Computer Science and Application 计算机科学与应用, 2025, 15(6), 35-44
Published Online June 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.156155

Hans汉斯

面向APT攻击调查的溯源图冗余结构压缩

李苍铭，徐志强

江西理工大学信息工程学院，江西 赣州

收稿日期：2025年5月6日；录用日期：2025年6月6日；发布日期：2025年6月13日

摘　要

面向系统审计日志的溯源图分析已经是APT攻击调查的主要手段。溯源图节点代表系统实体(包括进程、文件和网络)，边代表系统实体之间的依赖关系。攻击调查是在溯源图上追踪攻击源头并构建完整的攻击路径。依赖爆炸导致溯源图规模庞大，将为攻击调查带来巨大的存储开销和时间开销。为解决该问题，本文提出进程重复模式压缩和文件重复模式压缩去减小溯源图规模。其中，进程重复模式代表系统在不同时间调用相同进程执行相同的文件读写任务，而文件重复模式代表多个文件被相同进程处理。这些模式均表示重复的行为，不会带来更多有价值的信息，因此压缩它们不会影响攻击调查。本文在6个真实攻击数据集(约1948万个系统事件)进行实验验证，结果指出溯源图节点和边的压缩率平均分别为56.5%和58.0%。此外，在压缩前和压缩后的溯源图上分别执行攻击调查，结果证明本文的压缩方法不会影响攻击调查结果。

关键词

高级持续威胁，系统审计日志，攻击调查，图压缩

Provenance Graph Redundant Structure Compression for APT Attack Investigation

Cangming Li, Zhiqiang Xu

School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou Jiangxi

Received: May 6th, 2025; accepted: Jun. 6th, 2025; published: Jun. 13th, 2025

Abstract

A provenance graph analysis of system audit logs has become a primary method for investigating APT attacks. The nodes in the provenance graph represent system entities (including processes, files, and network activities), while the edges denote dependency relationships between these entities. Attack investigation involves tracing the attack origin and reconstructing the complete attack

文章引用: 李苍铭, 徐志强. 面向 APT 攻击调查的溯源图冗余结构压缩[J]. 计算机科学与应用, 2025, 15(6): 35-44.
DOI: 10.12677/csa.2025.156155

path on the provenance graph. However, dependency explosion leads to excessively large provenance graphs, imposing significant storage and computational overhead on attack investigations. To address this issue, this paper proposes process repetition pattern compression and file repetition pattern compression to reduce the scale of provenance graphs. Specifically, process repetition patterns refer to cases where the system repeatedly invokes the same process to perform identical file read/write operations at different times. File repetition patterns describe scenarios where multiple files are processed by the same process. Since these patterns represent redundant behaviors and do not provide additional valuable information, compressing them does not affect attack investigations. Experiments were conducted on six real-world attack datasets (comprising approximately 19.48 million system events). The results demonstrate an average compression rate of 56.5% for nodes and 58.0% for edges in the provenance graph. Furthermore, attack investigations (using Nodoze and DepComm) were performed on both the original and compressed provenance graphs, confirming that the proposed compression method does not compromise investigation accuracy.

Keywords

Advanced Persistent Threat, System Audit Logs, Attack Investigation, Graph Compression

1. 引言

高级持续威胁(APT)是一种多步渐进式攻击，具有攻击手段多样、隐蔽性强、持久性高等特点，可以长期潜伏在目标系统中，对系统造成敏感数据泄漏和可用性破坏等危害，已经成为企业和政府机构中主机系统的主要威胁之一。截至目前，APT 攻击已经造成 6 万亿美元的经济损失[1]。因此，有效防御 APT 攻击已经成为全球性重要课题。

攻击调查取证是 APT 防御的重要环节之一。某一攻击事件被检测(POI 事件)，针对该事件的溯源取证被启动，找出攻击源头，构建完整的攻击路径，同时分析对系统的后续影响。系统调用日志记录了系统实体(包括进程，文件和 IP 等)之间的交互行为(如读取、写入和进程创建等)。相比于应用层日志，系统调用日志记录更为全面，而且很难被篡改，是最常用的分析数据[2]。目前主要的取证方法是采用因果分析技术[2] [3]将日志数据转换为系统依赖关系图，之后以 POI 事件为起点，利用回溯算法[2]构建与该事件存在因果依赖关系的溯源图。最后采用信息流追踪[4]-[8]、异常检测[9]和机器学习[10]-[14]等技术在溯源图上进行分析，移除不相关事件呈现完整的攻击路径。

长时进程(如 bash，firefox 等)的运行通常会产生大量系统事件，导致依赖爆炸，使得溯源图规模庞大(平均 50 万事件/24 小时) [15] [16]。存储这些溯源图必然导致巨大的开销。最重要的是，在如此庞大的溯源图中分析攻击路径将会带来巨大的时间开销。因此，在不影响攻击调查的情况下减小溯源图规模十分必要。目前，已有的溯源图压缩工作是针对不影响依赖关系结构的节点或重复的平行边进行合并或移除[17]-[21]，虽然达到较高的压缩率，但并没有考虑冗余的子结构信息。这些冗余子结构代表相同的系统行为(如计划任务和配置文件更新)。由于重复的行为不能带来更多有价值的信息，因此，本文提出一套针对冗余子结构的压缩方法。

