ON THE FORENSIC VALIDITY OF APPROXIMATED AUDIT LOGS

BY

NOOR SULTAN MICHAEL

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Adviser:

Assistant Professor Adam Bates

ABSTRACT

Auditing is an increasingly essential tool for the defense of computing systems, but the unwieldy nature of log data imposes tremendous burdens on administrators and analysts. To address this issue, a variety of techniques have been proposed for *approximating* the contents of raw audit logs, facilitating efficient storage and analysis. However, the security value of these approximated logs is difficult to measure – relative to the original log, it is unclear if these techniques retain the forensic evidence needed to effectively investigate threats. Unfortunately, prior work has only been able to investigate this issue anecdotally, demonstrating sufficient evidence is retained for specific attack scenarios.

In this work, we address this gap in the literature through formalizing metrics for quantifying the *forensic validity* of an approximated audit log under differing threat models. In addition to providing quantifiable security arguments for prior work, we also identify a novel point in the approximation design space – that log events describing benign system activity can be aggressively approximated, while events that encode anomalous behavior should be preserved with lossless fidelity. We instantiate this notion of *Attack-Preserving* forensic validity in APPROX a new approximation technique that eliminates the redundancy of voluminous file I/O associated with benign process acitivities. We systematically evaluate APPROX alongside a corpus of exemplar approximation techniques from prior work. We demonstrate that, while APPROX enjoys comparable log reduction rates, it is able to retain 100% of attack-associated log events; in contrast, we make the surprising discovery that prior approaches for log approximation retain as little as 7.3% of forensic evidence under the Attack-Preserving metric. This work thus establishes trustworthy foundations for the design of the next generation of efficient auditing frameworks.

To my parents, for their love and support.

ACKNOWLEDGMENTS

I would like to thank my adviser, Assistant Professor Adam Bates, for his guidance, encouragement, and patience over the past two years. His willingness to provide me this research opportunity and the ability to oversee a significant engineering effort has allowed me to grow both as a research student and as a developer.

I would like to thank the members of the Secure & Transparent Systems (STS) Lab for teaching me the many facets of academic research. In particular, I thank Wajih Ul Hassan and Riccardo Paccagnella for imparting their experience and expertise upon me. I would also like to thank Jaron Mink, Sneha Gaur, Jason Liu, and Dawei Wang for the countless hours spent programming and debugging, and the fruitful discussions that came along the way.

Last but foremost, I would like to thank my father for laying the groundwork for my success and my mother for her unfaltering support.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Now more than ever, auditing is vital to the defense of computing systems. With alarming regularity [1, 2, 3, 4, 5], sophisticated threat actors are able to breach perimeter defenses and subsequently wreak havoc on organizational networks. Given our struggles keeping intruders *out* of systems, the onus shifts to quickly detecting and responding to threats in order to minimize their impact. Audit logs have proven invaluable to these tasks; today, 75% of cyber analysts report that logs are the most important resource when investigating threats [6]. Moreover, the importance of audit logs will only grow as state-of-the-art *causal analysis* techniques for detection [7, 8, 9, 10, 11], alert triage [12, 13], and investigation [14, 15, 16, 17] become widely available.

Unfortunately, the capabilities provided by system auditing come at a cost. Commodity audit frameworks are known to generate tremendous volumes of log data, upwards of a terabyte per machine in a single month [18]. Not only is storing and managing this data a burden, but the unwieldy nature of these logs also slows down time-sensitive investigation tasks during in-progress attacks. For example, Liu et al. observed that even a simple backtrace query to determine the root cause of an event may take days to return [19]. At present, the inefficiencies of system auditing seriously undermine its use in more effectively combatting combatting real-world threats.

In response to this issue, researchers have called for optimizing the contents of audit logs, based largely on the observation that most events described by the log are not strictly necessary when investigating threats. A variety of methods have been proposed for achieving this goal, ranging from the filtering of events that describe deleted system entities [20], do not connote a new information flow [21] or do not effect the conclusions reached by standard forensic queries [22], among many others [23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 18, 33, 34]. We refer to these methods as *approximation* techniques in this work. With many of these techniques reporting orders of magnitude reduction in log size, the goal of holistic auditing of large organizational networks seems within reach.

While these results are encouraging for the perspective of performance, it is much more difficult to quantify the *loss in utility* that arises from approximating the original log. Is an approximated log equally useful when investigating threats, and if not under what circumstances can we expect the approximation routine to introduce error? For example, Lee et al.'s pioneering LogGC system deletes "deadend" events describing entities that no longer exist on the system [20], but given the ephemeral nature of network sockets it is likely that LogGC may destroy log events describing data exfiltration tactics. Unfortunately, prior

work has offered only anecdotal arguments for the utility of approximated logs for threat investigation.

In this work, we conduct the first independent analysis of the utility of audit logs that have been subjected to approximation techniques. We argue that utility is a measure of the "forensic validity" of a log for investigating different kinds of threats. To enable measurement of forensic validity, we introduce and formalize three metrics for characterizing approximated logs: *Lossless*, *Causality-Preserving*, and *Attack-Preserving*. While the first two metrics are distilled from prior work, attack-preserving forensics is based on the novel observation that only events that exclusively describe attack behaviors are essential to retain, while events describing typical process activity can be aggressively approximated. We use these metrics to conduct a rigorous utility analysis of a set of exemplar approximation techniques, discovering in the process that prior work often filters attack-related events when approximating logs.

While optimizing logs for attack-preserving forensics is preferable for the efficiency of audit logs, it requires a method of delineating typical process activity from unexpected system events. To address this gap, we present APPROX [1], a regular expression (regex) learning approach to log approximation. APPROX targets the most space-intensive events found in audit logs, namely the file I/O activity which can account for up to 90% of log contents.[2] Once a regex for a given process has been learned, APPROX then matches and eliminates new log events that match the regex. Through the design of a carefully-constructed learning algorithm, we demonstrate that it is possible to generate a set of regexes that faithfully describe typical process activity while simultaneously avoiding the filtering of any attack-specific behaviors. We show that APPROX retains 100% utility under the attack-preserving model while exhibiting comparable performance to state-of-the-art audit reduction techniques.

The contributions of this work are as follows:

- *Forensic Validity Measurement.* To facilitate reasoning about the utility of approximated logs, we present a set of novel metrics that can be used to quantify the value of logs under different threat models. We conduct an independent analysis of an exemplar set of approximation techniques, discovering in the process that destruction of attack-relevant events is common. For instance, we discover that source dependence preservation [35] retains just 7.3% of attack information.

- *Attack-Preserving Approximation Techniques.* We present APPROX a log approximation method that is optimized for attack-preserving forensics. APPROX performs a bounded

[1] Attack-Preserving Provenance Reduction Over regeXes

[2] We observe in our evaluation datasets that 88.97% of events describe system calls associated with file I/O.

regex-learning routine over process executions to learn their file I/O behaviors. While aggressively filtering events related to known file behaviors, APPROX retains a complete record of process-to-process and process-to-network dependencies, thus facilitating the causal analysis techniques.

- *Evaluation and Attack Engagements.* We evaluate the performance of APPROX and subject it to a series of attack scenarios through which we measure forensic validity. These engagements confirm that APPROX fully satisfies the attack-preserving forensic metrics, in spite of offering comparable reduction rates to prior work. We plan to open source APPROX following publication.

CHAPTER 2: BACKGROUND

Logging is critical to defending sysyems, facilitating intrusion detection and post-mortem forensics of attacks. Audit logs can be generated at different software layers; for example, Windows Event Logs is an application-level logging framework for Active Directory environments [36], while Event Tracing for Windows [37] and Linux Audit [38] are kernel-level logging frameworks that primarily trace system call information. While all logs can be of potential use during threat investigations, low-level audit frameworks are especially useful in threat hunting because they can be used to reliable trace dependencies between applications.

