GrAALF:Supporting Graphical Analysis of Audit Logs for Forensics

Omid Setayeshfar[a], Christian Adkins[a], Matthew Jones[b], Kyu Hyung Lee[1a], Prashant Doshi[a]

[a] *University of Georgia, Athens, GA, 30602*
[b] *GTRI, Atlanta, GA 30332*

Abstract

System-level logs play a critical role in computer forensics. They capture interactions between programs and users in detail. However, a typical computer generates more than 2.5 million system events hourly, making finding malicious activities in such logs compute and time-intensive.

We introduce GrAALF a graphical system for efficiently loading, storing, processing, querying, and displaying system events for computer forensics. In comparison to similar systems, GrAALF offers the flexibility of storage, intuitive querying, and the tracing power for longer sequences of events in real-time to help identify attacks.

GrAALF is a robust solution for analysis to support computer forensics.

Keywords: Cyber Forensics, Provenance Tracking, Graphical Analysis

1. Introduction

Provenance data contained in system logs offers a rich source of information for computer forensics. The system logs can comprehensively capture how processes controlled by an attacker interact with resources such as disk and network. System events loggers such as Linux Audit [1], Sysdig [2], DTrace [3], and Event Trace for Windows (ETW) [4] are often used to generate these logs. Although logs are frequently analyzed offline after an attack has happened, real-time system monitoring can help the user in various ways: if forensic logs can be analyzed in real-time, this rich source of information allows investigation of ongoing abnormal behaviors and thus can protect the

[1]corresponding author; email:kyuhlee@uga.edu, address: 415 Boyd Graduate Studies Research Center, Dept. of Computer Science, Athens, Georgia, 30602

system more effectively. Also, timely attack investigations are essential to protecting the system from similar future attacks.

However, we see three main challenges in developing a practical system that can support (near) real-time analysis. First, system logs grow rapidly. We observe that a single host generates more than 2.5 million system events in an hour. Other studies [5, 6, 7] also reported that a single device generates over 3 GB of audit log daily. This challenge has motivated previous research into compression [5, 8, 9, 10, 11] and optimizing querying and pattern matching [12, 13].

Next, there exist various logging systems for different operating systems and architectures, and their definitions of the event entries, as well as output formats, are often different. For instance, Linux and Unix systems frequently use Linux Audit [1]. Windows systems mostly use Event Tracing for Windows (ETW) [4] to record system events. Linux and Windows have completely different system calls and system APIs. Furthermore, recent studies [14, 15, 16, 17] propose novel logging techniques to improve the effectiveness and efficiency of computer forensics. For instance, BEEP [14] proposes a technique to divide the long-running process into a finer-grain system object called *execution unit* to improve forensic accuracy. This heterogeneity makes it challenging to seamlessly support forensics on system logs generated by various logging platforms.

Third, backward and forward tracking techniques [6, 7] are essential in computer forensics to understand causal relations between system objects (e.g., process) and subjects (e.g., file, network socket). They often require tracking back to previous events to identify causal chains. In practice, database solutions are often used to store sequences of system events, and the user composes queries to identify causal relations between system components. However, the response time of backtrack queries in these traditional data stores is not acceptable as it takes over 20 minutes to extract causally dependent system objects for only a couple of files from a two-day system event log; thus can not provide the performance needed for real-time trackings. GrAALF flexibly offers the options of compressed in-memory storage, a traditional relational database system, and a graph database for storing the parsed audit logs. Because of their design nature, relational databases are highly optimized for storing tabular data; they may not perform well with a large number of chained joins. It can slow down operations that require an event stored in the database to be backtracked to its origin.

On the other hand, graph databases are designed with relationships in mind, making them much more efficient for backtracking relationships between events in a security log. However, graph databases tend to have low insertion performances, due to which they are unable to keep up with high-

speed streams of logged data. Consequently, graph databases and in-memory storage are well suited for (near) real-time forensics when mini-batches of data are inserted, and fast query execution on graphs is needed. For post-mortem analyses, the relational database is more appropriate as it allows fast loading, indexing, and subsequent querying of large amounts of system events.

