electronics

MDPI

Article

ProvGRP: A Context-Aware Provenance Graph Reduction and Partition Approach for Facilitating Attack Investigation

Jiawei Li, Ru Zhang * and Jianyi Liu

School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China; lijw960502@bupt.edu.cn (J.L.); liujy@bupt.edu.cn (J.L.)
* Correspondence: zhangru@bupt.edu.cn

Abstract: Attack investigation is a crucial technique in proactively defending against sophisticated attacks. Its purpose is to identify attack entry points and previously unknown attack traces through comprehensive analysis of audit data. However, a major challenge arises from the vast and redundant nature of audit logs, making attack investigation difficult and prohibitively expensive. To address this challenge, various technologies have been proposed to reduce audit data, facilitating efficient analysis. However, most of these techniques rely on defined templates without considering the rich context information of events. Moreover, these methods fail to remove false dependencies caused by the coarse-grained nature of logs. To address these limitations, this paper proposes a context-aware provenance graph reduction and partition approach for facilitating attack investigation named ProvGRP. Specifically, three features are proposed to determine whether system events are the same behavior from multiple dimensions. Based on the insight that *information paths belonging to the same high-level behavior share similar information flow patterns*, ProvGRP generates information paths containing context, and identifies and merges paths that share similar flow patterns. Experimental results show that ProvGRP can efficiently reduce provenance graphs with minimal loss of crucial information, thereby facilitating attack investigation in terms of runtime and results.

Keywords: provenance analysis; information path; graph reduction; audit log; attack investigation

Citation: Li, J.; Zhang, R.; Liu, J. ProvGRP: A Context-Aware Provenance Graph Reduction and Partition Approach for Facilitating Attack Investigation. *Electronics* **2024**, *13*, 100. https://doi.org/10.3390/electronics13010100

Academic Editor: Baris Aksanli

Received: 14 November 2023
Revised: 20 December 2023
Accepted: 22 December 2023
Published: 25 December 2023

1. Introduction

Enterprises face threats from covert and persistent multi-step attacks, such as Advanced Persistent Threats (APT). These sophisticated attacks employ complex strategies and state-of-the-art techniques to infiltrate target systems and remain lurking for months, stealthily gathering confidential information. To counter these attacks, attack investigation comprehensively analyzes audit logs to identify attack entry points and previously unknown attack traces through comprehensive analysis [1–4]. Audit logs are generated by monitoring system activity using kernel-level audit log collection tools such as Auditd, Sysmon, and Dtrace. These logs record system events and status, capturing information about system entities (e.g., processes, files, and sockets), dependency relationships between entities (e.g., process-file interactions), event timestamps, and other system information.

Attack investigation based on audit logs often employs provenance analysis, which constructs provenance graphs from audit logs, where nodes represent system entities and directed edges represent dependency relationships. Analyzing these provenance graphs enables investigation of attack behaviors and reconstruction of attack scenarios. Several works have demonstrated the effectiveness of utilizing provenance analysis in attack investigation [4–7] and detection [8–10]. However, it is complex and costly to analyze audit logs. To effectively detect Advanced Persistent Threat (APT) attacks with long durations and cover the complete attack process, it is necessary to collect audit logs over large time spans through system monitoring, resulting in a significant volume of audit data. Due to the dependency explosion problem [11,12], the size of the graphs generated from these

audit logs is complex and huge, typically containing 200,000 edges. This complexity makes it challenging for attack investigation approaches to identify attack-related edges and entities within the graphs.

To address this challenge, recent research has proposed methods for provenance graph reduction and compression [12–14]. These approaches are based on the observation that audit logs record a large number of system call events that are redundant and not strictly necessary for attack investigation. Therefore, these methods decrease the size of the provenance graphs by eliminating redundant and irrelevant system events while preserving crucial attack-related information. CPR/PCAR [13] merges repeated low-level events between two OS objects, in order to extract a high-level abstraction of system activities. NodeMerge [12] and LogApprox [15] identify redundant events using predefined templates. For instance, they merge events that involve a process reading or writing to multiple similar files in a short period. These reduction techniques are compared and illustrated in Figure 1.

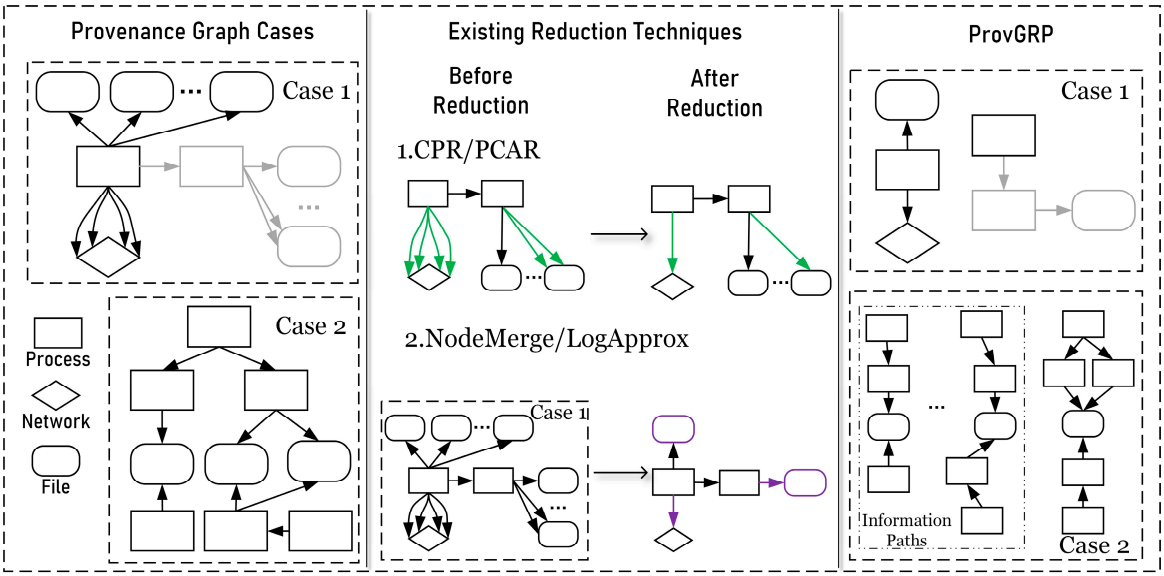


Figure 1. Cases are used to compare and illustrate our approach ProvGRP and existing reduction techniques. The gray nodes and edges in Case 1 represent events that are actually unrelated to the black nodes. CPR/PCAR [13], NodeMerge [12], and LogApprox [15] fail to reduce Case 2 effectively.

Such reduction techniques have demonstrated promising outcomes in enhancing the efficiency of attack investigation. However, there are two major limitations. (1) Audit logs record coarse-grained system operations, which can lead to unrelated dependencies among long-running processes. The main reason for this is that some processes have a long lifetime and iterative input/output processes. In Case 1 of Figure 1, the nodes and edges in gray have no relationship with the black nodes in high-level behavior, but are associated to the same graph. Existing methods fail to partition the two behaviors in the graph. (2) Existing methods cannot identify redundant patterns that have not been defined by schemes and templates. Furthermore, they do not consider the information flow patterns implicit in the deep causal relationships and contextual information of system events. Although the existing techniques can effectively reduce the graph in Case 1, they cannot deal with the redundant events in Case 2 of Figure 1 because they do not consider the context information. Indeed, Case 2 can still be reduced without loss of critical information, and the reduction result is shown on the right of Figure 1.

To address the above limitations, this paper proposes ProvGRP, a novel context-aware provenance graph reduction and partition approach. ProvGRP partitions provenance graphs into subgraphs describing different behaviors and eliminates redundant and behavior-unrelated events, thereby facilitating the efficiency of attack investigation.

Firstly, ProvGRP constructs platform-independent provenance graphs by parsing audit logs. Subsequently, it proposes three features to calculate dependency similarity and utilizes clustering technique HDBSCAN [16] to cluster events with high similarity into an execution partition. The above step is implemented to partition a provenance graph into multiple subgraphs to eliminate erroneous dependencies due to the coarse-grained nature of logs. A key insight of merging redundant events based on context information is that *information paths belonging to the same high-level behavior share similar information flow patterns.* Therefore, ProvGRP obtains context information through forward and backward tracing based on information tributaries, and generates information paths that effectively represent information flow patterns by identifying information paths with similar flow patterns and merging them to reduce redundant system events. Specifically, the information path representation is naturally fit with sequence distance calculation methods. Thus, an improved Levenshtein edit distance is designed to calculate the distance between two information paths. Finally, ProvGRP merges the information paths clustered into one class to achieve the reduction of the provenance graphs.

ProvGRP is evaluated on the publicly available datasets DAPRA TC and ATLAS, which contain labeled attack entities and events. We employ four metrics to conduct a comprehensive and quantitative analysis of the method's effectiveness. Specifically, the reduction factor evaluates the efficiency of the reduction, while Behavior Partition Accuracy evaluates the accuracy of the graph partitioning. Attack Information Loss and Causal Information Loss quantify the information loss situation caused by the reduction method. The experimental results show that the reduction factors achieved by ProvGRP on these two datasets are 10.10X and 42.21X, respectively. Taking into account both the reduction factor and information loss, ProvGRP achieves further data reduction with losing much less information than previous methods. The result of Behavior Partition Accuracy shows that ProvGRP can accurately partition the provenance graph to remove false dependencies. Furthermore, the evaluation results demonstrate that the use of ProvGRP-reduced graphs can facilitate the efficiency of attack investigation approaches while maintaining high performance.

2. Background and Motivation

2.1. Audit Logs and Provenance Analysis

Audit Logs: Audit logs are collected by system monitoring tools from various operating systems, which record in detail the actions and status of the system layer. Each audit log is an encapsulation of a specific system event or system call, containing details about system objects, relationships between objects, timestamps, event IDs, and other requisite system information. Audit logs capture three types of system entities: Process, File, and Network. The types of relationships are various according to different object operations. (1) The relationships between recorded processes include process execution, fork, and close, etc. (2) The relationship between processes and files records the operation from processes, such as read, write, and delete. (3) The relationships recorded in the logs of the network object include link, close, etc. Thereby, an audit log can be represented as a quadruple <*event= Subject, operation, Object, timestamp*>. For instance, the quadruple <*WORD.EXE, write, Desktop\my_report.docx*, 2023/4/22 9:31:32> represents the process, *WORD.EXE* writes a file named *my_report.docx* in the path *Desktop*, and the corresponding high-level user behavior saves the *my_report.docx* file on the Desktop.

