和检测勒索软件等恶意软件的异常活动[8-9]。这些活动包括频繁的内存访问, 以及与常规软件明显不同的数据加密和控制流模。勒索软件与良性软件的行为存在显著差异, 包括密集的数据加密操作、复杂的控制流、非常规代码路径和复杂决策逻辑。这些特征增加了分支预测难度并导致高缓存引用次数, 为基于 HPC 的勒索软件检测模型提供了依据。

尽管 HPC 提供了实时监控和系统性能的深度视图, 但它的测量结果会受到硬件配置、系统负载和操作系统的影响, 存在一定的不确定性[10]。即使是相同程序在不同系统上的HPC数据也可能不同。此外, 鉴于可监控的事件数量高达上百个, 选择有效特征并高效使用 HPC 数据, 需要精确的算法和充足的数据样本, 以提高恶意软件检测的准确性和效率。

2.2 基于 HPC 的恶意软件检测方法

近年来, 基于硬件性能计数器的恶意软件检测技术已引起了研究者的广泛关注, 这种方法利用硬件层面的系统数据来识别潜在的程序恶意活动, 是对传统软件层面检测方法的有力补充。其中建立有监督的机器学习模型方法, 通过利用 HPC 的状态值作为特征, 结合已标记的程序样本, 来训练恶意软件检测模型。Demme 等人[11]验证了此方法的可行性, 并证实了这些基于 HPC 的模型在检测恶意软件变体方面的有效性和鲁棒性。Patel 等人[12]对基于 HPC 分类器的硬件实现及其成本进行分析, 指出高准确性的模型在硬件实现后可能面临综合性能下降的问题。牛伟纳等人[13]通过 ROP 攻击会导致缓冲区未命中数异常增加、频繁出现有错误预测返回指令等特点, 通过采集相关 HPC 数据, 通过设置阈值的方法实现检测。姚梓豪等人[14]通过相关性分析从 HPC 数据中获取与缓存侧信道攻击相关事件, 结合机器学习算法实现检测。

无监督型模型也在恶意软件检测中展现出其独特优势, 特别是在处理新型攻击或攻击者行为演变方面。研究者 Tang 等人[15]探讨了使用无监督方法检测 ROP 攻击和栈溢出攻击的可能性, 通过分析受保护良性程序的HPC值变化来识别攻击。Garcia-Serrano 等人[16]通过检测局部异常因子, 尝试检恶意软件。Carnà 等人[17]通过深入分析 HPC 采集原理以及侧信道攻击方式, 构建了多种规则来检测侧信道攻击。这可能增加分析的复杂度, 但提供了检测的灵活性。

总的来说, HPC 在针对特定恶意软件检测时表现出了强大的潜力, 但多样的 HPC 事件以及 HPC 潜在的不确定性仍然限制了检测效果。

2.3 多元时间序列分类方法

HPC 会生成统一形式的整数流, 这种形式的数据自然而然地构成了时间序列。多元时间序列分类(Multivariate Time Series Classification, MTSC)方法因此在处理 HPC 数据时显得尤为重要。

在 MTSC 的研究中, Transformer 架构因其出色的数据模式处理能力而成为研究热点。Li 等人[18]使用具有 LogSparse 结构的卷积自注意力层来捕获局部信息并降低计算复杂度。Wu 等人[19]借鉴了传统时间序列分析方法使用 ProbSparse 自注意力机制处理超长输入序列。Nie 等人[20]引入图像处理中的 patch 机制, 通过划分原始序列为一个个子序列, 有效捕捉了时间序列的局部特征。此外, 对比学习在 MTSC 中的应用也开始受到关注, 因为它可以通过优化相似性和差异性来提高模型对时间序列数据的理解。Saeed 等人[21]通过对数据进行多次转换, 设计了一个能有效识别人类活动行为的模型。van den Oord 等人[22]通过在潜在空间中预测未来数据来学习表征; Sarkar 等人[23]通过对数据集进行 6 种转换来学习表征。

总的来说, 结合 Transformer 模型的序列处理能力, 以及对比学习的差异数据优化, 为多元时间序列分类提供了一种强大的工具。

综上所述, HPC 数据在恶意软件检测中的应用面临着数据敏感性高、数据复杂度大等挑战; 且现有 HPC 在勒索软件检测中未对 HPC 事件如何选择做出讨论。针对上述问题, 本文采用了对比学习和基于 Transformer 骨架的技术, 利用其在时间序列分析中的强大数据表征能力, 有效地克服了 HPC 数据的敏感性和复杂性问题。并使用基于模型的特征筛选方法, 对 HPC 事件在勒索软件检测领域中的重要性进行分析, 从而筛选出核心事件, 提升模型效果。

3 基于硬件性能计数器的勒索软件攻击检测方法

本文提出的基于硬件性能计数器的勒索软件检测方法整体流程如图 2 所示, 使用的数据包括监督数据集 $S=\left\{\left(x_i,y_i\right)\middle|x_i\in\Re^{T\times C},y_i\in\{0,1\}\right\}$ 以及无监督数据集 $U=\left\{x_i\middle|x_i\in\Re^{T\times C}\right\}$, $\Re^{T\times C}$ 为采集到的 HPC 样本合集, T 为时间序列长度, C 为序列维度, 每一行为运行时间 T 内特定事件的值, 每一列表示所有事件在时间 i 上的值; y_j 为样本标签, 0 表示样本为良性软

件, 1 表示样本为勒索软件。本方法主要分为三个部分: 数据采集、HPC 特征选择以及检测模型构建。在数据采集阶段, 使用虚拟机环境执行勒索软件并通过 Perf 工具收集相关的运行数据。接着, 对采集得到的特征进行特征选择, 降低冗余特征, 并构建数据集。最后, 以无监督数据集 U 的全部样本作为输入来训练无监督模型, 旨在捕捉和理解 HPC 数据的底层结构和模式, 在监督数据集 S 上获取分类特征对训练模型进行分类器构建, 以识别勒索软件, 从而得到最终的勒索软件检测模型。

