Log-Based Distributed Security Event Detection Using Simple Event Correlator

Justin Myers, Michael R. Grimaila, and Robert F. Mills
Center for Cyberspace Research, Air Force Institute of Technology, Wright-Patterson AFB, OH
{Justin.Myers,Michael.Grimaila,Robert.Mills}@afit.edu

Abstract
Log event correlation is an effective means of detecting system faults and security breaches encountered in information technology environments. Centralized, database-driven log event correlation is common, but suffers from flaws such as high network bandwidth utilization, significant requirements for system resources, and difficulty in detecting certain suspicious behaviors. Distributed event correlation is often assumed to be superior, but no research effort has been made which quantitatively evaluates its advantages and disadvantages. This research presents a distributed event correlation system which performs security event detection, and evaluates it experimentally, compared with a centralized alternative. The comparison measures the value in distributed event correlation by considering network bandwidth utilization, detection capability and database query efficiency. The implementation of these advantages allows a 99% reduction of network syslog traffic in the low accountability case. In addition, the system detects every implemented malicious use case, with a low false positive rate.

1. Introduction
The detection of malicious behavior on a network is a broad and difficult problem, with threats ranging from hackers and malware to insider attack. One method which has proven effective in detecting and combating these behaviors is analysis of log messages. Unfortunately, this analysis is expensive and difficult, and many organizations fail to properly resource log analysis tools [13][16]. The reasons for this include the sheer volume of data to collect, process and store [12], the difficulty of doing the actual analysis, the difficulty of effectively reporting the results, and limited investigative resources. This research develops and evaluates a distributed event correlation methodology to quantitatively examine its advantages and disadvantages, contributing a measurement and understanding of the value of distributed log event correlation and an example distributed log event correlation methodology, which together can lead to wider adoption of log analysis as an information security practice.

2. Log Analysis Today
The potential utility of log analysis for security event detection and mitigation is clear. A report on data breaches investigated by the Verizon Corp Business Risk Team reported in 2008 that 66% of organizations investigated had “sufficient evidence available within their logs to discover the breach had they been more diligent in analyzing such resources” [1]. In addition, NIST Special Publication 800-92 states that “routine log analysis is beneficial for identifying security incidents, policy violations, fraudulent activity, and operational problems” [4]. In spite of this potential, log analysis is not very commonly practiced. The Verizon Corp report found that only 20% of organizations do any automated log analysis, and even fewer use integrated Security Information and Event Management (SIEM) tools. An additional survey, the 2008 CSI Computer Crime and Security Survey, found that only 51% of respondents had a security log management system in place [10]. It is clear that while the analysis of log messages provides a great benefit to organizational security, it is still far from widely adopted.

This potential utility and limited penetration of log analysis for security event detection has resulted in several free and commercial log analysis tools. Many of these tools (including the one used in this research, the Simple Event Correlator) are quite good and are well positioned to solve a broad range of organizational needs [15]. However, the focus of this research is the quantitative evaluation of the benefits provided by distributing the event correlation activity itself, a concept which the authors have not found
3. Distributed Event Correlation

Event correlation, simply put, is a subset of log analysis which involves analyzing many aggregated pieces of information together and distilling the results of the analysis down into a single composite (or synthetic) event [2]. It is an increasingly important and accepted tool in managing complexity in enterprise networks [3]. It is also very well adapted to distributed log analysis since for many behaviors, a smaller group of events analyzed in aggregate can still detect the behavior.

Several of the difficulties encountered in log management can be answered by distributing the log collection and event correlation activities, as opposed to the more common practice of centralizing them. First, centralized log analysis is expensive because significant resources are required to transport the log messages to the centralized log server, do the analysis on all logs together and store the results. By contrast, distributing the event correlation activities reduces the resources needed to do the actual analysis, while providing flexibility in which messages are sent over the network and stored. Second, centralized log analysis is difficult because some behaviors present in log messages (such as those involving sliding time windows, state-based composite events and thresholds) are awkward for a centralized storage mechanism such as a database to detect. In addition, the sheer volume of events arriving at a centralized log server can flood the analysis engine [14]. Distributed event correlation provides solutions to both these issues, by using tools which are well-adapted for real-time behavior detection and by limiting the amount of overall traffic any one event correlation engine must analyze.

