SAND2023-02103C

ProvSec: Cybersecurity System Provenance Analysis Benchmark Dataset

Madhukar Shrestha, Yonghyun Kim, Jeehyun Oh,
Junghwan Rhee, Yung Ryn Choe*, Fei Zuo, Myungah Park, Gang Qian
University of Central Oklahoma, *Sandia National Laboratories
{mshrestha21,ykim26,joh8,jrhee2,fzuo,mpark5,gqian}@uco.edu, *yrchoe@sandia.gov

***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 can be used for 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. Also, we created two sets of benign and adversary scenarios which are manually labeled for supervised machine-learning analysis. We demonstrate the details of the dataset events and dependency analysis.**

I. 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]–[15] 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. However, they lack the following characteristics to be used for provenance analysis research and education.

- **RQ1: Dependencies Across Events.** — To conduct provenance analysis, such datasets should have dependency information intact so that the causality of events can be systematically reasoned. Operating system calls with required parameters are an example that qualifies for this purpose.
- **RQ2: 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.
- **RQ3: Accurate Labeling.** — The dataset should be labeled to be useful for validation purposes. Especially machine learning tasks with supervision require accurate labels.

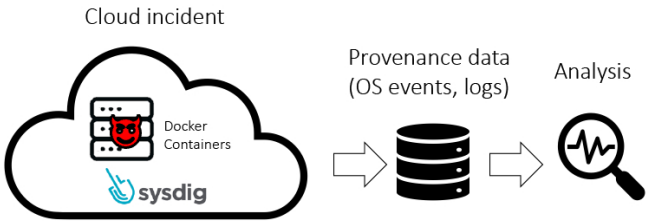

Fig. 1. Architecture of ProvSec Dataset.

This paper proposes a new dataset for system provenance analysis called ProvSec[1] to meet this need. Cyber attacks simulated in a cloud-based virtual environment provide detailed digital forensic artifacts. This paper is organized in the following way. Section II presents the design of the dataset. Its evaluation is presented in Section III. Section IV presents related work. Lastly, Section V concludes this paper.

II. Design of ProvSec

To meet the aforementioned requirements, we propose ProvSec, a cybersecurity provenance analysis dataset (Figure 1) comprising the following:

A. Cloud Incident

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.

B. 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

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

Sandia National Laboratories
U.S. DEPARTMENT OF ENERGY NNSA
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & 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.

potential incidents (e.g., lack of monitoring software, 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.

One important design issue of ProvSec is deciding which data to collect. Traditionally, provenance analysis research relies on the following:

- *System calls* — 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).
- *Software logs* — Most server programs commonly use log files to record any errors and operation status (e.g., access logs in web servers). Since the content of a log file is typically intended to be read by humans, it is generally easier to understand in higher-level formats (e.g., a web page access) compared to low-level operations addressed by system calls (e.g., file descriptors). This would provide useful and complementary information to understand software behavior. Therefore, it is widely used by Security Information and Event Management (SIEM) software for security purposes.
- *OS Service/System logs* — Operating systems have essential software services also known as daemons (e.g., init, systemd, crond, sshd), which also generate useful logs in the system administrator's perspective to indicate any notable events during the operations of a system. For instance, reconnaissance activities targeting OS services may leave unusual trails in their logs, which provide additional evidence for incident response.

C. Provenance Analysis with Graph Augmentation

These events are analyzed by event dependence analysis [16] known as a backtracking algorithm. We made several improvements in the original backtracking algorithm as shown in Algorithm 1 due to the following practical issues:

1) Augmentation #1: Incomplete capture of all processes: Unlike the original algorithm [16] where the data recorder is integrated with the hypervisor, 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 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.

Algorithm 1 Enhanced Backtracking Algorithm

Require: Backtrack graph $G = (N, E)$
Require: Log / System call trace L
Require: Anonymization list A
1: **for** foreach event e in log L **do**
2: **if** $e.src \notin N$ **then**
3: $E = \cup\{e.src.parent \rightarrow fork \rightarrow e.src\}$
4: $N = N \cup \{e.src.parent, e.src\}$
5: **end if**
6: **if** $e.tgt \notin N$ **then**
7: $E = \cup\{e.tgt.parent \rightarrow fork \rightarrow e.tgt\}$
8: $N = N \cup \{e.tgt.parent, e.tgt\}$
9: **end if**
10: **if** $e.tgt == null$ **then**
11: $e.tgt$ = $ExtractTarget(e.metadata)$
12: **end if**
13: **for** object O in N **do**
14: **if** $e.tgt$ == O by the time threshold for O **then**
15: **if** $e.src \notin N$ **then**
16: $N = N \cup \{e.src\}$
17: set time threshold for $e.src$ to time of e
18: **end if**
19: $E = E \cup \{e\}$
20: **end if**
21: **end for**
22: **end for**
23: **for** $n \in N$ **do**
24: **for** $(orig, new)$ in A **do**
25: **if** $n == orig$ **then**
26: $n = new$
27: **else**
28: $n_2 = Anonymize(n)$
29: $n = n_2$
30: $A = A \cup (n, n_2)$
31: **end if**
32: **end for**
33: **end for**