本文将冗余子结构分为两种重复模式：进程重复模式和文件重复模式。进程重复模式代表计算机系统在不同时间调用相同进程执行相同的文件读写任务(例如，linux 系统在每次执行定时任务时守护进程

cron 会调用子进程 cron 读写/proc/self/loginuid 文件)。针对进程重复模式，本文基于溯源图构建进程世系树，之后将进程执行的网络和文件信息关联到该进程节点，最后采用满子树搜索算法[22]搜索重复的子树结构，将重复结构进行合并。文件重复模式代表多个文件被相同进程处理(如 firefox 进程对多个配置文件进行更新)。针对文件重复模式，本文在压缩进程重复模式后搜索由相同进程节点连接的文件节点，之后将这些文件节点进行合并。

文本在 6 个真实场景的攻击数据集进行评估。评估数据共有 1948 万个系统事件，生成的溯源图平均有 5300 个节点和 1,909,019 条边。本文压缩方法的节点压缩率平均 56.5%，边压缩率平均 58.0%，压缩执行时间平均 0.146 秒，表现出较高的压缩效率。为了评估压缩后的溯源图对攻击调查的影响，本文采用 Nodoze [9]和 DepComm [11]攻击调查方法在原始溯源图和压缩后溯源图分别进行分析，检测出的攻击路径均达到 100%的相似性，而且在压缩后溯源图上的调查时间分别平均降低了 73.9%和 64.1%。因此，本文的压缩方法不仅没有影响攻击调查结果，而且降低了调查时间。

2. 相关工作

2.1. APT 攻击调查

为找出 APT 攻击源头和构建完整的攻击路径，针对系统审计日志的攻击溯源图被 King 等[2]提出。依赖爆炸问题导致构建的溯源图规模庞大，含有大量不相关的系统事件。基于节点分割技术的细粒度溯源分析方法被提出[15] [16] [23]-[27]，能够缩小溯源图规模。由于这些方法需要对系统内核和程序源代码进行改写，具有较差的适用性。为此，一些工作采用信息流追踪技术[4]-[8]、异常检测技术[9]和机器学习算法[10]-[14]在溯源图中直接搜索相关的攻击事件和滤除不相关事件。本文提出的溯源图压缩算法，可以为这些攻击调查方法提供更简洁的溯源图，提高分析效率。

2.2. 溯源图压缩

为减少溯源图分析带来的时间和存储开销，一些工作提出了有效的压缩方法。Lee 等[17]识别和移除只被一个进程访问的临时文件，如此的文件不能对其它进程产生影响，被认为是冗余的。Xu 等[18]通过合并系统实体之间重复的依赖关系，以达到节省系统依赖图的存储开销。Hossain 等[19]利用全局图信息对溯源图节点间重复的边进行合并。Tang 等[20]对不影响依赖关系分析的系统链接库，资源和配置等只读文件节点进行移除。Fei 等[21]在以上工作的基础上，同时考虑对溯源图节点和边进行压缩，此外考虑与 write 和 execute 事件相关的系统实体节点的压缩。尽管这些方法达到了很高的压缩效果，但是并没有考虑冗余子结构的压缩。本文基于上述压缩结果，对冗余子结构进行压缩，可以进一步减少溯源图规模。

3. 背景与动机

3.1. 系统审计日志和溯源图

由 Linux Audit [28]，Sysdig [29]和 Windows ETW [30]等监控系统生成的系统审计日志记录了系统实体(包括进程、文件和网络)之间的交互行为，即系统事件。根据文献[2] [3]的定义，系统事件表示成 3 元组：<主体，操作，客体>，代表主体对客体执行某一操作(如进程 cp 写入文件 list.txt)，其中主体只包括进程。而根据客体类型不同系统事件可以进一步分为进程事件(客体 = 进程)、文件事件(客体 = 文件)和网络事件(客体 = 网络)。

基于系统审计日志的溯源分析已经成为 APT 攻击调查的主要方法。首先采用因果分析技术[2] [3]推断系统事件之间的因果关系，构建系统依赖图 $G(E,V)$，其中节点 V 代表系统实体(即进程、文件和网络)，有向边代表系统事件 $e(u,v)$，其中 $u \in V$，$v \in V$，$e \in E$，边的方向代表数据流方向(即 $u \to v$)。在没有特

殊说明的情况下，本文的溯源图中的进程节点用矩形表示，文件节点用椭圆表示，网络节点用平行四边形表示。此外，一个事件记录了开始时间 $e.st$ 和结束时间 $e.et$ 。给定两个事件 $e_1(u_1,v_1)$ 和 $e_2(u_2,v_2)$，如果 $v_1=u_2$ 并且 $e_1.st \le e_2.et$ 则 e_1 和 e_2 存在因果依赖关系。给定一个告警事件(即 POI 事件)，以该事件为起点，沿着边的反方向迭代遍历与 POI 事件存在因果依赖关系的早期事件，将这些事件构建溯源图。溯源图完整描绘了生成 POI 事件的路径。