There are disadvantages to distributed event correlation. The fact that event correlation activities are distributed throughout the network produces a configuration disadvantage in that configuration changes must be made at many locations, instead of just one. To overcome this disadvantage, this research developed a remotely accessible shell script to dynamically switch between logging configurations. A second disadvantage could exist in the detection capability. Since attacks rarely involve only one machine, it is plausible that an attack could be distributed across multiple machines in such a way as to seem innocuous to any single machine. Since individual log producers (in this scheme) only have access to their own logs for event correlation, this attack would go undetected. In practice, there are several reasons why this is not a significant factor. First, while it is true that multiple machines are commonly involved in an attack, typically only one is the actual target. It is this machine which contains the most interesting information on the attack, and the one on which the attack is likely to be detected. Further, even an attack which has many targets interacts with each machine in some fashion. If that interaction can be characterized and detected, the sum of interactions with multiple machines can be detected as an attack even if a single machine only sees a small piece of the activity.

Another potential security issue with distributed event correlation lies in the increased attack surface presented by the many event correlation engines, as opposed to the centralized implementation and its single location. There are several dynamics that make this a difficult issue to evaluate. On one hand, if the single event correlator is compromised in the centralized scenario, the entire log collection and analysis process is compromised. On the other hand, an attacker with some knowledge of the distributed architecture (an insider, perhaps) would only need to compromise those log-producers which contained logs from the machines involved in the attack to discredit any evidence of the intrusion. If the attacker were more naïve, however, they might miss a log-producer which contained evidence of the intrusion. While not implemented in this research, a variation of the architecture could be envisioned where log-producers performed event correlation on their own logs as well as another's to increase the likelihood that reliable evidence of an attack would survive.

Distributed log analysis (and event correlation in particular) can be done many different ways. In this research, log messages are analyzed by an event correlation engine on the log-producing computer itself. Since a vast majority of suspicious behaviors (or an identifiable component of that behavior) can be detected by examining the logs of a specific target machine, this approach allows the reliable detection of suspicious behavior while providing the benefits listed above.

4. A Continuum of Accountability

The design of any log collection and analysis methodology must first account for the wide range of environments in which log analysis could be done. One important variable in these environments is the
existence of policy or regulatory requirements governing the collection and storage of log files. In the United States, regulations which have implications for log collection and analysis include the Federal Information Security Management Act (FISMA), the Gramm-Leach-Bliley Act (GLBA), the Health Insurance Portability and Accountability Act (HIPAA), the Sarbanes-Oxley Act (SOX), and the Payment Card Industry Data Security Standard (PCI DSS) [4]. This research describes this need to account for policy or regulatory requirements using a continuum of accountability levels. On one end are low accountability environments, in which there are few or no requirements regarding log collection and storage. On the other end are high-accountability environments, where there are strong policy and/or regulatory requirements mandating the (often centralized) collection and storage of log messages. As an organization moves along the continuum, the advantages provided by distributed event correlation change to reflect the changes in the logging environment.

In the low accountability case, distributed event correlation engines might be initially configured to send only synthetic events, generated once a suspicious behavior is detected. Since centralized collection and storage is not required, this configuration can dramatically reduce the bandwidth requirements necessary for conducting log analysis activities.

In high-accountability environments, log messages must be collected and stored centrally. In this case, a configuration which minimizes log message traffic over the network would be impractical. However, distributed event correlation is not without its advantages in this type of environment. In these environments, it is reasonable to assume that infrastructure has been built to support that volume of traffic. Thus one could run the event correlation engines and insert the synthetic events into the raw log message stream. These synthetic events add context to the raw events, offering greater efficiency in subsequent centralized event correlation activities. In addition to these two examples, variations and combinations can be imagined for many points along the continuum.