The contents of an audit log can be thought of as a sequence of temporally-ordered event tuples (i.e., $< subject, object, access, timestamp >$), which can be further parsed into a causal dependency (i.e., *provenance*) graph by incrementally linking entities associated with each event tuple according to the access type (see, e.g., [39, 40, 41, 42]). Specifically, a provenance graph can be defined as $G = (V, E)$. Each vertex $v \in V$ corresponds to a system object such as processes, files, and file-like objects (network sockets, virtual file system objects, etc.), while each $e \in E$ encodes a dependence relationship between those objects and roughly orresponds to a single log event. Typically, these edges direct in the opposite direction of information-flow, denoting a provenance (historical) relation on the events that occurred. Thus, a traversing the graph backwards or forwards respectively allows analysts to peer into the history or future of a given event.

Definition 2.1. Backward Trace: A backward trace of edge e is the subgraph of G reachable from e (or equivalently, the destination vertex of e).

Definition 2.2. Forward Trace: A forward trace of edge e is the subgraph of G reachable from e in the reverse graph of G (or equivalently, the souce vertex of e).

A backward trace enables an analyst to identify the *root cause(s)* of a particular event, e.g., the point of entry of an intruder into the system. Conversely, a forward trace identifies the impact of a particular event, with a forward trace from the root cause reporting all actions taken by the attacker.

2.1 AUDIT LOG APPROXIMATION

Although audit frameworks are extremely useful to threat investigation, the main limitation of is the sheer volume of logs generated. Log overheads vary depending upon on the load of the machine, but have been reported to be anywhere between 3 GB [20] and 33 GB

[18] per day for web servers and around 1 GB per day for workstations [20]. As a result storing and anlyzing these logs is often infeasible in practice – log data is often purged just a few days after its creation [43], and when it is retained for longer periods simple trace queries may take days to return [19].

To combat the limitations of pervasive system auditing, many techniques for *log approximation* have been proposed. These approximation methods analyze the structure and semantics of the provenance graph to identify components (i.e., log events) that are unlikely to be of use to an analyst and can therefore be removed. While a variety of approaches to log approximation have been proposed based on filtering policies [27, 30], taint analysis [28, 34], templatization [23, 32, 33], or simple compression [24, 25], we describe at length three influential exemplar approaches below:

- **Garbage Collection (GC).** First proposed by Lee et al. in their LogGC system [44], garbage collection is based on the observation that subgraphs that exclusively describe *dead* system entities do not affect the present state of the system and can therefore be removed. Consider a process that generates a temporary file then deletes it. If no other process accesses that temporary file, all log events associated with that file can be removed from the graph. A visualization of garbage collection is given in Figure **??**. Garbage collection has since been incorporated into a number of log analysis systems, e.g., [26, 28, 18].

- **Causality-Preserving Reduction (CPR).** Originally introduced by Xu et al. [45], CPR observes that many log events are redundant because they do not denote a new information flow in the provenance graph. Consider a process that writes to a file twice. If the process did not read from any other object between the two writes, we can remove the log event describing the second write because the process state did not change between writes. Note that, in actuality, the data buffer may have been completely different in each write event; however, because audit logs do not include data buffers, when interpreting provenance graphs we must always conservatively assume that *all* process state is transferred during each information flow event. Thus, the second write is completely redundent in this example. A visualization of CPR is given in Figure **??**. CPR has been adopted or extended by subsequent log analysis systems, including [32, 19, 12, 22].

- **Dependence-Preserving Reduction (DPR)** Building on the CPR concept, Hossain et al. go a step further by proposing that preserving causality is unnecessary so long as the provenance graph returns the correct system entities to forward and backward trace queries [35]. Consider a vertex for which their are two causal paths back to its root cause; this represents a potential redundancy and one of the two edges may be deleted, provided

that the edge is not necessary to preserve dependency for some other node. Hossain et al. introduce two variants of DPR, Source Dependency-Preserving Reduction (S-DPR) in which only backward trace reachablity is preserved, and Full Dependency-Preserving Reduction (F-DPR) in which both backward and forward trace reachability are preserved. A visualization of S-DPR is given in Figure **??**. As a more recent proposal, we are not aware of subsequent work that has incorporated DPR, although DPR is significant in that it boasts among the highest reduction rates in the literature.

Limitations of Prior Work. While the performance characteristics of these approaches were effectively evaluated in prior work (i.e., storage overheads), the security characteristics of logs that were approximated using these techniques has proven more difficult to quantify. When approximating the log, problems of graph reachability and interpretability may arise. Further, if events are merged or deleted, then interpreting the graph becomes more difficult. Worse yet, key details of the attack behaviors may be lost in ways that were unanticipated by the designers of the approximation technique.

Unfortunately, without exception [44, 45, 35, 46, 47], prior work was evaluated exclusively through attack scenario *case studies* in which a forensic analyst needs to answer a specific query during investigation. The issue with this approach is that it is anecdotal; it may be that analysts need to answer a broader range of queries that were not considered in the case study, or that the semantics of the specific attack scenario did not adequately explore the forensic utility of the approximated log. Worse yet, under this evaluation technique the efficacy of approximation methods is reduced to a binary 'yes' or 'no' depending on whether or not the analyst is able to achieve a hand-selected forensic goal. In the following section, we present a set of nuanced and descriptive methods for characterizing the security of an approximation technique. These metrics provide a continuous (i.e., non-binary) value for characterizing forensic validity, and further describe the value of an approximated log even for unanticipated queries.

CHAPTER 3: APPROXIMATION METRICS

To better characterize the security utility of log approximation techniques, we propose a set of three complementary *forensic validity metrics* that can be used to evaluate an approximated log. Each metric encodes utility under a different threat model that an analyst may encounter during the course of an investigation. The key insight behind these metrics is that, rather than anecdotally demonstrating value in a specific attack behavior, utility should be measured as a property of the approximated graph or log. We propose three such metrics: *Lossless*, *Causality-Preserving*, and *Attack-Preserving*.

3.1 LOSSLESS FORENSICS

It is important to recognize that *any* log approximation necessarily results in a loss of utility. We begin by proposing a strawman validity metric that allows us to measure this baseline loss by comparing an approximated log to the ideal notion of preserving all log entries associated with an attack. Lossless forensics retains sufficient information for any threat model, but its value becomes most clear when considering system-layer side channels. Consider a malicious process attempting to exfiltrate data over a timing side channel that is being observed by a colluding process (e.g., [48, 49]). Such an attack will only be identifiable if the timing and frequency of system calls is retained in the logs; however, this information is typically seen as redundandt and thus discarded.

Definition 3.1. Lossless Forensics: Given $G = (V, E)$ and approximation $G' = (V', E')$ in which $V' \subseteq V$ and $E' \subseteq E$, a lossless log it must be true that $E' = E$. Distance from losslessness can be measured as a continuous variable using the formula $1 - \frac{|E' \cap E|}{|E|}$.

In other words, losslessness can be measured by te fraction of edges in E that are missing from E'. Note that it is not necessary to directly test the completeness of V, as in practice each edge is a log event and the associated system entities in V are extracted from these tuples.

3.2 CAUSALITY-PRESERVING FORENSICS

As information flow (IF) is a well-studied subject in computer security, we believe it to be an excellent standard benchmark for log validity and thus encode Xu et al.'s Causality-Preserving Reduction technique as one of our metrics. Note that this metric describes explicit

information flows in the system, hence, implicit information flows like the one considered by the lossless metric are not captured in the approximated log. In a log that satisfies causality-preserving forensics, all events that encode new causal relationships are retained, whereas edges that are causally redundant are discarded. We define the metric as follows.

Definition 3.2. Causality-Preserving Forensics: Information flows from G are preserved in the approximated graph G'. An information flow is defined by the existence of a path between two edges in G. The following describes two situations where two edges describe the same information flow.

- Two read edges e_1, e_2 describe the same information flow if they have the same endpoints process p and file f, and no write to f or read from p to a file $f' \neq f$ occurred between e_1 and e_2.
- Two write edges e_1, e_2 describe the same information flow if they have the same endpoints process p and file f, read from p to a file occurred between e_1 and e_2.

G' maintains causality-preserving forensics if for all $u, v \in E'$, there is a causal relationship between u and v if there is a causal relationship between u and v in G. The distance from causality preservation can be measured by the formula $1 - \frac{|E' \cap E|}{|E|}$.

