Benefits of Centralized Log file Correlation

Robert Rinnan

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2005

The MSc programme in Information Security
is run in cooperation with the Royal Institute
of Technology (KTH) in Stockholm.

Institutt for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Abstract

Network administrators are able to correlate log file entries manually. The problem with this approach is lack of flexibility, it is time consuming, and one doesn't get the general view of the log files in the network. Without this general view it is hard to correlate information between the network components, and events seemingly unessential by them selves, can in reality be a piece of a larger threat.

This thesis analyses some of the existing products, and tries to correspond experience and design principles found in the information and whitepapers on their web pages with research in this area. A lot of claims are found in the literature, but it has not been possible to find any research trying to measure the alleged benefits of the approaches. As we shall see, such systems consist of several parts, and in this thesis the simplest form of prototype that centralizes log files from a small network is developed that gives the ability to browse and search in these log files. The purpose of the prototype is to prove that it is possible with minimal effort to reduce the time consumption doing log file correlation with such a system. In conjunction with this, metrics are developed to measure the difference in time consumed in a network with the prototype and a network without the prototype.

Keywords: log correlation, centralized access, log files, administration, metrics.

Sammendrag

Nettverks administratorer kan korrelere oppføringer i loggfiler manuelt, problemet med slike tilnærminger er manglende grad av fleksibilitet, det er tidkrevende, og man får ikke overblikket over loggfilene i nettverket. Uten dette overblikket er det vanskelig å korrelere informasjon mellom nettverkskomponenter, og hendelser som tilsynelatende er uvesentlige for seg selv, kan i virkeligheten være en bit i et større trusselbilde.

Denne oppgaven ser på noen av de produktene som eksisterer i dag, og forsøker å samsvare erfaringer og design prinsipper fra informasjon og whitepapers som finnes på deres hjemmesider, med forskning på området. Mange påstander finnes i litteraturen, men det har ikke vært mulig å finne forskning som forsøker å måle de påståtte gevinstene med slike tilnærminger. Som vi skal se består ett slikt system av flere deler, og i denne oppgaven er det utviklet den enkleste form for prototyp som sentraliserer loggfiler fra ett lite nettverk, og tilbyr muligheten til å browse og søke i disse loggfilene. Hensikten med prototypen er å påvise at det med minimal innsats er mulig å redusere tidsforbruket ved loggfilkorrelasjon med et slikt system, i den forbindelse er det utviklet metrikker for å måle forskjellen i tidsforbruk i et nettverk med prototypen og et nettverk uten prototypen.

Nøkkelord: loggkorrelasjon, sentralisert tilgang, loggfiler, administrasjon, metrikker.

Preface

This M. Sc. thesis concludes a study of information security at Gjøvik University College.

The choice of topic is the consequence of the fact that I had little knowledge of the log files in my own home network, and when I tried to read the log files it was hard to deduce the information I wanted. What I needed was a way of browsing and searching in the log files from all computers, from a centralized computer. This lead to a literature study that revealed that there exist commercial products capable of centralizing log files, and much more than I could hope to accomplish during the time period of this thesis. It is hard to find references in this area, but there are a lot of unsubstantiated claims on the benefits of such products; this became the framework for my thesis.

The thesis required configuration of several computers, refreshing my programming skills, developing the prototype, developing metrics, and conducting experiments, needless to say that it has been a lot of work. The personal benefits of undertaking a thesis like this though, are the amount of learning, and experience.

I would like to thank Professor Slobodan Petrović that has been my teaching supervisor during this thesis, and the system testers that have spent some of their spare time on conducting the experiments.

Table of Contents

List of Figures

List of Tables

1 Introduction

The topic of the thesis involves technological aspects such as configuration and setup of Linux servers. There has been some software development related to the development of the prototype, and metrics have been developed. The result of the thesis is a report, indicating the benefits of centralized log file correlation.

A short problem description is that the components in a network and applications generate log files. Even if one configures the components and applications to log appropriate information only, the following problems may occur.

- Log file entries seldom appeal to our cognitive domain.
- Log file entries usually have different formats.
- Correlation between log file entries from distributed components can be time consuming, because one has to log in to the different components before one can correlate the entries.
- If an event is identified there is not an easy way to check for possible relations between the event and entries on other network components.
- Problems with some of the existing products are that they are expensive, complex, and some have compatibility problems.
- Lack of control with log files is lack of control with events.

The thesis conducts controlled experiments in order to prove benefits from centralized access to log files. To conduct the experiments a prototype is developed. To develop the prototype a literature study was performed to get insight into existing products, and research in the area.

Research hypothesis: "Centralized log file correlation is beneficial."

The purpose of the study was to test the relation between centralized log file correlation, and benefits of such systems. Variables that might have affected the results are the test contender’s experience, knowledge, how the test system was configured, and the quality of the prototype. The test participants conducted the experiments by connecting to the test system through the Internet. The results reflect time consumption, and are processed in order to generalize and substantiate the hypothesis.

1.1 Motivation

Stakeholders that might benefit from research in this area are those involved in network administration. And small to medium sized companies may not have resources to outsource this part of their system due to economic considerations. Such companies can benefit from calculating saved resources of implementing a system. Benefits of such a rationalization and increased security are points that are easily supported by the management. The relation between expected benefits can be represented like this: centralized access to log files → easily surveyable → Better detection of correlating events → rationalization in time consumption of log file examination/cost effectiveness → quicker response time → increased security.

Matt Bishop [1] identifies some research problems in his book *Computer Security, art and science*. One of the open problems described is to develop a general method that maps a set of system log entries to the corresponding application log entries, to determine what happens at the lower system level. Another research problem is to improve the audit browsing techniques. Some other problems, and benefits are described in the literature, and a short summary is presented here as a basis for motivation. Some of these statements are substantiated through (SIM) product literature. (SIM: Security Information Management is real-time security data monitoring and correlation systems that can detect network breaches or vulnerabilities as (or even before) they happen [4].) The purpose of this thesis is not to look at every aspect of such products, but the literature does provide valid points, and experience.

The following problems are addressed in [2 - 6]:

- Automated security event collection and analysis when managing a security infrastructure requires a lot of people without the tools for automatic security event collection and analysis. This makes it an error-prone process [7].
- There is a lack of business-focused risk management applications that can turn data into meaningful and actionable information; at the same time securing enterprise information assets are rapidly growing in importance and complexity.
- Most data collected from Intrusion Detection Systems (IDS) and security devices may be treated as noise; separating this noise from risks is nearly impossible. This overwhelms IT staffs with data and makes it nearly impossible to correlate the data.
- Audit log data are formatted differently; the communication of logging is standardized but not the content or format.
- Using traditional security analysis methods is outdated, inefficient, time consuming, expensive, and it ties up expert resources that could be engaged in other, more productive operational and/or security activities. There are simply not enough resources to handle the growing number of security incidents. Furthermore, traditional security data analysis methodologies take days or weeks to perform.
- Legislation often requires "adequate" protection [8, 9].

The claimed benefits of centralized log file correlation found in the literature may be summarized as follows:

- Eliminates manual device monitoring [4].
- Resolves security events in real-time from a single console [4].
- Security events that seem unimportant by themselves, when put in context of all other events monitored, can suddenly illuminate a major threat that may be caught too late [6].
- Centralization enables IT staff to integrate real time and historical information, improving visibility into events, trends, and recurrences [6]. With the ability to centralize events also comes the ability to handle enterprise correlation, such systems can see "the big picture" by receiving events from all these locations; it can correlate these actions and detect a pattern that raises an alert, as well as making

an automated response, such as disabling the user ID for a given amount of time [7].

- Event normalization, aggregation, and comprehensive correlation capabilities enable staff to rationalize the massive volume of security events, and accelerate incident identification [6]. Accelerating identification of threats enables the organization to reduce Mean Time to Repair.
- Advanced graphical tools are keys to enabling staff to accelerate identification of security threats, perform fast, accurate analysis, and identify hidden patterns [6].
- Centralized log correlation improves an organization's ability to quickly and accurately identify threats and define appropriate responses [6].
- With a consistent means of handling security events, the organization reduces the risk inherent in lack of training or human error, as well as the time spent trying to figure out how to manage a situation [6].
- By being reactive watching and responding only to IDS alerts, a company may miss unusual traffic patterns that indicate hostile activity through the firewalls. Proactive risk assessment can provide a holistic view of a company's security posture [4].
- GuardedNet [15] claims to have reduced the average time spent in investigating and responding to an attack from 30 minutes to less than 3 minutes using the neuSECURE product [10].
- Consolidation, aggregation and correlation makes it possible for an operator to reduce the number of events that need to be investigated from several thousands or even millions, down to less than 10 [11].

1.2 Research Questions

To conduct experiments it was necessary to develop a prototype. Another approach would be to acquire an existing product, but it was not an option having in mind the prices of such solutions. How the products work is described in the literature and it was necessary to review the layout and design principles of such products to make a similar approach with the prototype. Some of the beneficial claims in the literature are a direct result of the time savings from implementing such products. This lead to the following research questions:

1. Can metrics be defined to measure the time savings of centralized log file correlation systems?
2. Does the prototype have to implement all aspects of a SIM product to measure the time savings of centralized log file correlation systems?

The research questions will not be answered directly through a Chapter or Section, but the questions are answered by creating metrics, and making choices for the prototype.

1.3 Choice of Methods

Qualitative, quantitative, and mixed methods approaches are defined in the literature and the methods in this thesis use John W. Creswell's [39] definitions as a basis for describing the method selection.

- Quantitative: Use postpositive claims to develop knowledge, and make use of strategies such as experiments, surveys, and data collection, that are treated statistically with predetermined instruments.

- Qualitative: Makes knowledge claims based on constructivist perspectives, with intent of developing a theory or pattern, or on advocacy/participatory perspectives, or both. This approach uses narratives, phenomenologies, ethnographies, grounded theory studies, or case studies, as strategies of inquiry. The collections are open-ended emerging data with the intent of developing themes from the data.
- Mixed methods: Use base knowledge claims on pragmatic grounds, strategies of inquiry involve collecting data simultaneously or sequentially, the data collection involves gathering numeric and text information.

This thesis involves a literature study for the previous work Chapter; the purpose of the literature study is to understand the theory, and to get a basis for the development of the prototype. The metrics developed are based on collecting data from several scenarios indicating that this is a quantitative study seeking to substantiate the thesis hypothesis. In addition, the thesis contains qualitative data from the observations of the system testers during the collection of data for the metrics, and the scenarios can be considered case studies. The simultaneous collection of both quantitative and qualitative data indicates the use of a mixed method approach, but because the qualitative data are not further treated, this thesis is a quantitative study, based on the following properties: experimental, seeks to substantiate a hypothesis, collects numerical data, and it is possible to treat the colleted data statistically.

1.4 Structure of the Report

Chapter 2 presents some properties of SIM products, and investigates the research done in this area. The end of the chapter explores some design principles found in the literature. Chapter 3 defines the choices made for the prototype. The description of the test system and software prototype is not of importance for the report but can be reviewed in Appendix A and B. Section 3.2 defines the metrics used in this thesis. To feed the metrics with data some scenarios are developed. These scenarios are commented and can be reviewed in Chapter 4. The experimental work chapter also includes the results of the test. These results are discussed in Chapter 5. The final chapters 6, and 7, conclude the report and present some thoughts about future work.

2 Previous Work

This chapter presents the general layout of how centralized log file correlation, and SIM systems work. This chapter is the result of the literature study, and represents the work that others have done. It was important to gather this information because it enabled the author of this thesis to make choices for the prototype, and it would be difficult to create metrics without the knowledge of how such systems work.

The problem faced today by system administrators are the correlation of Firewall logs, intrusion detection system event logs, operating system event logs, mail system logs, database logs, web server logs, antivirus logs, router/switch logs etc. [10] states that potentially, all these logs can identify a threat, and may receive hundreds or thousands of entries a day. The examination of these logs is often performed by staff short of time and knowledge, and for a company; resources may be a limitation. Despite the resource limitations we want log file correlation because it improves intrusion detection [12]. [13] claims that some of the problems with existing tools today are that they are expensive, and that the most popular free tool is plagued with performance problems. Another problem is the complexity of the tools.

Kevin Mcintyre [10] examines some of the existing products: netForensics [14], GuardedNet neuSECURE [15], and e-Security Management System [16]. The products support a variety of systems and devices, so the first thing to do if one wishes to implement such systems is to find the one that best fit the needs of the environment. [10] also provides some pricing information .

> "The goal of security information management systems is to reduce the total cost of ownership of security devices by reducing the time security professionals spend on threat analysis and incident management, but when these products can cost anywhere from $45,000 to $100,000 and even up to $300,000 or more depending on the number of devices supported, it may be hard for smaller companies to justify their purchase."

This implies that there exist products for log file correlation that probably fulfill the needs if one is willing to pay the price; it also implies the need for an approach suitable for smaller companies without the resources to benefit from commercial products. Reading some of the whitepapers and briefings on products web-pages provides some insight into their architecture, and motivation. There seem to be differences among the products, but they all seem to follow a layout consisting of normalization, aggregation, correlation, and visualization. As an example the netForensics product [17] automatically aggregates events from thousands of disparate sources into a single data format, normalizes the events into 100 event types across nine categories, making it possible to automatically identify events using correlation, generate reports, and perform analysis.

The research done in log correlation systems follows the layout presented for the existing SIM products. Some of the benefits reported are that we can hold users of the systems accountable for their actions [18]. Correlating data among different logs improves intrusion detection systems accuracy [12], and some attacks are not evident when a single log is analyzed [12]. As the reported benefits are approximately the same as with SIM products, the reported problems seem to be similar. The main problem is still that it is too easy to collect an overwhelming amount of audit data [18]. Let's now look at the approaches found in the literature that are relevant to this thesis.

2.1 Centralization

The centralization process may be defined as gathering the log files/entries in one server. The obvious problem when transferring the log files is if someone can tap into the traffic (if it is not encrypted) [19]. If this is the case, an attacker can alter the transmissions. Separate LANs can be used to transfer the log files, reducing the need for encrypted channels. The concept of separated LANs is also described when deploying IDS management networks in the book [20]. When transmitting the data, it is important to make sure the traffic arrives at its destination. This makes transitions such as Syslog RFC 3164 [21] that are based on the UDP protocol unreliable. [22] describes a system that uses agents on the different components, and that relies on SSL to communicate with the log server securely.

2.2 Normalization

The concept of normalization can be defined as follows:

> "Normalization is the process of reducing a complex data structure into its simplest, most stable structure. In general, the process entails the removal of redundant attributes, keys, and relationships from a conceptual data model [41]."

