Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Threat detection and investigation with system-level provenance graphs: A survey

Zhenyuan Li[a], *Qi Alfred Chen*[b], *Runqing Yang*[a], *Yan Chen*[c], *Wei Ruan*[a,*]

[a] *Zhejiang University, China*
[b] *University of California, Irvine, USA*
[c] *Northwestern University, USA*

ARTICLE INFO

Article history:
Received 9 June 2020
Revised 29 November 2020
Accepted 24 March 2021
Available online 18 April 2021

Keywords:
Cyber Threat
Provenance Graph
Intrusion Detection
Digital Forensic
Information Flow

ABSTRACT

With the development of information technology, the border of the cyberspace gets much broader and thus also exposes increasingly more vulnerabilities to attackers. Traditional mitigation-based defence strategies are challenging to cope with the current complicated situation. Security practitioners urgently need better tools to describe and modelling attacks for defense.

The provenance graph seems like an ideal method for threat modelling with powerful semantic expression ability and attacks historic correlation ability. In this paper, we firstly introduce the basic concepts about system-level provenance graph and present a typical system architecture for provenance graph-based threat detection and investigation. A comprehensive provenance graph-based threat detection system can be divided into three modules: *data collection module*, *data management module*, and *threat detection modules*. Each module contains several components and involves different research problems. We systematically taxonomize and compare the existing algorithms and designs involved in them. Based on these comparisons, we identify the strategy of technology selection for real-world deployment. We also provide insights and challenges about the existing work to guide future research in this area.

© 2021 Elsevier Ltd. All rights reserved.

1. Introduction

Threat detection in cyberspace is an arms race between adversaries and defenders. In this arms race, attackers can almost always bypass existing detection mechanisms by discovering new attack surfaces, while defenders are usually tired of plugging various vulnerabilities (Sym, 2020). Therefore, it is necessary for security researchers and practitioners to start rethinking about traditional mitigation techniques and try to design more robust and general detection mechanisms against not only the various existing attacks but also the previously unseen ones. The Defense's Advanced Research Projects Agency (DARPA) has launched a four-year project called Transparent Computing since 2015 (Darpa, 2015), trying to find a high-fidelity and visible method to abstract the interaction between components in the opaque system. The researchers found that the provenance graph may be a promising tool, with a strong abstract expression ability and relatively high efficiency.

Now more and more research works (Barre et al., 2019; Hassan et al., 2019; Hossain et al., 2017; Ma et al., 2015; Milajerdi et al., 2019a; 2019b; Xie et al., 2018; 2019) began to focus on detection and response algorithms based on provenance graphs and believe that provenance graph has the potential to be-

* Corresponding author.
E-mail address: ruanwei@zju.edu.cn (W. Ruan).
https://doi.org/10.1016/j.cose.2021.102282
0167-4048/© 2021 Elsevier Ltd. All rights reserved.

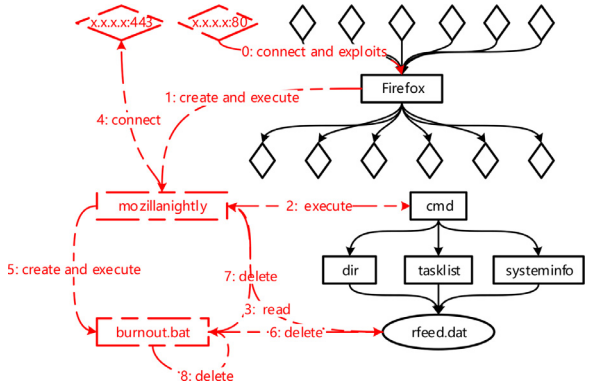


Fig. 1 – A provenance graph sample

come the next generation of more robust detection mechanisms. As shown in the Fig. 1, the provenance graph represents the relationship between the control flow and data flow between the subject (such as processes, threads, etc.) and the object (such as files, registry, network sockets) in the system through a directed graph with timing. The provenance graph can link causal events in the system, regardless of the time between the two events. All in all, utilizing provenance graphs for threat detection and investigation has the following advantages:

- Provenance graphs altogether show system execution by representing them as interactions between system objects. Such dependency is innate for all the execution trace. Unstructured log like Auditd (aud, 2020.3) can also be transformed into provenance graph (Gehani and Tariq, 2012).
- Provenance graphs enable semantic-aware and robust detection. Compared to unstructured audit logs, provenance graphs with spatial and temporal information are more difficult to forge by attackers (Han et al., 2018). Moreover, provenance graphs provide richer semantic; thus security analysts can conduct more effective and thorough attack investigation.
- Provenance graphs keep all the execution history. Advanced persistent threat (APT) attacks (APT, 2020) are long-running and stealthy attacks. To investigate such attacks, analysts need to access and understand the whole attack history. Actually, system execution history is necessary for any intrusion to trace the entry point and understand the impact.

To take advantage of the provenance graph, security researchers need to design and implement provenance graph-based detection systems. A typical system can be divided into three sub-modules: "data collection module" (Section 4), "data management module" (Section 5), and "threat detection module" (Section 6).

The data collection module is the foundation of the detection systems. It needs to be able to collect system-level provenance information efficiently and accurately. The data management module acts as a bridge between the collector and detector. It is responsible for providing efficient and fast query interfaces while storing massive amounts of data efficiently and economically. The threat detection module needs to process large amounts of data and locate stealth malicious behaviors with the lowest possible overhead and the shortest latency.

To design such an ideal provenance graph-based detection and investigation system, we should take the following four research questions into consideration:

RQ1: How to reduce the size of the data storage as much as possible while maintaining the semantics?

RQ2: How to balance the space efficiency of the provenance graph storage with the time efficiency of the query?

RQ3: How to design an efficient and robust intrusion detection algorithm and balance the true-positives and false positives?

RQ4: How to shorten the response time of detection and forensics as much as possible?

The potential answers to RQ1 and RQ2 are discussed in §5. The potential answers to RQ3 and RQ4 are discussed in §6. All in all, this survey makes the following

Contributions:

- We present the first thorough survey for threat detection and investigation with provenance graphs.
- We taxonomize various representative techniques used in existing papers and depict a typical architecture design of the provenance-based threat detection systems today in Section 2.2.
- We employ various performance indicators to systematically compare dozens of existing detection systems in Section 6. Based on the comparison, we identify the strategy of technology selection for real-world deployment in Section 7. Moreover, we provide multiple insights and challenges for future studies.

The rest of the paper is organized as follows: Section Section 2 introduced the background knowledge of the system-level provenance graph, including several basic definitions, the typical design of a detection system, and brief research history. Section Section 3 introduced related works and the scope of this survey. Section Sections 4, 5, and 6 focused on three sub-modules, respectively. Section Section 7 detailed described the advantages and disadvantages of different approaches from several perspectives and provided multiple insights and challenges. Section Section 8 presented conclusions.

2. Background

2.1. *Definition of system-level provenance graph*

System-level provenance graphs treat all system-level entities as vertices and all operations between entities as edges. The operations are collected by auditing tools and generate events stream with timestamps. The order of events affect semantics, and events are directed, which indicate the flow of data or control. Thus, provenance graphs have strong spatial and temporal properties. Such properties are called *causality* for provenance graphs. Correspondingly, provenance graphs are

Table 1 – Common provenance events list.

Sample graph	Description
	Write File
	Read File
	Send Data
	Receive Data
	Create New Process
	Inter-process communication

also called *causality graphs*. A series of related basic definitions are given as follows:

Definition 1 Subjects and Objects. Subject refer to the entity in the system that perform a operation to another entity that is called object. Subject and objects are denoted by u and v respectively.

It is worth mentioning that subjects and objects are relative, a subject of one operation can be the object of another event. Subjects can be processes, threads, etc. Moreover, Objects can be files, sockets, and so on. For different operation systems, the types of subject and object could be different. For example, Windows has unique registry objects and COM objects. However, it is not complicated to extend the provenance graph with more types of subjects and objects.

Definition 2. Events refer to the operations between entities in the system. An event includes four main attributes: the subject performing the operation, the object being operated, the time when the event occurred, and the specific content of the operation. Thus, a event can be denoted by a quad $< subject, object, time, operation >$ (or $< u, v, t, o >$ for short.) Table 1 lists the most commonly used events. And it is relatively easy for analysts to add more events.

Definition 3. Provenance Graph is the collection of all subjects, objects, and events, which can be denoted by $G = (S, O, E)$, where S represent the collection of subjects, O represent the collection of objects, E represent the collection of events.

In provenance graphs, both subjects and objects are represented as nodes, while events are represented as edges. There could be more than one edge between two nodes with different time or operation.

Definition 4. Causality Dependency. Two events $e_1 = (u_1, v_1, t_1)$ and $e_2 = (u_2, v_2, t_2)$ have causality dependency, if $v_1 = u_2 \wedge t_1 < t_2$.

Causality dependencies indicate the possible data and control flow between two events. However, two events are causality dependent does not necessarily mean there are data or control flow between them. Thus, compared to taint analysis (Newsome and Song, 2005), the causality-based analysis will introduce more false dependencies and cause more severe explosion problems.

Definition 5. Backward Tracking. Starting from a single detection point (e.g., a suspicious file), the backward tracking process tries to find all nodes in the provenance graph that causally affect the detection point.

Definition 6. Forward Tracking. Starting from a single detection point, the forward tracking process tries to find all nodes in the provenance graph that causally depend on the detection point.

The backward and forward tracking is widely used together in attack investigation to find the entry point and analysis the impact of the attack.