As a concrete example, consider a provenance graph consisting of a file node and two process nodes (A and B). Process A reads the file twice and process B writes to the file. If both reads sequentially occur before the write, then both reads return the same data, and thus the same information from the file. Therefore, the causal relationship between the file and process A can be encoded by only preserving one of the two read events. However, if the write were to occur between the two reads, then the information returned by the two reads may be different. Therefore, the information flow between process B, the file, and process A may possibly differ from the information flow of the first read. Therefore, we must preserve both read events. More precisely, we reduce a read event when there are no writes to the corresponding file since the last read. Similarly, we are able to reduce a stream of consecutive write events to a single write event if no read is interspliced between them. This reduction is expressed in Algorithm 3.

This model assumes that all interesting attack behavior is present in an log that preserves information flows. Our threat model may then be attacks which require all information transmitted is clearly present along an information flow. As mentioned in the previous section, in using a side-channel to transmit information, a causal path not present in the provenance graph and thus, would not be identified under this threat model. Fortunately,

most attack behavior that is present in the log follows some causal path from the initial intrusion to further actions on the host.

3.3 ATTACK-PRESERVING FORENSICS

Our final reduction metric will specifically consider attack events, similar to lossless forensics. The goal of this reduction metric is to preserve logs that uniquely corresponding to the attack, insofar as we preserve information flows. In other words, it is identical to the causality-preserving metric, with the additional relaxation that we must only preserve events associated with the attack that are not present in the benign graph. In terms of the provenance graph, this will be the intersection of the causality-preserving graph and the events of the attack graph that are unique to the attack.

Definition 3.3. Attack-Preserving Forensics: Given provenance graph G and the approximated graph G', attack-preserving forensics is satisfied if causality-preserving forensics is satisfied and E' is a subset of the difference between E and the set of all possible benign events B. Formally, for all $u, v \in E'$, there is a causal relationship between u and v if there is a causal relationship between u and v in G, and $E' \subseteq E \setminus B$. The set of all possible benign events B can be thought of as complete coverage over all behaviors present in an application. The distance from causality preservation can be measured by the formula $1 - \frac{|E' \cap E|}{|E|}$.

The goal of this metric is to ensure that forensic analysts have enough information to triage an attack. This is useful for reduction techniques because to perform well against this metric, they no longer have to distinguish whether seemingly identical behavior in the log is malicious or benign.

This metric makes an assumption that all interesting attack behavior is outside the realm of benign program behavior. Therefore, our threat model covers attacks that gain access to a system, such as remote code execution vulnerabilities. It considers attacks that conduct behavior outside of normal program behavior, which is necessary to gain access to a system. Once access, in particular root access, is gained, this assumption breaks down, because if programs are fundamentally modified, their malicious behavior may be disguised as benign behavior. The following attack exemplifies this limitation of attack semantics-preserving reduction. Consider a program that loads multiple shared libraries at startup. If a host is already completely compromised, one of these shared libraries can be replaced by a malicious library (superuser permissions are necessary to overwrite a system library). Once this program runs, then libraries will seem to load as usual, even though these libraries contain

malicious behavior. When executed, however, so long as that behavior differs from benign program behavior, it will be preserved within Attack-Preserving reduction.

CHAPTER 4: DESIGN

4.1 MOTIVATION

Our goal is to present a reduction strategy that performs well against the attack-semantics preserving approximation metric. We want to reduce benign behavior, even if it is part of higher-level malicious behavior, but faithfully preserve uniquely malicious behavior. Note that reducing benign behavior will not severely impact a forensic analyst's forward and backward traces in a provenance graph, because the causal links for the benign behavior are still there, but they are reduce across all instances thereof. An example of such a system is Winnower, which reduces common events across multiple executions of a program [46]. Our reduction algorithm first reduces events that preserve causal dependencies in the provenance graph. It then generalizes the processes behavior by learning patterns common in its behavior, and reducing further.

The overarching goal of our approximation technique is to learn regular expressions on benign program behavior, allowing aggressive reduction thereof. We show that these patterns do not generalize to uniquely malicious program behavior, and therefore satisfy the requirements for attack-semantics preserving approximation.

4.2 REDUCTION ALGORITHM

Our reduction algorithm begins by identifying, for each process, which files it has interacted with. Our provenance graph encodes this information by edges to (read event) and from (write event) a process node. From this list of files, we will generate groups of files with similar filenames. Replacing each group of files with a single placeholder in the provenance graph allows us to reduce the graph complexity and hence filter redundant log entries.

4.2.1 Regular Expression Learning

Our goal is to distribute a list of filenames associated with a process into groups of similar files. For a particular filename (eg. /usr/bin/ls) we distinguish the path (eg. /usr/bin/) and the name itself (eg. ls).

We define the *distance* between two names as the Levenshtein edit distance. Corresponding to this edit distance is an optimal alignment, which will be useful later in constructing a regular expression. For the distance between two paths, we treat each directory name as a

Algorithm 4.1: Generate Groups

Data: List of filenames $f_1, \ldots, f_n$
Result: Groups of filenames G

```
Empty list G;
while not added file f_i do
    Set G_i ← {f_i};
    for filenames f_j not added to a group do
        if SIMILARITY(PATH(f_i), PATH(f_j)) ≤ path_threshold and
         SIMILARITY(NAME(f_i), NAME(f_j)) ≤ name_threshold then
            Add f_j to G_i;
    Append G_i to G;
Return G;
```

Algorithm 4.2: Generate Regular Expressions

Data: Groups of filenames $G = [G_1, \ldots, G_n]$
Result: Regular Expressions $R = [r_1, \ldots, r_n]$

```
Empty list R;
for every G_i ∈ S do
    Set r_i ← G_i[1];
    for filenames f ∈ G_i[2, ..., m] do
        Find alignment a between r_i and f;
        Replace modifications in a with wildcards;
        Set r_i to a;
    Append r_i to R;
Return R;
```

token. We only consider the distance when both paths are of equal depth, because differences in depth contain semantically relevant information. In other words, the path distance is the number of differing directory names. The similarity between two names or paths x_1, x_2, where $max\_len = \text{MAX}(\text{LEN}(x_1), \text{LEN}(x_2))$, is then $(max\_len - \text{DIST}(x_1, x_2))/max\_len$.

We compute the path and name similarity between all pairs of filenames. We group them by those with a similarity below a certain threshold. We compute these thresholds empirically, by taking a subset of logs and finding the threshold at which the sum of similarities reaches 70% of the total sum.

The algorithm above groups the files into sets, where each file is at most a certain distance from another file in its set. We return a list of such groups, each of which corresponds to a regex.

To compute the regex corresponding to a group of files, we compute the path regex and name regex individually. To compute a regex from two strings, we find the edit distance

Algorithm 4.3: Log Reduction

```
Data: Log and Provenance Graph
Result: Reduced Log
for every process p do
    Compute list of groups G;
    Compute list of regexes R;
    for every group G_i ∈ G do
        for all reads e to file f ∈ G_i (in order) do
            if since the last read, there was a write to f or a read from p to a file ≠ f
             then
                Keep e and overwrite f with r_i;
            else
                Delete e;
        for all writes e to file f ∈ G_i (in order) do
            if since the last read, there was a read from p to a file ≠ f then
                Keep e and overwrite f with r_i;
            else
                Delete e;
Return log;
```

alignment, and for every location where the tokens do not match, we replace it with a placeholder. We reduce this binary operation across the list of filenames. We coalesce placeholders and replace each with a token matching zero or more occurrences of a wildcard. We then concatenate the path and name regexes to generate a regex matching all files in the group. If there is only one element in the group, we return that filename as the corresponding regex.

This algorithm uses a binary regex generation function, taking as input the current progress and the next filename. It reduces this function across all filenames in a group. If the current regex matches the next filename, it will remain unchanged.

4.2.2 Log Reduction

For every process, we generate a list of filenames corresponding to file accesses initiated by the process. These filenames are grouped and their corresponding regular expressions are generated. For every group of filenames, we reduce the log entries between the process and these files preserving information flow. We do not reduce filenames corresponding to regular expressions with a length below a certain threshold, in our case, 10 characters, since they can overgeneralize. In effect, we are treating this group of filenames as one large file. For the log entries that have not been removed, we overwrite the filename with the regular

expression corresponding to the group.

This reduction algorithm acts on every log entry corresponding to a file access. It either keeps or deletes a log entry based on the information-flow preservation criteria outlined in the background.