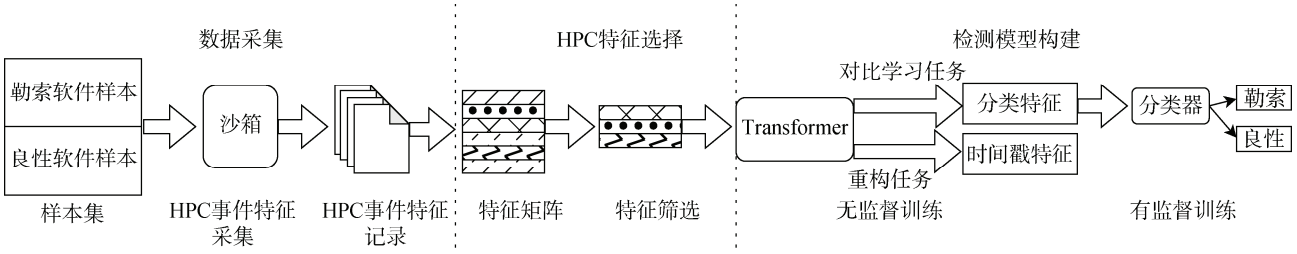

图 2 检测流程

Figure 2 Detection process

3.1 数据采集

本文采用 HPC 特征来检测勒索软件, 侧重于记录处理器运行时的 HPC 整数流。尽管 HPC 能够支持大量事件的监测, 但其内置的计数器数量通常有限, 这就意味着实际能够同时监测的事件数量受限于这些可用计数器的数目, 例如: Intel 的 Nehalem 架构提供了四个以上性能计数器, Sandy Bridge 架构增加性能计数器数量到八个, Skylake 架构开始 Intel 引入了更多高级性能监控特性和性能计数器[24]。尽管性能计数器数量有限, 用户仍可通过软件多路复用等技术, 设置合适采集间隔, 在牺牲一部分采样精度的情况下, 超越硬件限制, 实现对事件的无限制监控。

为降低风险并控制系统潜在破坏, 本研究在 Linux 托管的 Windows 7 虚拟机上的受控沙箱环境中收集了勒索软件的 HPC 数据。通过自动化脚本, 生成了 2321 个不同类型和大小的文件, 总计 27.31 GB, 模拟了用户设备上的数据多样性, 为收集不同样本运行时数据提供了一个内容丰富的测试环境。

前文的实证分析已经说明勒索软件一般预加密阶段持续 10s 左右, 故而本采集系统从减少监控记录数据量以及及时检测的角度考虑, 对每一个样本只持续监控 20s 的运行过程, 仅采集早期数据并且减少记录数据量。该采集系统以可执行勒索软件样本与良性软件样本为输入, 输出并保存样本预加密阶段所有 HPC 事件数据。

为确保实验一致性, 研究建立了未运行勒索软件前的系统基准快照。每次样本运行 20 秒后, 系统将回滚至快照, 重复多次执行样本, 以丰富数据集并减少 HPC 数据敏感性对估值一致性的影响。执行同一样本多次后, 系统将加载下一个样本。通过这种方法, 最终构建了勒索软件及良性软件样本的 HPC 数据集, 采集得到的部分数据如图 3 所示。

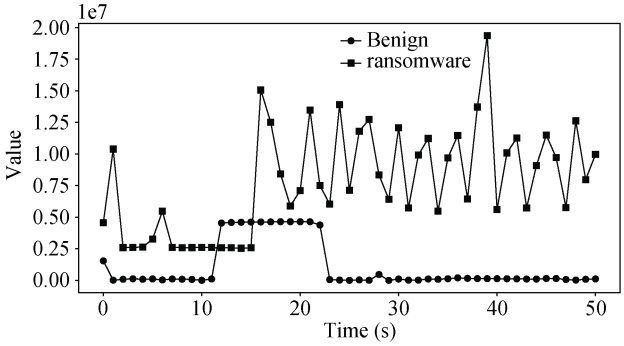

图 3 Cache-references 数据

Figure 3 Cache-references data

3.2 HPC 特征选择

尽管性能计数器的底层设计保证大部分事件在采集过程中对性能没有影响或者可以忽略不计, 但部分硬件事件需要额外使用计数器并结合相关逻辑进行运算, 仍然会增加额外资源开销。此外, 同时进行更多的事件采集意味着需要更长的采集间隔以及更低的采集精度。由于底层事件数量巨大, 在实际使用过程中长期监测软件所有事件并不现实, 为降低 HPC 特征的复杂度, 提高采集精度, 降低性能损耗, 本文从针对勒索软件检测角度出发, 使用 Boruta 算法以提取最有效事件[25]。

Boruta 是基于随机森林的特征选择方法, 可以全面评估和选择出与勒索软件行为最相关的特征, 而非仅仅是最小化模型损失, 从而提高检测系统的准确性和效率。它通过随机打乱原始特征创建影子特征(Shadow Features)并与原始特征拼接作为特征进行随机森林训练, 并使依赖于随机森林算法得出的特征重要性评分。在这个过程中, Boruta 不断地对

比原始特征和影子特征的重要性, 如果实际特征更重要则保留; 否则删除特征。随后使用删减后的特征进一步迭代。重复此过程直到预先确定的迭代次数或者所有特征都被删除, 算法流程如图 4 所示。