2.2. *Typical design of provenance graph-based detection system*

In this section, we will introduce the composition of a typical provenance graph-based detection system. As Figure 2 shows, firstly, data collection modules should be installed in the target hosts to collect operations between system objects, which indicates provenance information. Coarse-grained provenance (Section 4.1) information can be obtained with built-in auditing systems for most of today's operation systems, such as ETW ETW (2020.3) (Event Tracing for Windows) and Linux auditing system (aud, 2020.3). However, to collect more fine-grained provenance (Section 4.2), analyzer needs to install extra infrastructure, such as common libraries or hook into system calls. These fine-grained techniques have much higher overhead, ranging from 2× to 10×, and sometimes require support from vendors. The collected information will be parsed into a stream of events defined by Definition 3. The event stream will be transformed into a data management module or directly to a stream-based detection system.

In the data management module, a filter will apply different data reduction algorithm (Section 5.2) to remove redundant events according to different principles. Data reduction for provenance graph can not only reduce storage space but also reduce subsequent detection or investigation overhead. The compressed data will be stored in databases, which is appropriately designed to support frequent queries (Section 5.3) and persistent access (Section 5.1).

The last and most important module is threat detection modules (Section 6). Intrusion detection based on provenance graphs is not straight-forward. The most significant challenge comes from the massive amount of data generated in real time. A typical operating system will perform massive file read and write and network connection operations, which brings a lot of background noises. According to the survey results in Xu et al. (2016), for a typical bank with 20,000 hosts, about 70 PB of logs are generated annually. How to find out suspicious events timely is also challenging. One mitigation strategy for both challenges is to build a concise yet comprehensive model incrementally with stream data input.

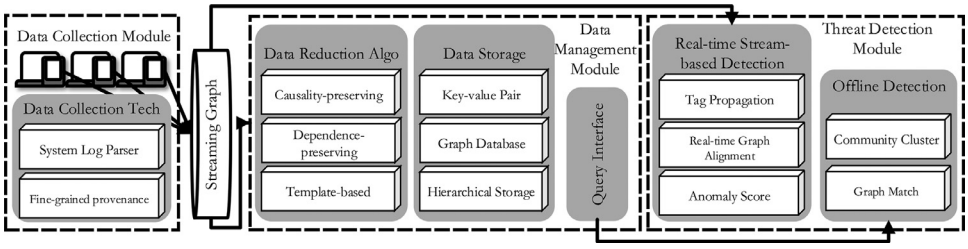


Fig. 2 – A general framework of provenance-based threat detection system.

2.3. *A brief history of the adoption of provenance graph in threat detection*

As shown in Fig. 3, we studied dozens of research work and observed two major technology trends. The first trend we find is *the study of fine-grained provenance graph collection*. The original system-level provenance graph is coarse-grained, which has lots of false dependence and thus leads to the "dependence explosion" problem. Fine-grained data collection can fundamentally mitigate this problem, while the overhead is much higher.

The second trend we find is *the study of realtime threat detection*. The response time is critical to real-world security investigation. For example, a quick response can effectively avoid the same attack and reduce the loss. However, the investigations after building the complete provenance graph introduce a long delay to such responses. So far, researchers have been focusing on streaming graph-based detection that can perform real-time detection and investigation.

3. Related work and scope of this survey

In this section, we present a holistic view of researches related to system-level provenance graph-based threat detection. Then, we define the scope of this survey and describe our survey methodology.

3.1. *Intrusion detection*

Intrusion detection (Axelsson, 2000; Buczak and Guven, 2016; Teresa F. Lunt, 1993) has been widely studied for several decades on different platforms, such as Host, cloud (Modi et al., 0000), Mobile platform, Cyber-Physical systems (Mitchell and Chen, 2014), etc.

In general, intrusion detection approaches can be divided into three categories: signature-based, anomaly-based, and hybrid. Signature-based approaches (Edge and Falcone Sampaio, 2009) are effective for detecting known attacks without many false-positives. However, lots of labor are required to maintain the signature database. Anomaly-based approaches (Hodge and Austin, 2004; Prasad et al., 2009; Yu, 2012) model the normal behavior and identify anomalies. They can not only detect zero-day attacks but can also produce lots of false-positives. Hybrid approaches combine multiple detection techniques to improve accuracy.

System-level provenance graph-based detection has a similar taxonomy. However, it utilizes a brand new data source, namely, system-level provenance graphs. Previous low-level data sources, such as system calls and taint analysis, suffer from high overhead and difficulty constructing semantic. In parallel, high-level data sources, such as system audits (aud, 2020.3), miss many behaviors and are easily bypassed. The system-level provenance graphs are believed to have the appropriate granularity. It can model all data flow and information flow in systems as graphs containing very rich semantic for intrusion detection. Moreover, it is relatively lightweight to be collected and analyzed in real-time.

Furthermore, attack techniques have evolved, becoming increasingly stealthy and persistent. It is difficult to distinguish malicious behavior based on single-point detection accurately. Correlation analysis (Ficco, 2013; Husák and Kašpar, 2019) can combine multi-source information and detect stealth threats effectively without much false-positive. However, most of the existing event-level correlation analyses rely on prior knowledge, therefore, hard to expand. Provenance graph-based detection supports correlation analysis naturally with causality analysis, which can help improve detection accuracy.

3.2. *Provenance*

Data provenance, also called data lineage, was initially introduced to find the origin of data in databases (Buneman et al., 2001; Woodruff and Stonebraker, 1997). It provides a historical record of data and its origins. With the provenance information of data, we can obtain the validity and confidence of data.

Data provenances are widely adopted for multiple different purposes, such as reproducibility (Greenwood et al., 2003; Miles et al., 2007), fault injection (Naughton et al., 2009), and so on. Several surveys have also been done for different provenance applications (Freire et al., 2008; Herschel, 2017; Simmhan et al., 2005; Zafar et al., 2017).

This survey focuses on system-level provenance information modeled as provenance graphs, which record the information flow between system-level objects in detail. Such information can be useful in locating potentially malicious behavior, such as information leakages, etc.

3.3. *Graphs for security purpose*

Graph structures are widely utilized in cyber security because of their rich semantics and powerful representation. Different kinds of graphs are extracted for different purpose according to their different properties.

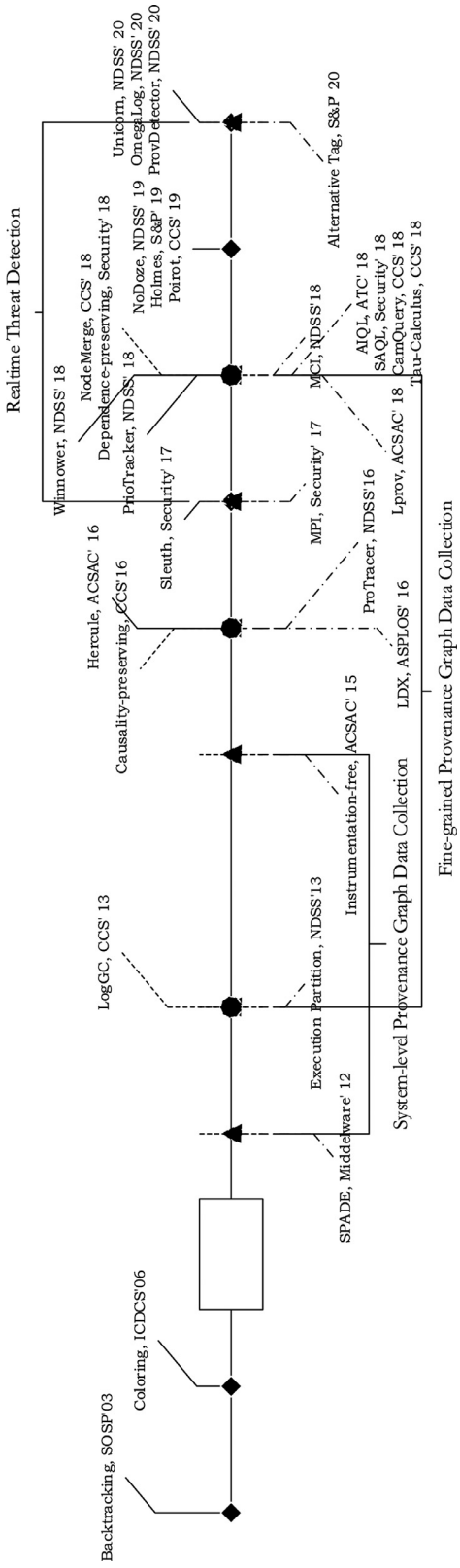


Fig. 3 – A brief history of the adoption of provenance graph in threat detection.

For example, control flow graphs (CFGs) (Petroni and Hicks, 2007; Venkatasubramanian et al., 2003) and abstract syntax trees (ASTs) (Li et al., 2019; Ndichu et al., 2019) can effectively model the structure and behavior of programs and are therefore widely used for program analysis and malware detection. Besides, Bayesian attack graph (Frigault and Wang, 2008; Muñoz-González et al., 2016; Wu et al., 2012) can quantify the risks and vulnerabilities in the system to measure the system's security. Petri net (Xu and Nygard, 2006) is a well-known operational model for formal analysis of control and composition of the distributed system. It can formally analyze the security of the system with considerable overhead.

However, none of the above approaches can effectively model information flow between system-level objects with acceptable overhead, which is critical for system-level threat detection. Therefore, in this paper, we focus on the system-level provenance graph, which can not only track the information flow but also support correlation analysis naturally. It is believed to be the next generation of detection technology.