Provenance Analysis: Provenance analysis has been widely applied to attack investigation and detection. Provenance analysis analyzes the audit logs to infer the dependencies and construct a directed labeled graph known as the provenance graph. By performing comprehensive causality analysis on the graphs, provenance analysis can identify attack behavior patterns and traces. In fact, the causal relationships and contextual data found within audit logs provide valuable insights into the tactics and goals of attackers, which are inherently difficult to hide. In the provenance graph $G = \left(V_{Entity}, E_{dep}\right)$, a node $v \in V_{Entity}$ represents a system entity (such as a process, a file, or an IP address). An edge $e \in E_{dep}$

represents a dependency relationship (such as process *write**read* file) with its direction indicating the relationship between two entities.

2.2. Motivating Example

Scenario. Consider the simulated APT attack scenario implemented by an APT group named Kimsuky [17]. This scenario encompasses both malicious activities and regular user actions. For instance, a user habitually checks emails and downloads attachments. Among these emails, there is a phishing attempt containing a *.zip* file carrying a malicious *.scr* file. It writes a DLL file and sets the *registry key* to establish persistence. Subsequently, the DLL file utilizes the technique of process hollowing to inject malware code into *explorer.exe* to avoid Anti-Virus detection. Finally, the attacker transmits encrypted information concerning the compromised machine. Concurrently, the user performs activities such as downloading documents, working with data samples, installing Python, and executing Python codes, etc. Figure 2 illustrates a simplified provenance graph depicting the aforementioned process. The gray boxes and lines in Figure 2 indicate the key nodes and steps of the attack.

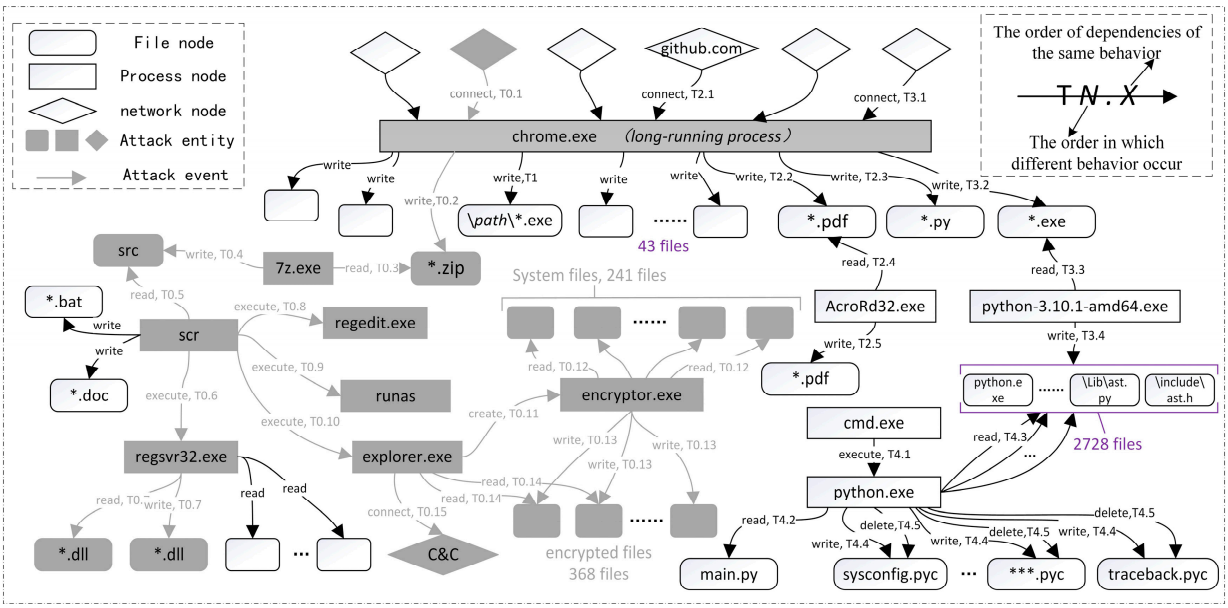


Figure 2. The provenance graph of the Kimsuky attack scenario. The gray boxes and directed edges represent attack events. The simplified provenance graph describes several behaviors, which are associated together by long-running process chrome.exe. **Limitation.** The complete provenance graph depicted in Figure 2 is huge and complex, and contains 19,306 nodes and 116,732 edges. It is difficult and time-consuming to conduct attack investigation in this graph. Moreover, the above graph is only constructed from audit logs collected on a single host within 12 h. We combine the above attack scenario to provide a more intuitive analysis of the two main limitations of existing reduction techniques.

CPR/PCAR [13] merges repeated low-level events between two objects, in order to extract a high-level abstraction of system activities. CPR/PCAP can merge the *python.exe* writing and deleting pyc files in the graph, and the reduced graph is still noticeably complex. NodeMerge [12] only merges a set of read-only files that are always accessed together in the system events. The system event of the encryptor.exe process reading 241 files in the graph can be merged to one node by NodeMerge. LogApprox [15] solely merges similar files read and written by a process, employing regular expression learning to assess their similarity. Compared to NodeMerge, it can achieve better reduction results because it can merge python.exe to write and delete all *pyc* files into a single event. However, these methods can only reduce redundant events for specific schemes and templates, and cannot effectively deal with undefined redundant patterns. More importantly, not considering the

context information limits these methods from further identifying multi-step redundant events.

It can be seen from Figure 2 that *chrome.exe* is a long-running process, and it associates multiple high-level behaviors related to it within the same graph. For example, there is no causal relationship between the behavior of download mail attachment <*chrome.exe write* *.*zip*, *T0.2*> and the behavior of downloading installation packages <*chrome.exe write* *.*exe*, *T3.2*> during the lifetime of the process. This makes it difficult to determine the correct information flow path. However, some previous work did not consider this situation. Existing methods do not solve this problem.

Using ProvGRP. ProvGRP first partitions the original provenance graphs into multiple subgraphs, thereby removing false dependencies between different high-level behaviors. Here, ProvGRP partitions *chorme.exe* into multiple execution partitions, so that high-level behaviors such as downloading malware, downloading code, and downloading python packages are separated in different subgraphs. The error dependencies between attack event <*chrome.exe write* *.*zip*, *T0.2*> and other normal events are removed correctly. Then, ProvGRP constructs information paths based on context information, and realizes the identification of redundant events by calculating the similarity between these information paths. System events describing the same behavior in Figure 2 are accurately and efficiently merged. For example, the system events that *python.exe* generates and deletes a large number of pyc files are merged into a single event <*python.exe read py file*, *T4.2*> to represent the high-level behavior of running the python code.

3. Approach Overview and Threat Model

3.1. ProvGRP Overview

The overall workflow of ProvGRP is shown in Figure 3. ProvGRP acts as the predecessor work of attack investigations to reduce provenance graphs. It takes the provenance graphs generated by the audit logs from different operating systems as input and outputs the reduced provenance graphs for attack investigation. ProvGRP consists of three components: (1) Provenance Graph Construction, (2) Provenance Graph Partition, (3) Behavior-unrelated Events Elimination. The Provenance Graph Construction component constructs platform-independent provenance graphs from collected audit logs by performing causality correlation (Section 4.1). The Provenance Graph Partition component implements the partitioning of long-running process nodes by event features (Section 4.2). The Behavior-unrelated Event Elimination component generates *information paths* and merges similar paths to reduce the behavior-unrelated events (Section 4.3).

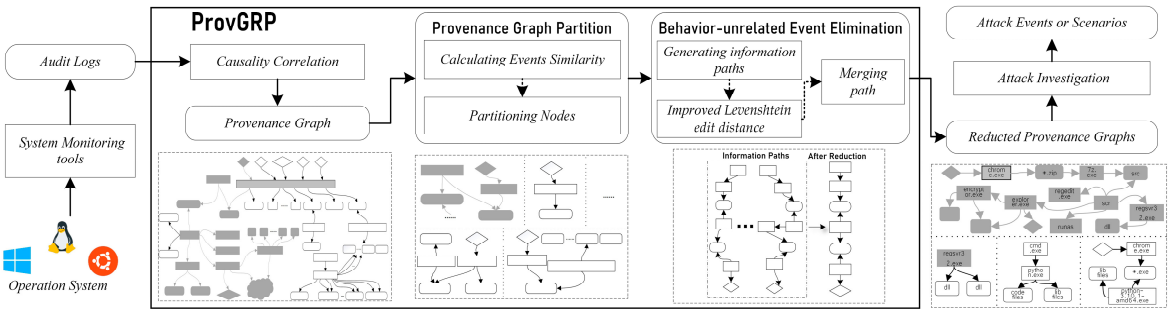


Figure 3. The black box indicates the overall workflow of ProvGRP. ProvGRP's input is audit logs collected from different operating systems (left of the black box). ProvGRP outputs reduced provenance graphs for facilitating attack investigation.

3.2. Threat Model

Our threat model is similar to the threat model of previous work [7,8,18–20], which assumes that the underlying operating system, audit engine, and monitoring data are part of the trusted computing base (TCB). Kernel-level attacks that compromise the audit monitoring and log collection systems are beyond the scope of this work. This paper also

assumes that the integrity of audit logs is maintained, meaning that attackers cannot modify or delete them. Additionally, the usage of existing software and secure provenance systems can ensure the secure capture and storage of logs.

4. Approach Design

4.1. Provenance Graph Construction

System monitoring tools are utilized to collect audit logs from various operating systems, such as Windows, Linux, and Unix. ProvGRP then parses the collected logs to construct platform-independent provenance graphs, which are directed labeled graphs. The nodes represent system entities, and the directed edges represent system events. Specifically, ProvGRP extracts information about system events recorded in audit logs including node attributes and dependency attributes. ProvGPR focuses on three types of node entities, process, file, and network. The dependency attributes include operations and timestamps, and ProvGRP uses a quadruple to represent system events $<$ $Subject$, $operation$, $Object$, $timestamp$ $>$. Table 1 displays the complete extraction information of system events. ProvGRP performs causality correlation on the extracted data to construct provenance graphs. The node entities include the file name (*c:/windows/system32/es.dll*), process name and PID (*svchost.exe_896*), and IP address (*192.168.223.2:53*). The labels of the directed edges include the operations (read, execute, and connect, etc.) and the timestamp. Finally, ProvGPR eliminates unknown nodes and isolated nodes which are the noise produced by the system monitoring.

