Provenance-based Intrusion Detection Systems: A Survey

MICHAEL ZIPPERLE, University of New South Wales, Canberra, Australia and Cyber Security Cooperative Research Centre, Australia
FLORIAN GOTTWALT, University of New South Wales, Canberra, Australia
ELIZABETH CHANG, Griffith University, Gold Coast, Australia
THARAM DILLON, La Trobe University, Melbourne, Australia

Traditional Intrusion Detection Systems (IDS) cannot cope with the increasing number and sophistication of cyberattacks such as Advanced Persistent Threats (APT). Due to their high false-positive rate and the required effort of security experts to validate them, incidents can remain undetected for up to several months. As a result, enterprises suffer from data loss and severe financial damage. Recent research explored data provenance for Host-based Intrusion Detection Systems (HIDS) as one promising data source to tackle this issue. Data provenance represents information flows between system entities as Direct Acyclic Graph (DAG). Provenance-based Intrusion Detection Systems (PIDS) utilize data provenance to enhance the detection performance of intrusions and reduce false-alarm rates compared to traditional IDS. This survey demonstrates the potential of PIDS by providing a detailed evaluation of recent research in the field, proposing a novel taxonomy for PIDS, discussing current issues and potential future research directions. This survey aims to help and motivate researchers to get started in the field of PIDS by tackling issues of data collection, graph summarization, intrusion detection, and developing real-world benchmark datasets.

CCS Concepts: • **Security and privacy → Intrusion detection systems**; • **Information systems** → *Graph-based database models.*

Additional Key Words and Phrases: intrusion detection, data provenance, graph summarization, machine learning, benchmark dataset, survey

1 INTRODUCTION

In recent years, the number of cyberattacks has increased significantly, and sophisticated attacks such as Advanced Persistent Threats (APT) make it more challenging than ever for defenders to protect enterprises. Attackers constantly change their attack patterns, seek new intrusion points, and use obfuscation methods to remain undetected.

To counter this, defenders must constantly adapt to detect and respond to malicious behaviors in a timely fashion. Enterprises deploy Intrusion Detection Systems (IDS) to detect potential security incidents and security experts analyze generated alerts for their validity. However, the limitations of current IDS often lead to a high number of false alarms overwhelming security experts with alerts and sometimes leading to intrusions being undetected up to multiple months later[28].

Authors' addresses: Michael Zipperle, m.zipperle@adfa.edu.au, University of New South Wales, Canberra, Northcott Dr, Canberra, ACT, Australia, 2612 and Cyber Security Cooperative Research Centre, Australia; Florian Gottwalt, f.gottwalt@adfa.edu.au, University of New South Wales, Canberra, Northcott Dr, Canberra, ACT, Australia, 2612; Elizabeth Chang, e.chang@griffith.edu.au, Griffith University, Gold Coast, Parklands Dr, Gold Coast, QLD, Australia, 4222; Tharam Dillon, La Trobe University, Melbourne, Plenty Rd &, Kingsbury Dr, Melbourne, VIC, Australia, 3086.

0360-0300/2022/6-ART $15.00
https://doi.org/10.1145/3539605

One example that illustrates the difficulty of detecting such sophisticated threats is the recent SolarWinds attack[13]. The nation-state attackers got access to SolarWinds's software build system, injected malicious code to a significant software update, and the update rollout distributed the malicious code to over 18,000 SolarWinds's customers' systems. Records state that over one thousand hackers must have been involved in the attack, and approximately 40 companies, including Fortune 500 companies and multiple US government agencies, have been compromised[56]. The financial impact of the attack is extreme; BITSIGHT estimates the cost of incident response and forensic analysis at $90,000,000 in addition to the financial damage from data exfiltration[11].

Cyber-security research has focused on increasing the detection performance and reducing the false alarms of traditional IDS to reduce the impact of APT and protect enterprises from financial impact and data loss. Recent IDS research has investigated data provenance methods which are very promising, demonstrating an improved detection performance and a reduced number of false alarms, and hence a reduction of cognitive load for security experts[4, 5, 8, 9, 25, 29–31, 43, 45–47, 49, 50, 52, 66, 69, 89, 90, 94, 106, 108, 109, 111, 114]. Data provenance represents the system execution as a Direct Acyclic Graph (DAG) by describing the information flow between subjects (e.g. processes, threads) and objects (e.g. files, network sockets). Despite recent progress of Provenance-based Intrusion Detection Systems (PIDS), current provenance approaches face the following challenges: 1) Data provenance collected in large enterprises can grow rapidly up to multiple Terabytes (TBs) 2) intrusion detection approaches rely on a threshold that is manually tuned for a specific scenario, 3) only a limited amount of contextual information has been explored to enhance the detection performance, and 4) existing benchmark datasets are either outdated or do not include real-world benign system events.

While there are numerous surveys for traditional Network-based Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS), only limited surveys review and discuss the shortcomings of PIDS. Li et al. [67] introduced the first survey introducing a framework for PIDS that consists of a data collection module, storage management, and threat detection module. Our survey distinguishes from theirs by first introducing a taxonomy of PIDS, second providing a detailed discussion on the need for real-world benchmark datasets, and third, by adding a more comprehensive literature study for the data collection, storage management, and threat detection module.

The rest of this survey is structured as follows: Section 2 provides background knowledge about IDS, data provenance and introduces a taxonomy of PIDS, which is used to categorize the literature in the remainder of the survey. Section 3 reviews related surveys on IDS and presents our survey methodology. Section 4, Section 5, and Section 6 review latest literature on data collection, graph summarization, and intrusion detection and discuss issues and challenges. Section 7 reviews benchmark datasets for PIDS, discusses issues and provides future directions to generate real-world benchmark datasets. Conclusions of this survey are given in Section 8.

2 BACKGROUND

This section provides preliminary knowledge about IDS, data provenance, and introduces a taxonomy for PIDS.

2.1 Intrusion Detection Systems (IDS)

IDS analyze different data sources and they can be deployed at network or host level, use signature- or anomaly-based intrusion detection techniques, and analyze data online or offline to detect cyberattacks. Figure 1 shows a taxonomy of IDS types and data sources utilized.

NIDS identify intrusions by monitoring the network traffic of multiple hosts on the network layer and can be easily integrated into an existing network's infrastructure. However, NIDS cannot analyze encrypted network traffic, and only external intrusions can be detected.

HIDS identify intrusions by monitoring the host's file system, system calls, and network events and usually outperform NIDS due to the availability of more fine-grained event logs. However, HIDS needs to be deployed on

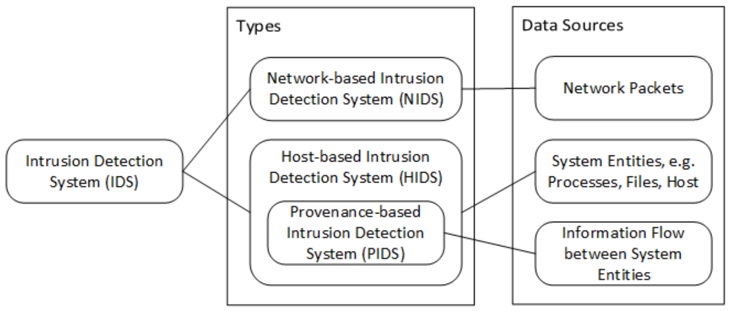

Types
Data Sources
Intrusion Detection System (IDS)
Network-based Intrusion Detection System (NIDS)
Host-based Intrusion Detection System (HIDS)
Provenance-based Intrusion Detection System (PIDS)
Network Packets
System Entities, e.g. Processes, Files, Host
Information Flow between System Entities

Fig. 1. Taxonomy of IDS's types and data sources

every host, can only monitor the host where it is deployed, and generate a massive amount of event logs. Many enterprises deploy NIDS and HIDS to take advantage of both[83].

Signature-based detection techniques aim to detect signatures created by security experts based on knowledge of malicious behavior from the past. Signature-based detection techniques work well for known cyberattacks but cannot detect unknown attacks and the detection performance depends on an up-to-date signature dataset[3].

Anomaly detection techniques aim to detect deviations in normal behavior to detect unknown attacks. Anomaly detectors learn historical benign behavior and classify new behavior based on their deviation from the learned normal behavior[3]. Anomaly detectors can detect unknown attacks such as zero-day attacks but often suffer from a high false alarm rate because it is challenging to distinguish between unknown benign behavior, unknown malicious behavior, and system errors. Also, system evolution causes difficulties in characterizing normal behavior, which may decrease the detection performance[43].

Online IDS monitor real-time data to detect intrusions, while offline IDS analyze historical data accessed from a persistent data source such as a database. Since intrusion detection is time-critical, online IDS are mostly used and offline IDS are utilized to support forensic analysis[3].

Due to the high number of false alarms of traditional IDS[20] and the failure of most commercial HIDS to detect APT[60], recent research has focused on automatically reducing the number of false alarms by using context for the intrusion detection process. One potential context is the data provenance of system entities for HIDS.

2.2 Data Provenance

Data provenance was initially proposed to determine the data flow and data origin in traditional databases. However here data provenance and related terms are defined for use in IDS.

Definition 2.1 (Data Provenance). Data provenance refers to a record trail representing an event's origin and explaining how and why it got to the current state.

Definition 2.2 (Provenance Graph). Data provenance can be represented as a DAG, namely a provenance graph. In this graph, system entities are represented as nodes N and system operations as directed edges E. A provenance graph G can be summarized as: $G = (N, E)$.

Figure 2 illustrates a provenance graph that shows a phishing attack example: A user receives an email, opens the attachment, the Microsoft Word file contains a malicious payload that creates a new command-line process which reads sensitive information and exfiltrates it to an external FTP server.

Definition 2.3 (Node). A node n in a provenance graph G represents a system entity such as a process, file, or host. The available system entities strongly depend on the underlying Operating System (OS). A node has a

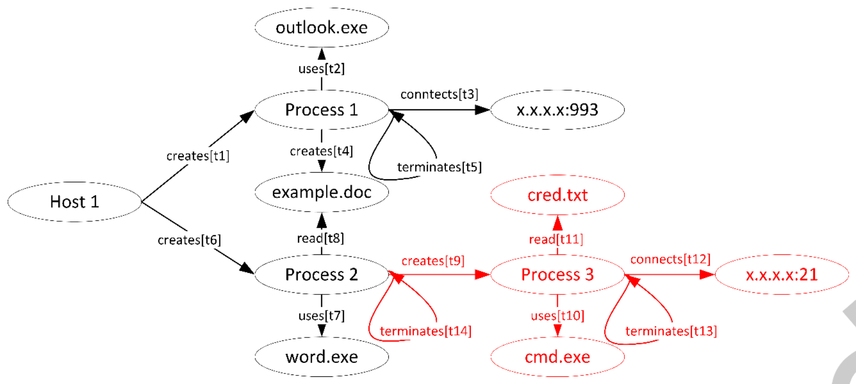


Fig. 2. Provenance Graph of a Phishing Attack Example

unique identifier, that may contain properties, and can be represented as $n = \{id, properties*\}$. The nodes N of a provenance graph G are represented as: $N = \{n_1, n_2, ...n_n\}$.

Definition 2.4 (Edge). An edge e in a provenance graph G represents the events between two nodes. An edge has a unique identifier, id of source and target node, timestamp and may have additional properties. It is represented as $e = \{id, source\_id, target\_id, timestamp, properties*\}$. The edges E of a provenance graph G are represented as: $E = \{e_1, e_2, ...e_n\}$. Timestamps are mandatory to keep information about the sequence of events.

Definition 2.5 (Backward Tracing). Backward tracing determines the record trace for the node of interest n in a provenance graph G. The resulting subgraph *trace_graph* contains all nodes and edges leading to node n.

Algorithm 1: Backward Tracing for node n in a Provenance Graph G

```
backward-tracing($G, n$)
    $trace\_graph = (nodes = \{\}, edges = \{\})$
    for $edge$ in $G.edges$ do
        if $edge.target\_id$ is $node.id$ then
            $node = G.nodes[id$ is $edge.source\_id]$
            $trace\_graph.nodes.append(node)$
            $trace\_graph.edges.append(edge)$
            $trace\_graph.extend($backward-tracing$(G, node))$
    return $trace\_graph$
```

Definition 2.6 (Forward Tracing). Forward tracing discovers the influence of the node of interest n in a provenance graph G. The resulting subgraph *trace_graph* contains all nodes and edges influenced by node n.

Definition 2.7 (Scenario Graph). A scenario graph G_s is a subgraph of a given provenance graph G that contains only nodes and edges causally dependent on a node n of interest. A scenario graph can be derived from the provenance graph G by applying backward- and forward tracing from node n.

A simple version of the backward tracing algorithm is shown in Algorithm 1, the forward tracing algorithm works accordingly. For forensic analysis, backward tracing determines the record trail of a potential security alarm and unveils the initial intrusion point. Forward tracing discovers system entities affected by the intrusion. The scenario graph helps security experts to make a timely forensic analysis of a security incident.

Definition 2.8 (Causality Analysis). The causality analysis determines whether two given nodes n_1 and n_2 in the provenance graph G are causally dependent.

Causality analysis can determine if two security alarms are correlated and thus, helps to detect multi-stage attacks. The algorithm for causality analysis is illustrated in Algorithm 2.

Algorithm 2: Causality Analysis of nodes n_1 and n_2 in Provenance Graph G

```
causality-analysis($G, n_1, n_2$)
    $trace\_graph = (nodes = \{\}, edges = \{\})$
    $trace\_graph.extend$(backward-tracing ($G, n_1$))
    $trace\_graph.extend$(forward-tracing ($G, n_1$))
    if $n_2$ in $trace\_graph.nodes$ then
        return True
    else
        return False
```

2.3 Provenance-based Intrusion Detection Systems (PIDS)

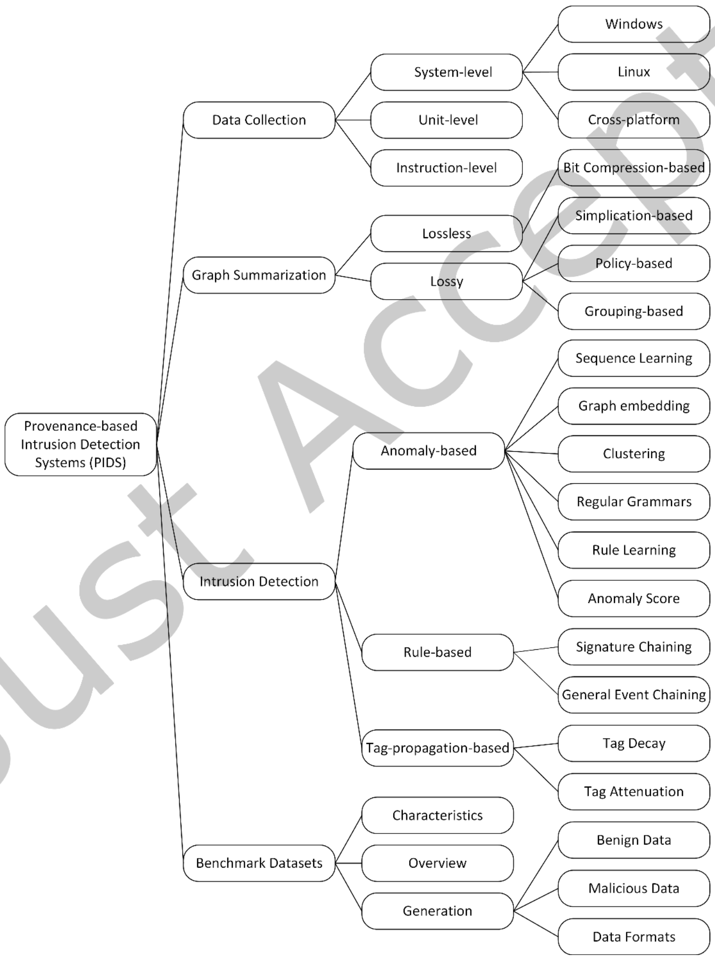


Fig. 3. Taxonomy of PIDS

Data provenance adds semantic information to flat system logs which have been used in traditional HIDS. These information help to improve the detection performance and reduce the false alarms as recent research has demonstrated[4, 5, 8, 9, 25, 29–31, 43, 45–47, 49, 50, 52, 66, 69, 89, 90, 94, 106, 108, 109, 111, 114]. We define a HIDS utilizing data provenance as follow:

Definition 2.9 (Provenance-based Intrusion Detection Systems (PIDS)). Provenance-based Intrusion Detection Systems (PIDS) utilizes data provenance to detect intrusion by not only analysing system entities and their properties but also analysing the causalities and information flow of system entities in a provenance graph.

