Intrusion Detection through Provenance-Based Histograms

Kenny Yu
Harvard University
kennyyu@college.harvard.edu

R. J. Aquino
Harvard University
rjaquino@college.harvard.edu

CS261, Fall 2013

Abstract

TODO

1 Introduction

Provenance is metadata that tracks the history of all changes to files. In provenance-aware storage systems like PASS [10] and PASSv2 [9], the file system automatically tracks all dependencies whenever a file is created, modified, or deleted. These dependencies include the command that was executed to modify the file, the environment of the command, and all input files. The generated provenance forms a directed acyclic graph of typed nodes (e.g. files, processes, pipes) with properties (e.g. name, execution time, pid) and typed edges (e.g. forked, input, versioning).

To make sense of the large amounts of data, Macko et. al. have developed Orbiter, a tool to visualize provenance graphs with semantic zoom [7]. Margo et. al. have developed techniques to analyze large provenance graphs and to extract useful information from these graphs, including semantic file attributes [8]. Using simple provenance statistics on nodes (e.g. neighbors, file count, process count, edge count) and features from `stat`, they determine with high accuracy the type of files (e.g. source files, configuration files). Furthermore, they have identified key properties of provenance graphs and developed new centrality metrics to perform local clustering of graphs, separating the huge provenance graphs into smaller semantic tasks or workloads [6].

Applications that require provenance collection typically place great emphasis on data integrity. Given this priority, a natural goal of provenance systems is to detect intrusions on the system. Somayaji et. al. have developed techniques for detecting intrusions based on system call sequences, and counteract these by exponentially delaying or aborting system calls, rendering the system useless for a malicious attacker [11][2]. These intrusions include exploiting vulnerabilities in the SSH daemon and sendmail to obtain a shell with root privileges. Somayaji's work relies on building a *normal* profile of a process, and then detecting when the process deviates from normal.

Given the extensive amount of data PASS collects on file-file, file-process, and process-process dependencies, it seems plausible to detect intrusions based on provenance data collected by PASS. Tariq et. al. generalize Somayaji's sequences of system calls to sequences of provenance system events [12] to develop an intrusion detection system (IDS) in a distributed system. Their system uses bloom filters to store hashes of k-tuples–representing a sequence of provenance events–and then use these bit vectors to correlate anomalies across multiple hosts. King et. al. have developed Backtrack [3], a modified Linux kernel that tracks dependencies between operating system objects (files, processes, file names) in a similar fashion to PASS. Given an intrusion, Backtrack builds a backwards casual graph to determine the entry point of an intrusion, and they have extended Backtrack with forward causal graphs to determine possibly tainted files, processes, and hosts from an intrusion in a distributed system [4]. However, their system does not find a way of detecting an intrusion. Lei et. al. build similar process-file dependency trees, which they call access trees, and define a compatibility function to compare access trees with the goal of predicting future file accesses based on the current access tree pattern [5].

Despite the existing work to detect intrusions and anomalies, Cao et. al. argue that any intrusion detection system cannot always distinguish good behaviors from bad behaviors because users behave too randomly to have an accurate model of the user's behavior [1]. They propose their fuzzy anomaly detection approach to extend the training models in existing intrusion detection systems to account for the inherent fuzzy nature of intrusion detection. When training an IDS, instead of assigning hard 0s or 1s for features, their approach provides a way to reliably map these features to weights in the interval $[0, 1]$.

In this paper, we present a technique to analyze existing provenance data to detect if an intrusion occurred.

TODO: TALK ABOUT OUR TECHNIQUES. We build on TODO's work and extend ... blah blah blah ... combine techniques together ... blah blah. The contributions of this paper are:

1. TODO: the techniques we use to analyze provenance data
2. TODO: types of intrusions that this system can detect, and the accuracy in detecting intrusions
3. TODO: future work for using provenance data to detect intrusions

2 Design and Implementation

TODO: ideas:

1. somayaji: sequence of system calls
2. semantic file attributes: determine types of files. use simple statistics on files.
3. histograms: use simple statistics and build histogram of file attributes on a process name/file
4. local clustering: use centrality metrics
5. fuzzy rules
6. hash k-tuples of provenance events and compare bit vectors
7. backwards and forwards causality graphs
8. access compatibility $\frac{1}{2}(C_d + C_e)$
9. intersect good DAG and bad DAG and look at intersection of DAG

3 Evaluation

3.1 Evaluation Methodology

TODO: types of exploits we ran. how we ran it. how we collected the data. the software we used.

3.2 Results

TODO: true positives. false positives. graphs?

3.3 Discussion

TODO: why the numbers are the way they are. what types of intrusions can we detect. which provenance data is most useful

4 Conclusion and Future Work

TODO not realtime. can it be made to detect intrusions in realtime?

References

[1] Cao, D., Qiu, M., Chen, Z., Hu, F., Zhu, Y., and Wang, B. Intelligent Fuzzy Anomaly Detection of Malicious Software. In *Internal Journal of Advanced Intelligence*, vol. 4, no. 1, pp 69-86 (December 2012).

[2] Inoue, H. and Somayaji, A. Lookahead Pairs and Full Sequences: A Tale of Two Anomaly Detection Methods. In *2nd Annual Symposium on Information Assurance* (June 2007).

[3] King, S. T. and Chen, P. M. Backtracking Intrusions. In *SOSP'03 Proceedings of the nineteenth ACM symposium on Operating systems principles* (December 2003).

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

[5] Lei, H. and Duchamp, D. An Analytical Approach to File Prefetching. In *Proceedings of the USENIX 1997 Annual Technical Conference* (January 1997).

[6] Macko, P., Margo, D., Seltzer, M. Local Clustering in Provenance Graphs (Extended Version). In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management* (August 2013).

[7] Macko, P. and Seltzer, M. Provenance Map Orbiter: Interactive Exploration of Large Provenance Graphs. In *TaPP'11 Proceedings of the 2nd conference on Theory and practice of provenance* (June 2011).

[8] Margo, D., and Smogor, R. Using Provenance to Extract Semantic File Attributes. In *TaPP'10 Proceedings of the 2nd conference on Theory and practice of provenance* (February 2010).

[9] Muniswamy-Reddy, K., Braun, U., Holland, D. A., Macko, P., Maclean, D., Margo, D., Seltzer, M., and Smogor, R. Layering in Provenance Systems. In *Proceedings of the 2009 USENIX Annual Technical Conference* (June 2009).

[10] Muniswamy-Reddy, K., Holland, D. A., Braun, U., and Seltzer, M. Provenance-Aware Storage Systems. In *Proceedings of the 2006 USENIX Annual Technical Conference* (June 2006).

[11] Somayaji, A. and Forrest, S. Automated Response Using System-Call Delays. In *Proceedings of the 2000 USENIX Annual Technical Conference* (August 2000).

[12] Tariq, D., Baig, B., Gehani, A., Mahmood, S., Tahir, R., Aqil, A., and Zaffar, F. Identifying the provenance of correlated anomalies. In *SAC'11 Proceedings of the 2011 ACM Symposium on Applied Computing* (March 2011).