International Journal of Networked and Distributed Computing (2023) 11:112–123
https://doi.org/10.1007/s44227-023-00014-9

RESEARCH ARTICLE

ProvSec: Open Cybersecurity System Provenance Analysis Benchmark Dataset with Labels

Madhukar Shrestha[1] · Yonghyun Kim[1] · Jeehyun Oh[1] · Junghwan (John) Rhee[1] · Yung Ryn Choe[2] · Fei Zuo[1] · Myungah Park[1] · Gang Qian[1]

Received: 20 June 2023 / Accepted: 7 November 2023 / Published online: 15 November 2023

Abstract
System provenance forensic analysis has been studied by a large body of research work. This area needs fine granularity data such as system calls along with event fields to track the dependencies of events. While prior work on security datasets has been proposed, we found a useful dataset of realistic attacks and details that are needed for high-quality provenance tracking is lacking. We created a new dataset of eleven vulnerable cases for system forensic analysis. It includes the full details of system calls including syscall parameters. Realistic attack scenarios with real software vulnerabilities and exploits are used. For each case, we created two sets of benign and adversary scenarios which are manually labeled for supervised machine-learning analysis. In addition, we present an algorithm to improve the data quality in the system provenance forensic analysis. We demonstrate the details of the dataset events and dependency analysis of our dataset cases.

Keywords Provenance · Dataset · Attack · Backtracking

1 Introduction

Cybersecurity incidents on our nation's government and commerce are soaring. In 2021 alone, critical infrastructures [1], companies [2], schools [3, 4], and municipal agencies [5] suffered major ransomware attacks and data breaches. The cybersecurity company Kaseya estimated that ransomware compromised up to 1500 businesses during this time [6]. Industry statistics show that more than a thousand annual data breach cases have occurred since 2016 [7] and federal agencies experience more than 30,000 cyber incidents annually [8].

System forensic analysis also known as system provenance analysis [9–25] is an effective technique to track the dependencies across system events in a cyber incident, therefore, assessing the scope of damage and understanding the attack route of an intrusion. Previous approaches in security datasets have been proposed for research and educational purposes [26–28]. However, they lack the following characteristics to be used for provenance analysis research and education.

- *High-quality dependencies across events.*—To conduct provenance analysis, such datasets should have dependency information intact, so that the causality of events

✉ Junghwan (John) Rhee
jrhee2@uco.edu

Madhukar Shrestha
mshrestha21@uco.edu

Yonghyun Kim
ykim26@uco.edu

Jeehyun Oh
joh8@uco.edu

Yung Ryn Choe
yrchoe@sandia.gov

Fei Zuo
fzuo@uco.edu

Myungah Park
mpark5@uco.edu

Gang Qian
gqian@uco.edu

[1] Computer Science Department, University of Central Oklahoma, 100 University North Drive, Edmond, OK 73034, USA

[2] Sandia National Laboratories, P.O. Box 969 MS 9105, Livermore, CA 94551, USA

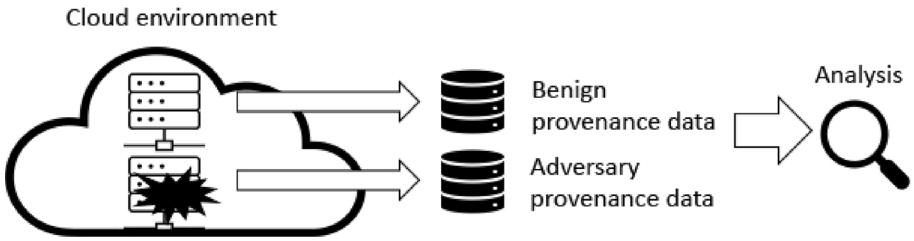

Fig. 1 Architecture of ProvSec dataset

can be systematically reasoned. Operating system calls with required parameters are an example that qualifies for this purpose.

- *Realistic threat behavior.*—The datasets should be based on a realistic scenario and real vulnerability exploits to reflect the characteristics and complexity of real software exploit attacks.
- *Explicit data labeling to assist machine-learning tasks.*—The dataset should be labeled to be useful for validation purposes. Also, machine-learning tasks with supervision require accurate labels. We prepared clearly labeled dataset where each scenario case is provided with two recorded runtime instances; a benign scenario without an attack, and an adversary scenario with an attack occurring. This structure can simplify manual examination and data pre-processing for machine-learning-based approaches.

This paper proposes a new dataset for system provenance analysis called ProvSec[1] to meet this need and provide an improved solution for past work's shortcomings. We use cyber attacks simulated in a cloud-based virtual environment to provide detailed high-quality digital forensic artifacts.

This paper is organized in the following way. Section 2 presents the design of the dataset. Its evaluation is presented in Sect. 3. Section 4 presents the details of the dataset shared. Section 5 discusses the information regarding data sharing and analysis. Section 6 presents related work. Finally, Sect. 7 concludes this paper.

2 Design of ProvSec

To meet the aforementioned qualities demanded, we propose ProvSec, a cybersecurity provenance analysis dataset (Fig. 1) comprising the following.

2.1 Cloud Incidents

Virtual machines simulating the hosts of cyber attacks will provide realistic and safe sandbox environments for cybersecurity experiments while preventing any unintended damages such as mistaken security operations during course modules. Also, virtualization technology is useful for integrating the management of virtual environments and data transfer, so that forensic data are collected, labeled, and managed with convenience.

2.2 Provenance Data

In practical incident response research and education, obtaining high-quality data is critical to *successfully expose attack sequences from piles of evidence*. This is one important implementation goal. In real incidents, investigators may end up with an incomplete attack scenario due to various reasons such as an organization's unprepared cyber infrastructure against potential incidents (e.g., lack of monitoring software and loss of logs).

ProvSec records and safely preserves system forensic event history and artifacts, so that we can analyze and recover the details of attack and defense system activities. This architecture will offer cyber analysts/investigators realistic environments, data navigation interfaces, and quality forensic data. They will access these historical data through well-defined interfaces and available functions for manual and automated investigation.

