Design and Analysis of a Dynamically Configured Log-based Distributed Security Event Detection Methodology

Michael R. Grimaila, Justin Myers, Robert F. Mills, and Gilbert L. Peterson
United States Air Force Institute of Technology
2950 Hobson Way
Wright-Patterson AFB, OH 45433-7765

Abstract
Military and defense organizations rely upon the security of data stored in, and communicated through, their cyber infrastructure to fulfill their mission objectives. It is essential to identify threats to the cyber infrastructure in a timely manner so mission risks can be recognized and mitigated. Centralized event logging and correlation is a proven method for identifying threats to cyber resources. However, centralized event logging is inflexible and does not scale well because it consumes excessive network bandwidth and imposes significant storage and processing requirements on the central event log server. In this paper, we present a flexible, distributed event correlation system designed to overcome these limitations by distributing the event correlation workload across the network of event producing systems. To demonstrate the utility of the methodology, we model and simulate centralized, decentralized, and hybrid log analysis environments over three accountability levels and compare their performance in terms of detection capability, network bandwidth utilization, database query efficiency, and configurability. The results show that when compared to centralized event correlation, dynamically configured distributed event correlation provides increased flexibility, a significant reduction in network traffic in low and medium accountability environments, and a decrease in database query execution time in the high-accountability case.

Keywords: Distributed event correlation, security incident detection, log analysis, insider threats

1. Introduction

Military and defense organizations have long recognized the benefits of embedding information and communication technology (ICT) into their core mission processes. Within the United States (US) Department of Defense (DoD) military environment, information is continuously collected, processed, analyzed, aggregated, stored, and distributed for multiple purposes including situational awareness, operations planning, intelligence, and command decision making [1; 2]. Within the Defense Industrial Base (DIB), ICT enables information sharing across and between business units, increases operational efficiency, improves decision making quality, reduces delays, and results in significant cost savings. The extreme dependence on ICT also creates significant risks from threats to the confidentiality, integrity, and/or the availability of the data contained in, or transported between, systems and devices [3]. For example, the leak of more than 75,000 US classified documents on the Afghan war to the Wikileaks web site in July 2010 places US forces at risk because it exposes military tactics and procedures that could be exploited by US adversaries [4]. Secret data was accessed and copied by a former intelligence analyst who used his position and clearance far in excess of his official duties. This represents only one of many possible scenarios where it is essential to identify potential threats to the organizational ICT in a timely manner so that appropriate actions can be taken. While potential threats sources are numerous and include natural disasters, equipment failure, accidents, and errors; our research focuses upon detecting threats arising from the malicious activities of external hackers, trusted insiders (intentional and unintentional) and malware.

One approach which has proven to be an effective at detecting malicious activities is the analysis of

event logs [5]. Virtually all systems and devices generate events which can be monitored, logged, and analyzed to help administrators maintain situational awareness of the status of their ICT. Log analysis provides the ability to understand, troubleshoot, and diagnose faults; identify root causes; and to identify anomalous system, application, and user behaviors which place the ICT at risk [5]. Within military environments, the need for high accountability event logging across the enterprise was only practical for limited access, high security networks such as the Secure Internet Protocol Router (SIPR) or Joint Warfighter Intelligence Communication System (JWICS) networks. However, the recent widespread deployment of the Host Based Security System (HBSS) across all DoD computers enables event collection and transport to centralized log servers to facilitate event correlation within the Non-Secure Internet Protocol Network (NIPR) network [6].

Security Information and Event Management (SIEM) is a term that encompasses all of the activities surrounding the collection, logging, and analysis of system and application events to identify potentially malicious activities and system errors [7]. SIEM provides an integrated view of the events generated by the organizations ICT so that queries can be made to identify activities of interest. In corporate environments, the need for SIEM is primarily driven by regulatory compliance requirements. However, in military environments the value of SIEM lies is the ability to identify malicious activities in real-time. While SIEM has proven to be an effective means of detecting attacks against organizational ICT resources, many organizations fail to properly implement and properly resource SIEM capabilities [8] [9]. There are several reasons for this including the sheer volume of log data for collection and storage [10], the network bandwidth consumed when transporting events to a centralized log server [11], the difficulty of conducting log analysis (including issues of log normalization and event correlation) [12], the lack of definitive action following the reporting of problems once they are found [12], and limited investigative resources [8; 12; 13].

In this paper, we present the design and analysis of a dynamically configurable, distributed event correlation system designed to overcome these identified barriers which cause organizations to fail to implement a SIEM. The purpose of this research is to quantitatively determine if distributed event correlation techniques are superior to centralized techniques with regard to flexibility, network bandwidth consumption, detection capability, and database query efficiency. The remainder of this paper is structured as follows: In Section 2, we present the background of, and motivation for, security event log analysis and discuss event logging, event correlation, and centralized versus distributed event correlation; in Section 3, we introduce a dynamically configurable distributed security event correlation methodology; in Section 4, we present the experimental modeling and simulation environment; in Section 5, we discuss the results and analysis of the experiment; and in Section 6, we present our conclusions and discuss future research directions.

2. Background

In this section, we present the evolution of event logging and analysis for the detection of security events, compare and contrast centralized and distributed event correlation, discuss data normalization, and review recent surveys which highlight the importance of security event logging and analysis to motivate the need for the research.

2.1. Event Logging and Analysis

In the early days of computer and network management, event logs were used simply for diagnosing when a system, application, or device stopped functioning properly. Events were reviewed primarily to discern the internal state of those systems to aid in troubleshooting and little else [10]. However the

value of event logs for security auditing purposes was recognized as early as 1980 when they were used to detect unauthorized access to files [14]. At this time, a collection of systems was typically monitored by using dedicated communications links which provided the ability for administrators to log into the systems from a central location and remotely review event logs. As networking technologies matured, standardization of internetworking protocols and lower cost networking hardware made it easier to remotely monitor networked systems and devices. One of the protocols that focused on the problem of event logging was the *syslog* protocol [15]. The *syslog* protocol provides a standard mechanism to transport events from any number of remote systems to a central log server where they are stored for subsequent review and analysis. Typically, a *syslog* process is run each system and is used to collect event messages from all running processes, log these event messages to one or more files as well as the system console, and forward the event messages across the network to a *syslog* process running on another machine using *syslog* protocol [16]. To assure the secure collection and retention of event messages, best practice requires implementing a hardened, centralized log server where all event messages sent from the organization's network devices are stored [5]. The idea is that if a system is compromised, the event messages related to the compromise will have already been forwarded to the central log server significantly limiting the ability of the attacker to erase evidence of their attack. In some cases, multiple log servers may be used and configured in a hierarchical fashion to aggregate messages. A key benefit of centralized event storage is that an analyst could use pattern matching utilities to filter event logs to identify events of interest. While useful, this approach was limited by an inability to capture temporal dependencies and track real-time state information which made it difficult to automatically detect potentially interesting complex event patterns [17; 18].

2.2 Event Correlation

Event correlation is the activity of finding relationships between two or more events [5]. Event correlation is usually implemented as a software tool that parses an event stream, or a stream of events in a log file, to locate predefined patterns of interest. Event correlation provides an analyst with the ability to automate the identification of complex patterns of interest in large amounts of data. Once an event of interest is detected, it can be used to trigger an alert, take a specific action, notify the analyst, and/or generate a new "synthetic" event, if desired.

A number of different approaches for implementing event correlation have been investigated including finite state machine based [19], rule based event correlation [20; 21; 22; 23; 24], codebook based [25], case-based reasoning [26; 27], genetic algorithms [28], graph based [29; 30; 31], model based reasoning [32; 33], neural network based [34], and probabilistic [35] methods. Each approach to event correlation provides unique benefits and limitations, so the best approach is strongly dependent on the application environment.

Event correlation overcomes the limitations of simple event filtering by incorporating the ability to capture and track temporal and state information, enabling the detection of more complex events. For example, suppose an analyst was reviewing a web server access log to identify any Internet Protocol (IP) addresses which downloaded more than 100 MB of web server content within a 24 hour period. Conceptually, this would require tracking accesses by each unique IP address over a 24 hour sliding window and integrating the amount of downloaded data within the specified time interval. Using conventional event filtering, it is not possible to answer this type of question in a timely manner. However, by encoding an event correlation rule which considers the relationships between web server access requests it is possible to quickly process large data sets in an automated manner. The use of event correlation provides significant benefits in the analysis of large event sequences because it allows for automated aggregation, compression, escalation, filtering, generalization, masking, and the removal of duplicate messages. Further, it aids in root cause analysis by providing valuable information that can

be used to identify dependencies between events which are precursors to failures [36].

Commercial log analysis tools incorporate event correlation capabilities because it is an increasingly important and accepted tool for managing complexity in enterprise networks [37; 38; 39]. There are numerous open source event correlation tools available including SWATCH [40], LogSurfer [41], SEC [24], OSSEC [42], Prelude [43], OSSIM [44], Drools [45], and Esper [46]. These tools have proven useful in a wide variety of public and private sector environments and provide robust capabilities at a low cost.

Event correlation has also been recognized as being a powerful tool for understanding real-time events in the military battlespace [47]. In any large scale network, event and alarm-producing systems are distributed across the entire network, comprising some (and possibly all) of the computing and infrastructure systems in the network. In common configurations this virtually guarantees a volume of data, in the form of log messages, which is infeasible for a human operator to manage efficiently [5; 10; 13; 48]. It is clear that using event correlation provides management with the ability to gain deeper insight into activities occurring within the organizations ICT infrastructure.

2.3 Centralized versus Distributed Event Correlation

The current best practice for event collection, monitoring and analysis is based upon a centralized architecture where each system or device that generates events sends the events to a single, hardened centralized log server where event correlation is performed [5]. This architecture is attractive because event correlation is conducted centrally and a malicious actor cannot eliminate evidence of their attack on a system if the events are transmitted to, and stored in, a centralized hardened log server. However, as the number of log-producing systems and devices increases, so does the volume of events that must be processed, stored, and correlated. This places a significant demand on the computing resources (processor time, memory, disk space) that must be devoted to event correlation activities. In extreme cases, the workload can overload the ability for the event correlation engine leading to loss of detection capability even when the centralized activity is distributed across network domains [49].

In contrast, the distributed event correlation architecture enables a portion of the event correlation workload to be allocated to the log producers [25; 49; 50; 51; 52]. Specifically, event correlation patterns that involve only events generated by a single system can be encoded and distributed to the log producers. The primary benefit of this approach is that the log producers use their own processing resources to conduct event correlation, reducing the resource burden on the centralized log server. When an event of interest occurs on a system as defined by the event correlation rules, a synthetic event is generated and sent to the centralized log server. Note that the synthetic event can be sent in addition to or in place of a portion of the existing event stream. In general, distributed event correlation addresses many of the disadvantages with centralized event correlation. First, distributing event correlation activities reduces the total workload on the centralized log server. This enables a more efficient use of the log server resources and reduces the likelihood of overwhelming the event correlation engine. Research has shown that the performance impact of co-locating a lightweight event correlation engine with a log producing application such as a web server is minimal [24]. Such co-location makes efficient use of resources and further reduces the cost of implementing such a system. Second, the ability to squelch event streams at the source based upon detected events provides the option to reduce the volume of log traffic that must traverse across a network. This reduces the overall network load, reducing the negative impact associated with the centralized logging approach.

2.4 Log Data Normalization

While virtually every system, application, and device has the capability to generate and to log event messages, the message format often varies widely. The *syslog* protocol defines how events are transported through the network, not the detailed elements that each device may generate. This presents a difficulty for centralized log analysis, since the analysis engine must be configured to understand and correlate event logs from every expected source. Since this can be a resource intensive activity for an organization with logs in many different formats, log normalization is an attractive alternative. Log normalization is defined as the conversion of each log data field to a particular data representation and categorizing the resulting fields consistently prior to storage [5]. This preprocessing activity typically occurs at the centralized log server prior to data storage. Note that log normalization is not the same activity as traditional database normalization. In log normalization, the focus is upon standardizing the log storage format so that event correlation can be easily conducted. In contrast, database normalization is focused upon efficiently storing data within the database by eliminating redundant data and enforcing consistency by requiring the data to be stored as collection of tables each which represents a well structured relation. It is important to note that both types of normalization can impact the efficiency of event correlation both positively and negatively depending upon the environment. While log normalization generally aids in the efficiency of log analysis activities, it incurs a performance cost at the log server due to processing that must occur prior to event storage and the increased complexity of database queries.