5. Dynamic Logging Configuration

For this research, four logging modes were designed to implement various configurations along the continuum [8]. Mode 1 is the low-accountability case described in the previous section, where only synthetic events are passed from log-producing machines to the central log server (or analyst workstation). Mode 2 is a hybrid mode, of which there could be many. In this implementation, Mode 2 behavior involves raw logs being passed along only when that combination of log messages triggered a synthetic event. These raw logs would then provide more granularity to an analyst, giving all available information on that behavior from that log source, while still providing a marked decrease in log-based network traffic. Mode 3 is the high-accountability case described in the previous section, containing all log messages and all synthetic events. Mode 4 contains all raw log messages and no synthetic events, to represent data collected without a distributed event correlation system in place. Conceptually, a Mode 0 could exist which passed along no log traffic. This could be useful in a small network, or to prevent traffic sniffing for short periods of time. Of course, there are many variations of Mode 2 which can be imagined as well.

In an ideal configuration, these modes would be more or less interchangeable. For instance, a lower accountability environment might use Mode 1 as a default configuration, switching to Mode 3 for some period of time after an alert is produced to be able to more closely monitor activity at that particular log-producer. This dynamic configuration would allow for the optimal blend of efficiency and detection capability, especially in low-accountability environments. The ability to dynamically change the type and amount of log events that are passed back to the log server is a key feature of our approach.

6. Simple Event Correlator

The event correlation engine chosen for this research is the Simple Event Correlator (SEC) [15]. SEC is a lightweight and flexible event correlation tool written in Perl by Risto Vaarandi. It is available for free download at http://simple-evcorr.sourceforge.net/. SEC was chosen for this research for several reasons. First, a lightweight solution was highly desirable since this experimental design places the distributed event correlation activities directly on the log-producing machines. Second, SEC is very easy to configure – the rules are written in a standard format, and there are several good sources of documentation on how to write them [11]. Third, SEC is used in many organizations over a wide range of industries, giving credence to its effectiveness in providing awareness of the status of organizational resources [14].
7. Experimental Design

The experimental setup for this research was designed with realism in mind, so that the results could be plausibly extrapolated to real-world systems. Even though the scope of this research was limited to web server logs, measures were taken to ensure that the architecture could provide benefit with logs from any source. With this in mind, use cases were chosen that represent either real-world attacks or represent more complex attacks that might be interesting from a web server perspective. These use cases and their sources are shown in Table 1.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL/Command Injection</td>
<td>OWASP Top Ten 2010 RC1</td>
</tr>
<tr>
<td>Cross Site Scripting</td>
<td>OWASP Top Ten 2010 RC1</td>
</tr>
<tr>
<td>Broken Authentication and Session Management</td>
<td>OWASP Top Ten 2010 RC1</td>
</tr>
<tr>
<td>Insecure Direct Object References</td>
<td>OWASP Top Ten 2010 RC1</td>
</tr>
<tr>
<td>Cross Site Request Forgery</td>
<td>OWASP Top Ten 2010 RC1</td>
</tr>
<tr>
<td>Security Misconfiguration</td>
<td>OWASP Top Ten 2010 RC1</td>
</tr>
<tr>
<td>Failure to Restrict URL Access</td>
<td>OWASP Top Ten 2010 RC1</td>
</tr>
<tr>
<td>Unvalidated Redirects and Forwards</td>
<td>OWASP Top Ten 2010 RC1</td>
</tr>
<tr>
<td>Insecure Cryptographic Storage</td>
<td>OWASP Top Ten 2010 RC1</td>
</tr>
<tr>
<td>Insufficient Transport Layer Protection</td>
<td>OWASP Top Ten 2010 RC1</td>
</tr>
<tr>
<td>Naïve web crawler</td>
<td>Previous research</td>
</tr>
<tr>
<td>Delayed web crawler</td>
<td>Previous research</td>
</tr>
<tr>
<td>Excessive downloads</td>
<td>Previous research</td>
</tr>
<tr>
<td>Excessive Access Attempts</td>
<td>Previous research</td>
</tr>
</tbody>
</table>