Another important design issue of ProvSec is deciding which data to collect. Traditionally, provenance analysis research relies on operating system calls, which we also chose for the data format. A system call is a lower level interface invoked by software to use the services of the operating system kernel. Critical services for resources and privileges (e.g., memory, file, network, and processes) are performed via system calls. Therefore, this interface is important to monitor to understand attack activities and determine their causalities (e.g., a network intrusion → login → data copy).

Each event is stored as a json object with 14 fields shown in Table 1. We selected and adopted several fields available in sysdig event tracer for our dataset format. datetime is the event time relative to the start of the execution of the case.

[1] ProvSec is an acronym of Open System *Prov*enance Analysis *Sec*urity Dataset.

Table 1 ProvSec event format

Field name	Description	Example value
Datetime	Time relative to the start time	49544039877
Type	System call name	execve
proc_name	Process name	ps
proc_pname	Parent process name	cloud-soft-01
proc_args	Process argument	-e -o pid,ppid,state,command
proc_pid	Process ID	9747
proc_ppid	Parent process ID	3247
fd_cip	Client IP	$< IP_1 >$
fd_cport	Client port	3425
fd_name	File descriptor name	/lib/x86_64-linux-gnu/libc.so.6
fd_sip	Server IP	$< IP_2 >$
fd_sport	Server port	80
fd_type	File descriptor type	ipv4
Order	Event order	1

`type` is the system call name. We provide the names and program IDs of the current process and the parent process. prog_args field is useful to show the program parameters. File events have a file as an object whose name is described in the fd_name field and the type is shown in the fd_type field. The network events have the IP addresses and ports for the client and server sides, which are respectively described in the fd_cip, fd_cport fields (client side) and in fd_sip, fd_sport fields (server side). Each system call can be generated as one or two events (e.g., the start event of a system call and the end event of the system call) depending on system call types. Then, the order field shows the order (e.g., 0, 1).

2.3 Provenance Analysis with Graph Improvements

These events are analyzed by event dependence analysis [17] known as a backtracking algorithm. We made several improvements in the original backtracking algorithm as shown in Algorithm 1 to improve multiple practical issues. Note this algorithm is general to any provenance data making it applicable to related work.

2.3.1 Improvement #1: Incomplete Capture of All Processes

In the original backtracking system [17], the data recorder is integrated with the hypervisor. Therefore, it tracks all processes starting from the very first one. However, we use a data recorder (sysdig) on top of a COTS operating system (ubuntu) which initiates recording after the machine has finished the booting sequence and loading daemons. This deployment issue causes the data recorder to miss the creation of certain processes.

While this issue can be partially alleviated by starting the recording software as early as possible in the booting stage, there is always a chance that some process starts could be missed from the recording, while their behavior is recorded. We handled this issue for practical usage by including such programs into the graph using *artificial process creation* when their behavior is observed for the first time. As shown in the lines 2–15 of Algorithm 1, when their first behavior is processed, the algorithm creates an artificial fork (process creation) event.

2.3.2 Improvement #2: Limited Data Fields from a Data Recorder

We found some recording fields from our data monitoring software; sysdig are missing as such data may not be available at the time when the data are retrieved and stored inside the OS kernel.

To improve data quality, we added logic to supplement such missing information as much as possible by extracting it from the event's metadata and other recorded history. This part is shown in the lines 16–18 of Algorithm 1.

2.3.3 Improvement #3: Anonymization

There are some names of processes or resources that might be sensitive to be identified. We applied an anonymization process to replace such names with artificial names. Lines 29–35 show this process. Generally, the anonymization of events is a complicated process. However, this is not the case for our approach, because we use a fixed list of event fields that can be properly examined and anonymized.

2.4 Attack Cases

We created several scenarios of cyber attacks where their data are generated by setting up virtual machines, software,

Algorithm 1 Enhanced backtracking algorithm

Require: Backtrack graph $G = (N, E)$
Require: System call trace L, Anonymization list A
for foreach event e in log L **do**
  if $e.src \notin N$ **then**
    if $e.src.parent \notin N$ **then**
      $N = N \cup \{e.src.parent\}$
    end if
    $E = \cup\{e.src.parent \rightarrow fork \rightarrow e.src\}$
    $N = N \cup \{e.src\}$
  end if
  if $e.tgt \notin N$ **then**
    if $e.tgt.parent \notin N$ **then**
      $N = N \cup \{e.tgt.parent\}$
    end if
    $E = \cup\{e.tgt.parent \rightarrow fork \rightarrow e.tgt\}$
    $N = N \cup \{e.tgt.parent, e.tgt\}$
  end if
  if $e.tgt == null$ **then**
    $e.tgt = ExtractTarget(e.metadata)$
  end if
  for object O in N **do**
    if $e.tgt == O$ by the time threshold for O **then**
      if $e.src \notin N$ **then**
        $N = N \cup \{e.src\}$
        set time threshold for $e.src$ to time of e
      end if
      $E = E \cup \{e\}$
    end if
  end for
end for
for $n \in N$ **do**
  for $(pattern, Anonymize())$ in A **do**
    if n matches $pattern$ **then**
      $n = Anonymize(n)$
    end if
  end for
end for

and triggering attack actions along with manual labeling of behavior.

- *Case 01—Nginx integer overflow vulnerability*: This case represents an integer overflow vulnerability that exists in Nginx software whose versions are between 0.5.6 and 1.13.2. This vulnerability is caused by insufficient bound checking (CVE-2017-7529).
- *Case 02—Path traversal and file disclosure vulnerability in Apache HTTP Server* : Apache 2.4.49 has a vulnerability that allows a path traversal attack to map URLs to files outside the expected document root (CVE-2021-41773). We used this vulnerability to execute several UNIX commands.
- *Case 03—Python PIL/pillow remote shell command execution via ghostscript*: Ghostscript whose version is before 9.24 has a vulnerability that allows the exploitation of a remote shell command. We create a file /tmp/test.txt remotely in the target server as a demonstration (CVE-2018-16509)
- *Case 04—PHP IMAP remote command execution vulnerability*: The PHP IMAP extension is used to send and receive emails. imap_open call internally uses ssh and an attacker can inject a parameter for a remote command execution. We conducted an attack to execute the command echo '1234567890'>/tmp/test0001 (CVE-2018-19518)
- *Case 05—Apache Log4j2 lookup feature JNDI injection with a reverse shell*: Apache Log4j, a Java-based logging utility, has a vulnerability CVE-2021-44228 in its support for JNDI (Java Naming and Directory Interface). We used this vulnerability to initiate a reverse shell.