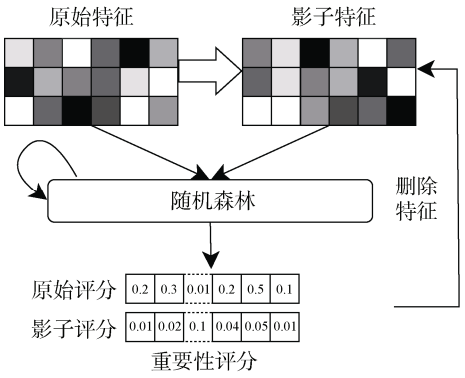

图 4　Boruta 算法流程

Figure 4　Boruta algorithm process

图 5 展示了 Boruta 特征选择技术的结果。以 20s 内的 HPC 统计数据中可以获得迭代过程中每个特征的平均重要性, 并排列出最具显著性的特征。

通过应用 Boruta 算法, 最终可以从大量 HPC 事件中识别出针对勒索软件检测最关键的特征。

3.3　检测模型构建

整体模型框架如图 6 所示, 其中首先对时间序列 $x \in \Re^{T \times C}$, 采用实例归一化(Instance Normalization, IN)处理以缓解分布偏移问题。接着, 利用 patch 方法将时间序列分割成多个较短的子序列, 缩短时间维度 T 至 T_P, 特征维度从 C 增大至 $C \cdot P$, 其中 P 是每个 patch 的长度。此步骤不仅提升了模型学习局部模式的能力, 同时也减少了 Transformer 的 token 数量。为了增强自监督学习过程, 应用了掩码策略, 遮盖 75%的片段, 迫使其基于未被掩盖片段提供的上下文来预测缺失的信息, 从而使模型学习出对下游任务有用的鲁棒性表示。最后, 一个特殊的向量[CLS]被添加至这些序列开头, 通过自注意力机制有效地整合整个序列的上下文信息, 从而捕捉实例级向量表征。

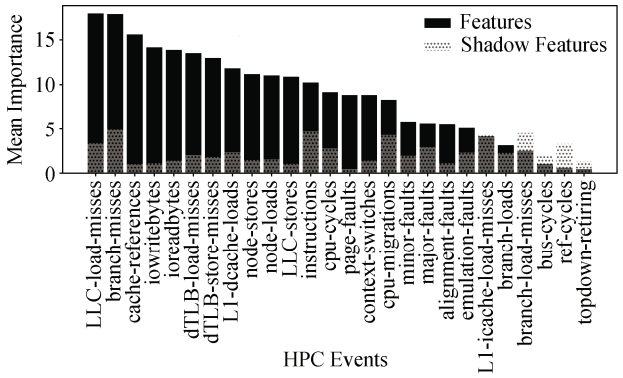

图 5　Boruta 结果

Figure 5　Boruta result

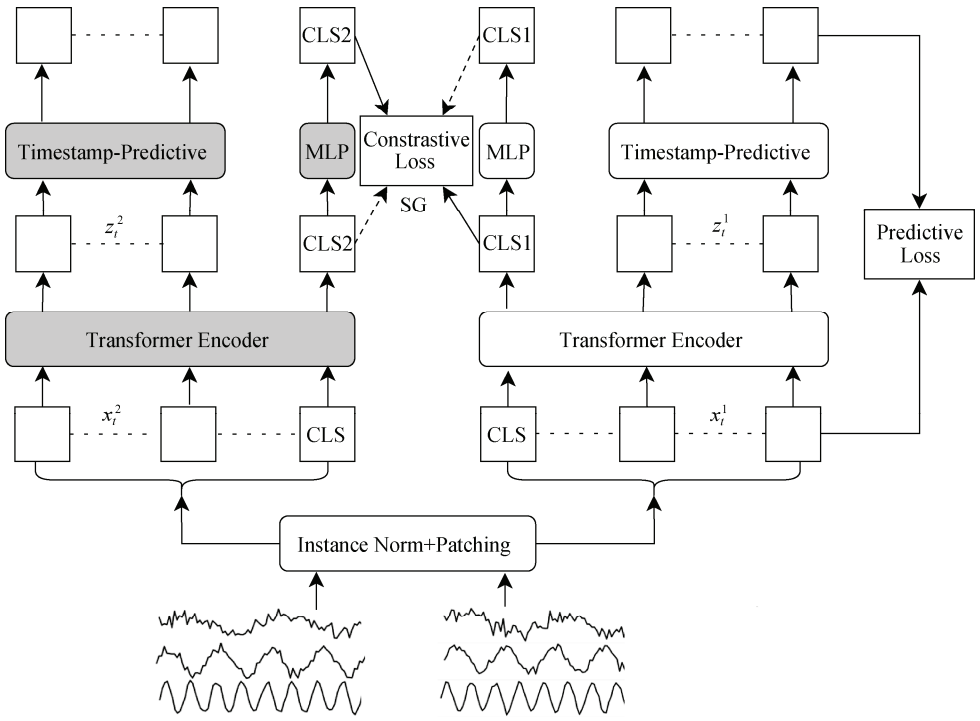

图 6　模型框架

Figure 6　Model framework

最终形成了 Transformer Encoders 的输入 $x_{\text{enc-in}}$，其维度为 $\Re^{(1+T_p)\times C\cdot P}$，计算公式如下所示:

$$x_{\text{enc-in}}=\text{concatenate}([\text{CLS}],\text{patch}(\text{IN}(x))) \tag{1}$$

随后，$x_{\text{enc-in}}$ 经过一个线性词元编码层 $W_{\text{token}}\in\Re^{D\times C\cdot P}$ 和一个线性位置编码层 $W_{\text{pos}}\in\Re^{D\times(1+T_p)}$ 组成的前馈网络，以及一系列自注意力模块 SAs，这个编码器产生最终的嵌入 $z\in\Re^{(1+T_p)\times D}$:

$$z=\text{SAs}\left(W_{\text{token}}x_{\text{enc-in}}+W_{\text{pos}}\right) \tag{2}$$

其中，D 表示 Transformer 隐藏层的维度。

在使用 Transformer Encoders 获得时间序列嵌入 z 之后，分别获得两个不同层面的嵌入 z_i 和 z_t:

$$z_i=z[0,:] \tag{3}$$

$$z_t=z\left[1:T_p+1,:\right] \tag{4}$$

z_i 用于表示[CLS]所代表的整体序列级别的嵌入，以及 z_t:用于表示单点时间戳级别的嵌入。

对于时间戳级别的嵌入 z_t，使用一个全连接层作为时间戳预测(Timestamp-Predictive)头 p_θ 用于生成预测序列。预测损失 L_p 是通过计算预测被掩码部分的输入输出之间的均方误差得出的:

$$L_p=\text{MSE}(x,p_\theta(z_t)) \tag{5}$$

为了精确捕捉时间序列的全局特征，采用了基于对比学习的孪生网络框架，并使用对比损失来优化序列级表征。此方法通过分析同一样本生成的两个嵌入来增强模型的鲁棒性，并有效区分不同实例。特别是在处理 HPC 数据时，考虑到数据的敏感性，孪生网络分别输入同一样本的两次采集数据，通过对比损失，这种设计将两组嵌入特征相互靠近，即使在数据微小波动的情况下也能维持一致的整体序列表征，提升了模型对时间序列全局特征的捕捉能力。