We need normalization because of the different log formats from different sources [2, 6, 23]. Normalization is also identified as event unification. It is the process of dividing log file entries into fields transforming them into a standard format. [19] claims that normalization is the process that enables the correlation tool to extract a datum from the source format of the log file. In addition to the datum, the origin, time, type, and privilege for network connections should be divided into log fields [24]. For commands executed, relevant fields are: program name, amount of execution time, system, user, time of execution, termination, suspension, resumption, and files accessed.

[4] defines normalization as the process of gathering individual security device data, and putting them into context that is easier to understand, and claims that this enables us to map different messages about the same security event to a common alarm ID. [4] claims that normalization alone is a tremendous asset to security teams. Some commercial products use XML to structure the entries and a paper from GuardedNet [5] claims that normalization is a process that breaks an event into its component parts and places them into individual database fields. The use of a database is also supported by Johan Beckers and Jean Paul Ballerini [11].

2.3 Consolidation

The definition of consolidation may be stated as follows:

> "Consolidation is the process that takes data from different systems and entities, and

> possibly disparate formats, and combines and aggregates that information to create a unified view [42].”

When the log entries are centralized the various security events are monitored to determine which events are significant according to a particular attack [10]. The main problem with consolidation to day is that vendors have not come up with a standardized naming scheme for security events [11]. Without such naming of events, it is impossible to analyze the data by each product effectively. To remedy this, work on the Common Vulnerabilities and Exposures (CVE®) Standard has emerged [25]. The purpose is to standardize the names for all publicly known vulnerabilities and security exposures. The CVE repository is downloadable from the CVE web site, where additional information on compatible products (and products working on becoming CVE compliant) can also be found. NIST [26] has made some recommendations of the use of the common vulnerabilities and exposures standard and [27] presents a way to categorize the vulnerabilities in the CVE repository. It also proposes a solution for standardization of the vulnerability categories using a data-clustering algorithm.

2.4 Aggregation

The following definition of aggregation can be found in [43]:

> “Aggregation is a process of grouping distinct data. The aggregated data set has a smaller number of data elements than the input data set.”

Data aggregation [2, 23] organizes normalized data by category, for instance IT systems, applications, etc. [4] states that it is necessary to eliminate redundant or duplicate event data from the security event data stream. This includes refining and optimizing the amount of information that is presented to security analysts. The reduction of data is important because:

> "It is conceivable for a large organization to accumulate upwards of a terabyte of data over a seven-day period [7]."

The reduction of data can be done by examining incoming events from multiple sources for duplicate information and removing redundancies. Then processing rules can filter the arriving data and decide what to keep and what to eliminate [7]. Grouping similar events together and providing answers to how many times an attack happened over a certain time period, enables security experts to group and count these events quickly by port, source IP, target IP, business asset (e.g. Web servers or mail servers.) or other parameters [11] . [19] provides filtering examples such as protocol type, time, IP, and MAC Address.

2.5 Correlation

The definition of the concept of correlation relevant for this thesis can be found in [44]:

> “Correlation is a synonym for association or the relationship between variables.”

Correlation can take aggregated data and analyzes them in real-time, to determine if specific patterns exist. The patterns of similar security events often correspond to specific attacks, such as denial of service, virus, or other forms of attack [4]. The correlation activity [19] can be carried out by one or more engines in order to reconstruct complex events that may be symptomatic of a past or current violation. There are two approaches suitable for correlation. The approaches are described in [12, 19].

1. Top-down approach means starting from an attack to trace back to the point of origin. In network forensics it means starting from a GUI display of the event to get back to the source log with the dual purpose of validating the correlation process used by the engine of the automatic log and event correlation tool and displayed to the security administrator, and seeking out the source logs that will then be used as evidence in the court or for subsequent analysis.
2. Bottom-up approach starts from the source log. Log parsers come to our aid to analyze source logs for a bottom-up correlation. A parser is usually written in a script language like Perl or Python. There are also parsers written in Java to provide a cross-platform approach to network forensics examiners.

Some of the different data correlation techniques found in the literature are listed below:

1. Rule based correlation [2, 23] delivers pre-packaged transformations of event data into different views of the data, and uses predefined rules that apply conditional logic to identify likely attack scenarios by observing a specific series of events within a specified amount of time [28]. In [29] rule based correlation is based on typical attack sequences and security best practices. In rule based event correlation and consolidation systems [10] patterns and definitions of known security threats can be defined by placing them in a database. These can be pre-defined rules provided by vendors, or developed by the system administrator over time. This type of event analysis can be compared to signature files used in virus detection software. The signatures/footprints must be updated on a regular basis to protect the systems.
2. Statistical correlation techniques provide useful insight, especially for time based events. As the name implies [28], it applies statistical algorithms to determine incident severity and then assigns a threat score based on asset value. Statistical correlation looks at network behaviour and identifies threat based on the presence and likely severity of anomalous event patterns [29]. It can detect threats that bypass signatures such as new attacks. It can also identify unknown items, such as the source of attacks that are outside the system. Statistical correlation uses complex algorithms to present users with unique insight into anomalous activity on their network. Statistical correlation systems [10] analyze events over a period of time, and assign weighted values to rate the assets, systems, and attackers; they set a baseline level of normal network activity and look for deviation from these baselines that may indicate an attack. The weighted values assigned to assets are performed manually by many corporations when inventorying. It is important to rank assets in terms of value according to potential loss of value if the system is

attacked. The anomaly based [7] baseline data often depend on collections of data from learning modes, where a database of measured events is built. Current snapshots are then compared with baselines to determine whether anomalies are occurring.

3. Vulnerability correlation [28] takes event data from network intrusion detection systems and correlates it against a database of known vulnerabilities and host vulnerability profiles retuned by vulnerability management scanners. A score is returned for each asset. Vulnerability correlation eliminates false positives by reducing "scanner noise," and helps security personnel determine which attacks are real, and which assets are actually vulnerable to the attack. Examples of scanners to correlate with found in [28, 29] are, Nessus [45], ISS Internet Scanner [46], Harris STAT [47], eEyE Retina [48], nCircle IP360 [49], QualysGuard [50], SPI Dynamics WebInspect [51], and Foundstone [52].
4. Susceptibility Correlation [29] determines the probability of an asset's exposure using all available information about that asset, such as services running, ports open, and the operating system on the machine.
5. [11] defines the following three correlation techniques as impact analysis.
   a. Local correlation is when intrusion detection verifies the consequences of an attack locally. If the local agent logs if the attack is blocked or not it helps in determining if it is necessary to escalate.
   b. OS correlation is when an attack is directed to a wrong operating system; the consequence is that the attack cannot be successful.
   c. Vulnerability correlation checks if an attack is trying to exploit a known vulnerability. If the vulnerability that the attack tries to attack is known and secured it cannot be successful.
6. [11] defines the following three correlation techniques as attack pattern analysis.
   a. Directional correlation determines if an attack is originating from the outside of the organization, from the inside to the outside, or between internal components.
   b. Attack pattern recognition makes use of the fact that some attacks leave patterns. For example, script kiddies will leave the same pattern from an attack due to the use of the same automated scripts they use.
   c. Multi-step attack pattern recognition can be used if an attack spans a long time period. Attack from compromised host is an event correlation technique used to inform the analyst that a component is compromised, and is now launching attacks on the network. In the same way "worm compromised host" is detected. If a component in the network is infected by a worm trying to infect other components on the network, a pattern will immerge due to the pattern a worm leaves. Other possible automated correlations are, "Logon from compromised host", "Logon to compromised host", "Logon failures against multiple hosts", and "Logon from spoofed source".

2.6 Design Principles

Sections 2.6.1 – 2.6.7 examine design principles found in the literature. It was important to obtain this knowledge before work on the prototype commenced.

Together with the knowledge from Sections 2.1 – 2.5 it made the choices for the prototype easier.

2.6.1 List of Objects to Log

[7] describes an event as the smallest element of intrusion detection data, which is an auditable occurrence in the network. We don't want to distinguish between network events and host events [2], because network devices, such as IDS sensors, routers, firewalls, etc. are also hosts. What to log is important and policies define standards describing what constitutes a secure state for each device or parameter being monitored [6]. A policy provides the blueprint for acceptable device implementation and configuration. There are several approaches to decide what to log. Only security relevant events coming from IDS sensors include audit data coming from related network security gateways, or include data from almost any application or system on the network. What to audit requires knowledge of the security policy of the system, what attempt to violate that policy involves, and how such attempts could be detected [30]. What to log involves looking at what commands a user must use to (attempt to) violate the security policy, what system calls must be made, who must issue the commands or system calls and in what order, what objects must be altered, and so forth. In [30] it is claimed that logging all events provides all this information but the problem is to determine relevant information from the data. Forte [19] proposes complete collection as one of the most important requisites, and states that it is important that all packets are captured or else that all losses are minimized and documented. [18] proposes a logging policy with the following requirements:

1. The system should be transparent to the user, i.e. it should behave in the manner to which he has been accustomed.
2. Since system resources are always sparse, as little as possible should be consumed. This means minimizing the use of storage space, processing time, and time spent by the administrator.
3. While meeting the above requirements, sufficient data should be recorded to maximize our chances to detect and trace any, and all, intrusions.

2.6.2 Discussion of Analysis on Demand or a Transfer of Log Files to a Centralized Server

In [5], GuardedNet points out that the volume of data and number of disparate machines in a typical enterprise network can make manual analysis of security data ineffective. Correcting this issue requires automation of the event aggregation process, bringing together data from disparate devices and systems into one central location for correlation. The point is supported by [10] where it is claimed that event correlation systems must be able to provide relevant information in a real-time or near real-time manner through a centralized management console. [11] also states that the first step towards an easier detection/analysis process is to have one console in place where all data is kept. GuardedNet [5] uses an agentless approach - Event Aggregation Module (EAM) that collects, normalizes, filters, encrypts, and forwards the event data securely to the correlation engine. The EAM collects data using standard protocols such as XML, SNMP(v1,2,3), Syslog and SMTP, removing the need to place software agents on each security device. The EAM can also aggregate data by using vendor specific protocols such as CheckPoint's OPSEC, Cisco's Secure POP and RDEP as well as SourceFire's eStreamer protocols. If no standard communications method is supported a light weight module is used. All complex tasks, such as normalization and filtering are still solved on the EAM, thereby minimizing the impact of the agent on host devices. netForensics [23] is also distributable, and uses agents. These agents and analytical engines can reside on one server or on many of them. If multiple engines are installed to process the data and send it to one or multiple databases [10], a master database must be configured to consolidate and correlate the data from the various distributed databases.

[22] claims that the task of keeping audit trail on a centralized server is crucial in large scale computing environments. It is also claimed that it is both inappropriate and time consuming to inspect audit trails on every single host. The proposed solution in [22] is to collect audit trail from the hosts at the time they are generated, and then to consolidate them into a centralized server in a real-time matter. For their solution it is proposed to store entries in a common log format, in a relational database, and at the same time write the entries on non-volatile media as a secure copy.

In [10] it is stated as a must that the systems include capabilities for contacting security administrators while they are off-site, such as paging, email, and remote access. Another point in [10] is that the systems should be able to store and report on historical data. This is necessary if one wants to identify attacks from the same IP address or identify similar types of attacks that have occurred over time.

Keeping the log files on each unit produces less traffic than transferring all events to a monitoring station, but keeping the log files on each network component burdens the components with extra processing. Transferring the log files to the monitoring station requires a potential attacker to break into the monitoring station to conceal his actions. Designating the normalization and aggregation process on the monitoring station removes the need to burden servers with extra processing, and there are also integrity and confidentiality benefits of transferring the files to the monitoring station. The conclusion is that the system should transfer the log files to the monitoring station.

2.6.3 Time

The log must guarantee reasonable certainty as to the date and hour a certain event was registered [19]. [24] supports that all systems must have synchronized clocks to make a comparison of times meaningful. Further [19] states that each report has to be 100% reliable, not only in terms of its integrity in the strict sense (IP, ports, payloads, etc.), but also in terms of the date and time of the event reported. [19] also states that time stamping is essential for two reasons: atomicity of the report and correlation. And the most common problems are the lack of synchronization and the lack of uniformity of the time zones. Reliance is usually placed on Network Time Protocol (NTP) for time synchronization, and the solution in [22] makes use of NTP for time synchronization. Another problem are time zones in distributed architectures on the international scale. This is addressed in [19] and the problem is that some information security managers believe that time zones should be maintained on the physical location of the system of network objects. [19] claims that the plus of using GMT for times zones simplifies management, but it requires the choice to be incorporated into a policy.

2.6.4 Standard Log Format

Since log files exist in different formats, it is practical to develop custom modules for each log file that is a part of the system. These log files also differ in content. To be able to find relations, it must be possible to search in log file entries based on segments of the event that one wants to search for. The solution that stands out is the use of a database to store the entries. This involves organizing each part of the log file entries in databases to make the information available. Searching the databases is a trivial task when the entries are sorted by: username, IP-addresses, time, and other appropriate fields. This approach requires some kind of log washing and Matt Bishop's standard audit trail format [24] might help implementing a system with a database. Most SIM products use XML to structure their data. The advantage of using XML is its flexibility.

If a system security officer wants to trace a connection back through other systems, he must be able to correlate the logs of many different heterogeneous systems through which the attacker may have come [24]. To do this, we need synchronization of time among hosts, a method for correlation of host-specific information, and a standard logging format. This format should be portable enough to pass through the SMTP protocol. [24] suggests that the best representation would involve printable ASCII characters only. The proposed standard log format is presented in Table 1.

Table 1: Standard log format [24].

#S#	start log record
#E#	end log record
#N#	next log record (same as #E#S#)
#	default field separator
#Fc#	change separator to c
#Cc#	change nonprinting delimiter to c
#I#	ignore next field
\	Default nonprinting delimiter

The architecture recommended for generation the log format is to build a filter tool that takes as input the raw log records, and generates as output the standard log format. This filter can reside on the system being monitored (in which case the records are sent in standard format) or on the analysis engine (in which case the logs are sent in native format).

2.6.5 Presenting Data in a Logical Way

After the information is normalized and correlated, it is to be visualized. How the reports are generated and visualized can be approached as in the existing SIM products.

NetForensics [23] uses XML for structuring data, analyzing security event data, handling security incidents, and reporting. The GuardedNet neuSECURE dashboard [5] offers data prioritization, presenting only the top threats and vulnerabilities, which further eliminate extraneous information. The tech brief [32] uses color coding to indicate prioritization of events. And claims that tabular reports provide an increased level of insight over the console by letting the user organize data in specific ways. [33] points out that the problem with log messages is that they are recorded as text. The solution proposed in [33] is to represent the information graphically. The netForensics product [23] displays correlated results on a centralized real-time virtual console with a graphical Java based interface. The visualization solution in [4] is a graphical representation of correlated information in a single, real-time console. [4] reports that effective visualization lets security operators quickly identify and respond to security threats as they occur, before they create problems within the enterprise. [7] reports that reaction to events can be handled in many different ways, starting from an alarm notifying a human that an intrusion is occurring: beep, playing WAV file, sending an e-mail message, or paging the system administrator, writing event details to the local Windows or UNIX event log, or launching programs or scripts to handle the event. [2] supports that events should be reported and states that risk events should be reported close to real-time, and be visualized in such a way that action can be taken.