- *Case 06—Apache Tomcat AJP Arbitrary File Read/Include Vulnerability*: Apache Tomcat has a vulnerability CVE-2020-1938 known as Ghostcat that allows an attacker a file read. We used this vulnerability to read a sensitive password file, /etc/passwd, as a demonstration of an arbitrary file read.
- *Case 07—Redis Lua Sandbox Escape and Remote Code Execution*: Redis, an open-source in-memory data structure store, has a vulnerability CVE-2022-0543 to allow an escape of Lua sandbox and an execution of an arbitrary remote command. We used this vulnerability to run UNIX commands and dump the password file.
- *Case 08—Consul service APIs' misconfiguration leading to Remote Code Execution (RCE) and reverse shell*: Consul is an open-source software to discover and configure services. It has a vulnerability that allows remote code execution. We created a remote shell followed by several attack commands.
- *Case 09—Path traversal and file disclosure vulnerability in Apache HTTP Server*: This attack case is regarding CVE-2021-42013 which is a vulnerability caused by an incomplete fix of CVE-2021-41773. After the fix, the Apache server still allows path traversals and execution of remote commands.
- *Case 10—Django QuerySet.order_by SQL Injection Vulnerability*: Django has a vulnerability that allows SQL injection (CVE-2021-35042). We used this vulnerability to collect information from the machine as an error message.
- *Case 11—Escape from a Docker container: Vulnerability on docker*: Docker has a vulnerability for an attacker to escape a container and run commands (CVE-2019-5736). We used this vulnerability to create a backdoor and execute several UNIX commands.

2.5 Dependency Graph Reduction

We identified a detection point of each dataset case and conducted dependency analysis to reduce the graph size. The examples of several dataset cases are presented in the evaluation section. They show a significant reduction in the sizes and complexity of graphs.

3 Evaluation

This section presents the evaluation of ProvSec datasets. We created a total of 11 attack scenarios using widely used software and vulnerabilities.

We created the ProvSec dataset using docker containers and sysdig on top of Ubuntu 20.04. We have prepared a total of eleven real attack scenarios for this dataset. The details for these cases are illustrated in Figs. 2, 4, and 6, which respectively show the full attack behavior of C02, C03, and C05 scenarios.

We have three different types of behavior: process, file, and network, which are shown in different colors. In each figure, the red nodes and edges represent processes and process creation events, such as execve, fork, and clone system calls. Blue nodes and edges represent files and file activities. Their examples include open, close, read, and write system calls and their variants. The green nodes and edges represent network addresses and network activities, such as connect and accept system calls.

3.1 Graph Complexity

Table 2 shows the details of 11 incident cases. The graph complexity of each case is presented in Table 3. $|N|$ represents the total number of nodes and $|E|$ represents the total number of edges. This table also shows the complexity of backtrack graphs which are simplified by applying a dependency analysis on the detection points. Their nodes and edges are shown in $|N_{bt}|$ and $|E_{bt}|$ columns and their reduction rates compared to the full graphs are respectively shown in $\frac{|N_{bt}|}{|N|}$ and $\frac{|E_{bt}|}{|E|}$. The nodes are simplified to 0.5–17.9% of the original graphs. The edge complexity got lower to 0.015–9.5%.

3.2 Simplified Backtracking Graphs

In this section, we explain three cases of attack graphs and their simplified attack behavior as examples.

Case 02: Apache Path Traversal and File Conflict of interest: Fig. 3 shows the simplified behavior of the original graph, Fig. 2 which demonstrates a path traversal and file disclosure vulnerability attack targeted on Apache http server. The simplified graph of Fig. 3 shows that the shell (sh) and ls processes were invoked from the httpd process exposing the paths of the server.

Case 03: Python PIL/Pillow RCE via Ghostscript: Fig. 5 highlights the core attack of the original graph, Fig. 4 by removing irrelevant nodes and edges of the C03 scenario. The intrusion was detected by the touch command which was triggered by shell processes (sh whose process IDs are 87004 and 87005). We can confirm that these processes were created by the Ghostscript (gs) processes whose process IDs are 87003 and 87004 which came from the python process (python). This graph indicates the root cause of an vulnerability exploit of Ghostscript in the Python program.

Case 05: Apache log4j lookup with JNDI injection: Fig. 6 illustrates a complex behavior of the Apache Log4j incident. This attack is initiated via the JNDI injection and a reverse shell demonstrated in its backtrack graph, Fig. 7. In this graph, we can observe a shell process (sh) of its process ID, 9743, was forked from a java process (PID 9712). Note