通过两次将同一样本的不同数据通过编码器，从同一样本的不同数据生成两个不同的嵌入特征，z^1 以及 z^2。然后，对于每个嵌入，提取第一个位置作为序列的整体嵌入:

$$z_i^1=z^1[0,:] \tag{6}$$

$$z_i^2=z^2[0,:] \tag{7}$$

通过编码器中获得两个特征向量 z_i^1 和 z_i^2 后，每个嵌入都通过一个 MLP 模块 c_θ 进行处理，如图 7 所示，以避免后续的对比学习任务不会因为编码器性能过强而导致模型只学习到简单特征。最后产生 $\hat{z}_i^1$ 和 $\hat{z}_i^2$:

$$\hat{z}_i^1=c_\theta\left(z_i^1\right) \tag{8}$$

$$\hat{z}_i^2=c_\theta\left(z_i^2\right) \tag{9}$$

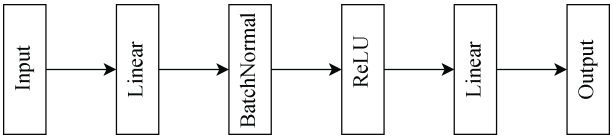

图 7　MLP 模块

Figure 7　MLP module

在计算对比损失时，目标是使 $\hat{z}_i^1$ 与 $\hat{z}_i^2$ 尽可能相似，保证同一样本得到一致的特征。将 z_i^2 作为对比损失中的常数，确保了主干模型更新仅基于预测的 $\hat{z}_i^1$，而不接收来自 z_i^2 的梯度。通过极小化这两个向量的负余弦相似度，来进行对比学习过程，提取时序数据鲁棒的特征表示。负余弦相似度计算如公式所示:

$$S(p,q)=-\frac{p}{\|p\|_2}\cdot\frac{q}{\|q\|_2} \tag{10}$$

其中，$\|\cdot\|_2$ 是 L2 范数。为了有效避免对比学习出现崩溃解的问题，引入了 stop-gradient 操作。因此对比学习的损失函数定义如公式(14)所示:

$$\mathcal{L}_{C^1}=-S\left(\hat{z}_i^1,\text{stop-gradient}\left(z_i^2\right)\right) \tag{11}$$

类似地，对于孪生网络，为了能对称地优化网络，使用同样的方式计算了 $\hat{z}_i^2$ 与 z_i^1 之间的对比损失:

$$\mathcal{L}_{C^2}=-S\left(\hat{z}_i^2,\text{stop-gradient}\left(z_i^1\right)\right) \tag{12}$$

模型损失函数 $\mathcal{L}_C$ 为

$$\mathcal{L}_C=\frac{1}{2}\mathcal{L}_{C^1}+\frac{1}{2}\mathcal{L}_{C^2} \tag{13}$$

最后，联合时序预测任务以及对比学习任务，整个模型的总损失函数为

$$\mathcal{L}=\mathcal{L}_p+\mathcal{L}_C \tag{14}$$

通过自监督学习，编码器被训练以解析和内化数据结构。这一阶段生成了代表整体时间序列的分类特征和单点时间序列的时间戳特征。训练完成后，编码器的权重被冻结以保持特征提取能力。

在编码器之上加入了全连接的线性层作为分类任务的决策层，负责将整体输入序列的表征映射到预定类别空间。接着，通过 softmax 层将线性层输出转换为分类概率，该层将输出标准化成概率分布，每个节点的值代表输入序列属于相应类别的概率。

4 实验结果与分析

4.1 数据集

为验证本文提出的勒索软件检测方法的有效性, 在 Cuckoo 沙箱上构建了 Windows7 操作系统作为软件的运行环境, 并使用 Python 脚本在桌面及磁盘分区自动生成多种类型的文档。勒索软件样本从 Virus-Share 采集, 选取超过 75%反病毒引擎认同的判断结果作为标签。正常样本采集于 Ninite 中, 覆盖了杀毒软件、办公软件、聊天工具、视频软件、浏览器等常用类型。在 10ms 的采集间隔下, 总共收集了来自 27 个勒索软件家族的 1561 个样本和 5226 组样本序列, 以及 7 种类型的 345 个良性软件的 5381 组样本序列。最后, 从良性软件样本集和勒索软件样本集中随机抽取 80%数据作为训练数据, 20%作为测试数据。构建的数据集组成如表 2 所示。效果评估采用准确率(Accuracy, Acc)、精确率(Precision, P)、召回率(Recall, R)和 F1 得分(F1 Score, F1)作为标准, 以全面衡量检测系统的效能。

表 2 数据集组成

Table 2 Dataset composition

类别	勒索家族	样本数量	序列数量
良性软件	--	345	5381
勒索软件	Ceber	232	925
	Conti	167	667
	Cuba	156	625
	DemonWare	153	613
	Lockbit	83	330
	HelloXD	40	152
	Other	730	1914

4.2 对比实验分析

将 HPC 时间序列作为模型输入, 在多个流行模型中在准确率、精确率、召回率等指标下进行比较, 实验使用 Python3.7 以及 PyTorch 库实现算法学习, GPU 型号为 RTX3090 24GB, 采用以下设置进行训练: 在预训练过程中初始学习率设置为 0.015, 并且采用 Adam 优化器进行模型训练, 并将权重衰减系数设置为 0.0001, batchsize 设置为 64, 训练 epoch 为 100。在下游分类任务中, 初始学习率设置为 0.0015, 其余超参数保持一致。模型参数设置如表 3 所示, 实验结果如表 4 所示。