3.4. *Survey methodology*

Multiple databases are used to conduct this survey. There are many keywords relevant to our topic, including provenance, causality, audit, logging, detection, forensic, investigation, apt, reduction, collection, etc. However, these keywords are also widely used in other fields. Thus, searching with a single keyword does not work well. As a first step, we searched for several sets of keywords on Google Scholar, including "provenance + causality + collection", "provenance + causality + reduction", and "provenance + causality + detection", which corresponded to three sub-modules in the typical design.

Nevertheless, the first round of search with keywords combination will miss lots of related works. Thus, we adopted a knowledge graph tool for research papers, namely, Connected Papers (con, 2020.3). This tool is able to search related papers according to not only the citation tree but also the co-citation and bibliographic coupling. It will construct a knowledge graph for every input paper, with this basis, we are able to find lots of related works. Finally, a number of articles are located with snowball methods.

4. Data collection module

As the first step, security analyzers need to deploy collectors on target hosts to collect provenance information. Generally, there are two kinds of collectors: The coarse-grained collectors that focus on system-level information flow, such as file reads, inter-process communication, and so on; and fine-grained collectors that involve intra-process information flow tracking. We will comparatively introduce the design and mechanism of these two kinds of collectors in Section 4.1 and Section 4.2.

4.1. *Coarse-grained provenance collection*

Coarse-grained data collectors only track the provenance between system-level objects, also called system-level collectors. The system-level provenance can be obtained from mul-

Table 2 – Comparison of provenance data collection approaches.

	O/H	Acc.	Granularity	Other Requirements
System-level (Gehani and Tariq, 2012; Pasquier et al., 2017)	Low	Low	Coarse	None
Execution Partition (Lee et al., 2013a; Ma et al., 2015; 2017; 2016; Yang et al., 2020)	Low	Mid	Mid	Instrumentation
Causality Inference (Hassan et al., 2018; Kwon et al., 2016; 2018)	Mid	Mid	Mid	Training or Dual-Exectuion
Taint Analysis (Ji et al., 2017; 2018; Kemerlis et al., 2012)	High	High	Fine	Tainting Framework
Multi-layer (Hassan et al., 2020)	Low	Low	Coarse	Static Analysis

tiple different sources. Most of today's operating systems have built-in audit system, which can provide necessary information flow among system-level objects. There are also third-party collectors, such as FUSE (fus, 2020). CamFlow (Pasquier et al., 2017) adopts LSM (Schaufler, 2016) and NetFilter (net, 2020) to hook kernel objects' security data structure on Linux. Ma et al. proposed (Ma et al., 2015) to obtain system event from windows built-in auditing system ETW (ETW, 2020.3). SPADE (Gehani and Tariq, 2012) provides multiple collector modules for different systems, for example, hooking system call through Auditd (aud, 2020.3) on Linux and MacFUSE (OSX, 2020) on Mac OS, etc.

For different operation system and audit tools, the event list could be different. For Linux, all objects are abstracted as files. Table 1 shows the simplest provenance events list. For windows, reading and writing to the registry is important. However, such extension is trivial and will not affect later data management and detection too much. W3C ProvDM (pro, 2020) provide more specific definition. In practice, security analyzer should customize the events list to reach a balance between overhead and functionality.

4.2. *Fine-grained provenance collection*

One common challenge for causality tracking with provenance graph is the "Dependence Explosion" problem, which causes a large number of benign nodes marked as malicious and brings a lot of computing overhead and human labor. Specifically, for a provenance node with m input edges and n output edges, there could be as much as $m \times n$ possible information flows. Fine-grained provenance collectors can solve the "Dependence Explosion" problem fundamentally by associating inputs and outputs more accurately. Ideally, the number of information flow can be reduce to $m + n$. Thus, researchers proposed lots of approaches to collect fine-grained provenance, as shown in Table 2.

Taint analysis that can accurately track information flow within processes are widely used to prevent information leak or zero-day attacks (Clause et al., 2007; Enck et al., 2014; Newsome and Song, 2005; Xu et al., 2006). By combining inter-process provenance analysis and intra-process analysis, researchers (Ji et al., 2017; 2018; Kemerlis et al., 2012) are able to accurately track the information flow. However, taint analysis introduces significant overhead, slowing down programs by 2× to 10× or more.

Excessive overhead makes Taint infeasible for large-scale threat detection. To reduce the overhead, Ma et al. (2015) first proposed execution partition-based approach. They figure out that taint analysis, which tracking information flow between variables, is too fine-grain and not necessary to build causality connection between inputs and outputs. Thus, they try to find a middle ground between coarse-grain processes and fine-grain variables, called *unit*. Many later works (Lee et al., 2013a; Ma et al., 2017; 2016; Yang et al., 2020) adopt a similar idea. All these works make a different assumption about what kind of unit the causality should be maintained in. For example, Ma et al. (2015) believes that processes can be split into many main loops, and each loop completes a task. Thus, the causality relationship will only be built in the loop. However, such assumptions do not always hold, and these approaches either need extra infrastructure or support from vendors.

Besides improving the accuracy of information flow tracking, causality inference can also effectively reduce false positives. Kwon et al. proposed dual execution-based causality inference (Kwon et al., 2016; 2018). By comparing the output buffer contents of the master and slave at the sink(s), they can determine if the sink(s) are causally dependent on the source(s). Hassan et al. proposed Winnower (Hassan et al., 2018) that tries to infer the connection by training a model to succinctly summarize the behavior of many nodes.

After this module, the collected provenance information can be transmit directly to detection module (Section 6) or through a data management module (Section 5) first.

5. Data management module

Ubiquitously monitoring system in an organization or enterprise will generate massive amount of data. An ideal data management module should consider how to reduce storage cost while providing effective query interface. In this section, we introduce how to design such an ideal data management module from 3 aspects: data storage models (Section 5.1), data reduction algorithms (Section 5.2), and query interface (Section 5.3), and try to answer two research questions, namely, RQ1: How to reduce the size of the data storage as much as possible while maintaining the semantics and RQ2:

How to balance the space efficiency of the prove-nance graph storage with the time efficiency of the query?

5.1. *Data storage models*

The data storage model is the foundation of the whole data management module. The data model used depends on subsequent operations. We will systematically analyze the relationship between different detection algorithms and their corresponding data models in Section 6.

A straightforward idea is to store provenance graph with a graph database. Graph database (gra, 2020) is a widely used NoSQL database, which stores all data as nodes and edges, and provide semantic query interfaces with nodes and edges. Thus, performing graph algorithms, such as backtracking and graph alignment, is relatively easy. However, existing graph database needs to load the whole graph database in the main memory to enable queries. In a large organization, terabytes of data needs to be loaded for a long-running attack campaigns. Even though allocating such large memory is still possible, such approaches incur significant I/O overhead. To mitigate this challenge, the security researchers design detection algorithms (Han et al., 2020; Hossain et al., 2017; Milajerdi et al., 2019a; 2019b) that consume every event in the stream only once, and adopt state stored in cache to represent the event history. Corresponding to the cached graph stored in memory, we call the input of such approaches as *streaming graph*.

Vertex-centric database, built on relational database, store all entries as $< K, V >$ pairs, where K is a identifier representing vertexes (nodes) and V is a list of several entries, such as parents nodes, child nodes, and rules (Xie et al., 2018). Such data model can easily count interaction between nodes, thus widely used in abnormal analysis-based detection systems. Furthermore, relational database can be stored in disk and accelerated with in-memory cache, and thus more feasible than graph database-based approaches.

5.2. *Data reduction algorithms for provenance graphs*

In recent years, more organizations, enterprises, and government agencies suffered from advanced persistent threat (APT) attacks [23-25]. These attacks often have multiple phases and last for quite some time. Moreover, these attacks are often very covert and difficult to detect. It has been reported that the average duration of advanced persistent threat attacks lurking within an enterprise is as long as 188 days [26]. However, the amount of data collected in the provenance graph is extremely large, and the amount of data for a single machine can easily exceed 1GB in one day. Moreover, the number of hosts in a large enterprise or organization can reach tens of thousands. This thus brings significant data storage overhead. At the same time, a massive amount of data also brings great difficulties to subsequent data backtracking. Therefore, the algorithm for compressing the provenance graph is a subject that researchers need to study.

The provenance graph is a special graph whose data mainly includes two parts: nodes (subjects and objects) and edges (events). The essence of the compression of the provenance graph is to remove as many unnecessary nodes and edges as possible while maintaining as much semantics as possible. Specifically, three questions need to be considered: 1) How to define the semantics that needs to be maintained? 2) What is the computational complexity of the compression algorithm? 3) How effective is the compression algorithm? With these three questions in mind, we discuss how to compress nodes and edges, respectively.

In this section, we mainly focus on data reduction methods, which refer to some data reduction principle with a guarantee of limited semantic loss.

5.2.1. *Data reduction for edges*

In a typical operating system, processes and file objects will exist for a while and generate lots of operations between them. Thus, the number of edges is much larger than that of nodes in most provenance graphs, especially for long-running systems. Data reduction algorithms for edges shall introduce higher data reduction ratios than the algorithms for nodes.

Data reduction approaches need to handle the trade-off between data compression ratio and semantic retention. It is almost impossible to prune data without losing any semantics. Thus, researchers should consider how much semantics should be preserved after data reduction. Causality-preserving reduction approach (Xu et al., 2016) and dependency-preserving reduction approach (Hossain et al., 2018) are proposed to define the loss. A simple and intuitive definition of causality is that the first write to an object will affect the subsequent readings.