CHAPTER 5: IMPLEMENTATION

We implemented a log analysis tool that parses Linux Audit (`audit`) logs and CDM provenance graphs, the DARPA Engagement data format. Out tool generated a provenance graph in memory using the `SNAP` graph library [50]. Our provenance graph representation has nodes corresponding to processes, files, and other file-like objects (e.g. VFS, network sockets). Edges correspond to individual log entries. Our entire tool is implemented in 4000 lines of C++ code (calculated with `cloc` [51]). Our reduction filter APPROX including the regular expression generation, is implemented in 1000 lines of code. We also implemented Causality-Preserving Reduction, introduced in Xu et. al [45], LogGC, introduced in Lee et. al [44], and Full and Source Dependence Preserving Reduction, introduced by Hossein et. al [35]. Our implementation of LogGC implements Basic GC, as defined in the original paper. Our implementations of the other reduction techniques follow their respective definitions in the original papers.

We partition execution into epochs of time, each of which is 5 minutes. This is necessary because our technique constructs a provenance graph online, and then filters depending on the graph structure. Introducing epochs limits graph size and allows approximately real-time telemetry for security monitoring services. Thus, anomalous behavior can be identified quickly and security analysts can respond appropriately. We acknowledge that redundancies between epochs will not be identified, however we observe that by using such epoch sizes, we achieve reduction figures similar to those in the original papers.

CHAPTER 6: EVALUATION

6.1 DATASETS

We leverage the DARPA Transparent Computing Program dataset for out evaluation [52]. This dataset was collected during an APT simulation exercise (Engagement #3) in April 2018. It contains log data from a series of target hosts, along with ground truth information about the attacks launched. Previous work has examined these logs in detail, and it has proven to be a good source for authentic examples of adversarial behavior [53] [54]. We select the traces corresponding to two hosts from the DARPA Engagement, Trace and Theia, because they are both Linux hosts, compatible with out implementation.

We select 6 attack scenarios leveraged in prior work [citation] to evaluate the efficacy of log reduction systems. The `unrealircd` [55], `vsftpd` [56], and `webmin` [57] exploits all leverage input vulnerabilities to achieve remote code execution (RCE). A payload is then executed, which launches a reverse shell back to the attacker machine which executes basic enumeration commands, such as `ifconfig`. The Wordpress vulnerability [58] and Webshell are exploits that execute a payload through a web server that calls back to the attacker machine as before. The Firefox vulnerability is an exploit on Firefox 54.0.1 that gains execution through a malicious ad server. This vulnerability was exploited as part of DARPA TC Engagement #3 [52]. We use the subset of logs associated with this particular exploit during this evaluation.

We compared the reduction performance of our approximation technique re-implementations against the figures in the original papers. Log GC had the broadest range of reduction, from full reduction to no reduction whatsoever. The figures above correspond to the web server benchmarks present in the paper. The reduction figures for dependence preserving reduction are quite staggering, and for certain log sources, such as Theia, it is unclear whether this level of reduction is meaningful. For all the approaches, our re-implementations reduced the logs within reasonable bounds.

6.2 PERFORMANCE EVALUATION

To evaluate the performance of our reduction technique at scale, we concatenate all logs for a particular host and run our reduction algorithm. The end of an epoch corresponds to either the end of a log file or when 5 minutes elapses.

We observe a 2.87X reduction on the Theia dataset and a 1.72X reduction on the Trace

Approximation Technique	Originally Reported Reduction	Observed Reduction
CPR [45]	1.3-3.4X (30% - 79%)	1.3X (78%)
GC [44]	0-77.0X (1% - 100%)	1.6X (63%)
F-DPR [35]	4.5-91.5X (1% - 22%)	6.6X (15%)
S-DPR [35]	4.5-122.5X (1% - 22%)	11.2X (9%)

Table 6.1: Comparison of log reduction between figures cited in the original papers and our re-implementations. The re-implementation figures were calculated on the Theia dataset. Our re-implementations reduce logs within the ranges observed by the original implementations. The figures are cited as the log reduction factor and as the percentage of the log that was retained.

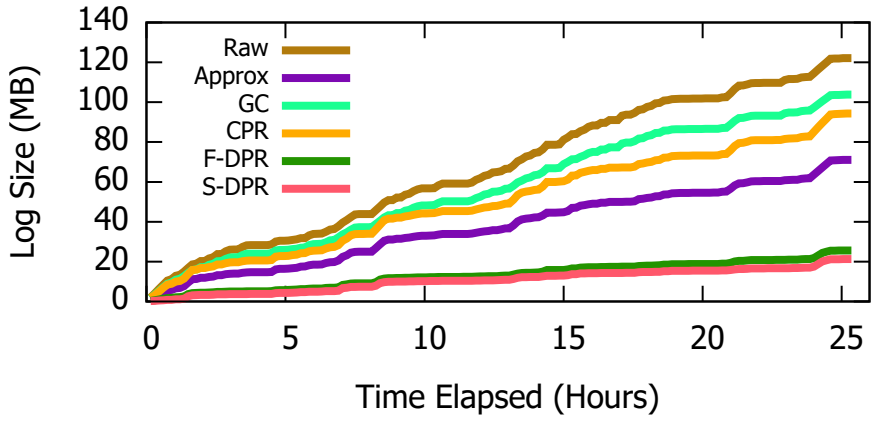

Figure 6.1: Cumulative Log Size Stored after different reduction techniques on the Trace datasets. APPROX outperforms all approaches besides Full and Source Dependence.

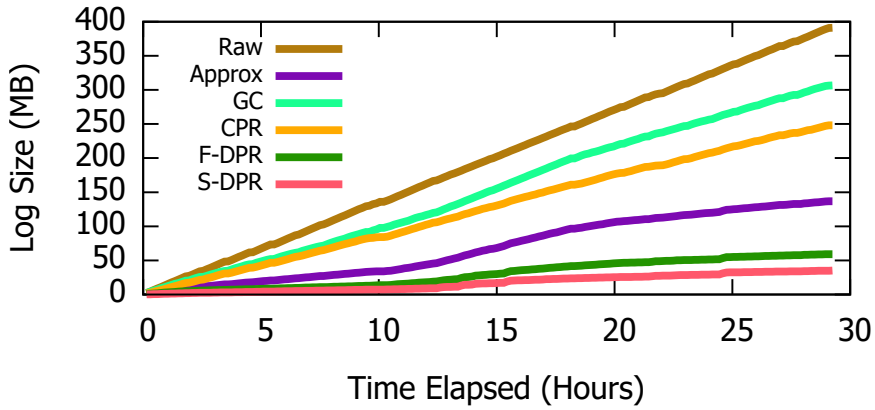

Figure 6.2: Cumulative Log Size Stored after different reduction techniques on the Theia datasets. APPROX outperforms all approaches besides Full and Source Dependence.

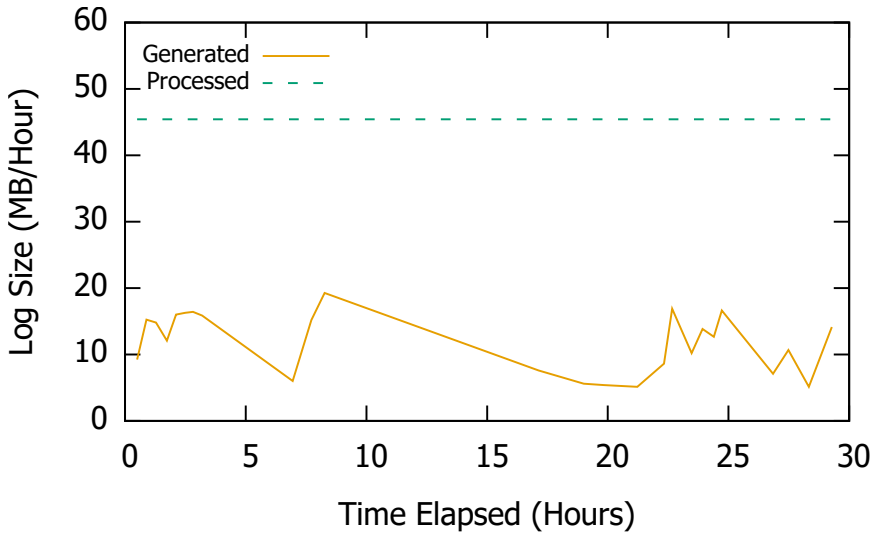

Figure 6.3: Log Processing speed relative to the generation of logs in the Theia dataset, which we replayed in real-time.

dataset. The Theia logs are few and large, corresponding to longer duration captures on the host. Thus, we see a more significant reduction over time, because our reduction algorithm is able to leverage a reasonably sized provenance graph to find patterns. The Trace dataset, on the other hand, contains many small log traces. This renders reduction techniques less effective, because they are unable to significantly reduce a provenance graph that is already small. They cannot find enough redundancy to collapse provenance graph structures.