Figure 3 shows a taxonomy of PIDS that structures the system in four modules, namely data collection, graph summarization, intrusion detection, and benchmark datasets. The remainder of this survey is structured according to these modules.

3 RELATED IDS SURVEYS

Survey	**Year**	**IDS Types**	**Data Collection**	**Graph Summarization**	**Benchmark Datasets**
Axelsson[3]	2000	HIDS, NIDS	N/A	N/A	Yes
Vasilomanolakis[103]	2015	CIDS	N/A	N/A	Yes
Han[44]	2018	PIDS	N/A	N/A	N/A
Bridges[12]	2019	HIDS	Yes	N/A	Yes
Lakshminarayana[63]	2019	NIDS	N/A	N/A	N/A
Liu[70]	2019	HIDS	N/A	Yes	Yes
Li[67]	2021	PIDS	Yes	Yes	N/A
This survey	2021	PIDS	Yes	Yes	Yes

Table 1. Related IDS Surveys

Several surveys on IDS have been published in recent years, while only a few surveys focused on PIDS. Table 1 provides an overview of selected related IDS surveys, and the following paragraphs will give a summary.

Axelsson [3] created a taxonomy for NIDS and HIDS and grouped proposed approaches by their detection goal highlighting the detection goals for which there is a lack of approaches. Vasilomanolakis et al. [103] created a taxonomy of Collaborative Intrusion Detection Systems (CIDS) by classifying the communication architecture. Further, they defined requirements of Collaborative Intrusion Detection Systems (CIDS) which they used to evaluate existing approaches. Han et al. [44] is the first survey that discusses the opportunities and challenges of PIDSm but it does not provide a general framework nor a taxonomy of PIDS. Lakshminarayana et al. [63] reviewed the latest approaches for NIDS with the main focus on new technologies such as blockchain and deep learning. Liu et al. [70] discuss feature extraction methods, various data mining methods, and recent research trends of HIDS and analyzed the existing benchmark datasets highlighting their issues. Li et al. [67] introduce a framework for PIDS, which consists of the data collection, data management, and threat detection modules. Recent approaches for each module are evaluated and future research trends are derived. Their survey lacks a taxonomy of PIDS and a comprehensive discussion about benchmark datasets and their importance for future research.

Our current paper addresses the drawbacks of previous surveys by introducing a taxonomy for PIDS, providing a detailed literature review of approaches based on the taxonomy, and a comprehensive discussion of benchmark datasets with the focus on multiple-stage attacks such as APT.

3.1 Survey Methodology

We started our literature review by searching in popular databases such as Scopus[27], Web of Science[15], IEEE Xplore[54] and ScienceDirect[26]. We derived our search keywords from terms relevant to PIDS, these include *data provenance, provenance graph, apt, intrusion detection, threat detection, forensic analysis, causality analysis, anomaly detection*. We identified venues that have a high impact factor and have published many works in the area of PIDS, these include *ACM Computing Surveys*[1], *The Network and Distributed System Security Symposium*[55], *USENIX Security Symposium*[102] or *IEEE Symposium on Security and Privacy*[53]. Finally, we analyzed cross-references by using Connected Papers[16] and bibliometrix[10] to validate that we have not missed important articles.

Based on the chosen survey methodology, we selected a total of 58 articles to review within the scope of this survey. Figure 4 shows the distribution of the selected articles across the three modules. It indicates that research initially focused on data collection, then moved on to graph summarization and storage, and most recent research focuses on intrusion detection.

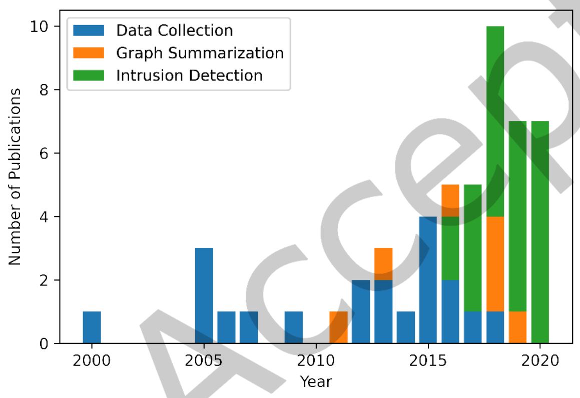


Fig. 4. Publication Distribution

4 DATA COLLECTION

The first step in a PIDS is the deployment of Data Provenance Capture (DPC) systems on the hosts to be analyzed. Data provenance can be collected at different granularity levels, namely system-level, unit-level, and instruction-level. The remainder of this section reviews DPC systems based on their granularity level.

4.1 System-level Data Provenance Capture (DPC) systems

The majority of DPC systems monitor system-level data provenance defined as follows:

Definition 4.1 (System-level data provenance). System-level data provenance describes the information flow between system entities by analyzing system calls (syscalls) in the user- or kernel space.

4.1.1 System-level Data Provenance Capture (DPC) systems for Windows. The Event Tracing for Windows (ETW) is Windows is a built-in device driver for logging kernel or application-defined events and consists of three main components: (1) the controller that starts and stops event tracing sessions; (2) the provider which creates events and publishes them in a session; (3) the consumer that consumes events from a session[77]. Sysmon logs

Name	Year	OS	Space	Granularity	Availability	Runtime Overhead
ETW[77]	2000	Windows	Kernel	System	Maintained	Low
Lineage File System[92]	2005	Linux	Kernel	System	Not available	N/A
DTrace[24]	2005	Windows, Linux, macOS	User/Kernel	System	Maintained/Open Source	N/A
PASSv1[85]	2006	Linux	User	System	Not available	10.50%
PASSv2[84]	2009	Linux	User	System	Not available	<23.1%
Panorama[113]	2007	Windows	User	Instruction	Not available	20x
SPADE[32]	2012	Windows, Linux, macOS	User	System	Maintained	53% (W), 10% (M), 5% (L)
Hi-Fi[91]	2012	Linux	Kernel	System	Not available	<6.2%
BEEP[64]	2013	Linux	Kernel	Unit	Not available	1.40%
LogGC[65]	2013	Linux	Kernel	Unit	Not available	<2.04
Sysmon[78]	2014	Windows	Kernel	System	Maintained	N/A
LPM[6]	2015	Linux	Kernel	System	Not available	<7.5%
Provmon[6]	2015	Linux	Kernel	System	Not available	<7.5%
DataTracker[97]	2015	Linux	User	Instruction	Open-Source	4-6x
PROV-Tracer[96]	2015	Linux	User	Instruction	Open-Source	5x
INSPECTOR[100]	2016	Linux	User	Instruction	Open-Source	2.50%
ProTracer[75]	2016	Linux	Kernel	Unit	Not available	<7%
RecProv[59]	2016	Linux	User	System	Not available	20%
MPI[74]	2017	Linux	User	Unit	Not available	<1%
RAIN[58]	2017	Linux	User/Kernel	Instruction	Not available	3.10%
CamFlow[88]	2017	Linux	Kernel	System	Maintained	N/A
LPROV[105]	2018	Linux	User/Kernel	Unit	Not available	7%

Table 2. Overview of Data Collection Approaches

security-related system activities by accessing and aggregating multiple log sources of ETW. It logs various events such as network connections, pipes, registry-, file-, and image operations, and a rule set defines the scope of events monitored[78].

4.1.2 System-level DPC systems for Linux. Lineage File System is one of the first DPC systems proposed to track, store, and query lineage information of files[92]. Linux has a built-in auditing system that monitors security and non-security-related events by intercepting syscalls in the user-space. It's shortcoming is the high runtime overhead, which can be as high as 43% [73, 75, 115]. RecProv addresses the runtime overhead issue by applying a record and replay method from software debugging to generate system-level data provenance. It has a secure data provenance store to protect the data integrity[59].

Early system-level DPC systems for Linux cannot monitor kernel-initiated actions and thus, could miss essential events for intrusion detection and forensic analysis. Hi-Fi is the first system-level DPC system that intercepts syscalls in the kernel-space and observes kernel-initiated actions to collect whole-system data provenance[91]. Provmon is a port of Hi-Fi that adds additional semantic information to system entities such as versions to files and remote IP addresses, and ports to network events. Hi-Fi and Provmon were designed for now outdated kernel releases and cannot run on current Linux systems. CamFlow, a provenance capture system in the kernel space, uses self-contained easily maintainable Linux Security Modules (LSM) and NetFilters to collect whole-system data provenance. It utilizes the latest kernel features to increase the efficiency to collect whole-system data provenance. CamFlow significantly reduces the runtime overhead compared to previous approaches[88].

4.1.3 Cross-platform system-level DPC systems. Data provenance contains OS specific types and properties of nodes and edges. As a result, data provenance from various Operating Systems (OSs) cannot be simply merged. To address this, multiple cross-platform system-level DPC systems have been proposed.

Dtrace is an early cross-platform system-level DPC systems developed by Sun Microsystems, which can be deployed on various OSs such as Windows, Linux, and MacOS[24, 107]. Nevertheless, Dtrace does not support the aggregation of heterogeneous system-level data provenance. Provenance-aware storage systems (PASS) address this through a provenance-aware filesystem that provides a disclosed provenance Application Programming Interface (API) to interface between the layers and naming conversions of various OSs[84, 85]. Another cross-platform system-level DPC system is Support for Provenance Auditing in Distributed Environments (SPADE) that aggregates heterogeneous system-level data provenance to support distributed debugging and causality analysis across multiple OSs[32].

A major problem of system-level DPC systems is that the resulting provenance graph can be too coarse, especially for long-running processes, leading to a dependency explosion problem[64, 74].

4.2 Unit-level Data Provenance Capture (DPC) systems

Unit-level DPC systems address the dependence explosion problem by partitioning long-running processes into units[64, 74, 75] and partitioning files into data units[65].

Definition 4.2 (Unit-level data provenance). Unit-level data provenance extends the system-level data provenance by describing more fine-grained information flows by splitting system entities into units.

Binary-based ExEcution Partition (BEEP) extends the Linux audit framework by dynamically partitioning long-running processes into autonomous execution segments, coined units. Long-running processes have the characteristics of being driven by external requests and dominated by event processing loops. BEEP detects these event processing loops and their causalities by reverse-engineering the application's binaries[64]. LogGC proposed in [65] extends BEEP by additionally partitioning files into logical data units and by pruning redundant events from the data provenance. LogGC uses a profiler to detect logical data units but human effort is still required to confirm the decision of the profiler. Both BEEP and LogGC have a high runtime overhead and suffer from a high space overhead as they partition every event processing loop, even mouse clicks, into units. To address this, ProTracer detects event processing loops by analyzing syscalls in the kernel space and only partitions long-running processes with critical actions, such as file writes and network connections, into units, reducing the space overhead to 1.28% of BEEP's on average.

Multiple Perspective attack Investigation (MPI) further reduces the space overhead by partitioning long-running processes based on user-defined tasks. A task indicates different perspectives of the user, such as a tab in a web browser. Tasks can be identified by adding annotations to the source code of an executable. To simplify the annotation process, the authors proposed a miner that helps to identify data structures and to add the right annotations. The evaluation shows, that MPI can reduce the space overhead of the provenance capture systems BEEP[64], HiFi[91], and ProTracer[75]. Previous DPC systems cannot trace libraries that are dynamically linked and executed at runtime. LPROV extends ProTracer to trace library calls and correlate them with syscalls. Consequently, LPROV can track data provenance of malicious library attacks and library vulnerability exploitation.

Other approaches have been proposed to tackle the dependency explosion problem, including applying a time window[62], tag propagation[50], or capturing more fine-grained data provenance for specific scenarios[78].

4.3 Instruction-level Data Provenance Capture (DPC) systems

Instruction-level DPC systems monitor high-fidelity information flow between system entities, thus delivering rich semantic information to enhance intrusion detection performance. Instruction-level data provenance is defined as followed:

Definition 4.3 (Instruction-level data provenance). Instruction-level data provenance describes the information flow between system entities with high-fidelity causalities derived from central processing unit (CPU) instructions.

An early instruction-level DPC systems is Panorama. It loads a potential malware into a test environment and executes it while the test engine monitors the fine-grained instructions of the executable[113]. Another instruction-level DPC system is DataTracker, which instruments a potential malware dynamically and applies taint analysis to create a provenance graph[97]. Panorama and DataTracker suffer from a high runtime overhead due to the heavy instrumentation process, and as a result, they can hardly be used in real-world scenarios. To address this, Inspector was proposed, which utilizes a parallel algorithm to monitor instruction-level data provenance of multi-threading applications using a Concurrent Provenance Graph (CPG). It also provides process-level isolation, MMU-assisted memory tracking, and Intel PT ISA extensions to increase efficiency[100]. While Panorama, DataTracker, and Inspector can collect high-fidelity system-level data provenance for a selected application, they cannot monitor whole-system data provenance and inter-process causalities, which is particularly important to detect sophisticated attacks, such as APTs.

PROV-Tracer is a whole-system reverse engineering tool that collects system-level data provenance and replays selected scenarios to monitor instruction-level data provenance. PROV-Tracer is built on top of PANDA[23], which leverages the QEMU emulator to support different architectures and thus, adds a high runtime overhead[96]. Refinable Attack INvestigation system (RAIN) records system-level data provenance and replays them on-demand to apply instruction-level Dynamic Information Flow Tracking (DIFT) to recover fine-grained causalities. It filters out unrelated processes by applying graph-based reachability analysis to reduce the number of processes that need to be replayed, resulting in significantly lower runtime overhead than that one of PROV-Tracer[58].

While instruction-level data provenance contains rich semantic information, the high runtime overhead is a major drawback that makes it challenging to apply in real-world scenarios. Researchers have tried to find a trade-off between system-level and instruction-level data provenance[58, 96]. However, in these approaches, security experts have to select when to apply instruction-level data provenance capture, and as a consequence, those approaches cannot be used for real-time intrusion detection.

4.4 Discussion and Challenges

While many DPC systems have been proposed in the literature, challenges arise regarding the runtime overhead, availability, fault tolerance, trustworthiness, and privacy.

4.4.1 Runtime Overhead. The runtime overhead is a major evaluation criterion of DPC systems. Table 2 shows that the data provenance's granularity is closely related to the runtime overhead of a DPC system. The finer the granularity of the data provenance, the higher the runtime overhead and impact on the overall system performance. Provenance capture systems which capture provenance data on instruction-level [6, 96, 97, 113] can pose a runtime overhead higher than 100%. Researchers tackled this problem by finding a trade-off between high-fidelity data provenance and low runtime overhead. E.g. RAIN[58] continuously captures system-level data provenance but can replay a scenario to apply instruction-level data provenance capture for further investigation.