GrAALF allows stored audit logs to be queried using a simple query language whose syntax and semantics are close to those of SQL. Importantly, and unlike SQL, this query language supports path queries and backtracking to an arbitrary depth from an identified resource. We note that the latter functionality is particularly crucial for successful forensics of prevalent computer attacks such as data exfiltration and kernel injections. User queries are parsed and interpreted by GrAALF's query and visualization layer; then, the satisfying data is displayed as a color-coded graph or tree that can be rearranged, focused, and magnified for study.

GrAALF belongs to a growing family of systems that support forensic analysis, which include Elastic [18], AIQL [12], SAQL [13], and Loglens [19]. However, these previous systems support simple keyword- and regular-expression based log data filtering only, and do not offer the useful backward trace query functionality. Furthermore, systems such as AIQL and SAQL are proprietary and not available for public use, in contrast with GrAALF, which is released under GNU AGPL V3.0 license[2].

2. Overview of GrAALF

We illustrate the high-level overview of GrAALF in Fig. 1. GrAALF consists of three layers: log ingestion, log storage, and query and visualization layers. The log ingestion layer receives streaming system logs from hosts in the enterprise and processes them. The output of this layer will be formatted data that represent causal relations between system subjects (e.g., process, thread) and objects (e.g., file, network socket).

The log storage layer stores the output from the log ingestion layer into a permanent database. GrAALF supports both relational and graph databases and allows a user to choose whichever is appropriate. GrAALF also has in-memory buffer storage to enable the processing of enormous streaming data from multiple sources.

The query and visualization layer interacts with a user to receive queries and provide output as an interactive graph. The user can iteratively make

[2] `https://github.com/omid-s/cyber_deception/`

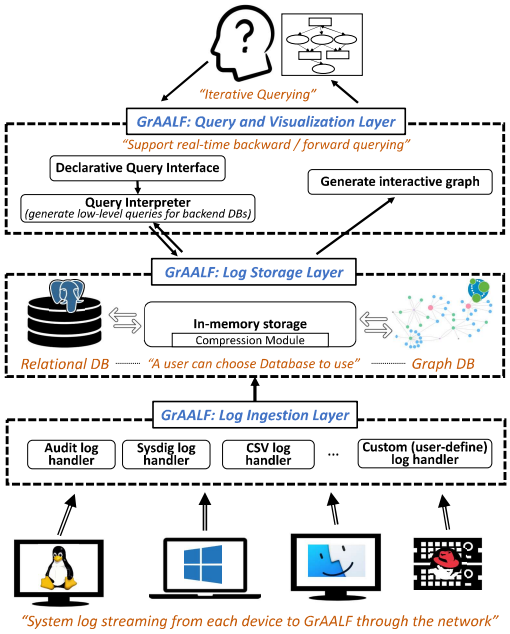

Figure 1: *A high level overview of GrAALF.*

queries based on the prior output graphs. The output graph visualizes causal relations in the system elements as well as interactions between different machines. More importantly, GrAALF supports (near) real-time backward and forward queries through which the user can easily understand the origin of each system component and how one affects other components as the changes happen. We carefully design the storage layer and query interpreter for querying in-memory storage with the permanent database seamlessly. As soon as the ingestion layer processes the logs and delivers the output to in-memory storage, they are ready for querying.

The query and visualization layer also accepts queries to monitor the system for both stability and security purposes; this module will continuously monitor graphs produced by these queries and will notify the user when changes happen in any of them. This enables automated monitoring as well as automated detection of potential security incidents as they happen.

3. Impact

Cyber threats are increasingly sophisticated and destructive. An important aspect of cyber attack investigation is understanding the details of attacks, what damage is caused, and who is responsible for the attack. It is essential to fully recover from past intrusions and for building more robust

defenses against future incidents. GrAALF benefits national security by advancing the state-of-the-art in cyber forensics and real-time threat detection.

First, GrAALF enables a detailed reconstruction and understanding of advanced cyber attacks. It will provide the foundations to fully recover from past intrusions and to build more robust defenses against future incidents.

Second, GrAALF provides real-time analysis that allows investigation of ongoing abnormal behaviors to protect the system timely and effectively. A user can use backward and forward queries to understand the origin of a system component and how one affects other system components in real-time.