We see that Full Dependence Preservation and Source Dependence Preservation, introduced by Hossein et. al [35], reduce much further than any other approaches discussed. However, in the next section we will see that these approaches do not preserve information critical to understanding attack behaviors, even if they preserve connectivity in the provenance graph. In comparison to GC, our approach reduces logs more consistently and without the need for execution partitioning. Relative to Causality-Preserving Reduction, which achieves an average reduction rate of 2X, our approach is able to reduce further without sacrificing unique attack information, provided the log trace has sufficient duration.

In Figure 3, we compare the amount of logs that can be processed by APPROX at maximum load, versus the amount of logs generated from the Theia dataset over time. We see that our system's ability to process logs far outpaces the amount of logs generated, and can thus scale to much larger deployments.

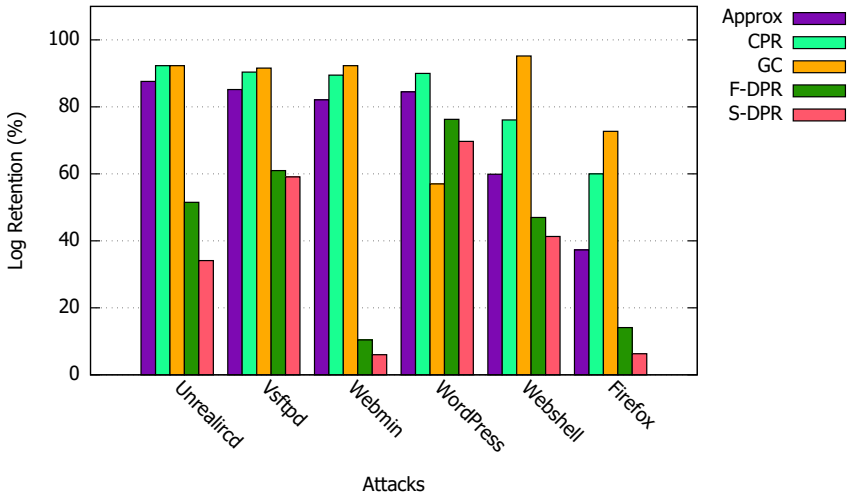

Figure 6.4: Log Retention percentages for different reduction techniques, relative to the Lossless Forensics approximation metric.

6.3 ATTACK EVALUATION

We evaluate the performance of our reduction technique against a variety of real-world program exploits. Our goal is to recover the causal information necessary to have a full picture of the attack behavior. We will compare the performance of various log reduction techniques against the approximation metrics described in Section 3.

To evaluate our attack scenarios, we collected Linux audit logs during the attack, and processed them with our reduction system. Since the logs contain solely attack behavior, the lossless forensics metric is with respect to the entire log, which we would ideally retain. The causality-preserving metric compared a technique's log reduction relative to the performance of the CPR filter originally proposed by Xu et al. [45]. To evaluate the attack-preserving reduction, we tagged events unique to the attack and evaluated how well they were preserved relative to the causality-preserving metric.

In Figure 3, we see that APPROX performs poorly against the lossless and causality-preserving metrics, meaning that it reduces further than reduction techniques limited by those metrics. However, it does preserve all attack-relevant events in an information-flow preserving manner. Thus, all causal relationships between system events during the attack are preserved. This also demonstrates that Causality-Preserving Reduction, as defined in Xu et al. [45], also preserves all attack-relevant events. This is at the expense of preserving more of the log, as seen in the performance evaluation section. GC retains a large portion of the log, but it does not entirely preserve all of the attack-relevant events. In particular, causality chains in the graph that solely interact with the network, critical to understanding

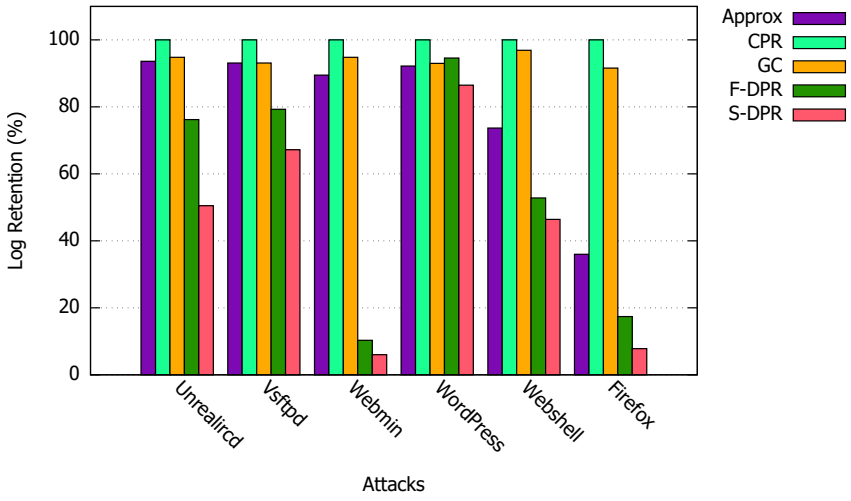

Figure 6.5: Log Retention percentages for different reduction techniques, relative to the Causality-Preserving Forensics approximation metric.

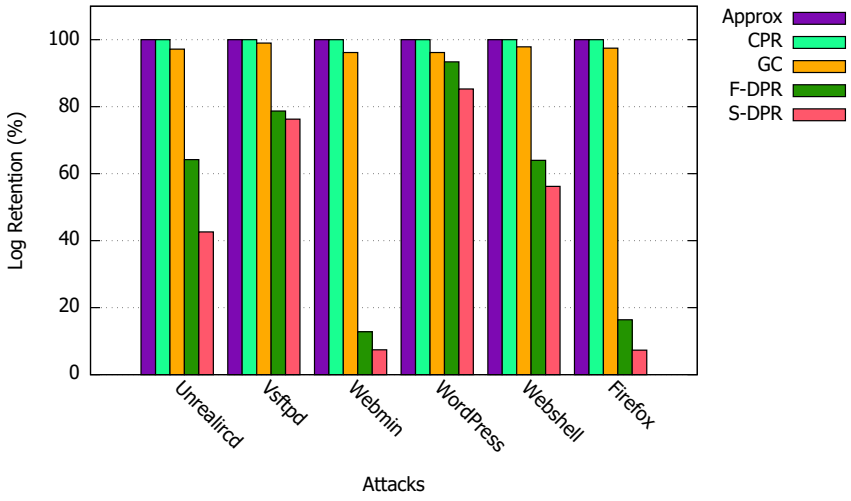

Figure 6.6: Log Retention percentages for different reduction techniques, relative to the Attack-Preserving Forensics approximation metric. Ideally, a reduction approach preserves semantics well, while retaining as few logs as possible.

the attack, are removed because GC treats sockets as a dead-end. GC reaps events that result in a dead-end, resulting in undesired behavior. Full and Source Dependence Preservation algorithms reduce logs considerably, reducing much of the attack-relevant behavior as well. Even by maintaining connectivity in the graph, they remove events that are necessary to understanding the behavior of the attacker on the system. Often times in an investigation, it is not sufficient to know which system entities are causally related; one must know what behavioral pattern the attack follows to understand how it occurred.