An alternate way to achieve log data normalization without the performance overhead on the log server is to embed standard data representations in each system, device, or application so that the log producers generate event messages in normalized form. This is stated purpose of MITRE's Common Event Expression (CEE) project [53]. While generating events in a standard format provides significant analysis benefits, ultimately the success of the approach will depend on vendor adoption.

2.5 The Importance of Security Event Logging and Analysis

A recent survey of 2,100 global organizations conducted by Symantec Corporation found that 75% experienced cyber attacks in 2009 with average combined loss of $2 million per year [54]. Of the organization surveyed, 42% of ranked cyber security as the top risk to their organization, ranking it higher than traditional crime, natural disasters and terrorism. A key recommendation of the report was the need for improved automation and efficiency in the event log monitoring and analysis activity.

The 2009 Verizon Corporation Business Risk Team report identified that 66% of security breach victims represented in their caseload had "sufficient evidence available within their logs to discover the breach had they been more diligent in analyzing such resources" [55]. In fact, only 6% of those surveyed discovered breaches through event monitoring or log analysis. While this highlights that effectiveness of real-time analysis of logs, it also reveals that many organizations fail to properly implement and resource the activity. Figure 1 shows how event monitoring and log analysis solutions are being implemented in organizations according to this report. These findings show that most organizations rely on basic system and device logs, with a surprisingly low number using solutions such as intrusion detection systems and automated log analysis. This means that while organizations are largely collecting the data, the analysis of that data is still very immature, operationally speaking.

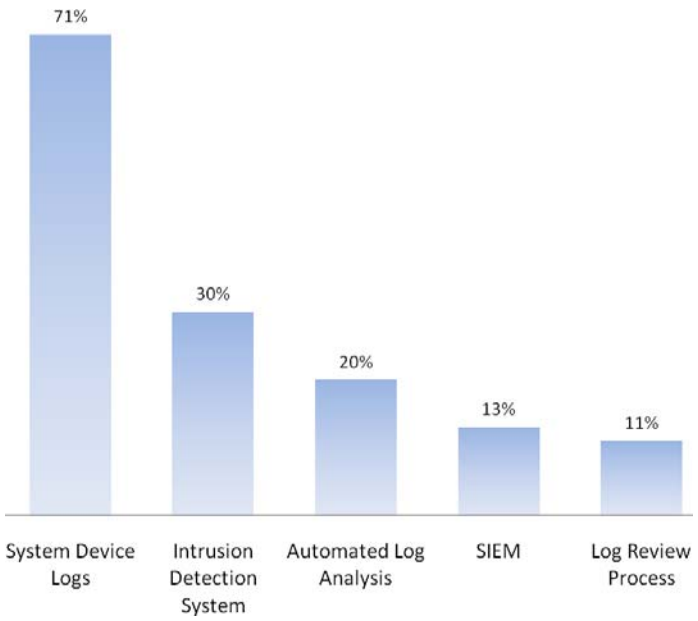

Figure 1. Verizon Data Breach Report: Detective Controls by percent of breach victims [55]

The 2008 CSI Computer Crime and Security Survey produced similar results, showing that failure in the area of system and transaction log monitoring was a significant factor in the success of attacks. The survey also showed that security log management is far from widely implemented, with 51% of respondents reporting that they have such a system in place [56].

Failing to properly implement an SIEM capability can result in legal liability should the organization experience a breach of Personally Identifiable Information (PII). Consider the 2009 Federal Trade Commission (FTC) decision against Geeks.com [57]. At issue in the ruling was the prolonged leakage of PII, including credit card information, from Geeks.com servers. The FTC ruling identified the lack of effective monitoring as one of the factors which contributed to this leak [8; 57]. Had Geeks.com implemented a log management strategy, they likely would have been able to detect the data leak early on and prevent further exploitation [8].

Monitoring event logs for security violations is an important activity for the DoD. US Deputy Defense Secretary William Lynn recently stated that "Rather than preventing people from having access to the data, could we do things like credit card companies do, which is to look for anomalous behavior" [4]. It is clear that if an effective SIEM capability was implemented across the DoD, the likelihood of the Wikileaks incident being detected would have been much higher. For this reason, we believe that it is essential to identify and overcome barriers in wide scale SIEM implementation in large organizations such as the DoD.

3. A Dynamically Configured Distributed Security Event Correlation and Detection Architecture

In order to address limitations of the centralized logging and analysis architecture identified in Section 2, we designed a dynamically configurable architecture where the event correlation activities are conducted not only on the log server, but also at the log-producing machines themselves. Our design was motivated by the need to distribute event correlation workload, to reduce network bandwidth consumption, improve the efficiency of centralized log server event correlation, and to enable dynamic

configuration of the event log collection and analysis. To achieve these objectives, our design sought to encompass three desirable properties.

The first desirable property is that we want to be able to selectively squelch event streams by developing event correlation rules that allowed us to transform a specific collection of events into a single synthetic event. For example, consider an adversary who is conducting a port mapping network reconnaissance activity against a networked system in preparation of an attack. This is the cyber equivalent of the Intelligence Preparation of the Battlespace (IPB) in the physical world [1]. In this case, a tool such as *nmap* [58] would be used to automatically access each of the ports on the system in rapid succession to determine if each port is open, a precursor condition to perform a network based attack. If the system were configured to generate an event for each failed connection attempt, there would be a large number of events that would be sent to the centralized log server. In contrast, if local event correlation was performed on the system, the port mapping activity would be detected after a small number of failed connection attempts from the same Internet Protocol (IP) address. Once detected, all subsequent failed connection attempt messages resulting from the suspect IP address would be filtered out, replaced by a single synthetic event indicating that the IP address is port mapping the system. Note that while events are filtered out of the event stream transported to the centralized log server, the events are still stored locally on the host so that an administrator could investigate the incident in more detail if needed. This is a simple example which shows that distributed event correlation can be used to reduce the amount of data transported back to the centralized log server while maintaining the ability to detect a specific malicious activity. Figure 2 shows how event logging, collection and transport are typically implemented within a computer system. In this case, the computer operating system generates three log files: an application log, a security log, and a system log. This computer is also running a web server application that generates two log files: an access log and an error log. The event transport agent collects these raw events from these log files and formats them as *syslog* messages for transport back to the centralized log server. Figure 3 shows how the event transport agent can be augmented with event correlation and filtering capabilities to determine which events, if any, are transported to the central log server. This architecture enables an administrator to dynamically configure the event correlation and filtering that occurs on each system so that resource utilization can be controlled remotely based upon operational needs.

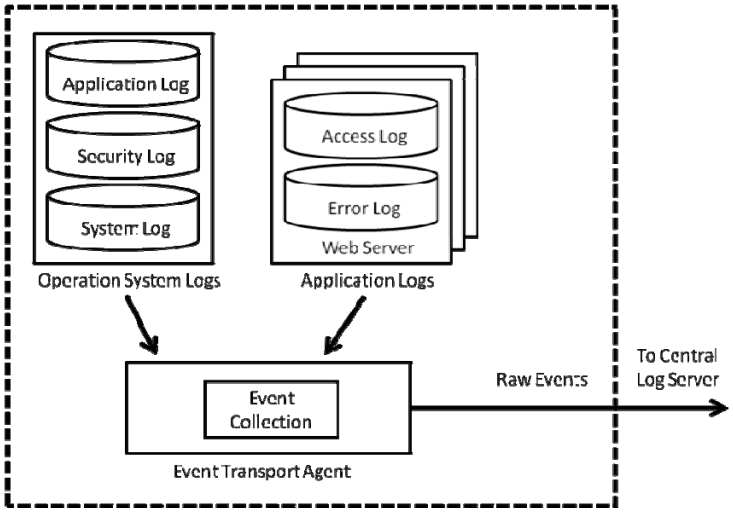

Figure 2. Typical event collection within a host

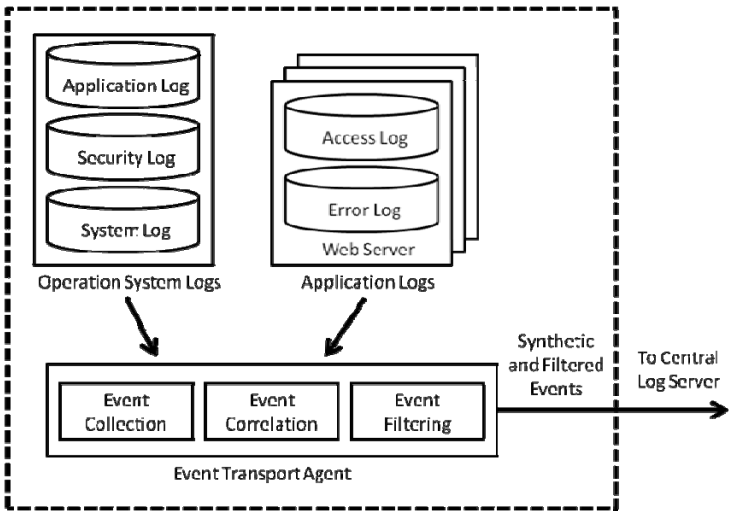

Figure 3. Enhanced event collection incorporating event correlation and filtering within a host

Second, we want to add synthetic events to the event stream generated by distributed event correlation that would improve the efficiency of the event correlation conducted at the centralized log server to more quickly identify event streams of interest. Figure 4 presents a visualization of how synthetic events produced through distributed event correlation would add value. Instead of querying each record (where one raw log entry equals one record) to search for a certain behavior, a query is made to search for a corresponding synthetic event which provides context on the behavior, enabling another, much smaller query to be made (if necessary) to locate the raw logs which triggered that synthetic event.

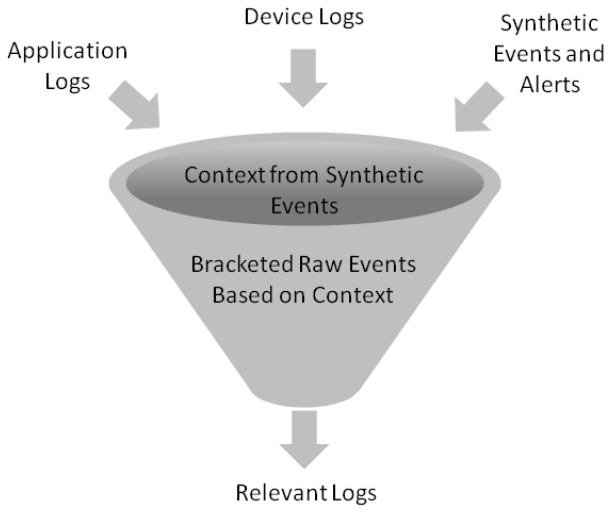

Figure 4. Bracketing of raw events in database queries using context from synthetic events [59]

The benefit of this approach is that when event correlation is conducted at the log server, the workload required to identify events of interest will be reduced. For example, consider a group of individuals

who possess the necessary clearances and are authorized to access an organizational intranet web server that contains sensitive intelligence information. While they may access any information on the web server, users are informed that it is a security policy violation to access data that is not relevant to their "need to know". In this case, it is highly desirable to alert security personnel if this security policy is violated so that a further investigation can be conducted. Now suppose a malicious insider attempted to download all of the web server content using a web crawler such as *wget* [60]. If the system were configured to generate an event for each request, there would be a large number of events that would be sent to the centralized log server. A security analyst would be responsible for running a set of queries periodically on the log server to identify patterns of interest. One of these queries would be constructed to identify all accesses from a single IP address, integrate the number of bytes in the downloaded web pages, and generate an alert when the sum exceeds a predetermined threshold value. The run time of the query depends on the complexity of the query and how much data resides in the database. As the database grows larger the amount of time required to complete the collection of queries would increase, leading to the possibility that the analyst would reduce the number of queries and miss detection of the potentially malicious activity. In contrast, if local event correlation was performed on the web server, a synthetic event indicating that a security policy violation has occurred would be inserted into the event stream and be stored in the central log server. Now the query run periodically on the central log server would look for the high priority synthetic events to identify malicious activity more quickly. This is a simple example which shows that distributed event correlation can be used to add context to the event stream reducing the workload on the centralized log server to identify specific malicious activity.