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 2-9 of Algorithm 1, when their first behavior is processed, the algorithm creates an artificial fork (process creation) event.

2) Augmentation #2: Limited data fields from sysdig: We found some recording fields from sysdig are missing as such data may not be available at the time when the data are retrieved and stored inside the OS kernel. 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 lines 10-12.

3) Augmentation #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. The lines 23-33 show this process. Generally, the anonymization of events is a compli-

cated 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.

D. Attack Cases

We created several scenarios of cyber attacks where their data are generated by setting up virtual machines, software, and triggering attack actions along with manual labeling of behavior.

- **C01 - 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).
- **C02 - 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.
- **C03 - 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)
- **C04 - 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)
- **C05 - 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.
- **C06 - 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.
- **C07 - 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.
- **C08 - 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.
- **C09 - 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.
- **C10 - 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.
- **C11 - 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.

E. 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.

III. Evaluation

This section presents the evaluation of ProvSec datasets. We created a total of eleven 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 total of eleven real attack scenarios for this dataset. The details for these cases are illustrated in Figure 2 and 4, which respectively show the full attack behavior of C02 and C03 scenarios. 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.

The graph complexity of each case is presented in Table I. $|N|$ represents the total number of nodes where $|N_p|$, $|N_f|$, and $|N_n|$ respectively represent the counts of nodes for processes, files, and network entities like IP addresses. Similarly, $|E|$ represents the total number of edges. $|E_p|$, $|E_f|$, and $|E_n|$ respectively show the total counts of edges for processes, files, and network entities.

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%.

As an example, Figure 3 shows the sequence of processes caused by the path traversal exploit (C02 scenario). Figure 5