Table 1. Attributes of extracted node attributes and dependency attributes.

Subject Entity		Object Entity		Denpendency (Edge)	
Type	**Attributes**	**Type**	**Attributes**	**Operations**	**Attributes**
Process	Name, PID, User, Type, Subject ID	Process	Name, PID, User, Type, Object ID	[execute, fork, clone, close]	Timestamp, Cmd lines, Event ID
		File	Name, Path, Type, Object ID	[read, write, delete]	Timestamp, Event ID
		Network	IP, Port, URL, Protocol, Object ID	[connect, connected session, sock send]	Timestamp, Evnet ID

4.2. Provenance Graph Partition

The goal of this component is to partition the provenance graphs into multiple subgraphs to remove false dependencies caused by the coarse-grained nature of logs. The subgraphs depict different high-level behaviors, which have no relationship with each other. To achieve this, ProvGRP proposes a new provenance graph partition algorithm. This algorithm abstracts three behavior features to judge which system events belong to the same high-level behavior. Based on these features, incoming and outgoing edges are clustered separately, and each class cluster corresponds to an execution partition. Subsequently, ProvGRP merges the execution partitions of the incoming edges and the execution partitions of the outgoing edges based on the time of the events in the clusters. Through the above steps, the original provenance graph is partitioned into multiple subgraphs.

4.2.1. Feature Definition

ProvGRP extracts three features to calculate the similarity between dependencies. These three features can comprehensively quantify the similarity between dependencies from multiple dimensions.

Time Interval Feature $f_{T(e)}$. Intuitively, dependencies belonging to the same behavior have much smaller time intervals compared to those belonging to different behaviors. To validate this intuition, we conducted a statistical analysis of the time intervals between system events belonging to the same behavior in two publicly available audit log datasets

(the results are depicted in Section 5.2). Thus, the feature $f_{T(e)}$ is designed to model this intuition.

$$f_T(e_i, e_j) = 2 \times atan(\frac{T_{end} - T_{start}}{\left|t_{e_i} - t_{e_j}\right| + \alpha} - \beta)/\pi \quad (1)$$

where t_{e_i} and t_{e_j} represent the timestamp of the events (edges) e_i and e_j, which are either incoming or outcoming from the same node. T_{end} is the latest timestamp among the edges, and T_{start} is the start timestamp. $atan(t) = tan^{-1}(t)$ is an arctangent function, and α (we set $\alpha = 0.001$) is a positive number used to make sure the denominator is not 0. When two events occur at the same time, the value of $f_T(e_i, e_j)$ is close to 1. $\beta = \frac{T_{end} - T_{start}}{T_{end} - T_{start} + \alpha}$ to ensure the value of $f_T(e_i, e_j)$ is 0 when the time interval between events is maximum. The value of the formula ranges from 0 to 1.

Entity Name Feature $f_{E(e)}$. The subjects or objects of system events that belong to the same behavior have high similarity. This intuition is based on observing the habits of users and computers. For example, when unzipping a zip file, the files are typically extracted to the same directory path. Then, the feature $f_{E(e)}$ is designed to model this intuition. Depending on the type of comparison nodes, $f_{E(e)}$ uses different formulas.

If the two entity nodes are different types, $f_{E(e)} = 0$. If the type of two entity nodes is process, the value of $f_{E(e)}$ is calculated by process name and PID, and

$$f_E(n_{e_i}, n_{e_j}) = \begin{cases} 1 \; if \; n_{e_i} = n_{e_j} \\ 0 \; otherwise \end{cases} \quad (2)$$

where n_{e_i} and n_{e_j} are entity node names of the events (edges) e_i and e_j, and n_e is represented as *process_PID*. If the type of two entity nodes is IP address, the value of $f_{E(e)}$ is calculated by counting the number of the same initial bits of IP address, and

$$f_E\left(n_{e_i}, n_{e_j}\right) = same\_bit\left(binary(n_{e_i}), binary\left(n_{e_j}\right)\right)/32 \quad (3)$$

where $binary(n_e)$ converts the decimal IP address to binary, and *same_bit* counts the number of the same initial bits. If the type of two entity nodes is file, path, or URL, the value of $f_{E(e)}$ is calculated by comparing path and file name. For the similarity between two paths, we treat each directory name as a token. Compare each token from the root directory and end when the token is different. Thus, the feature $f_{T(e)}$ is designed as follows:

$$f_E\left(n_{e_i}, n_{e_j}\right) = same\_token\left(token(n_{e_i}), token\left(n_{e_j}\right)\right)/max\_len \quad (4)$$

where $token(n_e)$ splits a file path into tokens, and $max\_len = max\left(len\left(token(n_{e_i}), token\left(n_{e_j}\right)\right)\right)$. These formulas all yield values in the range [0, 1].

Operation Feature $f_{O(e)}$. Operations belonging to the same behavior have a potential relationship, meaning that within the same behavior, there are only a few fixed types of operation. For example, the same operation most likely belongs to the same behavior, and write and delete operations often co-occur within the same behaviors. Therefore, we define the Operation Feature as follows:

$$f_O\left(o_{e_i}, o_{e_j}\right) = bool\_oper\left(o_{e_i}, o_{e_j}\right) \quad (5)$$

where *bool_oper* is a bool function that the value is set to 1 when the operations o_{e_i} and o_{e_j} match the rule that belongs to the same behavior, otherwise 0.

Finally, the confidence value that two system events e_i and e_j belong to the same behavior is obtained by weighting the three aforementioned feature values. The formula is defined as follows:

$$F(e_i, e_j) = \omega_T f_{T(e)} + \omega_E f_{E(e)} + \omega_O f_{O(e)} \quad (6)$$

ω_T, ω_E, and ω_O are the weight coefficients of each feature, and their sum is 1. In this work, the weight coefficients are empirically determined as $\omega_T = 0.5$, $\omega_E = 0.4$, and $\omega_O = 0.1$. The value range of $F(e_i, e_j)$ is [0, 1].

4.2.2. Provenance Graph Partition

ProvGRP uses the above features to calculate the confidence matrices $F_{in(i,j)} \in \boldsymbol{F}_{in}^{N\times N}$ and $F_{out(i,j)} \in \boldsymbol{F}_{out}^{M\times M}$ for the in-edges and out-edges of each long-running node. Since the number of partitions is uncertain, ProvGRP uses HDBSCAN to implement our clustering task, which does not need to declare the number of clusters in advance and has good robustness to outliers. HDBSCAN can receive the all pairs matrix, and the code published by McInnes L et al. [21] is adopted to implement the clustering task. For long-running process nodes, incoming and outgoing edges are gathered separately. The incoming and outgoing edges are then associated according to information reachability, with the requirement that the information flow follows the chronological order of events. This means that the incoming edge should exist before any of the outgoing edges.

Figure 4 illustrates the partitioned subgraphs. By partitioning the long-running process (*chrome.exe*), ProvGRP divides the provenance graph into subgraphs that describe different behaviors. Specifically, ProvGRP calculates the dependency distance between dependencies associated with the process chrome.exe and performs clustering, as shown in the upper part of Figure 4. Subsequently, based on the reachability of the information flow, the incoming and outgoing dependencies belonging to the same information flow are associated within the same execution partition. Finally, by splitting the nodes, the original graph is divided into multiple subgraphs, as depicted in the lower part of Figure 4.

Based on the clustering results, ProvGRP merges the leaf nodes (nodes with in-degree or out-degree of 0) within the same cluster. This reduces the number of information paths that are subsequently generated, thus improving the efficiency of log reduction.

4.3. Behavior-Unrelated Events Elimination

The low-level and verbose nature of audit logs makes the presence of behavior-unrelated events severely impact the efficiency of the attack investigation. In behavioral instances, many events are redundant, and removing or merging them does not alter the information flows. Previous approaches to event merging and elimination have focused on individual event characteristics without considering contextual information and information transfer paths. As a result, a large number of redundant events still exist in the reduced data. To identify these redundant and behavior-unrelated events, ProvGRP obtains the context information of events based on the information flow and generates *information paths* of a specific length. By merging these paths, ProvGRP can remove or merge behavior-unrelated events effectively.

Information Path. Information flows represent the transfer path of information between system entities in provenance graphs. As can be seen from Figure 5A, information from one node will pass through multiple information tributaries and merge at a certain node. These information tributaries can be merged into one information flow without loss of critical information describing high-level behavior, because they belong to the same behavior instance and have a similar flow pattern. An information path, denoted as P, indicates an information tributary that is a chain of events in which all events record the same flow of information. It is an ordered sequence of dependency events and represented as $P = \{e_1, e_2, \ldots, e_n\}$. For example, a specific information path $P = \{< chrome.exe, write, report.pdf >, < AcroRd32.exe, read, report.pdf >, < AcroRd32.exe, write, report*.pdf >\}$ has a length of 3. First, the information is transferred into *report.pdf* by *chrome.exe*. Then, the information flows to the *AcroRd32.exe* by reading the *report.pdf*. Finally, *AcroRd32.exe* writes a new file *report*.pdf*, which represents the information flows into the file. It should be noted that the flow direction of information is not always consistent with the direction of edges. Furthermore, events may appear in multiple paths, indicating the existence of multiple information flows through these events.