2.6.6 Storage

Many applications face the problem that sensitive information must be kept in log files on un-trusted machines. If an attacker captures this machine, we would like to guarantee that he will gain nothing or little from the log files, and to limit his ability to corrupt the log files [34]. We should therefore ensure that the logs must be unaltered and not permit any tampering or modification by unauthorized operators as a principle [19]. As a consequence during the collection and examination, the logs should be read only. [35] developed a cryptographic mechanism for securing the contents of an audit log against alteration. Even the logging machine itself is unable to read or undetectably alter previously-written log entries. Reading and verification of log entries are accomplished with the help of a trusted server: a remote machine on the network, or a remote machine to be dialed into, a tamper-resistant token (such as a smart card), or even a remote machine communicated with by means of mailing diskettes back and forth. [35] points out that it would be useful to keep audit logs on a machine with the property that they are available only for writing, not for deletion, updating, or reading. [36] archives network packet data, by capturing network packets, encrypting them, and writing them to long-term CD-ROM storage for later analysis and evidentiary purposes.

[31] examines the massive amounts of data collected by IDSs, and how to build network intrusion detection system using open source software. [31] states that Ethernet packets can grow to 1500 bytes, with a fairly high-speed connection, and this can require large amounts of disk space and significant CPU time to process the data. Their solution is to avoid collecting the whole packet, as it may be sufficient to only collect the packet headers for traffic analysis, and a snap length of the packets for content analysis. It is claimed that this reduces resource consumption and at the same

time captures most violations. Another limiting factor reported is bandwidth; some of the commercial products claim to be able to handle 100 Mb/s. But the maximum manageable throughput will ultimately be affected by the speed of the sensor's processor and its disk regardless of the efficiency of the code. [31] claims that a fast disk array will go a long way in logging large amounts of traffic without loss.

Another point when it comes to the storage is log rotation. [19] proposes a rotation scheme as follows.

> "Once the log has reached the destination machine (Called the Log Machine) it may be temporarily memorized in a pre-assigned slot or input to a database for later consultation. Once the policy-determined disc capacity has been reached, the data are stored in a predetermined location. The original logs are deleted to make room for new files from the source object. This method is known as log rotation [19]."

Instead of deleting the logs they can be stored on non-volatile media. The storage needed will rely on the amount of traffic in the network. Memory will eventually run out if there is no sort of backup scheme or log rotation. How to handle the log rotation, should be part of the logging policy. It is also important as [10] states, that the correlation tool should be able to store and report on historical data, in order to identify attacks from the same IP address or identify similar types of attacks that have occurred over time. This indicates that the logs should be stored for some time on the centralized log file correlation system.

3 Solutions for Centralized Log File Correlation

Vokar et al. [37] claim that conducting a log files analysis typically consists of performing five activities:

1. **Collecting:** software captures and gathers data into a data base for every usage from any user.
2. **Analysis:** software organizes and structures collected log data to produce an analytical report of a site usage, along with graphics.
3. **Visualization:** a software graphically and interactively presents log data resulting from the analysis to facilitate the manipulation of high level data and its understanding.
4. **Critique:** a software attempts to critique the results provided by an analysis and to identify potential utility and usability flaws.
5. **Remediation:** software suggests to repair utility and usability flaws identified in the previous step either in an automated way or a computer-aided manner.

As we have seen from existing products and previous research in the field, an overall design should follow the principle of centralization, normalization, consolidation, correlation, and visualization. [12] makes a valid point when designing such systems, namely an intrusion correlation system should analyze activity, which might initially appear to be unrelated, in order to determine whether it is a part of an intrusion attempt and to characterize that pattern.

This thesis does not intend to implement a complete SIM system, but a lot of the work done in previous research and SIM product presentations, provides an idea of what to focus on. A complete system should implement the following criteria:

1. **Centralization:** The principle is to collect all logs from all devices; the consequence is that it has to incorporate components such as routers, managed switches and other dedicated hardware with logging capabilities. As we have seen, a lot of these devices use different and even special purpose protocols to transfer log data. Another factor related with such hardware is that they often have limited logging capabilities, and/or little log storage memory. For the concept of letting devices connect to the monitoring station to transfer logs, we have to make sure that a connection can be established at all times. We also have to make sure that the log transfers are able to reach the destination for connectionless protocols as well. In a complete logging scheme, we have to not only log network traffic or network IDS events, but also make sure that the server and the user activity are logged. As systems differ in their logging capabilities, it is appropriate to use an agent approach, where the agent connects to the centralized server for transferring log files. If the system is developed for a large network, one has to make sure that the system is scalable as described in the netForensics product [23], or [10].
2. **Normalization:** XML seems to be the preferred choice of structuring the data. The log entries should be structured by all possible fields. The most appropriate fields are dynamic fields such as time, date, src/dst ip and port, user, etc.

3. **Consolidation:** The Common vulnerabilities and exposures standard makes companies and vendors able to compare and analyze data gathered, by naming all publicly known vulnerabilities and exposures. By naming an event, it is possible to determine which events are significant and relate to a particular attack.
4. **Aggregation:** Group similar events together and give answers to how many times an attack happened over a certain time period. (E.g. we don't want thousands of port scan reports; we only need to know how many times we are scanned from a fixed IP.)
5. **Correlation:** A product should be able to use a top-down, and bottom-up approach for correlation. How good we are at incorporating normalization, consolidation, and aggregation will affect our ability to correlate. As we have seen, there exist a lot of correlation techniques, but the point is that we want to be able to receive an alert and trace this event, or to get to the level of a single log file entry, and trace events from that entry. The quality of the correlation step relies on the level of automation of the tool.
6. **Visualization:** The ultimate goal is to reduce complexity, and the visualization part of the tool should be as simple as possible. We need to have a view of all networks and components, and at the same time to be able to look at entries from a single log file from a single component. Designing graphical representations is desirable, but such representations must simplify the view, not make it more complex.
7. **Remediation:** Remediation may not be the responsibility of such products, but can rather be done manually. Reactive responses are certainly possible because the tool has the "big picture", but such capabilities should be limited, because we do not want to block legitimate use. Maybe reactive logging is more appropriate to gather more information in the event of probable malicious activity.

3.1 Prototype Choices

The choice of programming language is Java because it does not involve investments, and because it has simple APIs for GUI, networking, databases, and XML.

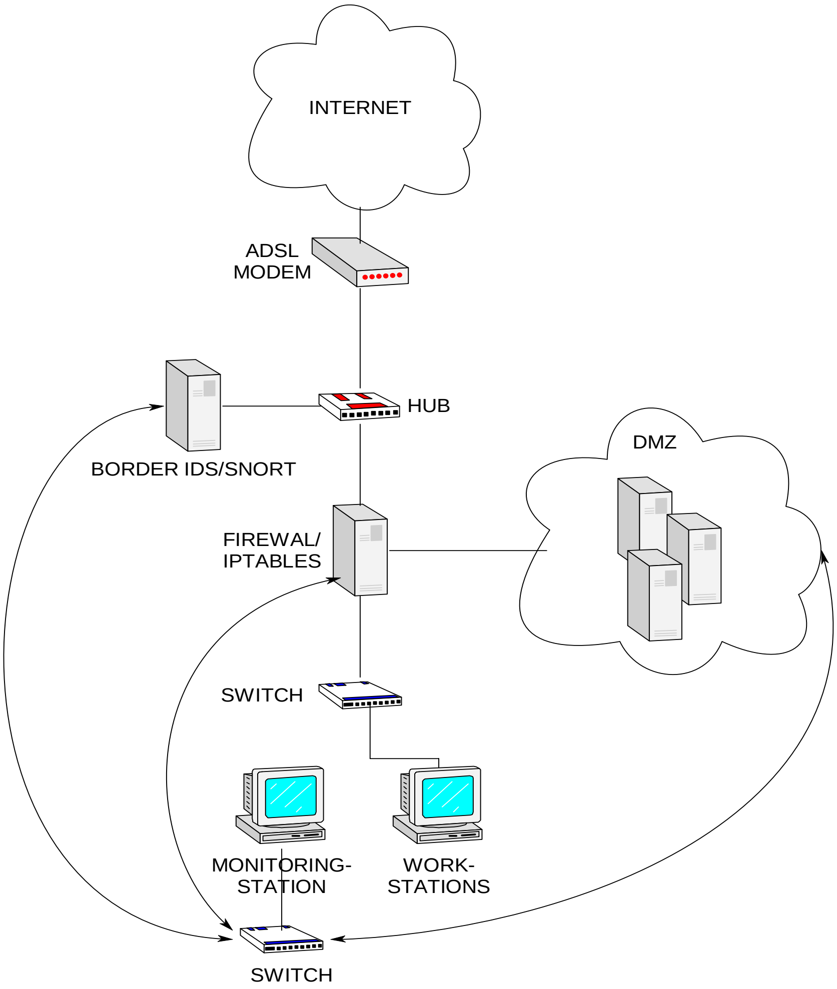


Figure 1: Communication.

As it involves little cost, the approach of a separated log file network is chosen. Figure 1 shows the communication channels. It is important that the monitoring station has access to the log files from all units in the network that generate log files. The IDS between the firewall and ADSL modem has to have its own communication channel since it can't send packets to the network. Units such as routers and switches usually don't have the capacity to store log files by themselves. It would be desirable to develop modules for such units. Such administrable hardware, however, is expensive, and modules for such hardware have not been developed due to economic considerations.

Systems in DMZ and the border IDS will not be able to initiate communications towards the internal network. This implies that the monitoring station has to initiate all communications.

As Figure 1 shows, the DMZ has also its own communication channel. This reduces traffic, and the extra traffic generated by the centralized system won't be logged by the firewall. Giving the firewall a dedicated communication channel makes it possible to completely separate the logging network from the system network. All systems except workstations and the monitoring station run the Fedora Core3 operating system.

Remembering that the purpose of the prototype is to measure the time difference between a system with a centralized log file correlation system, and a system without, and that this thesis seeks to indicate the benefits of centralized log file correlation, the prototype does not focus on normalization, consolidation, or aggregation. We want the prototype to have the properties of browsing and correlating between centralized log files. This also indicates the need for a GUI. The prototype choices are as follows:

- NTP to ensure time consistency.
- Separated logging network.
- Mapping log file directories from network components on the monitoring station. It is a simple solution to just map the directory containing the log files to the monitoring station. This way the log files are up to date when they are searched, and it is easy to export the log file directory with NFS or Samba. This is keeping the log files on each unit solution. If it is desirable to integrity check the log files, some kind of backup solution has to be implemented. Using this approach seems like the easiest and most appropriate solution for the prototype.
- If we want to correlate information manually, we would probably have used *grep* to extract information; this is adequate for small and medium networks too. The most common way to search a file on a Unix system is to use the *grep* command. This command has also been ported to Windows (requires a download). If we have access to the complete log files from the different network components in our network, then using *grep* to deduce information from the log files would be possible, with little effort. This involves trying to configure the different logs to the same format; this is especially true for the timestamp. For example. [Thu Jan 27 17:28:30 2005], Jan 30 15:17:36, 2005/01/31 13:52:23, 27/Jan/2005:15:54:31. As the examples show, the time is mostly represented the same way. The date, however, may become an obstacle unless the date format is configured to be represented the same way in all the log files. For the prototype this is the most appropriate solution, and also helps reduce the complexity of the program, without reducing the outcome of the metrics significantly.
- Log file entries as a minimum should be able to be correlated by: IP-address, user, and a free text argument.
- The application presents the data as shown in Figure 2. One identifies an event one wishes to follow up, and drags a part of the log file entry into the log file one wishes to correlate with.

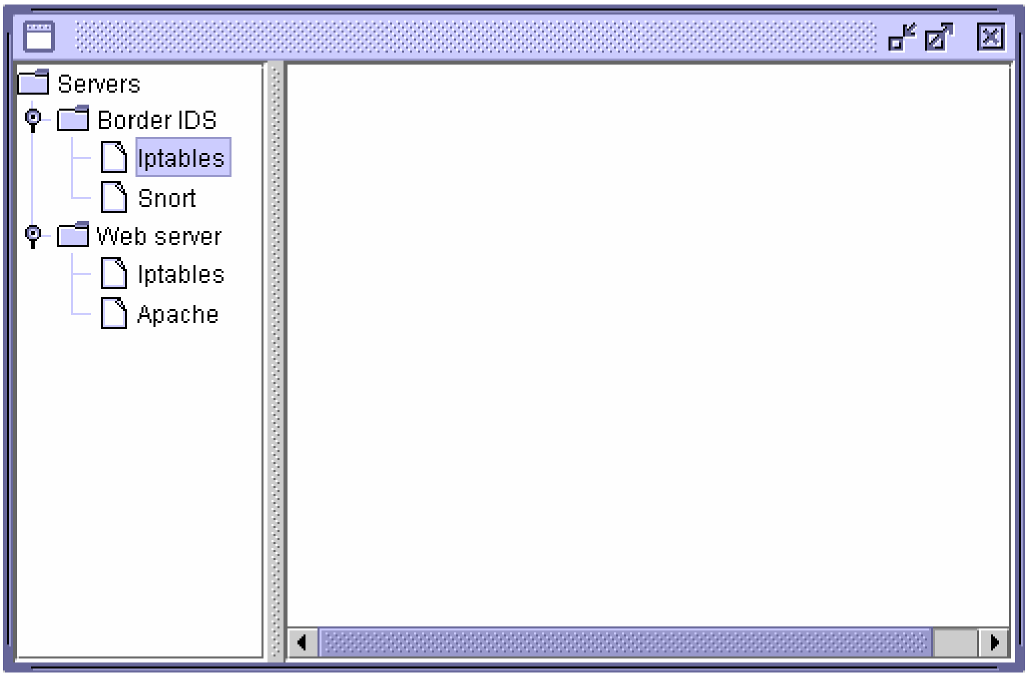


Figure 2: Proposed GUI.