Causality-Preserving Reduction (CPR). As we discussed in Section 1, causality analysis is the most commonly used operation in provenance graph. Xu et al. proposed causality-preserving reduction (Xu et al., 2016) that maintains the ability to causality analysis on provenance graphs. A simple and intuitive definition of causality is that the first write to an object will affect the subsequent readings. Thus, to avoid changing the causality between objects, CPR will only remove any repeated writes/reads between a pair of objects with no read/write to the destination object. CPR can completely preserve the topology of the graph, and ensure that most detection algorithms are still valid on the compressed graph. However, the algorithm will lose statistical information, including the access frequency, etc. In real-world scenarios, analyzers should pick reduction algorithms according to the subsequent analysis.

Full Dependence-Preserving Reduction (FDR) and Source Dependence-Preserving Reduction (SDR). As Fig. 4 shows, while CPR preserves the semantics in provenance graphs well, it has limited data reduction ratio. To further compress the provenance graph, Hossain et al. (2018) proposed dependence-preserving data compaction. Dependence-preserving reduction only considers the basic operation on provenance graphs, namely, backward tracking and forward tracking. FDR and SDR rely on global reachability of provenance graphs, which is much more expensive to compute than CPR. To overcome these computational challenges, they proposed versioned dependence graphs, which are widely used to simplify computation produce of provenance (Chavan et al., 2015; noa, 2016).

5.2.2. *Data reduction for nodes*

Some techniques try compressing provenance logs via web graph compression algorithm (Chapman et al., 2008) or de-

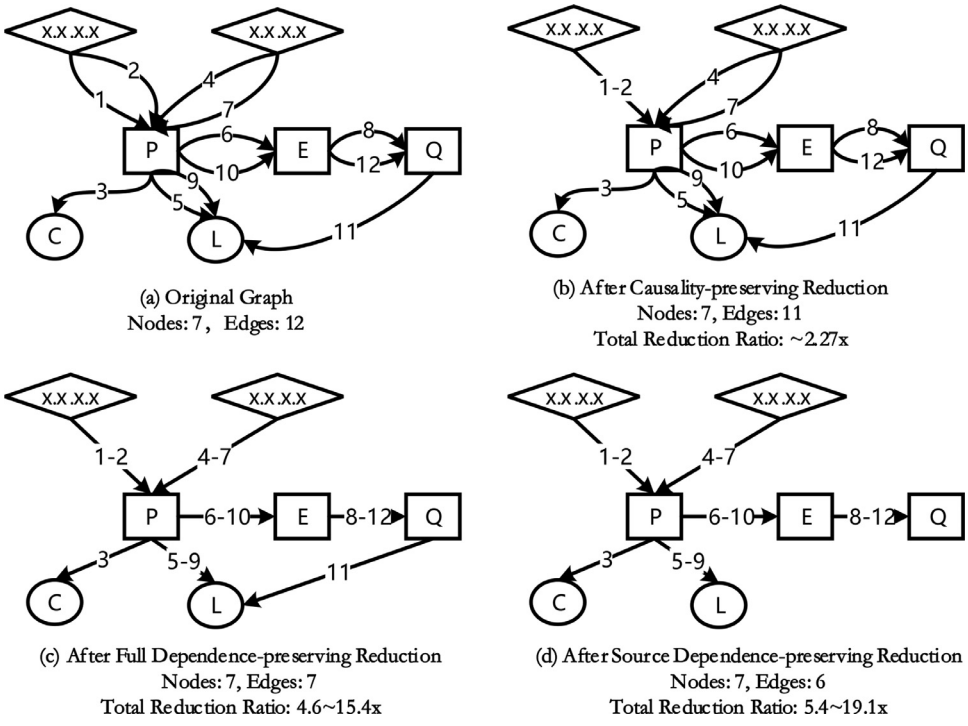


Fig. 4 – Data reduction algorithms for edges.

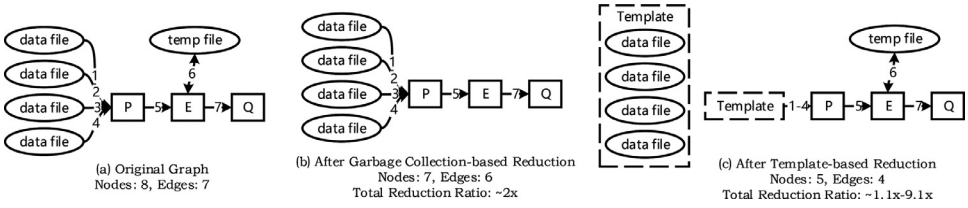


Fig. 5 – Data reduction algorithms for nodes.

tecting common sub-graphs and compressing them (Xie et al., 2012). The main problem of these techniques is that they involve expensive runtime overhead. However, system-level provenance graphs expand quickly. Thus, light-weight compression algorithms are required.

Towards designing efficient compression algorithms, Lee et al. (2013b) designed garbage collecting for provenance, which can locate isolated temporary nodes. Removing these nodes will not affect causality in provenance graphs. Tang et al. proposed nodemerge (Tang et al., 2018), which adopt enhanced FP-growth algorithm to find common access patterns during program initialization. As Fig. 5 shows, the compression ratio of both algorithms is lower than edge-based reduction algorithm.

5.3. *Query interface*

Most detection approaches tend to use naive database query interfaces and fixed data structure to ensure universality. However, for customized attack investigation requirement, the naive query interfaces may not be flexible enough. To fill this research gap, researchers proposed series of provenance graph query systems (Gao et al., 2018a; 2018b; Pasquier et al., 2018; Shu et al., 2018).

These query systems provide investigation capabilities that naive databases cannot provide or require extra effect. These capabilities are list as following:

Causality Tracking. Provenance graphs have strong spatial and temporal properties, which is thus different from ordinary graphs. Backward and forward tracking should take these properties, which are called causality, into consideration. Such tracking operations are common tasks in forensic for root cause discovery and impact analysis (King and Chen, 2003). Almost all the query systems regard the causality tracking as their basic function and provide convenient language or interface support (Gao et al., 2018b; Pasquier et al., 2018; Shu et al., 2018).

Provenance Graph Pattern Matching. Graph pattern matching is at the core of graph query. For threat detection with provenance graph, graph patterns can be used to represent attack behaviors with rich semantics. Thus, pattern matching is equivalent to threat detection. Shu et al. (2018) points out that an ideal pattern matching system should be able to treat patterns as values and compose larger patterns based on

others to enable pattern reuse and abstraction. To accomplish such targets, Shu et al. adopt well-designed query language and typing system.

Stream-based Query. Threat detection is a time-critical mission. To reduce the delay between the attack and the investigation and response, Gao proposed SAQL (Gao et al., 2018a), which is able to take real-time event feed aggregated from multiple hosts as input and provide rich interface. They built the query engine on the top of Siddhi (Sid, 2020) to leverage its mature stream management engine. To tackle the scalability challenge, they designed a master-dependent query scheme that identifies compatible queries and groups them to use a single copy.

Anomaly Analysis. Security log auditing and threat detection rely heavily on expert experience. In order to adopt domain knowledge from expert to express anomalies, Gao provides a domain-specific query language, SAQL (Gao et al., 2018a), which allows analysts to express models for (1) rule-based anomalies, (2) time-series anomalies, (3) invariant-based anomalies, and (4) outlier-based anomalies.

All in all, the query systems provide analysts with a thorough attack investigation capability. These systems typically build on mature stream processing system or database, but take provenance graphs' special properties into consideration with specifically-designed data model and query language.

6. Threat detection module

Using the traceability diagram, security analysts can link causal events and entities in the host to obtain a good abstraction ability, which can well describe the data flow and control flow in the system. In order to connect multiple points involved in an attack, the simplest method is to backtrack [22, 30]. However, the simple backtracking algorithm is difficult to distinguish normal data flow from malicious control flow. There is a problem of dependence explosion, so the accuracy is very low. In order to solve this problem and provide a real-time, efficient, and low false positive threat detection system, researchers have proposed many different schemes. In this section, we first give several threat models (Section 6.1) commonly used in threat detection research using provenance graphs. Then we give a comparison of the existing intrusion detection systems and try to answer two of the research questions we summarize in Section 2.2: RQ3: How to design an efficient and robust intrusion detection algorithm and balance the true-positives and false positives? and RQ4: How to shorten the response time of detection or traceability forensics as much as possible?

6.1. *Attack models*

6.1.1. *Multi-stage APT attack (APT) model*

A large part of threat detection using a provenance graph aims at detecting advanced persistent threat (APT) attacks. APT attacks have the characteristics of advancedness, complexity, concealment, and persistence. Typical APT attacks can be divided into multiple stages, as ATT&CK Metrics (Mit, 2020) shows. Every stage has a particular target and a variety of different technologies to achieve the target. Real-world attacks usually involve three or more stages. Thus, even if missing some stages, security analyzers can still identify a threat and complete the missing piece with digital forensic techniques. Meanwhile, analyzers can also adopt the multi-stage feature to filter out false alerts.

6.1.2. *Information leakage (leakage) model*

The information leakage model assumes that the attackers are able to take control the entire target system. The goal is to pass the specified sensitive information to endpoints controlled by the attacker in various ways. A large part of APT attacks is also aimed at information leakage. However, unlike the multi-stage APT attack model, the information leakage model does not focus on specific attack technologies, but focuses on the information flow in the system, and continuously monitors whether sensitive information flows to unauthorized points.