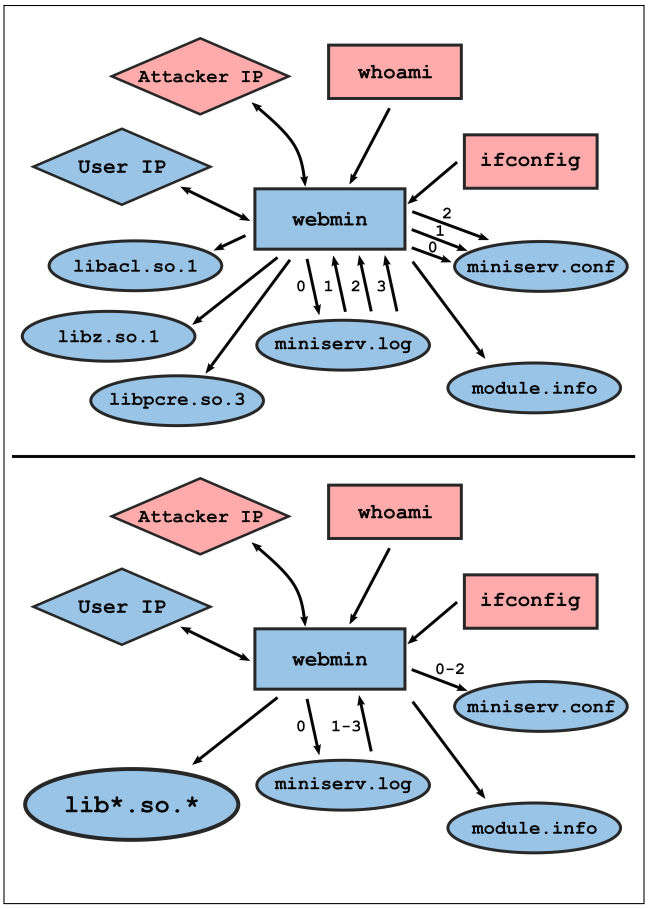

Figure 6.7: Provenance Subgraph of the Webmin exploit. The subgraph before reduction is above, and the subgraph after reduction is below. Numbered events are reduced through Causality-Preserving Reduction, and the shared object files are reduced through our novel regular expression learning approach.

6.4 CASE STUDY

We will now examine the Webmin exploit above in more detail. Webmin is a web-based configuration tool for Unix systems, thus it is a prime target for attackers as it can be leveraged for lateral movement on the network. The exploit allows for unauthenticated remote code execution when the web server is configured for "users with an expired password to enter a new one". With the appropriate payload, a reverse shell can be spawned and post-exploitation tools (eg. `LinEnum.sh` [59]) can be downloaded and run on the server machine. We chose to run commands manually, namely `whoami` and `ifconfig`, to demonstrate that we have remote access.

The provenance graphs in Figure 5 correspond to the raw provenance graph on top, and the reduced provenance graph on the bottom. They are subgraphs of the original provenance

graphs, the raw one containing 13026 edges and the reduced one containing 11013 edges. The numbers correspond to timestamps. Directory names are omitted due to space constraints. The first form of reduction we observe is through a regular expression (`lib*.so.*`) that learns to reduce the shared object libraries shown above. Note that the labels in Figure 5 omit directory information that is crucial in ensuring that our reduction technique does not reduce shared object files dropped by the attacker in another directory. The other reductions that occur on sequential reads and writes are a consequence of a causality-preserving reduction that takes place after regular expressions collapse nodes.

CHAPTER 7: DISCUSSION

Optimality of APPROX. We do not argue that APPROX is the ideal instantiation of attack-preserving forensics. In this work, we only target one major subsystem of system activity, namely file I/O. We elect to focus on file I/O in this work not only because it commonly accounts for a large percentage of log volume, but also because this activity can safely be filtered without disrupting the causality of more complex attack behaviors such as lateral movement or data exfiltration. Approximating other system entities, e.g., remote IP addresses, is much more perilous, as even a common remote connection may describe an important attack behavior during lateral movement. It is interesting to consider how the notion of attack-preserving forensics could be extended in future work.

Limitations of Evaluation. We set out to define metrics for evaluating the security of approximated logs. We make significant progress towards this goal in this work, but one notable limitation that we share with prior work is that we are still limited in the corpus of attack behaviors that we evaluate against. It would be useful to have a standardized battery of attack behaviors that coincide with our metrics; maintenance of such an attack corpus is extremely difficult, due to ever changing adversary techniques and tactics, and hence we leave it to future work [60]. However, using our techniques, authors can go from a binary "works/doesn't work" case study to a more honest and nuanced gradient of efficacy based on our metrics.

CHAPTER 8: RELATED WORK

This work is the first to formalize and quantify the forensic validity of audit logs that have been subjected to approximation techniques. We discuss past approximation techniques, which are most closely related to our own work, in Section 2.

The auditing literature is also experiencing a renaissance beyond the challenge of log reduction. One important consideration is log security; if audit logs can be manipulated by the attacker, they cannot be trusted in an investigation. A variety of cryptographic approaches enable tamper-evident audits of log contents (e.g., [61, 62, 63, 64, 65, 66, 67, 68, 69]). Research has also explored software-based solutions to securing log contents, demonstrating that reference monitor guarantees [70] were sufficient to assure log integrity [71, 72, 73, 74]. Recently, Karande et al. [75] and Paccagnella et al. [76] explore how cryptographic approaches to log integrity can be integrated into operating system auditing through trusted execution environments. Like all other work in the space, APPROX depends on the presence of mechanisms that can assure and attest to the integrity of audit logs.

Considerable attention has given to extracting high-level semantic insights from low-level system logs and graphs. A central issue with system logs is *dependency explosion*, a semantic gap problem in which long-lived processes (or data objects) appear to have a large number dependencies when viewed from the system layer. A variety of execution partitioning techniques have been proposed to partition opaque dependencies into small autonomous units of work [17, 26, 77, 28, 78, 79, 18, 12, 14]. We believe these techniques are interoperable with APPROX although they may not be necessary because APPROX will have removed a large percentage of false dependencies related to benign execution units. Prior work has also considered related semantic gap problems, including the reconciliation of system-level logs with application logs [15, 14] and the identification of high-level tasks [16, 80, 81]. These techniques should also be compatible with APPROX provided that the analyst is only interested in reconstructing attack-related sequences of events.

Increasingly, causal analysis techniques are being incorporated intrusion detection tasks. Manzoor et al. [9], Han et al.'s Unicorn [7], and Wang et al. [8] present anomaly detection algorithms based on the analysis of system dependency (i.e., provenance) graphs. Hassan et al. address the false alert problem common in commercial threat detection software using provenance-based alert triage [12]. Milajerdi et al. present a rule-based approached for detecting attacker tactics [10], similar to commercially-available Endpoint Detection & Response (EDR) software, but based on provenance graph structures instead of flat audit event sequences. Subsequently, Hassan et al. extend a commercial EDR tool with lightweight

provenance-based alert correlation [13]. Their approach to making their technique practical for large enterprise environments is to aggressively filter provenance graphs such that only queries about inter-alert dependency can be answered by the approximated log. In contrast to other approximation techniques discussed in this work, Hassan et al.'s approach is not intended for use in generic threat investigation scenarios where many forms of causal query must be supported. While we primarily consider threat investigation, the forensic validity analysis presented in this work can also be interpreted as an indicator of how log approximation techniques may assist or impair intrusion detection tasks.

CHAPTER 9: CONCLUSION

In this work, we present APPROX a log approximation technique that performs well against attack-preserving forensics, a metric we introduce that ensures preservation of events unique to an attack. APPROX leverages repetition in file I/O behavior to learn regular expressions that characterize common access patterns. This allows APPROX to aggressively reduce the number of file I/O events, which make up the majority of audit logs. We evaluated the performance of APPROX against a variety of attack scenarios, including the DARPA dataset, comparing its performance against the prior work. We observe that APPROX achieves comparable reduction rates while being able to retain 100% of attack-associated log events, in contrast to prior work which retains as little as 7.3%.

REFERENCES

[1] "Equifax Says Cyberattack May Have Affected 143 Million in the U.S." https://www.nytimes.com/2017/09/07/business/equifax-cyberattack.html.

[2] "Inside the Cyberattack That Shocked the US Government," https://www.wired.com/2016/10/inside-cyberattack-shocked-us-government/.

[3] "Target Missed Warnings in Epic Hack of Credit Card Data," https://bloom.bg/2KjElxM.