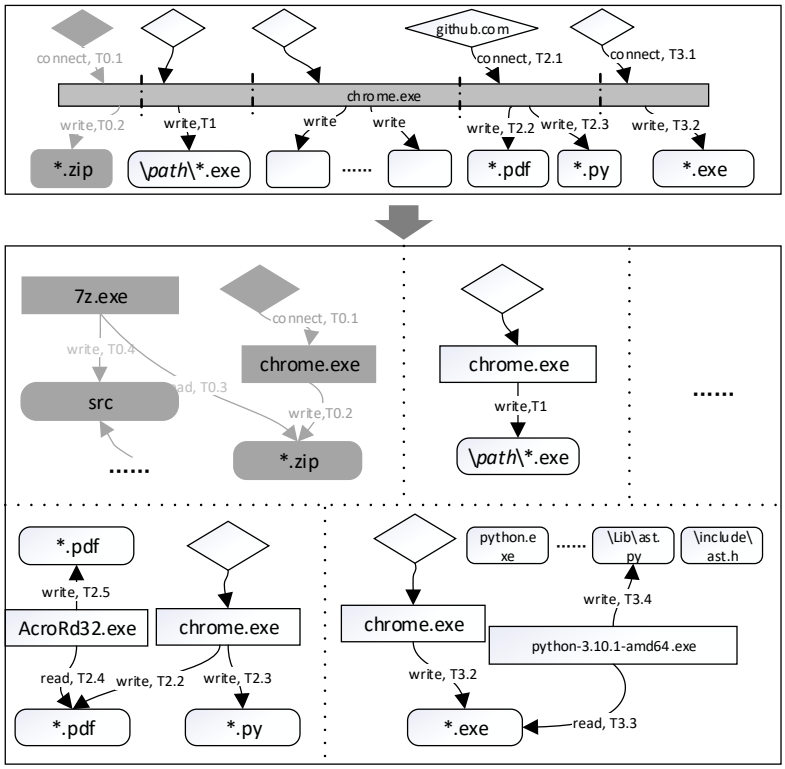


Figure 4. Different behaviors are divided into different subgraphs by partitioning long-running processes into execution partitions.

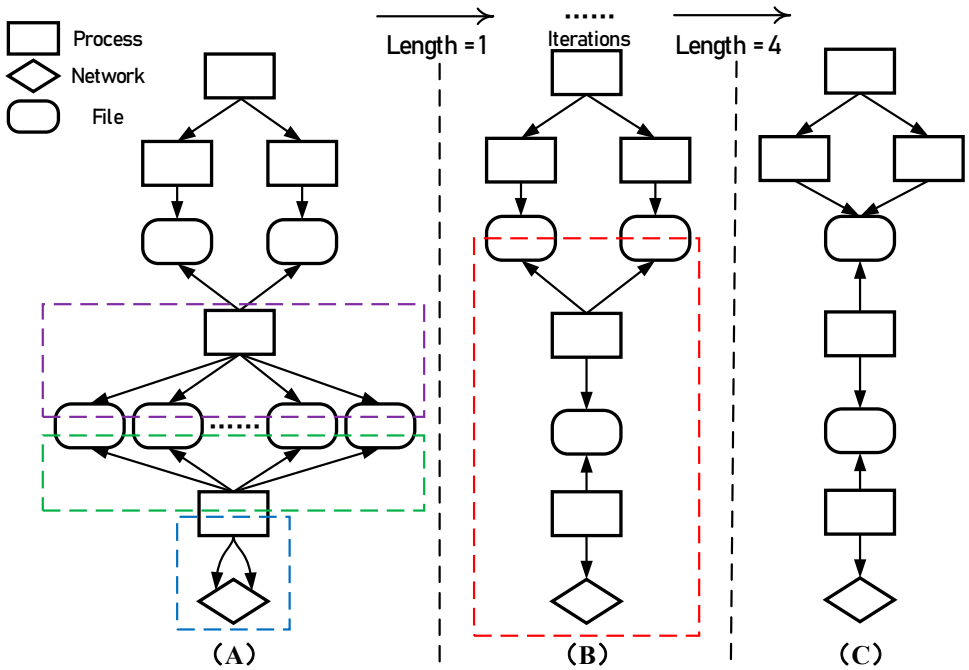


Figure 5. The process of merging paths based on similarity. Nodes and edges in the dotted box represent the merged path. (**A**) represents the segmented subgraph, (**B**) represents the intermediate result in the iteration, and (**C**) represents the final result of the provenance subgraph.

Merging Information Paths. ProvGRP employs an iterative process to merge similar paths based on their length. This has the advantage of reducing the number of long information paths and improving merge efficiency. Our goal is to derive an algorithm

that accurately quantifies information path similarity based on these three features. The information path representation is naturally fit with sequence distance calculation methods. Thus, an improved Levenshtein edit distance is designed to calculated the distance between two information paths P_i and P_j:

$$LP(m,n) = \begin{cases} max(m,n)\ if\ min(m,n) = 0 \\ min\begin{cases} LP(m-1,n)+1 \\ LP(m,n-1)+1 \\ LP(m-1,n-1)+(1-F(e_m,e_n)) \end{cases} otherwise. \end{cases} \quad (7)$$

where $1 - F(e_m, e_n)$ calculate the distance between e_m and e_n in P_i and P_j, respectively. For information path pair P_i^L and P_j^L of a certain length L, the normalization distance between them can be expressed as $LP_{P_i^L,P_j^L}(L,L)/L$, the value range of which is [0, 1]. We empirically determined the threshold to be 0.2. If the distance is lower than the threshold, the information paths P_i^L and P_j^L are merged. ProvGRP only calculates the distance of information paths that contain at least one identical event. Moreover, it is important to note that process nodes are not merged because most processes are directly associated with behaviors, and merging them would result in the loss of crucial behavior information.

Figure 5 visually illustrates the path merging process. Figure 5A is a provenance subgraph generated from Provenance Graph Partition, where the dotted box indicates information paths of length 1. These paths are subjected to similarity calculation and merged, resulting in Figure 5B. when the path length reaches 4, the two paths within the red dotted box are merged, as shown in Figure 5C. Subsequent iterations do not identify path pairs that exceed the similarity threshold, and the merging process terminates when the path length reaches 6, which is the maximum length. Algorithm 1 outlines the iterative calculation of path similarity for path merging.

Algorithm 1: Behavior-unrelated Event Elimination

Input: *Provenance subgraphs* $G_1, G_2, \ldots G_n$
Result: *Merged Provenance subgraphs* $G'_1, G'_2, \ldots, G'_n$
for each G_i **do**
  Set $max\_len \leftarrow G_i$;
  Empty list P;
  while $len\_path < max\_len$ **do**
    Add P_n to P;
    Calculate each pair paths $P_i,\ P_j \epsilon P$ with $S_{path}\left(P_i,\ P_j\right)$
    if $S_{path}\left(P_i,\ P_j\right) > 0.8$ **then**
      Merge paths P_i and P_j;
Return G'_i

5. Evaluation

5.1. Datasets and Evaluation Metrics

5.1.1. Datasets

The effectiveness and generalizability of ProvGRP are evaluated using two publicly available audit log datasets, ATLAS Dataset [1] and DAPRA CADETS dataset [22]. These datasets are widely used for evaluating attack investigation methods and contain ground truth information about attack scenarios.

ATLAS Dataset. The ATLAS Dataset, as provided in reference [1], consists of 10 repeated APT attacks with different vulnerabilities and attack strategies. The dataset covers a time span of 24 h, and each attack is completed within one hour. During the 24 h emulation, an average of 20,784 unique entities and 249 K events are generated for each attack. This paper constructs provenance graphs based on the attack scenarios in the dataset. In the previous work, these attack scenarios were numbered. For ease of reference, this paper uses the dataset name followed by the corresponding number to represent

the attack scenarios in the evaluation. For example, ATLAS. S-1 represents Strategic web compromise, which is an APT campaign.

DAPRA CADETS dataset. The DAPRA CADETS dataset [22] is released by the DARPA Transparent Computing program. This dataset was collected during DARPA's two-week red team vs. blue team Engagement 3. The dataset includes attacks against the FreeBSD system, with the red team executing multiple attack methods four times during different time periods. Throughout the attack, normal behaviors such as SSH login may also occur on the host. A total of 44,404,339 OS-level audit logs were collected over the two-week period.

5.1.2. Evaluation Metrics

Reduction factor. The reduction factor measures the rate of edges (events) removed or merged. Therefore, the reduction factor is defined as Num_{orig}/Num_{redu}, where Num_{orig} is the number of edges in the original provenance graphs and Num_{redu} is the number of edges after Reduction and Partition by ProvGRP.

Behavior Partition Accuracy. This metric evaluates the accuracy of the provenance graph partition. Here, we evaluate the accuracy of ProvGRP in separating attack behaviors from normal behaviors, i.e., the proportion of events related to the attack in the subgraph describing the attack scenario. Thus, Behavior Partition Accuracy is defined as follows.

Given a ground truth provenance graph of an attack scenario $G_{att} = (V, E)$ and partitioned provenance subgraph $G'_{att} = (V', E')$, a lossless subgraph implies that $E' = E$ and $V' = V$. The accuracy of provenance graph partition can be measured using the formula.

$$\mathcal{A}cc_{\mathcal{BP}} = \frac{|E' - E| + |V' - V|}{|E| + |V|} \tag{8}$$

Attack Information Loss. This metric evaluates the loss of critical information related to the attack after removing behavior-unrelated events. The key attack information is crucial in the attack investigation. Mistakenly removing key attack information (nodes or events) may disrupt the attack path, making it challenging for context-based attack investigation methods to obtain complete context information of the attack events. It may also hinder matching-based attack investigation methods in effectively matching similar attack graphs. Therefore, the metric is used to describe the attack information loss.

Given a ground truth provenance graph of an attack scenario $G_{att} = (V, E)$, where $\varepsilon_i \epsilon \mathbb{E}$ represents attack-related system events in G_{att}, G_{rm_att} represents a provenance graph that contains attack events and has removed behavior-unrelated events, and $\varepsilon'_i \epsilon \mathbb{E}'$ represents attack-related system events in G_{rm_att}. The loss of the attack information can be measured using the formula:

$$\mathcal{L}_{AI} = 1 - \frac{|\mathbb{E} \cap \mathbb{E}'|}{|\mathbb{E}|} \tag{9}$$

Causal Information Loss. Graph reduction methods may mistakenly remove events related to normal behaviors, impacting the effectiveness of attack investigation methods that learn from both attack and normal behaviors.

Give all ground truth events $\mathbb{E}_{GT} = \{\varepsilon_1^{att}, \ldots, \varepsilon_m^{att}, \varepsilon_1^{nor}, \ldots, \varepsilon_n^{nor}\}$ that are directly related to behaviors. $\mathbb{G} = \{G_1, G_2, \ldots, G_n\}$ represents a set of subgraphs after removing behavior-independent events. Loss is defined as the proportion of any event $\varepsilon \in \mathbb{E}_{GT}$ that does not appear on any graph $G \in \mathbb{G}$, which can be measured using the formula:

$$\mathcal{L}_{CI} = 1 - \frac{|\mathbb{E}_{GT} \cap \mathbb{G}|}{|\mathbb{E}_{GT}|} \tag{10}$$

5.2. Feature Verification

The features used in ProvGRP are proposed based on intuition and statistical analysis. Below, we illustrate the validity of these features through statistical analysis results on two publicly available datasets used in our experiments.

