System-Auditing, Data Analysis and Characteristics of Cyber Attacks for Big Data Systems

Liangyi Huang
Arizona State University
Tempe, AZ, USA
lhuan139@asu.edu

Sophia Hall
Case Western Reserve University
Cleveland, OH, USA
srh95@case.edu

Fei Shao
Case Western Reserve University
Cleveland, OH, USA
fxs128@case.edu

Arafath Nihar
Case Western Reserve University
Cleveland, OH, USA
axn392@case.edu

Vipin Chaudhary
Case Western Reserve University
Cleveland, OH, USA
vipin@case.edu

Yinghui Wu
Case Western Reserve University
Cleveland, OH, USA
yxw1650@case.edu

Roger French
Case Western Reserve University
Cleveland, OH, USA
rxf131@case.edu

Xusheng Xiao
Arizona State University
Tempe, AZ, USA
xusheng.xiao@asu.edu

ABSTRACT

Using big data, distributed computing systems such as Apache Hadoop requires processing massive amount of data to support business and research applications. Thus, it is critical to ensure the cyber security of such systems. To better defend from advanced cyber attacks that pose threats to even well-protected enterprises, system-auditing based techniques have been adopted for monitoring system activities and assisting attack investigation. In this demo, we are building a system that collects system auditing logs from a big data system and performs data analysis to understand how system auditing can be used more effectively to assist attack investigation on big systems. We also built a demo application that detects unexpected file deletion and presents root causes for the deletion.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**; • **Information systems → Parallel and distributed DBMSs**.

KEYWORDS

system auditing;big data systems;cyber attack investigation

ACM Reference Format:
Liangyi Huang, Sophia Hall, Fei Shao, Arafath Nihar, Vipin Chaudhary, Yinghui Wu, Roger French, and Xusheng Xiao. 2022. System-Auditing, Data Analysis and Characteristics of Cyber Attacks for Big Data Systems. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557185

CIKM '22, October 17–21, 2022, Atlanta, GA, USA.

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557185

1 INTRODUCTION

Big data distributed computing systems like Apache Hadoop [4], Hbase [7], Spark [2] or Cloudera Data Platform [3], play an important role in today's enterprises since they process massive amount of data to support business and research intelligence applications. For example, we currently host 800 TB of storage in our Hadoop clusters, storing timeseries and image datasets that originate from Terabyte-scale experiments in renewable energy areas of photovoltaic power plants and energy efficiency of buildings and other spatiotemporal research challenges that are advanced by data-driven analyses [8–11, 15, 17]. It is critical to protect these systems from cyber attacks, which have become the major threats to enterprises [1, 20].

Recent incidents show that advanced cyber attacks have compromised well-protected companies by exploiting multiple vulnerabilities. To better defend from these attacks, the emerging solution in both academia and industry is to adopt system monitoring [13, 14] to monitor system activities from the kernel and then perform causality analysis [1, 5] to identity the root causes of the attacks, thus *assisting attack investigation*. While these techniques have shown promising results in addressing advanced cyber attacks, they generally suffer from the big data issue [19, 21], as system auditing produces massive amount of logs[1]. This problem becomes worse as big data systems are much busier than normal computers and thus the collected logs are much larger.

To address the challenges in investigating cyber attacks on big data distributed computing systems, this demo aims to show the data characteristics of the system auditing logs collected from big data systems based on Hadoop/Hbase/Spark, and identify opportunities to promote research on system auditing for big data systems. More specifically, we build a demo monitoring system based on Sysdig [18] to collect system auditing logs from a big data system. A big data system usually consists of data nodes that store the big data in the Hadoop Distributed File System (HDFS) and the name nodes that manages the data nodes. To better understand the activities of these two types of nodes, we analyze these in the collected logs to

[1] 1 TB for monitoring 100 hosts for a month.

identify the hot entities (process/file/network connection) in these nodes, and compare them to reveal insights of their behavior. In addition, we build a demo detection system that detects unexpected file deletion on the important data stored in the big data systems and apply causality analysis to reveal the root causes of the deletion for assisting attack investigation.