Finally, we wanted the ability to dynamically change the configuration on the log producing systems so that we could change the behavior of logging architecture on demand. The attractiveness of this methodology is that it provides the ability to dynamically configure systems to distribute event correlation workload, filter event streams before sending them to the central log server, and dynamically add context to event streams through the insertion of synthetic events which mark the location of "interesting" event clusters. Depending upon the desired accountability level, differing combinations of events will be sent from each log producer to the central log server. Benefits include the ability to limit network bandwidth utilization, the ability to focus collection efforts where they are most needed, and the ability to distribute the event correlation workload to the log producers to reduce the performance and resource demands experienced by the centralized log server. For example, Table 1 shows an example of how each log producing system can be dynamically configured to operate in one of four accountability levels, each with a differing amount of events that are transported to the central log server. The modes are listed in order of increasing amount of events generated, and hence the network bandwidth utilization required to transport the events to the centralized log server.

Table 1. Logging Mode Attributes

Logging Mode	Accountability Level	Raw Events	Filtered Events	Synthetic Events
0	None	No	No	No
1	Low	No	No	Yes
2	Medium	No	Yes	Yes
3	High	Yes	No	No
4	High	Yes	No	Yes

Mode 0 represents a no accountability case where the log producer sends no events to the centralized log server. This mode is used to disable the transport of events back to the log server and is needed

when network resources are too scarce or when central collection of the events is not required. It is important to emphasize that in this mode, as with all other modes, the raw events are stored locally on the system for a time period that is determined by the size of the disk storage allocated to local logs. Periodically, the local log files may be archived to another storage device or may be overwritten as determined by the organizations data retention policy.

Mode 1 represents a low accountability case where the log producer only generates synthetic events and does not pass any raw or filtered events to the central log server. Mode 1 is useful when there is a well defined subset of activities of interest at each system and the network bandwidth utilization is scarce. Systems and devices that are not mission critical can be configured in Mode 1 by default.

Mode 2 represents a medium accountability case where the log producer passes only some of the raw events and also sends generated synthetic events to the central log server. Mode 2 is useful when some detailed information must be passed to the central log server but the use of network bandwidth is to be reduced. Mode 2 is highly configurable in that the administrator defines exactly which raw events are suppressed, which raw events passed, and when synthetic events are generated. For instance, raw logs which do not correspond to malicious behavior could be suppressed, but raw events related to the suspect behavior as well as synthetic events could be forwarded on to the log server. This way, an analyst could examine the logs for more detail on the attack. Conversely, raw logs pertaining to the attack could be suppressed, and synthetic events would be sent along with non-alert-generating logs, so that additional analysis could be done on them later. In either case, the benefits of the distributed approach are readily apparent, but the amount of benefit depends upon the particular configuration. The resource consumption experienced by a system in Mode 2 is proportional to the complexity of the event correlation workload.

Mode 3 represents a high accountability case where the log producer sends all of raw events it generates, and no synthetic events, to the central log server. Mode 3 is the current best practice in centralized security event logging. Our research and the development of a method for dynamic configuration was driven by the drawbacks of this mode and the realization that we could be more efficient while still detecting event patterns of interest. While intuitively it does make sense to transport all events to a central log server for correlation, in practice the excessive consumption of network bandwidth, the required computational resources needed to conduct event correlation, and disk space are prohibitive for large enterprise networks.

Mode 4 represents a high accountability case where the log producer sends all of raw events plus the generated synthetic events to the central log server. Mode 4 exploits the idea that inserting synthetic events generated by local event correlation into the event stream reduces the workload and increases the detection efficiency at the central log server. In a high-accountability environment, such as highly classified networks where centralized, database-driven event correlation is mandatory, this architecture can sit lightly on top of existing infrastructure and provide context with synthetic events, improve semantic understanding of events, and provide higher level reasoning about the behavior captured in log events, making the centralized correlation activities more efficient and effective.

While each of the modes discussed above has benefits and drawbacks, we contend that switching between modes based upon the changing operational environment provides more efficient utilization of network, computational, and storage resources. In the next section, we seek to quantitatively determine the performance of each of these modes in a modeled and simulated network environment.

4. Experiment Design

In this section, we present the modeling and simulation environment used to quantitatively evaluate the performance of the proposed methodology in a realistic, repeatable manner. The experiment design section is segmented into five areas: network design, software selection, use case selection,

performance metrics, and experimental runs.

4.1. Network Model Design

In order to determine the effectiveness of the proposed log analysis methodology, we designed a network model to simulate a typical enterprise logging infrastructure. Figure 5 shows the overall configuration of the network annotated with IP address ranges and mission functions.

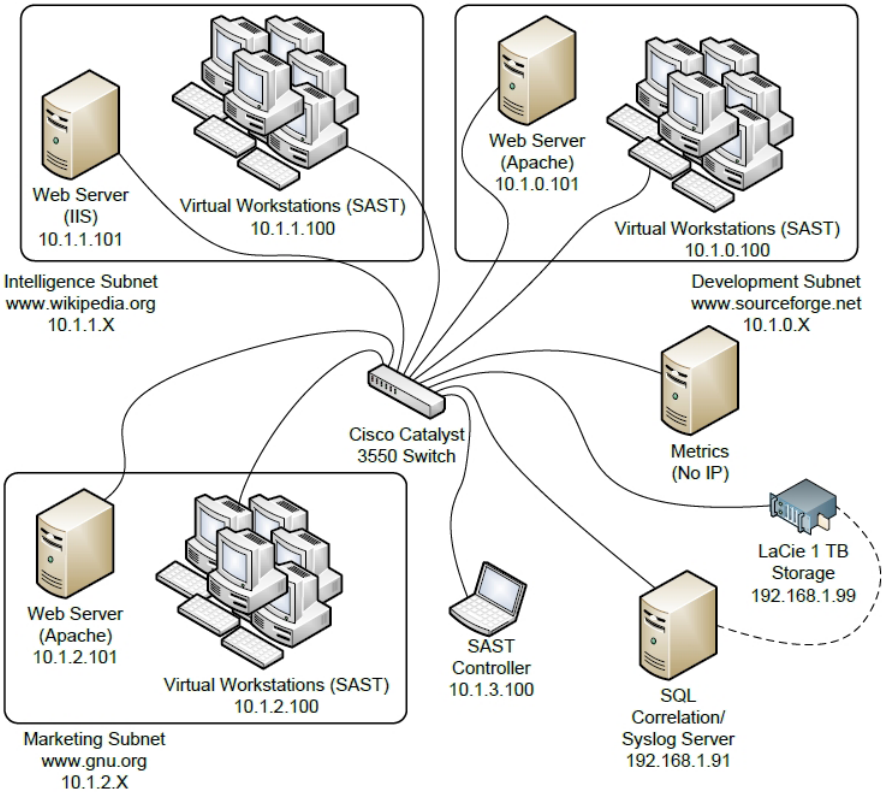

Figure 5. Experimental Network Model

The network model is comprised of nine computers and two network devices. The workstations and servers were implemented using mini-computers, and the SAST controller and log server were implemented using laptops. Table 2 gives the hardware specifications for each machine. The two network devices include a Cisco Catalyst 3550 switch with IOS version 12.1(22)EA4 that was used to facilitate the creation of Virtual Local Area Networks (VLANs) and a LaCie 1TB network storage disk was used to record event logs. The Cisco switch is used to divide the network into five VLANs with routing between each of the VLANs. Three of the VLANs correspond to the simulated Development, Intelligence and Marketing subnets, each containing a web server and a workstation. The workstation in each subnet is configured with SAST traffic generation software that makes the workstation appear on the network as 5 virtual workstations, for a total of 15 simulated virtual workstations for the whole network. The last two VLANs are administrative: one contains the SAST controller, and one contains the log server and network storage device.

Table 2. Hardware Specifications for Experimental Network Computers

Function	Servers, Workstations	SAST Controller	Log Server
Type	Mini PC	Laptop	Laptop
Model	AOpen MP945-D	Dell Latitude D630	HP Compaq 8710w
CPU	Celeron M 1.73 GHz	Core 2 Duo 2.6 GHz	Core 2 Duo 2.6 GHz
Memory	1 GB	4 GB	3 GB
Disk Space	150 GB	150 GB	150 GB

4.2. Software Selection

The configuration of software on the model network was chosen not only to facilitate the experiments to be performed on the network, but also to add realism and applicability to the results, so that they could plausibly apply to many common real-world network configurations. To achieve this goal, common popular operating systems and software packages were chosen when possible. There are five key software packages that were used for this experiment: the Security Assessment Simulation Toolkit (SAST) traffic generation package, the Simple Event Correlator (SEC) [24], Apache web server [61], IIS web server [62], Kiwi [63], *ntop* [64], and *syslog* [16]. Table 3 shows the operating system and software used on each machine in the experimental network model.

Table 3. Operating System and Software for Network Model Computers

Name	**Operating System**	**Software**
Development Workstations	Windows XP SP3	SAST 3.3.1
Development Server	Ubuntu 9.04 Server	Apache 2.2.11, SEC 2.5.3
Intelligence Workstations	Windows XP SP3	SAST 3.3.1
Intelligence Server	Windows Server 2003	Microsoft IIS, SEC 2.5.3
Marketing Workstations	Windows XP SP3	SAST 3.3.1
Marketing Server	Ubuntu 9.04 Server	Apache 2.2.11, SEC 2.5.3
SAST Controller	Windows 7	SAST 3.3.1
Log Server	Windows XP SP3	Kiwi Syslog Server 9.03, Oracle Database 10g
Network Monitor	Ubuntu 9.04	NTOP 3.3

4.2.1. Traffic Generator Configuration

Security Assessment Simulation Toolkit (SAST) is a proprietary tool developed by Pacific Northwest National Laboratory (PNNL) for use by US government organizations to generate realistic-looking traffic and facilitate exercise environments with benign and malicious traffic. In this research, SAST is being leveraged as a traffic generation tool, as well as a scheduler to run attack code in a scriptable, repeatable fashion. The configuration of SAST involves five interrelated components - tasks, actors, timelines, a scenario, and host service applications (HSAs). Figure 6 shows the relationships between these components in the configuration process.