Time Interval Feature. Figure 6 shows the number of events belonging to the same behavior for different time densities. The time density represents the relative time interval, that is, the proportion of the time interval in the total lifetime of the node. It is calculated by $\frac{|t_{e_i} - t_{e_j}| + \alpha}{T_{end} - T_{start}}$, and a lower value indicates that two events are closer to each other. In the time density distribution, events are mainly concentrated in the 0–0.05 and 0.95–1 ranges. Events in the 0–0.05 range belong to the same behavior of long-running processes, while events in the 0.95–1 range are executed by short-lived processes, where each process carries out a single behavior, resulting in event intervals approximately equal to the process lifecycle.

Entity Name Feature. Figure 7 shows the number of event pairs with different entity similarities within the same behavior. The value is calculated by $f_{E(e)}$. In the similarity distribution, events within the same behavior exhibit entity similarities greater than 0.6, whereas in attack-related behaviors, entity similarity is mostly distributed in the range of 0.8–1.

Operation Feature. Figure 8 represents the distribution of operations in the same behavior. We can see that the distribution is similar in both datasets. In the operation distribution, there are potential relationships between different operations within the same operation.

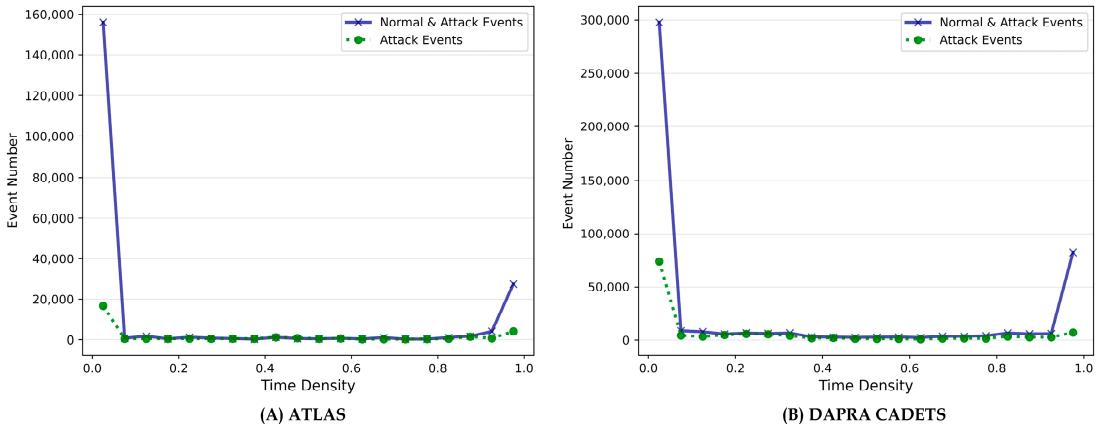


Figure 6. Statistical analysis results of Time Density of two publicly available datasets (**A**) ATLAS and (**B**) DAPRA CADETS. The blue line indicates the analysis result of all events, and the dotted green line indicates the statistics result of attack events.

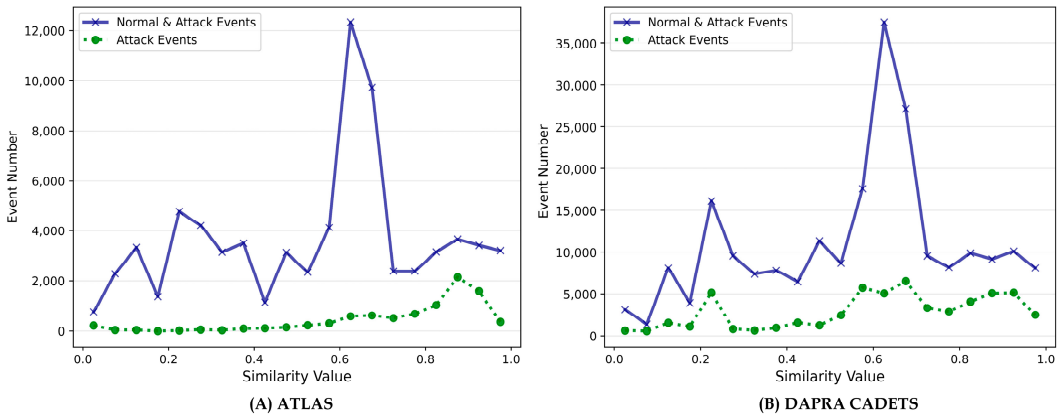
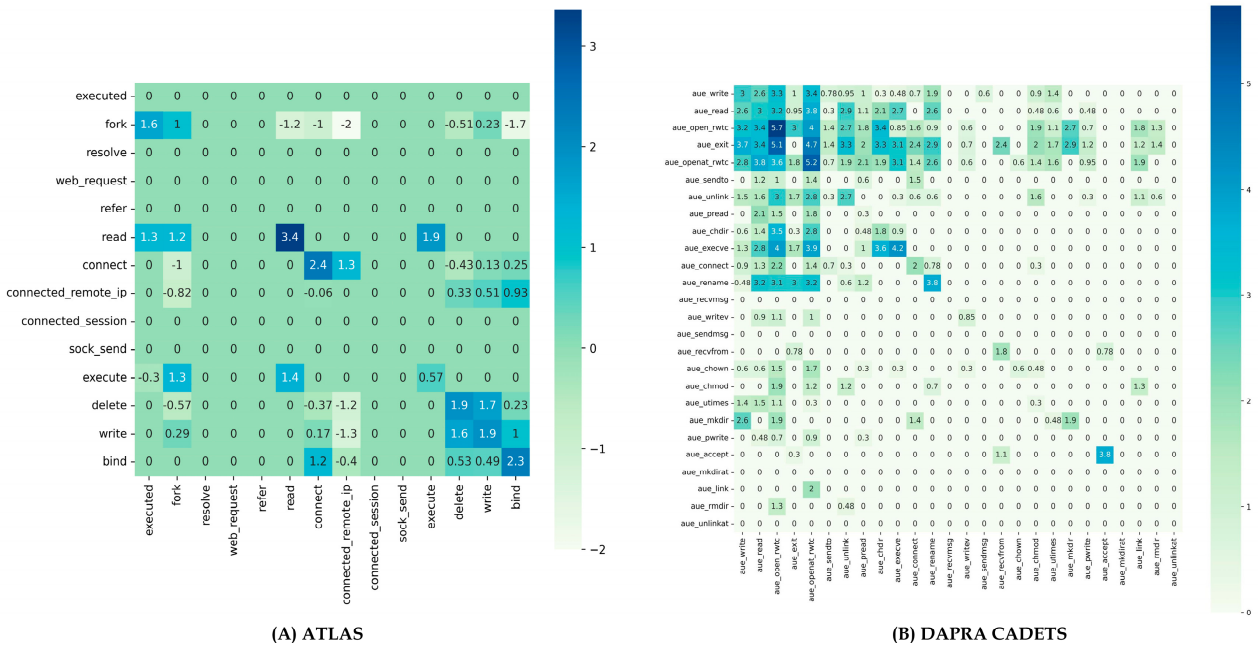


Figure 7. Statistical analysis results of Similarity Value of two publicly available datasets (**A**) ATLAS and (**B**) DAPRA CADETS.



(A) ATLAS **(B) DAPRA CADETS**

Figure 8. Statistical analysis results of operation distribution of two publicly available datasets (**A**) ATLAS and (**B**) DAPRA CADETS. The base 10 logarithm is performed on the number of events.

5.3. Performance Evaluation

Reduction Performance. The effectiveness of ProvGRP is evaluated using the reduction factor. Table 2 shows the reduction factors for the attack scenarios in the publicly available datasets. It can be observed that ProvGRP achieves reduction factors ranging from 6.02X to 55.81X for different attack scenarios. On average, ProvGRP reduces the size of provenance graphs generated from the ATLAS dataset by 10.10X, and from the CADETS dataset by 42.21X. The variation in reduction factors is a result of the distinct data characteristics between the two datasets, which utilize different system monitoring tools. It is evident that the reduction factors are similar within the same dataset. The reduction factors of different attack scenarios within the same dataset are similar. Additionally, certain attack scenarios involve user actions that generate a large number of dependencies in a short period of time, such as installing software and running codes. These dependencies are effectively merged and removed by ProvGRP.

Effectiveness Evaluation. The effectiveness of ProvGRP is evaluated using the three metrics: Behavior Partition Accuracy, Attack Information Loss, and Causal Information Loss. The two public datasets contain multiple attack scenarios, with marked information related to the attacks such as entities and events. For each attack scenario, quantitative evaluations of the reduction effectiveness are conducted. The results of the evaluation are presented in Figure 9.

For Behavior Partition Accuracy, taller bars indicate better partition accuracy, meaning that our method correctly divides different behaviors into separate subgraphs. The partition results obtained by our method range from 0.805 to 0.954 across different attack scenarios. These results indicate that the provenance subgraphs describing the attack behaviors retain the majority of attack-related events while containing only a small number of normal system events. This provides greater accuracy for some attack investigation methods. As shown in Figure 9, the partition accuracy for the initial six attack scenarios is generally higher than that of the last three. This difference arises from the presence of numerous normal behaviors and system calls in CADETS that occur closely with the attack behaviors, resulting in their inclusion in the same subgraph. However, the predominantly leaf nodes representing normal behaviors do not significantly affect the topology of the subgraph that describes the attack behavior.

Table 2. The scale of the reduction graph and reduction factor obtained by CaDR in different attack scenarios.