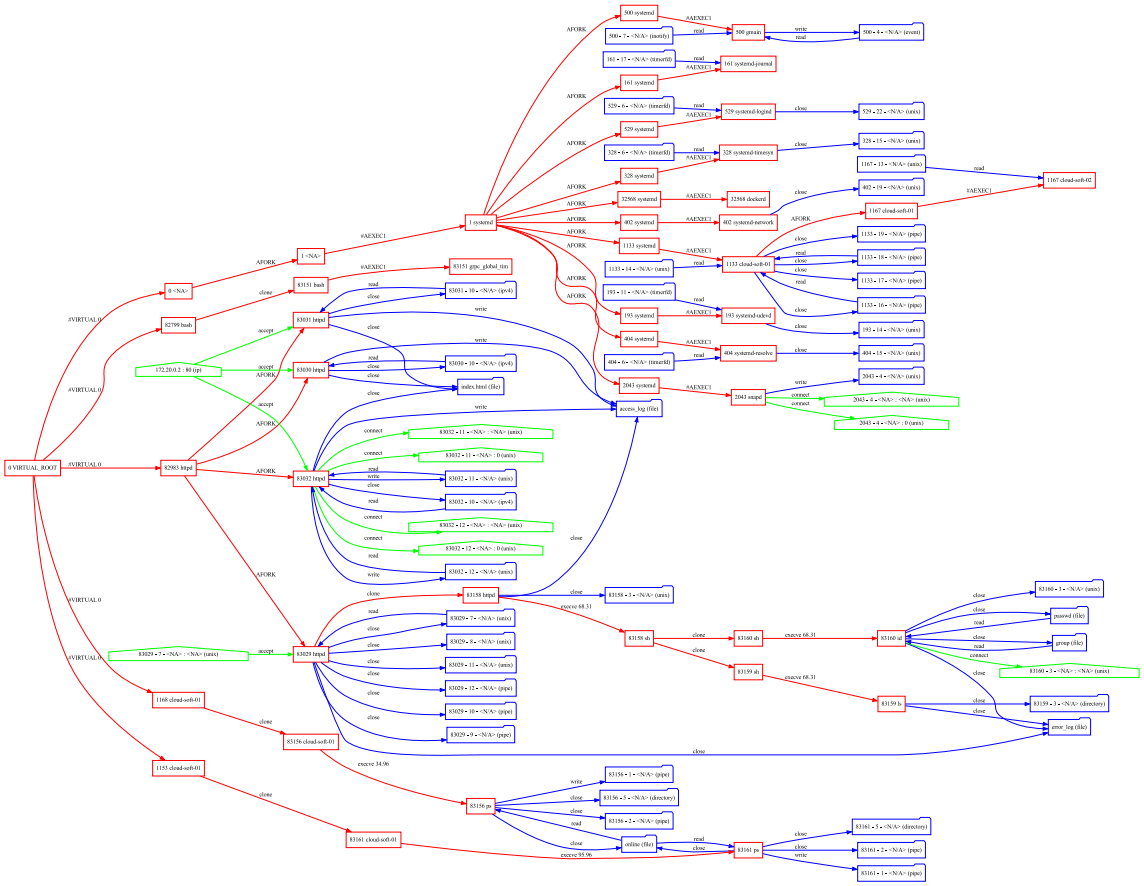

Fig. 2 Dependency graph of C02-path traversal and file disclosure vulnerability in Apache HTTP Server

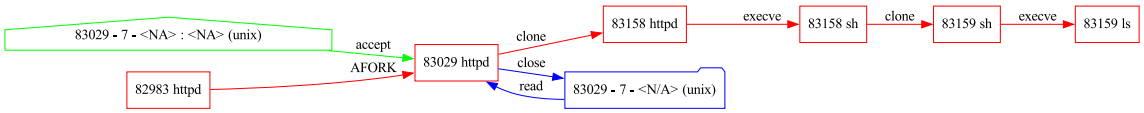

Fig. 3 Simplified backtrack graph of C02-path traversal and file disclosure vulnerability in Apache HTTP server

this shell process was initially a java process and then turns into a shell using a execve system call. This shell process conducts two attack behavior copying (cp) and modifying (touch) a sensitive file (FiscalYearEndReport.xlsx). This simplified graph demonstrates what accesses have occurred on the sensitive file as a summary of attack behavior.

4 Data Characteristics

Table 4 shows the characteristics of the dataset events that we share. For 11 attack cases, we have two samples of recordings: one for a benign workload without any attack and the other for an adversary workload with attack behavior. For each recording, the number of events (#E), the number of distinct process names (#P), the number of distinct IP addresses (#I), and the number of different system call types (#T) are presented. The data recorder, sysdig, that we use generates one or two events per system call. Therefore, the total number of system calls will be less than #E. The total number of events of a benign case or an adversary case is different because of different workloads. In all eleven data cases, our dataset has 341.7K events in the benign cases and 987.7K events in the abnormal cases.

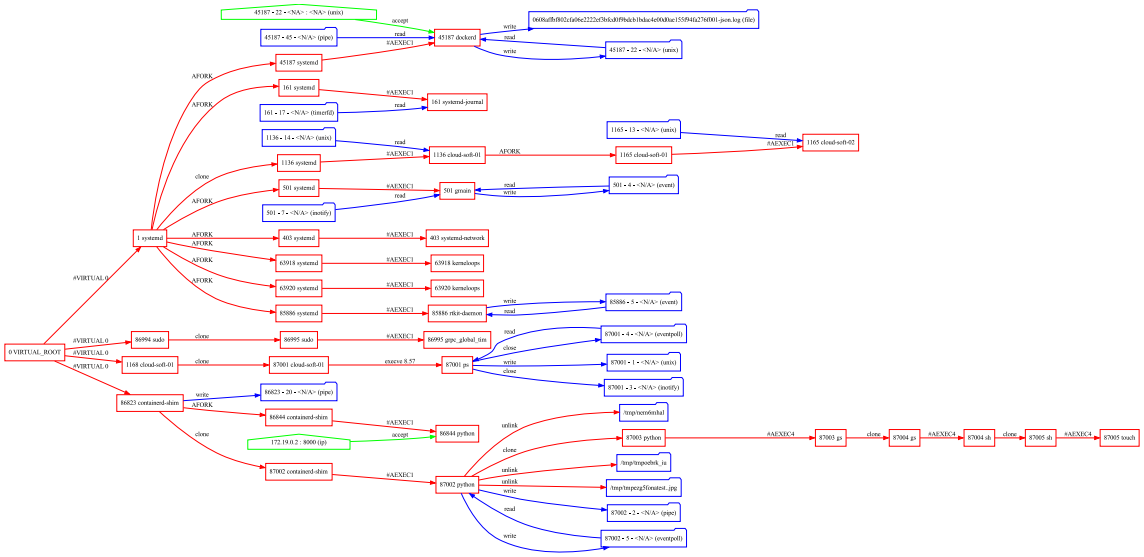

Fig. 4 Dependency graph of C03-python PIL/pillow remote shell command execution via ghostscript

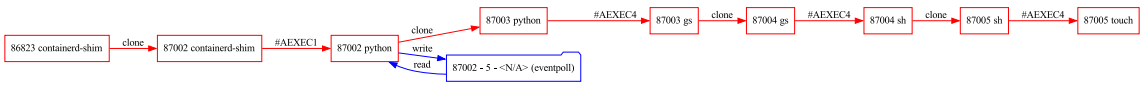

Fig. 5 Simplified backtrack graph of C03-python PIL/Pillow remote shell command execution via ghostscript