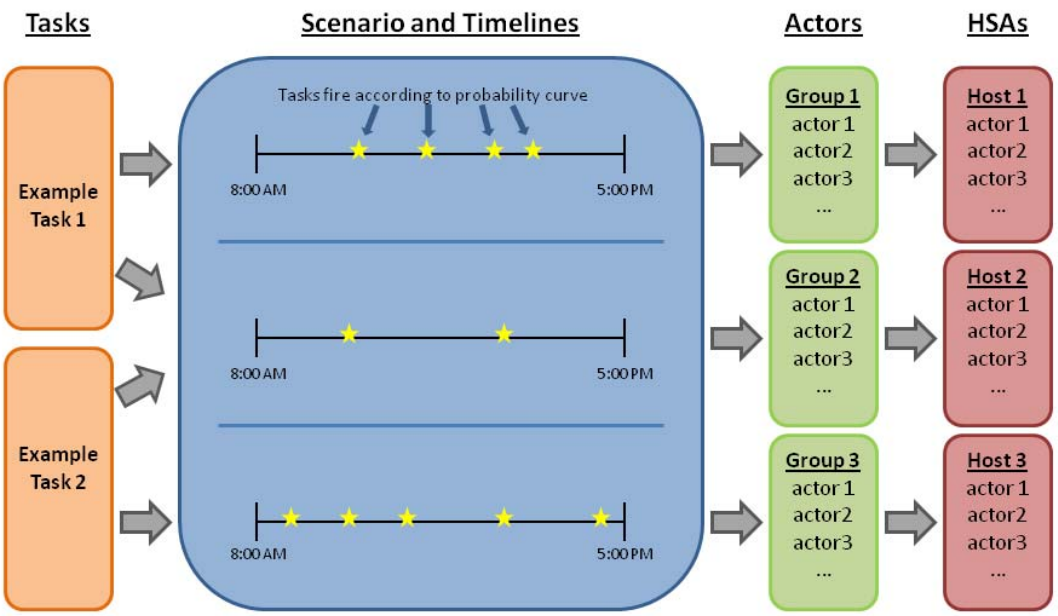

Figure 6. SAST Configuration

We now present the process for building a SAST scenario to gives insight into how a scenario executes. First, the specific tasks to be performed must be defined. These tasks can include surfing the web, checking e-mail, connecting to an FTP server, or any other behaviors. Next, those tasks are assigned to a timeline. As part of the configuration they are given a start and stop time, as well as assigned a probability curve that dictates the frequency and pattern of execution for that task in the timeline. The timeline, with its tasks, can then be assigned to one or more groups of actors. Each actor will appear on the network as a distinct entity when the scenario executes. Lastly, some combination of actors is assigned to a Host Service Application (HSA). An HSA may be located on a remote machine or it may be local. When the scenario is loaded, each HSA is given its assigned group of actors, along with those actor's timelines and tasks. Upon execution of the scenario, all HSAs move together through their respective timelines, executing tasks as configured.

For this research, SAST was configured with two types of tasks - web download tasks and command line tasks. The web download tasks are configured to download random pages on each of the three web servers, simulating a group of users surfing the web. They were run using a "5 per minute" probability curve, which means 5 requests were made at random times every minute. The command line tasks are configured to do one-time runs of Python scripts which perform attacks corresponding to each chosen use case. These tasks were run using the "single-shot" probability curve, which means that they only happened once, at a time specified in the configuration. Those tasks are then assigned to three timelines, one for each subnet of abstract clients on the network. The malicious tasks were spread evenly over three malicious timelines, while the three benign timelines downloaded websites. Actors are similarly organized in three groups - Development, Intelligence and Marketing, and each group is assigned its corresponding timeline. Finally, five actors from each benign group and one malicious actor are assigned to the HSA corresponding with the Workstation machine in their subnet. Each workstation is assigned an actor group, so that there are Development, Intelligence and Marketing workstations. Upon execution, the three workstation machines (with HSA software running) receive the same timeline from and sync up with the central controller. When the "Play" button is pressed, each

workstation performs the tasks included in the timelines assigned to it according to the configured probability curves and schedules.

4.2.2. Web Server Configuration

The choice was made early on that the content on each of the three web servers should be real-world content, rather than fabricated websites. This provides another layer of realism to the experiment. In keeping with the subnet naming convention, the content on the Development server was gathered from Sourceforge.net, a popular website for hosting of and collaboration on open source projects; the content on the Intelligence server was gathered from Wikipedia, the free online encyclopedia; and the content from the Marketing server was gathered from the GNU Operating System's homepage, a site dedicated to increasing awareness of free software. All content was gathered using *wget*.

Web servers can be configured to log many different information elements. The process of writing event correlation rules for detecting malicious web server activities requires developing a complete understanding of the possible observable events generated by a web server. Ideally, one would only select for the final logging configuration the subset of event elements that provides value in the detection of the set of malicious activities of interest. This process revealed not only which log elements were commonly relevant to all use cases, but also forced certain design decisions. For instance, SQL injection attacks are detected in the query string as GET parameters for the purposes of this research. While that is a plausible location for an injection attack, it is equally if not more likely that such an attack would be located in POST parameters. This information is not logged by default; in fact, IIS requires third-party software to log POST data. In Apache, the configuration is straightforward with the addition of the *mod_security* or *mod_dumpio*. Table 4 shows the Apache and IIS web server event log elements that were collected as determined by the logging configuration in this experiment [61; 62].

Table 4. Web Server Event Logging Configuration

Description	Apache Format String	IIS Element Name
Remote IP Address	%a	c-ip
Server Date/Time	%t	date, time
Response size (bytes)	%b	sc-bytes
Response HTTP status	%s	sc-status
Requested URL	%U	cs-uri-stem
Referer	%{Referer}i	cs(Referer)
Query String	%q	cs-uri-query
SSL Version	%{version}c	N/A
SSL Cipher	%{cipher}c	N/A
SSL Error code	%{errcode}c	N/A
SSL Error string	%{errstr}c	N/A

4.2.3. Event Correlation Engine Configuration

This research used the Simple Event Correlator (SEC) as the primary event correlation engine [24]. SEC is a lightweight, open-source, and platform independent tool for rule-based event correlation used

worldwide by organizations in industries such as banking, telecommunications, retail, and software development, with cited benefits including low cost, flexibility, efficiency and ease of configuration. SEC is written in Perl, has a very small footprint (less than 250 KB) and utilizes tools and concepts which are familiar to system and network administrators such as regular expressions, file streams, and named pipes. Configuration files in SEC are plain text files, created and modified with any text editor. These configuration files may contain one or more rules, which are evaluated in the order in which they appear in the file. These rules may be one of nine supported rule types, which together enable signature detection, one or more sliding time windows, calendar events and many more advanced detection abilities.

In the network model used in this experiment, SEC was used to collect events from the Apache and IIS log files. The collected, synthetic, and possibly filtered events were transported back to the central log server using *syslog* protocol. Within SEC, four basic rule types were used: Single, SingleWithSuppress, Suppress and SingleWithThreshold. These rules are grouped into fourteen SEC configuration files - twelve rulesets to detect the twelve detectable use cases, and two rulesets to create certain conditions necessary for implementing the ability to switch between modes. The organization of these rulesets on the filesystem takes advantage of SEC's ability to process multiple configuration files in parallel, and is designed to effectively implement the logging modes previously discussed. On each log-producing system, three directories were created in the same directory as SEC. These directories were named "common," "conf-available" and "conf-enabled." The implementation was inspired by Apache's configuration paradigm: all configuration files reside in the "conf-available" directory, and those which are desired for any particular run of the event correlator are copied into "conf-enabled." This allows SEC to be initialized with the same command each time, by specifying the configuration to load. This configuration also allows the modes of operation to be implemented as combinations of configuration files. To configure a specific mode, a script is executed which copies only the configuration files for the desired behaviors in the "conf-enabled" directory. The script can be remotely executed from a central management console using a secure protocol.

4.3. Use Case Selection

The unit of analysis in the modeling and simulation of an enterprise logging environment is web server generated events. To make the research relevant to real world activities, we used the Open Web Application Security Project (OWASP) Top Ten as a source of realistic use cases [65]. The OWASP is an organization focused on improving the security of web applications worldwide that compiles an annual list of the "Top Ten" web application security risks to help organizations focus their risk mitigation efforts to combat the most prevalent attacks. The OWASP Top Ten 2010 web server risks are shown in Table 5 [65].

Table 5. 2010 Open Web Application Security Project (OWASP) Top Ten [65]

Threat Vector	Description
Injection	Injection flaws, such as SQL, OS, and LDAP injection
Cross Site Scripting (XSS)	Application takes untrusted data and sends it to a web browser without proper validation and escaping
Broken Authentication and Session Management	Application functions related to authentication and

	session management are often not implemented correctly
Insecure Direct Object References	Application developer exposes a reference to an internal implementation object, such as a file, directory, or database key
Cross Site Request Forgery (CSRF)	An application forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application
Security Misconfiguration	Secure configuration defined and deployed for all web applications
Insecure Cryptographic Storage	Applications does not properly protect sensitive data with appropriate encryption or hashing
Failure to Restrict URL Access	Attackers can forge URLs to access hidden pages because URL access rights are unchecked
Insufficient Transport Layer Protection	Application fails to authenticate, encrypt, and protect the confidentiality and integrity of sensitive network traffic
Unvalidated Redirects and Forwards	Application redirect and forward users to other pages and websites using untrusted data to determine the destination pages

From the Top Ten, we generated eight use cases and encoded the corresponding behavior in SAST. We did not use the "Security Misconfiguration" or "Insecure Cryptographic Storage" as use cases because they result from configuration errors that occur during the set up of the web server which would normally be manually checked after each installation.

When developing event correlation rule sets to detect potentially malicious activities, there are two main categories that should be detected: Threat vectors and violations of security policy. Threat vectors are behaviors and events that should never occur in the ICT, such as an SQL injection attack. When these activities occur, it is virtually certain the resource is under attack. The OWASP Top Ten falls largely into this category. In contrast, violations of security policy are often behaviors and events which occur normally but constitute a policy violation when defined thresholds are exceeded. For example, a policy might exist that says that no employee should be accessing a given resource outside of normal work hours (e.g., 6pm - 6am). Access to that resource is a legitimate action, but when the "access time" is outside of the defined interval, that legitimate action becomes a policy violation.

To ensure that both event classes are covered in this research, five additional use cases were added to represent combinations of legitimate activities which, taken together, are regarded as policy violations in the experimental network. These five security policy use cases are shown in Table 6:

Table 6. Security Policy Violations

Policy Violation	Description
Naive Web Crawler	A client IP address is using a web crawler to access all web server content in rapid succession
Delayed Web Crawler	A client IP address is using a web crawler with a uniform delay constant in

	an attempt to mask their access of all web server content
Excessive Downloads	A client IP address downloads more than X bytes in Y minutes
Excessive HTTP Errors	A client IP address accesses an excessive number of non-existent pages

While this research focuses on the analysis of web server logs to detect attacks, there are many other event log sources which may provide value in detection malicious activities. To demonstrate this concept using our distributed event correlation approach, we added the “Injection Sequence” use case shown in Table 7. This use case combines the query log from MySQL and the Apache access log to discern suspicious behavior without the use of a regular expression signature. In addition, the “Insufficient Transport Layer Protection” use case used the Apache error log, which does not have the rigid format of the access log, to detect accesses by the Firefox web browser to an SSL-enabled page with an invalid certificate.

Table 7. Multiple Log Fusion

Policy Violation	Description
Injection Sequence	A client IP address conducts a SQL injection attack detected using both web server logs and database server logs

Overall, the 13 use cases shown in Table 8 were selected to evaluate the ability of different logging modes to detect the malicious behavior.

Table 8. Experimental Use Cases

Use Case	Type
Cross Site Request Forgery (CSRF)	Threat Vector
Injection	Threat Vector
Injection Sequence	Multiple Log Fusion
Insecure Direct Object References	Threat Vector
Excessive Downloads	Policy Violation
Unvalidated Redirects and Forwards	Threat Vector
Failure to Restrict URL Access	Threat Vector
Excessive HTTP Errors	Policy Violation
Broken Authentication and Session Management	Threat Vector
Cross Site Scripting (XSS)	Threat Vector
Naive Web crawler	Policy Violation
Delayed Web crawler	Policy Violation
Insufficient Transport Layer Protection	Threat Vector

4.4. Metrics of Performance

To facilitate the quantitative analysis of the modeling and simulation results for each model of operation in the experimental network, four metrics were identified:

- Use Case Detection Percentage
- Number of False Positives
- Network Bandwidth Consumption
- Log Server Event Correlation Time

The first metric is the use case detection percentage. The use case detection percentage is defined as the percentage of number of use cases detected compared to the total use cases executed during the experiment. In this experiment, each simulation run contains thirteen malicious use cases. A use case detection is deemed to occur if the event correlation rule corresponding to the malicious use case is triggered when the use case occurs. It is expected that this measure will be high (e.g., 100%) for all modes of operation as the event correlation rules are specifically written to detect the malicious use cases. This metric is used to verify that the malicious activities of interest can be detected in each mode of operation.