3.1.1 Configurations

Since the prototype only checks log files in the mapped directory, all log files have to be stored in it. In this case the directory is /var/log. How to configure the components is beyond the scope of this thesis. It is, however, important to make sure that logging is enabled. At a user level, one can use process accounting, and history on Linux. The problem with process accounting is that it does not log command arguments. The history command provides the arguments with a time stamp if the HISTTIMEFORMAT (environmental variable) is set. The history file is usually stored in the user's home directories in .bash_history. This can be changed with exporting the HISTFILE. The user history is a possible security risk as it may contain passwords typed on the command line. It is therefore often recommended not to keep this file but to rather write history to /dev/null, or set the HISTFILESIZE=0. One should also be sure to check legal considerations when logging user activity. It may be appropriate to inform the users that all commands are logged. Process accounting is described in [18]. The observations in [18] are rather intuitive, but [18] gives a framework for argumentation for what to log. The Argumentation in Table 2 can be thought of as a logging policy for this thesis.

Table 2: Test system logging argumentation.

Argumentation:			
Component	Dropped traffic.	Allowed traffic.	Solution:
ADSL/Modem	Dropped traffic from the Internet does not enter the network; it will however give an	Traffic passing through the modem from the Internet should	The modem in the test system is restricted by the ISP, so it does not

	indication on attempts to enter the network and port scanning. Summary of the traffic logging should be presented as graphs to detect spikes in traffic. Dropped traffic from the inside will help detect missconfigurations, and malicious traffic from a hacked component.	rather be logged by the IDS or the firewall than the modem. Traffic getting out from the network is of importance. One should monitor this traffic to see if confidential information is leaking out of the network. This traffic should not be logged on the modem as logging of this traffic should occur on the information servers, firewall or IDS.	provide any logging capabilities. As a result this thesis will not provide logging of dropped traffic from the Internet or internal network as it **Should**.
Border IDS	The border IDS's task is to detect unwanted traffic based on predefined rules.		**Must** log all detected events. **Should** log all traffic header information, and a snap length of the packet content, enabling us to perform later analysis [31].
Network firewall Some argumentation is provided by a discussion started by Paul D. Robertson [38].	Logging of dropped traffic from the inside: This traffic does not leave the network, and it is not required to check these logs as often as the logs on traffic that really leaves the network. These logs will however detect missconfigurations	Logging of allowed traffic from the inside: Can detect unused rules that can be deleted. These logs should be checked periodically to see what traffic is leaving the network; does it originate from the	**Should** log dropped traffic from the inside. **Should** log allowed traffic from the inside. **Can** log allowed traffic from the

	on the internal network, if malicious software tries to "call home", or if someone tries to circumvent the security policy. Logging of dropped traffic from the outside: one could use the same argumentation as with the modem, and say that it is not appropriate to log this traffic unless we want statistics on this traffic; it may be suitable to log the traffic for some period to look for attempt to break into the network.	R&D department? Does a disloyal employee send business secrets to the competition? Does malicious software "call home"? Logging of allowed traffic from the outside: The choice depends on the configuration, but it would seem like the logging operations should be performed by the receiving server, rather than the firewall.	outside. **Must** log locally executed commands.
Servers DMZ These servers have IpTables as personal firewalls.	The personal firewalls should log all dropped traffic because such log entries represent malicious events or missconfigurations. The different servers usually provide some sort of logging capability, and what to log is dependent on the needs and server capabilities. For example Apache web server, will log illegal attempts to access the cgi-bin.	Logging of allowed traffic is not logged on the network firewall and should be performed on the servers host firewall, for traceability reasons. The server itself recognizes this as legal traffic and should not log the allowed requests. There is of course a possibility that allowed requests are malicious. This should rather be prevented by keeping server software	**Must** log locally executed commands. **Should** log dropped host firewall traffic. **Should** log allowed host firewall traffic. **Should** log dropped server traffic.

		up-to-date rather than logging.	
Workstations. Workstations usually run Windows, with some kind of host Firewalls for example ZoneAlarm.	The network firewall logs both allowed and dropped traffic from the internal network. As a result, this traffic does not need to be logged on the workstations. Dropped traffic against the workstations should be logged as it will help to detect malicious software trying to connect, and missconfiguration. Windows also provides some logging capabilities.		Windows logging **should** be turned on. Incoming dropped traffic **should** be logged.

There are no single or correct answers to what to log, as the answer will differ between the different network topologies. The arguments in Table 2 may serve as a basis for what to log in the test system. In a live system, the company policy and common sense will determine what to log.

For the prototype, a backup scheme will not be applied. If we want a more permanent scheme we should look to the solutions like those presented in [36]. Most operating systems handle log rotation and keeping the log files on each component makes the storage problem transparent for the testers of the prototype.

3.2 Metrics

In order to conduct the experiments a prototype is developed, and the software is described in Appendix B. Furthermore metrics are required to control the data collection and to handle the data correctly. In short a metric is a measurement standard. In this case the metrics define the framework for the measurement between a system with centralized log file correlation software, and a system without it.

The metrics developed in this thesis are defined by means of the template security metrics guide for information technology from NIST [40].

Table 3: Metric 1, is the implemented system capable of tracking events?

Critical Element	Are there personnel with adequate knowledge of the system to perform the tests, and develop scenarios? Scenarios have to be developed before the test.	
Subordinate Question	Is the system well documented, and are there any dangers of performing the tests with respect to system downtime, or unforeseen events.	
Metric	Percentage of test objectives achieved.	
Purpose	To determine if the centralized log file correlation system is able to track events.	
Implementation Evidence	For each scenario:	
	Scenario description	
	Objective 1	Was the objective discovered by the system tester? ? Yes ? No
	Objective 2	Was the objective discovered by the system tester? ? Yes ? No
	Objective 3	Was the objective discovered by the system tester? ? Yes ? No
	Objective n	Was the objective discovered by the system tester? ? Yes ? No
	# of obtainable objectives	# of observed objectives
Frequency	Upon installation of the centralized log file correlation system.	
Formula	$\mathrm{Re}\,sult = \frac{\#ofObserved Objectives * 100}{\#ofObtainab leObjectives}$	
Data Source	Scenario creator(s) and system tester(s)	
Data Source indicators	This metric should be as close to 100% as possible, to indicate the system's quality.	

Comments: A low score may be the result of inadequate argumentation for what to log in the system. It is not possible to trace events if the system does not log the appropriate information. This metric is to be utilized upon installation of the centralized log file correlation system, and the purpose is to determine if the implemented system operates according to the needs. For this thesis the metric is used to confirm the quality of the prototype. This is important as there is a dependency for conducting further testing with metric 2. The metric is also applicable for a company wanting to determine if a certain product contains requirements specified. Before one can use this metric, scenarios have to be developed. These scenarios should be closely related to the requirements for the centralized log file correlation system. Obtaining scenario objectives could be done by consulting the person(s) involved in system administration, and/or personnel responsible for security incident handling.

Table 4: Metric 2, does centralized log file correlation lead to time savings?

<table>
<tr><td><b>Critical Element</b></td><td colspan="5">It is critical to determine in advance that the implemented system operates properly. Metric 1 should be used to determine this in advance.</td></tr>
<tr><td><b>Subordinate Question</b></td><td colspan="5">Is the system well documented, and are there any dangers of performing the tests with respect to system downtime, or unforeseen events.</td></tr>
<tr><td><b>Metric</b></td><td colspan="5">Percentage of time difference between systems. Affecting parameter: undiscovered objectives.</td></tr>
<tr><td><b>Purpose</b></td><td colspan="5">To determine if centralized log file correlation lead to time savings.</td></tr>
<tr><td rowspan="6"><b>Implementation Evidence</b></td><td></td><td>Time consumption with system. T_1</td><td>Time penalty for undiscovered objectives. T_2</td><td>Time consumption without the system. T_3</td><td>Time penalty
For undiscovered objectives. T_4</td></tr>
<tr><td>Scenario 1</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Scenario 2</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Scenario 3</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Scenario:n</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Total time:</td><td colspan="2">$T_1 + T_2$ = Total1</td><td colspan="2">$T_3 + T_4$ = Total2</td></tr>
<tr><td><b>Frequency</b></td><td colspan="5">When additional servers or equipment with logging capabilities are introduced into the network, or demand analysis.</td></tr>
<tr><td><b>Formula</b></td><td colspan="5">$$100 - \frac{Total1*100}{Total2}$$</td></tr>
<tr><td><b>Data Source</b></td><td colspan="5">Scenario creator(s) and system tester(s).</td></tr>
<tr><td><b>Indicators</b></td><td colspan="5">The goal of this metric is to prove time savings between a system with a centralized log file correlation implementation, and a system without a log file correlation implementation. The system with the lowest time consumption is the better choice. The resulting score gives the time advantage in %. A negative score indicates that there are no benefits from implementing the system.</td></tr>
</table>

Comments: It is expected that implementing such systems lead to time savings. If we get a negative score on this test, we should re-evaluate if the implemented system really works as it should, and look for missconfigurations. The time difference is also expected to increase as the system size increases. The purpose of the metric is to determine if centralized log file correlation systems lead to time savings, and for this thesis the metric was utilized as soon as the quality of the prototype was adequate. The metric states that it should be used when additional servers or equipment with logging capabilities are introduced into the network. This frequency is appropriate when measuring the time gap in different network sizes, and is mostly applicable for experimental work. For live systems it is more appropriate when there is a demand for the results, for example upon choosing a centralized log file correlation system. For this thesis the same scenarios used for metric 1 were used for metric 2. If a positive score is achieved in this metric, it is possible to calculate the resources saved for implementing the centralized log file correlation system; a positive score on this metric is a critical element for metric 3.

Table 5: Metric 3, does centralized log file correlation lead to resource savings?

Critical Element	Relies on the results from Metric 2; it has to be determined in advance that the system really leads to time savings to calculate the resource savings.
Subordinate Question	Variables have to be calculated: • T = Expected time used for log file correlation for a year (hours). • Y = Time saved by using log file correlation system in %, deduced from Table 4 Metric 2 • X = The companies log file analysts hourly salary.
Metric	Amount of money saved by using a log file correlation system.
Purpose	Determine if centralized log file correlation lead to resource savings.
Implementation Evidence	N/A
Frequency	When time savings have been proved.
Formula	T * Y *X
Data Source	Table 4 Metric 2
Indicators	The higher the amount saved the better.

Comments: This metric further treats the results from metric 2, and could arguably be a part of metric 2. Separating it as a metric of its own, however, clarifies the measurements in this thesis. To determine the resource savings, a positive result from metric 2 is required. If we utilize a negative score on this metric, loss of implementing the centralized log file correlation system is calculated. If time savings are calculated from metric 2, resource savings can be calculated if one has the expected time used for log file correlation for a year, and the log file analyst(s) salary. We want a positive score on this metric, and we may calculate the return of investment for implementing a centralized log file correlation system after obtaining these results.

4 Experimental Work

To feed the metrics with data, several scenarios have been developed. The general nature of the metrics makes it easy to develop further scenarios. The scenarios developed in this thesis reflect some easy objectives, and some more advanced objectives. Some of the scenarios refer to the variables Date, Time, and IP. The reason for this is that the system testers conducted the testing at different times, to counter loss of log entries due to log rotation. It is necessary to prepare the system with the users and events before each test. This implies that the objective time will differ between the results, but not the outcome.

Scenarios 6 and 7 had to be tested at the same time, by the system tester with the implemented software, and the system tester without the implemented software.

The scenarios can be reviewed in Table 6 – Table 12. The scenarios described are used for Metric 1, and Metric 2. To improve readability, the observations from the testing with Metric 2, can be reviewed below the scenarios, these are the author's observations during the testing. The results of the experiments are summarized in Sections 4.1 – 4.3.

Table 6: Scenario 1

Scenario description: Scenario provided to the system tester: Find out what activities have been logged for user scenario1user between time1, and time2. Scenario details not disclosed to the system tester: scenario1user logged in via ssh to the DMZ server, created a file, logged in to the firewall via ssh. Then the user exited.		
Objectives:	Test with log file correlation system.	Test without log file correlation system.
Trace what the user has done.	Was the objective discovered by the system tester? ? Yes ? No	Was the objective discovered by the system tester? ? Yes ? No
# of obtainable objectives = 1.	# of observed objectives.	# of observed objectives.

03:09 minutes with the implemented software. Comments: The system tester searched for all entries from all log files for scenario1user. Then he refined the search to only include entries containing the string "session". This made a clean report, but the information about the IP-addresses was missing.

# of scenario objectives obtained = 1.

05:53 minutes without implemented software. Comments: The system tester used recursive grep, and also searched for scenario1user. He then used a text editor to modify the results. Despite his efforts, his report was difficult to read, but contained a trace of what scenario1user had done, and the IP-addresses scenario1user logged in from.

# of scenario objectives obtained = 1

Table 7: Scenario 2.

Scenario description: Scenario provided to the system tester: Discover all users that logged in to the servers on date through ssh. Scenario details not disclosed to the system tester: N/A		
Objectives:	Test with log file correlation system.	Test without log file correlation system.
Who logged in to any of the servers on date?	Was the objective discovered by the system tester? ? Yes ? No	Was the objective discovered by the system tester? ? Yes ? No
# of obtainable objectives = 1.	# of observed objectives.	# of observed objectives.

02:21 minutes with the implemented software. Comments: the system tester started out by searching the DMZ server to figure out what to search for. The system tester ended up using the search arguments {"Mar 13", "Accepted password"}, when all servers were searched the user cleaned the results to create a report.

# of scenario objectives obtained = 1.

13:28 minutes without implemented software. Comments: It seems like it was more difficult to figure out what to search for when the system tester did not have the ability to browse the log files. The system tester ran the following command on all servers: (grep "-r user /var/log/ | grep "Mar 13" | grep "session opened" | grep ssh | less). Both system testers ended up with the same result, but from different log files.

# of scenario objectives obtained = 1.

Table 8: Scenario 3.

Scenario description: Scenario provided to the system tester: The web server does not respond as it should. The first report that the server was not working properly, came at date/time; only the test page appears when contacting the web server with a web browser. Find out everything you can about the event. Scenario details not disclosed to the system tester: index.html is removed from the server by the user root, at 15 Mar 12:00.		
Objectives:	Test with log file correlation system.	Test without log file correlation system.
1) At what time did the event occur?	Was the objective discovered by the system tester? ? Yes ? No	Was the objective discovered by the system tester? ? Yes ? No
2) Who performed the action?	Was the objective discovered by the system tester? ? Yes ? No	Was the objective discovered by the system tester? ? Yes ? No
3) What IP was the user responsible for the event, logged in from?	Was the objective discovered by the system tester? ? Yes ? No	Was the objective discovered by the system tester? ? Yes ? No
# of obtainable objectives = 3.	# of observed objectives.	# of observed objectives.

10:48 minutes with implemented software. Comments: the system tester searched the server for all entries containing html; after browsing the result, he discovered that index.html was removed by the user root. Most of the time the system tester used searching for users logged in at the time the file was removed. The system tester concluded that the file was removed by the user logged in, but could not find any trace of it. The tester also searched the firewall for traffic, but the firewall did not log any ssh traffic at the time of the event. When the test was over, he was shown that the root user had modified the log files, but had failed to modify the history. Discovering this was not an objective, but would have been an explanation for why the objective could not have been discovered.