6.1.3. *General attack (general) model*

General attacks are much more diverse. There are low and stealth attacks like APT but also quick and overt attacks such as ransomware. The target could be stealing information but also pure destruction. Thus, more general and detailed attack models are required to detect such attacks.

6.2. *Threat detection and investigation system design*

Provenance graphs are able to link events in system with causality, regardless of the time between events, which thus have a overall view of entire attacks. Backtracking, proposed by King (King and Chen, 2003), is the earliest and most fundamental attack investigation method on provenance graph. Given a detection point, backtracking is able to traverse the whole historical context of system execution. However, naive backtracking requires complete provenance graph and too much human intervention, which thus is neither timely nor efficient.

An ideal threat detection system needs to consider three attributes at the same time: fast response, high efficiency, and high accuracy. However, the size of a provenance graph, even pruned, is very large. Therefore, threat detection on provenance graphs could introduce high space and computing overhead. In order to find a balance between the three attributes, researchers have made many attempts. These approaches can be divided into 3 categories according to the main detection design.

Firstly, *tag propagation-based approaches* (Hossain et al., 2017; Milajerdi et al., 2019a) try to store system execution history incrementally in tags and utilize tag propagation process to trace the causality. These algorithms have roughly linear time complexity. Moreover, they can take streaming graph as input and respond fast. Secondly, *abnormal detection* (Hassan et al., 2019; Liu et al., 2018; Xie et al., 2018; 2019) try to identify abnormal interaction between nodes. Thus, these approaches will model normal behaviors by collecting historical data or data from parallel systems. Finally, *graph matching-based approaches* (Han et al., 2020; Liu et al., 2019; Milajerdi et al., 2019a) try to identify suspicious behavior by matching sub-structure in graphs. However, graph matching is computational complex.

Table 3 – Taxonomy of existing provenance graph-based threat detection system designs.

Approaches	Attack Models	Detection Models	Data Models	Alert Detection	Alert Correlration	Response Time	Overhead	True Positive	False Positive
Back-tracking (King and Chen, 2003)	General	Naive Backtracking	Cached Graph	✗	√	Long	Mid	-	High
HERCULE (Pei et al., 2016)	General	Community Detection	Cached Graph	✗	√	Long	Low	-	High
POIROT (Milajerdi et al., 2019a)	APT	Graph Alignment	Streaming Graph	√	√	Short	Mid	Mid	Low
Log2vec (Liu et al., 2019)	General	Graph Embedding	Cached Graph	√	✗	Long	Low	Mid	Mid
ProvDetector (Wang et al., 2020)	General	Graph Embedding	Cached Graph	√	✗	Long	Low	Mid	Mid
UNICORN (Han et al., 2020)	APT	Graph Sketch Cluster	Cached Graph	√	✗	Mid	Low	High	High
PrioTracker (Liu et al., 2018)	APT	Anomaly Scores	Cached Graph	√	✗	Mid	Low	Mid	Mid
NoDoze Hassan et al. (2019)	APT	Anomaly Scores	Vertex-centric DB	√	✗	Mid	Low	Mid	Mid
P-gaussian (Xie et al., 2019)	APT	Anomaly Scores	Vertex-centric DB	√	✗	Mid	Mid	Mid	Mid
Pagoda (Xie et al., 2018)	APT	Anomaly Scores	Vertex-centric DB	√	✗	Mid	Low	Mid	Mid
SWIFT (Ul Hassan et al., 2020)	APT	Anomaly Scores	Vertex-centric DB	√	✗	Mid	Low	Mid	Mid
Coloring (Jiang et al., 2006)	General	Process Coloring	Cached Graph	✗	√	Long	Low	-	High
SLEUTH (Hossain et al., 2017)	Leakage	Tag Propagation	Streaming Graph	√	✗	Short	Mid	High	High
HOLMES (Milajerdi et al., 2019b)	APT	Tag Propagation	Streaming Graph	√	√	Short	Low	Mid	Low
MORSE (Hossain et al., 2020)	APT	Tag Propagation	Streaming Graph	√	√	Short	Low	Mid	Low

Researchers try to extract the graphs' features with graph embedding or graph sketch algorithm or use approximate methods.

As shown in Table 3, the target attack models, fundamental detection algorithms, and data management model affect each other and basically determine the design of the detection system. We will compare the system properties according to the ideal system properties introduced in Section 2.2.

6.2.1. *Graph matching-based detection*

The graph representation ensures the adversarially robustness of provenance graph-based detection approaches. The connections between nodes indicate the relationship between system entities. Nodes close to each other are more likely to serve the same function. Thus, utilizing community detection algorithm, analysts are able to correlate nodes in the same attack scenarios. Substructures in a provenance graph can completely describe the malicious behavior. Therefore, it is a very straightforward idea to detect by graph matching. However, graph matching is NP-complete problem (De Nardo et al., 2008). Thus, researchers have proposed many approximate methods.

Milajerdi et al. proposed POIROT (Milajerdi et al., 2019a) and the key online graph alignment algorithm. Utilizing query graph manually extracted from threat intelligence and the graph alignment algorithm, they could locate threats in provenance graph quickly. However, extracting query graphs requires a lot of manual work. Thus, it is difficult to cover all kind of advanced attacks in various forms.

Graph embedding are widely used to extract graph features into vertexs while maximally preserving properties like graph structure and information (Goyal and Ferrara, 2018; Wang et al., 2014; Yan et al., 2006). Utilizing the graph embedding, researchers can effectively and efficiently detect threats by separating malicious and benign log entries into different clusters and identifying malicious ones (Liu et al., 2019; Wang et al., 2020). However, such methods typical work on cached graph, so the response is slower; meanwhile, it requires a lot of training data, so it is not suitable for advanced attacks.

To tackle the above two challenges, Han et al. proposed UNICORN (Han et al., 2020), which adopts a historical graph

sketch approach to build an incrementally updatable, xed size, longitudinal graph data structure. So, they can find threats when the graph structure changed. However, this is an anomaly detection-based approaches, which thus suffers from the limitation of anomaly detection.

6.2.2. *Anomaly score-based detection*

Anomaly score-based detection tries to quantify the suspiciousness of each edge between node pairs. Using historical statistics, researchers can find abnormal access in system. Specifically, Pagoda (Xie et al., 2018) takes into account the anomaly degree of both a single provenance path and the whole provenance graph. Their subsequence work P-Gaussian (Xie et al., 2019) can detect variants using gaussian distribution scheme. PrioTracker (Liu et al., 2018) and NoDoze (Hassan et al., 2019) adjust the events' suspiciousness based on its nighber's suspiciousness.

Compared with graph-based anomaly detection, anomaly score-based detection has much less parameters to tune, which thus is much easier to implement and deploy. Meanwhile, anomaly score-based detection typically adopts a vertex-centric relational database, which is much faster than graph database.

6.2.3. *Tag propagation-based detection*

Tag propagation-based detection can be divided into two phases, namely, tag initialization and tag propagation. In tag initialization phase, tags are assigned to nodes. The amount of nodes is much less than edges. Thus, storing and updating tags is efficient. In tag propagation phase, tags are passed along the edge according to the pre-designed rules. In this phase, different tags could meet at the same node and triage future calculations together.

Process coloring proposed by Jiang et al. (2006) is a simplified tag-based approach. In the tag initialization phase, tags (colors) are assigned to each remotely-accessible server or process. Then, in the tag propagation phase, tags can be inherited by spawned child processes or diffused indirectly through process actions. As a result, analysts can quickly identify the break-in point without tedious backtracking.

Follow-up works adopt more complex tag design to implement more functions. SLEUTH (Hossain et al., 2017) utilizes two types of tags, namely, trustworthiness tags (t-tags) and condentiality tags (c-tags), to implement a policy enforcement framework. In short, an alarm is triggered when a node with low trustworthiness accesses a node with high confidentiality. Specifically, in the tag initialization phase, t-tags and c-tags are assigned to the nodes according to the predefined trustworthiness and confidentiality respectively. In the tag propagation phase, the trustworthiness and confidentiality are propagated, and the accesses that violate the policy will be captured.

However, tag propagation-based approaches also suffer from the "dependency explosion" problem. Without extra control, single tag can spread to everywhere and cause a lot of false positives. To tackle this challenge, Milajerdi et al. proposed HOLMES (Milajerdi et al., 2019b), which raises the detection threshold by requiring the aggregation of more tags. In the tag initialization phase, HOLMES assigns fewer tags only to process with suspicious behaviors. These suspicious behaviors contain lots of false positives. Thus, in the tag propagation phase, HOLMES requires multiple tags to aggregate and reach a pre-defined threshold, and then triage the alert. Another way to avoid the dependency explosion problem is to make the impact decrease as the number of transmission rounds increases. MORSE (Hossain et al., 2020) achieves this with tag decay and tag attenuation techniques.

All in all, tag propagation-based approaches have the following advantages. Firstly, tag initialization and propagation processes replace computationally expensive graph matching algorithm and lower the overhead. Secondly, tag propagation-based approaches take one event at a time and update states correspondingly, which thus can support streaming graph input naturally and can respond quickly. Last but not least, the information stored in tags can be used to locate the point involved in the intrusion quickly, and thus avoid the tedious backtracking algorithm.