Attack Scenarios	Prov. Graph		Reduction Graph		Reduction Factor
	$\|V\|$	$\|E\|$	$\|V\|$	$\|E\|$	
ATLAS. S-1	7468	14,820	1265	1330	11.14X
ATLAS. S-2	33,990	42,126	2749	3981	10.58X
ATLAS. S-3	8975	19,545	1985	3248	6.02X
ATLAS. S-4	13,021	23,669	1025	1730	13.68X
ATLAS. M-1	17,633	38,131	1842	2965	12.86X
ATLAS. M-2	24,489	45,775	2246	4782	9.57X
ATLAS. M-3	24,472	40,040	2312	4293	9.33X
ATLAS. M-4	15,405	32,217	987	1382	23.31X
ATLAS. M-5	35,716	52,934	3268	6092	8.69X
ATLAS. M-6	26,685	37,722	3109	4562	8.27X
ATLAS. Total	207,854	346,979	20,788	34,365	10.10X
CADETS. case-1	237,722	519,657	10,731	14,194	36.61X
CADETS. case-2	298,722	730,717	12,889	13,094	55.81X
CADETS. case-3	175,607	296,206	5106	9350	31.68X
CADETS. Total	712,051	1,546,580	28,726	36,638	42.21X

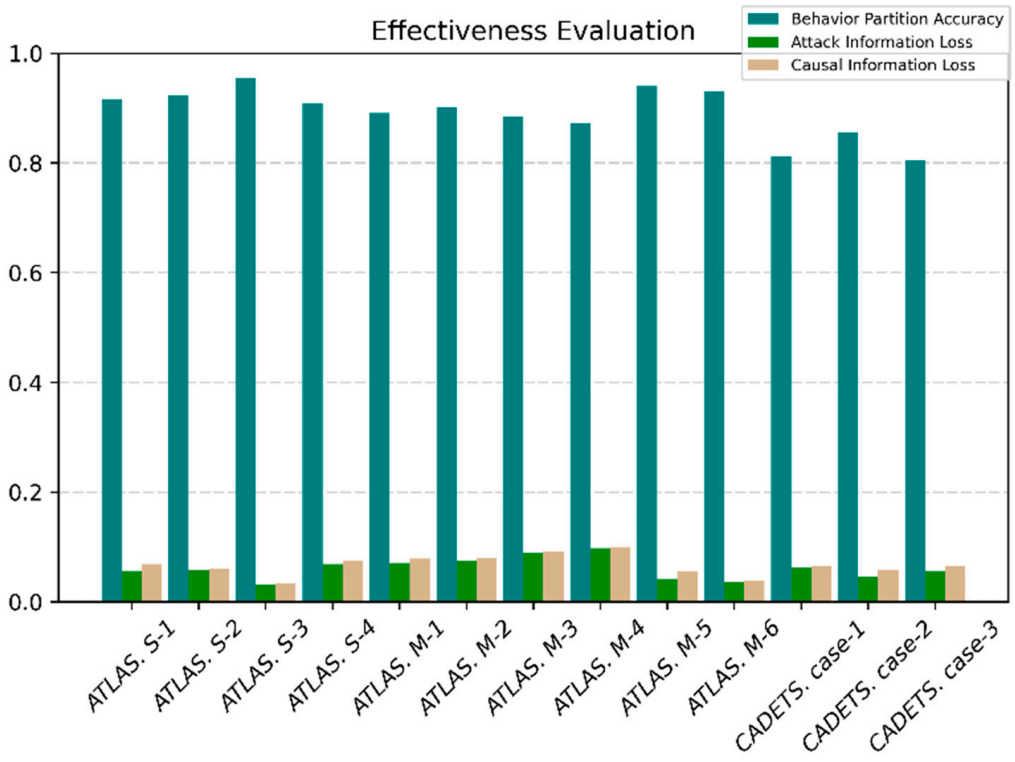


Figure 9. The three metrics value for ProvGRP in different attack scenarios.

The quantification of loss includes Attack Information Loss and Causal Information Loss. The loss of attack information is controlled within the range of 0.031 to 0.098 across all attack scenarios. This demonstrates that our method avoids substantial loss of attack information. The analysis reveals that the lost attack information mainly consists of repeated

or failed attack events in the initial stages, such as port scanning and unsuccessful SSH connections. The loss of such information does not significantly impact subsequent attack investigations. Causal Information Loss includes both normal and attack information loss. The loss of causal information is controlled within the range of 0.033 to 0.1 across all attack scenarios.

5.4. Comparative Evaluation

ProvGRP is compared with other reduction methods using the publicly available CADETS dataset. Since the source code of these methods is unavailable, we attempted to recreate them based on their descriptions in the original papers. Figure 10 shows the change in reduction ratios for the generated graphs compared to the original provenance graphs as the original provenance graphs are reduced. ProvGRP generates the smallest graphs, removing 97.63% of the system events from the provenance graphs of CADETS. In contrast, LogApprox produces the largest graphs, removing 74.05% of the system events. ProvGRP still achieves the best results on ATLAS dataset, and the reduction performance of the other two methods is equal to that achieved by the method proposed by ATLAS in data preprocessing. The experimental results show that ProvGRP can achieve better data reduction results than the existing methods. Additionally, we compare the performance of our re-implemented method with the original work. Although the re-implemented method does not reach the peak of the previous work, the difference is not significant. The main reason for this difference is that the datasets used are different.

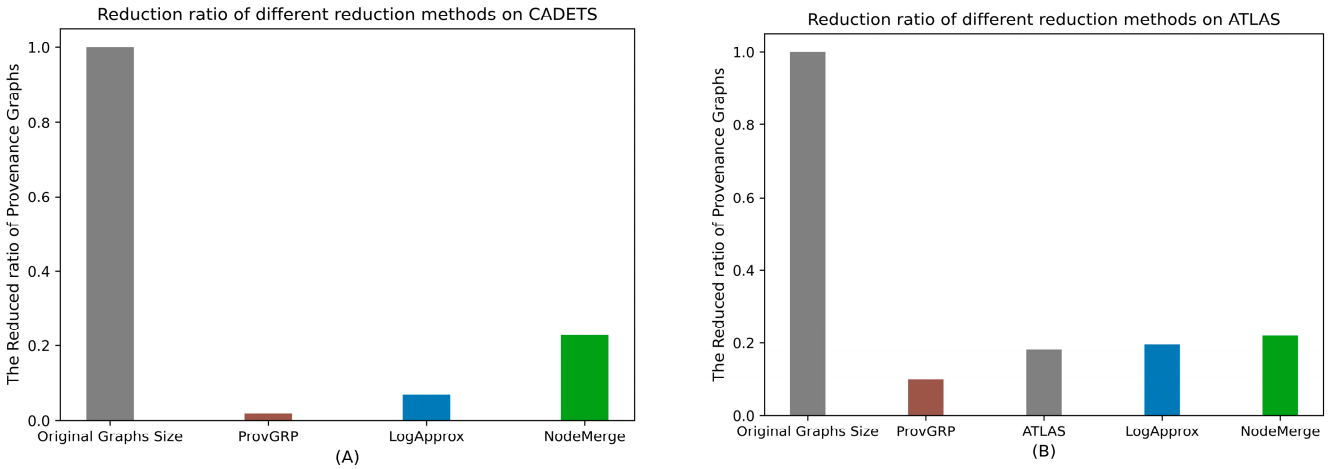


Figure 10. Comparison of reduction performance of different reduction methods on the two datasets CADETS (**A**) and ATLAS (**B**).

The existing methods are also evaluated using the three proposed metrics and compared with ProvGRP. The results are shown in Figure 11. Since LogApprox and NodeMerge do not partition provenance graphs, the graphs they generate contain numerous normal events and entities, resulting in lower values for Behavior Partition Accuracy. LogApprox exhibits nearly no loss of attack information, which aligns with the results from the original work. However, LogApprox experiences higher causal information loss compared to ProvGRP, while its reduction factor is the smallest. Considering the comprehensive evaluation, ProvGRP can achieve further data reduction with losing much less information than previous methods, and can accurately partition the provenance graph to remove false dependencies.

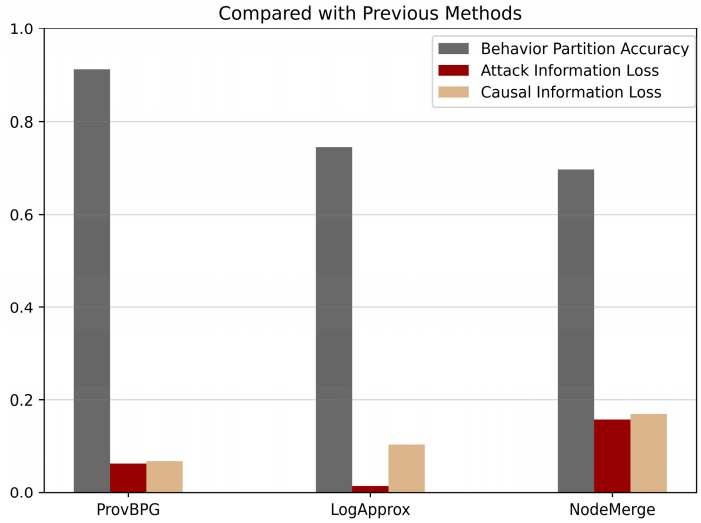


Figure 11. Comparison of metrics of different methods.

5.5. Attack Investigation Facilitation

To evaluate the facilitation effect of ProvGRP for attack investigation, two attack investigation methods are selected for analysis, which, respectively, used the above two public datasets. ATLAS [1] provides the ATLAS dataset and uses the dataset for the experimental evaluation. Although this work optimized the provenance graph, the average reduction factor is 5.21X. Since the primary goal of this work is to perform attack investigations, its graph optimization method is relatively simple and does not consider context information. The LogKernel [23] method performs experimental evaluation on the CADETS dataset. This work only partitions the long-running process nodes and does not reduce the graphs. In our experiments, we replaced the original graphs used as inputs in the two methods with the graphs reduced by ProvGRP, and compared the changes in runtime and their impact on the results. Table 3 shows the comparison results. When utilizing the reduced graphs G_{RA} and G_{RC}, the runtime of the attack investigation is reduced by 34.4% (ATLAS) and 61.0% (LogKernel), respectively.

Table 3. The changes of the above three indicators and running time after using the reduced graphs.