3.2. 动机例子

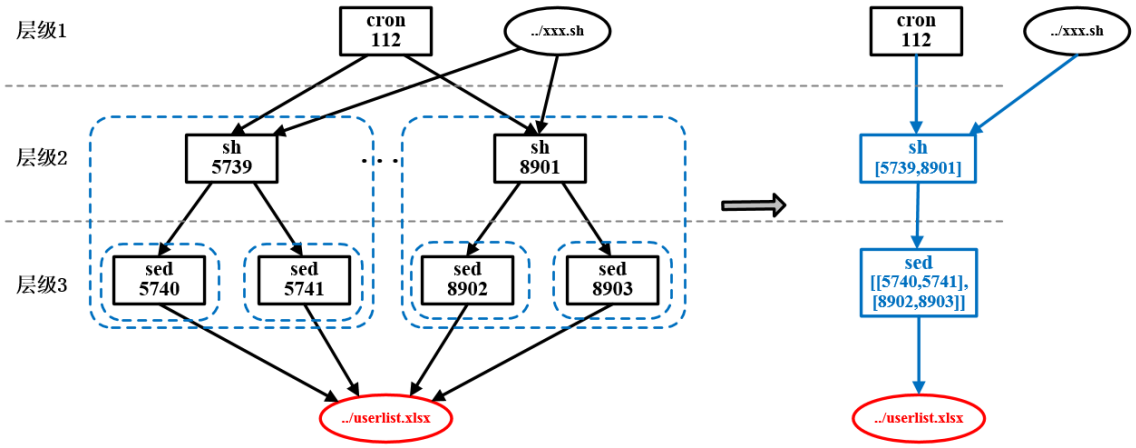

Figure 1. Example of process repetitive pattern compression. (a) Uncompressed, (b) Compressed
图 1. 进程重复模式压缩例子。(a) 压缩前，(b) 压缩后

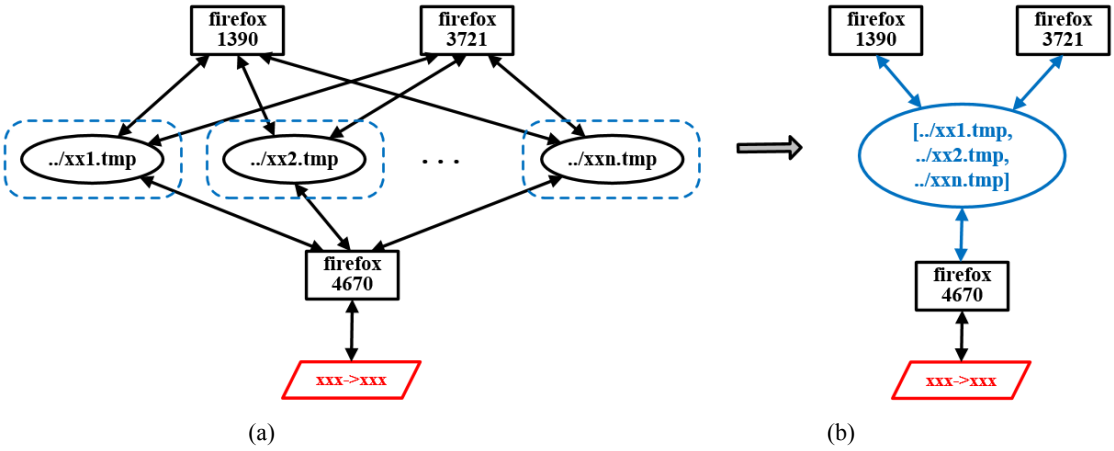

Figure 2. Example of file repetitive pattern compression. (a) Uncompressed, (b) Compressed
图 2. 文件重复模式压缩例子。(a) 压缩前，(b) 压缩后

进程重复模式压缩例子：本文以攻击者频繁改写目标文件为例阐述进程重复模式压缩的动机。具体的说，攻击者利用 Linux 计划服务定期执行恶意脚本 xxx.sh 对目标文件../userlist.xlsx 进行多次恶意篡改。该恶意行为被完整记录到系统审计日志中。本文将改写../userlist.xlsx 事件作为 POI 事件，通过溯源分析构建溯源图。图 2 展示了溯源图的一部分。从图中可以看到计划服务进程 cron 多次调用进程 sh，进程 sh 又多次调用进程 sed 对同一目标文件../usrlist.xlsx 执行写操作。这些结构(虚线框所示)均表现出重复的进程行为，不会带来更多有价值的信息。本文将这些重复进程模式进行压缩，得到图 2(b)，其中蓝色部分

为压缩后的节点和边。节点从原来的 6 个减少到 2 个，边从原来的 12 条减少到 4 条，极大减小了图的规模。此外为了防止事件信息的丢失，本文将原事件的信息(包括进程号，开始时间和结束时间)整合到压缩后的节点和边中。例如，压缩后的进程节点 sh 的进程号由原进程号 5739 和 8901 组成。