[4] "Information on the Capital One Cyber Incident," https://www.capitalone.com/facts2019/.

[5] "Clearview AI has billions of our photos. Its entire client list was just stolen," https://www.cnn.com/2020/02/26/tech/clearview-ai-hack/index.html.

[6] Carbon Black, "Global incident response threat report," https://www.carbonblack.com/global-incident-response-threat-report/november-2018/, November 2018, last accessed 04-20-2019.

[7] X. Han, T. Pasqueir, A. Bates, J. Mickens, and M. Seltzer, "Unicorn: Runtime Provenance-Based Detector for Advanced Persistent Threats," in *27th ISOC Network and Distributed System Security Symposium*, ser. NDSS'20, February 2020.

[8] Q. Wang, W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, J. Rhee, Z. Zhen, W. Cheng, C. A. Gunter, and H. chen, "You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis," in *27th ISOC Network and Distributed System Security Symposium*, ser. NDSS'20, February 2020.

[9] E. Manzoor, S. M. Milajerdi, and L. Akoglu, "Fast memory-efficient anomaly detection in streaming heterogeneous graphs," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2939672.2939783 pp. 1035–1044.

[10] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "Holmes: Real-time apt detection through correlation of suspicious information flows," in *2019 2019 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2019. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP.2019.00026

[11] P. Gao, X. Xiao, D. Li, Z. Li, K. Jee, Z. Wu, C. H. Kim, S. R. Kulkarni, and P. Mittal, "SAQL: A stream-based query system for real-time abnormal system behavior detection," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/gao-peng pp. 639–656.

[12] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates, "NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage," in *26th ISOC Network and Distributed System Security Symposium*, ser. NDSS'19, February 2019.

[13] W. U. Hassan, A. Bates, and D. Marino, "Tactical Provenance Analysis for Endpoint Detection and Response Systems," in *41st IEEE Symposium on Security and Privacy (SP)*, ser. Oakland'20, May 2020.

[14] P. D. Wajih Ul Hassan, Mohammad Noureddine and A. Bates, "OmegaLog: High-Fidelity Attack Investigation via Transparent Multi-layer Log Analysis," in *27th ISOC Network and Distributed System Security Symposium*, ser. NDSS'20, February 2020.

[15] K. Pei, Z. Gu, B. Saltaformaggio, S. Ma, F. Wang, Z. Zhang, L. Si, X. Zhang, and D. Xu, "Hercule: Attack story reconstruction via community discovery on correlated log graph," in *Proceedings of the 32Nd Annual Conference on Computer Security Applications*, ser. ACSAC '16. New York, NY, USA: ACM, 2016. [Online]. Available: http://doi.acm.org/10.1145/2991079.2991122 pp. 583–595.

[16] M. N. Hossain, S. M. Milajerdi, J. Wang, B. Eshete, R. Gjomemo, R. Sekar, S. Stoller, and V. Venkatakrishnan, "SLEUTH: Real-time attack scenario reconstruction from COTS audit data," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/hossain pp. 487–504.

[17] K. H. Lee, X. Zhang, and D. Xu, "High Accuracy Attack Provenance via Binary-based Execution Partition," in *Proceedings of NDSS '13*, Feb. 2013.

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

[19] 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 *Proceedings of the 25th ISOC Network and Distributed System Security Symposium*, ser. NDSS'18, San Diego, CA, USA, February 2018.

[20] K. H. Lee, X. Zhang, and D. Xu, "LogGC: Garbage Collecting Audit Log," in *Proceedings of the 2013 ACM SIGSAC conference on Computer and Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516731 pp. 1005–1016.

[21] 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*, ser. CCS '16. New York, NY, USA: ACM, 2016. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978378 pp. 504–516.

[22] M. N. Hossain, J. Wang, R. Sekar, and S. D. Stoller, "Dependence-preserving data compaction for scalable forensic analysis," in *Proceedings of the 27th USENIX Conference on Security Symposium*, ser. SEC'18. Berkeley, CA, USA: USENIX Association, 2018. [Online]. Available: http://dl.acm.org/citation.cfm?id=3277203.3277331 pp. 1723–1740.

[23] A. Gehani, M. Kim, and J. Zhang, "Steps Toward Managing Lineage Metadata in Grid Clusters," in *1st Workshop on the Theory and Practice of Provenance*, ser. TaPP'09, February 2009.

[24] Y. Xie, D. Feng, Z. Tan, L. Chen, K.-K. Muniswamy-Reddy, Y. Li, and D. D. Long, "A Hybrid Approach for Efficient Provenance Storage," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, ser. CIKM '12, 2012.

[25] Y. Xie, K.-K. Muniswamy-Reddy, D. Feng, Y. Li, and D. D. E. Long, "Evaluation of a Hybrid Approach for Efficient Provenance Storage," *Trans. Storage*, vol. 9, no. 4, pp. 14:1–14:29, Nov. 2013. [Online]. Available: http://doi.acm.org/10.1145/2501986

[26] 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: ACM, 2015. [Online]. Available: http://doi.acm.org/10.1145/2818000.2818039 pp. 401–410.

[27] A. Bates, K. R. B. Butler, and T. Moyer, "Take Only What You Need: Leveraging Mandatory Access Control Policy to Reduce Provenance Storage Costs," in *7th Workshop on the Theory and Practice of Provenance*, ser. TaPP'15, July 2015.

[28] S. Ma, X. Zhang, and D. Xu, "ProTracer: Towards Practical Provenance Tracing by Alternating Between Logging and Tainting," in *Proceedings of NDSS '16*, Feb. 2016.

[29] C. Chen, H. T. Lehri, L. Kuan Loh, A. Alur, L. Jia, B. T. Loo, and W. Zhou, "Distributed provenance compression," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. New York, NY, USA: ACM, 2017. [Online]. Available: http://doi.acm.org/10.1145/3035918.3035926 pp. 203–218.

[30] A. Bates, D. Tian, G. Hernandez, T. Moyer, K. R. Butler, and T. Jaeger, "Taming the Costs of Trustworthy Provenance through Policy Reduction," *ACM Trans. on Internet Technology*, vol. 17, no. 4, pp. 34:1–34:21, sep 2017.

[31] R. Ahmad, M. Bru, and A. Gehani, "Streaming provenance compression," in *Provenance and Annotation of Data and Processes*, K. Belhajjame, A. Gehani, and P. Alper, Eds. Cham: Springer International Publishing, 2018, pp. 236–240.

[32] 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*, ser. CCS '18. New York, NY, USA: ACM, 2018. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243763 pp. 1324–1337.

[33] W. U. Hassan, N. Aguse, M. Lemay, T. Moyer, and A. Bates, "Towards Scalable Cluster Auditing through Grammatical Inference over Provenance Graphs," in *Proceedings of the 25th ISOC Network and Distributed System Security Symposium*, ser. NDSS'18, San Diego, CA, USA, February 2018.

[34] Y. Ben, Y. Han, N. Cai, W. An, and Z. Xu, "T-tracker: Compressing system audit log by taint tracking," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, Dec 2018, pp. 1–9.

[35] 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.

[36] M. W. D. Center, "Event logging," 2018, https://docs.microsoft.com/en-us/windows/win32/eventlog/event-logging.

[37] M. W. D. Center, "About event tracing," 2018, https://docs.microsoft.com/en-us/windows/win32/etw/about-event-tracing.

[38] RedHat, "Linux audit," 2019, https://people.redhat.com/sgrubb/audit/.

[39] S. T. King and P. M. Chen, "Backtracking intrusions," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, p. 223âĂŞ236, Oct. 2003. [Online]. Available: https://doi.org/10.1145/1165389.945467

[40] K.-K. Muniswamy-Reddy, U. Braun, D. A. Holland, P. Macko, D. Maclean, D. Margo, M. Seltzer, and R. Smogor, "Layering in provenance systems," in *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, ser. USENIX'09. Berkeley, CA, USA: USENIX Association, 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855807.1855817 pp. 10–10.

[41] A. Gehani and D. Tariq, "Spade: Support for provenance auditing in distributed environments," in *Proceedings of the 13th International Middleware Conference*, ser. Middleware '12. New York, NY, USA: Springer-Verlag New York, Inc., 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2442626.2442634 pp. 101–120.