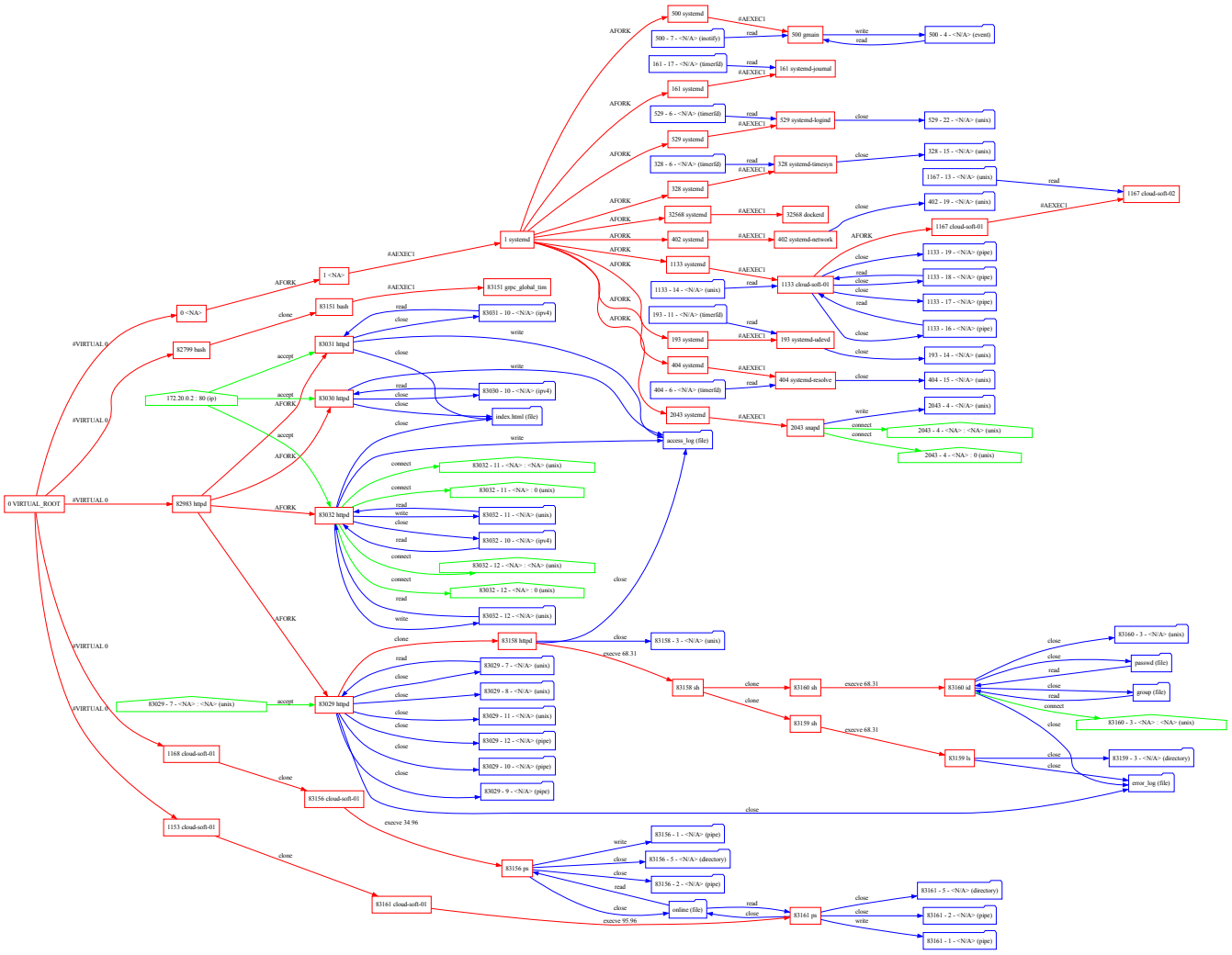

Fig. 2. Dependency Graph of C02 - Path traversal and file disclosure vulnerability in Apache HTTP Server.

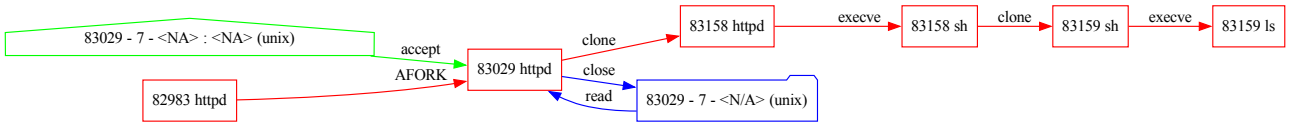

Fig. 3. Simplified Backtrack Graph of C02 - Path traversal and file disclosure vulnerability in Apache HTTP Server.

highlights a part of the original graph by removing unrelevant nodes and edges of the C03 scenario.

IV. Related Work

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

Network-oriented dataset Many existing work focuses on network-oriented data such as five-tuples or full packet recordings (e.g., PCAP) [17]–[22]. 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 [23]–[26] 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