The runtime overhead values in Table 2 can only be compared to a limited extent. First, researchers have used different system behavior as benchmarks for their evaluations[6]. Second, the experimental setup of the evaluation was significantly different, e.g. PASS was evaluated on a system with a single 500MHZ Intel Pentium 3 CPU, whereas Linux Provenance Modules (LPM) was evaluated on a system with a dual Intel Xeon CPU[88]. Third, the number of applications running on a system has increased significantly over the years. Therefore, provenance capture systems that were proposed ten years ago had significantly fewer syscalls to monitor.

4.4.2 Availability. Most DPC systems monitor syscalls in the kernel-space and, therefore, are implemented as kernel modules. Due to rapid changes in the kernel versions, researchers could not keep up with maintaining their source code. As a result, many DPC systems proposed in the literature are outdated due to lack of maintenance[88]. Therefore, researchers are falling back to built-in DPC systems that are provided by the OS[78, 98] or maintained through open-source research projects[24, 32, 88]. The results of the usage analysis of DPC systems in graph summarization and intrusion detection research are shown in Figure 5.

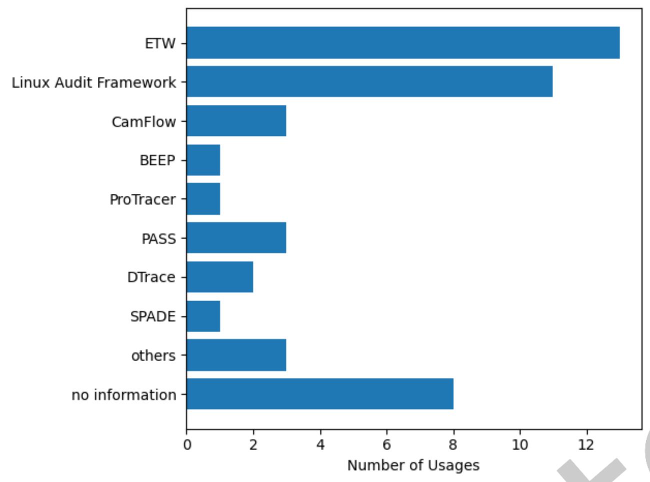


Fig. 5. Data Collection Approach Usage Comparison

4.4.3 Fault Tolerance. DPC systems must tolerate failures such as full disk, network outages, reboots, or system overloads caused by both malicious and non-malicious faults. In such cases, the provenance data's integrity and completeness have to be ensured[57].

4.4.4 Trustworthiness. The goal of attackers is to stay undetected while pursuing their goal, e.g. to compromise the system. To do so, attackers may try to compromise the DPC system itself, delete data provenance that includes their malicious behavior or inject additional provenance data to veil their traces. Hence, PIDS must consider the trustworthiness of the data provenance captured[57]. Many researchers assume that the data provenance is trustworthy.

4.4.5 Privacy. Data provenance contains privacy-sensitive information of users such as websites visited or files accessed. While general privacy issues of IDS have been discussed[72, 87], to the best of our knowledge, there is no research on privacy issues of DPC systems of PIDS nor proposed privacy-preserving mechanisms.

4.4.6 Provenance Graph Extension. Reviewed DPC systems collect data provenance based on system calls or CPU instructions. Multiple other existing data sources can be incorporated to enrich a provenance graph's contextual information. Possible data sources include network, application and database logs. For example, a system call that creates a network connection can be linked to a network packet in the network logs. While adding more contextual information would likely improve the detection accuracy, it would significantly increase the storage overhead, and negatively affect the detection time.

5 GRAPH SUMMARIZATION

One major challenge of PIDS is the large amount of data provenance generated. The amount of data can rapidly increase up to multiple TBs, depending on the number of hosts the data provenance is collected from and the number of days the data is stored. Given the nature of certain attacks such as APT, they can persist for up to multiple months, and thus, the data provenance must be stored for a long enough period. This leads not only to a high space overhead but also challenges in analyzing the data in real-time. Researchers proposed different approaches to summarize data provenance while keeping as much semantic information as possible, and an overview of proposed approaches is shown in Table 3.

Name	Year	Category	Mode	Requirements	Rate/Factor	Baseline	Dataset
Graph Compression[110]	2011	compression	Offline	None	61.5% (2.6x)	No Compression	Own dataset
LogGC[65]	2013	simplification	Online	Unit Instrumentation	98%	BEEP + No Compression	Own dataset
ProTracer[75]	2016	simplification	Online	Unit Instrumentation	92.9% (14x)	BEEP + No Compression	Own dataset
CPR + PCAR[112]	2016	grouping (edge)	Online	None	89%	No Compression	Own dataset
ProvWalls[7]	2017	policy	Online	MAC-enabled system	90%	BEEP + No Compression	Own dataset
KCAL[73]	2018	simplification	Online	Unit Instrumentation	70%	No Compression	Own dataset
CD + FD + SD[51]	2018	grouping (edge)	Online	None	89.1% (9.2x)	CPR + PCAR[112]	DARPA TC E2 + own dataset
NodeMerge[99]	2018	grouping (node)	Online	None	98.7% (75.7x)	No Compression	Own dataset
Winnower[48]	2018	grouping (node)	Online	None	99.9%	No Compression	Own dataset
GrAALF[93]	2019	grouping (edge)	Online	None	N/A	N/A	N/A
LogApprox[76]	2020	grouping (node)	Online	None	65% (2.87x)	No Compression	DARPA TC E3

Table 3. Overview of Graph Summarization Approaches

5.1 Criteria for reviewing Graph Summarization Techniques for PIDS

Graph summarization techniques aim to reduce the storage overhead, improve the intrusion detection efficiency, and reduce the scenario graph in forensic analysis. We reviewed graph summarization techniques based on their category, mode, and reduction rate. The category, mode and reduction rate are defined as follows:

5.1.1 *Categorization of Graph Summarization Techniques for PIDS.*

Reference [71] proposed a taxonomy for graph summarization techniques. We extend their taxonomy for PIDS data summarization techniques. We categorize the approaches into bit compression-based reduction, simplification-based reduction, policy-based reduction, and grouping-based reduction methods, which are defined as follows:

Definition 5.1 (Bit Compression-based Reduction). Bit compression-based reduction aims to reduce the number of bits required to store the data on a disk. Most methods apply lossless compression techniques and allow the reconstruction of the original input provenance graph.

Definition 5.2 (Simplification-based Reduction). Simplification-based reduction utilizes the security context to remove events which are considered as irrelevant for intrusion detection and forensic analysis.

Definition 5.3 (Policy-based Reduction). Policy-based reduction uses predefined policies defined by a security expert to summarize a provenance graph.

Definition 5.4 (Grouping-based Reduction). Grouping-based reduction aggregates nodes, edges, and their properties based on the security context.

5.1.2 *Mode of Graph Summarization Techniques for PIDS.*

Online graph summarization approaches can compress real-time data provenance without using historical data provenance. Offline graph summarization methods query the data provenance from the persistent storage, compress it, and then push it back to the persistent storage. PIDS profit from online graph summarization approaches because they reduce the sheer size of data before their transmission over the network. This reduces the complexity of the data provenance and the create, read, update, and delete (CRUD) operations made to the persistent storage[7, 48, 51, 65, 73, 75, 93, 99, 112].

5.1.3 *Reduction Rate of Graph Summarization Techniques for PIDS.*

The reduction rate is a commonly used evaluation metric of graph summarization techniques for PIDS and indicates how many nodes and edges in a provenance graph are removed by a technique to improve storage and analysis efficiency. We have used this evaluation metric to compare the reviewed techniques.

However, the reduction rate does not reflect the security value of these techniques. To address this issue, metrics to measure the security value of a particular graph summarization approach under different attack models were proposed in [76]. Lossless forensics defines the percentage of the number of removed edges compared to the number of edges in the original provenance graph. Causality-preserving forensics defines the percentage in which the graph summary approach preserves the information flow and causal relationships of the original provenance data. Attack-preserving forensics defines the percentage in which the graph summarization approach can remove benign information flows while preserving all malicious information flows. Since these metrics have only been proposed recently, none of the authors have evaluated their graph summarization technique by these metrics. We suggest authors of upcoming graph summary techniques to use these metrics for their evaluation.

The remainder of this section reviews graph summarization techniques for PIDS and is structured based on their categories.

5.2 Bit compression-based Reduction

One of the first compression approaches proposed adAPT a web graph compression technique to provenance graphs reducing the storage overhead of a provenance graph by up to 2.71 times. This technique searches for nodes with common child nodes in a provenance graph and then encodes them. Child nodes that have consecutive numbers can be encoded by noting the first one and the length. The remaining child nodes can be encoded by subtracting their previous child node number from their number. A drawback of this approach is that the nodes' and edges' properties are not considered[110].

5.3 Simplification-based Reduction

To reduce the sheer size of logs, a garbage collector for audit logs, namely LogGC, was proposed in [65]. A modified version of the classic reachability-based memory garbage collection algorithm removes redundant and unreachable nodes. LogGC optimizes the garbage collection process by partitioning long-running processes into units and files into logical data units and thus, can reduce the audit log size for forensic analysis by 14 times for regular applications and 37 times for server applications compared to BEEP[64].

Another simplification-based reduction approach is ProTracer[75], which reduces the sheer amount of provenance data by alternating between logging and provenance propagation. Based on the observation that often processes only read files but neither write to the permanent storage nor the external environment such as sending data over the network, those traces can be removed from the provenance data. ProTracer first partitions long-running processes into units and taints units that conduct read operations with the source they have read. The provenance data of these units is only getting logged when the units conduct any write operations before they are terminated. Secondly, ProTracer avoids logging dead events that do not permanently affect the systems by tainting a unit that conducts internal write operations. For example, if another unit does not access the created files during its lifecycle, the files are temporary files and thus, are not getting logged. Thirdly, ProTracer avoids the redundant logging of units by tainting units that behave the same as already logged units. As long as their behavior doesn't change, the provenance data of these units is not getting logged. Hence, ProTracer can reduce the space overhead by a minimum of 96% for log entries and 98% for disk space compared to BEEP[64].

Previous graph summarization approaches [65, 75] were applied after the logs have already been collected, transferred, and temporally stored. Those steps already produce a significant runtime overhead and temporal space overhead. To address this issue, a cache-based in-kernel online graph summarization system, namely Kernel-supported Cost-effective Audit Logging (KCAL), was proposed[73]. KCAL is a modification of the Linux Audit Systems and applies BEEP[64] to split long-running processes into units. First, in-unit redundancies, i.e. a unit performs the same operations on the same object, and cross-unit redundancies, i.e. different units that perform the same operations, are detected. Second, temporary files, i.e. files that get created, finished, and deleted by the same process, are identified. As a result, KCAL can reduce the runtime overhead of the Linux Audit System from 40% to 15%, and the space overhead by 90% on average.

5.4 Policy-based Reduction

One policy-based reduction approach is ProvWalls that reduces the space overhead of data provenance by limiting the monitoring to events that reside in the Trusted Computing Base (TCB) of an application[7]. By analyzing the system's Mandatory Access Control (MAC) policy, the information flow of provenance-sensitive objects can be identified. Thus, the TCB of an application can be specified by defining a provenance policy. The provenance capture system then uses the provenance policies to decide if the provenance of an event should be logged or not. ProvWalls adds only a small runtime overhead of 1.5% but can reduce the space overhead by up to 89% while assuring complete provenance. However, one requirement of this approach is that the system to be audited needs to be MAC enabled. A general drawback of policy-based reduction approaches is that they may miss data provenance of attacks that are designed to get around the predefined policies.

5.5 Edge-grouping-based Reduction

LogGC[65], ProTracer[75], and KCAL[73] require the presence of unit instrumentation to be effective. The drawbacks of unit instrumentation are that the source code needs to be accessible and that the instrumentation itself adds significant runtime overhead. To address this issue, two edge-grouping methods, coined Causality-Preserving Reduction (CPR) and Process-centric Causality Approximation Reduction (PCAR), were proposed in [112].

CPR is based on the observation that only a small number of key events show causal importance to other events. Thus, irrelevant events can be removed, and shadowed events can be aggregated with their key event. Figure 6 shows an example graph, whereby process *A* is the Point of Interest (POI) for forward tracing in the forensic analysis. The graph clearly shows that first, event *E5* is a shadow event of event *E2*, and thus, the semantic information such as the timestamp can be aggregated. Second, event *E3* is an irrelevant event that can be removed because it doesn't have any effect on the result of forward tracing in a forensic analysis[112].

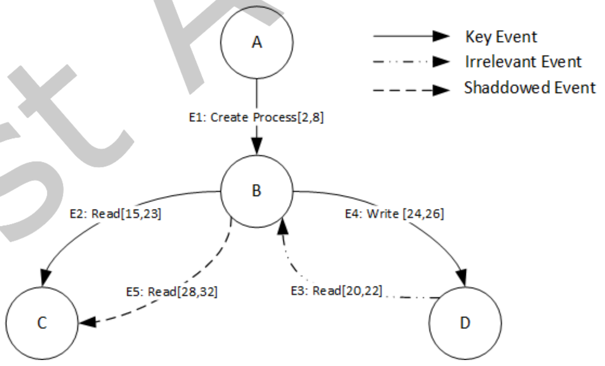


Fig. 6. Causality-Preserving Reduction (CPR)[112]

PCAR is based on the observation that there are processes that produce an intense burst of events, such as scanning for files or devices, which are semantically similar but cannot be reduced by CPR due to their interleaved causalities. PCAR detects such burst events, creates a neighbor set around the burst event, and checks traceability only for information flow from and into the neighbor set. With this approach, approximately shadowed events within the neighbor set can be detected and aggregated. In Figure 7 process *C* is a burst event, the dotted circle shows its neighbor set, and event *E3* is an approximately shadowed event. It can be aggregated with event *E2* even though it has interleaved causalities. Event *E5* and event *E6*, however, cannot be aggregated as their interleaved event *E7* is an information flow going outside of the neighbor set. As a result, CPR can reduce the space overhead by 56% and in combination with PCAR by 70%[112]. While the combination of CPR[112] and PCAR[112] reduce the space overhead by a factor of 1.8, they do not consider the global context of events. Continuous Dependence (CD), Full Dependence (FD) and Source Dependence (SD) preservation can further improve the space overhead reduction rate by considering the global context [51].

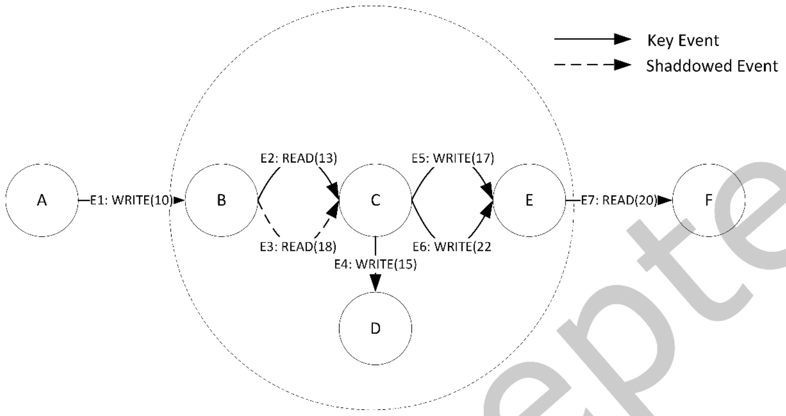


Fig. 7. Process-centric Causality Approximation Reduction (PCAR)[112]

CD preservation reduction works similar to CPR[112] and PCAR[112], but also aggregates duplicate events by their global reachability properties by considering the context of the event itself rather than by checking their local interleaving causalities. By applying CD preservation to the graph in Figure 8, event *E1* can be aggregated with event *E2* even though there is the interleaving causality of event *E3*[51].