# of objectives obtained = 2.

23:11 minutes without implemented software. Comments: the system tester discovered that the user root deleted index.html at the correct time after 07:30 minutes. The tester tried to find the last objective, but had difficulties relating the different log files; he ran the same search arguments several times, and had a hard time reading the search results. The system tester tried to find any connection by any user that might have been logged in at the time of the event; at the end the system tester concluded that he could not find out what IP-addresses the root user was logged in from. When the test was over, he was shown that the root user had modified the log files, but had failed to modify the history. Discovering this was not an objective, but would have been an explanation for why the objective could not have been discovered.

# of objectives obtained = 2.

Table 9: Scenario 4.

Scenario description: Scenario provided to the system tester: The user scenario4user can't login to the DMZ server, what's wrong? Scenario details not disclosed to the system tester: User root deleted scenario4user. The objectives are not disclosed to the system testers.		
Objectives:	Test with log file correlation system.	Test without log file correlation system.
1) Who deleted the user?	Was the objective discovered by the system tester? ? Yes ? No	Was the objective discovered by the system tester? ? Yes ? No
2) When was the user deleted?	Was the objective discovered by the system tester? ? Yes ? No	Was the objective discovered by the system tester? ? Yes ? No
3) From which IP address did the user log in?	Was the objective discovered by the system tester? ? Yes ? No	Was the objective discovered by the system tester? ? Yes ? No
# of obtainable objectives = 3.	# of observed objectives.	# of observed objectives.

02:13 minutes with implemented software. Comments: the system tester started by searching for all entries on the DMZ server containing scenario4user. He discovered that root deleted the user, and performed a search to check if someone had escalated rights around the time the user was deleted. The IP-address was discovered by the same search.

# of objectives obtained = 3.

08:32 minutes without implemented software. Comments: the system tester provided the same logic as the system tester with the implemented software, when trying to get the information he wanted. Even though the information he wanted was listed in the result when searching for someone that escalated the rights, he overlooked the entries. He then went on to search one log file at the time, and finally got the results. He then discovered objective 3.

# of objectives obtained = 3.

Table 10: Scenario 5.

<table>
<tr><td colspan="3">Scenario description:
Scenario provided to the system tester: The border IDS seems to have stopped working, Why?
Scenario details not disclosed to the system tester: scenario5user have stopped Snort on the border IDS. The user logged in on the DMZ from the internal network, and logged in to the border IDS. The objectives are not disclosed to the system testers.</td></tr>
<tr><td>Objectives:</td><td>Test with log file correlation system.</td><td>Test without log file correlation system.</td></tr>
<tr><td>1) Who stopped Snort?</td><td>Was the objective discovered by the system tester?
? Yes ? No</td><td>Was the objective discovered by the system tester?
? Yes ? No</td></tr>
<tr><td>2) When was the Snort daemon stopped?</td><td>Was the objective discovered by the system tester?
? Yes ? No</td><td>Was the objective discovered by the system tester?
? Yes ? No</td></tr>
<tr><td>3) From which IP address did the user log in?</td><td>Was the objective discovered by the system tester?
? Yes ? No</td><td>Was the objective discovered by the system tester?
? Yes ? No</td></tr>
<tr><td># of obtainable objectives = 3.</td><td># of observed objectives.</td><td># of observed objectives.</td></tr>
</table>

08:51 minutes with implemented software.

# of objectives obtained =3.

15:42 minutes without implemented software.

# of objectives obtained =3.

Table 11: Scenario 6.

<table>
<tr><td colspan="3">Scenario description:
Scenario provided to the system tester: Have there been any "Failed password" attempts, or "authentication failure" against ssh?
Scenario details not disclosed to the system tester: This is not a fixed scenario; the purpose of the test is to see if both systems detect the same data.</td></tr>
<tr><td>Objectives:</td><td>Test with log file correlation system.</td><td>Test without log file correlation system.</td></tr>
<tr><td>1) Have there been any "Failed password" attempts?</td><td>Was the objective discovered by the system tester?
? Yes ? No</td><td>Was the objective discovered by the system tester?
? Yes ? No</td></tr>
<tr><td>2) Have there been any "authentication failure" attempts?</td><td>Was the objective discovered by the system tester?
? Yes ? No</td><td>Was the objective discovered by the system tester?
? Yes ? No</td></tr>
<tr><td>3) Has anyone tried to access the cgi-bin?</td><td>Was the objective discovered by the system tester?
? Yes ? No</td><td>Was the objective discovered by the system tester?
? Yes ? No</td></tr>
<tr><td># of obtainable objectives = 3.</td><td># of observed objectives.</td><td># of observed objectives.</td></tr>
</table>

02:53 minutes with implemented software. Comments: The system testers made similar reports.

# of objectives obtained = 3.

03:32 minutes without implemented software.

# of objectives obtained = 3.

Table 12: Scenario 7.

Scenario description: Create a report of everything logged about the user zyrus on all systems. Scenario provided to the system tester: Create a report. Scenario details not disclosed to the system tester: This is not a fixed scenario; the purpose of the test is to see if both systems detect the same data.		
Objectives:	Test with log file correlation system.	Test without log file correlation system.
Report all logged activity of user zyrus on all systems.	Was the objective discovered by the system tester? ? Yes ? No	Was the objective discovered by the system tester? ? Yes ? No
# of obtainable objectives = 1.	# of observed objectives.	# of observed objectives.

00:55 minutes with implemented software. Comments: the system tester only made one search for all entries in all log files containing zyrus. The report was not as complementary as the report from the system tester without the implemented software. Time penalty = 1 minute.

# of objectives obtained = 1.

02:23 minutes without implemented software. Comments: the system tester searched all log files for entries containing zyrus. He then added the history for user zyrus from all the servers. The time spent searching was approximately the same for both systems, but the system without the implemented software spent some additional time to organize a report.

# of objectives obtained = 1.

4.1 Results of the Initial Test

This testing is based on Metric 1, and is performed by the system developer. The purpose is to make sure that the objectives in the scenarios are possible to detect with or without the centralized log file correlation system.

Results: The author conducted these tests by himself, and the results indicate if the implemented software operates correctly. In addition to the author, two test subjects have been involved in the testing for Metric 2, further referenced to as system tester 1, and system tester 2, where system tester 1 used the implemented software, and system tester 2 tested without the implemented software. Where system tester 1, and 2 were involved, the author only worked as an observer. 7 scenarios were developed by the author, and the initial testing was also conducted by the author. The results of the initial test can be seen in Table 13, and a summary in Table 14.

Table 13: Initial test.

Scenario:	Result:
Scenario 1	Yes, the implemented software detected that the user logged in to DMZ. Then the user created a file, and logged into the Firewall through SSH. Then the user closed the sessions.
Scenario 2	Yes, the implemented software detected the objective. Once the search arguments leading to the result were discovered, finding the objectives was trivial.
Scenario 3	Yes, the implemented software detected that the user root deleted the file index.html. The time were then deduced from the log files.
Scenario 4	Yes, the software detected that the user root deleted scenario4user at the correct time, and from the correct IP address from the internal network.
Scenario 5	Tracing what happened in this scenario was not as easy as expected, because many users were logged in at the time of the events the author was searching for.
Scenario 6	Quick and easy scenario.
Scenario 7	Yes a report was quickly generated, but it does not say anything about the users' activities after escalating rights.

Comment: Some configuration was required to detect the scenario objectives. This proves the importance of providing argumentation for what to log at the early stage.

Table 14: Initial test summary.

Scenario.	# of observed objectives.
Scenario 1	1
Scenario 2	1
Scenario 3	3
Scenario 4	3
Scenario 5	3
Scenario 6	3
Scenario 7	1
# of observed objectives.	15

$$\mathrm{Re}\,sult = \frac{\#ofObservedObjectives * 100}{\#ofObtainableObjectives} = \frac{15 * 100}{15} = 100\%$$

The goal of the metric was a result as close as possible to 100%. The result show that the system works as expected, and it indicates that the quality of the prototype is adequate to perform the testing for metric 2.

4.2 Results of Measuring Time Savings

Observations during the testing are commented with the scenario descriptions, and a summary of the results is provided in Table 15.

Table 15: Time savings summary.

	Time consumption with the system. T_1	Time penalty for undiscovered objectives. T_2	Time consumption without the system. T_3	Time penalty for undiscovered objectives. T_4
Scenario 1	03:09	None	05:53	None
Scenario 2	02:21	None	13:28	None
Scenario 3	10:48	None	23:11	None
Scenario 4	02:13	None	08:32	None
Scenario 5	08:51	None	15:42	None
Scenario 6	02:53	None	03:32	None
Scenario 7	00:55	01:00	02:23	None
Sum:	31:10	01:00	01:12:41	None

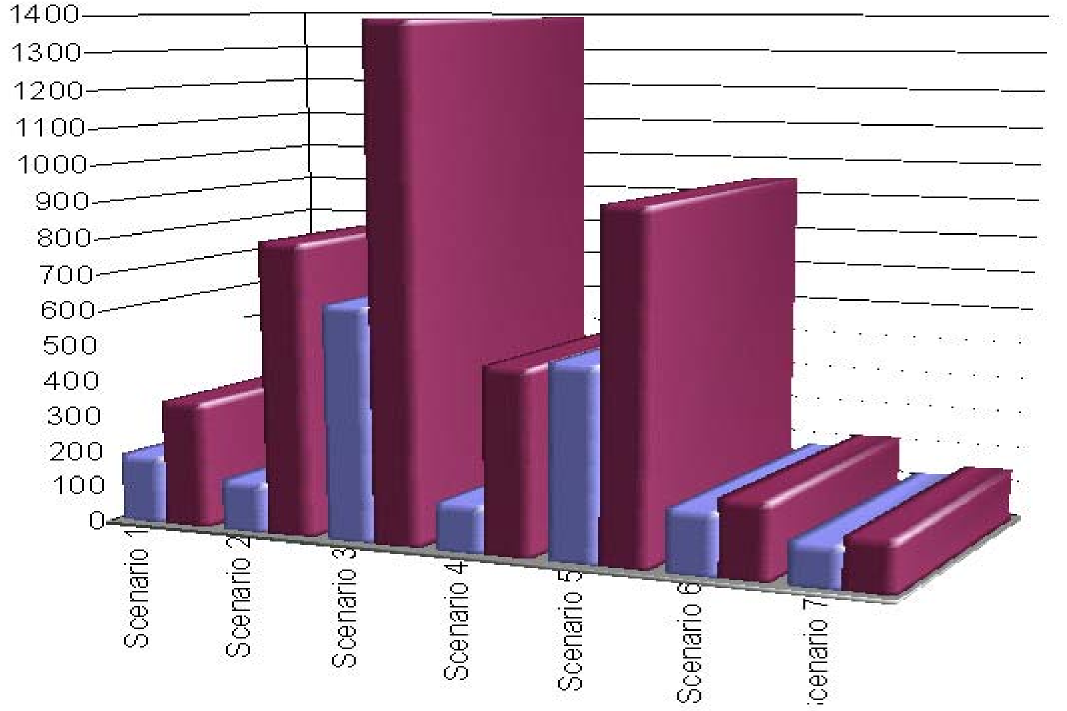


Figure 3: Time difference between the systems in seconds.

Applying Metric 2 on the summary provides the results given in Figure 3. The total time difference in % is shown in the equation below, where Total1 represents the total time consumption with the prototype, and Total2 represent the total time consumption without the prototype.

$$100-\frac{Total1*100}{Total2}=100-\frac{1930*100}{4361}=55.74\%$$

4.3 Results of Measuring Resource Savings

To use this metric we have to predetermine some variables. Let’s look at a company that has one employee that spends one hour a day for log file examination.

Yearly salary = 300.000, - NOK.

Working hours pr. Year = 2000.

X = Hourly salary = 300000/2000 = 150, - NOK.

T = Hours spent on log file examination each year = 2000/8 = 250

Y = 55.74%

T * Y * X = 250 * 55.74 % * 150 = 20902, 5 NOK

In this imaginary example we got a positive result, and time savings of 20902, 5 NOK.

5 Discussion

This chapter discusses the results from the testing. It also provides some critique to the experiments. As discussed below, there are several variables that affect the results. It might be valuable for others interested in conducting similar experiments to review these observations.

5.1 Centralized Log File Correlation and Time Savings

Results from Metric 2 indicate time savings of up to 55.74 %. There are, however, some factors affecting this result.

The system testers reported that if they had conducted similar testing at a later time they would probably spend less time to conduct the tests, because they would know how to search for the right information. Training in searching log files is probably a factor that affects the metric results. To correct this factor, it may have been appropriate to provide the same system testers with the same scenarios at a later time, and observe the effects.

The system tester without the implemented software reported that as the size of the network increased, the time spent searching log files would increase, until a point where searching probably would be impossible, due to the lack of overview of network components and their log files. To check the ratio between the time spent searching log files and network size, one could introduce one server at a time, and run similar scenarios with the same system testers.

How familiar the system testers are with the network they examine, affects the result. To correct this factor, both system testers were provided with the network sketch, and an explanation of the network. This is probably the correct way to perform the test since personnel handling live systems most likely will have extended knowledge on their own network.

How familiar the system testers are with reading log files, affects the result. This factor was considered when picking system testers and both testers have approximately the same experience. The system testers were picked based on their experience with Linux, and they reported that they had used Linux daily for about 5 years; this condition might not be adequate. This is probably a variable that was not given enough attention, and it might be better to make an interview of the system testers based on their experience with reading log files, experience with SIM products, experience with other log file correlation tools, and for experimental work including other operating systems or components, it is advisable to map the experience for those too. A possible method for removing this affecting variable, can be to perform the testing at one point in time, shut down the system, and wait until the system tester have forgotten the scenarios. The first time the test is performed the system tester can test without the prototype. When the system tester has forgotten how he performed the first test, the same system tester can perform the test with the prototype. The danger of conducting the test this way, is that we have to be certain that the system tester has completely forgotten what to search for in the scenarios, if he does remember what he searched for during the first test, he will spend less time to find the scenario objectives during the second testing. This is also the reason why two system testers was involved, because if a system tester finds the scenario objectives with the prototype one day, the same system tester will spend less time finding the objectives without the prototype if this testing occurs the next day.

What the different network components log affects the result. This is because some objectives are impossible to discover if the components do not log the appropriate information.

The quality of the implemented software affects the results. This is due to the level of automation in the software. If the software is developed to automatically trace the events, stated as objectives in the scenarios, then the system tester with the implemented software would spend less time reorganizing the search results. The prototype in this thesis was based on grep and did not use a database or XML to organize the log file entries. This eliminates an affecting parameter because both system testers were limited to the same "search engine". So why did the system testers spend different times during the testing? The observations made during the system testing, provide the answer. A simple GUI showing what log files are available and a simple way of browsing the log files provides overview, and makes it easier to deduce information. The way the search result is presented also makes it easier to deduce the correct information.

