Poster: What Behaviors Are in Your System Log Dataset?

Jason Liu, Andy Riddle, Kim Westfall, and Adam Bates
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
{jdliu2, rriddle2, kw26, batesa}@illinois.edu

***Abstract*—Despite their promising experimental performance, intrusion detection systems in practice are beset by false positives, leading to threat alert fatigue. We believe a fundamental cause of this disconnect is that datasets used to evaluate anomaly detection systems fail to meet the assumption that the benign data is representative of normal user behaviors. To analyze this issue, we present a method to visualize the behaviors system log datasets and apply it to various commonly-used datasets, as well as a VM used as one of our workstations to generate an example log containing realistic user behaviors. We then present a method to uncover the periodicity of behaviors arising from workload generators used in certain datasets.**

1. Introduction

Recent software attacks and vulnerabilities, e.g., Darkside ransomware attacks [1] and the Apache Log4j vulnerability [2], show a need for better attack detection and prevention systems. Outright prevention of attacks remains a lofty goal, as the potential attack surface for a system extends from software vulnerabilities — which are already difficult to eliminate in large and complex codebases — to social manipulation, e.g., phishing. Therefore, realtime monitoring systems to identify potential attacks are necessary to mitigate potential damage.

Many such intrusion detection systems (IDSs) are based on the insight of Forrest et al. that anomalous behaviors, which may be malicious, should be distinguishable from "normal" behavior [3]. During evaluation, these systems rely on log datasets being sufficiently representative of both normal and malicious behavior. If this assumption is not met, then it is unclear whether or not their evaluated performance will actually hold up to real data. Notably, all IDSs deployed in practice typically generate massive amounts of alerts, the majority of which are false positives, leading to the threat alert fatigue problem where security analysts are unable to respond to real attacks in time [6].

Verifying the robustness of benign behaviors is difficult. For example, a script that repeatedly runs command line utilities can easily generate non-attack data. However, to emulate the behaviors of humans, this script would need to consider the timings at which humans tend to interact with computers, whether or not a human would actually run that sequence of commands with the particular arguments for any reason, how frequently such commands tend to be run, etc. This is particularly problematic as it means it is challenging to realistically assess the quality of the vast majority of data used to train and evaluate anomaly-based IDSs.

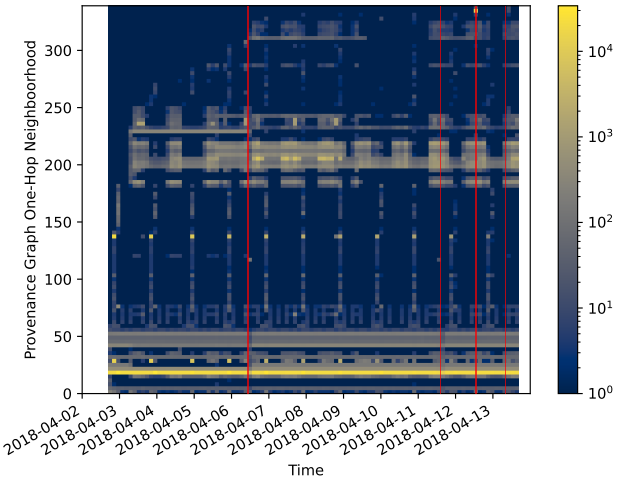

Figure 1. 2D Histogram of one-hop provenance neighborhoods vs. time for the DARPA TC E3 CADETS dataset. Each dot at a set y-value corresponds to one occurence of a particular provenance neighborhood. Note the clearly periodic vertical stacks, which indicate repeated groups of highly temporally-correlated behaviors. The horizontally dense behaviors are due to long-running daemons or services periodically repeating the same behavior (e.g., polling).

To our knowledge, no work analyzes the behaviors in benign logs of host IDS datasets. To address this, we present a method to visualize the behaviors contained in a log dataset by leveraging data provenance. Our visualization of many datasets show strong evidence of a workload generator being used to periodically simulate the same behaviors. Therefore, we also present a method to algorithmically discover such periodicity uncovered by our dataset visualization.

2. Methodology

Visualizing Behaviors in a Dataset. We leverage provenance graphs to structurally represent the relationships between system objects found in an audit log. Provenance

graphs are directed graphs which encode the flow of causality in system events; for example, when a process p reads from a file f, then information in f may affect the behavior of p. We represent this in a provenance graph by having nodes for system objects (in this case, p and f), with an edge from f to p indicating f may affect the behavior of p.

Directly visualizing a provenance graph is an unfeasible way to understand a log dataset due to the sheer size of these graphs. Typical system logs are very large, generating hundreds of gigabytes per day containing millions of events [5]. The provenance graphs built from these logs are thus also very large, easily also containing millions of nodes and edges.