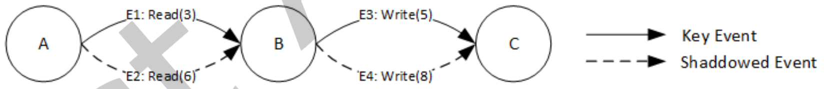


Fig. 8. Continuous Dependence (CD) Preservation Reduction[51]

In Figure 9 the previous graph has been extended by process *D* and thus, CD preservation reduction cannot aggregate the events *E3* and *E4* anymore. Nevertheless, the aggregation of those events would not affect the forward- and backward tracing in forensic analysis. Therefore, FD preservation aggregates events by checking if the resulted reduced graph would generate the same output for forward- and backward tracing as the original graph, and thus, events *E3* and *E4* can be aggregated again. On average, FD preservation can reduce the space overhead by a factor of 7[51].

To further reduce the space overhead, SD preservation removes events that do not affect the forward- and backward tracing in forensic analysis. Given the example graph in Figure 10, events *E5* and *E6* can be removed because the forward tracing from node *A* to *E* and backward tracing from node *E* to *A* on the reduced graph still results in the same set of nodes than applying it on the original graph. SD achieves a reduction factor of 9.2[51].

One drawback of CD, FD and SD is that they depend on global properties of graphs, and thus, computing it on a timestamped graph is expensive, mainly because the reachability changes over time. As a result, the authors

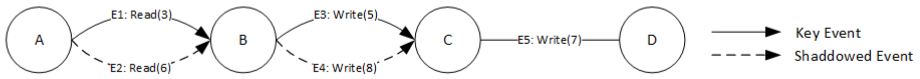


Fig. 9. Full Dependence (FD) Preservation Reduction[51]

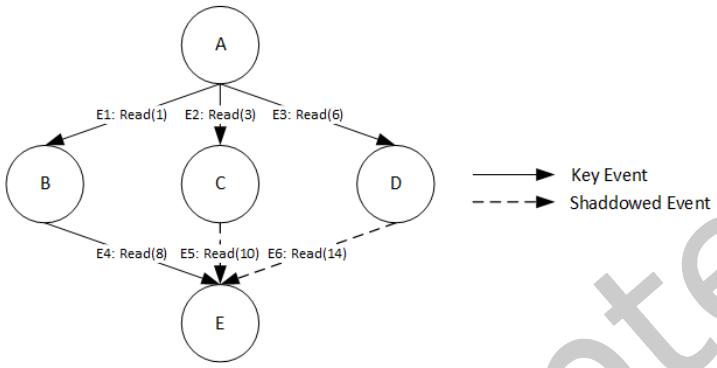


Fig. 10. Source Dependence (SD) Preservation[51]

proposed to convert the timestamped graph into a naive versioned graph and then apply different optimization techniques to reduce the number of edges and versions.

GrAALF [93] is a system for forensic analysis that collects data from heterogeneous sources, stores the data in one or multiple of the provided backend storage solutions, and enables real-time forward and backward tracing by using their proposed query language. To store the provenance data in multiple backend storage solutions efficiently, three graph summarization methods were proposed (see Figure 11). *Lossless Compression (C1)* aggregates the edge properties for causalities with the same subject node, object node, and edge type. The accuracy of *C2*, is the same as *C1* but only keeps the first and last occurrence of edge properties. *Lossy Compression (C3)*, is the same as *C1* but only keeps the first occurrence of edge properties. However, no evaluation of these graph summarization methods is provided so their effectiveness cannot be compared with other approaches.

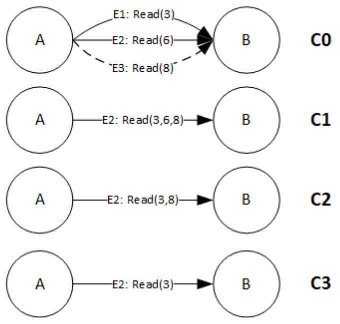
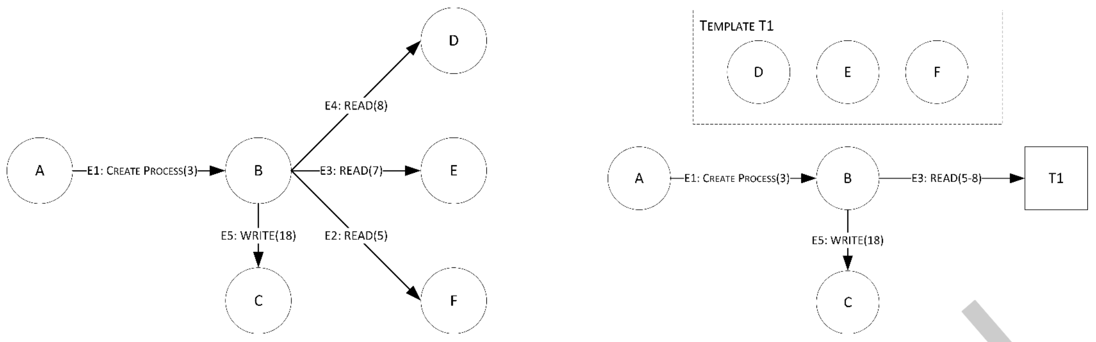


Fig. 11. GrAAFL[93]



(a) Original Graph (b) NodeMerge Reduction

Fig. 12. NodeMerge[99]

5.6 Node-grouping-based Reduction

Another approach to reducing the space overhead is NodeMerge[99], which is based on the observation that processes produce many redundant events during their initialization, such as loading libraries, accessing read-only resources, or retrieving configurations. NodeMerge detects and summarizes such event patterns by first, creating Frequence Access Pattern (FAP)s, second, automatically learning templates from the FAP based on an optimized Frequent Pattern (FP)-Growth algorithm, and third, using those templates to compress the further event data. Thus, the template-based approach can reduce the space overhead by 75 times for raw data and by 32 times compared to previous approaches such as LogGC[65] or CPR/PCAR[112]. The approach is particularly efficient for hosts who repeatedly run the same processes, but it may not be as efficient for hosts who execute mainly write-intensive processes. Figure 12 shows an example graph in which process *B* reads each initialization the files *D* - *F*, which NodeMerge detects and summarizes as template *T1* to reduce the space overhead.

Winnower is the first graph summarization approach that offers scalability for clusters. Replicated microservices in a cluster generate both structurally and semantically similar provenance graphs[48]. In Winnower, firstly, deterministic and node-specific information gets removed to create an abstract provenance graph on each worker node. Secondly, on each worker node, the abstract provenance graph gets converted into a behavior model by using Deterministic Finite State Automata (DFA) learning to generate graph grammar. Thirdly, the behavior models are aggregated into a unified model on a master node, which gets sent back to all worker nodes. Additionally, the unified model adds a confidence level to each node in the graph to reflect the consensus across the worker nodes. For example, a subgraph of the unified model with a low confidence level indicates that this behavior only occurred on one or a few worker nodes and thus, could represent an anomalous activity. Lastly, new provenance data on each worker node is checked if the unified model already models the data. If not, the behavior model gets updated and sent to the master node for aggregation. The graph in Figure 13 shows the resulting provenance graph by using winnower for monitoring cluster-wide behavior. For the nodes, *A* - *C* the confidence level is high, which implies that the worker nodes generate homogeneous behavior. However, for the nodes *D* - *E* the confidence level is low, which indicates that the behavior was generated by a single or only a few nodes, and those could reflect malicious behavior and have to be further analyzed. Winnower achieves a space overhead reduction of 98% while maintaining the important information required for attack investigation.

The authors proposed an attack-preserving graph summarization approach, called LogApprox, based on the observation that most of the storage of provenance data is occupied by I/O events (88.97%). LogApprox generates regular expressions to describe benign I/O events and then uses these regular expressions to summarize the

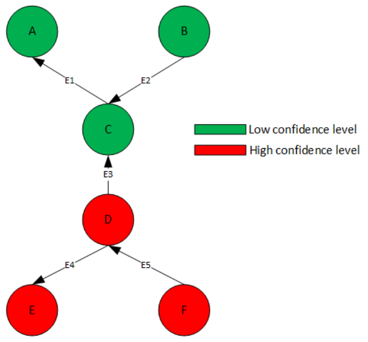
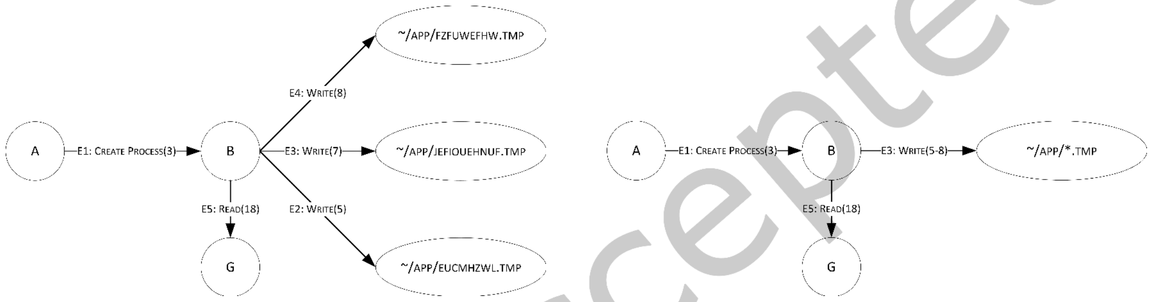


Fig. 13. Winnower[48]



(a) Original Graph (b) LogApprox Reduction

Fig. 14. LogApprox[76]

provenance graph. Figure 14 shows an example graph in which process *B* writes to multiple files. LogApprox detects these I/O events, creates a regular expression, and uses the regular expression to summarize the I/O events. The authors evaluated LogApprox against previous approaches such as LogGC[65], CPR[112], FD, and SD[51] by using their proposed metrics. The results show that only LogApprox and CPR achieve the highest forensic validity for attack-preserving. LogApprox could further achieve a higher data reduction rate than CPR. FD and SD achieved the highest data reduction rates but also the lowest forensic validity for attack-preserving forensics.

5.7 Discussion and Challenges

Graph summarization approaches proposed in the literature can efficiently summarize provenance graphs and significantly reduce the space overhead. Despite this, there are still open challenges and potential areas to further improve the space overhead reduction.

5.7.1 Lossy vs. lossless compression. The majority of the proposed approaches apply lossy graph summarization techniques to remove events that are not malicious or events that do not have a notable effect on intrusion detection performance. For instance, the data provenance of temporary files [7, 48, 51, 65, 75, 99, 112], or of read-only libraries or of dead files[75] can be removed without affecting the intrusion detection performance. In general, graph summarization of data provenance is a trade-off between compression rate and preserving enough semantic information required for sufficient intrusion detection and forward- and backward tracing in forensic analysis.

5.7.2 Scalability. The scalability of a graph summarization technique describes the property to handle a growing number of hosts. Notably, large enterprise networks, where sophisticated attacks such as APT are most common, can contain several hundred computers, and thus, graph summarization techniques need to be scalable. Most existing approaches focus only on the graph summarization rate itself. They are not considering any other factors, such as the number of hosts the data provenance is collected from or the number of resources available for graph summarization. Furthermore, the datasets used for evaluation are collected from a limited number of hosts and not replayed in real-time to apply and evaluate a graph summarization technique in a realistic setup. Thus, more research is required on scalable graph summarization techniques in a realistic setup.

5.7.3 Reduction Rate. The most important evaluation criteria of data compressions approaches are their compression rates. Table 3 shows that all proposed techniques can reduce the space overhead of data provenance significantly. However, with the following differences; Firstly, some techniques require the partitioning of long-running processes into units[65, 73, 75], which initially generates additional logs to be included in the evaluation dataset. Secondly, most graph summarization techniques have been proposed and optimized for a specific scenario. For example, Winnower[48] compresses data provenance of cluster environments that run multiple instances of the same service. Since each instance of one service generally generates semantically similar data provenance, it can be significantly compressed by first generalizing it and then finding consensus with other instances of the same service. Another example is LogGC[65], which achieved better compression rates in a server environment than a client environment.

5.7.4 Benchmark Dataset. A common issue in evaluating graph summarization approaches for data provenance is the benchmark dataset used. As highlighted in Table 3, most of the datasets were created by the authors of the approach itself, and the results are not reproducible because they have not published their datasets. Consequently, there is a high demand for a publicly available benchmark dataset to evaluate graph summarization approaches for PIDS.

5.7.5 Runtime Overhead. Another important evaluation criterion is the runtime overhead generated by graph summarization techniques. This measure is crucial when the graph summarization technique is executed on a device with limited resources. Table 3 gives an overview of the runtime overhead for each of the previously reviewed approaches, and the differences are in their details. First, ProTracer[75], LogGC[65], ProvWalls[7], and KCAL[73] are graph summarization techniques with an integrated data provenance capture system. Their stated runtime overhead includes both the runtime overhead of the graph summarization and the data provenance capture system. Second, the runtime overhead can vary in different execution scenarios of the graph summarization techniques. For example, KCAL generates only 1% of runtime overhead on clients, but up to 10% runtime overhead on servers. The runtime overhead on a server is significantly higher than on a client due to the enormous number of clients that a server application must serve, resulting in many dependencies[7].

6 INTRUSION DETECTION

Intrusion detection approaches for PIDS analyze the captured and summarized data provenance to detect intrusions. An overview of recent approaches is given in Table 4.

6.1 Criteria for reviewing Intrusion Detection Techniques for PIDS

We reviewed the intrusion detection approaches for PIDS based on their category, attack type and performance.

Name	Year	Category	Attack Type	Threshold	Dataset	False-Positives	True-Positives	Detection Time
AVF and OC3[8]	2019	Anomaly Score	APT	yes	DARPA	N/A	N/A	N/A
NoDoze[47]	2019	Anomaly Score	APT	yes	Own Dataset	N/A	N/A	Fast (<40s)
RepSheet[46]	2020	Anomaly Score	APT	yes	Own Dataset	Medium (2.2%)	High (100%)	Fast (<1ms)
Unsupervised Learning[9]	2020	Anomaly Score	APT	yes	DARPA[22]	N/A	N/A	N/A
HERCULE[90]	2016	Clustering	APT	yes	Own Dataset	Low (<0.0126%)	Medium (>80%)	N/A
AOML[4]	2020	Graph Embedding	APT	yes	Own Dataset	Low	High	Fast (<2s)
Random Forest[5]	2019	Graph Embedding	APT	yes	Own Dataset	High (<4%)	Low (>50%)	N/A
Pagoda[108]	2018	Rule Learning	APT	yes	Own Dataset	Low (<0.1%)	Medium (>75%)	Fast (<50s)
p-Gaussian[111]	2019	Rule Learning	APT	yes	DARPA TC E3	Medium (0.2%)	Medium (avg. 0.83)	1 event/s
AIQL[30]	2018	Rule-based	APT	no	Own Dataset	N/A	High	Fast (<3min)
NeedleHunter[114]	2019	Sequence Learning	APT	no	DARPA TC E3	N/A	High	N/A
TIRESIAS[94]	2018	Sequence-Learning	APT	yes	Own Dataset	N/A	Medium (>80%)	N/A
MORSE[50]	2020	Tag-propagation	APT	yes	DARPA TC E3	N/A	High	Fast (<1s)
SLEUTH[49]	2017	Tag-propagation	APT	no	DARPA TC E1	Low	High	Fast (<1s)
Log2Vec[69]	2019	Clustering	APT, Insider Threats	yes	CERT v6.2[68], LANL[61]	Low (<0.1%)	Medium	N/A
SAQL[29]	2018	Rule-based	APT, SQL Injection	no	Own Dataset	N/A	High	Fast (<2s)
Provenance Metrics[52]	2018	Graph Embedding	Data Exfiltration	yes	Own Dataset	N/A	N/A	N/A
DAGr and dDAGa[66]	2017	Regular Grammars	Data Exfiltration	yes	Own Dataset	Medium	Medium	N/A
FRAPpuccino[43]	2017	Clustering	DoS	yes	Own Dataset	Medium	Medium	N/A
DeepLog[25]	2017	Sequence Learning	DoS	yes	Own Dataset	Low (0.14%)	High (99.7%)	N/A
ADSAGE[31]	2020	Sequence Learning	Insider Threats	yes	CERT[68]	Medium	Medium	N/A
ProvDetector[106]	2020	Graph Embedding	Malware	yes	Own Dataset	Low	High	6 seconds/path
SIGL[45]	2020	Graph Embedding	Malware	yes	Own Dataset	Low	High	N/A
PIDAS[109]	2016	Rule Learning	Malware	yes	Own Dataset	Low	High	Fast (<6s)
CamQuery[89]	2018	Sequence Learning	Malware	no	N/A	N/A	N/A	N/A

Table 4. Overview of Intrusion Detection Approaches

6.1.1 Categorization of Intrusion Detection Techniques for PIDS. The proposed approaches can be categorized into anomaly-based, rule-based and, tag-propagation-based intrusion detection techniques, which we define as follows:

Definition 6.1 (Anomaly-based Intrusion Detection for PIDS). Anomaly-based approaches learn historical benign data provenance patterns and use these patterns to detect deviations of new data provenance.