Next, GrAALF's source code, toolsets, and datasets are disseminated to the public under GNU AGPL V3.0 license for empowering future research in this area. This system has been used in a recent study that focuses on identifying the phases of a cyber attack [20]. The result obtained using GrAALF shows that an attack phase can be classified with over 90% accuracy. Another study [21] leverages GrAALF to automatically extract behavior patterns that are used to study a cyber attacker's behavior and detect their intent.

4. Related Work

Tools such as Elastic Stack are widely adopted in industry and academia for their agility and high performance when dealing with logs. Plaso [22] and SOF ELK [23] are two of the tools built on Elastic Stack; these tools have rich visualization and parsing features.Such tools are not an effective tool for forensics and provenance tracking. They provide statistics-based reports, not provenance graph models, nor they provide users with much queryabilty beyond filtering.

Other tools such as [24, 22, 25] provide models based on artificial intelligence which will assist the forensics expert in monitoring the system and detecting malicious behaviors based on known patterns; however, these tools are not designed for manual forensics tasks such as whole system provenance tracking and are often bound to one scheme of proprietary data stream.

GrAALF is designed to give the user more flexibility in both data source and data exploration while maintaining features present in other works such as pattern matching on the streaming data and provenance tracking capabilities. Table 1 shows a comparison of GrAALF with some related works by their capabilities.

Several causality analysis techniques exist [6, 7, 26, 27, 28], which use system call loggers to record important system events at run time and analyze recorded events to find causal relations between system subjects (e.g., process) and system objects (e.g., file or network socket). For instance, BackTracker [6] and Taser [7] propose backward and forward analysis techniques

Table 1: *GrAALF compared to related systems.* × *shows lack of support,* □ *shows support with limited functionality, and* ■ *shows full support.* ***S*** *shows streaming analysis capability;* ***PG****, provenance tracking of long causal sequences;* ***F*** *schema flexibility;* ***O*** *openly available;* ***G*** *granularity smaller than process level;* ***Q*** *support for a query language.****A*** *automatic anomaly detection.*

Title	S	PG	F	O	G	Q	A
GrAALF	■	■	■	■	■	■	×
AIQL	×	□	×	×	×	■	×
SAQL	■	□	×	×	×	■	×
SOF ELK	×	×	□	□	×	□	■
Carbon Black LiveOps	■	□	×	×	×	□	■
NoDoze	×	□	×	×	×	×	■
Plaso	×	×	□	□	×	□	■

in order to analyze system call logs and construct causal graphs for effective attack investigation. Recently, a series of works [14, 8, 29, 30, 31] have proposed to provide accurate and fine-grained attack analysis. They divide long-running processes into multiple autonomous execution units and identify causal dependencies between them. LDX [32] proposes a dual execution-based causality inference technique. When a user executes a process, LDX automatically starts a slave execution by mutating input sources. Then LDX identifies causal dependencies between the input source and outputs by comparing the outputs from the original execution and the slave execution. MCI [30] leverages LDX [32] to build a model-based causality inference technique for audit logs to infer causal relations between system calls.

GrAALF can support all of the proposed logging techniques with the proper definition of subjects, objects, and relations. We can process their logs in real-time and provide GrAALF's query interface to the user to enable interactive investigation and monitoring of the system.

5. Conclusion

We present GrAALF, a graphical forensic analysis system for efficient loading, storing, processing, querying, and displaying of causal relations extracted from system events to support computer forensics. GrAALF offers the flexibility of choice between a relational database (e.g., PostgreSQL) and a graph database (e.g., Neo4j) for backend storage. GrAALF's in-memory storage can be seamlessly integrated with either backend and provides (near) real-time forensic analysis of streaming system event logs. GrAALF provides a simple query language whose syntax and semantics are close to SQL. The user can compose a query to perform a backward and forward analysis to identify causal relations between system subjects and objects. We use an extensive system call audit log that contains realistic attacks to demonstrate

the practical utility of GrAALF. We will release the source code of GrAALF as open-source software.