Our results show that the *major activities across data nodes (i.e., the hot entities) are similar*, as these data nodes are handling different parts of the big data, *while the activities in the name nodes are different from those of the data nodes*. Our detection system can also detect unexpected file deletions and correctly identify the steps that lead to the deletion.

2 BACKGROUND

System Auditing. System auditing monitors system calls that are critical in security analysis, with the focus on the system events related to (1) file access, (2) processes creation and destruction, and (3) network access [13, 14]. Following the established trend, we consider a *system event* as the interaction between two system entities represented as ⟨*subject, operation, object*⟩ where subjects are processes (e.g., Chrome), and objects can be files, processes, and network connections. Both entities and events have critical security-related attributes, as shown in Table 1.

Causality Analysis. Causality analysis [5, 12] analyzes the system auditing events to infer their dependencies and present the dependencies as a directed graph, where nodes represent system entities and edges represent dependencies. Two entities are considered to have dependencies if they appear in the same system call, or there exists data flows that connect them. Causality analysis enables two important security applications: (1) *backward causality analysis* that identifies entry points of attacks, and (2) *forward causality analysis* that investigates ramifications of attacks. Given a Point-Of-Interest (POI) event such as an alarm event, a backward causality analysis traces back from the source node u to find all events that have causal dependencies on u, and a forward causality analysis traces forward from the sink node v to find all events on which v has causal dependencies.

3 ARCHITECTURE

Overview of architecture. Figure 1 shows the architecture of our demo system. Our demo system consists of three major components: (1) the system monitoring component collects system auditing logs and generates the corresponding event graphs, (2) the hot entity summarization component analyzes the generated event graphs and compute the similarity of different hosts in the big data system, and (3) the unexpected activity detection component detects and reveals the detail about unexpected system activities.

System Monitoring. This component monitors system activities such as file operations and network accesses for further analysis. We adopt Sysdig, an industry leading tool for system auditing, to collect the information of all system calls from OS kernels. Sysdig records the system calls in temporal order and outputs the collected information as log files in both the txt format and the scap format. Linux abstracts various functions into files, so the 'Read/Write' monitoring function of Sysdig actually has the ability to monitor hardware devices such as a printer or a network adapter.

Table 1: Representative attributes of system call events

Operation	Read/Write, Execute, Start/End
Time	Start Time/End Time, Duration
Misc.	Subject/Object ID, Data Amount, Failure Code

Table 2: Node coverage for the top-K hot entities

Top-K	192.168.x.a	192.168.x.b	192.168.x.c	192.168.x.d
10	82%	83%	81%	64%
100	89%	89%	87%	82%

To enable further analysis of the log files, we builds the graph representation of the events recorded in the logs file, following our prior works [1, 20]. To better represent the relationship of these system activities, we convert these events into a temporal graph, where a node represents a unique file/process/network connection, and an edge connecting two nodes represents a system call event involving two entities. A one-hour log file from a data node typically contains about 10 million events and consumes about 2GB storage space. The temporal graph of this file needs only 50MB storage space to store the 1.5 million nodes and events. With this graph, security analysts can inspect the graph to understand how system entities are interacting with each other.

Hot Entity Summarization. To understand the major activities in big data systems, we perform a graph analysis on the built event graphs from both the name nodes and the data nodes. We count the degree of incoming/outgoing edges for each node and identify the "hot" entities that have the highest degrees, indicating the most active processes or files in the hosts. We then extract the sub-graphs of these nodes that represent their major activities and study these activities to better understand the behaviors of big data systems. Additionally, we perform similarity analysis by comparing the hot entities from different nodes in the big data system. As shown in Table 2, the top 10 hot entities are connected to 64% of the entities, and the top 100 hot entities are connected to 82% of the entities. Thus, for each node in the big data data system, we choose the 100 processes/files/network connections that have the highest degrees of incoming/outgoing edges. Then we compare the similarity of these entities' names to see whether the major activities among these hosts are similar or very different.