The time penalty given in Scenario 7 was appointed because it seemed like an appropriate penalty. For those wanting to perform similar testing, it should be clearly stated in advance how much time such a penalty should represent. This was not stated in advance for this testing and might be considered a shortcoming. The experience obtained from this testing also indicates that the observer should have the opportunity to end a scenario, or intervene in some way. This is because it exist a possibility that the system tester gets stuck for an excessively long time, in such cases the observer could end the testing for the given scenario, and rather give a time penalty. This is just a proposal, but such eventualities should be considered in advance, and the role and actions for the observer should be stated in advance.

The prototype was designed as simple as possible and further developing software that automates search results and structures log file entries for use with faster "search engines", would probably raise the time gap between the systems. With all the affecting parameters, it is impossible to say that centralized log file correlation leads to time savings of 55.74 %, but the number holds true for this simple prototype, applied to this simple network. It is safe to say that centralized log file correlation leads to time savings, and with simple software applied to a large network, time spent searching log files is at least expected to be halved.

As the experience of conducting the testing shows, there are a lot of affecting parameters, and the author recommends incorporating these variables in the metrics to correct for them, if one wants to conduct similar testing. Developing new metrics that incorporates the affecting variables will raise the quality of the measurements. Similar testing should also raise the number of scenarios; because a base of 7 is to small to generalize the results statistically (we want to generalize, not just indicate).

5.2 Centralized Log File Correlation and Resource Savings

The answer to the question if centralized log file correlation leads to resource savings is provided by the positive result from Metric 2, and for the prototype in this thesis the answer is yes. Let's look at our example with the following predetermined parameters, one employee that spends one hour a day for log file examination.

Yearly salary = 300.000, - NOK.

Working hours pr. Year = 2000.

X = Hourly salary = 300000/2000 = 150, - NOK.

T = Hours spent on log file examination each year = 2000/8 = 250

Y = 55.74%

T * Y * X = 250 * 55.74 % * 150 = 20902.5

For this example company, developing similar software to this thesis prototype, and configuring the network components, is estimated to 2 months of work. The cost of development and configuration would pay itself in two years time for this example.

This is not a general result, only an example of how this can be done, and determining if a company has resource savings upon implementing such software, requires collecting the parameters needed for Metric 3, and calculating the resource savings.

6 Conclusions

The thesis proves that the prototype developed gave both time and resource savings, but the observations made make it clear that centralization by itself is not very helpful. Centralization of log files together with some kind of visualization is proven to be very helpful; as it will reduce the time spent searching for chains of events. This thesis only scratches the surface of possible benefits from such systems, but it is a step in the right direction. Let's look at the argument chain of expected benefits introduced in the motivation chapter: centralized access to log files → easily surveyable → better detection of correlating events → rationalization in time consumption of log file examination/cost effectiveness → quicker response time → increased security. The prototype covers centralization, some correlation, and visualization. The observations obtained during the testing lead to the conclusion that the system made the log files easily surveyable. Having the "big picture" provided better detection of correlating events, and the time gap between the test systems, showed a rationalization in time consumption of log file examination that leads to cost reductions. The last points in the argument chain claim that quicker response time leads to increased security. This is a valid assumption, because if security analysts can monitor and response to events in an almost real-time manner, we are moving towards being proactive instead of reactive. Having the "big picture" may also prove to be useful in automated reactions, because events can be correlated, making the reactive response with a higher degree of certainty. For example if a user makes 3 failed login attempts, the user would probably be blocked from further login attempts, but we don't know if this is a legitimate user having problems remembering his password. If a port scan of the network perimeter has been initiated from the same IP-address as the login attempts, we can with a higher degree of certainty conclude that it is a malicious attempt. Together with the claimed benefits found in the literature, there is no doubt of such systems' usefulness.

The motivation of the thesis was to demonstrate that centralized log file correlation is beneficial, and with a simple prototype, time spent in the investigation of log files was more than halved. Taking a step closer to SIM products with aggregation, consolidation, automated correlation, and advanced visualization, would most probably reduce the time spent on log file examination further, but at the cost of complexity. Research and existing products claim a lot of benefits from implementing such systems, indicating the need to further study this area. This thesis demonstrates that such approaches are beneficial with little effort. Because the prototype was not developed to be optimized, it can be concluded that centralized log file correlation lead to time savings because a system developed to be optimized will most likely lead to higher time savings. The results in this thesis enable us to substantiate some of the claims based on time savings found in the literature (it is important to substantiate the claims found in the SIM product literature in a general matter as possible, because these claims probably have a bias towards promoting the products). The results also substantiate the hypothesis that centralized log file correlation is beneficial.

7 Future Work

The author of this thesis will continue to use the implemented software in his own network. It is however a limited prototype, as it only uses the bottom-up approach. Making the prototype able to use the top-down approach requires considerable work, but the author believes that creating a prototype containing more of the elements seen in SIM products, makes it possible to also measure automated correlation techniques. The metrics developed are still suitable as they are both reliable and have validity in the way that they measure what they are supposed to measure and they are repeatable. Without further testing with other system testers though, this thesis cannot determine if the results are reproducible (similar result with other system testers). The metrics are not reproducible unless the test systems are unaltered, because one cannot get the same results if log file sizes are different, the number of network components is different, configurations are different, scenarios are different, or the prototype is different. Keeping these variables static but changing the log file correlation system makes the metrics suitable for those wanting to compare SIM products. Further development of scenarios should have a close relationship to the correlation techniques discussed in the previous work chapter, as it will probably indicate if a product is unable to perform a specific correlation technique.

The terms normalization, consolidation, and aggregation are not always used consistently in the literature, and it is difficult to find implementation details. It is the author's belief that further work that involves generalization of the terms, together with a how-to and points of best practice will be valuable.

How to perform the normalization process is described in the literature, but upon examining different log files it becomes evident that it is not as trivial as expected. A definition of normalization found in database literature is.

> "**Normalization:** A technique for producing a set of relations with desirable properties, given the data requirements of an enterprise [53]."

This is worth remembering when deducing fields from entries. And to begin it would be appropriate to deduce the dynamic fields from entries, and maybe use a code to represent the static content. This will reduce the amount of data, and at the same time it enables us to reproduce the original log file entry.

Methods used for implementation details on the aggregation process are also important. We want to reduce the number of entries as much as we can without loosing our ability to trace events. The aggregation benefit will implicitly come from the normalization research, but may be further enhanced by refining database recommendations. Database literature describes aggregate functions. These functions are COUNT, SUM, AVG, MIN, MAX [53], and can be used on columns in a database. This makes the aggregation process for visualization easy.

This thesis has looked at some of the design principles found in the literature, but it would be useful to make a comparison of the most common SIM products to find the best way of visualizing the data. From the author's point of view the visualization process should start by presenting a complete view of the components in the network(s), and an overview of the top threats for a top-down correlation approach, but it is also important to have the ability to browse individual log file entries for a bottom-up approach, as this approach is also important for debugging purposes. It is also important when creating such tools to incorporate the CVE standard and give the ability to rank network components by risk.

The author has started some initial work to create a prototype using a database, and one of the observations are the ease of using a database after normalizing the log file events. For the next generation prototype the normalization process is performed on each network component, and written directly to the centralized database. Most database vendors support writing to a database over TCP/IP, making it just as easy to update a database remotely as locally.

The author recommends the following approaches to conduct this research further.

1. It would be valuable to create a report trying to either counter or substantiate the findings in this thesis.
2. It would be valuable to create a report that generalizes terms, and in detail motivates implementation details of centralized log file correlation systems. E.g. (how log files should be normalized, motivated by comparison of different log files from different components, and operating systems.), (The structure of the centralized log file database should be like "this…", motivated by how log files should be normalized.), (Ranking of assets and the incorporation of the CVE standard should be done like "this…", because…), (The normalization process should be performed by agents residing on each component, because…)etc.
3. It would be valuable to create a report on how to visualize networks, components, log files, entries/events, correlated events, aggregated events, etc. The findings may be motivated by how good the visualization appeals to our cognitive domain, or time saving for visualization the data from the proposed method.

As a caveat the author would like to point out that creating a centralized log file correlation system by itself adds no research value, since such products already exist. Because of this it is important to early state the purpose of the work.

Bibliography

[1] Bishop, M. 2003. COMPUTER SECURITY, Art and Science. Addison Wesley. 689-721.

[2] AberdeenGroup, Inc. July 2003. An Executive White Paper, Turning it security into effective business risk management. Electronic version found at http://www.netforensics.com/download/Aberdeen_IT_Security_WP.pdf (Visited June.2005).

[3] netForensics inc. 2005. Tech Brief, Incident resolution management unifying the security team to eliminate threats. Electronic version found at http://www.netforensics.com/download/nF_IRM.pdf (Visited June.2005).

[4] netForensics inc. 2002. An Executive Briefing. Security information management, a solution to enterprise security management. Electronic version found at http://www.netforensics.com/download/netForensics_SIM_Strategies.pdf (Visited June.2005).

[5] GuardedNet. Feb 2005. Aggregation Drill Down. Event and log aggregation for real-time monitoring, historical reporting and storage. Electronic version found at http://www.guarded.net/docs/partner_portal/Aggregation%20Drill%20Down%20020205.pdf (Visited June.2005).

[6] netForensics inc. 2005. White Paper. An approach for planning effective security management strategy. Electronic version found at http://www.netforensics.com/download/netForensics_Methodology_WP.pdf (Visited June.2005).

[7] netIQ John Q, W. 2001. White Paper. Security event correlation: Where are we now? Electronic version found at http://download.netiq.com/CMS/Security_Event_Correlation-Where_Are_We_Now.pdf (Visited June.2005).

[8] Fisma solutions. 2003. Comprehensive Information Security is not Optional for Federal Agencies. Electronic version found at http://www.netforensics.com/download/netforensics_fisma_datasheet.pdf (Visited June.2005).

[9] FOR 2000-12-15 nr 1265. Forskrift om behandling av personopplysninger (personopplysningsforskriften). http://www.lovdata.no/all/index.html (Visited June.2005).

[10] McIntyre, K. 2.18.2003. Event correlation systems - the new threat frontline.

[11] Beckers, J. & Ballerini, J. P. 6.June 2003 Computer Fraud & Security. Issue 6. Advanced analysis of intrusion detection logs, 9-12.

[12] Abad, C., Taylor, J., Sengul, C., Yurcik, W., Zhou, Y., & Rowe, K. 2003. Log correlation for intrusion detection: A proof of concept. In ACSAC ’03: Proceedings of the 19th Annual Computer Security Applications Conference, 255, Washington, DC, USA. IEEE Computer Society. Electronic version found at http://www.ncassr.org/projects/sift/papers/ACSAC03.PDF (Visited June.2005).

[13] Dillis, C. D. 27 June, 2003. Ids Event Correlation with SEC - The Simple Event Correlator. Electronic version found at http://www.giac.org/certified_professionals/practicals/gcia/0650.php (Visited

June.2005).

[14] netforensics, inc. web site http://www.netforensics.com/

[15] Guardednet, inc. web site http://www.guarded.net/

[16] e-security, inc. web site http://www.esecurityinc.com/

[17] netForensics inc. 2004. Open security platform: Optimizing security operations. Electronic version found at http://www.netforensics.com/download/nF_products_overview.pdf (Visited June.2005).

[18] Axelsson, S., Lindqvist, U., Gustafson, U., & Jonsson, E. 1998. An approach to UNIX security logging. In Proc. 21st NIST-NCSC National Information Systems Security Conference, 62–75. Electronic version found at http://www.ce.chalmers.se/old/staff/ulfl/pubs/nissc98a.pdf (Visited June.2005).

[19] Forte, D. V. 2004. The "art" of log correlation. Electronic version found at http://www.infosecsa.co.za/proceedings2004/006.pdf (Visited June.2005).

[20] Northcut, S., Zeltser, L., Winters, S., Frederick, K., & Ritchey, R. 2003. Innside Network Perimiter Security. New Riders Publishing.

[21] Rfc3164. 2001. The BSD syslog protocol. web site http://asg.web.cmu.edu/rfc/rfc3164.html (Visited June.2005).

[22] Özgit, A., Dayiogly, B., Anuk, E., Kanbur, I., Alptekin, O., & Ermis, U. 2003. Design of a log server for distributed and large-scale server environments.

[23] AberdeenGroup, I. 2003. Risk and protection spotlight. Electronic version found at http://www.netforensics.com/download/Aberdeen_Spotlight_netForensics.pdf (Visited June.2005).

[24] Bishop, M. 1995. A standard audit trail format. In Proc. 18th NIST-NCSC National Information Systems Security Conference, 136–145.

[25] MITRE Inc. web site http://www.cve.mitre.org/

[26] Mell, P. & Grance, T. 2002. Use of the common vulnerabilities and exposures (cve) vulnerability naming scheme. Electronic version found at http://csrc.nist.gov/publications/nistpubs/800-51/sp800-51.pdf (Visited June.2005).

[27] Li, Y., Venter, H.S., & Eloff, J.H.P. 2004. Categorizing vulnerabilities using data clustering techniques. Electronic version found at http://www.infosecsa.co.za/proceedings2004/063.pdf (Visited June.2005).

[28] netForensics inc. 2005. Tech brief. Rationalizing security events with three dimensions of correlation. Electronic version found at http://www.netforensics.com/download/nF_Comprehensive_Correlation.pdf (Visited June.2005).

[29] GuardedNet. 2005. Four correlation technologies for improved incident recognition and accelerated response. Electronic version found at http://www.guarded.net/docs/partner_portal/Correlation%20Drill%20Down%20022505.pdf (Visited June.2005).

[30] Bishop, M., Wee, C., & Frank, J. 1997. Goal-oriented auditing and logging. Electronic version found at http://seclab.cs.ucdavis.edu/papers/tocs-96.pdf (Visited June.2005).

[31] Del Vecchio, Anthony. 2002. Building network intrusion detection systems using open source software. Electronic version found at http://www.giac.org/certified_professionals/practicals/gsec/0844.php (Visited June.2005).

[32] netForensics inc. 2005. Tech brief. Advanced threat visualization providing information security analysts with better insight for faster response. Electronic version found at http://www.netforensics.com/download/nF_Threat_Visualization.pdf (Visited June.2005).

[33] Takada, T. & Koike, H. 2002. Tudumi: Information visualization system for monitoring and auditing computer logs. Electronic version found at http://www.vogue.is.uec.ac.jp/~koike/papers/tudumi/tudumiIV2002.pdf (Visited June.2005).