The first ten use cases in this table are taken from the OWASP Top Ten Release Candidate for 2010 [5]. The OWASP Top Ten is a well-respected list of the most common and dangerous attacks on web applications. The last four are insider threat-related scenarios which were identified as interesting by previous research [6][7][8][9]. They are more complex than the OWASP use cases, and provide insight into how the distributed event correlation architecture works with more involved behavior detection.

The network design for this experiment was likewise designed with realism in mind. The network contains three operational subnets, each with a web server and a workstation. The web servers run the two most popular web server platforms, with two running Apache, and one running Microsoft's Internet Information Services (IIS). The workstations are outfitted with traffic generation software which makes them function as several seemingly independent (abstract) machines. For this research, each workstation was configured with a total of six abstract machines - five benign users and one malicious user. In addition, there are two administrative subnets – one for the traffic generation controller and one for the log server and a network storage device for backups. The log server is running Kiwi log server, a popular syslog-compatible log collection application. In addition, the log server is configured with Oracle database 10g to handle the centralized event correlation activities.

To ensure that each run of the experimental network produced the same conditions, the abstract machines were configured to perform their assigned tasks on a schedule over each 8-hour scenario. Web surfing traffic was scheduled on a 5 access per minute probability curve, and happened during the entire scenario. Malicious behaviors (use cases shown in Table 1) were distributed equally between the three workstations, and were scheduled to execute at regular intervals.

8. Metrics

Several metrics were chosen to measure the advantages of distributed event correlation on the experimental network. These metrics were the detection rate, the false positive rate, network load and database query execution time.

The first metric chosen is the detection rate. In order to add value to log event correlation activities, the methodology must be able to fulfill a basic requirement of event correlation - detecting suspicious behavior. The detection rate measures the methodology’s ability to detect the selected use cases. This measurement is taken by monitoring at the log server alerts produced by SEC instances at each web server. If an alert is generated when the attack is executed, the scenario is said to have been detected.

The second metric chosen is the false positive rate. Even if every scenario is detected correctly, a high false positive rate would decrease the credibility of alerts, as well as increase the expense associated with investigating alerts (in an operational environment). Thus, it is important that false positive rates be kept as low as possible. This measurement is taken by monitoring SEC alerts at the log server. If an alert is generated but no attack was executed, a false alarm is said to have been detected and this is factored into the overall false positive rate.

The third metric chosen addresses the value added through decreased resource consumption by monitoring the composition of network traffic. This
metric is composed of two components. First, the overall proportion of benign and known malicious traffic is monitored to ensure that the base rate of malicious traffic is somewhat realistic. It would be easy to detect attacks if all the traffic on the network were attack traffic. For the purposes of determining this base rate, malicious traffic is defined as log messages which are involved in malicious behavior. The second component is the amount of Syslog traffic - specifically, the difference in the volume of Syslog traffic between various logging configurations. This measurement is taken by monitoring all traffic via a SPAN port on the switch, using Ntop (http://www.ntop.org) to generate statistics on the data.

The fourth metric addresses the value of providing context with synthetic events by monitoring the execution times of database queries. With the addition of synthetic events, the database should be able to more efficiently identify a particular attack than if it had to find them without assistance. The magnitude of this boost in efficiency is measured by recording execution times of queries with and without synthetic event context in the database. The measurement and analysis of this metric is ongoing.

9. Experimental Results

This section presents the results of measuring the detection rate and network composition metrics.