作为对比的方法: RDRFT[26]从可执行文件的原始字节中使用 n-gram 提取特征, 并使用随机森林模型进行检测; SDRLNP[27]则是将可执行文件前 1024 个字节转化为序列, 并使用多层 LSTM 模型进行检测; SimCLR[28]采用对比学习, 将同一类样本数据的增强视图作为 Positive Pair, 以及不同类样本数据的增强视图作为 Negative Pair; TS-TCC[29]通过分别生成强与弱两种增强视图, 来学习表征; DeepWare[30]使用软件运行期间 HPC 数据作为特征, 将其转化为图像, 并使用 CNN 模型进行分类; Rapper[31]通过对正常软件运行期间的 HPC 数据使用基于 LSTM 的自编码器进行学习, 通过重构误差实现检测。

表 3 模型参数

Table 3 Model parameters

参数	值
patch 大小	20
patch 步长	10
Transformer 隐藏层维度	384
Transformer 深度	12
Transformer 头数	6

表 4 对比实验结果

Table 4 Comparison of detection results

模型	Acc	P	Recall	F1
RDRFT	0.926	0.935	0.937	0.936
SDRLNP	0.923	0.931	0.935	0.933
SimCLR	0.942	0.945	0.940	0.942
TSTCC	0.972	0.977	0.964	0.964
DeepWare	0.937	0.943	0.939	0.941
Rapper	0.940	0.943	0.942	0.942
Proposed	0.982	0.984	0.971	0.978

实验结果表明, RDRFT 和 SDRLNP 方法均使用可执行文件进行静态特征提取, 使模型能学习恶意软件的相似特征, 但是静态特征通常较为保守, 容易产生误判, 同时也易被代码混淆技术绕过, 所获得特征不足以完全描述勒索软件; SimCLP 以及 TS-TCC 方法通过对比学习方法显示了对比学习在提高模型泛化能力和减少误判方面的优势; DeepWare 采用 CNN 作为框架, 在长时间序列分类任务中表现受限, 且仅使用了直观上的 HPC 事件数据, 并未对 HPC 特征选择更进一步分析; Rapper 对 HPC 数据使用自编码器对重构误差进行判断, 阈值的设定影响了模型的最终效果。总体而言, 提出模型的优越表现证实了其在勒索软件检测上的有效性, 同时有效的特征筛选可以提高最终检测性能。

4.3 消融实验分析

4.3.1 无监督训练任务

在本文所提出模型中, 使用了重构任务作为获得时间序列表征向量[CLS]的方法, 并使用对比学习

方法强化模型表征能力。为探讨两种任务对模型表征能力有效性的影响, 分别去除模型中的重构任务以及对比学习任务, 探究不同任务对模型效果的影响。根据表 5 的结果, 对比学习优化了数据分布, 拉近了同类样本的表征向量, 最终得到了鲁棒的特征表示, 而重构任务可以帮助整个模型深入理解整体序列, 同时使用两种任务可以使得模型获得最优效果, 由此可以说明两种任务对提高模型效果均有效。

表 5　训练任务消融实验

Table 5　Ablation study on tasks

模型	Acc	F1
Proposed	0.982	0.978
去除重构	0.932	0.874
去除对比学习	0.925	0.861

4.3.2　模型框架

Transformer 模型在长时间序列任务中表现优异, 故而在本文所提模型中, 采用了 Transformer 框架, 并在头部插入[CLS]向量获得表征向量。在时间序列表征向量提取方法中, 1D-CNN, LSTM, Bi-LSTM 是三种常见模型框架, 为了比较表征向量提取在不同框架上的表现, 通过仅更改模型骨架对上述三种模型框架进行了实验。在 1D-CNN 模型中, 经过多层卷积层以及平均池化层, 提取时序表征; LSTM 模型使用最后一个时间步的隐藏状态作为整个序列的表征; Bi-LSTM 模型通过最大池化方法, 逐元素比较正向和反向的最后状态, 取每个位置上的最大值, 作为整个序列的表征。最终实验结果如表 6 所示。

表 6　模型框架消融实验

Table 6　Ablation study on backbones

框架	Acc	F1
Transformer	0.982	0.978
1D-CNN	0.922	0.915
LSTM	0.932	0.926
Bi-LSTM	0.937	0.927

使用 Transformer 框架取得的结果最优。相比之下, 1D-CNN 模型准确率下降幅度较大, 这是因为卷积网络可以从局部时序中识别出简单模式, 然后使用这些简单模式在更高级的层中生成更加复杂的模式, 这一特点更适用于分析存在周期性较强的数据, 而对于需要捕获长期依赖关系的序列并不合适。LSTM 和 Bi-LSTM 效果优于 1D-CNN, 进一步展现了捕获时序中长期依赖关系在理解整个 HPC 序列中的必要性。Bi-LSTM 优于 LSTM 则是因为相较于单向的 LSTM 仅能感知过去及当前时间片信息, Bi-LSTM 通过双向循环结构, 能感知序列的完整上下文信息。更进一步, Transformer 通过自注意力机制能够直接捕捉序列中任意两个位置之间的依赖关系, 而 LSTM 和 Bi-LSTM 循环架构在处理长序列时, 因为梯度消失或者爆炸的问题, 难以保持有效的性能。

4.4　无监督效果分析

为了评估无监督训练对性能的影响, 本文从两个方面进行了评估: 一是通过改变无监督数据集的大小来探究其对预训练模型性能的影响; 二是通过减少有监督数据集的大小来分析预训练模型的鲁棒性。实验中, 当无监督训练数据减少到原数据的 20%、40%、60%、80%时, 有监督数据集大小保持不变, 以单独观察无监督训练的影响。另一方面, 也测试了在有监督数据量减少至相同比例时, 无监督数据集大小保持固定, 从而评估有限有监督数据对分类性能的影响。