文件重复模式压缩例子：图 2(a)为溯源图中文件重复模式例子。firefox 应用程序的多个临时文件(虚线框所示)同时被不同时间的 firefox 进程改写。这些文件的读写被看作是平行行为，压缩它们不会影响溯源图结构。压缩后的溯源图如图 2(b)所示，其中蓝色部分为压缩后的节点和边，节点由原来的 3 个变为 1 个，边由原来的 9 条变为 3 条，极大减小了图的规模。此外，本文将原文件事件信息(包括文件目录、文件名、开始时间和结束时间)保留并整合到压缩后的节点和边中。

4. 本文方法

4.1. 溯源图生成与预处理

Table 1. System entity and event attribute information
表 1. 系统实体和事件属性信息

	属性
进程	PID，进程名
文件	目录，文件名
网络	IP，Port
事件	开始时间，结束时间，操作类型

溯源图生成：本文采用 Sysdig [29]系统监控工具在主流操作系统(如 Windows 和 Linux)上采集系统审计日志，包括进程事件、文件事件和网络事件。其中每个采集的实体和事件记录了重要的属性信息，如表 1 所示。利用因果分析技术对采集的日志进行依赖分析，构建依赖关系图，其中每一个节点被赋予唯一的 ID 号。之后，基于 POI 事件，通过 backtrack 算法[2]反向迭代追踪与 POI 事件存在因果依赖关系的早期事件，将这些事件构成该 POI 事件的溯源图。

预处理：本文的预处理涵盖了已有的溯源图压缩工作，包括平行边合并、只读文件节点移除和临时文件节点移除。Xu 等[18]指出进程节点和文件/网络节点之间存在多条平行边，这些平行边代表重复的读/写操作，并没有提供更多有价值的信息。因此，本文直接将具有相同操作类型的平行边合并为一条。Tang 等[20]指出只读文件通常为用于系统初始化的配置文件和库文件。由于这些只读文件没有被写入，所以不含有攻击注入的信息。本文将溯源图中不含入射边的文件节点从溯源图中移除。Lee 等[17]指出一个文件如果只被一个进程读取和写入，则该文件被看作是不可达实体，将它移除不会影响溯源分析。本文将只有一个邻居进程节点的文件节点从溯源图中移除。

4.2. 进程重复模式压缩

进程重复模式代表一个进程多次创建相同子进程，由这些子进程对相同文件执行相同操作。本文是在预处理后的溯源图上执行如此模式的识别，之后合并其中相同的节点和边。该过程包括两个阶段：重复模式识别和重复模式压缩。

重复模式识别：这个过程包括如下步骤：

步骤 1：建立进程世系树。在预处理后的溯源图上，本文通过遍历进程节点构建进程世系树。具体来说，将没有父进程的进程节点作为根，以根为起点沿着进程事件之间的依赖关系向前遍历溯源图，之后将得到的进程节点添加到树中。进程世系树呈现了进程之间的创建关系。

步骤 2：划分层级。本文对构建的进程世系树从根节点开始划分层级。具体地，所有根节点为层级 1，根节点的所有子节点为层级 2，该子节点的所有子节点为层级 3，依次类推，直到树的最底层为层级 N。如图 1 所示，进程 cron 作为根节点为层级 1，其两个子进程 sh 为层级 2，sh 的子进程 sed 为层级 3，sed 没有子进程，因此最大层级数为 3。

步骤 3：关联访问的文件和网络。为捕获进程世系树中进程与文件和网络的交互行为，本文将进程在溯源图中直接连接的文件节点、网络节点和边属性信息关联到进程世系树节点中。具体为，将进程名、父进程名和 PID、访问的文件名、访问的网络 IP、对文件和网络的操作类型作为进程世系树节点的属性信息。如图 1(a)所示，进程节点 sh (5739)关联的属性包括：sh、cron (112)、xxx.sh、read。

步骤 4：提取子树。进程重复模式中的每一次进程调用行为在进程世系树中是一颗满子树[22]。满子树包括一个节点和它的所有后代。具体为，以层级 n (n ≥ 2)的进程作为子树的根节点，采用自上向下层次遍历算法遍历所有后代，得到一颗满子树。

步骤 5：挖掘重复模式。进程重复模式是一组完全相同的满子树，包括相同的结构和节点属性。首先，对提取的满子树进行前续遍历。之后，根据节点访问次序和关联的节点属性，将子树编码为字符串，并标识相同的字符串，即重复的子树。最后转到下一层级，即 n = n + 1，返回步骤 4 继续搜索，直到最高层级(即 n = N)。

重复模式压缩：这里将重复子树中相同位置的节点和边进行合并。为了保留原节点和边属性信息，本文将其整合到压缩后的节点和边中。对于节点属性，压缩后的 PID 是原节点 PID 的并集。对于边属性，压缩后的起始时间为原边起始时间的最小值，结束时间为原边结束时间的最大值。