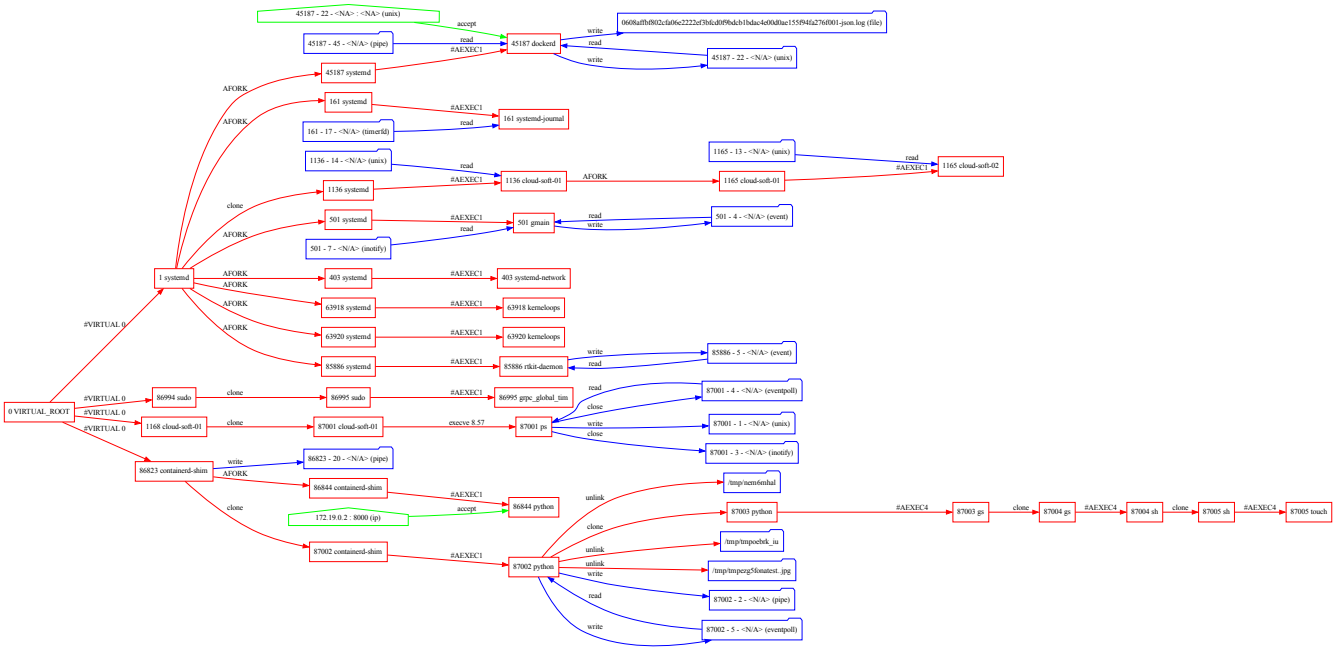

Fig. 4. Dependency Graph of C03 - Python PIL/Pillow Remote Shell Command Execution via Ghostscript.

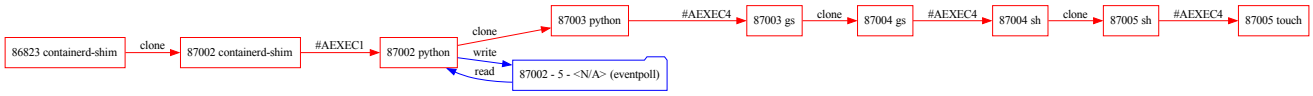

Fig. 5. Simplified Backtrack Graph of C03 - Python PIL/Pillow Remote Shell Command Execution via Ghostscript.

TABLE I
DETAILS OF THE PROVENANCE GRAPHS OF INCIDENT CASES.

#	Name	CVE	$\|N\|$	$\|N_p\|$	$\|N_f\|$	$\|N_n\|$	$\|E\|$	$\|E_p\|$	$\|E_f\|$	$\|E_n\|$	$\|N_{bt}\|$	$\|E_{bt}\|$	$\frac{\|N_{bt}\|}{\|N\|}$	$\frac{\|E_{bt}\|}{\|E\|}$
1	nginx	CVE-2017-7529	124	48	48	48	2758	42	2572	144	4	3	3.2%	0.11%
2	apache	CVE-2021-41773	99	45	45	9	1044	40	946	58	8	8	8.1%	0.77%
3	ghostscript	CVE-2018-16509	56	36	18	2	105	35	68	2	10	10	17.9%	9.5%
4	php	CVE-2018-19518	212	70	110	32	1454	64	1283	107	11	13	5.2%	0.89%
5	log4j	CVE-2021-44228	601	110	445	46	6177	135	5915	127	9	10	1.5%	0.16%
6	tomcat	CVE-2020-1938	143	66	69	8	898	64	785	49	7	8	4.9%	0.89%
7	redis	CVE-2022-0543	80	57	27	3	202	51	143	8	10	10	12.5%	5.0%
8	consul	N/A	1203	281	392	530	34173	291	32789	1093	6	5	0.5%	0.015%
9	apache	CVE-2021-42013	117	48	52	17	1383	44	1268	71	8	8	6.8%	0.59%
10	django	CVE-2021-35042	176	42	41	93	1704	36	1402	266	5	5	2.8%	0.29%
11	docker	CVE-2019-5736	1764	358	1263	143	47183	557	46272	354	295	729	16.7%	1.55%

parameters which are necessary to conduct system provenance analysis.

System call dataset ISOT-CID dataset [27] includes data of multiple formats including network traffic, system logs, performance data (e.g. CPU utilization), and system calls. While this data is 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.

Compared to these approaches, ProvSec has several advantages. Our dataset has full details of system calls and parameters that enable the construction of operating system dependencies and system provenance analysis. We utilized real vulnerabilities to simulate attack scenarios inside docker environments which are recorded in the operating system kernel. Our manual labeling of the attacks 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.

V. Conclusion

In this paper, we introduce a new dataset for security provenance analysis. This dataset is differentiated from past work in that it provides detailed data with causal dependencies across events, it uses real vulnerabilities and exploits, and it provides manual labeling so that it can be used for validation and supervised machine learning tasks. We performed an enhanced causality dependence analysis and demonstrated how the dependency analysis can simplify the analysis of each attack scenario.