实验结果如图 8 所示, 随着无监督数据量的增加, 模型性能明显提升, 突出了其在学习大规模未标记数据的深层特征方面的潜力。在有监督数据较少的微调阶段, 尽管模型的分类性能有所下降, 但预训练部分的影响相对较小。结果表明, 无监督训练显著提升了模型在处理未知样本的鲁棒性, 并能在有监督数据有限的情况下保持较高的准确率, 证实了其在勒索软件检测任务中的有效性。

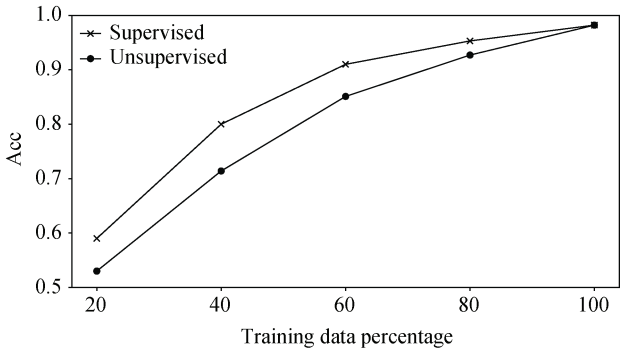

图 8　无监督效果分析

Figure 8　Unsupervised effect analysis

4.5　泛化能力及鲁棒性分析

为进一步验证模型的泛化能力, 本节测试了未用于模型训练的新勒索软件家族, 包括 CoronaVirus、Polyransom、GlobeImposter、Dharma、Hydracrypt、LooCipher、MegaCortex 和 Sodinokibi。这些家族的样本总数为 1831 条。实验结果如图 9 所示, 本文的方法在检测这些未见勒索软件类别时实现了 93.8%的准确率, 明显优于其他模型。此外, 包含无监督训

练的模型在识别未知勒索软件方面展现了卓越性能。无监督模型通过学习大量非特定任务的数据, 已形成丰富的特征表示, 从而在遇到未知的勒索软件时展示出强大的泛化能力。这些结果突显了无监督训练在增强模型对新型未知威胁识别能力方面的重要性, 证实了其在应对未曾遇到的勒索软件家族时保持高效检测性能的能力。

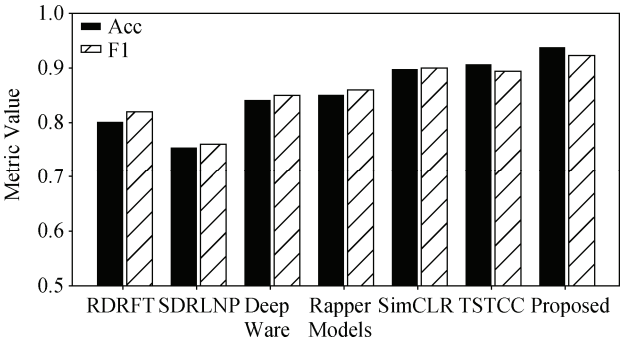

图 9 未知勒索软件检测结果

Figure 9 Unknown ransomware detection results

为了评估本文所提出方法在处理实际环境数据时的鲁棒性, 通过在测试数据中加入高斯噪声来模拟现实世界中可能遇到的各种干扰。对于高斯噪声, 本文以均值为 0, 标准差分别为数据平均值的 1%, 5%和 10%对原始数据进行扰动, 以评估模型对于轻微到中等程度噪声的敏感性, 实验结果如表 7 所示。

表 7 鲁棒性分析结果

Table 7 Robustness analysis results

扰动方法	参数	Acc	F1
高斯噪声	1%	0.979	0.973
	5%	0.965	0.959
	10%	0.934	0.932

从结果中可以看出, 随着高斯噪声水平的增加, 模型的效果有下降趋势。尽管噪声水平增加, 但模型性能的下降幅度仍然保持在一个相对较小的范围内。这说明该模型在面对不同程度的高斯噪声时表现出了良好的鲁棒性。

4.6 检测窗口大小分析

在本节中, 探讨了 10s 及时检测窗口大小的有效性以及检测窗口大小对于识别勒索软件行为的影响, 并展示了实验结果, 以证明本文提出的检测模型在及时识别勒索软件攻击方面的有效性。

为验证及时检测窗口大小的有效性, 扩展了实证分析范围, 考虑了在不同 CPU 性能、RAM 容量下的预加密阶段, 多样化评估硬件差异对勒索软件预加密行为的潜在影响。

预加密耗时如表 8 所示, 在高性能 CPU 和大容量 RAM 的环境下, 勒索软件的预加密过程因更优的数据处理能力和内存空间而加速。在预加密阶段, 诸如环境检测、权限提升、持久性确保等行为会因为硬件性能而进一步加快。因此, 对能迅速开始加密的勒索软件家族而言, 缩短检测窗口至关重要。

表 8 预加密阶段平均耗时(s)

Table 8 Pre-encryption average time(s)

硬件性能	4GB	8GB	16GB
2.5GHz	18.2	15.3	12.8
3.2GHz	14.0	11.7	9.9

为了精确评估模型在不同检测窗口大小的影响, 本研究采用了一种灵活的方法, 不同于传统的为每种数据长度训练独立模型的做法。通过在无监督训练过程中利用时间戳特征预测数据长度, 本方法允许使用单一模型处理不同长度的数据, 避免了训练多个模型的需求。

方法如图 10 所示, 通过将连续数据按每秒等长切割成多个子段, 对于长度为 i 秒的检测窗口, 选取前 i 个数据段进行分析, 对后续数据采用零填充处理, 以生成符合模型输入需求的不同检测窗口数据集。然后, 利用基于模型重构任务生成的时间戳特征, 可以对未知数据进行补全, 形成预测的完整数据。这种方法不仅能模拟检测窗口大小对模型性能的影响, 而且有助于在实际应用中在尽可能小的检测窗口内提高对勒索软件活动的检测效率和准确性。