4.3. 文件重复模式压缩

文件重复模式代表多个文件被相同进程处理。本文首先从溯源图中提取每一个文件节点的一阶邻居子图。例如，图 2(a)中文件../xx1.tmp 的一阶邻居子图由进程 firefox (1390)、firefox (3721)和 firefox (4670)及相连的边构成。最后，识别具有相同邻居和边操作类型的子图，将这些子图的中心文件节点和边合并。为了防止原信息丢失，本文将原文件节点和边属性整合到压缩后的节点和边中。具体地，压缩后节点的文件名是原节点文件名的并集，边的起始时间为原边起始时间的最小值，边的结束时间为原边结束时间的最大值。

5. 实验评估

本文在一台具有两颗 Intel Xeon Gold 6226R 2.9 GHz CPU 和 256 GB 内存的服务器上评估压缩方法。评估数据集是在实验环境中使用系统监控工具采集的 6 种攻击数据。这里主要评估所提方法的压缩效果，压缩后的溯源图对攻击调查的影响及压缩算法的运行时间。

5.1. 数据集

本文在 6 台 Linux 主机上部署系统监控工具 Sysdig [29]去采集系统审计数据。在监控期间，本文基于已知的系统漏洞和 Kill Chain 框架[31]在这些主机上执行 6 种 APT 攻击，具体的攻击行为信息如下：

攻击 email 服务器(A1)：某公司内部的一名员工将恶意代码段写入到正常软件中，并将其上传到公司内部的资源网站。之后，该软件被一名员工下载并执行，此时恶意代码段将创建与攻击者远程服务器之间的后门通道，并监控发送的电子邮件内容。

破坏编译系统(A2)：攻击者将恶意 C 源代码上传到内部资源网站。当受害者下载并编译恶意源代码时，编译过程将改写编译器配置文件，导致编译系统崩溃。

篡改敏感数据(A3)：攻击者在一天时间里多次使用窃取的密码登录目标主机，然后收集并篡改敏感数据。最后，攻击者通过电子邮件传回这些敏感数据

泄露敏感文件(A4)：攻击者利用 Shellshock 漏洞[32]侵入目标主机，并建立后门通道。之后，攻击者通过后门发送恶意软件并执行，该恶意软件能够扫描并传回敏感文件。

破解密码(A5)：攻击者通过后门向目标主机发送密码破解器 payload 并执行获取 root 密码。然后，攻击者利用 root 权限渗透同一网段内的其它主机。

VPN Filter(A6)：VPN Filter [33]是一种执行 APT 攻击的模块化软件。攻击者使用 VPN Filter 渗透目标主机，建立后面通道，窃取敏感数据。

这些被监视的主机拥有 10 个活跃用户，执行日常的系统任务(如，网页浏览、文本编辑和数据分析等)。在一天内，Sysdig 收集的日志事件共有 1,948 万个。本实验在采集的日志数据上执行因果分析，并构建依赖关系图，之后针对上述 6 个攻击行为在依赖关系图上分别执行溯源分析，并构建溯源图，其节点数和边数在表 2 中第 2 列所示。

5.2. 压缩效果评估

Table 2. Compression results of 6 attack provenance graphs
表 2. 6 个攻击溯源图的压缩结果

攻击场景	溯源图		预处理		**进程重复模式压缩**		**文件重复模式压缩**	
	节点数	边数	节点数	边数	**节点数**	**边数**	**节点数**	**边数**
A1	527	89,020	167	414	**158**	**402**	**103**	**260**
A2	461	20,505	223	573	**174**	**502**	**154**	**326**
A3	6584	10,726,103	505	1291	**381**	**1017**	**258**	**709**
A4	1105	111,759	128	362	**104**	**316**	**99**	**259**
A5	173	22,675	38	69	**32**	**62**	**31**	**59**
A6	22,953	484,050	595	1549	**79**	**185**	**75**	**175**
平均	5300	1,909,019	276	710	**155**	**414**	**120**	**298**

本文所提的进程重复模式和文件重复模式压缩是在预处理后的溯源图上执行的。预处理包括已有的压缩过程：合并平行边[18]、移除只读文件节点[20]、移除不可达文件节点[17]。本节实验是验证所提压缩方法能否在已有的压缩工作基础上进一步减小溯源图规模。表 2 给出了在 6 个攻击溯源图上的压缩结果。针对进程重复模式，压缩后的节点数和边数相比压缩前(预处理后)分别平均减少了 121 个和 296 条，平均压缩率为 43.84%和 41.69%，最大压缩率为 86.72%和 88.06% (A6)。文件重复模式压缩是在进程重复模式压缩后进行，其压缩后的节点数和边数相比压缩前(进程重复模式压缩后)分别平均减少了 35 个和 116 条，平均压缩率为 22.59%和 28.02%，最大压缩率为 32.28%和 30.29%。由此可见，进程重复模式压缩和文件重复模式压缩在已有压缩工作基础上再一次减小了溯源图规模。