VI. Acknowledgement

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.

References

[1] Bloomberg, "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/.

[2] Reuters, "Toshibas european business hit by cyberattack," https://www.reuters.com/business/autos-transportation/toshibas-european-business-hit-by-cyberattack-source-2021-05-14/ .

[3] O. of the Superintendent, "Cybersecurity attack on the buffalo public schools."

[4] S. Magazine, "Now ransomware is inundating public school systems," https://www.securitymagazine.com/articles/95164-now-ransomware-is-inundating-public-school-systems.

[5] . N. Oklahoma, "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 .

[6] CNN, "Kaseya ransomware attack businesses affected," https://www.cnn.com/2021/07/06/tech/kaseya-ransomware-attack-businesses-affected/index.html.

[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/.

[8] ——, "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/.

[9] Y. Liu, M. Zhang, D. Li, K. Jee, Z. Li, Z. Wu, J. Rhee, and P. Mittal, "Towards a timely causality analysis for enterprise security." in *NDSS*, 2018.

[10] Q. Wang, W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, J. Rhee, Z. Chen, W. Cheng, C. A. Gunter *et al.*, "You are what you do: Hunting stealthy malware via data provenance analysis." in *NDSS*, 2020.

[11] Z. Xu, Z. Wu, Z. Li, K. Jee, J. Rhee, X. Xiao, F. Xu, H. Wang, and G. Jiang, "High fidelity data reduction for big data security dependency analyses," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[12] Y. Tang, D. Li, Z. Li, M. Zhang, K. Jee, X. Xiao, Z. Wu, J. Rhee, F. Xu, and Q. Li, "Nodemerge: Template based efficient data reduction for big-data causality analysis," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[13] W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, D. Wang, Z. Chen, Z. Li, J. Rhee, J. Gui *et al.*, "This is why we can't cache nice things: Lightning-fast threat hunting using suspicion-based hierarchical storage," in *Annual Computer Security Applications Conference*, 2020.

[14] S. Ma, K. H. Lee, C. H. Kim, J. Rhee, X. Zhang, and D. Xu, "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, 2015. [Online]. Available: https://doi.org/10.1145/2818000.2818039

[15] Y. Sun, K. Jee, S. Sivakorn, Z. Li, C. Lumezanu, L. Korts-Parn, Z. Wu, J. Rhee, C. H. Kim, M. Chiang *et al.*, "Detecting malware injection with program-dns behavior," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2020.

[16] S. T. King and P. M. Chen, "Backtracking intrusions," in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, ser. SOSP '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 223–236. [Online]. Available: https://doi.org/10.1145/945445.945467

[17] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham, and M. Zissman, "Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation," in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, vol. 2, 2000, pp. 12–26 vol.2.

[18] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[19] Y. M. Banadaki, "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, 2020. [Online]. Available: http://pubs.sciepub.com/jcsa/8/2/2

[20] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "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, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18327687

[21] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *International Conference on Information Systems Security and Privacy*, 2018.

[22] M. Jonker, A. King, J. Krupp, C. Rossow, A. Sperotto, and A. Dainotti, "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, 2017, p. 100–113. [Online]. Available: https://doi.org/10.1145/3131365.3131383

[23] A. Gkortzis, D. Mitropoulos, and D. Spinellis, "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, 2018, p. 18–21. [Online]. Available: https://doi.org/10.1145/3196398.3196454

[24] V. Nguyen, "Some software vulnerability real-world data sets," 2021. [Online]. Available: https://dx.doi.org/10.21227/1m98-5h52

[25] D. Kim, E. Kim, S. K. Cha, S. Son, and Y. Kim, "Revisiting binary code similarity analysis using interpretable feature engineering and lessons learned," *CoRR*, vol. abs/2011.10749, 2020. [Online]. Available: https://arxiv.org/abs/2011.10749

[26] A. Marcelli, M. Graziano, X. Ugarte-Pedrero, Y. Fratantonio, M. Mansouri, and D. Balzarotti, "How machine learning is solving the binary function similarity problem," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 2099–2116. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/marcelli

[27] A. Aldribi, I. Traore, and B. Moa, *Data Sources and Datasets for Cloud Intrusion Detection Modeling and Evaluation*. Cham: Springer International Publishing, 2018, pp. 333–366. [Online]. Available: https://doi.org/10.1007/978-3-319-73676-1$_1$3