Definition 6.2 (Rule-based Intrusion Detection for PIDS). Rule-based approaches chain one or more signatures or events to create a rule describing a cyberattack. A signature is a malicious pattern of a node or edge in a provenance graph, and typical examples are the hash of a malicious file or the network connection to a malicious host. An event is a benign or malicious pattern of a node or edge in a provenance graph, and a rule describes a subgraph consisting of signatures or events.

Definition 6.3 (Tag-propagation-based Intrusion Detection for PIDS). Tag-propagation-based approaches assign tags to nodes or edges in a provenance graph and utilize tag propagation to detect and trace malicious causalities in a provenance graph.

Traditional IDS used either signature- or anomaly-based intrusion detection techniques. Signature-based techniques have been only partly used for PIDS due to the following reasons. First, PIDS have been introduced to overcome the challenges of traditional IDS to detect more sophisticated attack types such as APT. Second, a provenance graph enables fine-grained event correlation, and thus, signatures in nodes or edges can be chained to a rule, which can describe a cyberattack more comprehensively.

6.2 Anomaly-based Intrusion Detection

Anomaly-based approaches can be further divided into sequence-learning, graph embedding, clustering, regular grammars, anomaly score, and rule learning.

6.2.1 Sequence Learning. A sequence learning method builds a model based on the causal order of benign events in a provenance graph. This model is then used to predict the next event for a given sequence of events. A deviation of the predicted event from the actual event indicates a potentially malicious event.

DeepLog[25] utilizes a multi-class classifier to predict the next event based on a sequence of past events. It also applies stacked Long Short-Term Memory (LSTM) to detect anomalies in the sequence of event properties.

Tiresias[94] is a security event predictor that uses a LSTM to learn from a Endpoint Detection and Response (EDR) dataset of past security events to then predict an attacker's next attack steps with up to 93% accuracy.

CamQuery[89] provides a vertex-centric API to implement provenance queries as value propagation applications. An example propagation application generates a feature vector for each vertex in the provenance graph, trains a Replicator Neural Network (RNN) and the resulting model is used to detect anomalies in streaming data provenance.

NeedleHunter[114] generates version-based provenance graphs to detect state changes of objects over time. It uses rules derived from the Tactics, Techniques, and Procedures (TTP) defined in the MITRE ATT&CK Matrix for Enterprises to taint malicious objects and then analyze the sequential dependencies between malicious tainted objects to detect APT[82].

ADSAGE[31] uses a Recurrent Neural Network (ReNN) to model the sequences of application logs and uses this model to predict future events and a Feed-Forward Neural Network (FFNN) to model the validity of the events to predict the anomaly score of future events.

Sequence learning methods can detect malicious behavior with high accuracy and are suitable for real-time detection. However, they might suffer from false alarms if the learned benign behavior changes due to evolution.

6.2.2 Graph Embedding. A graph embedding technique transforms nodes, edges, and their properties into vectors while conserving properties such as the structure of the provenance graph. Next, traditional anomaly detection methods for vector spaces can be applied to detect malicious behaviors.

SIGL[45] transforms provenance graphs into the vector space by using a component-based node embedding technique for graphs. It applies a LSTM to extract the graph features, calculates the anomaly scores for new software installations, and classifies them as benign or malicious based on a predefined threshold.

ProvDetector[106] selects the k rarest paths of a provenance graph and uses word2vector[79] to protect the paths to the vector space. The Local Outlier Factor (LOF) algorithm is used for each path of a new provenance graph to predict if it is malicious. A threshold is then used to judge if the whole provenance graph is considered malicious.

Another approach uses generic and provenance-specific network metrics to summarize the meaningful information and knowledge of a large and complex provenance graph as a vector. Generic network metrics include the number of nodes and edges, graph diameter, associative coefficient, average clustering coefficient, and degree distribution. Provenance-specific network metrics incorporate the number of node and event types, Maximum finite distance (MFD), MFD of derivations and average clustering coefficient by node type. Then, the authors evaluated their proposed metrics by training a decision tree classifier to detect malicious provenance graphs[52].

Reference [4] suggested Node2Vec[40] to learn continuous feature representations for each node in a provenance graph and then applies Adaptive Online Metric Learning (AOML) to minimize the separation between malicious nodes and maximize the separation between malicious and benign nodes.

Reference [5] proposed a supervised learning approach to detect APT infections of a system. Each node of a provenance graph is considered an instance labeled by a continuous labeling algorithm. For each instance, the algorithm calculates a probability of being malicious based on its distance and path to a known bad instance. Then, for each instance, a feature vector is produced by combining the provenance graph context, APT inception features and accounting for agents, processes, and file names. Finally, a random forest classifier is trained, which is then used to classify further instances as benign or malicious. The classifier was able to detect malicious instances with an accuracy of 50% and false positive rate (FPR) under 4%.

Various graph embedding techniques have been proposed to transform provenance graphs into the vector space while preserving important information such as the graph structure. The evaluation results have demonstrated that the combination of graph embedding techniques and traditional anomaly detection methods can effectively detect malicious behavior. However, graph embedding techniques are computationally expensive and, thus, not always applicable for real-time detection.

6.2.3 Clustering. Clustering-based approaches cluster nodes of a provenance graph into benign or malicious clusters by detecting behavior deviations of a system over time.

Clustering-based approaches in the literature utilize the Louvain method[90], Kullback-Leibler Distance (KLD)[43], or pair-wise similarity comparison of nodes[69] to cluster nodes in provenance graphs.

One common issue of clustering-based approaches is that they suffer from a high false alarm rate because it is challenging to distinguish between unknown benign system behavior, unknown malicious system behavior, and system errors[43].

6.2.4 Regular Grammar. Regular grammar-based approaches describe benign provenance graphs with regular grammars and then classify new provenance graphs, which cannot be described with these grammars as malicious.

Reference [66] proposed Directed Acyclic Graph regular grammars (DAGr) to model benign provenance graphs with regular grammars and deterministic Directed Acyclic Graph automata (dDAGa)s to represent those regular grammars as Deterministic Finite State Automatas (DFAs). New provenance graphs which cannot be modeled with these DFAs are considered malicious. Regular grammar-based approaches suffer from a high false alarm rate if the normal behavior changes and cannot be described with the initially created grammar.

6.2.5 Rule Learning. Rule learning-based approaches automatically learn rules from benign provenance graphs and then classify new provenance graphs, which these rules cannot describe as malicious.

PIDAS[109] and Pagoda[108] built rules based on the causalities between nodes in benign provenance graphs of the training dataset. For a new provenance graph, the provenance graph is classified as malicious if the ratio between matching and non-matching rules exceeds a predefined threshold. However, PIDAS and Pagoda cannot detect variances in benign or malicious provenance graphs that are not in the training dataset.

To overcome this shortcoming, P-Gaussian[111] uses a gaussian distribution detection schema that determines the similarity between two provenance graph paths while considering event orders and the number of events to detect variances of provenance graph paths. Like regular grammar-based approaches, rule learning-based approaches suffer from high false alarm rates if the normal behavior changes.

6.2.6 Anomaly Score. Anomaly Score-based approaches calculate an anomaly score for a provenance graph and judge it as malicious if the score exceeds a predefined threshold.

Reference [9] evaluated the following generic unsupervised learning algorithms to assess their effectiveness of detecting APT: 1) Fpoutlier (FPOF) determines frequent patterns of the properties across the objects in the dataset based on a predefined threshold. Then, the anomaly score of an object is calculated by the number of frequent patterns it includes. Objects with lower anomaly scores would be considered possible anomalies. 2) Outlier Degree (OD) determines first, the frequent patterns of the properties across the objects in the dataset and second, high-confidence rules, which describe the associations of different patterns. Such rules describe, for example, if object *o1* contains the pattern *p1*, then it must also contain the pattern *p2*. Then, each object is scored by applying the high-confidence rules to it. High scores correspond to high anomaly possibility. 3) Attribute Value Frequency (AVF) determines the frequency numbers of the individual properties and then scores each object in the dataset by adding up the probabilities of the actual properties in that specific object. So lower scores refer to rare properties which imply a possible anomaly. 4) One Class Classification by Compression (OC3) compresses objects in the dataset by using the compression algorithm Krimp. Krimp identifies common properties, stores them in a table, and uses them to compress the objects. Therefore, low compression rates of objects refer to a possible anomaly. 5) Similar to OC3, CompreX compresses the objects in the dataset, and the compression rates refer to the anomaly score. However, a more sophisticated compression strategy is used, which stores the identified patterns in multiple tables. This allows better exploitation of correlations between groups of patterns and improves the runtime overhead of compression.

The results show that FPOF and OD were not competitive on any of their evaluation dataset. AVF and OC3 achieved high normalized Discounted Cumulative Gain (nDCG) scores across all datasets and completed within 3 minutes. CompreX archived the highest nDCG score for the datasets, which include the smallest amount of object properties but fails or could not be completed within 3 hours for the datasets with more object properties. A downside of FPOF, OD and OC3 is that they require parameter tuning, which is not always ideal for an unsupervised learning setting. Also, only AVF can be applied in a streaming anomaly detection environment.

Further, the same authors suggested in [8] score aggregation techniques to improve the detection performance of the generic unsupervised learning algorithms AVF and OC3. Different aggregation techniques such as sum, average, median, geometric mean, and minimum have been evaluated in different contextual settings. The evaluation results show that aggregating anomaly scores can improve detection performance. However, it is not predictable which aggregation technique will deliver the best anomaly detection performance for a specific context.

NoDoze[47] is a ranking system for security alerts generated by an IDS to reduce false-positives (FP). NoDoze creates a scenario graph for each security alert, assigns anomaly scores to the edges based on their frequency of occurrence, and then determines the overall anomaly score by aggregating the anomaly score of each edge.

However, NoDoze lacks in the reasoning about the causal dependencies between such security alerts. RapSheet[46] addresses this issue by annotating edges in the scenario graph with attack context, in particular TTP of the MITRE ATT&CK Matrix for Enterprises. Then, it calculates the anomaly score of a scenario graph based on the causal order of security alerts.

One major issue of anomaly score-based approaches is that they rely heavily on a predefined threshold. This threshold is defined in advance and optimized based on contextual information about the current scenario. Consequently, the anomaly score of a malicious provenance graph can fall below this threshold.

6.3 Rule-based Intrusion Detection

Attack Investigation Query Language (AIQL)[30], and Stream-based Anomaly Query Language (SAQL)[29] are domain-specific query languages that can express fundamental events of cyberattacks in a provenance graph to allow security experts to write advanced detection rules. AIQL targets offline provenance graphs and, SAQL targets streaming provenance graphs.

Rule-based approaches can achieve high detection performances and are suitable for real-time detection, however, as with traditional signature-based methods, the performance highly relies on predefined rules defined by security experts[8].

6.4 Tag-propagation-based Intrusion Detection

SLEUTH[49] assigns trustworthiness and confidentiality tags to nodes in a provenance graph and detects intrusions by checking predefined policies. For instance, a potential intrusion could be a node with low trustworthiness accesses a node with high confidentiality. SEUTH can efficiently and precisely detect malicious behavior, however, it also suffers from high false alarms. MORSE[50] addresses this shortcoming by applying tag attenuation to reduce the impact of tag propagation from benign subjects to objects and tag decay to reduce the impact of tag propagation if benign subjects read suspicious objects but do not change their behavior patterns.

The evaluation of tag-propagation-based approaches demonstrates fast detection times and hence, provides real-time detection capabilities. Tag propagation on provenance graphs with long-running processes can, however, lead to the dependence explosion problem by propagating a tag to all its dependencies.

6.5 Discussion and Challenges

Even though most of the reviewed approaches still suffer from false alarms, the number of false alarms is significantly less than the one of traditional IDS. Nevertheless, more robust intrusion detection approaches, additional contextual information, and a real-world benchmark dataset could further reduce the number of false alarms.

6.5.1 Comparative Analysis. We analyzed the reviewed intrusion detection approaches for correlations between their category, attack type, and performance.

The majority of the approaches aimed to detect APT, most likely to overcome the low performance of traditional intrusion detection techniques for this attack type. These approaches cover all of the categories introduced in the taxonomy (see Figure 3). It is noticeable that tag-propagation-based approaches show outstanding performance with low false-positive rates, high true-positive rates, and fast detection times[49, 50].

The remaining approaches are all anomaly-based and aim to detect insider threats, SQL injection, data exfiltration, and Denial of Service (DoS) attacks. There is no clear correlation between their category, attack type, and performance. Notably, most of the approaches have been evaluated with a self-made and not publicly available dataset and thus, crucial correlations between the category, attack type, and performance can be hidden.

6.5.2 Threshold and Robustness. A majority of intrusion detection approaches for PIDS require a predefined threshold to distinguish malicious from benign events (see Table 4). Most researchers evaluate the impact of the selected threshold on the detection accuracy and false alarm rate. The final evaluation results are obtained from the threshold with the best trade-off between detection accuracy and false alarm rate. As a result, the selected threshold might work exceptionally well for the scenarios in the benchmark dataset used for the evaluation but might fail for other scenarios. Therefore, more research on adaptive or contextual threshold selection is required to make intrusion detection approaches more robust for real-world environments.

6.5.3 Contextual Information. Up to date, only a few researchers have explored additional contextual sources to enhance the detection performance and reduce the false alarm rate of PIDS. Explored sources are security alarms gathered from commercial EDR[46, 47, 94] and attack context such as the TTP of the MITRE ATT&CK Matrix for Enterprises[46, 94, 114]. In future work, other potential contextual sources should be considered which include the system's properties and user behaviors. The former could be used to select a threshold based on the system's properties. For example, benign data provenance on a server environment shows significantly different characteristics as on a client environment. An optimized threshold for each environment is likely to increase the detection performance and reduce false alarms. The latter one could be used to better distinguish between unknown benign behavior, unknown malicious behavior, and system errors, a known issue in PIDS[43].

6.5.4 Automatic Rule and Signature Generation and Chaining. The performance of rule-based intrusion detection techniques highly relies on predefined rules composed of chained signatures defined by security experts[8]. We see a high potential to automate rules and signatures generation by using provenance graphs. A likely approach could be as follow: A malicious file is executed in a sandbox environment with a data provenance capture system. The captured data provenance is transformed into a provenance graph. Signatures are derived from the nodes and edges in the provenance graph. Finally, the derived signatures are chained to create a rule. As a result, rules and signatures can be generated in a more timely manner so that other affected systems can be identified more quickly.