5.3. 攻击调查影响评估

为了评估本文压缩算法是否会影响后续的攻击调查，本实验选择两个已有的攻击调查方法 Nodoze [9]和 DepComm [11]分别在压缩前和压缩后的溯源图上执行攻击调查，并比较检测出攻击事件的相似性。Nodoze 首先维护一个良性事件库，该库记录了良性事件频率信息，之后根据库中的良性事件频率计算溯源图中每个路径的异常分数，具有较高异常分数的路径为攻击路径。DepComm 将溯源图根据行为任务阶段划分社区，

之后根据关键特征(如事件唯一性、时间跨度等)提取能代表社区的重要行为路径，该路径已包含攻击事件。表 3 指出 Nodoze 和 DepComm 方法在压缩前和压缩后的 6 个溯源图上调查出的攻击事件的相似性均为 100%。由此可见，本文提出的压缩方法不会影响后续的攻击调查，归因于被压缩的结构是重复的不会带来更多有价值的信息，而且重复结构中的重要信息如进程号、文件名、时间等被保留在压缩后的节点和边中。

Table 3. Similarity of attack events investigated on uncompressed and compressed provenance graphs
表 3. 在压缩前和压缩后溯源图上调查出攻击事件的相似性结果

攻击调查方法	A1	A2	A3	A4	A5	A6
Nodoze	100%	100%	100%	100%	100%	100%
DepComm	100%	100%	100%	100%	100%	100%

为了评估压缩后的溯源图是否会减少攻击调查时间，本实验统计了 Nodoze 和 DepComm 方法分别在压缩前和压缩后溯源图上的运行时间(图 3 所示)。Nodoze 方法在 6 个压缩前溯源图上的平均运行时间为 4.190 秒，压缩后的平均运行时间为 1.092，平均减少了 73.9%。DepComm 方法在 6 个压缩前溯源图上的平均运行时间为 373.535 秒，压缩后的平均运行时间为 134.112，平均减少了 64.1%。因此，本文提出的压缩方法能够有效减小攻击调查时间。

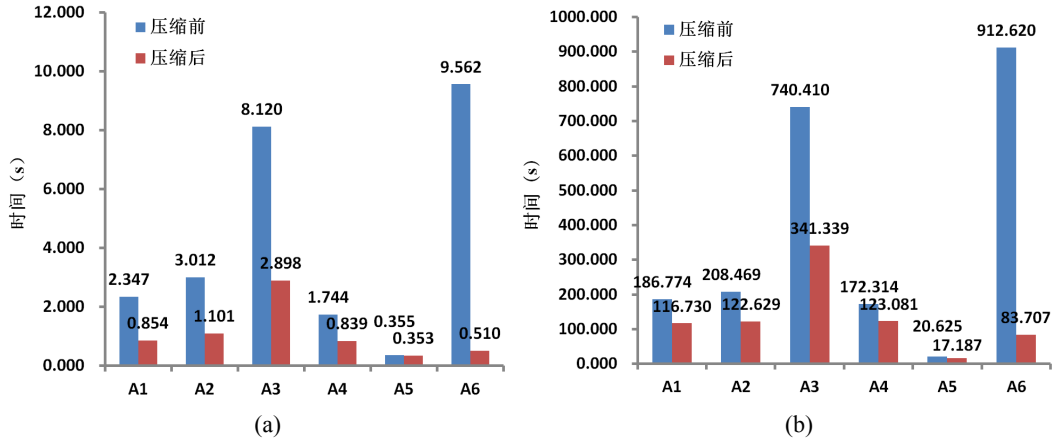

Figure 3. Runtime of attack investigation execution on uncompressed and compressed provenance graphs. (a) Nodoze, (b) DepComm
图 3. 在压缩前和压缩后溯源图上执行攻击调查的运行时间对比。(a) Nodoze，(b) DepComm

5.4. 运行时间评估

Table 4. Runtime
表 4. 运行时间

攻击场景	溯源图	预处理	**进程重复模式压缩**	**文件重复模式压缩**
A1	60.890	17.523	0.036	0.034
A2	17.585	3.866	0.053	0.041
A3	614.540	2946.905	0.155	0.105
A4	255.498	21.717	0.034	0.026
A5	228.755	4.644	0.006	0.005
A6	4692.534	97.983	0.357	0.022
平均	978.300	515.440	0.107	0.039

为评估运行时间，本实验针对 6 个攻击溯源图统计了每个阶段的运行时间(如表 4 所示)。具体来说，溯源图构建平均用时 978.300 秒，预处理平均用时 515.440 秒，进程重复模式压缩平均用时 0.107 秒，文件重复模式压缩平均用时 0.039 秒。由此可见，本文提出的压缩方法总用时(重复模式压缩时间 + 文件重复模式压缩时间)不超过 0.2 秒，用时较少，不会占用更多攻击调查时间。