Table 2 Details of the incident cases

#	Name	CVE	Description
C01	nginx	CVE-2017-7529	Nginx integer overflow vulnerability
C02	apache	CVE-2021-41773	Path traversal and file disclosure vulnerability
C03	ghostscript	CVE-2018-16509	Python PIL/Pillow RCE via Ghostscript
C04	php	CVE-2018-19518	php IMAP RCE vulnerability
C05	log4j	CVE-2021-44228	Apache log4j lookup with JNDI injection
C06	tomcat	CVE-2020-1938	Apache Tomcat AJP arbitrary file read/include
C07	redis	CVE-2022-0543	Redis Lua sandbox escape and RCE
C08	consul	N/A	Consul service APIs misconfiguration, RCE
C09	apache	CVE-2021-42013	Path traversal and file disclosure
C10	django	CVE-2021-35042	Django QuerySet.order_by SQL injection
C11	docker	CVE-2019-5736	Escape from a Docker container

5 Discussion

5.1 Data Sharing

We share our dataset with the cybersecurity community in the following link: https://uco-cyber.github.io/research/#provsec.

5.2 Processing Time for a Real-Time System

We used the Python language to write data processing code. Loading and analyzing the entire 1.3 million events take less than a minute with our Python implementation. If a compiled native program written in C or C++ is used, we can speed up this processing time further significantly. As the next step of this project, we are processing these events collected from multiple machines for anomaly detection in a live fashion. Therefore, we are able to use this type of data in a real-time environment.

Table 3 Details of the incident provenance graphs

#	Original graph		Reduced graph		Reduction rate	
	$\|N\|$	$\|E\|$	$\|N_{bt}\|$	$\|E_{bt}\|$	$\frac{\|N_{bt}\|}{\|N\|}$	$\frac{\|E_{bt}\|}{\|E\|}$
C01	124	2758	4	3	3.2%	0.11%
C02	99	1044	8	8	8.1%	0.77%
C03	56	105	10	10	17.9%	9.5%
C04	212	1454	11	13	5.2%	0.89%
C05	601	6177	9	10	1.5%	0.16%
C06	143	898	7	8	4.9%	0.89%
C07	80	202	10	10	12.5%	5.0%
C08	1203	34173	6	5	0.5%	0.015%
C09	117	1383	8	8	6.8%	0.59%
C10	176	1704	5	5	2.8%	0.29%
C11	1764	47183	295	729	16.7%	1.55%

6 Related Work

In this section, we compare our work with multiple prior works proposed for security datasets.

Network-oriented dataset: Many existing works focus on network-oriented data, such as five-tuples or full packet recordings (e.g., PCAP) [29–34]. While these datasets have an influence on multiple research works, they lack the information necessary to conduct dependency analysis of operating system events for system provenance analysis.

Software vulnerability dataset: Other dataset work [35–38] is regarding software vulnerability including useful features, such as source code information, CWE (Common Weakness Enumeration), CVE (Common Vulnerability Enumeration), code metrics, etc. The datasets of this category have full details at the code level. However, they do not provide the runtime data on how they use operating system services and their parameters which are necessary to conduct system provenance analysis.

Provenance dataset: Multiple cybersecurity datasets have been introduced for the details of system behavior which enables provenance analysis. ISOT-CID dataset [26] includes data of multiple formats including network traffic, system logs, performance data (e.g., CPU utilization), and system calls. While these data are quite close to what we provide, the system call data are incomplete and not structured. They lack full details and the records are in a non-standard format similar to the strace output. Therefore, it takes manual effort to parse, curate, and extract useful information from the records. ProvMark [39] is a benchmarking system regarding provenance expressiveness, which evaluates three types of provenance recorders: OPUS [40], CamFlow[41], and SPADE [42], [43].

DARPA released Operationally Transparent Cyber (OpTC) data that was used to evaluate the DARPA Transparent Computing (TC) program [27]. These data have been used in multiple papers for analyzing APT attacks. While this dataset has a large volume of rich data, it lacks proper explanation, so that the details of attacks are understood by researchers. In this regard, Anjun et al. analyzed and published the details of OpTC dataset [28] explaining the details of characteristics. However, still, this paper describes overall statistics such as the types of actions and objects.

Compared to these approaches, ProvSec has several advantages that can help researchers conduct research with provenance data especially for machine-learning tasks. Our dataset has full details of system calls and parameters that are organized in the json format and enable the construction of operating system dependencies and system provenance analysis. We utilized real vulnerabilities and proof-of-concept (PoC) code to simulate attack scenarios inside docker environments which are recorded in the operating system kernel.

As a most useful characteristic, we provide manual labeling of the attacks that are helpful to identify the root causes of attacks and the full details of attack behavior which will help experiments that need ground truth validation or supervised machine-learning experiments. Each scenario data case is organized into *two separate runtime instances and corresponding recording files*, (1) one benign case and (2) an adversary case which is recorded without and with attack behavior. This clear labeling structure can significantly facilitate the data pre-processing for machine-learning tasks.

Provenance analysis: Provenance analysis has been studied by a large body of work in recent years [9–25]. Recent survey papers [24, 25] summarize multiple approaches conducted in the provenance tracking and dependency analysis according to their categorizations. Multiple attack detection approaches [23, 44–47] have been proposed to detect APT campaign effectively. Due to a large volume of data, several ideas for data reduction have been explored, such as execution partitioning, garbage collection, and approximations of behavior patterns [18, 48–53]. Regarding the mechanisms of collecting provenance data, several approaches used