Acknowledgements

This research is funded in part by grant # W911NF-18-1-0288 from the Army Research Office. Authors would also like to thank Sean Frankum, Aditya Shinde and Muhammed AbuOdeh for their valuable inputs.

References

[1] R. Inc., Redhat linux audit, `https://people.redhat.com/sgrubb/audit/`, [Online; accessed 25 May 2019] (2019).

[2] I. sysdig, Sysdig, `https://sysdig.com/`, [Online; accessed 25 May 2019] (2019).

[3] Dtrace, `http://dtrace.org/blogs/about`, [Online; accessed 25 May 2019] (2019).

[4] Event tracing for windows, `https://docs.microsoft.com/en-us/windows/desktop/etw/event-tracing-portal`, [Online; accessed 25 May 2019] (2019).

[5] K. H. Lee, X. Zhang, D. Xu, Loggc: garbage collecting audit log, in: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, ACM, 2013, pp. 1005–1016.

[6] S. T. King, P. M. Chen, Backtracking intrusions, ACM SIGOPS Operating Systems Review 37 (5) (2003) 223–236.

[7] A. Goel, K. Po, K. Farhadi, Z. Li, E. De Lara, The taser intrusion recovery system, ACM SIGOPS Operating Systems Review 39 (5) (2005) 163–176.

[8] S. Ma, X. Zhang, D. Xu, Protracer: Towards practical provenance tracing by alternating between logging and tainting., in: Proceedings of the Network and Distributed System Security Symposium (NDSS), 2016.

[9] Z. Xu, Z. Wu, Z. Li, K. Jee, J. Rhee, X. Xiao, F. Xu, H. Wang, 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, ACM, 2016, pp. 504–516.

[10] M. N. Hossain, J. Wang, O. Weisse, R. Sekar, D. Genkin, B. He, S. D. Stoller, G. Fang, F. Piessens, E. Downing, et al., Dependence-preserving data compaction for scalable forensic analysis, in: 27th USENIX Security Symposium (USENIX Security 18), 2018, pp. 1723–1740.

[11] Y. Tang, D. Li, Z. Li, M. Zhang, K. Jee, X. Xiao, Z. Wu, J. Rhee, F. Xu, 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, ACM, 2018, pp. 1324–1337.

[12] P. Gao, X. Xiao, Z. Li, F. Xu, S. R. Kulkarni, P. Mittal, Aiql: Enabling efficient attack investigation from system monitoring data, in: 2018 USENIX Annual Technical Conference (USENIX ATC 18), USENIX, 2018, pp. 113–126.

[13] P. Gao, X. Xiao, D. Li, Z. Li, K. Jee, Z. Wu, C. H. Kim, S. R. Kulkarni, P. Mittal, Saql: A stream-based query system for real-time abnormal system behavior detection, in: 27th USENIX Security Symposium (USENIX Security 18), USENIX, 2018, pp. 639–656.

[14] K. H. Lee, X. Zhang, D. Xu, High accuracy attack provenance via binary-based execution partition., in: Proceedings of the Network and Distributed System Security Symposium (NDSS), 2013.

[15] D. J. Pohly, S. McLaughlin, P. McDaniel, K. Butler, Hi-fi: collecting high-fidelity whole-system provenance, in: Proceedings of the 28th Annual Computer Security Applications Conference, ACM, 2012, pp. 259–268.

[16] A. Bates, D. J. Tian, K. R. Butler, T. Moyer, Trustworthy whole-system provenance for the linux kernel, in: 24th USENIX Security Symposium (USENIX Security 15), 2015, pp. 319–334.

[17] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, M. I. Seltzer, Provenance-aware storage systems., in: USENIX Annual Technical Conference, General Track, 2006, pp. 43–56.

[18] E. B.V., Elasticsearch, `https://www.elastic.co/`, [Online; accessed 25 May 2019] (2019).

[19] B. Debnath, M. Solaimani, M. A. G. Gulzar, N. Arora, C. Lumezanu, J. Xu, B. Zong, H. Zhang, G. Jiang, L. Khan, Loglens: A real-time log analysis system, in: 2018 IEEE 38th International Conference on

Distributed Computing Systems (ICDCS), IEEE, IEEE, 2018, pp. 1052–1062.