6. 总结与展望

本文发现从系统审计日志构建的溯源图存在大量重复的冗余结构，这些冗余结构被分为进程重复模式和文件重复模式，并提出有效的压缩方法。本文通过进程世系树构建，文件和网络信息关联，重复满子树遍历完成进程重复模式压缩。本文通过一阶邻居相似性搜索实现文件重复模式压缩。最后，本文通过实验在真实数据集上进行压缩效果评估、攻击调查影响评估和运行时间评估，证明所提压缩方法的有效性。

基金项目

国家自然科学基金(62362038)，博士启动基金(205200100654)。

参考文献

[1] Steve, M. (2020) Cybersecurity Ventures Official Annual Cybercrime Report. https://cybersecurityventures.com/annual-cybercrime-report-2020/

[2] King, S.T. and Chen, P.M. (2003) Backtracking Intrusions. *Proceedings of the* 19*th ACM Symposium on Operating Systems Principles*, The Sagamore, 19-22 October 2003, 223-236. https://doi.org/10.1145/945445.945467

[3] King, S.T., Mao, Z.M., Lucchetti, D.G., *et al*. (2005) Enriching Intrusion Alerts through Multi-Host Causality. *Proceedings of the Annual Network and Distributed System Security Symposium* (*NDSS*), San Diego, 20 January 2005, 1-12.

[4] Ji, Y., Lee, S., Downing, E., Wang, W., Fazzini, M., Kim, T., *et al*. (2017) RAIN: Refinable Attack Investigation with On-Demand Inter-Process Information Flow Tracking. *Proceedings of the* 2017 *ACM SIGSAC Conference on Computer and Communications Security*, Dallas, 30 October-3 November 2017, 377-390. https://doi.org/10.1145/3133956.3134045

[5] Ji, Y., Lee, S., Fazzini, M., *et al*. (2018) Enabling Refinable Cross-Host Attack Investigation with Efficient Data Flow Tagging and Tracking. 27*th USENIX Security Symposium* (*USENIX Security*), Baltimore, 15-17 August 2018, 1705-1722.

[6] Liu, Y., Zhang, M., Li, D., Jee, K., Li, Z., Wu, Z., *et al*. (2018) Towards a Timely Causality Analysis for Enterprise Security. *Proceedings* 2018 *Network and Distributed System Security Symposium*, San Diego, 18-21 February 2018, 1-15. https://doi.org/10.14722/ndss.2018.23254

[7] Hossain, M.N., Sheikhi, S. and Sekar, R. (2020) Combating Dependence Explosion in Forensic Analysis Using Alternative Tag Propagation Semantics. 2020 *IEEE Symposium on Security and Privacy* (*SP*), San Francisco, 18-21 May 2020, 1139-1155. https://doi.org/10.1109/sp40000.2020.00064

[8] Fang, P.C., Gao, P., Liu, C.L., *et al*. (2022) Back-Propagating System Dependency Impact for Attack Investigation. 31*st USENIX Security Symposium* (*USENIX Security*), Boston, 10-12 August 2022, 1-18.

[9] Hassan, W.U., Guo, S., Li, D., Chen, Z., Jee, K., Li, Z., *et al*. (2019) Nodoze: Combatting Threat Alert Fatigue with Automated Provenance Triage. *Proceedings* 2019 *Network and Distributed System Security Symposium*, San Diego, 24-27 February 2019, 1-15. https://doi.org/10.14722/ndss.2019.23349

[10] Alsaheel, A., Nan, X.Y., Ma, S.Q., *et al*. (2021) ATLAS: A Sequence-Based Learning Approach for Attack Investigation. 30*th USENIX Security Symposium* (*USENIX Security*), Vancouver, 11-13 August 2021, 3005-3022.

[11] Xu, Z., Fang, P., Liu, C., Xiao, X., Wen, Y. and Meng, D. (2022) DEPCOMM: Graph Summarization on System Audit Logs for Attack Investigation. 2022 *IEEE Symposium on Security and Privacy* (*SP*), San Francisco, 22-26 May 2022, 70-87. https://doi.org/10.1109/sp46214.2022.9833632

[12] Yang, F., Xu, J.C., Xiong, C.L., *et al*. (2023) ProGrapher: An Anomaly Detection System Based on Provenance Graph Embedding, 32*nd USENIX Security Symposium* (*USENIX Security*), Anaheim, 9-11 August 2023, 4355-4372.

[13] Goyal, A., Wang, G. and Bates, A. (2024) R-CAID: Embedding Root Cause Analysis within Provenance-Based Intrusion Detection. 2024 *IEEE Symposium on Security and Privacy* (*SP*), San Francisco, 20-22 May 2024, 3515-3532.

https://doi.org/10.1109/sp54263.2024.00253

[14] Ur Rehman, M., Ahmadi, H. and Ul Hassan, W. (2024) FLASH: A Comprehensive Approach to Intrusion Detection via Provenance Graph Representation Learning. 2024 *IEEE Symposium on Security and Privacy* (*SP*), San Francisco, 20-22 May 2024, 3552-3570. https://doi.org/10.1109/sp54263.2024.00139

[15] Goel, A., Po, K., Farhadi, K., Li, Z. and de Lara, E. (2005) The Taser Intrusion Recovery System. *ACM SIGOPS Operating Systems Review*, **39**, 163-176. https://doi.org/10.1145/1095809.1095826

[16] Lee, K.H., Zhang, X.Y. and Xu, D.Y. (2013) High Accuracy Attack Provenance via Binary-Based Execution Partition. *Proceedings of the Annual Network and Distributed System Security Symposium* (*NDSS*), San Diego, 24-27 February 2013, 1-16.