[34] Schneier, B. & Kelsey, J. May.1999. Secure audit logs to support computer forensics. ACM Transactions on Information and System Security. Col.2 No.2. 159–176. Electronic version found at http://www.schneier.com/paper-auditlogs.pdf (Visited June.2005).

[35] Kelsey, J. & Schneier, B. 1999 Minimizing bandwidth for remote access to cryptographically protected audit logs. In proc. Of Recent Advances in Intrusion Detection. Electronic version found at http://www.raid-symposium.org/raid99/PAPERS/Kelsey.pdf (Visited June.2005).

[36] Antonelli, C., Undy, M., & Honeyman, P. April.1999. The packet vault: Secure storage of nework data. In proc. Of the 1st Workshop on Intrusion Detection and Network Monitoring. Electronic version found at http://www.citi.umich.edu/techreports/reports/citi-tr-98-5.pdf (Visited June.2005).

[37] Vokar, S., Mariage, C., & Vanderdonckt, J. 2001. Log files analysis to measure the utility of an intranet. Electronic version found at http://www.isys.ucl.ac.be/bchi/publications/2001/Vokar-HCIInt2001.pdf (Visited June.2005).

[38] Paul D. Robertson, with others. Web site http://honor.icsalabs.com/pipermail/firewall-wizards/2004-September/thread.html (Visited June.2005).

[39] W.Creswell, J. 2003. Research Design Qualitative, Quantitative, and Mixed Methods Approaches, Second Edition. SAGE, Publications.

[40] Swanson, M., Baro, N., Sabato, J., Hash, J., & Graffo.l, L. July 2003. Nist special publication 800-55 security metrics guide for information technology systems.

[41] Definition of Normalization. Web site http://www.dmreview.com/resources/glossary.cfm?keywordId=N (Visited June.2005).

[42] Definition of consolidation. Web site http://dmreview.com/resources/glossary.cfm?keywordId=C (Visited June.2005).

[43] Definition of aggregation. Web site http://www.orafaq.com/glossary/faqglosa.htm (Visited June.2005).

[44] Definition of correlation. Web site http://www.ojp.usdoj.gov/BJA/evaluation/glossary/glossary_c.htm (Visited June.2005).

[45] Nessus. Web site http://www.nessus.org/

[46] ISS Internet Scanner. Web site http://www.iss.net/products_services/enterprise_protection/vulnerability_assessment/scanner_internet.php

[47] Harris STAT. Web site http://www.stat.harris.com/index.asp

[48] eEyE Retina. Web site http://www.eeye.com/html/products/Retina/

[49] nCircle IP363. Web site http://www.ncircle.com/index.php?s=products_ip360

[50] QualysGuard. Web site http://www.qualys.com/

[51] SPI Dynamics WebInspect. Web site http://www.spidynamics.com/

[52] Foundstone. Web site http://www.foundstone.com/

[53] Connolly, T., & Begg, C. 2002. Database Systems, A Practical Approach to Design, Implementation, and Management. Addison Wesley. 110-155, 375-414.

Appendix A – Test System Description

The prototype runs WindowsXP, and maps the /var/log directories from the border IDS, firewall, and a server from the DMZ zone. Adding a server to the prototype is now a trivial task.

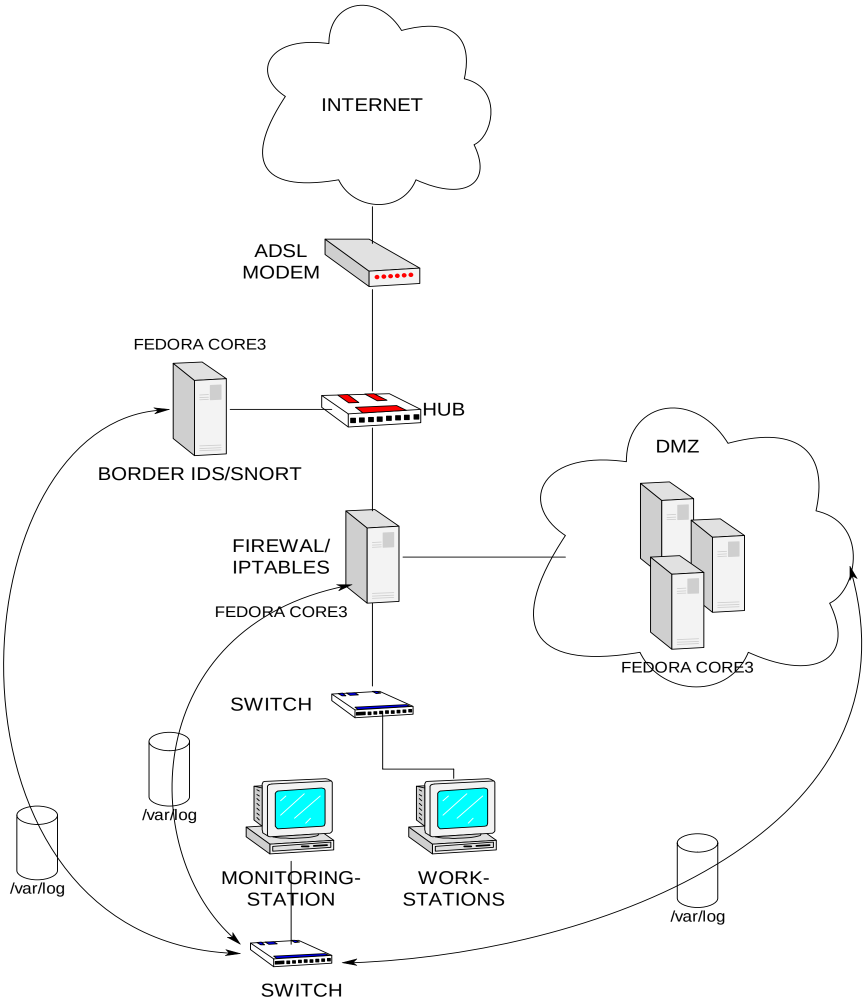
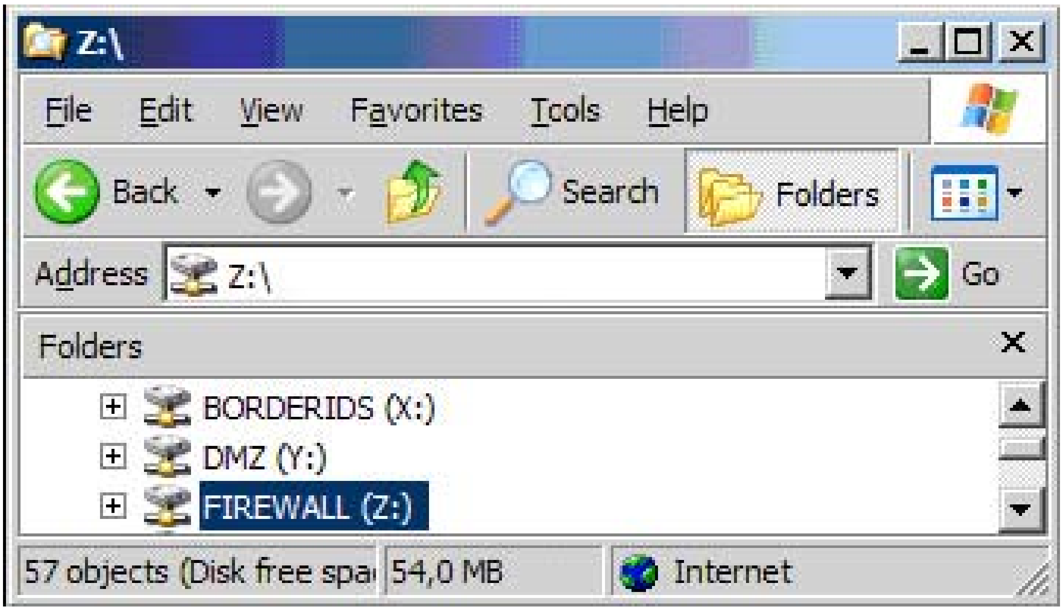


Figure 4: Concept.

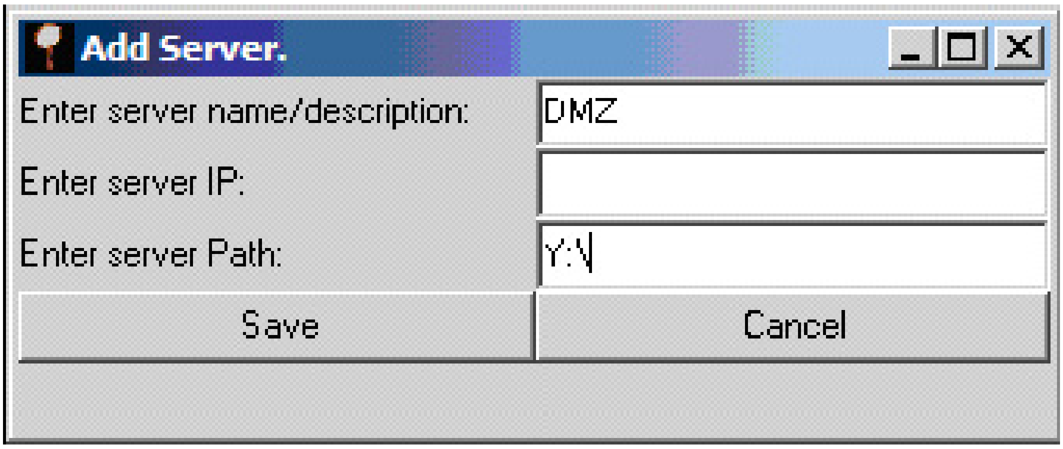


Address: Z:\

- BORDERIDS (X:)
- DMZ (Y:)
- FIREWALL (Z:)

57 objects (Disk free spa 54,0 MB Internet

Figure 5: This figure shows the directories mapped in Windows Explorer.



Enter server name/description:	DMZ
Enter server IP:	
Enter server Path:	Y:\

Figure 6: Adding a server.

This approach makes it easy to extend the network with more computers or servers; all one has to do to make a component ready, is to share the directory containing the log files, and map that directory on the monitoring station. It may seam risky to use a samba server on the firewall, but using a dedicated interface for the log file network, making the server only accessible by the monitoring station, and locking it down with for instance Iptables, is a small price to pay for having control with the logs.

Each interface on the firewall represents a subnet; a brief summary of the addresses is listed in Table 16.

Table 16: Firewall subnets.

Device:	Configuration:
Router/ modem:	Internal ip: 10.0.0.1/24 Gateway for firewall External dynamic ip.
Firewall:	eth0: 10.0.0.2/24 eth1: 10.0.1.1/24 Gateway for DMZ zone. eht2: 10.0.2.1/24 Gateway for internal network. eth3: 10.0.3.1/24 Gateway for monitoring network. Lets the monitoring station connect to the Samba server via this interface.
DMZ:	Consists of only one server. eth0: 10.0.1.2/24 eth1: 10.0.3.4/24 Lets the monitoring station connect to the samba server via this interface.
Border IDS:	eth0: 10.0.0.3 this interface is connected between the router/modem and the firewall as a passive network tap. eth1: 10.0.3.3/24 Lets the monitoring station connect to the samba server via this interface.
Monitoring station:	10.0.3.2/24 connects to Firewall, DMZ, and Border IDS on network 10.0.3.0/24

Appendix B – Software Description

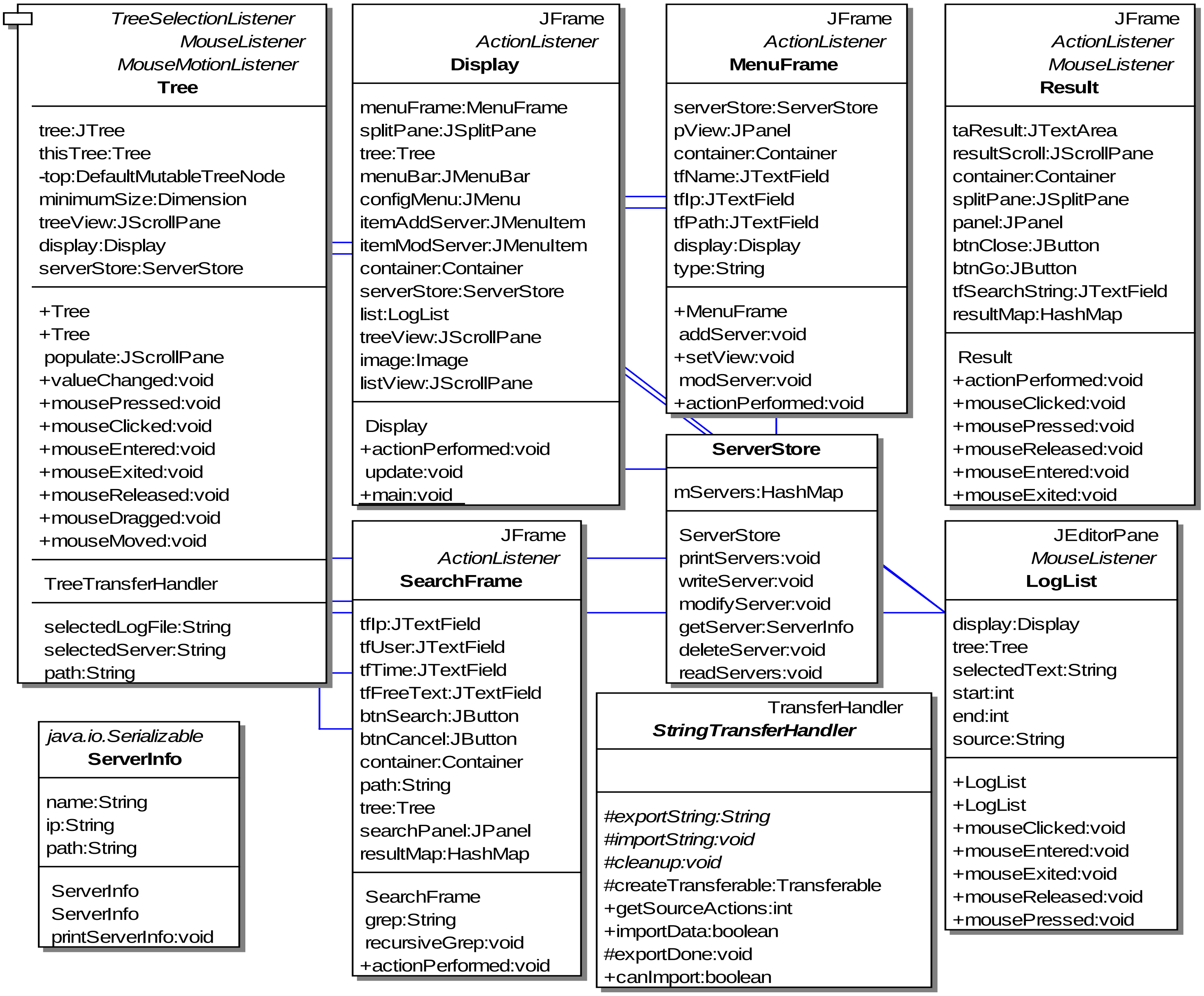
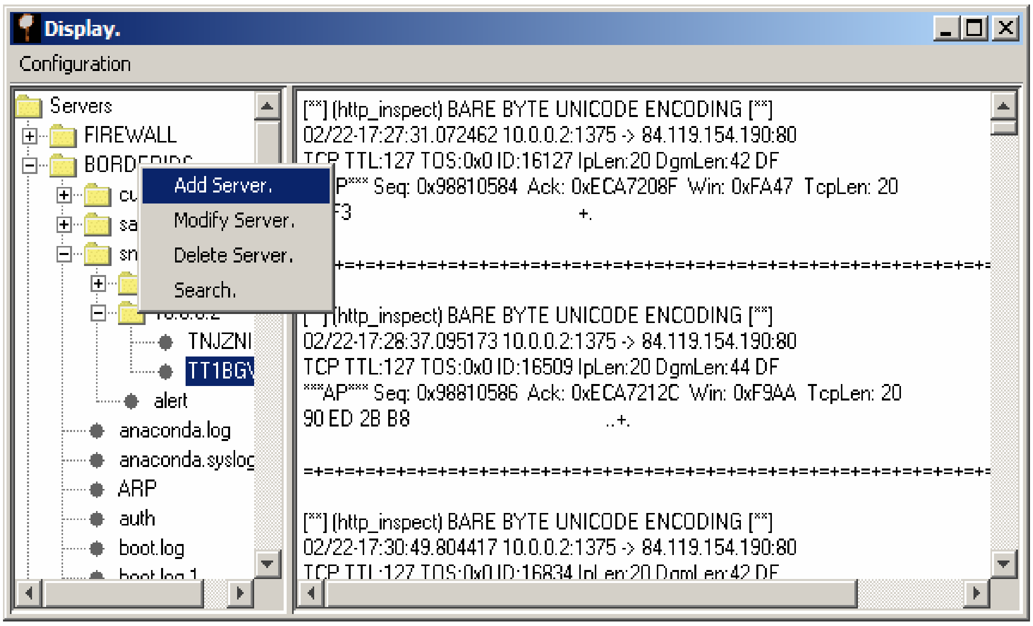