The second metric is the number of false positives. The number of false positives is defined as the raw number of event rule correlation detection alerts that occur during each 8 hour experimental run that do not match a corresponding malicious use case. Typically, false positives occur as the result of the event correlation rules not being specific enough or because some non-malicious behavior matches the pattern of an activity of interest. This represents one of the real challenges when conducting event correlation: you try to define event correlation rules specific enough to minimize the number of false positives to reduce investigative costs while be general enough to identify potentially malicious activities that require further investigation.

The third metric is the network bandwidth consumption. This metric is defined as the percentage of raw *syslog* event data transported to the total traffic in the network over the duration of the experiment. The total traffic in the modeled network is the sum of the SAST controller traffic, the HTTP web server requests and responses, and the resulting *syslog* traffic. For a fixed set of use cases, this metric is a measure of the burden placed on the network resulting from logging in proportion to the total network traffic. This measurement is taken by monitoring all traffic via a SPAN port on the switch, using *ntop* [65] to collect data and generate summary statistics. This metric is most important for determining the value of this research in terms of reducing network bandwidth consumption.

The fourth metric is the log server event correlation time. This metric is used to quantify the difference in database query execution time at the centralized log server in high accountability environments. This metric is calculated for Mode 4 (e.g., all raw events) and Mode 5 (e.g. all raw events plus synthetic events). The hypothesis is that addition of synthetic events reduces the time required to complete an event correlation database queries.

4.5. Experimental Runs

This section describes the simulation of the network model in detail, including which use cases and logging modes were simulated. It also describes the protocols followed during testing, including how data and statistics are collected and the clearing of log files and databases between runs.

Table 9 shows the details for each of the five experimental simulation runs. These runs test each logging mode with all use cases, to simulate detection of any scenario under normal circumstances. In addition to the use case detection percentage, false positive count, and network bandwidth consumption, which are calculated for each experimental run; the log server event correlation time was determined for two database queries only for the data collected Mode 3 and Mode 4.

Table 9. Experimental Run Detail

Logging Mode	Events Collected
0	No Events Collected
1	Synthetic Events Only
2	Synthetic Events + Filtered Events
3	Raw Events Only
4	Synthetic Events + Raw Events

A SAST scenario containing all of the experimental use cases presented in Table 8 was designed to run over an eight hour time period. The scheduling timeline of the SAST scenario is shown in Table 10. A set of non-malicious web surfing activities was programmed to occur uniformly throughout each experimental run. Within the SAST scenario, there are two significant sources of variation from run to run: 1) the target of each attack, and 2) the execution time for three specific use cases. The target of each attack is chosen randomly by the python script executing the attack, adding an element of unpredictability to the simulation. The three use cases with variable execution times are the Naive web crawler, Delayed web crawler and Insufficient Transport Layer Protection use cases. These cases had to be executed by hand, so the execution times differed slightly. Generally, the web crawlers were executed early in the simulation, while the Insufficient Transport Layer Protection use case was executed later. Note that the variation in the execution times does not impact the performance metrics in this experiment.

Table 10. SAST Scenario Time Line

Task	Scheduled Offset (hours)	Targeted Server
Scenario Start	+0	All
Web Surfing	Uniformly Throughout	All
Cross Site Request Forgery (CSRF)	+0	Development
Injection	+1	Intelligence
Injection Sequence	+1.5	Marketing
Insecure Direct Object References	+2	Marketing
Excessive Downloads	+3	Development
Unvalidated Redirects and Forwards	+4	Intelligence
Failure to Restrict URL Access	+5	Marketing
Excessive HTTP Errors	+6	Development
Broken Authentication and Session Management	+7	Intelligence
Cross Site Scripting (XSS)	+7.5	Marketing
Naive Web crawler	+1 to +2	Marketing
Delayed Web crawler	+3 to +4	Marketing
Insufficient Transport Layer Protection	+2 to +7	Development
Scenario End	+8	All

Prior to beginning a run of the experiment, several tasks must be completed to ensure that the data collected is correct for that run. There are five such tasks: ensuring that data is properly collected from

the previous run, restarting SAST HSAs and resetting *ntop*. When the above steps have been performed, the network model is ready to be prepared for the next experimental run. Three configuration steps are needed to prepare the network for a run. First, the SEC instances on each web server must be configured. This is done remotely through a configuration shell script written for this purpose. Once SEC has been configured, the SAST scenario must be loaded at the controller. Next, the database on the central log server which will store log messages from this run must be created, Kiwi must be configured to write to a new text file, and the log server SEC instance must be configured with the correct source file and destination database. The last step involved in preparing the network for a new run is running the "resetStats.sh" tool on the server running *ntop*. Database queries are run after all data collection has been completed. At this point, the SAST controller is used to initiate the defined SAST scenario.

5. Simulation Results

In this section, we present the results of the network model simulation and present an analysis of the results with respect to the goals of the research. A separate experimental run was conducted for each of the modes of operation. The network model was configured for each of the modes of operation using remote shell scripts which validated the ability to dynamically configure the collection, correlation, and filtering of events.

5.1. Use Case Detection Percentage

Table 11 shows the use case detection percentage, number of false positives, and number of false negatives for each experimental run. As expected, all thirteen use cases were detected for each of the modes of operation resulting in 100% use case detection percentage for each of the modes. A manual verification of the results was performed to assure that detection alerts generated by the event correlation rule correctly matched each use case activity.

Table 11. Use Case Detection Summary

Mode	Use Cases Detection Percentage	False Positives	False Negatives
0	0.0 %	0	0
1	100 %	1	0
2	100 %	2	0
3	100 %	1	0
4	100 %	1	0

5.2. Number of False Positives

Table 11 also reveals that a single false positive alert occurred during the experimental runs for Modes 1, 3, and 4, and two false positives occurred during the experimental run for Mode 2. Further investigation of the raw data collected revealed that in all four modes of operation, the naïve web crawler event correlation detection rule fired once. This occurred because the SAST generated behavior for "Excessive Access Attempts" use case matched the naïve web crawler detection criteria (e.g., the

download of more than 100MB bytes by the same IP address within a 1 hour window). This illustrates the difficulty of coding event correlation rule sets to detect policy violations: the policies must specifically define suspicious or disallowed behavior and that behavior must result in observables that can be encoded in to the detection mechanism. A second false positive occurred during Mode 3 data collection, but this resulted from the delayed web crawler event correlation detection rule which fired once because of the regularity of a download from one specific SAST client IP address. Since the false positives were not the focus of the research, the event correlation rules were not adjusted to reduce the false positive count. In an operational environment, event correlation rules would be continuously refined in an effort to minimize the number of false positives without excluding behavior which may be malicious.

5.3. Network Bandwidth Consumption

During each experimental run, *ntop* was used to monitor all traffic traversing the central switch. The network traffic was composed of SAST controller traffic, HTTP traffic, and the *syslog* traffic which is the focus of this research. Table 12 shows the raw network traffic over each 8 hour experimental run. Note that there are variations in the total SAST and HTTP traffic for each mode due to the stochastic nature of the SAST scenario execution.

Table 12. Raw Network Traffic Composition

Mode	Total Throughput	SAST Traffic	HTTP Traffic	Syslog Traffic
0	2.0 GB	56.3 MB	1.9 GB	0.0 KB
1	2.0 GB	56.3 MB	1.9 GB	7.8 KB
2	2.0 GB	56.2 MB	1.9 GB	31.5 KB
3	2.1 GB	57.2 MB	2.0 GB	6.80 MB
4	2.4 GB	57.3 MB	2.3 GB	6.83 MB

The results for Mode 0 verify the ability to filter all events generated from remote servers so that none are transported to the central log server. In this case, no *syslog* data traversed the network. While the experiment was designed to configure all servers in the same way, a practical application of the methodology would allow the administrator to selectively disable event collecting when there was a need to conserve network bandwidth at all costs.

In Mode 1, only 7.8 KB of *syslog* data traversed the network to the central log server. This data consisted only of the synthetic events generated by the remote event correlation rules executed on each of the servers. Despite the small amount of *syslog* traffic generated, all 13 use cases were detected.

In Mode 2, 31.5 KB of *syslog* data traversed the network to the central log server. This data consisted of the synthetic events plus the raw events which are related to the synthetic events. The ability to pass raw events related to the generated synthetic events is analogous to a motion sensitive camera in its ability to only record activities of interest.

In Mode 3, 6.80 MB of *syslog* traffic traversed the network to the central log server. This data contained all events generated by the system and is representative of the best practice in event log collection and analysis (e.g., the transport of all logs to a central server for event correlation).

In Mode 4, 6.83 MB of *syslog* traffic traversed the network to the central log server. Mode 4 is identical to Mode 3, but adds additional context to the data for the purposes of improving the efficiency

of central event correlation in high accountability environments.

The traffic from *syslog* is an incredibly small percentage of the overall traffic. However, it is important to remember that this research utilizes only a small number of log-producing applications. If this were a full-scale enterprise network, there would be other forms of logs, including workstation, router, firewall, and other application logs. These logs, if added to this experimental network with no further modification, could easily push the percentage of traffic identified as *syslog* significantly higher.

Table 13 shows the average network throughput as well as the network composition of each type of traffic as a percentage of the total throughput. Note that Mode 1 is the most efficient in terms of its ability to minimize the *syslog* traffic while still detecting all of the experimental use cases. In contrast, Mode 4 results in the maximum *syslog* traffic as it contains all raw and synthetic events.

Table 13. Network Traffic Composition Percentages

Mode	Average Network Throughput	SAST Traffic Percentage	HTTP Traffic Percentage	Syslog Traffic Percentage
0	587.7 Kb/s	2.8 %	97.1 %	0.0 %
1	587.7 Kb/s	2.8 %	97.1 %	0.00039 %
2	584.2 Kb/s	2.8 %	97.1 %	0.00158 %
3	622.4 Kb/s	2.3 %	97.3 %	0.283 %
4	714.3 Kb/s	2.7 %	96.9 %	0.324 %

Figure 7 shows the *syslog* traffic as a function of the mode of operation graphed on a logarithmic scale. Visualization of the *syslog* traffic makes the benefits of using Mode 1 or Mode 2 clear whenever it is possible to do so. It also shows that Mode 3 and Mode 4 differ only slightly in the amount of *syslog* traffic.

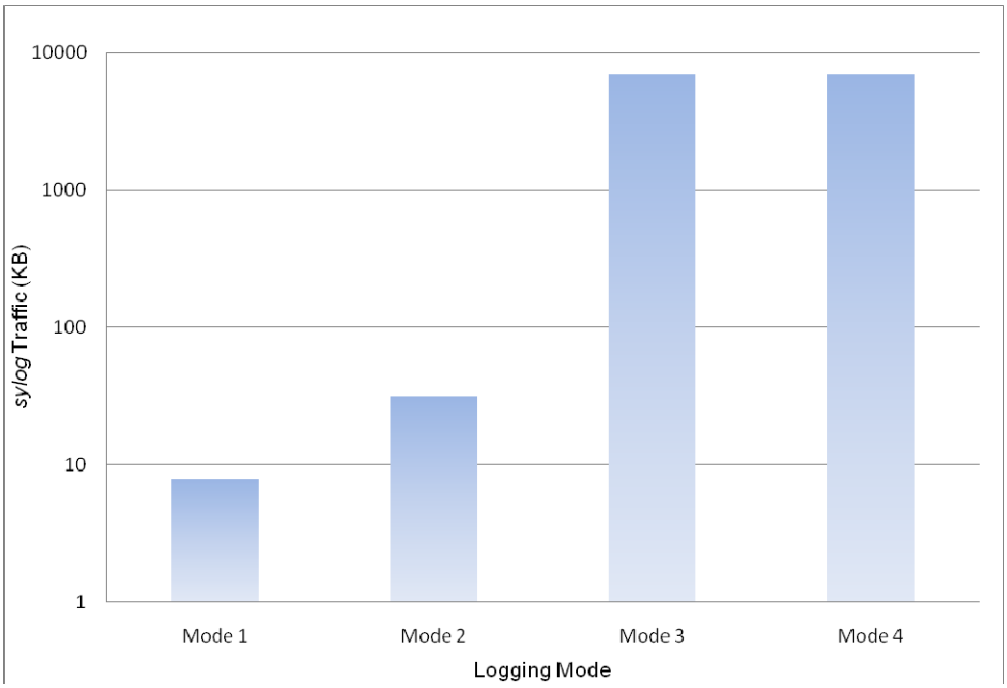

Figure 7. Syslog Traffic as a Function of the Logging Mode

Overall, the flexibility offered by the logging modes becomes apparent in terms of the ability to adjusting the amount of *syslog* traffic. In the best case, where a Mode 4 configuration is compared with a Mode 1 configuration, there is a 99.88% reduction in the amount of *syslog* traffic on the network. These results clearly show the value of a configurable, distributed event correlation infrastructure with regards to network utilization. This is especially true in a low-accountability environment, where no centralized logging is required and thus the smallest possible amount of *syslog* traffic can be sent over the network.