[20] M. AbuOdeh, C. Adkins, O. Setayeshfar, P. Doshi, K. H. Lee, A novel ai-based methodology for identifying cyber attacks in honey pots, in: IAAI-2021, AAAI, 2021.

[21] A. P. Shinde, Active cyber deception and attacker intent recognition using factored interactive pomdps, Ph.D. thesis, University of Georgia (2020).

[22] Plaso, https://plaso.readthedocs.io/en/latest/, [Online; accessed 25 May 2019] (2019).

[23] L. Lewes Technology Consulting, Sof-elk® virtual machine distribution, https://github.com/philhagen/sof-elk/blob/master/VM_README.md, [Online; accessed 25 May 2019] (2019).

[24] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, A. Bates, Nodoze: Combatting threat alert fatigue with automated provenance triage., in: NDSS, 2019.

[25] I. Carbon Black, Cb liveops, https://www.carbonblack.com/products/cb-liveops/, [Online; accessed 25 May 2019] (2019).

[26] T. Kim, X. Wang, N. Zeldovich, M. F. Kaashoek, Intrusion recovery using selective re-execution, in: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, OSDI'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 89–104.
URL http://dl.acm.org/citation.cfm?id=1924943.1924950

[27] S. T. King, Z. M. Mao, D. G. Lucchetti, P. M. Chen, Enriching intrusion alerts through multi-host causality, in: Proceedings of the Network and Distributed System Security Symposium (NDSS), 2005.

[28] S. Krishnan, K. Z. Snow, F. Monrose, Trail of bytes: efficient support for forensic analysis, in: E. Al-Shaer, A. D. Keromytis, V. Shmatikov (Eds.), Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010, ACM, 2010, pp. 50–60. doi:10.1145/1866307.1866314.
URL https://doi.org/10.1145/1866307.1866314

[29] S. Ma, J. Zhai, F. Wang, K. H. Lee, X. Zhang, D. Xu, Mpi: Multiple perspective attack investigation with semantic aware execution partitioning, in: 26th USENIX Security Symposium (USENIX Security 17), 2017, pp. 1111–1128.

[30] Y. Kwon, F. Wang, W. Wang, K. H. Lee, W.-C. Lee, S. Ma, X. Zhang, D. Xu, S. Jha, G. F. Cretu-Ciocarlie, A. Gehani, V. Yegneswaran, Mci : Modeling-based causality inference in audit logging for attack investigation, in: Proceedings of the Network and Distributed System Security Symposium (NDSS), NDSS '18, 2018.

[31] S. Ma, J. Zhai, Y. Kwon, K. H. Lee, X. Zhang, G. Ciocarlie, A. Gehani, V. Yegneswaran, D. Xu, S. Jha, Kernel-supported cost-effective audit logging for causality tracking, in: 2018 USENIX Annual Technical Conference (USENIX ATC 18), USENIX Association, Boston, MA, 2018, pp. 241–254.
URL https://www.usenix.org/conference/atc18/presentation/ma-shiqing

[32] Y. Kwon, D. Kim, W. N. Sumner, K. Kim, B. Saltaformaggio, X. Zhang, D. Xu, LDX: causality inference by lightweight dual execution, in: T. Conte, Y. Zhou (Eds.), Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '16, Atlanta, GA, USA, April 2-6, 2016, ACM, 2016, pp. 503–515. doi:10.1145/2872362.2872395.
URL https://doi.org/10.1145/2872362.2872395

Required Metadata

Current code version

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v3.0
C2	Permanent link to code/repository used for this code version	`https://github.com/omid-s/cyber_deception/`
C3	Permanent link to Reproducible Capsule	
C4	Legal Code License	GNU AGPL V3.0
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	Java
C7	Compilation requirements, operating environments & dependencies	Cross Platform, Java Runtime Environment 8+ needed
C8	If available Link to developer documentation/manual	For example: `https://github.com/omid-s/cyber_deception/`
C9	Support email for questions	kyuhlee@uga.edu, omid.s@uga.edu

Table 2: *Code metadata*