7. Discussion

In this section, we discuss provenance-based threat detection as a whole from several essential perspectives. First, we compare the effects of different combinations of the data and detection model on the performance metrics. Then, we will describe the prevailing dependence explosion problem and possible solutions. Finally, we discuss how different approaches strike a balance between true-positive and false-positive. Moreover, as summarized in Table 4, multiple insights and challenges will be provided for real-world practice and future studies.

7.1. *How the selection of data models and detection models will affect performance?*

The detection model and the data model are two important parts of the threat detection system, directly determining the performance. As shown in Table 3 **(Insight 1) there are three frequent combinations of two models, namely, "anomaly score + vertex-centric DB," "tag propagation + Streaming graph," and "cached graph + others."**

Caching the data as graphs in the graph database is the most intuitive and convenient way. Almost all detection models work on cached graphs. However, **(Insight 2) it is an inefficient way because of the poor performance of the graph database. Thus, the cached graph is not recommended in practice.** NoDoze et al. (Hassan et al., 2019; Xie et al., 2018; 2019) proposed vertex-centric DB, which is essentially a relational database, as an alternative. While adopting vertex-centric makes access to nodes faster, it also makes access to edges more complicated and slower. Thus, NoDoze et al. adopted anomaly scores-based detection approaches that only need to access nodes' information. Sleuth et al. (Hossain et al., 2017; Milajerdi et al., 2019a; 2019b) choose not to cache the provenance graph. Instead, they process all nodes and edges once, and cache processing results in tags. Moreover, they embed the graph structure information in the tag propagation process. By this means, **(Insight 3) tag propagation-based detec-**

Table 4 – Insights and challenges on provenance graph-based threat detection.

Insight 1	There are common combinations between the detection model and the data model, which enable optimal performance.
Insight 2	The graph database has poor performance. Thus, the cached graph is not recommended in practice.
Insight 3	Tag propagation-based detection can handle the provenance graph as a stream in real-time and thus have the shortest response time.
Insight 4	The dependence explosion problem can be addressed fundamentally by adopting fine-grained data collection methods.
Challenge 1	Existing fine-grained data collection methods involve significant overhead. How to build a low overhead fine-grained collector is still a pressing research problem.
Insight 5	Existing approaches can only mitigate the dependence explosion problem and may involve potential vulnerabilities.
Challenge 2	More efficient and robust algorithm-based solutions are still direly needed for the dependence explosion problem.
Insight 6	For provenance graph-based detection, most existing detection models are sequence-based rather than graph-based.
Challenge 3	Existing sequence-based real-time detection approaches may not be robust enough to distinguish malicious behavior from benign ones accurately. Therefore, it is still necessary to design and implement more robust detection models.
Challenge 4	There is a lacking of unified datasets and data format for provenance graph-based detection.
Challenge 5	There is a lacking of study on potential evasion for provenance graph-based detection.

tion can handle the provenance graph as a stream in real-time and thus have the shortest response time.

7.2. *How to solve the dependence explosion problem?*

As discussed in §6, dependence explosion is a common problem caused by the coarse-grained provenance graph and causality analysis, which will bring extra false-positive and overhead. Hence, **(Insight 4) we can address the dependence explosion problem fundamentally by adopting fine-grained data collection methods, as discussed in Section 4.2. (Challenge 1) However, these methods involve significant runtime or development overhead and, therefore, hard to be utilized in real-world scenarios.**

To mitigating the dependence explosion problem, several algorithm-based approaches are proposed based on different assumptions. Nodoze et al. (Hassan et al., 2019; Xie et al., 2018) assign anomaly scores to each edge based on the frequency with which related events have happened before. Then, the anomaly score will be propagated along the paths. And paths with low anomaly scores will be ignored. Their underlying assumption is that an attack will always involve unusual edges in provenance graphs. However, such an assumption not always holds. One real-world example is the gitpwnd attack (git, 2020), which completes the attack exclusively with the git workflow. Attackers can intentionally avoid unusual dependencies that trigger such detection.

SLEUTH and subsequent works (Hossain et al., 2017; 2020; Milajerdi et al., 2019b) adopt tag decay based approaches. These works try to limit the spread of tags by limiting the number of rounds or time of tag propagation. Their underlying assumption is that the attack will perform the attack as soon as possible. Nevertheless, apparently, attackers can bypass such detection by maintaining stealth for a long time or involving more intermediate nodes to extend the attack chain.

All in all, **(Insight 5) existing algorithm-based approaches can only mitigate the dependence explosion problem and may involve potential vulnerabilities.** Therefore, **(Challenge 2) more efficient and robust algorithm-based solutions are still direly needed for the dependence explosion problem.** One possible solution is to determine the underlying information flow with high-level semantic information, as discussed in Section 4.2.

7.3. *How to balance the true-positive and false-positive?*

True-positive and false-positive are the most critical and fundamental indicators for a detection system. In general, complicated detection models are better at distinguishing malicious and benign behavior and thus having higher accuracy. For example, the multi-stage model utilized by provenance graph-based detection systems can improve accuracy by alert correlation and perform better than the single-point detection model. However, more complicated models tend to have higher overheads as well.

Specifically, **(Insight 6) for provenance graph-based detection, most existing detection models are sequence-based rather than graph-based**, including tag propagation-based approaches (Hossain et al., 2017; 2020; Milajerdi et al., 2019a; 2019b) and most anomaly detection approaches (Hassan et al., 2019; Xie et al., 2018). While graph-based detection approaches (Han et al., 2020; Liu et al., 2019; Pei et al., 2016; Wang et al., 2020) typically have longer response times and higher overhead.

High true-positive and low false-positive are often contradictory when adopting the same detection model. However, security analyzers can still seek a balance between them through a combination of techniques and parameter tuning. For example, HOLMES (Milajerdi et al., 2019b) utilizes relatively simple signatures to cover as many malicious behaviors as possible. Meanwhile, it adopts the alert correlation to filter false alarms. This process involves lots of parameters, whose tuning significantly affects the accuracy and efficiency of systems. Nevertheless, these parameters are often determined empirically, which makes the detection results not stable. In comparison, POIROT (Milajerdi et al., 2019a) uses more sophisticated signatures to avoid false-positive. To improve the coverage, they need to collect a signature database from massive real-world threat intelligence. However, such an approach still cannot detect previously unseen malicious behaviors.

(Challenge 3) Existing sequence-based real-time detection approaches may not be robust enough to distinguish malicious behavior from benign ones accurately. Therefore, it is still necessary to design and implement more robust detection models.

7.4. *Other practical challenges*

Challenge 4: Lacking of unified datasets and data format. Unified datasets and data format can significantly lower the barriers for further research, reproduction, and quantitative comparison. However, as far as we know, the only publicly available dataset for provenance graph-based detection is the Engagement 3 and 5 datasets from the Transparent Computing program (Darpa, 2015). Most existing work has to rely on limited self-collected attack data. These datasets only contain dozens of attacks, which can hardly represent various sophisticated attacks in the real-world. Thus, it is claimed that a unified dataset and data format are direly needed.

Challenge 5: Lacking of study on potential evasion. Anti-evasion is a core competency for detection systems. Research into the potential evasion problem is essential for new detection mechanisms, making the detection result more reliable. However, such studies are still missing for system-level provenance graph-based detection.

7.5. *An ideal detection approach*

Synthesizing the above discussion, we propose what an ideal system should like:

- **A real-time approach:** An ideal detection system should have the lowest possible overhead and the shortest possible response time. Therefore, the system must be able to process streaming provenance graphs without caching too much data. From a performance perspective, tag propagation-based approaches are the best.
- **A robust and effective approach:** For a detection system, robustness and effectiveness means that it needs to distinguish malicious behavior from benign ones accurately in any case. That is to say, the detection model should be complicated enough to demonstrate the difference between malicious behavior from benign ones. From this point of view, the graph-based modeling approach is better than the others.

Specifically, we can try to design and implement such a system by answering the research questions mentioned in Section 2.2 and the following the pace of existing work detailed described in Sections 5 and 6.

8. Conclusion

As a system behavior abstraction tool, provenance graphs are widely accepted for endpoint threat detection. In this paper, we present the typical system architecture for provenance graph-based threat detection. Then, we systematically introduced and compared techniques choice involved and concluded existing research challenges for future study.

Declaration of Competing Interest

None.

Acknowledgment

We would like to thank the anonymous reviewers for providing valuable feedback on our work. This work is supported by the Joint Funds of the National Natural Science Foundation of China (U1936215) and the Zhejiang Lab's International Talent Fund for Young Professionals.