Unexpected Activity Detection. To demonstrate the effectiveness of our system, we build a detection component that monitors unexpected file deletion and reports the root causes of the detected deletions. We build our detection based on a Python package, Watchdog [16], to monitor a folder that contains important data and detect any file changes, especially file deletion. Once a change is detected, our system performs causality analysis by using the file change event as the Point-Of-Interest (POI) event and obtaining the dependency graph for the file deletion. This graph typically contains contains the deleted file, the process that runs the delete command, and the other files that are also related to the process. With this graph as the contextual information, the detection system raises an alert to the administrator if the process is not in the whitelist.

4 DEMONSTRATION AND RESULTS

Deployment Environment. Our demonstration system is built upon a high performance computing (HPC) cluster that includes servers with different types of hardware and access permissions.

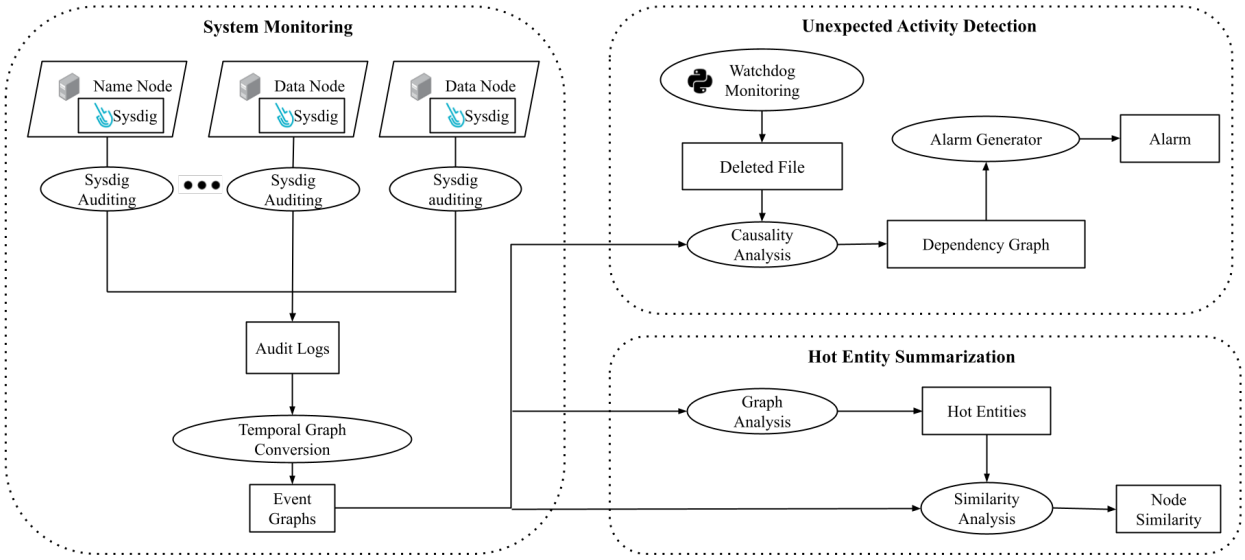

Figure 1: Architecture of the demo monitoring and detection system

Table 3: Top 10 hot entities in one data node (192.168.x.a)

System Entity	Entity Description	Functionality Description
rt.jar	Java 1.8 JAR Runtime	Use for support core 1.8 Java API
/etc/hosts	Host Table File	File that maps servers or hostnames to IP addresses
rt.jar	Java 1.7 JAR Runtime	Use for support core 1.7 Java API
java	Hadoop Distributed File System Process	Process for Hadoop Distributed File System
dcevst64.ini	Linux Configuration File	Meta package for installing all Server Administrator features (Dell Linux)
raw_text.tmpl	Raw Text Template	Text Template for process impala
libzip.so	Java Runtime Environment	Use for support 1.8 Java
libc.so.6	Symbolic Link to C Library	so. 6 is a symbolic link to the 64-bit x86 C library, used to run 64-bit executables
dcsdst64.ini	Dell Configuration Files	File for Dell pre-installed software
java	Apache Hadoop Yarn	The resource management and job scheduling technology in the open source Hadoop distributed processing framework.