6.5.5 Benchmark Dataset. Like the graph summarization approaches, most intrusion detection approaches have been evaluated on a self-made dataset by the authors (see Table 4). As a result, there is a high demand for a publicly available benchmark dataset to evaluate data analysis approaches for PIDS.

7 BENCHMARK DATASETS FOR PIDS

A majority of PIDS researchers utilized self-made benchmark datasets to evaluate their approaches and did not publish them. Hence, it is challenging for other researchers to reproduce evaluation results and compare new approaches with previous ones. Figure 15 gives an overview of benchmark datasets used by researchers to highlight the importance of the problem.

This section defines characteristics of benchmark datasets, reviews publicly available benchmark datasets, and discusses issues and possible research directions to overcome these issues.

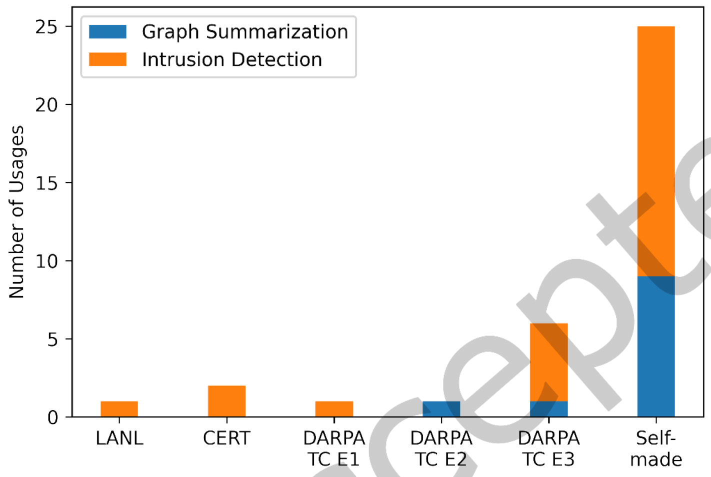


Fig. 15. Analysis of Benchmark Dataset utilized in PIDS research

7.1 Characteristics of Benchmark Datasets for PIDS

We define the characteristics of a realistic benchmark dataset for PIDS as follows:

- Benign Data: Benign data should include various user behaviors collected from real-world user studies.
- Malicious Data: Malicious data should contain various attack scenarios, ranging from various general to multi-stage attacks.
- Data Labels: The data should be labeled as benign or malicious. In the best case, malicious labels should include techniques defined in the MITRE Adversarial Tactics, Techniques & Common Knowledge (ATT&CK) matrix[82].
- Real-world Scenario: The data should replicate a real-world scenario, characterized by the ratio between benign and malicious data, number of hosts, number of unique events per host, and duration of the data collection.
- Completeness: The dataset should contain all data from the beginning until the end of the real-world scenario.
- Data Format: The data should be represented in a common format that enables interoperability with data analysis tools.
- Tools: If required, tools for data parsing, analysis, and visualization should be provided.
- Documentation: The dataset should contain documentation describing the benign data, malicious data, data labels, real-world scenarios, data presentation, and instructions on how to use the data.

7.2 Publicly-available Benchmark Datasets for PIDS

Name	OS	Year	Completeness for PIDS	Data Format	Tools	Documentation
DARPA TC E1	Windows, Linux, FreeBSD, Android	2016	Yes	CDM	Yes	Insufficient
DARPA TC E2	Windows, Linux, FreeBSD, Android	2017	Yes	CDM	Yes	Insufficient
DARPA TC E3	Windows, Linux, FreeBSD, Android	2018	Yes	CDM v18	Yes	Insufficient
DARPA TC E4	Windows, Linux, FreeBSD, Android	2018	Yes	CDM	Yes	Insufficient
DARPA TC E5[22]	Windows, Linux, FreeBSD, Android	2019	Yes	CDM v20	Yes	Insufficient
DARPA OpTC[21]	Windows 10	2020	Yes	eCar	No	Insufficient
CERT's Dataset[68]	Windows	2016	No	.csv	No	Medium
LANL's Dataset[61]	Windows	2015	No	.txt	No	Medium
ADFA-LD[19]	Ubuntu 11.04	2013	No	.txt	No	Good
ADFA-WD[41]	Windows XP	2016	No	.GHC	No	Good
ADFA-WD:SAA[41]	Windows XP	2016	No	.GHC	No	Good
AWSCTD[14]	Windows 7	2018	No	SQLite	No	Good

Table 5. Overview of Publicly-available Benchmark Datasets for PIDS

Name	Beningn Data	Malicious Data		Data Labels	Real-world Scenario			
		Scenarios	Attack Model		Total Events	Malicious Events	Ratio	Duration
DARPA TC E1	Synthetic	2	General	Yes	4 billion[39]	N/A	0.1%[49]	5 days [49]
DARPA TC E2	Synthetic	2	General	Yes	N/A	N/A	N/A	27 days [39]
DARPA TC E3	Synthetic	2	Multi-Stage	Yes	2 billion[39]	N/A	0.001%[114]	13 days [39]
DARPA TC E4	Synthetic	12	Multi-Stage	Yes	N/A	N/A	N/A	13 days [39]
DARPA TC E5[22]	Synthetic	8	Multi-Stage	Yes	12 billion[39]	N/A	N/A	10 days [39]
DARPA OpTC[21]	Synthetic	1	Multi-Stage	Yes	17 billion	0.3 million	0.0016%	14 days
CERT's Dataset[68]	Synthetic	6	General	Yes	135 million	470	0.0000035%	516 days
LANL's Dataset[61]	Realistic (Re-identified)	1	Multi-Stage	Yes	1 billion events	N/A	N/A	58 days
ADFA-LD[19]	Synthetic	6	General	Yes	5,206 events	N/A	N/A	N/A
ADFA-WD[41]	Synthetic	12	General	Yes	2,184	1,828	0.46%	N/A
ADFA-WD:SAA[41]	Synthetic	3	Multi-Stage	Yes	2,184	863	0.28%	N/A
AWSCTD[14]	No	12,110	General	N/A	112.56	112.56	100%	N/A

Table 6. Analysis of Publicly-available Benchmark Datasets for PIDS

As shown in Figure 15, PIDS researchers have utilized the Los Alamos National Lab (LANL)'s comprehensive cyber-security events dataset, the CERT Insider Threat Test Dataset, and the Defense Advanced Research Projects Agency (DARPA)'s Transparent Computing (TC) Datasets. This section summarizes these benchmark datasets and shows other potential benchmark datasets for PIDS and evaluates the datasets using the characteristics defined in subsection 7.1. An overview and evaluation of benchmark datasets is given in Table 5 and Table 6.

7.2.1 DARPA TC Datasets. DARPA conducted five engagements as part of their transparent computing program, intending to develop technologies and prototypes utilizing data provenance for real-time detection and forensic analysis of APT. Each engagement had specific objectives, involved simulated APTs and benign background activity on realistic server infrastructure, and generated a public benchmark dataset as a result. Multiple data provenance capturing systems such as ETW for Windows or SPADE for Linux have been utilized to collect data provenance[22]. Nevertheless, the benign background activity has not been published as part of the datasets[42].

7.2.2 DARPA Operationally Transparent Cyber (OpTC) Dataset. To determine the scalability of the TC program, DARPA created the OpTC dataset[21]. A two-week experiment was conducted on a test network with 1000 hosts running Windows 10. An APT was simulated over a three-day time window. System-level data provenance and network logs from Zeek sensors[101] have been collected, and the resulting OpTC dataset contains over 17 billion events, 0.3 million of them being malicious ones (0.0016%)[2]. Drawbacks of the dataset are the lack of documentation, information about the generation of the benign data, and that it is limited to one APT scenario.

In recent years DARPA has put in a significant effort to create benchmark datasets for PIDS. Their two latest datasets, the TC Engagement 5 and OpTC, have not been used yet by researchers; one potential reason could be the lack of documentation[2].

7.2.3 *CERT Insider Threat Test Dataset.*

The CERT Insider Threat Test Dataset contains synthetic logon events, email traffic, web browsing traces, file access logs, removable media usage, and LDAP information describing organizational hierarchy and user roles[68]. The dataset contains a total of 135,117,169 operations of 4,000 users during 516 days, whereby six users have 470 malicious operations resulting from six attack scenarios. Nonetheless, researchers utilizing this dataset stated an extreme imbalance problem[31, 69].

7.2.4 *LANL's comprehensive cyber-security events dataset.*

LANL 's comprehensive cybersecurity event dataset contains 58 days of de-identified events collected from five data sources within LANL 's corporate computer network. The sources are logon events, process events, DNS lookups, network flow events, and simulated red team events. In total, the dataset includes 1,648,275,307 events of 12,425 users on 17,684 computers. Well-known event properties such as system users (e.g. SYSTEM) or network ports (e.g. 80) have not been re-identified[61].

7.2.5 *ADFA IDS datasets.*

The ADFA IDS datasets consist of three datasets, the ADFA Linux Dataset (ADFA-LD)[19], the ADFA Windows Dataset (ADFA-WD), and the ADFA Windows Dataset: Stealth Attacks Addendum (ADFA-WD:SAA)[18].

ADFA-LD contains system-level data provenance from one ubuntu host collected by the Linux Audit Framework. Benign data provenance contains activities such as web browsing and latex document preparation. Malicious data provenance contains events from six simulated general attacks[19]. Despite all that, the dataset only contains system call numbers, and thus, this dataset cannot be used to evaluate approaches that need system call properties[70].

The ADFA-WD contains DLL traces of processes from a Windows XP host collected by Procmon. The datasets include 356 normal-, 1828 validation-, and 5773 attack traces. The attack traces are generated by exploiting twelve zero-day attacks on the host[41].

ADFA-WD:SAA: The ADFA-WD:SAA extends the ADFA-WD by 863 additional attack traces produced by three stealthy attacks, namely Doppelganger, Chimera, and Chameleon. The objective of these traces is to validate the resistivity of future HIDS[41]. Similar to ADFA-LD, ADFA-WD, and ADFA-WD:SAA are incomplete as they only contain DLL traces of processes, and the attack traces are only based on a few vulnerabilities[14].

7.2.6 *Attack-Caused Windows System Calls Traces Dataset (AWSCTD).*

To address the issues of the ADFA IDS datasets, the authors of [14] published the AWSCTD dataset. The objectives of this dataset are to use malware from a public repository to renew the attack traces quickly, utilize a wide selection of malware to generate attack traces, and contain complete system call traces and their properties. The dataset contains a total of 112.56 million attack traces generated from 10,276 malware. For comparison, the ADFA IDS datasets contain a total of 6636 attack traces generated from 15 malware. However, the AWSCTD dataset does not contain benign traces nor multi-stage attacks such as APT.

7.3 Discussion and Challenges

As demonstrated in Table 5, there are publicly available benchmark datasets to evaluate PIDS. The following paragraphs highlight major issues of publicly available benchmark datasets to explain why researchers utilized self-made benchmark datasets.

7.3.1 Lack of real-world datasets. It is challenging to create a benchmark dataset that mimics a real-world environment. The rapid changes in attack and defense techniques lead to publicly available datasets quickly become outdated and no longer represent the latest attack patterns[17, 19, 50, 67, 109].

The ratio between benign and malicious events in benchmark datasets does not reflect the ratio in real-world environments. Benchmark datasets tend to have a higher percentage of malicious events compared to real-wolrd data[86].

Most benchmark datasets contain synthetic benign data that is easier to distinguish from malicious behavior than real-world benign data. Intrusion detection approaches could archive a high detection performance on this data but could fail in real-world environments[86].

7.3.2 Lack of high-quality benign data. Benign data represents the most significant portion of a benchmark dataset. A majority of the benchmark datasets contain synthetic benign data because data provenance contains privacy-sensitive information of users such as websites visited, files created, and applications used. Hence, collecting real-world benign data is not practical due to privacy issues which would not only expose the users but also the network topology of an organization[2, 17, 95].

Researchers tried to overcome the privacy issue by anonymizing and re-identifying the real-world benign data, e.g. LANL 's Dataset[61] contains re-identified real-world benign data. However, researchers have stated that heavily anonymized real-world benign data reduces the data quality and amount of semantic information[95].

There are several ways to generate synthetic benign data. Most often, an autonomous agent randomly performs a predefined set of actions on a system. One drawback is the predictability of future events due to the limited number of actions an agent performs.

7.3.3 Lack of documentation and tools. It is challenging to utilize recent benchmark datasets due to a lack of documentation and tools. E.g. DARPA OpTC[21] contains billions of events resulting in Terabyte (TB)s of data. Even though this dataset contains high-quality data, researchers struggle to use this dataset because detailed explanations of the attack scenarios and information and tools on how to use the data are missing. The authors of [2] published additional documentation and analysis results of the DARPA OpTC[21] to make it easier for other researchers to use this dataset.

7.3.4 Issues of self-made benchmark datasets. Due to the lack of real-world benchmark datasets, lack of high-quality benign data, and lack of documentation and tools, many researchers created a self-made benchmark dataset to evaluate their approaches. Even though these datasets contain the latest attack scenarios and high-quality, real-world benign data, researchers cannot share these datasets due to privacy issues. This is a significant drawback because evaluation results cannot be reproduced, so researchers cannot compare their approaches. Another drawback is that self-made benchmark datasets may favor the approach of authors[17, 50].

7.4 Towards generating a real-world Benchmark Dataset for PIDS

Due to the high demand and importance of real-world benchmark datasets for PIDS, the following paragraphs discuss potential approaches to overcome the described issues.

7.4.1 Generation of Benign Data. Benign data can be generated either through real-world user studies or through simulations of user behavior. While the former suffers from privacy issues, the latter suffers from correctly mimicing reality. Little research on data obfuscation techniques for data provenance has been conducted. One major challenge is to obfuscate privacy-sensitive information while keeping as much semantic information as possible.[95]. Multiple data obfuscation techniques have been proposed for NIDS, such as homomorphic encryption, hash functions, bloom filters, and more[87]. There is a need to evaluate the effectiveness of these techniques on data provenance.

Since many researchers have used self-made benchmark datasets, metrics are needed to describe the data quality of these datasets. Researchers could then publish these metrics to allow other researchers to generate comparable benchmark datasets that replicate these metrics making the release of benign data redundant.

7.4.2 Generation of Malicious Data. Malicious data can be generated by red teams trying to exploit vulnerable targets or by simulations, which mimic realistic APTs. Simulations can be described by a configuration file, which enables the reproducibility of the scenarios at any time and is hence, the preferred method to generate malicious data.

There are various frameworks to simulate APTs, an overview is given in Table 7. Especially Xanthus[42], Caldera[81], Atomic Red Team[36], Splunk Attack Range[38], and PruleSharp[35] are very promising, since they allow one to configure simulations by describing cyberattacks with the TTP defined in the MITRE ATT&CK matrix[82]. Besides, the malicious data generated by these frameworks can be labeled more precisely by using TTP identifiers.

Name	**Author**	**Description**
Xanthus[42]	Han et al.	Framework that automates the configuration of data provenance capture system, attack execution, data collection, and data publishing.
Caldera[81]	MITRE	Cyber security framework that enables to fully automate APT simulations based on TTP defined in the MITRE ATT&CK matrix[82].
Atomic Red Team[36]	Red Canary	Simulates single TTP defined in the MITRE ATT&CK matrix[82] to test incidence and response systems.
Splunk Attack Range[38]	Splunk	Detection development platform that allows to quickly build lab environments, simulate attacks by using Caldera or Automatic Red Team, and automated detection rule testing.
PruleSharp[35]	Mauricio Velazco	Cyber security framework that simulates APTs based on the TTP defined in the MITRE ATT&CK matrix[82] in Windows Active Directory environments.
Simuland[37]	Microsoft	Microsoft Azure lab environment to simulate well-known techniques used in real attack scenarios.
CyberBattleSim[33]	Microsoft	Research platform in which automated agents use reinforcement learning algorithms to try to exploit vulnerabilities in a simulated corporate network.