REFERENCES

Advanced, 2020. persistent threat.
Axelsson S. In: Technical Report. Intrusion Detection Systems: a Survey and Taxonomy; 2000.
Barre M, Gehani A, Yegneswaran V. In: 11th International Workshop on Theory and Practice of Provenance (TaPP 2019). Mining data provenance to detect advanced persistent threats; 2019.
Buczak AL, Guven E. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surv. Tutor. 2016;18(2):1153–76. doi:10.1109/COMST.2015.2494502.
Buneman P, Khanna S, Wang-Chiew T. Why and where: a characterization of data provenance. In: International Conference on Database Theory. Springer; 2001. p. 316–30.
Connected, 2020.3. papers.
Chapman AP, Jagadish HV, Ramanan P. Efficient provenance storage. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data; 2008. p. 993–1006.
Chavan A, Huang S, Deshpande A, Elmore A, Madden S, Parameswaran A. In: 7th USENIX Workshop on the Theory and Practice of Provenance (TaPP 15). Towards a unified query language for provenance and versioning. USENIX Association; 2015.
Clause J, Li W, Orso A. Dytan: a generic dynamic taint analysis framework. In: Proceedings of the 2007 International Symposium on Software Testing and Analysis; 2007. p. 196–206.
De Nardo L, Ranzato F, Tapparo F. The subgraph similarity problem. IEEE Trans. Knowl. Data Eng. 2008;21(5):748–9.
Darpa, 2015.2. Darpa transparent computing.
Event, 2020.3. tracing for windows.
Edge ME, Falcone Sampaio PR. A survey of signature based methods for financial fraud detection. Comput. Secur. 2009;28(6):381–94. doi:10.1016/j.cose.2009.02.001.
Enck W, Gilbert P, Han S, Tendulkar V, Chun B-G, Cox LP, Jung J, McDaniel P, Sheth AN. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM Trans. Comput. Syst. (TOCS) 2014;32(2):1–29.
Ficco M. Security event correlation approach for cloud computing. Int. J. High Perform. Comput. Netw. 2013;7(3):173. doi:10.1504/IJHPCN.2013.056525.
Freire J, Koop D, Santos E, Silva C. Provenance for computational tasks: a survey. Comput. Sci. Eng. 2008;10(3):11–21. doi:10.1109/MCSE.2008.79.
Frigault M, Wang L. Measuring network security using Bayesian network-based attack graphs. In: Proceedings - International Computer Software and Applications Conference; 2008. p. 698–703. doi:10.1109/COMPSAC.2008.88.
Graph, 2020. Graph database.
Gao P, Xiao X, Li D, Li Z, Jee K, Wu Z, Kim CH, Kulkarni SR, Mittal P. SAQL: a stream-based query system for real-time abnormal system behavior detection. In: 27th USENIX Security Symposium (USENIX Security 18). Baltimore, MD: USENIX Association; 2018. p. 639–56.

Gao P, Xiao X, Li Z, Jee K, Xu F, Kulkarni SR, Mittal P. AIQL: Enabling Efficient Attack Investigation from System Monitoring Data; 2018. arXiv:1806.02290 [cs]. ArXiv: 1806.02290

Gehani A, Tariq D. In: SPADE: Support for Provenance Auditing in Distributed Environments. Springer-Verlag New York, Inc.; 2012. p. 101–20.

Goyal P, Ferrara E. Graph embedding techniques, applications, and performance: a survey. Knowl.-Based Syst. 2018;151:78–94.

Greenwood, M., Goble, C., Stevens, R., Zhao, J., Addis, M., Marvin, D., Moreau, L., Oinn, T., 2003. Provenance of e-science experiments-experience from bioinformatics.

Han X, Pasquier T, Bates A, Mickens J, Seltzer M. UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats; 2020. arXiv preprint arXiv:2001.01525

Han X, Pasquier T, Seltzer M. In: 10th {USENIX} Workshop on the Theory and Practice of Provenance (TaPP 2018). Provenance-based intrusion detection: opportunities and challenges; 2018.

Hassan WU, Guo S, Li D, Li Z, Chen Z, Jee K, Li Z, Bates A. In: NDSS. NoDoze: combatting threat alert fatigue with automated provenance triage; 2019.

Hassan WU, Lemay M, Aguse N, Bates A, Moyer T. towards scalable cluster auditing through grammatical inference over provenance graphs. Proceedings 2018 Network and Distributed System Security Symposium. San Diego, CA: Internet Society, 2018.

Hassan WU, Noureddine MA, Datta P, Bates A. OmegaLog: High-Fidelity Attack Investigation via Transparent Multi-layer Log Analysis; 2020. p. 16.

Herschel, M., 2017. A survey on provenance: What for? What form? What from?, 26.

Hodge, V. J., Austin, J., 2004. A survey of outlier detection methodologies. 10.1023/B:AIRE.0000045502.10941.a9

Hossain MN, Milajerdi SM, Wang J, Eshete B, Gjomemo R, Sekar R, Stoller S, Venkatakrishnan VN. SLEUTH: real-time attack scenario reconstruction from COTS audit data. In: 26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017.; 2017. p. 487–504.

Hossain MN, Sheikhi S, Sekar R. In: IEEE S&P 2020. Combating dependence explosion in forensic analysis using alternative tag propagation semantics; 2020.

Hossain MN, Wang J, Sekar R, Stoller SD. Dependence-Preserving Data Compaction for Scalable Forensic Analysis; 2018. p. 1723–40.

Husák M, Kašpar J. AIDA framework: real-time correlation and prediction of intrusion detection alerts. In: ACM International Conference Proceeding Series. New York, New York, USA: Association for Computing Machinery; 2019. p. 1–8. doi:10.1145/3339252.3340513.

Ji Y, Lee S, Downing E, Wang W, Fazzini M, Kim T, Orso A, Lee W. RAIN: refinable attack investigation with on-demand inter-process information flow tracking. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS '17. Dallas, Texas, USA: ACM Press; 2017. p. 377–90. doi:10.1145/3133956.3134045.

Ji Y, Lee S, Fazzini M, Allen J, Downing E, Kim T, Orso A, Lee W. Enabling refinable cross-host attack investigation with efficient data flow tagging and tracking. In: 27th USENIX Security Symposium (USENIX Security 18). Baltimore, MD: USENIX Association; 2018. p. 1705–22.

Jiang X, Walters A, Xu D, Spafford EH, Buchholz F, Wang Y-M. Provenance-aware tracing of worm break-in and contaminations: a process coloring approach. In: 26th IEEE International Conference on Distributed Computing Systems (ICDCS'06); 2006. p. 38. doi:10.1109/ICDCS.2006.69.

Kemerlis VP, Portokalidis G, Jee K, Keromytis AD. Libdft: practical dynamic data flow tracking for commodity systems. In: Proceedings of the 8th ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments. New York, NY, USA: ACM; 2012. p. 121–32. doi:10.1145/2151024.2151042.

King ST, Chen PM. Backtracking intrusions. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles. New York, NY, USA: ACM; 2003. p. 223–36. doi:10.1145/945445.945467.

Kwon Y, Kim D, Sumner WN, Kim K, Saltaformaggio B, Zhang X, Xu D. LDX: causality inference by lightweight dual execution. In: Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems. New York, NY, USA: ACM; 2016. p. 503–15. doi:10.1145/2872362.2872395.

Kwon Y, Wang F, Wang W, Lee KH, Lee W-C, Ma S, Zhang X, Xu D, Jha S, Ciocarlie G, Gehani A, Yegneswaran V. MCI : Modeling-based Causality Inference in Audit Logging for Attack Investigation. Internet Society; 2018. doi:10.14722/ndss.2018.23306.

Linux, 2020. fuse.

Linux,, 2020.3. auditd.

Lee KH, Zhang X, Xu D. High accuracy attack provenance via binary-based execution partition. In: NDSS; 2013. p. 16.

Lee KH, Zhang X, Xu D. LogGC: garbage collecting audit log. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. New York, NY, USA: ACM; 2013. p. 1005–16. doi:10.1145/2508859.2516731.

Li Z, Chen Y, Chen Q, Zhu T, Xiong C, Yang H. Effective and light-weight deobfuscation and semantic-aware attack detection for powershell scripts. Proceedings of the ACM Conference on Computer and Communications Security, 2019.

Liu F, Wen Y, Zhang D, Jiang X, Xing X, Meng D. 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. New York, NY, USA: ACM; 2019. p. 1777–94. doi:10.1145/3319535.3363224. Event-place: London, United Kingdom

Liu Y, Zhang M, Li D, Jee K, Li Z, Wu Z, Rhee J, Mittal P. Towards a Timely Causality Analysis for Enterprise Security. Internet Society; 2018. doi:10.14722/ndss.2018.23254.

Mitre, 2020. att&ck metrics.

Ma S, Lee KH, Kim CH, Rhee J, Zhang X, Xu D. Accurate, low cost and instrumentation-free security audit logging for windows. In: Proceedings of the 31st Annual Computer Security Applications Conference. New York, NY, USA: ACM; 2015. p. 401–10. doi:10.1145/2818000.2818039.

Ma S, Zhai J, Wang F, Lee KH, Zhang X, Xu D. MPI: multiple perspective attack investigation with semantic aware execution partitioning. In: 26th USENIX Security Symposium (USENIX Security 17). Vancouver, BC: USENIX Association; 2017. p. 1111–28.

Ma S, Zhang X, Xu D. ProTracer: towards practical provenance tracing by alternating between logging and tainting. Internet Society; 2016. doi:10.14722/ndss.2016.23350.

Milajerdi SM, Eshete B, Gjomemo R, Venkatakrishnan V. POIROT: aligning attack behavior with kernel audit records for cyber threat hunting. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: ACM; 2019. p. 1795–812. doi:10.1145/3319535.3363217. Event-place: London, United Kingdom

Milajerdi SM, Gjomemo R, Eshete B, Sekar R, Venkatakrishnan V. HOLMES: real-time apt detection through correlation of suspicious information flows. In: 2019 IEEE Symposium on Security and Privacy (SP); 2019. p. 1137–52. doi:10.1109/SP.2019.00026. ISSN: 1081-6011

Miles S, Groth P, Branco M, Moreau L. The requirements of using provenance in e-science experiments. J. Grid Comput. 2007;5(1):1–25.