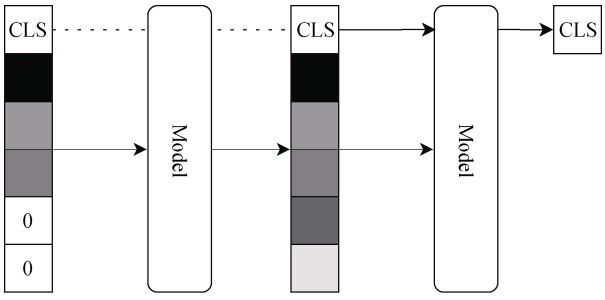

图 10 检测窗口大小分析流程

Figure 10 Analysis process of detection window sizes

实验结果显示, 在较小的检测窗口下, 模型的检测率较低。然而, 随着窗口大小的增加, 准确率也显著提高。实验数据表明, 在软件运行后的前 4 秒内, 模型的准确率可达 93.3%; 当窗口增至 7 秒时, 准确率提高至 97.8%, 在 10 秒时进一步达到 98.2%。这些结果验证了模型在早期有效检测勒索软件方面的优

越性能, 并展示了通过利用时间戳级嵌入, 模型能在保证高准确率的同时缩短所需的检测时间。

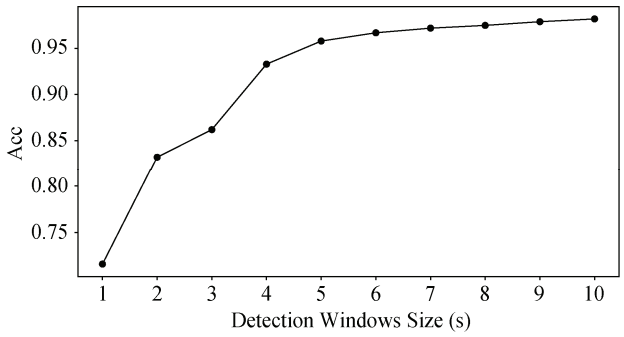

图 11　不同大小检测窗口大小实验结果
Figure 11　Detection results with different window sizes

4.7　特征选择分析

在本项研究中, 采用了 Boruta 技术用以从每个10秒的时间窗口收集的HPC数据中确定最关键的特征。本测试基于 Boruta 算法选出的前 N 个最重要的特征分别构建了数据集, 并以本文所提出模型作为基准模型进行评测, 结果如表 9 所示。

表 9　特征选择分析
Table 9　Feature selection analysis

特征数量	Acc	F1
Top5	0.968	0.970
Top8	0.980	0.970
Top10	0.982	0.978
Top13	0.983	0.977
Top15	0.981	0.974
ALL	0.865	0.843

实验结果表明, 随着特征数量从 Top5 逐渐增加到Top10, 准确率和F1分数都有所提高, 但随着特征数量进一步增加, 模型效果提升较小, 甚至轻微下降。这归功于 Transformer 结合 patch 方法在处理多维时间序列时仍旧能够有效捕获时间序列之间潜在的关系, 但随着特征数量的增加, 其余特征对于勒索软件检测性能提升效果降低, 并且可能存在对少数类过拟合风险。而在使用经过特征筛选后所有的有效特征时, 因为加入更多有效性较小的特征, 反而使得模型难以学到更为有效的泛化信息, 进一步导致了分类效果变差。

5　总结与展望

本研究采用了无监督结合有监督的学习方法, 提出了一种基于硬件性能计数器的勒索软件检测技术。通过分析勒索软件行为, 可以发现在软件运行的早期阶段, 勒索软件和良性软件之间存在显著的差异。虽然 HPC 事件特征能有效检测勒索软件, 但却仍然存在敏感性以及复杂性问题。因此, 本研究运用 Boruta 技术筛选出最有效的特征, 以降低数据的复杂度。同时, 使用对比学习方法缩小同一样本中分类特征的差距, 解决由敏感性引起的 HPC 数据不一致问题。此外, 本文通过 Transformer 架构感知序列整体的能力, 进一步提高了模型检测性能。在未知数据集和已知数据集上的综合实验表明, 本文检测方法比其他模型具有更优的性能。目前, 本研究主要聚焦于勒索软件的检测, 未来将采集更多样本数据, 进一步将研究范围扩展到恶意软件分类的多分类问题。

参考文献

[1] O'Kane P, Sezer S, Carlin D. Evolution of Ransomware[J]. IET Networks, 2018, 7(5): 321-327.

[2] Scrutiny of LockBit3.0 ransomware gang intensifies. https://dailybrief.oxan.com/Analysis/ES275389/Scrutiny-of-LockBit30-ransomware-gang-intensifies. Jan. 2023.

[3] Morato D, Berrueta E, Magaña E, et al. Ransomware Early Detection by the Analysis of File Sharing Traffic[J]. Journal of Network and Computer Applications, 2018, 124: 14-32.

[4] Kok S, Abdullah A, Jhanjhi N, et al. Ransomware, threat and detection techniques: A review[J]. International Journal of Computer Science and Network Security, 2019, 19(2): 136-146.

[5] Das S, Werner J, Antonakakis M, et al. SoK: The Challenges, Pitfalls, and Perils of Using Hardware Performance Counters for Security[C]. 2019 IEEE Symposium on Security and Privacy, 2019: 20-38.

[6] Detecting process hijacking and software supply chain attacks using intel® threat detection technology. https://www.intel.com.tr/content/dam/www/central-libraries/us/en/documents/white-paper-inteltdt-abd.pdf. Feb. 2023.

[7] Zhou B Y, Gupta A, Jahanshahi R, et al. Hardware Performance Counters Can Detect Malware: Myth or Fact? [C]. The 2018 on Asia Conference on Computer and Communications Security, 2018: 457-468.

[8] Li C M, Gaudiot J L. Detecting Malicious Attacks Exploiting Hardware Vulnerabilities Using Performance Counters[C]. 2019 IEEE 43rd Annual Computer Software and Applications Conference, 2019: 588-597.

[9] Malone C, Zahran M, Karri R. Are Hardware Performance Counters a Cost Effective Way for Integrity Checking of Programs[C]. The Sixth ACM Workshop on Scalable Trusted Computing, 2011: 71-76.