5.4. Log Server Event Correlation Time

In this section, we examine the effect of adding synthetic events to the raw events and the effect of using traditional database normalization on the time required to conduct database driven event correlation at the central log server in high accountability (e.g., Mode 3 and Mode 4) networks. These metrics were collected after all network simulation runs were completed because they only involved post processing of the data stored on the central log server.

5.4.1. Addition of Synthetic Events

One of the theories tested in this research was that the use of distributed event correlation and filtering not only provides benefits in reducing network bandwidth consumption, but also improve the efficiency of centralized event correlation that occurs in high accountability environments. To determine the impact of the addition of synthetic events on the time to complete event correlation at the central log server, two typical tasks were chosen and database queries were written to implement the event correlation necessary to complete the tasks.

The "excessive downloads" use case was selected as a representative example due to the complexity of the case (e.g., implement a sliding window to identify when more than 100MB of data are downloaded within a 1 hour period by any single IP address). The composition of a database query to identify violations of the “excessive downloads” rule differs depending on whether synthetic events are present or not in the data. In the case where no synthetic events are present (e.g., Mode 3), the database query must be constructed to identify all unique IP addresses that accessed the server of interest, identify all web page accesses from each of unique IP address that accessed the server, integrate all of data transfers from this set over a sliding window of the given time period for each unique IP address, and return the IP address of the offending system if the sum of accesses within the sliding window exceeds a specified threshold. In contrast, the database query is much simpler when synthetic events are present (e.g., Mode 4) because the query simply returns the synthetic events to identify the offending IP addresses. In some cases, it may be desirable to identify only the IP address of the offending system(s) and in other cases it may be desirable to identify both the IP addresses of the offending system and other parameters from the related raw events that will aid in an investigation. For this reason, we constructed two different types of database queries to detect violations of the "Excessive Downloads" use case:

1. Return the IP addresses of each offending systems.
2. Return the IP addresses of each offending systems and the related raw events.

For each of the two query types, a separate query was written to process Mode 3 and Mode 4 data. Each of these four unique queries was executed and the results are shown in Table 14.

The results of the first query to return only the IP addresses of offending systems show that adding synthetic events to the event stream resulted in a 1405:1 improvement in query performance. This large performance improvement was expected and is due to the fact that the query workload was allocated to the remote event producing systems. The remote systems processed their own raw event data, identified event sequences which matched the excessive download rule, and when detected inserted the "excessive download" synthetic events into the event stream that was transported back to the central log server. When querying Mode 4 data, the database only had to search for the synthetic events corresponding to the excessive download event correlation rule. In contrast, when querying Mode 3 data the event correlation workload burden was placed solely on central log server which had to process large amounts of data to identify the event streams of interest.

The results of the second query to return the IP addresses of offending systems and related events shows that adding synthetic events to the event stream resulted in a 289:1 improvement in query performance. This performance improvement occurs for the same reasons as Query 1 but is also due to the fact that the addition of synthetic events in the event stream enables the database to more quickly identity the location of the violations (e.g., where "excessive download" event correlation rules fired). When querying Mode 4 data, the database only had to search for "excessive download" synthetic events and if found, return the raw events which caused the production of the synthetic event. In contrast, when querying Mode 3 data the event correlation workload burden, as well as the requirement to return the related events, was placed solely on central log server.

Table 14. Performance Improvement for Excessive Download Event Correlation Queries

Query	Returned Results	Mode 3 – Raw Events	Mode 4 – Raw Events + Synthetic Events	Performance Improvement
1	IP Address Only	120.9s	0.086s	1405:1
2	IP Address and Related Raw Events	289.4s	1.0s	289:1

Our results clearly show that even when related logs are collected, the time required in the context-aware case is several orders of magnitude smaller than in the context-less case. This is an intuitive result - in the context-aware case, the hard work of actually correlating individual log-based events was done in real-time as they happened, allowing the context-aware script to merely query the database for related logs. The context-less case had to do both the task of correlating individual events and the task of searching for related logs. In this experimental high-accountability environment the raw logs are still accessible at the centralized log server, but these results demonstrate that the addition of context in real time through distributed event correlation can foster a remarkable decrease in the amount of time it takes to interact with those raw logs.

It is important to note that the purpose of this experiment was to show experimentally if a performance increase existed. We are planning to further explore this in more detail to identify the factors which determine the magnitude of the performance improvement (e.g., size of the database, complexity of the correlation rules, CPU loading) so that rational tradeoffs can be made when designing a SIEM solution.

5.4.3. Database Normalization

Another theory tested in this research was that the use of database normalization would improve the efficiency of centralized event correlation that occurs in high accountability environments. To determine the impact of database normalization, the data collected for Mode 4 (e.g., raw and synthetic events) was stored in two separate databases: 1) a non-normalized database, and 2) a normalized database. The non-normalized database is the current best practice where all *syslog* events are stored in a single event table within the database using the database table schema shown in Figure 8. The *syslog* data field encapsulated all of the raw and synthetic events within a single table.

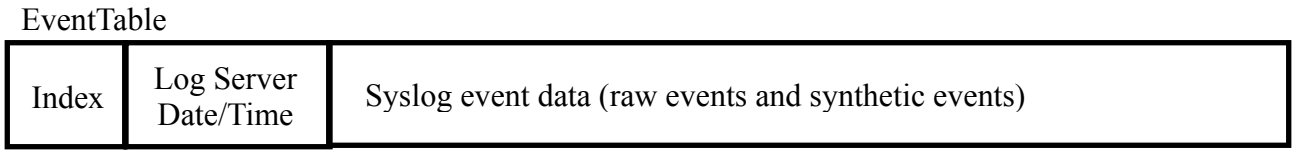

Figure 8. Non-Normalized Log Server Database Table Schema

In contrast, the normalized database stores each incoming *syslog* event into either a raw event table or a synthetic event table using the schema shown in Figure 9. Before a record is inserted into the database, code within the database is used to parse the *syslog* event and store the incoming *syslog* event into the appropriate table. The benefit of this approach is that the synthetic events can be more quickly searched to identity activities of interest.

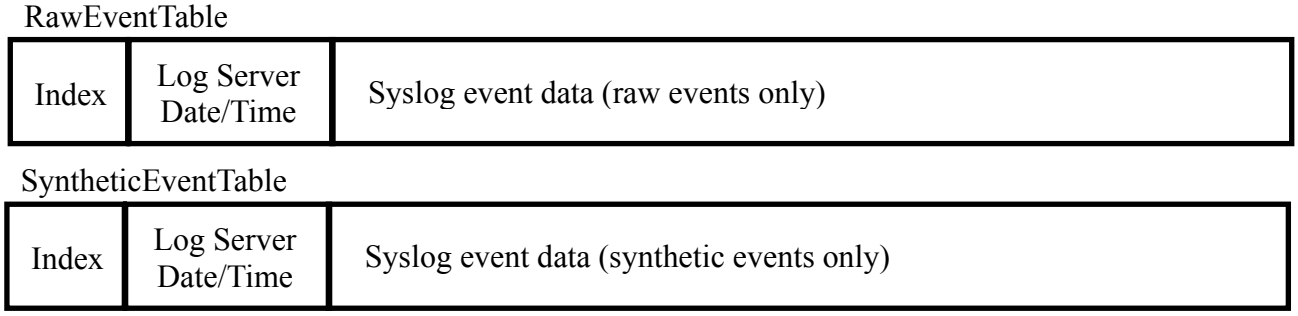

Figure 9. Normalized Log Server Database Table Schema

Table 15 shows the mean and variance of both the normalized and non-normalized runtime of queries for ten queries for each of the use cases using the data collected for Mode 4. All queries were written in Oracle PL/SQL with the exception of the Naive Web Crawler, Excessive Downloads, and Excessive Access Attempts queries. Due to the complex nature of these use cases, these behaviors were detected in the database using Perl scripts. The null hypothesis, *Ho*, is that the mean runtimes are the same; that is to say, normalization of the database has no effect on the runtime of queries. The alternate hypothesis, *Ha*, is that the mean runtimes are statistically different.

Table 15. Query Runtime for Normalized and Non-Normalized Mode 4 Data

Use Case	Mean (normalized)	Mean (non-normalized)	Variance (normalized)	Variance (non-normalized)	p-value
Injection	**0.158**	0.206	0.000307	0.000093	2.50 E-06
XSS	**0.029**	0.079	0.000543	0.000010	7.28 E-05
Authentication	**0.015**	0.505	0.000094	0.025783	4.63 E-06
Object References	**0.123**	0.245	0.000179	0.000027	6.81 E-12
CSRF	**0.031**	0.085	0.000187	0.001783	2.77 E-03
URL Access	**0.055**	0.083	0.000250	0.000157	394. E-06
Redirects	**0.058**	0.073	0.000351	0.000134	48.0 E-03
SSL	**0.054**	0.070	0.000271	0.000000	13.3 E-03
Naïve Web	**325.0**	8063.3	0.888889	64.3333	321. E-09
Excessive Downloads	265.9	**76.6**	0.544444	0.266667	< 2.2E-16
Excessive Access Attempts	324.7	**210.1**	0.455556	0.100000	< 2.2E-16

The p-value is the observed level of significance, which is the smallest value at which *Ho* can be rejected for the collected data. If the p-value is greater than or equal to α, the null hypothesis (*Ho*) is not rejected. If the p-value is smaller than α, the null hypothesis (*Ho*) is rejected. The resulting p-values reveal that database normalization makes a statistically significant difference in the runtime of all of the use case queries. All p-values are less are under the 1% ($\alpha = 0.01$) significance level (e.g., the p-value is less than 0.01) which means the null hypothesis (*Ho*: mean query runtimes are the same) is rejected in favor of the alternative hypothesis (*Ha*: mean query runtimes are statistically significantly different). The use of database normalization reduced the query runtimes for the Injection, XSS, Authentication, Object References, CSRF, URL Access, Redirects, SSL, and Naïve Web Crawler use case; it increased the query runtime for the Excessive Downloads and Excessive Access Attempts use cases.

The results clearly demonstrated the value of normalizing a database for increasing the efficiency of queries made on that database, with an average reduction in query time of 15.31%. This benefit is even clearer when only the OWASP Top Ten use cases are considered which results in an average percent reduction in query time of 46.76%. Figure 10 shows the runtimes of the queries, along with the average runtime. The three insider threat use cases (the Delayed Web Crawler was omitted due to its similarity to the Naive Web Crawler and the long runtimes of each) did not all exhibit such a clear-cut benefit. While the Naive Web Crawler queries showed the most runtime reduction of the set (absolute reduction of over 2 hours or 95.97%), the Excessive Downloads and Excessive Access Attempts queries were actually slower by several minutes in the normalized case. The higher runtimes for the last two queries is likely due to the data structures used to perform the detection - in the non-normalized queries, the sliding time window was implemented using a Perl array, while the normalized queries implemented the window as a series of SQL queries. It was thought that the ability to leverage optimizations in Oracle would cause the normalized queries to perform more efficiently. Clearly, this is not universally the case.