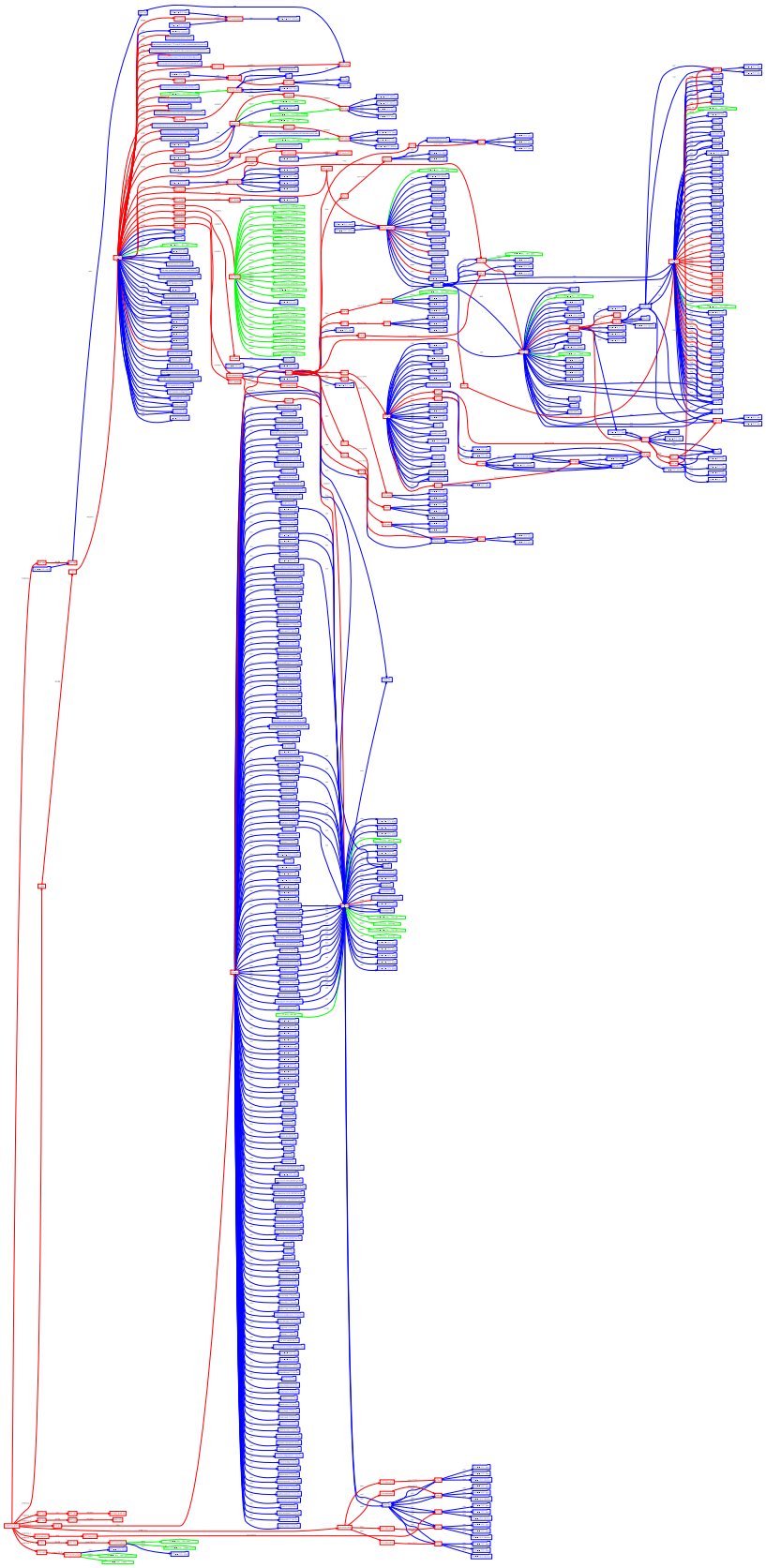

Fig. 6 Dependency Graph of C05-Apache Log4j2 lookup feature JNDI injection with a reverse shell

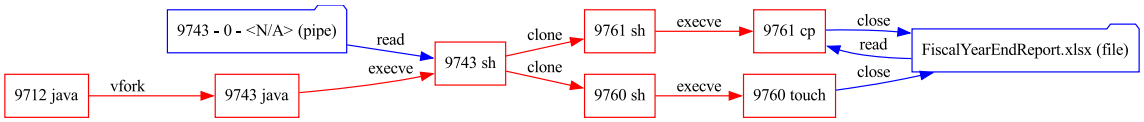

Fig. 7 Backtrack Graph of C05-Apache Log4j2 lookup feature JNDI injection with a reverse shell

Table 4 Data characteristics

#	Name	Benign				Adversary			
		#E	#P	#I	#T	#E	#P	#I	#T
C01	Nginx	10268	25	7	47	7095	17	7	26
C02	Apache	3469	16	4	24	3204	18	3	26
C03	Ghostscript	5240	17	2	26	3065	15	2	25
C04	Php	4337	19	4	36	5380	21	8	39
C05	Log4j	13257	30	4	45	808501	36	10	54
C06	Tomcat	167430	25	2	28	7848	30	5	33
C07	Redis	20617	26	3	37	7538	21	4	23
C08	Consul	72254	43	28	50	100785	62	31	61
C09	Apache	2934	15	4	23	3886	18	2	26
C10	Django	13392	16	7	27	15676	17	6	27
C11	Docker	28539	39	5	62	24742	33	5	60

#E total number of events, #P total number of distinct process names, #I total number of distinct IP addresses, #T total number of distinct system call types

OS-level data collectors with program instrumentation [18, 19]. A kernel-based framework [21] and hardware-based technique [22] are also proposed. As another trend, machine-learning-based solutions [10, 44, 54] are increasingly being used to improve detection. All such approaches can benefit from the provenance dataset with high-quality labeling. Our work can contribute to such approaches as additional evaluation data.

7 Conclusion

In this paper, we introduce a new dataset for security provenance analysis along with a detailed description, analysis, and clearly provided labels with two separate execution traces of a benign scenario and an adversary scenario. This dataset is differentiated from prior work with detailed data for causal dependencies across events, the usage of real vulnerabilities and PoC exploits, and manual labeling which particularly would be helpful for validation and supervised machine-learning tasks. We performed an enhanced causality dependence analysis with our improved algorithm and demonstrated how the dependency analysis can simplify the analysis of each attack scenario with our dataset cases. We made our dataset public, so that the research and education communities advancing provenance analysis can benefit from this dataset.

Acknowledgements Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This article describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government. This work was supported through contract #70RSAT21KPM000105 with the U.S. Department of Homeland Security Science and Technology Directorate. Junghwan Rhee is the corresponding author of this work.

Data availability The supplementary data underlying this article are available at https://uco-cyber.github.io/research/#provsec.