Table 7. Overview of APT Simulation Frameworks

7.4.3 Data Format. Previous benchmark datasets for PIDS have been published in heterogeneous data formats, which makes it challenging to evaluate an approach with multiple benchmark datasets. In addition, there is a lack of tools to parse data provenance for PIDS from one format to another. Table 8 gives an overview of data formats utilized in existing benchmark datasets for PIDS. Especially eCAR[34] and CDM[22] provide not only promising data models to represent data provenance for PIDS but also support various OS. Despite this, these data models lack tools to parse data provenance captured by data provenance capturing systems.

7.4.4 Customizability. Ideally, benchmark datasets for PIDS should be easily customizable to add and publish new attack scenarios for other researchers quickly. Not only attack trends but also software evolve rapidly.

The trend towards frequently changing attack surfaces creates increased vulnerabilities for attackers. To adapt intrusion detection approaches to the latest attack trends, benchmark datasets need to include these trends.

Data Format	Author	Description
Prov-O[104]	W3C	Prov-O is a universal data model to describe data provenance in different systems and under different contexts using the Web Ontology Language (OWL2). It can be easily customized to optimize it for specific systems and contexts.
CAR[80]	MITRE	Cyber Analytics Repository (CAR) provides a data model inspired by Cyber Observable eXpression (CybOX) to describe observable objects that are monitored by an intrusion detection system. An object can be described by actions and fields and results in a tuple consisting of (object, action, field). Data provenance can be mapped to this tuple.
eCAR[34]	DARPA	extended CAR (eCAR) extends the CAR model by adding metadata such as host, user, and process information to events.
CDM[22]	DARPA	The Common Data Model (CDM) for Tagged Event Streams was introduced by DARPA to parse heterogenous data provenance from various OS into a common data model. The data model consists of six core entities: host, principals, subjects, events, objects and tags.

Table 8. Overview of Data Formats of Benchmark Datasets for PIDS

Frequent releases of new software versions including OS, data provenance capture system, and other application updates may lead to changes in the data provenance that need to be reflected in benchmark dataset.

8 CONCLUSION

Over the last few years, the number of cyberattacks has increased significantly, and enterprises defend themselves by deploying IDS to detect and respond to cyber incidents. Due to the high false alarm rate of traditional IDS and the immense required labor of security experts to validate these alarms, security incidents stay undetected for a long time. As a result, incident victims suffer from severe financial damage and data loss. The latest research on IDS has begun to explore data provenance to address the false alarm rate, and the first result shows good potential. With this survey, we presented an overview of PIDS including the demonstration of its potential, the introduction of a taxonomy of PIDS, the evaluation of recent research, and a discussion about issues and potential future research directions.

The major research issues we have identified are 1) The high runtime overhead posed by data provenance capture systems to collect fine-grained data provenance 2) Privacy issue of data provenance, which makes data sharing across enterprises impossible 3) Lack of scalable graph summarization techniques, 4) Lack of graph summarization techniques that utilize lossless and lossy reduction techniques, 5) Lack of robust real-time intrusion detection approaches that automatically adapt to the current scenario, and 6) Lack of real-world benchmark datasets to evaluate graph summarization and intrusion detection approaches.

Crucial future research directions incorporate 1) Privacy-preserving data provenance capturing approaches to enable data sharing, 2) Scalable graph summarization approaches that use lossless and lossy reduction techniques to reduce the storage overhead and improve the intrusion detection efficiency, 3) Robust real-time intrusion detection approaches that use additional contextual information to select a threshold for the current scenario, and 4) Real-world benchmark datasets to evaluate graph summarization and intrusion detection approaches.

ACKNOWLEDGMENTS

The work has been supported by the Cyber Security Research Centre Limited whose activities are partially funded by the Australian Government's Cooperative Research Centres Programme. We thank Dr. Omar Hussain and Dr. Keith Joiner for their valuable comments and helpful suggestions.

REFERENCES

[1] ACM. 2021. ACM computing surveys. https://dl.acm.org/journal/csur

[2] Md. Monowar Anjum, Shahrear Iqbal, and Benoit Hamelin. 2021. Analyzing the Usefulness of the DARPA OpTC Dataset in Cyber Threat Detection Research. arXiv:2103.03080 [cs.CR]

[3] Stefan Axelsson. 2000. Intrusion Detection Systems: A Survey and Taxonomy. (2000). https://doi.org/10.1.1.1.6603

[4] Gbadebo Ayoade, Khandakar Ashrafi Akbar, Pracheta Sahoo, Yang Gao, Anmol Agarwal, Kangkook Jee, Latifur Khan, and Anoop Singhal. 2020. Evolving Advanced Persistent Threat Detection using Provenance Graph and Metric Learning. In *2020 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 1–9.

[5] Mathieu Barre, Ashish Gehani, and Vinod Yegneswaran. 2019. Mining Data Provenance to Detect Advanced Persistent Threats. In *11th International Workshop on Theory and Practice of Provenance (TaPP 2019)*. USENIX Association. https://www.usenix.org/conference/tapp2019/presentation/barre

[6] Adam Bates, Dave (Jing) Tian, Kevin R B Butler, and Thomas Moyer. 2015. Trustworthy Whole-System Provenance for the Linux Kernel. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., 319–334. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/bates

[7] Adam Bates, Dave Jing Tian, Grant Hernandez, Thomas Moyer, Kevin R.B. Butler, and Trent Jaeger. 2017. Taming the costs of trustworthy provenance through policy reduction. *ACM Transactions on Internet Technology* 17, 4 (2017). https://doi.org/10.1145/3062180

[8] Ghita Berrada and James Cheney. 2019. Aggregating unsupervised provenance anomaly detectors. In *11th International Workshop on Theory and Practice of Provenance (TaPP 2019)*. USENIX Association, Philadelphia, PA. https://www.usenix.org/conference/tapp2019/presentation/berrada

[9] Ghita Berrada, James Cheney, Sidahmed Benabderrahmane, William Maxwell, Himan Mookherjee, Alec Theriault, and Ryan Wright. 2020. A baseline for unsupervised advanced persistent threat detection in system-level provenance. *Future Generation Computer Systems* (2020).

[10] Bibliometrix. 2021. Bibliometrix R Package. https://www.bibliometrix.org/index.html

[11] BITSIGHT. 2021. The Financial Impact of SolarWinds Breach. https://www.bitsight.com/blog/the-financial-impact-of-solarwinds-a-cyber-catastrophe-but-insurance-disaster-avoided

[12] Robert A Bridges, Tarrah R Glass-Vanderlan, Michael D Iannacone, and Maria S Vincent. 2019. A Survey of Intrusion Detection Systems Leveraging Host Data. *Comput. Surveys* 52, 6 (2019). https://doi.org/10.1145/3344382

[13] Business Insider. 2020. Here's a simple explanation of how the massive SolarWinds hack happened and why it's such a big deal. https://www.businessinsider.com.au/solarwinds-hack-explained-government-agencies-cyber-security-2020-12?r=US{&}IR=T

[14] Dainius Ceponis and Nikolaj Goranin. 2018. Towards a Robust Method of Dataset Generation of Malicious Activity for Anomaly-Based HIDS Training and Presentation of AWSCTD Dataset. *Baltic Journal of Modern Computing* 6 (2018). https://doi.org/10.22364/bjmc.2018.6.3.01

[15] Clarivate. 2021. Web of Science Core Collection Basic Search. https://apps.webofknowledge.com/WOS{_}GeneralSearch{_}input.do?product=WOS{&}search{_}mode=GeneralSearch{&}SID=F4RZJg74Nnxe9ZS23ay{&}preferencesSaved=

[16] Connected Papers. 2021. Explore connected papers in a visual graph. https://www.connectedpapers.com/

[17] Carlos Garcia Cordero, Emmanouil Vasilomanolakis, Nikolay Milanov, Christian Koch, David Hausheer, and Max Mühlhäuser. 2015. ID2T: A DIY dataset creation toolkit for Intrusion Detection Systems. In *2015 IEEE Conference on Communications and Network Security (CNS)*. 739–740. https://doi.org/10.1109/CNS.2015.7346912

[18] Gideon Creech. 2014. *Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks.* Ph. D. Dissertation. University of New South Wales, Canberra, Australia.

[19] Gideon Creech and Jiankun Hu. 2013. Generation of a new IDS test dataset: Time to retire the KDD collection. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. 4487–4492. https://doi.org/10.1109/WCNC.2013.6555301

[20] CrowdStrike. 2021. What Causes IT Alert Fatigue and How to Avoid It. https://www.crowdstrike.com/blog/causes-alert-fatigue-avoid/

[21] DARPA. 2021. Operationally Transparent Cyber (OpTC) Data Release. https://github.com/FiveDirections/OpTC-data

[22] DARPA. 2021. Transparent Computing Engagement 5. https://github.com/darpa-i2o/Transparent-Computing

[23] Brendan Dolan-Gavitt, Josh Hodosh, Patrick Hulin, Tim Leek, and Ryan Whelan. 2015. Repeatable Reverse Engineering with PANDA. In *Proceedings of the 5th Program Protection and Reverse Engineering Workshop (PPREW-5)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2843859.2843867

[24] DTrace. 2020. About DTrace. http://dtrace.org/blogs/about/

[25] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1285–1298.

[26] Elsevier. 2021. Science, health and medical journals, full text articles and books. https://www.sciencedirect.com/

[27] Elsevier. 2021. Scopus. https://www.scopus.com/search/form.uri?display=basic{#}basic

[28] FireEye. 2020. The Numbers Game: How Many Alerts are too Many to Handle? https://www.fireeye.com/offers/rpt-idc-the-numbers-game.html

[29] Peng Gao, Xusheng Xiao, Ding Li, Zhichun Li, Kangkook Jee, Zhenyu Wu, Chung Hwan Kim, Sanjeev R Kulkarni, and Prateek Mittal. 2018. SAQL: A Stream-based Query System for Real-Time Abnormal System Behavior Detection. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 639–656. https://www.usenix.org/conference/usenixsecurity18/presentation/gao-peng

[30] Peng Gao, Xusheng Xiao, Zhichun Li, Fengyuan Xu, Sanjeev R Kulkarni, and Prateek Mittal. 2018. AIQL: Enabling Efficient Attack Investigation from System Monitoring Data. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. 113–126.

[31] Mathieu Garchery and Michael Granitzer. 2020. ADSAGE: Anomaly Detection in Sequences of Attributed Graph Edges applied to insider threat detection at fine-grained level. *arXiv preprint arXiv:2007.06985* (2020).

[32] Ashish Gehani and Dawood Tariq. 2012. SPADE: Support for Provenance Auditing in Distributed Environments. In *Middleware 2012*, Priya Narasimhan and Peter Triantafillou (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 101–120.

[33] GitHub. 2021. CyberBattleSim. https://github.com/microsoft/CyberBattleSim

[34] GitHub. 2021. FiveDirections/OpTC-data. https://github.com/FiveDirections/OpTC-data/blob/master/ecar.md

[35] GitHub. 2021. PurpleSharp. https://github.com/mvelazc0/PurpleSharp

[36] GitHub. 2021. redcanaryco/atomic-red-team: Small and highly portable detection tests based on MITRE's ATT&CK. https://github.com/redcanaryco/atomic-red-team

[37] GitHub. 2021. SimuLand. https://github.com/Azure/SimuLand

[38] GitHub. 2021. Splunk Attack Range. https://github.com/splunk/attack{_}range

[39] John Griffith, Derrick Kong, Armando Caro, Brett Benyo, Joud Khoury, Timothy Upthegrove, Timothy Christovich, Stanislav Ponomorov, Ali Sydney, Arjun Saini, Vladimir Shurbanov, Christopher Willig, David Levin, and Jack Dietz. 2020. Scalable Transparency Architecture for Research Collaboration (STARC)-DARPA Transparent Computing (TC) Program. https://apps.dtic.mil/sti/citations/AD1092961

[40] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. arXiv:1607.00653 [cs.SI]

[41] Waqas Haider, Gideon Creech, Yi Xie, and Jiankun Hu. 2016. Windows based data sets for evaluation of robustness of Host based Intrusion Detection Systems (IDS) to zero-day and stealth attacks. *Future Internet* 8, 3 (sep 2016). https://doi.org/10.3390/FI8030029

[42] Xueyuan Han, James Mickens, Ashish Gehani, Margo Seltzer, and Thomas Pasquier. 2020. Xanthus: Push-button Orchestration of Host Provenance Data Collection. arXiv:2005.04717 [cs.CR]

[43] Xueyuan Han, Thomas Pasquier, Tanvi Ranjan, Mark Goldstein, and Margo Seltzer. 2017. FRAPpuccino: Fault-detection through Runtime Analysis of Provenance. In *9th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 17)*. USENIX Association, Santa Clara, CA. https://www.usenix.org/conference/hotcloud17/program/presentation/han

[44] Xueyuan Han, Thomas Pasquier, and Margo Seltzer. 2018. Provenance-based Intrusion Detection: Opportunities and Challenges. In *10th USENIX Workshop on the Theory and Practice of Provenance (TaPP 2018)*. arXiv:1806.00934 http://arxiv.org/abs/1806.00934

[45] Xueyuan Han, Xiao Yu, Thomas Pasquier, Ding Li, Junghwan Rhee, James Mickens, Margo Seltzer, and Haifeng Chen. 2020. SIGL: Securing Software Installations Through Deep Graph Learning. arXiv:2008.11533 [cs.CR]

[46] Wajih Ul Hassan, Adam Bates, and Daniel Marino. 2020. Tactical Provenance Analysis for Endpoint Detection and Response Systems. In *Proceedings of the IEEE Symposium on Security and Privacy*.

[47] Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. 2019. NODOZE: Combatting Threat Alert Fatigue with Automated Provenance Triage. In *Network and Distributed Systems Security (NDSS) Symposium*. Internet Society. https://doi.org/10.14722/ndss.2019.23349

[48] Wajih Ul Hassan, Mark Lemay, Nuraini Aguse, Adam Bates, and Thomas Moyer. 2018. Towards Scalable Cluster Auditing through Grammatical Inference over Provenance Graphs. *Network and Distributed Systems Security Symposium* (2018). https://doi.org/10.14722/ndss.2018.23141

[49] Md Nahid Hossain, Sadegh M Milajerdi, Junao Wang, Birhanu Eshete, Rigel Gjomemo, R Sekar, Scott Stoller, and V N Venkatakrishnan. 2017. SLEUTH: Real-time Attack Scenario Reconstruction from COTS Audit Data. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 487–504. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/hossain

[50] M N Hossain, S Sheikhi, and R Sekar. 2020. Combating Dependence Explosion in Forensic Analysis Using Alternative Tag Propagation Semantics. In *2020 IEEE Symposium on Security and Privacy (SP)*. 1139–1155. https://doi.org/10.1109/SP40000.2020.00064

[51] Md Nahid Hossain, Junao Wang, R Sekar, and Scott D Stoller. 2018. Dependence-Preserving Data Compaction for Scalable Forensic Analysis. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 1723–1740. https://www.usenix.org/conference/usenixsecurity18/presentation/hossain

[52] Trung Dong Huynh, Mark Ebden, Joel Fischer, Stephen Roberts, and Luc Moreau. 2018. Provenance Network Analytics. *Data Mining and Knowledge Discovery* 32, 3 (2018), 708–735. https://doi.org/10.1007/s10618-017-0549-3

[53] IEEE. 2011. IEEE Symposium on Security and Privacy. https://www.ieee-security.org/TC/SP2021/cfpapers.html

[54] IEEE. 2021. Xplore. https://ieeexplore.ieee.org/Xplore/home.jsp

[55] Internet Society. 2021. The Network and Distributed System Security Symposium (NDSS). https://www.ndss-symposium.org/

[56] ITnews. 2021. SolarWinds hack was 'largest and most sophisticated attack' ever. https://www.itnews.com.au/news/solarwinds-hack-was-largest-and-most-sophisticated-attack-ever-microsoft-561065

[57] Graeme Jenkinson, Lucian Carata, Thomas Bytheway, Ripduman Sohan, Robert N M Watson, Jonathan Anderson, Brian Kidney, Amanda Strnad, Arun Thomas, and George Neville-Neil. 2017. Applying Provenance in APT Monitoring and Analysis: Practical Challenges for Scalable, Efficient and Trustworthy Distributed Provenance. In *9th USENIX Workshop on the Theory and Practice of Provenance (TaPP 2017)*.