Methods	Graphs	Time (h:m:s)	Precision	Recall	F1-Score
ATLAS	Original G_{OA}	1:10:37	99.88%	99.89%	99.88%
	Reduction G_{RA}	0:46:21	99.73%	99.68%	99.70%
LogKernel	Original G_{OC}	1:18:08	3\(3 + 0)	3\(3 + 0)	1
	Reduction G_{RC}	0:30:29	3\(3 + 0)	3\(3 + 0)	1

Precision, recall, and F1-score are employed to evaluate the impact of using ProvGRP-reduced graphs on attack investigation results. As can be seen in Table 3, when using the reduced G_{RA}, the precision of ATLAS reached 99.73%, the recall reached 99.68%, and the F1-score reached 99.70%. These metrics are slightly lower compared to using the original graph G_{OA}. A more in-depth analysis of the attack results shows that ATLAS is able to more accurately identify attacks that share processes with normal users when performing attack investigation in the reduced graphs G_{RA}. ProvGRP can help identify this type of attack more accurately by removing false dependencies caused by long-running processes. This ensures that the origin graphs describing attacks are devoid of unrelated normal behavior instances. As a result, the number of false positives (FP), i.e., the misclassification of normal events as attack events, is reduced. However, some attack information was lost during the reduction process, leading to attack-related events being incorrectly classified as normal

events. By utilizing the causal relationships between events to construct attack scenarios, these false negatives (FN) can be associated with the corresponding attack scenarios.

Overall, the evaluation results demonstrate that the use of ProvGRP-reduced graphs significantly reduces the runtime of attack investigations while maintaining high precision. Although a slight decrease in recall and F1-score was observed, the improved representativeness of the obtained sequence and the ability to associate false negatives with attack scenarios compensate for this loss. Thus, ProvGRP effectively promotes the efficiency and accuracy of attack investigations.

5.6. Running Time Performance

The time consumption of ProvGRP is measured on two publicly available datasets. The size of these two datasets is comparable to real-world data. The runtime overhead of ProvGRP consists of three phases. The first phase is to construct the audit logs as provenance graphs, the second phase is provenance graphs partition, and the third phase is behavior-unrelated event elimination and semantic extraction. We perform the experiments on a server with an Intel(R) Xeon(R) Silver 4215R CPU (with 8 cores and 3.20 GHz of speed each) and 256 GB of memory running on Ubuntu 18.04.5 LTS.

In this setting, the processing speed of constructing provenance graphs from ATLAS and CADETS datasets are on average 169 MB/min and 483 MB/min, respectively. The runtime overhead of provenance graph partitioning was measured on both datasets, where our approach was able to handle an average of 38 and 45 long-running nodes per second, respectively. Finally, we analyze each provenance subgraph to remove events that are not related to behaviors. Although the removal was performed in a circular iteration, the size of the graphs decreased and the number of generated paths decreased as the iteration progressed. As a result, the time overhead for this phase was not as high as initially anticipated. Specifically, it took 8 min and 35 s and 12 min and 19 s to process the graphs generated by the two datasets, respectively.

From the reading of log data to construct the provenance graphs to the generation of the reduced graphs after semantic extraction, the total time cost by ProvGRP was 17 min and 43 s and 25 min and 39 s, respectively. These results are close to or even less than the data preprocessing time in multiple attack investigation methods, indicating that the time overhead of our method is acceptable. It will not significantly impact the time efficiency of attack investigations.

6. Related Work

Data Reduction. Various approaches have been proposed to address the challenges posed by large-scale provenance graphs in provenance analysis. Prior work has focused on targeted reduction methods based on the characteristics of redundant data. One category of approaches, including LogGC [24], FD-SD [25], KCAL [26], and Nodemerge [12], adopts a lossy reduction strategy by removing logs based on predefined patterns. While these methods have demonstrated effectiveness through case experiments, their reliance on predefined patterns may limit their adaptability to diverse scenarios, raising concerns about the completeness and correctness of results. In contrast, other reduction techniques such as CPR [13], DPR [27], and ProTracer [12] take a more selective approach, retaining only those system events deemed essential for constructing an accurate information flow graph. Similarly, methods like PCAR [13], DFA [27], and LogApprox [15] explore the bounded approximation of audit logs, accepting a controlled loss of accuracy to gain improved space efficiency. These techniques have shown promise in enhancing the efficiency and accuracy of forensic analysis, especially in scenarios where specific schemes and templates can be applied effectively. It is worth noting that lossless compression methods, typified by SEAL [14], prioritize the preservation of all information for detailed causal analysis. SEAL achieves this by generating a dependency graph from system logs and subsequently losslessly compressing the graph structure and attributes, including timestamps. This approach ensures accurate query results while maintaining efficiency in query processing.

Although these methods have introduced innovative ideas for log reduction and achieved good results, they overlook the erroneous dependency relationships caused by the coarse-grained nature of logs. The erroneous dependency relationships are also a significant factor leading to the generation of exceptionally large and complex provenance graphs. Moreover, the method is unable to identify redundant patterns not defined by schemes and templates. In fact, this paper's approach demonstrates that leveraging context information can more accurately and efficiently reduce audit data.

Attack Investigation. Sophisticated attacks disguise their behavior to evade the monitoring of intrusion detection systems when infiltrating information systems [28]. Provenance analysis is commonly used to investigate attacks and uncover hidden attack behaviors at the system layer. Nodoze [18] proposes attack investigation methods based on statistical characteristics. They prioritize abnormal events and causal dependencies, considering frequency and topological metrics. However, Priotracker focuses on individual event anomalies, while Nodoze considers anomalies across event chains and employs statistical low-frequency path mining. The statistical low-frequency path mining method is proposed to solve the dependence explosion problem, so as to restore the subgraph of the traceability data corresponding to the alarm generation more accurately. However, the IP address of abnormal transmission cannot be accurately located, and this method based on statistics may lead to unstable results.

Other approaches, Holmes [7] and RapSheet [29], treat multi-stage attacks as chains of causal events that conform to the TTP specification. WATSON [30] uses context information based on knowledge graph of system audit logs to realize semantic inference, expresses different behavior semantics through vectors, and uses semantically similar behaviors for clustering. The results show that benign and malicious behaviors can be accurately abstracted. OmegaLog [2] can accurately coordinate application events with system layer access by identifying and simulating application layer log behavior. Then, it intercepts application runtime log activities and migrates these events to the system layer traceback graph, so that investigators can more accurately infer the nature of attacks.

Hercule [31], Tiresias [32], Attack2vec [33], ATLAS [1], and IDERES [34] use machine learning techniques to model attack behaviors. Hercule uses community detection algorithms to correlate attack events, identifying clear behavioral divisions between threat events and normal events. Tiresias and Attack2vec are limited to identifying and reporting attack events within a single log. ATLAS aims to locate attacks through sequence learning and reconstructing attack paths based on known attack features.

Overall, these previous works contribute valuable insights and techniques to attack investigation and data reduction in provenance analysis, but each has its limitations and focuses on specific aspects of the problem.

7. Discussion

7.1. Limitation of ProvGRP

Although ProvGRP has achieved good results in data reduction, the method still has limitations. Our approach can identify more redundant events using contextual information, but it may produce errors when dealing with lengthy and intricate information paths. The extended length of information paths may accumulate minor differences, leading to the risk of the method incorrectly categorizing similar information paths as dissimilar. Another limitation is that ProvGRP cannot accurately partition two behaviors into different subgraphs when they perform operations using the same process almost simultaneously. If both of these behaviors are normal, they will not affect the subsequent investigation of the attack. However, if one of them is an attack behavior, it may impact the assessment of that behavior. Fortunately, the probability of this scenario occurring in practice is very low. In addition, ProvGRP has effectively divided most behaviors into subgraphs, a practice that is often overlooked in current methods for reducing audit data and investigating attacks. The last obvious limitation is that ProvGRP consumes more computational resources than

previous methods based on predefined templates. This level of increased complexity is acceptable compared to the efficiency of facilitating attack investigation.

7.2. Security Analysis of ProvGRP

ProvGRP is generally robust to attacks including APT attacks. Attackers typically evade detection through spoofing or camouflage techniques, such as spoofing the IP address, modifying the time information, double extension on file names, encapsulation techniques, etc. These modifications do pose significant challenges to intrusion detection systems, especially rule-based and matching IOC approaches. To address this challenge, attack investigation methods are widely used to perform a comprehensive causal analysis on audit data. Causal analysis is effective in identifying attacks because even if the attacker conceals or alters some of their indicators of compromise (IOCs) at certain stages, the behavioral patterns and objectives displayed throughout the attack process are markedly different from normal behavior. ProvGRP, by generating information paths based on contextual information, aids in recognizing similar behavior patterns, mitigating the impact of deceptive techniques that alter information at various steps.

APT attacks typically change their behavior gradually to evade detection by template-based and complex attack detection methods. However, this alteration does not significantly impact the ProvGRP outcomes. Our approach does not require matching predefined behavioral pattern templates, which means it does not need constant updates as attackers change their strategies and behaviors. ProvGRP generates information paths that describe all the paths that the information transfers pass through in the same attack instance. These paths share similar flow patterns that describe this attack example. We identify comparable information pathways and combine them to achieve efficient data reduction. Thus, ProvGRP is capable of effectively handling APT attacks that frequently alter their behavior.

8. Conclusions

Existing audit data reduction methods generally face the following two limitations: (1) false dependencies caused by the coarse-grained nature of logs, and (2) the information flow patterns implicit in the deep causal relationships and contextual information of system events are not taken into account. In order to solve the above problems, this paper proposes ProvGRP, a novel context-aware provenance graph reduction and partition approach that partitions provenance graphs into subgraphs describing different behaviors and eliminates redundant and behavior-unrelated events. It proposes three comprehensive features to determine whether the system events belong to the same event, so as to realize the segmentation of the provenance graph to remove the wrong dependencies. Subsequently, it uses information paths containing rich context information to represent information flow patterns, and removes redundant system events by identifying similar information paths. Experimental results show that ProvGRP can effectively reduce audit data while retaining key information of attacks. Furthermore, ProvGRP outperforms state-of-the-art data reduction methods and reduces the runtime of attack investigation methods.

Due to the wide use of deep learning, it is possible to use deep learning models to automatically learn information flow patterns. Therefore, exploring the use of deep learning models to realize information path merging will be one of the future works. In addition, we plan to explore new methods for segmenting graphs, such as analyzing the underlying logic of audit logs to discover deeper features.