Table 4: Top 10 hot entities in the name node (192.168.x.d)

System Entity	Entity Description	Functionality Description
192.168.x.d:40124-192.168.x.d:2181	Local Network Connection	Network connection at local with different ports
rt.jar	Java 1.8 JAR Runtime	Support core 1.8 Java API
pgstat.stat	Postgres Statistics File	File for postgres stats collector
postgres	Postgres Stats Collector Process	Collector for database statistics
java	Cloudera Service Monitor	Cloudera Manager to manage and monitor clusters.
/etc/hosts	Host Table File	File that maps servers or hostnames to IP
java	Apache Zookeeper	Distributed configuration and synchronization service,naming registry for distributed systems.
java	Hadoop Distributed File System Process	Process for HDFS
java	Cloudera Service Monitor	Cloudera Manager to manage and monitor clusters.
fsimage_0000 000000112550939	FSImage File	File of complete directory structure of the HDFS

We deploy our monitoring component in the HDFS (Hadoop Distributed File System) servers that are exclusive to MDLE members. The HDFS servers include two types of server nodes: name nodes and data nodes. The name node is the core of the HDFS system and the interface for normal users. The group members have a static interaction interface with IP address `192.168.x.a`. The data nodes are controlled by the name node and automatically perform the tasks of data segmentation, data storage, and data reads. These HDFS servers provide two types of services: HDFS file storage service and HBase database service. The file storage service provides an abstracted disk interface to store/read/modify files and folders. The HBase database service provides an abstract database interface to create/modify/delete table and insert/query/modify rows or columns within a table. Both services are distributed designs. Users interact with one interface, while the actual data is sliced and stored by multiple data nodes. A typical use of the server is that an MDLE researcher uses R scripts to read the files from the HDFS file storage service and store the results in the HBase dataset by using the HBase database service. Thus, these servers have much more activities than other systems studied in the prior works [1, 6, 20].

Table 5: Node similarity of datanodes and namenode

	192.168.x.a	192.168.x.b	192.168.x.c	192.168.x.d
192.168.x.a	–	49%	63%	23%
192.168.x.b	49%	–	49%	22%
192.168.x.c	63%	49%	–	23%
192.168.x.d	23%	22%	23%	–

We have deployed Sysdig on 4 server nodes, including 1 name node with the IP address `192.168.x.d` and 3 data nodes with IP addresses `192.168.x.a/b/c`. It is scheduled to run for one hour every

day using a cronjob, which will also automatically move the log files to the logs directory.

Hot Entity Summarization. Our study aims to determine whether the information in the auditing logs can capture (1) the major activities in the big data system and (2) the unique behaviors for the name node and the data nodes. We convert the auditing logs collected from sysdig on these nodes into temporal graphs and study the data characteristics of the nodes with the most incoming/outgoing edges. We refer the top-ranked nodes as "hot entities". We measure the similarity of two nodes using the number of common hot entities in their top K hot entities:

$$Similarity = \frac{\#\ of\ common\ top\ K\ hot\ entities}{\#\ of\ top\ K\ entities} \tag{1}$$

Below is the summary of our findings:

- For both name nodes and data nodes, we find that the temporal graphs are highly similar across different days, and thus the hot entities are almost the same every day. This indicates that these nodes perform regular data processing tasks and exhibit very different behavior patterns than the other types of systems such as work stations and database servers studied in the prior works.
- For data nodes, we find that the temporal graphs for different data nodes have similar entities(processes, files, and network connections). When we only focus on the top 10 entities with the most incoming/outgoing edges, the similarity is above 70%. If we focus on the top 100 entities with the most incoming/outgoing edges, as shown in Table 5, the similarities between the data nodes are still above 50%.
- When we compute the similarities for the hot entities between the data nodes and the name node, the similarities are low, indicating that they are very different. Table 5 shows the computed similarities for the top 100 nodes in the data nodes (192.168.x.a/b/c) and the name node (192.168.x.d). As we can see, the similarity is merely 23%.
- When we plot the temporal graphs of the data nodes and the name node, we can clearly see the differences between the data nodes and the name node due to the different daily tasks performed by the hot entities. Table 4 and Table 3 provide more details about the roles of the top 10 hot entities in the temporal graphs.
- These results show that major activities of these hosts follow specific patterns and present opportunities for *data compression on a single host and across all data nodes*. Furthermore, the patterns of major activities will *allow machine learning techniques to perform anomaly detection effectively*.