[10] He Z Y, Miari T, Makrani H M, et al. When Machine Learning Meets Hardware Cybersecurity: Delving into Accurate Zero-Day Malware Detection[C]. 2021 22nd International Symposium on Quality Electronic Design, 2021: 85-90.

[11] Demme J, Maycock M, Schmitz J, et al. On the Feasibility of Online Malware Detection with Performance Counters[J]. ACM SIGARCH Computer Architecture News, 2013, 41(3): 559-570.

[12] Patel N, Sasan A, Homayoun H. Analyzing Hardware Based Malware Detectors[C]. 2017 54th ACM/EDAC/IEEE Design Automation Conference, 2017: 1-6.

[13] Niu W N, Zhao C Y, Zhang X S, et al. ROPDetector: A Real-Time Detection Method of ROP Attack Based on Hardware Performance Counter[J]. Chinese Journal of Computers, 2021, 44(4): 761-772.
(牛伟纳, 赵成洋, 张小松, 等. ROPDetector: 一种基于硬件性能计数器的 ROP 攻击实时检测方法[J]. 计算机学报, 2021, 44(4): 761-772.)

[14] Yao Z H, Li Y M, Ma Z Q, et al. Multi-Object Cache Side-Channel Attack Detection Model Based on Machine Learning[J]. Journal of Computer Applications, 2024, 44(6): 1862-1871.
(姚梓豪, 栗远明, 马自强, 等. 基于机器学习的多目标缓存侧信道攻击检测模型[J]. 计算机应用, 2024, 44(6): 1862-1871.)

[15] Tang A, Sethumadhavan S, Stolfo S J. Unsupervised Anomaly-Based Malware Detection Using Hardware Features[C]. Research in Attacks, Intrusions and Defenses, 2014: 109-129.

[16] Garcia-Serrano A. Anomaly Detection for Malware Identification Using Hardware Performance Counters[EB/OL]. 2015: arXiv: 1508.07482. https://arxiv.org/abs/1508.07482.

[17] Carnà S, Ferracci S, Quaglia F, et al. Fight Hardware with Hardware: Systemwide Detection and Mitigation of Side-Channel Attacks Using Performance Counters[J]. Digital Threats: Research and Practice, 2023, 4(1): 1-24.

[18] Li S Y, Jin X Y, Xuan Y, et al. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting[EB/OL]. 2019: arXiv: 1907.00235. https://arxiv.org/abs/1907.00235.

[19] Wu H X, Xu J H, Wang J M, et al. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting[C]. Neural Information Processing Systems, 2021.

[20] Nie Y Q, Nguyen N H, Sinthong P, et al. A Time Series Is Worth 64 Words: Long-Term Forecasting with Transformers[EB/OL]. 2022: arXiv: 2211.14730. https://arxiv.org/abs/2211.14730.

[21] Saeed A, Ozcelebi T, Lukkien J. Multi-Task Self-Supervised Learning for Human Activity Detection[J]. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2019, 3(2): 1-30.

[22] van den Oord A, Li Y Z, Vinyals O. Representation Learning with Contrastive Predictive Coding[EB/OL]. 2018: arXiv: 1807.03748. https://arxiv.org/abs/1807.03748.

[23] Sarkar P, Etemad A. Self-Supervised ECG Representation Learning for Emotion Recognition[J]. IEEE Transactions on Affective Computing, 2022, 13(3): 1541-1554.

[24] Guide P. Intel® 64 and ia-32 architectures software developer's manual[J]. System programming Guide, 2011, 2(11): 1-64.

[25] Kursa M B, Rudnicki W R. Feature Selection with theBoruta-Package[J]. Journal of Statistical Software, 2010, 36(11): 1-13.

[26] Khammas B M. Ransomware Detection Using Random Forest Technique[J]. ICT Express, 2020, 6(4): 325-331.

[27] Manavi F, Hamzeh A. Static Detection of Ransomware Using LSTM Network and PE Header[C]. 2021 26th International Computer Conference, Computer Society of Iran, 2021: 1-5.

[28] Chen T, Kornblith S, Norouzi M, et al. A Simple Framework for Contrastive Learning of Visual Representations[EB/OL]. 2020: arXiv: 2002.05709. https://arxiv.org/abs/2002.05709.

[29] Eldele E, Ragab M, Chen Z H, et al. Time-Series Representation Learning via Temporal and Contextual Contrasting[EB/OL]. 2021: arXiv: 2106.14112. https://arxiv.org/abs/2106.14112.

[30] Ganfure G O, Wu C F, Chang Y H, et al. DeepWare: Imaging Performance Counters with Deep Learning to Detect Ransomware[J]. IEEE Transactions on Computers, 2023, 72(3): 600-613.

[31] Alam M, Sinha S, Bhattacharya S, et al. RAPPER: Ransomware Prevention via Performance Counters[EB/OL]. 2020: arXiv: 2004.01712. https://arxiv.org/abs/2004.01712.

袁军翼 于 2021 年在苏州大学信息与计算科学专业获得学士学位。现在南京理工大学软件工程专业攻读硕士学位。研究领域为网络与信息安全。研究兴趣包括恶意软件检测、计算机系统结构等。Email: njustyjy@njust.edu.cn

杨志鹏 于 2021 年在南京理工大学软件工程专业获得学士学位。现在南京理工大学电子信息专业攻读硕士学位。研究领域为网络态势感知。研究兴趣包括分布式计算、恶意软件检测。Email: yangzhipeng@njust.edu.cn

刘代东 于 2021 年在南京理工大学计算机科学与技术专业获得学士学位。现在南京理工大学电子信息专业攻读硕士学位。研究领域为区块链技术。研究兴趣包括恶意软件检测、区块链应用等。Email: 121127223771@njust.edu.cn

魏松杰 于 2009 年在美国特拉华大学计算机网络专业获得博士学位。现为南京理工大学计算机工程学院网络与通信技术系副教授。研究领域为网络与信息安全、网络流量分析与检测等。Email: swei@njust.edu.cn