Declarations

Conflict of Interest The authors declare they have no conflicts of financial, non-financial, proprietary, or competing interests.

References

1. Bloomberg (2021) Colonial pipeline paid hackers nearly 5 million in ransom, https://www.bloomberg.com/news/articles/2021-05-13/colonial-pipeline-paid-hackers-nearly-5-million-in-ransom/. Accessed 11 Nov 2023
2. Reuters (2021) Toshibas european business hit by cyberattack, https://www.reuters.com/business/autos-transportation/toshibas-european-business-hit-by-cyberattack-source-2021-05-14/. Accessed 11 Nov 2023
3. Schools BP (2021) Cybersecurity attack on the buffalo public schools, https://www.buffaloschools.org/cms/lib/NY01913551/Centricity/Domain/8/Cybersecurity%20Update%203-15-21.pdf. Accessed: 03 Dec 2023
4. Magazine S (2021) Now ransomware is inundating public school systems, https://www.securitymagazine.com/articles/95164-now-ransomware-is-inundating-public-school-systems. Accessed: 11 Nov 2023
5. Oklahoma N (2021) Tulsa system shutdown alters backside operations ransomware attack still being investigated, https://www.kjrh.com/news/local-news/tulsa-system-shutdown-alters-backside-operations-ransomware-attack-still-being-investigated. Accessed 11 Nov 2023
6. CNN, Kaseya ransomware attack businesses affected. (2021). https://www.cnn.com/2021/07/06/tech/kaseya-ransomware-attack-businesses-affected/index.html. Accessed 11 Nov 2023
7. Statista, Annual number of data breaches and exposed records in the united states from 2005 to 2020," https://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/. Accessed 11 Nov 2023
8. Statista, Number of cyber security incident reports by federal agencies in the united states from fy 2006 to 2018. https://www.statista.com/statistics/677015/number-cyber-incident-reported-usa-gov/. Accessed 11 Nov 2023.
9. Liu Y, Zhang M, Li D, Jee K, Li Z, Wu Z, Rhee J, Mittal P (2018) Towards a timely causality analysis for enterprise security. in NDSS
10. Wang Q, Hassan WU, Li D, Jee K, Yu X, Zou K, Rhee J, Chen Z, Cheng W, Gunter CA et al (2020) You are what you do: Hunting stealthy malware via data provenance analysis. in NDSS
11. Xu Z, Wu Z, Li Z, Jee K, Rhee J, Xiao X, Xu F, Wang H, Jiang G (2016) High fidelity data reduction for big data security dependency analyses, in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security
12. Tang Y, Li D, Li Z, Zhang M, Jee K, Xiao X, Wu Z, Rhee J, Xu F, Li Q (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
13. Hassan WU, Li D, Jee K, Yu X, Zou K, Wang D, Chen Z, Li Z, Rhee J, Gui J et al (2020) This is why we can't cache nice things: Lightning-fast threat hunting using suspicion-based hierarchical storage, in Annual Computer Security Applications Conference
14. Ma S, Lee KH, Kim CH, Rhee J, Zhang X, Xu D (2015) Accurate, low cost and instrumentation-free security audit logging for windows," in Proceedings of the 31st Annual Computer Security Applications Conference, ser. ACSAC 2015. New York, NY, USA: Association for Computing Machinery. [Online]. Available: https://doi.org/10.1145/2818000.2818039
15. Sun Y, Jee K, Sivakorn S, Li Z, Lumezanu C, Korts-Parn L, Wu Z, Rhee J, Kim CH, Chiang M et al (2020) Detecting malware injection with program-dns behavior, in 2020 IEEE European Symposium on Security and Privacy (EuroS &P). IEEE
16. Zipperle M, Gottwalt F, Chang E, Dillon T (2022) Provenance-based intrusion detection systems: A survey, ACM Computing Surveys, vol. 55, no. 7
17. King ST, Chen PM (2003) Backtracking intrusions," in Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, ser. SOSP '03. New York, NY, USA: Association for Computing Machinery, p. 223-236. [Online]. Available: https://doi.org/10.1145/945445.945467
18. Lee KH, Zhang X, Xu D (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. [Online]. Available: https://www.ndss-symposium.org/ndss2013/high-accuracy-attack-provenance-binary-based-execution-partition
19. Ma S, Zhang X, Xu D (2016) Protracer: Towards practical provenance tracing by alternating between logging and tainting, in Network and Distributed System Security Symposium (NDSS)
20. Liu Y, Zhang M, Li D, Jee K, Li Z, Wu Z, Rhee JJ, Mittal P (2018) Towards a timely causality analysis for enterprise security, in Network and Distributed System Security Symposium (NDSS)
21. Bates A, Tian D, Butler KRB, Moyer T (2015) Trustworthy whole-system provenance for the linux kernel, in 24th USENIX Security Symposium. USENIX Association, p. 319-334
22. Zeng J, Zhang C, Liang Z (2022) Palantír: Optimizing attack provenance with hardware-enhanced system observability, in Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, p. 3135-3149
23. Hassan W, Guo S, Li D, Chen Z, Jee K, Li Z, Bates A (2019) Nodoze: Combatting threat alert fatigue with automated provenance triage, in Network and Distributed System Security Symposium (NDSS)
24. Pan B, Stakhanova N, Ray S (2023) Data provenance in security and privacy," ACM Comput. Surv., vol. 55, no. 14s, jul. [Online]. Available: https://doi.org/10.1145/3593294
25. Inam M, Chen Y, Goyal A, Liu J, Mink J, Michael N, Gaur S, Bates A, Hassan W (2023) Sok: History is a vast early warning system: Auditing the provenance of system intrusions, in 2023 IEEE Symposium on Security and Privacy (SP), 2620–2638
26. Aldribi A, Traore I, Moa B (2018) Data Sources and Datasets for Cloud Intrusion Detection Modeling and Evaluation. Cham: Springer International Publishing, pp. 333–366. [Online]. Available: https://doi.org/10.1007/978-3-319-73676-1_13
27. DARPA, Operationally transparent cyber (optc) data release. https://github.com/FiveDirections/OpTC- data, (2021)
28. Anjum MM, Iqbal S, Hamelin B (2021) Analyzing the usefulness of the darpa optc dataset in cyber threat detection research, in Proceedings of the 26th ACM Symposium on Access Control Models and Technologies, ser. SACMAT. ACM, p. 27-32
29. Lippmann R, Fried D, Graf I, Haines J, Kendall K, McClung D, Weber D, Webster S, Wyschogrod D, Cunningham R, Zissman M (2000) Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation, in Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00, vol. 2, pp. 12–26 vol.2
30. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the kdd cup 99 data set, in 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 1–6
31. Banadaki YM (2020) Detecting malicious dns over https traffic in domain name system using machine learning classifiers, Journal of Computer Sciences and Applications, vol. 8, no. 2, pp. 46–55. [Online]. Available: http://pubs.sciepub.com/jcsa/8/2/2
32. Koroniotis N, Moustafa N, Sitnikova E, Turnbull B (2019) Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset, Future