Unexpected Activity Detection. To demonstrate the effectiveness of our system, we have built a detection system for detecting unexpected system activities, focusing on file deletion. In our daily works that use the HDFS server, we rarely delete files because every data file is valuable to us, and we primarily work at the HPC Virtual Machines and only use the HDFS server as a storage space. We prefer to let lab members move data files rather than delete them because of the low costs of the hard disks, and we use the Linux permission system to ensure that only administrators can delete files. Nevertheless, if an attacker gains administrator access to the HDFS server or any insider has a malicious intention, these valuable files can be tampered or even deleted. Thus, our task is to monitor any unexpected file operations for the folder that contains the valuable files. Note that due to the distributed nature of the HDFS servers, we have no idea which hosts actually store these files. But our auditing tool can work seamlessly inside the containers of the HDFS servers and can effectively monitor the system activities.

To illustrate the effectiveness of the detection system, we placed a file in the monitored folder and used the JupyterLab delete button to delete the file. The python program based on Watchdog was set to monitor all file deletion events and detected that a file deletion event happened at the target folder. So it starts to run causality analysis using the file deletion event to obtain the dependency graph for further analysis.

By inspecting the dependency graph provided in the alarm, the administrator found out that the JupyterLab process deleted the file `target1.txt` and copied the file to another folder before deleting it. In addition, the file `target2.txt` and other files were copied to the same folder as `target1.txt`'s' and then deleted by JupyterLab. The JupyterLab was also interacting with a file 'nohup.out', indicating some of its background processing activities. After manually inspecting the file, the administrator realized that JupyterLab stored the terminal information in the file 'nohup.out'.

Summary. This demo illustrates the effectiveness of our demo system for monitoring sensitive system activities in big data systems, promoting further research in this direction. Existing tools such as the watchdog tool can only monitor which files are deleted or modified but cannot identify the correlated processes and files, providing limited support for security analysis. Furthermore, the size and the complexity of the raw outputs from the Sysdig tool makes it almost impossible for security analysts to obtain such detailed analysis through manual inspection, as indicated in the prior works [19, 21].

5 CONCLUSION

We have developed a monitoring system for big data systems based on system auditing tools. Our system collects the information of system calls and converts them into temporal graphs. Our system then analyzes these temporal graphs to identify hot entities that have most interactions with other system entities and also reveals the common and unique behaviors of different types of hosts in the big data systems. We further develop a detection system that can detect unexpected system activities in the big data systems and have shown its effectiveness using a cast study. In the demo presentation, we will show the audience how to build dependency graphs from sysdig logs, visualize the behaviors inferred from the graphs, and detect deletion events for the protected files.

ACKNOWLEDGMENTS

This research was funded by DOE-NNSA-LLNS Project, Materials Degradation and Lifetime Extension, under contract number B647887. Liangyi Huang and Xusheng Xiao's work are partially supported by the National Science Foundation under the grants CCF-2046953 and CNS-2028748. Wu is partially supported by NSF OAC-2104007. Chaudhary's work is partially supported by NSF award OAC-2137603.

REFERENCES

[1] 2021. DepImpact Project Website. https://github.com/usenixsub/DepImpact.

[2] 2022. Apache Spark™ - Unified Engine for Large-Scale Data Analytics. https://spark.apache.org/.

[3] 2022. Cloudera Data Platform (CDP). https://www.cloudera.com/products/cloudera-data-platform.html.

[4] Apache Software Foundation. 2022. Hadoop. https://hadoop.apache.org.

[5] Ashvin Goel, Kenneth Po, Kamran Farhadi, Zheng Li, and Eyal de Lara. 2005. The Taser Intrusion Recovery System. In *Proceedings of the ACM Symposium on Operating systems principles (SOSP)*. 163–176.

[6] Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. 2019. NODOZE: Combatting Threat Alert Fatigue with Automated Provenance Triage. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

[7] HBase. 2022. Apache HBase. https://hbase.apache.org/.