To reduce the information found in a provenance graph to a more human-digestible format focusing specifically on the number of unique behaviors contained in the log, we instead focus on one-hop provenance neighborhoods. For each node in a provenance graph with at least one outgoing edge, we can construct a neighborhood, or subgraph, rooted at that node containing each outgoing edge from that node and edge's other node. We choose not to include leaf nodes with no outgoing edges to minimize visual noise, as these nodes will already be included as leaves of at least one other neighborhood. We place each one-hop neighborhood into an equivalence class based on the neighborhood's structure (i.e., number of children) and the names of each object in the neighborhood (i.e., the file path for files, the path of the running program for processes, and the address for sockets). If a particular equivalence class occurs less than some threshold number of times (we choose five by default), then we instead mark these neighborhoods as "outliers" to reduce the noise caused by extremely infrequent behaviors.

We can directly plot the neighborhood equivalence class versus time for every node in the provenance graph to visualize when specific behaviors occur. However, this loses frequency information if many instances of one behavior occur in a short timespan, as we cannot distinguish one dot from many dots at the same point. Therefore, we instead plot a 2D histogram of the neighborhoods vs. time to additionally show the corresponding frequency information.

Figure 1 is an example neighborhood plot for the DARPA Transparent Computing Engagement 3 (DARPA TC E3) CADETS dataset. We see strong evidence of a workload generator being used to generate benign data for this dataset, as we mostly see either long-running services or behaviors that are highly temporally correlated to other behaviors. While long-running services are natural to any system, only having highly temporally-correlated behaviors is evidence that some reptitive script is being fired on a timer to generate log data.

Analyzing Periodicity in the Neighborhood Plot. Many datasets, such as E3 CADETS shown in Figure 1, show clear evidence of a workload generator when viewed by a human. We next show how to algorithmically demonstrate this evidence by finding periodicity in the neighborhood plot. We use 2D autocorrelation to discover the periodicity of a dataset's behaviors [4]. Autocorrelation correlates a function with a shifted version of itself; in this case, we correlate our histograms with shifts in the time axis. Because 2D autocorrelations output a vector of probabilities per lag, we take the average probability at each lag to condense this vector into a single scalar.

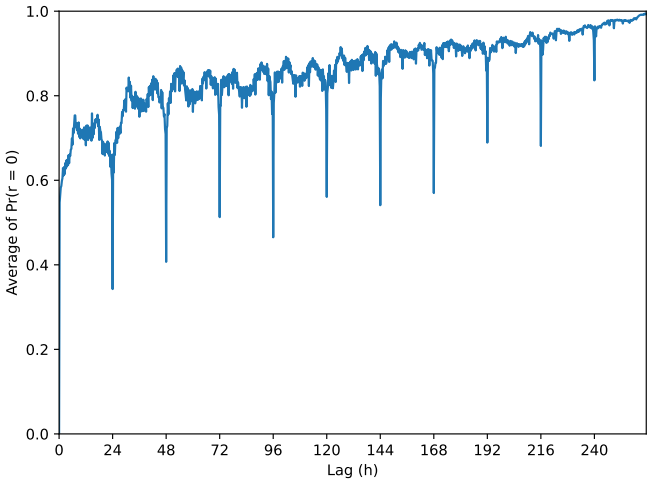

Figure 2. Average probability that the E3 CADETS neighborhood plot is not periodic vs time. The spike at 24 hours corresponds to the period we can visually confirm in Figure 1.

References

[1] Cybersecurity & Infrastructure Security Agency. *DarkSide Ransomware: Best Practices for Preventing Business Disruption from Ransomware Attacks*. 2021. URL: https://www.cisa.gov/news-events/cybersecurity-advisories/aa21-131a.

[2] The MITRE Corporation. *CVE-2021-44228*. 2022. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-44228.

[3] S. Forrest et al. "A sense of self for Unix processes". In: *Proceedings 1996 IEEE Symposium on Security and Privacy*. 1996, pp. 120–128. DOI: 10.1109/SECPRI.1996.502675.

[4] Hsin-Chih Lin, Ling-Ling Wang, and Shi-Nine Yang. "Extracting periodicity of a regular texture based on autocorrelation functions". In: *Pattern recognition letters* 18.5 (1997), pp. 433–443.

[5] Shiqing Ma et al. "Kernel-Supported Cost-Effective Audit Logging for Causality Tracking". In: *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. Boston, MA: USENIX Association, July 2018, pp. 241–254. ISBN: ISBN 978-1-939133-01-4. URL: https://www.usenix.org/conference/atc18/presentation/ma-shiqing.

[6] Dimensional Research. *2020 State of SecOps and Automation: A Survey of IT Security Professionals*. 2020. URL: https://assets-www.sumologic.com/resources/brief/2020_State_of_SecOps_and_Automation.pdf.