[17] Lee, K.H., Zhang, X. and Xu, D. (2013) LogGC: Garbage Collecting Audit Log. *Proceedings of the* 2013 *ACM SIGSAC Conference on Computer & Communications Security*, Berlin, 4-8 November 2013, 1005-1016. https://doi.org/10.1145/2508859.2516731

[18] Xu, Z., Wu, Z., Li, Z., Jee, K., Rhee, J., Xiao, X., *et al*. (2016) High Fidelity Data Reduction for Big Data Security Dependency Analyses. *Proceedings of the* 2016 *ACM SIGSAC Conference on Computer and Communications Security*, Vienna, 24-28 October 2016, 504-516. https://doi.org/10.1145/2976749.2978378

[19] Hossain, M.N., Wang, J.A., Sekar, R., *et al*. (2018) Dependence-Preserving Data Compaction for Scalable Forensic Analysis. 27*th USENIX Security Symposium* (*USENIX Security*), Baltimore, 15-17 August 2018, 1723-1740.

[20] Tang, Y., Li, D., Li, Z., Zhang, M., Jee, K., Xiao, X., *et al*. (2018) NodeMerge: Template Based Efficient Data Reduction for Big-Data Causality Analysis. *Proceedings of the* 2018 *ACM SIGSAC Conference on Computer and Communications Security*, Toronto, 15-19 October 2018, 1324-1337. https://doi.org/10.1145/3243734.3243763

[21] Fei, P., Li, Z., Wang, Z.Y., *et al*. (2021) SEAL: Storage-Efficient Causality Analysis on Enterprise Logs with Query-Friendly Compression. 30*th USENIX Security Symposium* (*USENIX Security*), Vancouver, 11-13 August 2021, 2987-3004.

[22] Luccio, F., Pagli, L., Enriquez, A.M., *et al*. (2007) Bottom-Up Subtree Isomorphism for Unordered Labeled Trees. *International Journal of Pure and Applied Mathematics*, **38**, 325-343.

[23] Sitaraman, S. and Venkatesan, S. (2005) Forensic Analysis of File System Intrusions Using Improved Backtracking. 3*rd IEEE International Workshop on Information Assurance* (*IWIA*'05), College Park, 23-24 March 2005, 154-163. https://doi.org/10.1109/iwia.2005.9

[24] Ma, S.Q., Zhai, J., Wang, F., *et al*. (2017) MPI: Multiple Perspective Attack Investigation with Semantic Aware Execution Partitioning. 26*th USENIX Security Symposium* (*USENIX Security*), Vancouver, 16-18 August 2017, 1111-1128.

[25] Yang, R., Ma, S., Xu, H., Zhang, X. and Chen, Y. (2020) UISCOPE: Accurate, Instrumentation-Free, and Visible Attack Investigation for GUI Applications. *Proceedings* 2020 *Network and Distributed System Security Symposium*, San Diego, 23-26 February 2020, 1-18. https://doi.org/10.14722/ndss.2020.24329

[26] Hassan, W.U., Noureddine, M.A., Datta, P. and Bates, A. (2020) Omegalog: High-Fidelity Attack Investigation via Transparent Multi-Layer Log Analysis. *Proceedings* 2020 *Network and Distributed System Security Symposium*, San Diego, 23-26 February 2020, 1-16. https://doi.org/10.14722/ndss.2020.24270

[27] Yu, L., Ma, S., Zhang, Z., Tao, G., Zhang, X., Xu, D., *et al*. (2021) ALchemist: Fusing Application and Audit Logs for Precise Attack Provenance without Instrumentation. *Proceedings* 2021 *Network and Distributed System Security Symposium*, 21-25 February 2021, 1-18. https://doi.org/10.14722/ndss.2021.24445

[28] Grubb, S. (2020) Redhat Linux Audit. https://people.redhat.com/sgrubb/audit/

[29] Sysdig (2017). https://sysdig.com/

[30] Event Tracing for Windows (ETW) (2020). https://docs.microsoft.com/en-us/windows/win32/etw/

[31] Chen, P., Desmet, L. and Huygens, C. (2014) A Study on Advanced Persistent Threats. In: Decker, B. and Zúquete, A., Eds., *Communications and Multimedia Security*, Springer, 63-72. https://doi.org/10.1007/978-3-662-44885-4_5

[32] Shellshock (2014) CVE-2014-6271: Bash: Specially-Crafted Environment Variables Can Be Used to Inject Shell Commands. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271

[33] Vpnfilter (2018) VPNFilter: New Router Malware with Destructive Capabilities. https://symc.ly/2IPGGVE