Mitchell, R., Chen, I. R., 2014. A survey of intrusion detection techniques for cyber-physical systems. 10.1145/2542049

Modi, C., Patel, D., Borisaniya, B., Patel, A., Rajarajan, M.,. A survey of intrusion detection techniques in cloud. J. Netw. Comput. Appl. 36 (1), 42–57. 10.1016/j.jnca.2012.05.003

Muñoz-González L, Sgandurra D, Paudice A, Lupu EC. Efficient attack graph analysis through approximate inference. ACM Trans. Privacy Secur. 2016;20(3).

nccgroup, 2020. /gitpwnd: Gitpwnd is a network penetration tool that lets you use a git repo for command and control of compromised machines.

Net, 2020. filter.

Naughton T, Bland W, Vallee G, Engelmann C, Scott SL. Fault injection framework for system resilience evaluation: fake faults for finding future failures. In: Proceedings of the 2009 Workshop on Resiliency in High Performance; 2009. p. 23–8.

Ndichu S, Kim S, Ozawa S, Misu T, Makishima K. A machine learning approach to detection of JavaScript-based attacks using AST features and paragraph vectors. Appl. Soft Comput. J. 2019;84:105721. doi:10.1016/j.asoc.2019.105721.

Newsome J, Song DX. Dynamic taint analysis for automatic detection, analysis, and signaturegeneration of exploits on commodity software., 5. Citeseer; 2005. p. 3–4.

Provenance-aware. In: 8th USENIX Workshop on the Theory and Practice of Provenance (TaPP 16). versioned dataworkspaces. USENIX Association; 2016.

Osx, 2020. fuse.

Pasquier T, Han X, Goldstein M, Moyer T, Eyers D, Seltzer M, Bacon J. Practical whole-system provenance capture. In: Proceedings of the 2017 Symposium on Cloud Computing. Santa Clara, California: Association for Computing Machinery; 2017. p. 405–18. doi:10.1145/3127479.3129249.

Pasquier T, Han X, Moyer T, Bates A, Hermant O, Eyers D, Bacon J, Seltzer M. Runtime analysis of whole-system provenance. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security - CCS '18. Toronto, Canada: ACM Press; 2018. p. 1601–16. doi:10.1145/3243734.3243776.

Pei K, Gu Z, Saltaformaggio B, Ma S, Wang F, Zhang Z, Si L, Zhang X, Xu D. HERCULE: attack story reconstruction via community discovery on correlated log graph. In: Proceedings of the 32Nd Annual Conference on Computer Security Applications. New York, NY, USA: ACM; 2016. p. 583–95. doi:10.1145/2991079.2991122.

Petroni NL, Hicks M. Automated detection of persistent kernel control-flow attacks. In: Proceedings of the ACM Conference on Computer and Communications Security. New York, New York, USA: ACM Press; 2007. p. 103–15. doi:10.1145/1315245.1315260.

Prasad NR, Almanza-Garcia S, Lu TT. Anomaly detection. Comput. Mater. Contin. 2009;14(1):1–22. doi:10.1145/1541880.1541882.

Schaufler, C., 2016. Lsm: Stacking for major security modules.

Siddhi, 2020. complex event processing engine.

Symantec, 2020. internet security threat report.

Shu X, Araujo F, Schales DL, Stoecklin MP, Jang J, Huang H, Rao JR. Threat intelligence computing. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: ACM; 2018. p. 1883–98. doi:10.1145/3243734.3243829.

Simmhan YL, Plale B, Gannon D. A survey of data provenance in e-science. ACM Sigmod Record 2005;34(3):31–6.

Tang Y, Li Q, Li D, Li Z, Zhang M, Jee K, Xiao X, Wu Z, Rhee J, Xu F. 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. Toronto, Canada: ACM Press; 2018. p. 1324–37. doi:10.1145/3243734.3243763.

Teresa F. Lunt. A survey of intrusion detection techniques. Comput. Secur. 1993:405–518.

Ul Hassan W, Li D, Jee K, Yu X, Zou K, Wang D, Chen Z, Li Z, Gui J, Bates A, Gui J-i. In: ACSAC 2020. This is why we can't cache nice things: lightning-fast threat hunting using suspicion-based hierarchical storage. ACM; 2020. doi:10.1145/3427228.3427255.

Venkatasubramanian R, Hayes JP, Murray BT. Low-cost on-line fault detection using control flow assertions. In: Proceedings - 9th IEEE International On-Line Testing Symposium, IOLTS 2003. Institute of Electrical and Electronics Engineers Inc.; 2003. p. 137–43. doi:10.1109/OLT.2003.1214380.

W3c, 2020. prov-dm.

Wang Q, Hassan WU, Li D, Jee K, Yu X, Zou K, Rhee J, Chen Z, Cheng W, Gunter CA, Chen H. You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis; 2020. p. 17.

Wang Z, Zhang J, Feng J, Chen Z. Knowledge graph embedding by translating on hyperplanes. Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.

Woodruff A, Stonebraker M. Supporting fine-grained data lineage in a database visualization environment. In: Proceedings 13th International Conference on Data Engineering. IEEE; 1997. p. 91–102.

Wu J, Yin L, Guo Y. Cyber attacks prediction model based on Bayesian network. In: Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS; 2012. p. 730–1. doi:10.1109/ICPADS.2012.117.

Xie Y, Feng D, Hu Y, Li Y, Sample S, Long D. Pagoda: a hybrid approach to enable efficient real-time provenance based intrusion detection in big data environments. IEEE Trans. Dependable SecurComput. 2018:1. doi:10.1109/TDSC.2018.2867595.

Xie Y, Feng D, Tan Z, Chen L, Muniswamy-Reddy K-K, Li Y, Long DD. A hybrid approach for efficient provenance storage. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management; 2012. p. 1752–6.

Xie Y, Wu Y, Feng D, Long D. P-Gaussian: provenance-based gaussian distribution for detecting intrusion behavior variants using high efficient and real time memory databases. IEEE Trans. Dependable Secur.Comput. 2019:1. doi:10.1109/TDSC.2019.2960353.

Xu D, Nygard K. Threat-driven modeling and verification of secure software using aspect-oriented Petri nets. IEEE Trans. Softw. Eng. 2006;32(4):265–78. doi:10.1109/tse.2006.40.

Xu W, Bhatkar S, Sekar R. Taint-enhanced policy enforcement: a practical approach to defeat a wide range of attacks.. In: USENIX Security Symposium; 2006. p. 121–36.

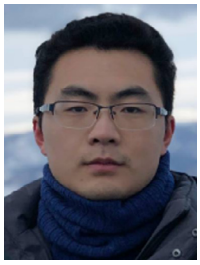
Xu Z, Wu Z, Li Z, Jee K, Rhee J, Xiao X, Xu F, Wang H, Jiang G. High fidelity data reduction for big data security dependency analyses. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: ACM; 2016. p. 504–16. doi:10.1145/2976749.2978378.

Yan S, Xu D, Zhang B, Zhang H-J, Yang Q, Lin S. Graph embedding and extensions: a general framework for dimensionality reduction. IEEE Trans. Pattern Anal. Mach.Intel. 2006;29(1):40–51.

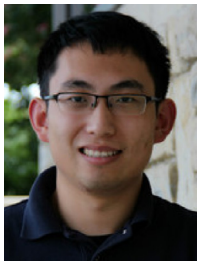
Yang, R., Ma, S., Xu, H., Zhang, X., Chen, Y., 2020. Uiscope: accurate, instrumentation-free, and visible attack investigation for GUI applications.

Yu Y. A survey of anomaly intrusion detection techniques. J. Comput. Sci. Coll. 2012. doi:10.5555/2379703.

Zafar F, Khan A, Suhail S, Ahmed I, Hameed K, Khan HM, Jabeen F, Anjum A. Trustworthy data: a survey, taxonomy and future trends of secure provenance schemes. J. Netw. Comput. Appl. 2017;94:50–68. doi:10.1016/j.jnca.2017.06.003.



Zhenyuan Li received the B.E. from Xidian University, Xi'an, China, in 2017. He is currently a Ph.D. candidate at Zhejiang University, Hangzhou, China. His research interests include systems security, threat detection and forensic.



Qi Alfred Chen is an Assistant Professor in the Department of Computer Science at the University of California, Irvine. Before coming to UCI, he received his Ph.D. degree from Computer Science and Engineering at the University of Michigan, Ann Arbor in 2018. Based on Google Scholar, his papers have been cited over 900 times and his h-index is 14. His research interests include network and systems security. Most recently, his research focuses mainly on security problems in smart systems and IoT.

Runqing Yang received his B.Eng. degree in software engineering from HeFei University of Technology, China, in 2015. He is currently pursuing the Ph.D. degree with the college of computer science and technology, Zhejiang University, Hangzhou, China. His research interests include intrusion detection and attack investigation.

Yan Chen received the Ph.D. degree in computer science from the University of California, Berkeley, CA, USA, in 2003. He is a Professor with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, USA. Based on Google Scholar, his papers have been cited over 10000 times and his h-index is 49. His research interests include network security, measurement, and diagnosis for large-scale networks and distributed systems. Prof. Chen won the Department of Energy (DoE) Early CAREER Award in 2005, the Department of Defense (DoD) Young Investigator Award in 2007, and the Microsoft Trustworthy Computing Awards in 2004 and 2005 with his colleagues.

Wei Ruan is a professorate senior engineer. After graduating from Shanghai Jiao Tong University in 1991, he received M.S. and Ph.D. degrees from Zhejiang University majoring in the Department of Energy in 1997 and 2000, respectively. He is currently a teacher with the college of control, Zhejiang University, serving as the director of the equipment automation center of the advanced technology research institute of Zhejiang University at the same time.