[58] Yang Ji, Sangho Lee, Evan Downing, Weiren Wang, Mattia Fazzini, Taesoo Kim, Alessandro Orso, and Wenke Lee. 2017. RAIN: Refinable Attack Investigation with On-demand Inter-Process Information Flow Tracking. In *24th ACM Conference on Computer and Communications Security (CCS 2017)*. https://www.microsoft.com/en-us/research/publication/rain-refinable-attack-investigation-with-on-demand-inter-process-information-flow-tracking/

[59] Yang Ji, Sangho Lee, and Wenke Lee. 2016. RecProv: Towards Provenance-Aware User Space Record and Replay. In *Provenance and Annotation of Data and Processes*, Marta Mattoso and Boris Glavic (Eds.). Springer International Publishing, Cham, 3–15.

[60] George Karantzas and Constantinos Patsakis. 2021. An Empirical Assessment of Endpoint Detection and Response Systems against Advanced Persistent Threats Attack Vectors. *Journal of Cybersecurity and Privacy 2021, Vol. 1, Pages 387-421* 1, 3 (jul 2021), 387–421. https://doi.org/10.3390/JCP1030021

[61] Alexander D Kent. 2015. Comprehensive, Multi-Source Cyber-Security Events. Los Alamos National Laboratory. https://doi.org/10.17021/1179829

[62] Samuel T King and Peter M Chen. 2003. Backtracking Intrusions. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP '03)*. Association for Computing Machinery, New York, NY, USA, 223–236. https://doi.org/10.1145/945445.945467

[63] Deepthi Hassan Lakshminarayana, James Philips, and Nasseh Tabrizi. 2019. A Survey of Intrusion Detection Techniques. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. 1122–1129. https://doi.org/10.1109/ICMLA.2019.00187

[64] Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. 2013. High Accuracy Attack Provenance via Binary-based Execution Partition. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*. The Internet Society. https://www.ndss-symposium.org/ndss2013/high-accuracy-attack-provenance-binary-based-execution-partition

[65] Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. 2013. LogGC: Garbage Collecting Audit Log. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS '13)*. Association for Computing Machinery, New York, NY, USA, 1005–1016. https://doi.org/10.1145/2508859.2516731

[66] Mark Lemay, Wajih Ul Hassan, Thomas Moyer, Nabil Schear, and Warren Smith. 2017. Automated provenance analytics: a regular grammar based approach with applications in security. In *9th USENIX Workshop on the Theory and Practice of Provenance (TaPP 2017)*.

[67] Zhenyuan Li, Qi Alfred Chen, Runqing Yang, Yan Chen, and Wei Ruan. 2021. Threat Detection and Investigation with System-level Provenance Graphs: A Survey. *Computers & Security* 106 (2021), 102282. https://doi.org/10.1016/j.cose.2021.102282

[68] Brian Lindauer. 2020. Insider Threat Test Dataset. https://doi.org/10.1184/R1/12841247.v1

[69] Fucheng Liu, Yu Wen, Dongxue Zhang, Xihe Jiang, Xinyu Xing, and Dan Meng. 2019. Log2vec: A Heterogeneous Graph Embedding Based Approach for Detecting Cyber Threats within Enterprise. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1777–1794.

[70] Ming Liu, Zhi Xue, Xianghua Xu, Changmin Zhong, and Jinjun Chen. 2019. Host-based intrusion detection system with system calls: Review and future trends. https://doi.org/10.1145/3214304

[71] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph summarization methods and applications: A survey. *Comput. Surveys* 51, 3 (apr 2018). https://doi.org/10.1145/3186727

[72] Emilie Lundin and Erland Jonsson. 2000. Anomaly-based intrusion detection: privacy concerns and other problems. *Computer Networks* 34, 4 (2000), 623–640. https://doi.org/10.1016/S1389-1286(00)00134-1

[73] Shiqing Ma, Juan Zhai, Yonghwi Kwon, Kyu Hyung Lee, Xiangyu Zhang, Gabriela Ciocarlie, Ashish Gehani, Vinod Yegneswaran, Dongyan Xu, and Somesh Jha. 2018. Kernel-Supported Cost-Effective Audit Logging for Causality Tracking. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, Boston, MA, 241–254. https://www.usenix.org/conference/atc18/presentation/ma-shiqing

[74] Shiqing Ma, Juan Zhai, Fei Wang, Kyu Hyung Lee, Xiangyu Zhang, and Dongyan Xu. 2017. MPI: Multiple Perspective Attack Investigation with Semantic Aware Execution Partitioning. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1111–1128. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/ma

[75] Shiqing Ma, Xiangyu Zhang, and Dongyan Xu. 2016. ProTracer: Towards Practical Provenance Tracing by Alternating Between Logging and Tainting. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society. http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/protracer-towards-practical-provenance-tracing-alternating-logging-tainting.pdf

[76] Noor Michael, Jaron Mink, Jason Liu, Sneha Gaur, Wajih Ul Hassan, and Adam Bates. 2020. On the Forensic Validity of Approximated Audit Logs. In *Annual Computer Security Applications Conference*. 189–202.

[77] Microsoft. 2020. Event Tracing. https://docs.microsoft.com/en-us/windows/win32/etw/event-tracing-portal
[78] Microsoft. 2020. Sysmon - Windows Sysinternals. https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon
[79] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546* (2013).
[80] MITR. 2021. Analytics | MITRE Cyber Analytics Repository. https://car.mitre.org/analytics/
[81] MITRE. 2021. Caldera: Scalable Automated Adversary Emulation Platform. https://github.com/mitre/caldera
[82] MITRE ATT&CK. 2021. Matrix - Enterprise. https://attack.mitre.org/matrices/enterprise/
[83] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. 2013. A survey of intrusion detection techniques in Cloud. *Journal of Network and Computer Applications* 36, 1 (2013), 42–57. https://doi.org/10.1016/j.jnca.2012.05.003
[84] Kiran-Kumar Muniswamy-Reddy, U Braun, D Holland, P Macko, D MacLean, Daniel W Margo, Margo I Seltzer, and Robin Smogor. 2009. Layering in Provenance Systems. In *USENIX Annual Technical Conference*.
[85] Kiran-Kumar Muniswamy-Reddy, David A Holland, Uri Braun, and Margo I Seltzer. 2006. Provenance-aware storage systems.. In *Usenix annual technical conference, general track*. 43–56.
[86] Sowmya Myneni, Ankur Chowdhary, Abdulhakim Sabur, Sailik Sengupta, Garima Agrawal, Dijiang Huang, and Myong Kang. 2020. DAPT 2020 - Constructing a Benchmark Dataset for Advanced Persistent Threats. In *Deployable Machine Learning for Security Defense*, Gang Wang, Arridhana Ciptadi, and Ali Ahmadzadeh (Eds.). Springer International Publishing, Cham, 138–163.
[87] Salman Niksefat, Parisa Kaghazgaran, and Babak Sadeghiyan. 2017. Privacy issues in intrusion detection systems: A taxonomy, survey and future directions. *Computer Science Review* 25 (2017), 69–78. https://doi.org/10.1016/j.cosrev.2017.07.001
[88] Thomas Pasquier, Xueyuan Han, Mark Goldstein, Thomas Moyer, David Eyers, Margo Seltzer, and Jean Bacon. 2017. Practical whole-system provenance capture. In *SoCC 2017 - Proceedings of the 2017 Symposium on Cloud Computing*. Association for Computing Machinery, Inc, 405–418. https://doi.org/10.1145/3127479.3129249 arXiv:1711.05296
[89] Thomas Pasquier, Olivier Hermant, Xueyuan Han, David Eyers, Thomas Moyer, Jean Bacon, Adam Bates, and Margo Seltzer. 2018. Runtime analysis of whole-system provenance. In *Proceedings of the ACM Conference on Computer and Communications Security*. Association for Computing Machinery, New York, NY, USA, 1601–1616. https://doi.org/10.1145/3243734.3243776 arXiv:1808.06049
[90] Kexin Pei, Zhongshu Gu, Brendan Saltaformaggio, Shiqing Ma, Fei Wang, Zhiwei Zhang, Luo Si, Xiangyu Zhang, and Dongyan Xu. 2016. HERCULE: Attack Story Reconstruction via Community Discovery on Correlated Log Graph. In *Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC '16)*. Association for Computing Machinery, New York, NY, USA, 583–595. https://doi.org/10.1145/2991079.2991122
[91] Devin J Pohly, Stephen McLaughlin, Patrick McDaniel, and Kevin Butler. 2012. Hi-Fi: Collecting High-Fidelity Whole-System Provenance. In *Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC '12)*. Association for Computing Machinery, New York, NY, USA, 259–268. https://doi.org/10.1145/2420950.2420989
[92] Can Sar and Pei Cao. 2005. Lineage file system. *Online at http://crypto. stanford. edu/cao/lineage. html* (2005), 411–414.
[93] Omid Setayeshfar, Christian Adkins, Matthew Jones, Kyu Hyung Lee, and Prashant Doshi. 2019. GrAALF: Supporting Graphical Analysis of Audit Logs for Forensics. *arXiv preprint arXiv:1909.00902* (2019).
[94] Yun Shen, Enrico Mariconti, Pierre Antoine Vervier, and Gianluca Stringhini. 2018. Tiresias: Predicting Security Events Through Deep Learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. Association for Computing Machinery, New York, NY, USA, 592–605. https://doi.org/10.1145/3243734.3243811
[95] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers and Security* 31, 3 (may 2012), 357–374. https://doi.org/10.1016/j.cose.2011.12.012
[96] Manolis Stamatogiannakis, Paul Groth, and Herbert Bos. 2015. Decoupling Provenance Capture and Analysis from Execution. In *Proceedings of the 7th USENIX Conference on Theory and Practice of Provenance (TaPP'15)*. USENIX Association, USA, 3.
[97] Manolis Stamatogiannakis, Paul Groth, and Herbert Bos. 2015. Looking Inside the Black-Box: Capturing Data Provenance Using Dynamic Instrumentation. In *Provenance and Annotation of Data and Processes*, Bertram Ludäscher and Beth Plale (Eds.). Springer International Publishing, Cham, 155–167.
[98] SUSE. 2020. Understanding Linux Audit. https://documentation.suse.com/sles/15-SP1/html/SLES-all/cha-audit-comp.html
[99] Yutao Tang, Ding Li, Zhichun Li, Mu Zhang, Kangkook Jee, Xusheng Xiao, Zhenyu Wu, Junghwan Rhee, Fengyuan Xu, and Qun Li. 2018. NodeMerge: Template Based Efficient Data Reduction For Big-Data Causality Analysis. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. Association for Computing Machinery, New York, NY, USA, 1324–1337. https://doi.org/10.1145/3243734.3243763
[100] J Thalheim, P Bhatotia, and C Fetzer. 2016. INSPECTOR: Data Provenance Using Intel Processor Trace (PT). In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. 25–34. https://doi.org/10.1109/ICDCS.2016.86
[101] The Zeek Project. 2021. The Zeek Network Security Monitor. https://zeek.org/
[102] USENIX. 2021. USENIX Security Symposium. https://www.usenix.org/conference/usenixsecurity21

[103] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max M ¨ Uhlh¨auser, Uhlh¨ Uhlh¨auser, and Mathias Fischer. 2015. Taxonomy and Survey of Collaborative Intrusion Detection. *Comput. Surveys* 47 (2015). https://doi.org/10.1145/2716260
[104] W3. 2021. PROV-O: The PROV Ontology. https://www.w3.org/TR/prov-o/
[105] Fei Wang, Yonghwi Kwon, Shiqing Ma, Xiangyu Zhang, and Dongyan Xu. 2018. Lprov: Practical Library-Aware Provenance Tracing. In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC '18)*. Association for Computing Machinery, New York, NY, USA, 605–617. https://doi.org/10.1145/3274694.3274751
[106] Qi Wang, Wajih Ul Hassan, Ding Li, Kangkook Jee, Xiao Yu, Kexuan Zou, Junghwan Rhee, Zhengzhang Chen, Wei Cheng, C Gunter, and Others. 2020. You are what you do: Hunting stealthy malware via data provenance analysis. In *Symposium on Network and Distributed System Security (NDSS)*.
[107] Wikipedia. 2020. DTrace. https://en.wikipedia.org/wiki/DTrace
[108] Yulai Xie, Dan Feng, Yuchong Hu, Yan Li, Staunton Sample, and Darrell Long. 2018. Pagoda: A Hybrid Approach to Enable Efficient Real-time Provenance Based Intrusion Detection in Big Data Environments. *IEEE Transactions on Dependable and Secure Computing* (aug 2018). https://doi.org/10.1109/TDSC.2018.2867595
[109] Yulai Xie, Dan Feng, Zhipeng Tan, and Junzhe Zhou. 2016. Unifying intrusion detection and forensic analysis via provenance awareness. *Future Generation Computer Systems* 61 (aug 2016), 26–36. https://doi.org/10.1016/j.future.2016.02.005
[110] Yulai Xie, Kiran Kumar Muniswamy-Reddy, Darrell D.E. Long, Ahmed Amer, Dan Feng, and Zhipeng Tan. 2011. Compressing Provenance Graphs. *3rd Workshop on the Theory and Practice of Provenance, TaPP 2011* (2011).
[111] Y Xie, Y Wu, D Feng, and D Long. 2019. P-Gaussian: Provenance-Based Gaussian Distribution for Detecting Intrusion Behavior Variants Using High Efficient and Real Time Memory Databases. *IEEE Transactions on Dependable and Secure Computing* (2019), 1. https://doi.org/10.1109/TDSC.2019.2960353
[112] Zhang Xu, Zhenyu Wu, Zhichun Li, Kangkook Jee, Junghwan Rhee, Xusheng Xiao, Fengyuan Xu, Haining Wang, and Guofei Jiang. 2016. High Fidelity Data Reduction for Big Data Security Dependency Analyses. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. Association for Computing Machinery, New York, NY, USA, 504–516. https://doi.org/10.1145/2976749.2978378
[113] Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, and Engin Kirda. 2007. Panorama: Capturing System-Wide Information Flow for Malware Detection and Analysis. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*. Association for Computing Machinery, New York, NY, USA, 116–127. https://doi.org/10.1145/1315245.1315261
[114] Han Yu, Aiping Li, and Rong Jiang. 2019. Needle in a Haystack: Attack Detection from Large-Scale System Audit. In *2019 IEEE 19th International Conference on Communication Technology (ICCT)*. IEEE, 1418–1426.
[115] L Zeng, Y Xiao, and H Chen. 2015. Linux auditing: Overhead and adaptation. In *2015 IEEE International Conference on Communications (ICC)*. 7168–7173. https://doi.org/10.1109/ICC.2015.7249470