To analyze the detectability metrics (detection rate and false positive rate), each alert generated was mapped and compared with scheduled attacks to determine whether or not that alert represented an actual attack. Table 2 shows the total number of alerts for each logging mode and the number of true and false positives. For Modes 1-3, alerts are defined as synthetic log events produced by SEC instances. For Mode 4, the output of database queries against the normalized database was used.

<table>
<thead>
<tr>
<th>Logging Mode</th>
<th>Total Alerts</th>
<th>True Positives</th>
<th>False Positives</th>
<th>False Negatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode 1</td>
<td>46</td>
<td>45</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Mode 2</td>
<td>42</td>
<td>40</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Mode 3</td>
<td>48</td>
<td>47</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Mode 4</td>
<td>28</td>
<td>26</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

These results show a low false positive rate and a perfect score for detection – all malicious behavior was detected, giving zero false negatives for all four runs of the experimental network.

Additional analysis was done to determine the base rate of malicious traffic. Mode 4 (raw events only) identified 49,770 log messages produced over the duration of the experimental run. Mode 2 (synthetic events and related raw events) identified 157 raw events which were related to attacks. Thus the overall base rate of malicious log messages was 157/49,770 = 0.00315. While this may be a high number for operational environments, it is an effective number in ensuring the realism of the experimental system.

These results together demonstrate that the distributed event correlation methodology which was implemented is indeed effective at detecting realistic malicious behavior through analysis of web server logs.

In measuring the network composition, the key focus was on the amount of Syslog traffic sent over the network in each logging mode. Figure 1 shows the absolute amount of Syslog traffic sent across the network in each logging mode.

The results presented in Figure 1 are dramatic, though not unexpected. The amount of Syslog traffic sent over the network increases significantly in Modes 3 and 4 as compared to Modes 1 and 2, due to the addition of every raw log generated on the log-producers. If Mode 4 is taken to represent a centralized architecture with no distributed event correlators, the best case reduction in Syslog traffic (Mode 4 to Mode 1) is 99.88%. The inclusion of hybrid logging modes demonstrates the potential range of traffic reduction which is possible, depending on organizational needs. The fact that Mode 3 and Mode 4 show the same amount of traffic is an anomaly – in fact, the proportion of Syslog traffic relative to all network traffic decreased in Mode 4 from Mode 3. This is what would be expected, since synthetic events are included in Mode 3 but not in Mode 4. However, Mode 4 tended to have more HTTP traffic for an unknown reason, and thus the total amount of Syslog traffic was the same as that of Mode 3. Still, these results show that distributed event correlation can provide a great deal of flexibility in Syslog network bandwidth utilization in a low to medium accountability environment.
10. Conclusions and Future Research

There is clearly untapped potential in log analysis as it is done currently. In spite of the proven security value in analyzing log messages, log analysis and event correlation are not often done due to difficulties and inadequacies of current techniques. Distributed event correlation has been proposed as a promising solution, but the advantages and disadvantages had not been quantitatively measured. This research found that not only is distributed event correlation capable of detecting many common web application attacks effectively and with a low false positive rate, it also has measurable advantages in providing up to 99.88% reduction in Syslog traffic and a great deal of flexibility in adapting network Syslog traffic utilization to meet the resource, policy, and regulatory needs of an organization. Future research is expected to show that it also provides a measurable boost in database query efficiency when centralized event correlation must be done.

Analysis of the database query efficiency measurements is ongoing. This ongoing research focuses on the case (discussed briefly in section 4) where policy or regulatory compliance mandates the centralized collection and storage of logs. Specifically, the measurements taken will fall into two categories – analysis of the addition of context (distributed synthetic events) into the raw event stream, and evaluation of the impact of log normalization on the resulting database. The anticipated value in these areas is that the addition of context and the normalization of log messages (through post-processing of database records) will yield greater efficiency in the centralized detection of security events.

11. Disclaimer

The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

12. References