Author Contributions: Conceptualization, J.L. (Jiawei Li), R.Z. and J.L. (Jianyi Liu); Methodology, J.L. (Jiawei Li); Software, J.L. (Jiawei Li); Validation, J.L. (Jiawei Li); Writing—original draft, J.L. (Jiawei Li); Writing—review & editing, J.L. (Jiawei Li), R.Z. and J.L. (Jianyi Liu); Supervision, R.Z. and J.L. (Jianyi Liu); Funding acquisition, R.Z. and J.L. (Jianyi Liu). All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by the National Natural Science Foundation of China under Grant U21B2020 and Grant U1936216, and the Fundamental Research Funds for the Central Universities (Beijing university of posts and telecommunications) for Action Plan under Grant 2021XD-A11-3.

Data Availability Statement: The data DAPRA CADETS and ATALS supporting this paper are from previously reported studies and datasets, which have been cited in this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alsaheel, A.; Nan, Y.; Ma, S.; Yu, L.; Walkup, G.; Celik, Z.B.; Zhang, X.; Xu, D. ATLAS: A Sequence-based Learning Approach for Attack Investigation. In Proceedings of the 30th USENIX Security Symposium, Vancouver, BC, Canada, 11–13 August 2021; pp. 3005–3022.
2. Hassan, W.U.; Noureddine, M.A.; Datta, P.; Bates, A. OmegaLog: High-Fidelity Attack Investigation via Transparent Multi-layer Log Analysis. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 23–26 February 2020.
3. Gao, P.; Xiao, X.; Li, Z.; Xu, F.; Kulkarni, S.R.; Mittal, P. AIQL: Enabling Efficient Attack Investigation from System Monitoring Data. In Proceedings of the 2018 USENIX Annual Technical Conference (USENIX ATC 18), Boston, MA, USA, 11–13 July 2018; pp. 113–126.
4. Milajerdi, S.M.; Eshete, B.; Gjomemo, R.; Venkatakrishnan, V.N. POIROT: Aligning Attack Behavior with Kernel Audit Records for Cyber Threat Hunting. In Proceedings of the ACM Conference on Computer and Communications Security, New York, NY, USA, 9–13 December 2019; pp. 1795–1812.
5. Kwon, Y.; Wang, F.; Wang, W.; Lee, K.H.; Lee, W.C.; Ma, S.; Zhang, X.; Xu, D.; Jha, S.; Ciocarlie, G.; et al. MCI: Modeling-based Causality Inference in Audit Logging for Attack Investigation. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 18–21 February 2018; Volume 2, p. 4.
6. Zhao, J.; Yan, Q.; Liu, X.; Li, B.; Zuo, G. Cyber Threat Intelligence Modeling Based on Heterogeneous Graph Convolutional Network. In Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020), San Sebastian, Spain, 14–16 October 2020; pp. 241–256.
7. Milajerdi, S.M.; Gjomemo, R.; Eshete, B.; Sekar, R.; Venkatakrishnan, V.N. Holmes: Real-time apt detection through correlation of suspicious information flows. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019.
8. Hossain, M.N.; Sheikhi, S.; Sekar, R. Combating dependence explosion in forensic analysis using alternative tag propagation semantics. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–21 May 2020; pp. 1139–1155.
9. Zhu, T.; Wang, J.; Ruan, L.; Xiong, C.; Yu, J.; Li, Y.; Chen, Y.; Lv, M.; Chen, T. General, Efficient, and Real-time Data Compaction Strategy for APT Forensic Analysis. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 3312–3325. [CrossRef]
10. Yang, R.; Chen, X.; Xu, H.; Cheng, Y.; Xiong, C.; Ruan, L.; Kavousi, M.; Li, Z.; Xu, L.; Chen, Y. RATScope: Recording and Reconstructing Missing RAT Semantic Behaviors for Forensic Analysis on Windows. *IEEE Trans. Dependable Secur. Comput.* **2020**, *19*, 1621–1638. [CrossRef]
11. Lee, K.H.; Zhang, X.; Xu, D. High accuracy attack provenance via binary-based execution partition. In Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, CA, USA, 24–27 February 2013.
12. Tang, Y.; Li, D.; Li, Z.; Zhang, M.; Jee, K.; Xiao, X.; Wu, Z.; Rhee, J.; Xu, F.; Li, Q. Nodemerge: Template based efficient data reduction for big-data causality analysis. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1324–1337.
13. Xu, Z.; Wu, Z.; Li, Z.; Jee, K.; Rhee, J.; Xiao, X.; Xu, F.; Wang, H.; Jiang, G. High fidelity data reduction for big data security dependency analyses. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24 October 2016.
14. Fei, P.; Li, Z.; Wang, Z.; Yu, X.; Li, D.; Jee, K. Seal: Storage-efficient causality analysis on enterprise logs with query-friendly compression. In Proceedings of the USENIX Security Symposium, Online, 11–13 August 2021.
15. Michael, N.; Mink, J.; Liu, J.; Gaur, S.; Hassan, W.U.; Bates, A. On the forensic validity of approximated audit logs. In Proceedings of the Annual Computer Security Applications Conference, Austin, TX, USA, 7–11 December 2020; pp. 189–202.
16. McInnes, L.; Healy, J. Accelerated Hierarchical Density Based Clustering. In Proceedings of the 2017 IEEE International Conference on Data Mining Workshops (ICDMW), New Orleans, LA, USA, 18–21 November 2017; IEEE: Piscateville, NJ, USA, 2017; pp. 33–42.
17. Blog, Y. The North Korean Kimsuky APT Keeps Threatening South Korea Evolving Its TTPs. YOROI TINXTA CYBER. 2020. Available online: https://blog.yoroi.company/research/the-north-korean-kimsuky-apt-keeps-threatening-south-korea-evolving-its-ttps/ (accessed on 13 May 2023).
18. Hassan, W.U.; Guo, S.; Li, D.; Chen, Z.; Jee, K.; Li, Z.; Bates, A. Nodoze: Combatting threat alert fatigue with automated provenance triage. In Proceedings of the Network and Distributed Systems Security Symposium, San Diego, CA, USA, 24–27 February 2019.

19. Hossain, M.N.; Milajerdi, S.; Wang, J.; Eshete, B.; Gjomemo, R.; Sekar, R.; Stoller, S.; Venkatakrishnan, V.N. Sleuth: Real-time attack scenario reconstruction from cots audit data. In Proceedings of the USENIX Security Symposium, Vancouver, BC, Canada, 16–18 August 2017.
20. Liu, Y.; Zhang, M.; Li, D.; Jee, K.; Li, Z.; Wu, Z.; Rhee, J.; Mittal, P. Towards a timely causality analysis for enterprise security. In Proceedings of the NDSS, San Diego, CA, USA, 18–21 February 2018.
21. Leland, M.; John, H.; Steve, A. How HDBSCAN Works. Available online: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html (accessed on 5 July 2023).
22. Torrey, J. Transparent Computing Engagement 3 Data Release. 2020. Available online: https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md (accessed on 13 May 2023).
23. Li, J.; Zhang, R.; Liu, J.; Liu, G. LogKernel: A Threat Hunting Approach Based on Behaviour Provenance Graph and Graph Kernel Clustering. *Secur. Commun. Netw.* **2022**, *2022*, 4577141. [CrossRef]
24. Lee, K.H.; Zhang, X.; Xu, D. Loggc: Garbage collecting audit log. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4 November 2013.
25. Hossain, M.N.; Wang, J.; Weisse, O.; Sekar, R.; Genkin, D.; He, B.; Stoller, S.D.; Fang, G.; Piessens, F.; Downing, E.; et al. Dependence-preserving data compaction for scalable forensic analysis. In Proceedings of the 27th {USENIX} Security Symposium ({USENIX} Security 18), Baltimore, MD, USA, 15–17 August 2018; pp. 1723–1740.
26. Ma, S.; Zhai, J.; Kwon, Y.; Lee, K.H.; Zhang, X.; Ciocarlie, G.; Gehani, A.; Yegneswaran, V.; Xu, D.; Jha, S. Kernel-supported cost-effective audit logging for causality tracking. In Proceedings of the 2018 USENIX Annual Technical Conference (USENIX ATC 18), Boston, MA, USA, 11–13 July 2018.
27. Hassan, W.U.; Aguse, L.; Aguse, N.; Bates, A.; Moyer, T. Towards scalable cluster auditing through grammatical inference over provenance graphs. In Proceedings of the Network and Distributed Systems Security Symposium, San Diego, CA, USA, 18–21 February 2018.
28. Saračević, M.; Selimi, A.; Plojović, Š. Some specific examples of attacks on information systems and smart cities applications. In *Cybersecurity and Secure Information Systems: Challenges and Solutions in Smart Environments*; Springer International Publishing: Cham, Switzerland, 2019; pp. 205–226.
29. Hassan, W.U.; Bates, A.; Marino, D. Tactical provenance analysis for endpoint detection and response systems. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–20 May 2020; IEEE: Piscateville, NJ, USA, 2020; pp. 1172–1189.
30. Zeng, J.; Chua, Z.L.; Chen, Y.; Ji, K.; Liang, Z.; Mao, J. Watson: Abstracting behaviors from audit logs via aggregation of contextual semantics. In Proceedings of the 28th Annual Network and Distributed System Security Symposium, NDSS, Online, 21–25 February 2021.
31. Pei, K.; Gu, Z.; Saltaformaggio, B.; Ma, S.; Wang, F.; Zhang, Z.; Si, L.; Zhang, X.; Xu, D. Hercule: Attack story reconstruction via community discovery on correlated log graph. In Proceedings of the 32nd Annual Conference on Computer Security Applications, Los Angeles, CA, USA, 5–8 December 2016; pp. 583–595.
32. Shen, Y.; Mariconti, E.; Vervier, P.A.; Stringhini, G. Tiresias: Predicting security events through deep learning. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 592–605.
33. Shen, Y.; Stringhini, G. Attack2vec: Leveraging temporal word embeddings to understand the evolution of cyberattacks. In Proceedings of the 28th {USENIX} Security Symposium ({USENIX} Security 19), Santa Clara, CA, USA, 14–16 August 2019; pp. 905–921.
34. Rose, J.R.; Swann, M.; Grammatikakis, K.P.; Koufos, I.; Bendiab, G.; Shiaeles, S.; Kolokotronis, N. IDERES: Intrusion detection and response system using machine learning and attack graphs. *J. Syst. Archit.* **2022**, *131*, 102722. [CrossRef]