[8] Mohammad Akram Hossain, Arash Khalilnejad, Rojiar Haddadian, Ethan M. Pickering, Roger H. French, and Alexis R. Abramson. 2021. Data Analytics Applied to the Electricity Consumption of Office Buildings to Reveal Building Operational Characteristics. *Advances in Building Energy Research* 15, 6 (Nov. 2021), 755–773. https://doi.org/10.1080/17512549.2020.1730239

[9] Y. Hu, V. Y. Gunapati, P. Zhao, D. Gordon, N. R. Wheeler, M. A. Hossain, T. J. Peshek, L. S. Bruckman, G. Zhang, and R. H. French. 2017. A Nonrelational Data Warehouse for the Analysis of Field and Laboratory Data From Multiple Heterogeneous Photovoltaic Test Sites. *IEEE Journal of Photovoltaics* 7, 1 (Jan. 2017), 230–236. https://doi.org/10.1109/JPHOTOV.2016.2626919

[10] Ahmad Maroof Karimi, Yinghui Wu, Mehmet Koyuturk, and Roger H French. 2021. Spatiotemporal Graph Neural Network for Performance Prediction of Photovoltaic Power Systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. Association for the Advancement of Artificial Intelligence, Virtual, 8.

[11] Arash Khalilnejad, Ahmad M. Karimi, Shreyas Kamath, Rojiar Haddadian, Roger H. French, and Alexis R. Abramson. 2020. Automated Pipeline Framework for Processing of Large-Scale Building Energy Time Series Data. *PLOS ONE* 15, 12 (Dec. 2020), e0240461. https://doi.org/10.1371/journal.pone.0240461

[12] Taesoo Kim, Xi Wang, Nickolai Zeldovich, and M. Frans Kaashoek. 2010. Intrusion Recovery Using Selective Re-execution. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 89–104.

[13] Samuel T. King and Peter M. Chen. 2003. Backtracking intrusions. In *Proceedings of the ACM Symposium on Operating systems principles (SOSP)*. ACM, 223–236.

[14] Samuel T. King, Zhuoqing Morley Mao, Dominic G. Lucchetti, and Peter M. Chen. 2005. Enriching Intrusion Alerts Through Multi-Host Causality. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*.

[15] Jiqi Liu, Menghong Wang, Alan J. Curran, Erdmut Schnabel, Michael Köhl, Jennifer L. Braid, and Roger H. French. 2021. Degradation Mechanisms and Partial Shading of Glass-Backsheet and Double-Glass Photovoltaic Modules in Three Climate Zones Determined by Remote Monitoring of Time-Series Current–Voltage and Power Datastreams. *Solar Energy* 224 (Aug. 2021), 1291–1301. https://doi.org/10.1016/j.solener.2021.06.022

[16] Yesudeep Mangalapilly. 2014. Watchdog: Filesystem Events Monitoring. https://github.com/gorakhargosh/watchdog.

[17] Ruben Mayer and Hans-Arno Jacobsen. 2020. Scalable Deep Learning on Distributed Infrastructures: Challenges, Techniques, and Tools. *Comput. Surveys* 53, 1 (Feb. 2020), 3:1–3:37. https://doi.org/10.1145/3363554

[18] Sysdig. 2017. Sysdig. https://sysdig.com/.

[19] Yu Tao Tang, Ding Li, Zhi Chun Li, Mu Zhang, Kangkook Jee, Xu Sheng Xiao, Zhen Yu Wu, Junghwan Rhee, Feng Yuan Xu, and Qun Li. 2018. NodeMerge: Template Based Efficient Data Reduction For Big-Data Causality Analysis. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. 1324–1337.

[20] Zhiqiang Xu, Pengcheng Fang, Changlin Liu Liu, Xusheng Xiao, Yu Wen, and Dan Meng. 2021. DEPCOMM: Graph Summarization on System Audit Logs for Attack Investigation. In *Proceedings of the IEEE Symposium on Security and Privacy (SP), San Francisco, CA*. 22–26.

[21] Zhang Xu, Zhenyu Wu, Zhichun Li, Kangkook Jee, Junghwan Rhee, Xusheng Xiao, Fengyuan Xu, Haining Wang, and Guofei Jiang. 2016. High Fidelity Data Reduction for Big Data Security Dependency Analyses. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. 504–516.