[42] A. Bates, D. J. Tian, K. R. Butler, and T. Moyer, "Trustworthy whole-system provenance for the linux kernel," in *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, 2015. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/bates pp. 319–334.

[43] "About purging reports," https://support.symantec.com/us/en/article.howto129116.html, 2019.

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

[45] 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, pp. 504–516.

[46] W. U. Hassan, M. Lemay, N. Aguse, A. Bates, and T. Moyer, "Towards scalable cluster auditing through grammatical inference over provenance graphs," in *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018. [Online]. Available: http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_07B-1_Hassan_paper.pdf

[47] 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, pp. 1324–1337.

[48] G. Shah, A. Molina, and M. Blaze, "Keyboards and covert channels," in *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, ser. USENIX-SS'06. USA: USENIX Association, 2006.

[49] A. Chen, W. B. Moore, H. Xiao, A. Haeberlen, L. T. X. Phan, M. Sherr, and W. Zhou, "Detecting covert timing channels with time-deterministic replay," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. Broomfield, CO: USENIX Association, 2014. [Online]. Available: https://www.usenix.org/conference/osdi14/technical-sessions/presentation/chen_ang pp. 541–554.

[50] J. Leskovec, "Snap: Stanford network analysis project," 2009, http://snap.stanford.edu/.

[51] AlDanial, "`cloc`: Count lines of code," 2019, https://github.com/AlDanial/cloc.

[52] D. T. Computing, "Transparent computing engagement 3 data release," 2020, https://github.com/darpa-i2o/Transparent-Computing.

[53] B. Eshete, R. Gjomemo, M. N. Hossain, S. Momeni, R. Sekar, S. D. Stoller, V. N. Venkatakrishnan, and J. Wang, "Attack analysis results for adversarial engagement 1 of the darpa transparent computing program," *ArXiv*, vol. abs/1610.06936, 2016.

[54] M. N. Hossain, S. M. Milajerdi, J. Wang, B. Eshete, R. Gjomemo, R. Sekar, S. Stoller, and V. Venkatakrishnan, "{SLEUTH}: Real-time attack scenario reconstruction from {COTS} audit data," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 487–504.

[55] Exploit-DB, "Unrealircd 3.2.8.1 - backdoor command execution," 2010, https://www.exploit-db.com/exploits/16922.

[56] Exploit-DB, "vsftpd 2.3.4 - backdoor command execution," 2011, https://www.exploit-db.com/exploits/17491.

[57] Exploit-DB, "Webmin 1.920 - unauthenticated remote code execution," 2019, https://www.exploit-db.com/exploits/47230.

[58] Rapid7, "Wordpress admin shell upload," 2018, https://www.rapid7.com/db/modules/exploit/unix/webapp/wp_admin_shell_upload.

[59] rebootuser, "Linenum," 2019, https://github.com/rebootuser/LinEnum.

[60] "MITRE ATT&CK," https://attack.mitre.org, 2019.

[61] M. Bellare and B. Yee, "Forward integrity for secure audit logs," Computer Science and Engineering Department, University of California at San Diego, Tech. Rep., 1997.

[62] B. Schneier and J. Kelsey, "Cryptographic support for secure logs on untrusted machines." in *Proc. of the USENIX Security Symposium (USENIX)*, 1998.

[63] B. Schneier and J. Kelsey, "Secure audit logs to support computer forensics," *ACM Transactions on Information and System Security (TISSEC)*, 1999.

[64] J. E. Holt, "Logcrypt: Forward security and public verification for secure audit logs," in *Proc. of the Australasian Information Security Workshop (AISW-NetSec)*, 2006.

[65] S. A. Crosby and D. S. Wallach, "Efficient data structures for tamper-evident logging," in *In Proceedings of the 18th USENIX Security Symposium*, 2009.

[66] A. A. Yavuz and P. Ning, "BAF: An efficient publicly verifiable secure audit logging scheme for distributed systems," in *Proc. of the Annual Computer Security Applications Conference (ACSAC)*, 2009.

[67] D. Ma and G. Tsudik, "A new approach to secure logging," *ACM Transactions on Storage (TOS)*, vol. 5, no. 1, 2009.

[68] R. Hasan, R. Sion, and M. Winslett, "The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance," in *Proceedings of the 7th USENIX Conference on File and Storage Technologies*, ser. FAST'09, San Francisco, CA, USA, Feb. 2009.

[69] A. A. Yavuz, P. Ning, and M. K. Reiter, "Efficient, compromise resilient and append-only cryptographic schemes for secure audit logging," in *Proc. of the International Conference on Financial Cryptography and Data Security (FC)*, 2012.

[70] J. P. Anderson, "Computer Security Technology Planning Study," Air Force Electronic Systems Division, Tech. Rep. ESD-TR-73-51, 1972.

[71] P. McDaniel, K. Butler, S. McLaughlin, R. Sion, E. Zadok, and M. Winslett, "Towards a Secure and Efficient System for End-to-End Provenance," in *Proceedings of the 2nd conference on Theory and practice of provenance*. San Jose, CA, USA: USENIX Association, Feb. 2010.

[72] D. Pohly, S. McLaughlin, P. McDaniel, and K. Butler, "Hi-Fi: Collecting High-Fidelity Whole-System Provenance," in *Proceedings of the 2012 Annual Computer Security Applications Conference*, ser. ACSAC '12, Orlando, FL, USA, 2012.

[73] A. Bates, D. Tian, K. R. Butler, and T. Moyer, "Trustworthy Whole-System Provenance for the Linux Kernel," in *Proceedings of 24th USENIX Security Symposium*, Aug. 2015.

[74] T. Pasquier, X. Han, M. Goldstein, T. Moyer, D. Eyers, M. Seltzer, and J. Bacon, "Practical whole-system provenance capture," in *Proceedings of the 2017 Symposium on Cloud Computing*, ser. SoCC '17. New York, NY, USA: ACM, 2017. [Online]. Available: http://doi.acm.org/10.1145/3127479.3129249 pp. 405–418.

[75] V. Karande, E. Bauman, Z. Lin, and L. Khan, "SGX-Log: Securing System Logs With SGX," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17, 2017.

[76] R. Paccagnella, P. Datta, W. U. Hassan, A. Bates, C. W. Fletcher, A. Miller, and D. Tian, "Custos: Practical Tamper-Evident Auditing of Operating Systems Using Trusted Execution," in *27th ISOC Network and Distributed System Security Symposium*, ser. NDSS'20, February 2020.

[77] Y. Kwon, D. Kim, W. N. Sumner, K. Kim, B. Saltaformaggio, X. Zhang, and D. Xu, "Ldx: Causality inference by lightweight dual execution," in *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '16. New York, NY, USA: ACM, 2016. [Online]. Available: http://doi.acm.org/10.1145/2872362.2872395 pp. 503–515.

[78] S. Ma, J. Zhai, F. Wang, K. H. Lee, X. Zhang, and D. Xu, "MPI: Multiple Perspective Attack Investigation with Semantic Aware Execution Partitioning," in *26th USENIX Security Symposium*, August 2017.

[79] Y. Kwon, F. Wang, W. Wang, K. H. Lee, W.-C. Lee, S. Ma, X. Zhang, D. Xu, S. Jha, G. Ciocarlie, A. Gehani, and V. Yegneswaran, "Mci: Modeling-based causality inference in audit logging for attack investigation," in *Proc. of the 25th Network and Distributed System Security Symposium (NDSS'18)*, 2018.

[80] S. M. Milajerdi, B. Eshete, R. Gjomemo, and V. N. Venkatakrishnan, "Propatrol: Attack investigation via extracted high-level tasks," in *Information Systems Security*, V. Ganapathy, T. Jaeger, and R. Shyamasundar, Eds. Cham: Springer International Publishing, 2018, pp. 107–126.

[81] F. Wang, Y. Kwon, S. Ma, X. Zhang, and D. Xu, "Lprov: Practical Library-aware Provenance Tracing," in *Proceedings of the 34th Annual Computer Security Applications Conference*, ser. ACSAC '18. New York, NY, USA: ACM, 2018. [Online]. Available: http://doi.acm.org/10.1145/3274694.3274751 pp. 605–617.