Generation Computer Systems, vol. 100, pp. 779–796. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18327687. Accessed 11 Nov 2023
33. Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization, in International Conference on Information Systems Security and Privacy
34. Jonker M, King A, Krupp J, Rossow C, Sperotto A, Dainotti A (2017) Millions of targets under attack: A macroscopic characterization of the dos ecosystem, in Proceedings of the 2017 Internet Measurement Conference, ser. IMC '17. New York, NY, USA: Association for Computing Machinery, p. 100-113. [Online]. Available: https://doi.org/10.1145/3131365.3131383
35. Gkortzis A, Mitropoulos D, Spinellis D (2018) Vulinoss: A dataset of security vulnerabilities in open-source systems," in Proceedings of the 15th International Conference on Mining Software Repositories, ser. MSR '18. New York, NY, USA: Association for Computing Machinery, p. 18-21. [Online]. Available: https://doi.org/10.1145/3196398.3196454
36. Nguyen V (2021) Some software vulnerability real-world data sets. [Online]. Available: https://doi.org/10.21227/1m98-5h52. Accessed 11 Nov 2023
37. Kim D, Kim E, Cha SK, Son S, Kim Y (2020) Revisiting binary code similarity analysis using interpretable feature engineering and lessons learned, CoRR, vol. abs/2011.10749. [Online]. Available: https://arxiv.org/abs/2011.10749
38. Marcelli A, Graziano M, Ugarte-Pedrero X, Fratantonio Y, Mansouri M, Balzarotti D (2022) How machine learning is solving the binary function similarity problem, in 31st USENIX Security Symposium (USENIX Security 22). Boston, MA: USENIX Association, Aug, pp. 2099–2116. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/marcelli. Accessed 11 Nov 2023
39. Chan SC, Gehani A, Cheney J, Sohan R, Irshad H (2017) Expressiveness benchmarking for system-level provenance," in 9th USENIX Workshop on the Theory and Practice of Provenance
40. Balakrishnan N, Bytheway T, Sohan R, Hopper A (2013) {OPUS}: A lightweight system for observational provenance in user space, in 5th USENIX Workshop on the Theory and Practice of Provenance (TaPP 13)
41. Pasquier TF-M, Singh J, Eyers D, Bacon J (2015) Camflow: managed data-sharing for cloud services. IEEE Trans Cloud Computi 5(3):472–484
42. Gehani A, Tariq D (2012) Spade: Support for provenance auditing in distributed environments, in International Middleware Conference, [Online]. Available: https://api.semanticscholar.org/CorpusID:7346628. Accessed 11 Nov 2023
43. Zuo F, Rhee J, Kim Y, Oh J, Qian G (2023) A Comprehensive Dataset Towards Hands-on Experience Enhancement in a Research-Involved Cybersecurity Program. Proceedings of the 24th Annual Conference on Information Technology Education. https://doi.org/10.1145/3585059.3611416
44. Milajerdi SM, Gjomemo R, Eshete B, Sekar R, Venkatakrishnan V (2019) Holmes: Real-time apt detection through correlation of suspicious information flows," in IEEE Symposium on Security and Privacy (SP), pp. 1137–1152
45. Hossain MN, Milajerdi SM, Wang J, Eshete B, Gjomemo R, Sekar R, Stoller S, Venkatakrishnan V (2017) SLEUTH: Real-time attack scenario reconstruction from COTS audit data, in 26th USENIX Security Symposium). USENIX Association, pp. 487–504
46. Hossain MN, Sheikhi S, Sekar R (2020) Combating dependence explosion in forensic analysis using alternative tag propagation semantics, in IEEE Symposium on Security and Privacy (SP), pp. 1139–1155
47. Hassan WU, Bates A, Marino D (2020) Tactical provenance analysis for endpoint detection and response systems, in IEEE Symposium on Security and Privacy (SP), pp. 1172–1189
48. Lee KH, Zhang X, Xu D (2013) Loggc: Garbage collecting audit log, in Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, ser. CCS. ACM, p. 1005-1016
49. Xu Z, Wu Z, Li Z, Jee K, Rhee J, Xiao X, Xu F, Wang H, Jiang G (2016) High fidelity data reduction for big data security dependency analyses, in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS. ACM, p. 504-516
50. Tang Y, Li D, Li Z, Zhang M, Jee K, Xiao X, Wu Z, Rhee J, Xu F, Li Q (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, ser. CCS. ACM, p. 1324-1337
51. Hossain MN, Wang J, Sekar R, Stoller SD (2018) Dependence-Preserving data compaction for scalable forensic analysis," in 27th USENIX Security Symposium. USENIX Association, pp. 1723–1740
52. Michael N, Mink J, Liu J, Gaur S, Hassan WU, Bates A (2020) On the forensic validity of approximated audit logs," in Annual Computer Security Applications Conference, ser. ACSAC. ACM, p. 189-202
53. Hassan W, Lemay M, Aguse N, Bates A, Moyer T (2018) Towards scalable cluster auditing through grammatical inference over provenance graphs, in Network and Distributed System Security Symposium (NDSS), 01
54. Cheng Z, Lv Q, Liang J, Wang Y, Sun D, Pasquier T, Han X (2024) KAIROS: Practical Intrusion Detection and Investigation using Whole-system Provenance, in IEEE Symposium on Security and Privacy (SP)