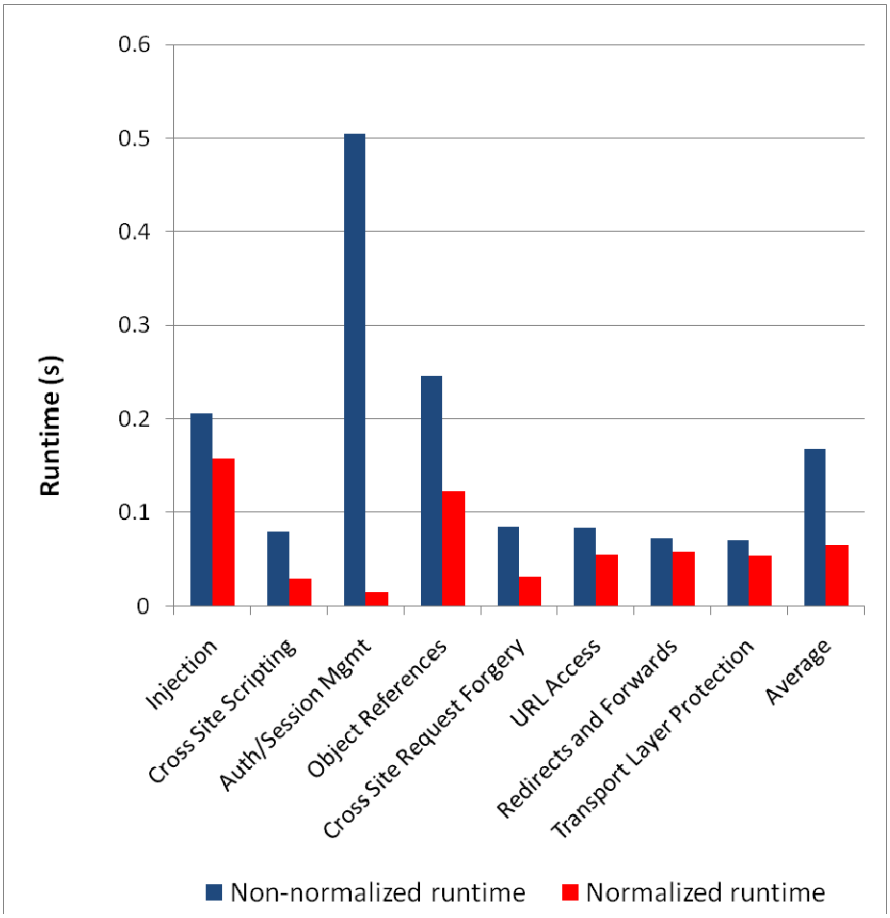

Figure 10. Normalized and Non-normalized queries for OWASP use cases

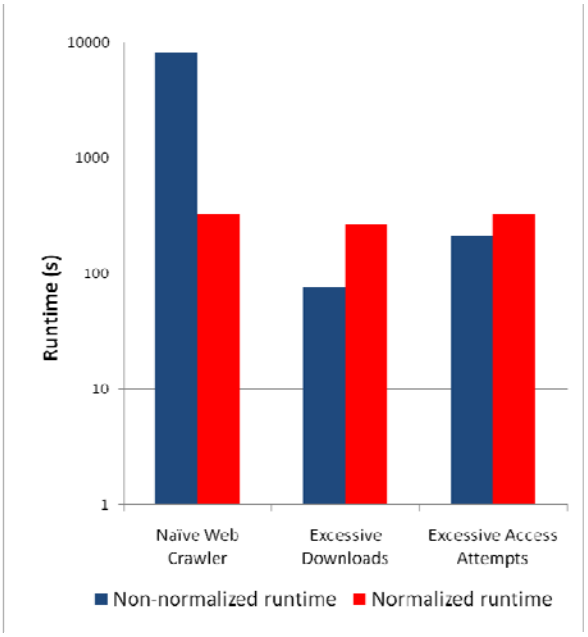

Figure 11. Normalized and Non-normalized queries for insider threat use cases

An examination of the algorithms used in these queries provides insight into these results. Essentially, the only difference between the Naive Web crawler algorithm and the Excessive Access Attempts algorithm is that the query which pulls all relevant records from the database includes an additional qualifier in the latter that only selects those log messages with an HTTP status of "404." Similarly, The Excessive Downloads algorithm requires a single pass through the records, whereas the Naive Web crawler algorithm requires multiple passes for each record. This results in a much lower number of records to consider for the Excessive Access Attempts and Excessive Downloads algorithms. This difference in complexity exposes the underlying mechanisms used to conduct the detection. In the non-normalized cases, a Perl array was used to store the relevant information parsed out of the log messages. These Perl arrays were then used to implement the sliding time windows. In the normalized queries, no such reliance on arrays was necessary, since the sliding time window could be implemented entirely in SQL queries. It seems, therefore, that the implementation of sliding time windows in Perl arrays is more efficient than their implementation in SQL queries. The relatively lower complexity of the Excessive Downloads and Excessive Access Attempts algorithms allows that difference to manifest itself. These results make two points about database normalization of log messages. First, they show that in many cases there can be a distinct efficiency advantage in doing event correlation using a normalized database. Second, these results illustrate the fact that normalization by itself is unlikely to solve the major issues with centralized log management. In the Excessive Downloads and Excessive Access Attempts queries, a more efficient detection implementation actually outweighed any benefit presented by the use of a normalized database.

5.5. Remote Configurability and Log Source Flexibility

In a small experimental network, it is feasible to manually configure each machine in a network. Indeed, a centralized event correlator can be reasonably configured manually, since all event correlation activities happen in one place. It is essential that a distributed event correlation scheme have the capacity for remote configuration to overcome this disadvantage. To provide this capability, a bash script was written which utilized SEC's capability for dynamic configuration via Linux operating system signals. The script takes a logging mode and a reset type as parameters, the reset type being either 'hard' (terminate ongoing event correlation activities) or 'soft' (maintain ongoing event correlation activities and just reload the configuration). The script chooses the appropriate configuration files for the indicated logging mode, and moves them from "conf-available" to "conf-enabled." It then sends either SIGHUP (hard reset) or SIGABRT (soft reset) to SEC, causing it to reload its configuration. We were able to implement and test this capability under a Linux operating system, but the Windows operating system does not have a comparable signaling system and thus an additional process would be required to implement this capability. Despite this limitation, we have proven that a logging environment can be dynamically configured from a remote location making administration of an enterprise wide SIEM solution feasible.

6. Conclusions

In this paper, we presented the design and analysis of a flexible, distributed event correlation system designed to overcome limitations in security event logging and analysis. To demonstrate the utility of the methodology, we modeled and simulated centralized, decentralized, and hybrid log analysis

environments over three accountability levels and compare their performance in terms of configurability, network bandwidth utilization, detection capability and database query efficiency. The results show that when compared to centralized event correlation, dynamically configured distributed event correlation provides increased flexibility, a significant reduction in network traffic in the low-accountability case, and a decrease in database query execution time through context addition in the high-accountability case while detecting all modeled malicious use cases with a low false positive rate. The goal of this research had two components, namely developing a distributed log event correlation methodology and to quantify the value provided by that methodology over a centralized alternative. That two-part goal is met when the chosen metrics for measuring the value of a methodology demonstrate that the distributed methodology does in fact outperform the centralized methodology. The secondary goal of this research was to demonstrate additional advantages of a distributed approach which provide useful, if not quantifiable, value. This goal is met when a plausible implementation of the advantages is demonstrated and shown to provide the anticipated value.

The primary goal of this research was met in part by the measurement of network utilization, showing a best-case reduction in *syslog* traffic of 99.88% between a raw log only and synthetic-event only configuration. The goal was further met by the measurement of query efficiencies, showing that adding context to the database has a dramatic effect in reducing the time necessary to detect suspicious behavior by querying the database.

The secondary goal of the research was met through the implementation of several techniques which take advantage of the distributed architecture to add additional value. Those techniques were remote configuration, inclusion of multiple log sources and analysis of the differences between normalized and non-normalized databases. The remote configuration showed that distributed event correlation architectures can be easily managed remotely, making it much more practical at a larger scale. The inclusion of multiple log sources showed that the methodology can correlate sources with different formats and information, even combining information from multiple sources to detect behavior that is not evident in only one or the other. Lastly, the value of normalizing a database was shown through analysis that revealed that as the complexity of detection algorithms increase, the reduction in query time becomes even more pronounced.

6.1 Future Work

There are several issues in distributed event correlation that need further exploration. First, this research effort focused primarily on the use of web server access logs and, to a lesser extent, web server error logs for detecting malicious activities on a system. We included one use case that utilized the MySQL event log to demonstrate the value of fusing multiple event log sources on the same system. However, further work needs to be conducted to identify tradeoffs in the benefits provided by fusing multiple logs compared to the costs in terms of processing and storage requirements. Second, a study of the expansion of this methodology to include other types of logs, such as router logs, workstation logs, firewall and intrusion detection logs, and other application logs should be conducted to develop an understanding of what is activities are detectable how their inclusion affects event correlation activities. Third, further research is recommended in the area of emerging common log event description standards such as MITRE's Common Event Expression. This area provides great potential for log normalization and event correlation, and parallel research efforts would benefit all concerned communities. Fourth, further research needs to address the presentation and the prioritization of alerts with respect the organizational high value assets. As noted in the introduction, SIEM tools are focused not only on collecting logs and detecting behavior, but also on incident management, reporting and visualization. This research did not meaningfully address the reporting of events once they were generated, or the organizational processes which would be necessary to respond to incidents once they

were reported. Further research in this area would give the log collection and incident detection components of this research additional organizational relevance.

7. Disclaimer

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

8. References

[1] Joint Chiefs of Staff, "Joint Publication 3-13: Information Operations," United States Department of Defense, 13 February 2006.

[2] Libicki, M.C. and Johnson, S.E. (Ed), "Dominant Battlespace Knowledge," National Defense University Press, October, 1995.

[3] Grimaila, M.R., Fortson, L.W., and Sutton, J.L., "Design Considerations for a Cyber Incident Mission Impact Assessment (CIMIA) Process," Proc. of the 2009 International Conference on Security and Management (SAM09), Las Vegas, Nevada, 2009.

[4] Christie, M., "U.S. Mulls Credit Card[7] Type Monitoring To Halt Leaks," Reuters, October 26, 2010, http://in.reuters.com/article/idINIndia-52465720101026 (accessed October 28, 2010).

[5] Kent, K. and Souppaya, M. "Guide to Computer Security Log Management", NIST Special Publication 800-92:, September 2006, http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf (accessed October 23, 2009).

[6] DISA, "Host Based Security System," Defense Information Systems Agency, 2010, http://disa.mil/hbss (accessed October 23, 2010).

[7] Swift, D., "A Practical Application of SIM/SEM/SIEM Automating Threat Identification," Technical report, SANS Institute, December 23, 2006.

[8] Shenk, J., "SANS Annual 2009 Log Management Survey," Technical report, SANS, 2009.

[9] Wilshusen, G. and Powner, D., "CYBERSECURITY: Continued Efforts Are Needed to Protect Information Systems from Evolving Threats", 2009, http://www.gao.gov/new.items/d10230t.pdf (accessed December 10, 2009).

[10] Sah, A., "A New Architecture for Managing Enterprise Log Data". LISA, 121–132, 2002.

[11] Myers, J., Grimaila, M.R., and Mills, R.F., "Towards Insider Threat Detection using Web Server Logs," Proc. of the Cyber Security and Information Intelligence Research Workshop (CSIIRW 2009), Oak Ridge National Laboratory, Oak Ridge, TN, April 13-15, 2009.