Figure 7: UML diagram.

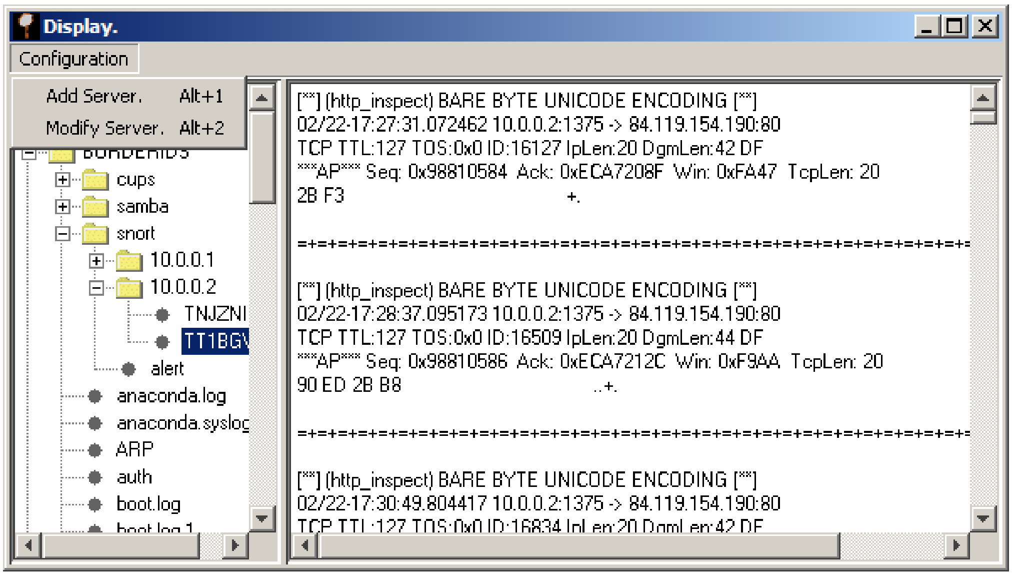
Figure 7 shows the classes developed for the package LogCorrelator. The Display class is a JFrame with a Tree, and a LogList. The MenuFrame allows the user to add or modify a server. The StringTransferHandler is used for dragging text from the LogList to the Tree; this triggers an event that opens a SearchFrame. The SearchFrame will also appear if the user right clicks on the Tree and selects "Search" from the popup menu. The Result class consists of a JFrame that displays the results of a search. The ServerStore contains ServerInfo objects that contain information about the different servers entered by the user.



```
[**] [http_inspect] BARE BYTE UNICODE ENCODING [**]
02/22-17:27:31.072462 10.0.0.2:1375 -> 84.119.154.190:80
TCP TTL:127 TOS:0x0 ID:16127 IpLen:20 DgmLen:42 DF
P*** Seq: 0x98810584  Ack: 0xECA7208F  Win: 0xFA47  TcpLen: 20
F3                              +.

+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

  ] [http_inspect] BARE BYTE UNICODE ENCODING [**]
02/22-17:28:37.095173 10.0.0.2:1375 -> 84.119.154.190:80
TCP TTL:127 TOS:0x0 ID:16509 IpLen:20 DgmLen:44 DF
***AP*** Seq: 0x98810586  Ack: 0xECA7212C  Win: 0xF9AA  TcpLen: 20
90 ED 2B B8                     ..+.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] [http_inspect] BARE BYTE UNICODE ENCODING [**]
02/22-17:30:49.804417 10.0.0.2:1375 -> 84.119.154.190:80
TCP TTL:127 TOS:0x0 ID:16834 IpLen:20 DgmLen:42 DF
```

Figure 8: Searching a log file.

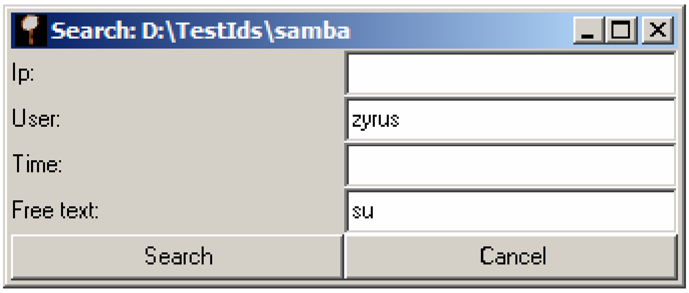
The system tester can choose to search one log file, a directory, a server, or all servers, just by right clicking in the tree, or drag from the log file into the tree component the tester wants to search in.



```
[**] [http_inspect] BARE BYTE UNICODE ENCODING [**]
02/22-17:27:31.072462 10.0.0.2:1375 -> 84.119.154.190:80
TCP TTL:127 TOS:0x0 ID:16127 IpLen:20 DgmLen:42 DF
***AP*** Seq: 0x98810584  Ack: 0xECA7208F  Win: 0xFA47  TcpLen: 20
2B F3                           +.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] [http_inspect] BARE BYTE UNICODE ENCODING [**]
02/22-17:28:37.095173 10.0.0.2:1375 -> 84.119.154.190:80
TCP TTL:127 TOS:0x0 ID:16509 IpLen:20 DgmLen:44 DF
***AP*** Seq: 0x98810586  Ack: 0xECA7212C  Win: 0xF9AA  TcpLen: 20
90 ED 2B B8                     ..+.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

[**] [http_inspect] BARE BYTE UNICODE ENCODING [**]
02/22-17:30:49.804417 10.0.0.2:1375 -> 84.119.154.190:80
TCP TTL:127 TOS:0x0 ID:16834 IpLen:20 DgmLen:42 DF
```

Figure 9: The configuration menu.

The system tester may add a server or modify a server by using the configuration menu, or by right clicking the component.



Ip:

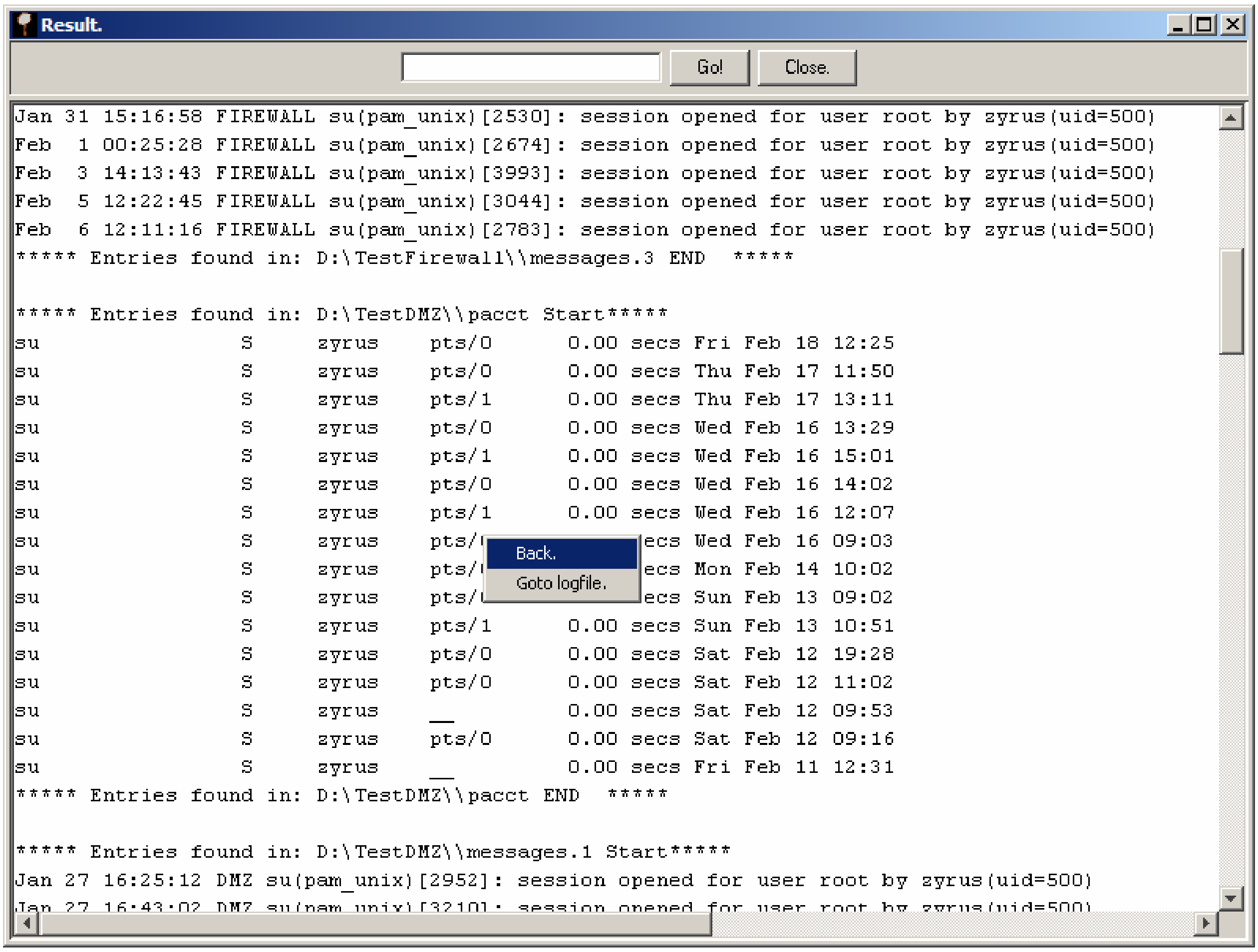
User: zyrus

Time:

Free text: su

Figure 10: The search frame.

The prototype does not enter the argument when dragging an entry from the log file into the tree; when the user drags text from the log file into the tree, the selected text then appears in the free text argument. The user then has to format the entry to the right arguments. The result will only contain log file entries that mach all of the arguments.



```
Jan 31 15:16:58 FIREWALL su(pam_unix)[2530]: session opened for user root by zyrus(uid=500)
Feb  1 00:25:28 FIREWALL su(pam_unix)[2674]: session opened for user root by zyrus(uid=500)
Feb  3 14:13:43 FIREWALL su(pam_unix)[3993]: session opened for user root by zyrus(uid=500)
Feb  5 12:22:45 FIREWALL su(pam_unix)[3044]: session opened for user root by zyrus(uid=500)
Feb  6 12:11:16 FIREWALL su(pam_unix)[2783]: session opened for user root by zyrus(uid=500)
***** Entries found in: D:\TestFirewall\\messages.3 END  *****

***** Entries found in: D:\TestDMZ\\pacct Start*****
su                S      zyrus    pts/0      0.00 secs Fri Feb 18 12:25
su                S      zyrus    pts/0      0.00 secs Thu Feb 17 11:50
su                S      zyrus    pts/1      0.00 secs Thu Feb 17 13:11
su                S      zyrus    pts/0      0.00 secs Wed Feb 16 13:29
su                S      zyrus    pts/1      0.00 secs Wed Feb 16 15:01
su                S      zyrus    pts/0      0.00 secs Wed Feb 16 14:02
su                S      zyrus    pts/1      0.00 secs Wed Feb 16 12:07
su                S      zyrus    pts/           ecs Wed Feb 16 09:03
su                S      zyrus    pts/           ecs Mon Feb 14 10:02
su                S      zyrus    pts/           ecs Sun Feb 13 09:02
su                S      zyrus    pts/1      0.00 secs Sun Feb 13 10:51
su                S      zyrus    pts/0      0.00 secs Sat Feb 12 19:28
su                S      zyrus    pts/0      0.00 secs Sat Feb 12 11:02
su                S      zyrus    __         0.00 secs Sat Feb 12 09:53
su                S      zyrus    pts/0      0.00 secs Sat Feb 12 09:16
su                S      zyrus    __         0.00 secs Fri Feb 11 12:31
***** Entries found in: D:\TestDMZ\\pacct END  *****

***** Entries found in: D:\TestDMZ\\messages.1 Start*****
Jan 27 16:25:12 DMZ su(pam_unix)[2952]: session opened for user root by zyrus(uid=500)
Jan 27 16:43:02 DMZ su(pam_unix)[3210]: session opened for user root by zyrus(uid=500)
```

Figure 11: Search result.

The result of searching all log files in all servers for the user zyrus, and free text argument “su”. When a result is displayed the user can further refine the search by entering free text into the text area and by pressing “go”. If the user presses the right mouse button, the user can go to the log file, and when inside a log file, he can return to the result.