[12] Myers, J. Grimaila, M.R., and Mills, R.F., "Insider Threat Detection using Distributed Event Correlation of Web Server Logs," Proceedings of the 2010 International Conference on Information Warfare and Security (ICIW 2010), WPAFB, OH, April 8-9, 2010.

[13] Shenk, J., "Demanding More from Log Management Systems," Technical report, SANS, 2008.

[14] Anderson, J. P., Computer Security Threat Monitoring and Surveillance, James P. Anderson Co., Fort Washington, PA, 1980.

[15] Lonvick, C., "RFC 5424: The Syslog Protocol," Internet Engineering Task Force, Network Working Group, March 2009.

[16] Gerhards, R., "RFC 3164: The BSD syslog Protocol," Internet Engineering Task Force, Network Working Group, August 2001.

[17] Bing, M. and Erickson, C., "Extending UNIX System Logging with SHARP," LISA '00: Proceedings of the 14th USENIX conference on System administration, pp. 101–108, 2000.

[18] Jakobson, G. “The Technology and Practice of Integrated Multi-Agent Event Correlation Systems,” KIMAS, 568–573, 2003.
[19] Bouloutas, A., Hart, G.W., and Schwartz, M., "Simple finite-state fault detectors for communication networks," Communications, IEEE Transactions on, 40(3):477-479,1992.
[20] Cronk, R.N., Callahan, P.H., and Bernstein, L., "Rule-based expert systems for network management and operations: an introduction," Network, IEEE, 2(5): 7-21, September 1988.
[21] Doorenbos, R.B., "Production Matching for Large Learning Systems," PhD Dissertation, CMU-CS-95-113, Department of Computer Science, Carnegie Mellon University, January 31, 1995.
[22] Pouget, F. and Dacier, M., "Alert correlation: Review of the state of the art," Technical Report, EURECOM Research Report RR-03-093, Institut Eurecom, France, November 2003.
[23] Sadoddin, R. and Ghorbani, A., "Alert correlation survey: framework and techniques," In proceeding of the 2006 International Conference on Privacy, Security and Trust (PST '06), 37, 2006.
[24] Vaarandi, R. “SEC - a Lightweight Event Correlation Tool,” Proceedings of the 2002 IEEE Workshop on IP Operations and Management 2002, http://ristov.users.sourceforge.net/publications/sec-ipom02-web.pdf (accessed October 23, 2010).
[25] Yemini, S.A., Kliger, S., Mozes, E., Yemini, Y., and Ohsie, D., “High speed and robust event correlation,” IEEE Communications Magazine, 34(5): 82-90, 1996.
[26] Aamodt, A. and Plaza, E., "Case-based reasoning; Foundational issues, methodological variations, and system approaches," AI Communications, IOS Press, 7(1):39-59, 1994.
[27] Lewis, L., "A case-based reasoning approach to the management of faults in communication networks," In proceedings of the Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '93), pp. 1422-1429, April 1993.
[28] Me, L., "GasSATA: a Genetic Algorithm as an Alternative Tool for Security Audit Trails Analysis," 2000, http://www.rennes.supelec.fr/ren/rd/ssir/publis/raid98_me.pdf (accessed October 14, 2010).
[29] Ben-Gal I., Bayesian Networks, in Ruggeri F., Faltin F. & Kenett R., Encyclopedia of Statistics in Quality & Reliability, Wiley & Sons, 2007.
[30] Lazar, A.A., Wang, W., and Deng, R.H., "Models and algorithms for network fault detection and identification: a review," In Singapore ICCS/ISITA '92, pp. 999-1003, Nov 1992.
[31] Gruschke, B.,"Integrated Event Management: Event Correlation Using Dependency Graphs," Ninth Annual IFIP/IEEE International Workshop on Distributed Systems, Operations, and Management, October 1998.
[32] Davis, R., Shrobe, H., Hamscher, W., Wieckert, K., Shirley, M., and Polit, S., "Diagnosis Based on Description of Structure and Function," Second National Conference on Artificial Intelligence, American Association for Artificial Intelligence, pp. 137-142, 1982.
[33] Hanemann, A. and Sailer, M., "A framework for service quality assurance using event correlation techniques," Telecommunications, Advanced industrial conference on telecommunications/service assurance with partial and intermittent resources conference/e-learning on telecommunications workshop, pp. 428-433, July 2005.
[34] Wietgrefe, H., Tuchs, K., Jobmann, K., Carls, G., Fröhlich, P., Nejdl, W., and Steinfeld , S., "Using Neural Networks for Alarm Correlation in Cellular Phone Networks," In Proc. International Workshop on Applications of Neural Networks in Telecommunications, pp. 248-255, 1997.
[35] Steinder, M. and Sethib, A.S., "A survey of fault localization techniques in computer networks," Science of Computer Programming, 53(2): 165-194, November 2004.
[36] Hasan, M., Viswanathan, R., and Sugla, R., “A Conceptual Framework for Network Management Event Correlation and Filtering Systems," Integrated Network Management, pp. 233–246, 1999.

[37] HP, "ArcSight: Enterprise Security Manager," http://www.arcsight.com/products/products-esm/ (accessed October 23, 2010).
[38] IBM, "Tivoli Netview Documentation," http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/ (accessed October 23, 2010).
[39] HP, "HP OpenView Event Correlation Services (ECS)," http://h20229.www2.hp.com/products/ecs/ (accessed October 22, 2010).
[40] Hansen, S.E. and Atkins, E.T., "Automated system monitoring and notification with swatch," Proceedings of the 7th USENIX conference on System administration, November 1993.
[41] Thompson, K., "An introduction to logsurfer," SysAdmin magazine, 2004, http://www.crypt.gen.nz/papers/logsurfer.html (accessed 22 October 2010).
[42] OSSEC, "Open Source Host-based Intrusion Detection System," http://ossec.net/ (accessed October 24, 2010).
[43] Prelude Technologies, "Prelude: A Universal "Security Information Management" (SIM) system," http://www.prelude-technologies.com/en/development/documentation/index.html (accessed October 28, 2010).
[44] OSSIM, "Open Source Security Information Management," http://www.ossim.net/ (accessed October 28, 2010).
[45] Drools, "Drools - The Business Logic integration Platform," http://www.jboss.org/drools/ (accessed October 25, 2010).
[46] EsperTech, "Event Stream Intelligence: Esper & NEsper," http://esper.codehaus.org (accessed October 29, 2010).
[47] Jakobson, G., Lewis, L., Buford, C.J. and Sherman, C.E., “Battlespace situation analysis: the dynamic CBR approach”. Military Communications Conference (MILCOM 2004), pp. 941–947, 2004.
[48] Jakobson, G., Buford, J., and Lewis, L., "Towards an architecture for reasoning about complex event-based dynamic situations," International Workshop on Distributed Event-based Systems (DEBS 2004), 2004.
[49] Tai, W., O'Sullivan, D., Keeney, J., "Distributed Fault Correlation Scheme using a Semantic Publish/Subscribe system," in Proceedings of The 11th IEEE/IFIP Network Operations and Management Symposium (NOMS 2008), pp. 835-838, April 2008.
[50] Jakobson, G., Weissman, M., Brenner, L., Lafond, C., Matheus, C., “GRACE: building next generation event correlation services,” IEEE/IFIP Network Operations and Management Symposium, pp. 701–714, 2000.
[51] Zach, M., Parker, D., Fallon, L., Unfried, C., Ponce De Leon, M., Van Der Meer, S., Georgalas, N., and Nielsen, J., “CELTIC Initiative Project Madeira: A P2P Approach to Network Management”, 2005, http://www.celtic-madeira.org/publications/summit05_49_madeira.pdf (accessed October 20, 2010).
[52] Sailhan, F. and Bourgeois, J. “Log-based Distributed Intrusion Detection for Hybrid Networks,” Proceedings of the Cyber Security and Information Intelligence Research Workshop (CSIIRW 2008), Oak Ridge National Laboratory, Oak Ridge, TN, May 12-14, 2008.
[53] MITRE, “Common Event Expression: A Standard Log Language for Event Interoperability in Electronic Systems”, The MITRE Corporation, http://cee.mitre.org/ (accessed November 13, 2009).
[54] Symantec, “State of Enterprise Security,” Technical report, Symantec Corporation, 2010.
[55] Baker, W., Hutton, A., Hylender, C.D., Novak, C., Porter, C., Sartin, B., Tippett, P., and Valentine, J.A., “2009 Data Breach Investigations Report,” Technical report, Verizon Business RISK Team, 2009, http://www.verizonbusiness.com/resources/security/reports/2009_databreach_rp.pdf (accessed December 10, 2009).

[56] Richardson, R., “2008 CSI Computer Crime & Security Survey,” The 13th Annual Computer Crime and Security Survey (Computer Security Institute), Technical report, 2008.
[57] Federal Trade Commission, “In the Matter of Genica Corporation, a corporation, and Compgeeks.com, also doing business as Computer Geeks Discount Outlet and Geeks.com, a corporation,” FTC File No. 082 3113, 2009, http://www2.ftc.gov/os/caselist/0823113/index.shtm (accessed June 2, 2010).
[58] Lyon, G., "NMAP: Network Mapper," http://nmap.org/ (accessed October 15, 2010).
[59] Myers, J. Grimaila, M.R., and Mills, R.F., “Adding Value to Log Event Correlation Using Distributed Techniques,” Proceedings of the Cyber Security and Information Intelligence Research Workshop (CSIIRW 2010), Oak Ridge National Laboratory, Oak Ridge, TN, April 21-23, 2010.
[60] GNU, "Wget," http://www.gnu.org/software/wget/ (accessed October 15, 2010).
[61] Apache Software Foundation, “Apache Web Server,” http://www.apache.org/ (accessed October 15, 2010).
[62] IIS, “Microsoft Internet Information Service (IIS),” http://www.iis.net/ (accessed October 15, 2010).
[63] Kiwi Enterprises, “Kiwi Syslog Server,” http://www.kiwisyslog.com/kiwi-syslog-server-overview/ (accessed October 15, 2010).
[64] ntop, “Network top,” http://www.ntop.org/ (accessed October 15, 2010).
[65] OWASP, "OWASP Top 10 2010 RC1," The OWASP Foundation, http://www.owasp.org (accessed October 15, 2010).

9. Author Biographies

Michael R. Grimaila is an Associate Professor of Information Resource Management at the Air Force Institute of Technology. He received his PhD in Computer Engineering from Texas A&M University in 1999, his MSEE from Texas A&M University in 1995, and his BSEE from Texas A&M University in 1993. He holds the Certified Information Security Manager (CISM), the Certified Information Systems Security Professional (CISSP), and the National Security Agency's INFOSEC Assessment Methodology and INFOSEC Evaluation Methodology certifications. Dr. Grimaila’s research interests include cyber incident detection, mission assurance, network management and security, information warfare, and systems engineering. He is a member of the ACM, Eta Kappa Nu, ISACA, ISC^2, ISSA, Tau Beta Pi, and he is a Senior Member of the IEEE.

Justin Myers is an analyst with the Naval Criminal Investigative Service (NCIS). Mr. Myers holds the following degrees: BS, Northwestern University, 1993; MS, Air Force Institute of Technology, 2010.

Robert F. Mills is an Associate Professor of Electrical Engineering at the Air Force Institute of Technology. He received his PhD in Electrical Engineering from the University of Kansas in 1994, his MSEE from AFIT in 1987, and his BSEE from Montana State University in 1983. His research interests are in communication systems, network management and security, information warfare, and systems engineering. He is a member of Eta Kappa Nu and Tau Beta Pi, and is a Senior Member of the IEEE.

Gilbert L. Peterson is an Associate Professor of Computer Science at the Air Force Institute of Technology. He received his PhD in Computer Science from the University of Texas Arlington in 2001, his MS in Computer Science from the University of Texas Arlington in 1998, and a BS in Architecture from the University of Texas Arlington in 1995. His research interests include artificial intelligence,

autonomous robots, multi-robot systems, digital forensics, steganalysis, and machine learning.