TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Computer Science

Chair of Network Software

Event Management and active defense framework for small companies

Master's thesis

ITI70LT

Author: Markus Kont

Student code: 121785IVCMM

Supervisor: Risto Vaarandi, Ph.D

Tallinn 2014

Declaration

I hereby declare that I am the sole author of this thesis. The work is original and has not been submitted for any degree or diploma at any other University. I further declare that the material obtained from other sources has been duly acknowledged in the thesis.

....

(date) (signature)

List of Acronyms and Abbreviations

IT	Information Technology
IS	Information Systems
OS	Operating system
SEC	Simple Event Correlator
SIEM	Security Information and Event Management
CPU	Central Processing Unit
RAM	Random Access Memory
SQL	Structured Query Language
DBMS	Database Management System
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
OWASP	Open Web Application Security Project
NIDS	Network IDS
HIDS	Host IDS
HIPS	Host IPS
OSI	Open Systems Interconnection
DoS	Denial of Service
ISP	Internet Service Provider
OSSIM	Open Source Security Information Management
SEM	Security Event Management
SIM	Security Information Management
RFC	Request For Comments
UDP	User Datagram Protocol

IP	Internet Protocol
MitM	Man-in-the-Middle
IETF	Internet Engineering Task Force
TLS	Transport Layer Security
TCP	Transmission Control Protocol
ELSA	Enterprise Log Search and Archive
GNU	Gnu's Not Unix
GPL	General Public License
GSS-API	Generic Security Service Application Program Interface
ACL	Access Control List
CSV	Comma Separated Values
JSON	JavaScript Object Notation
XML	Extensible Markup Language
STOMP	Simple Text Oriented Message Protocol
SMTP	Simple Mail Transfer Protocol
GELF	Graylog Extended Log Format
ERE	Extended Regular Expressions
PCRE	Perl Compatible Regular Expression
LTS	Long Term Support
HTTPS	Hypertext Transfer Protocol Secure
SSH	Secure Shell
PHP	PHP: Hypertext Preprocessor
NTP	Network Time Protocol
DNS	Domain Name System
SLD	Second-level domain
TLD	Top-level domain
POP	Post Office Protocol
LAMP	Linux Apache MySQL PHP
SSH	Secure Shell
SNMP	Simple Network Management Protocol
NIC	Network Interface Card

RAID	Redundant Array of Independent Disks
CA	Certificate Authority
CN	Common Name
FQDN	Fully Qualified Domain Name
ARP	Address Resolution Protocol
VPN	Virtual Private Network
VM	Virtual Machine

Abstract

This thesis focuses on building an event monitoring and active defense framework for a small company, which could automate security incident handling. One of the main contributions of this thesis is an analytical and performance comparison of existing log collection and event correlation tools. Analysis of log collection tools includes three syslog message collection solutions – Rsyslog, Syslog-ng and NXLog. A number of tests were conducted by the author, in order to evaluate the performance of these solutions for different event collection scenarios. Apart from the tests, the thesis also provides a detailed assessment of event correlation capabilities of SEC and NXLog.

Another contribution of this thesis is the description of event monitoring and active defense framework for a small company environment. Based on conducted tests and comparisons, Syslog-ng and SEC were selected for building the framework. The proof-of-concept of the proposed framework has been implemented in Spin TEK AS by the author. Apart from describing the architecture of the implementation, the thesis also provides a publicly available repository of SEC rules, infrastructure configuration guidelines and control scripts. The content of the repository is maintained by the author, and helps the reader to set up a similar system for real-time event monitoring and active defense.

Table of content

Declaration....1
List of Acronyms and Abbreviations....2
Abstract....5
Table of content....6
List of Diagrams....8
List of Tables....9
1. Introduction....10
    1.1. Problem statement and contribution of the thesis....12
    1.2. Outline....13
    1.3. Acknowledgments....14
2. Overview of existing solutions and related work....15
    2.1. The roles of IDS and IPS in evolving IT landscape....15
    2.2. SIEM systems....18
    2.3. Syslog protocol....19
    2.4. Log collection software....21
    2.5. Event Correlation....22
3. Comparative analysis of open-source log collection and correlation tools....24
    3.1. Log Collection tools....25
        3.1.1. Functionality....26
        3.1.2. Configuration....31
        3.1.3. Performance....35
    3.2. Comparison of SEC and NXLog....40
        3.2.1. Functionality....40
        3.2.2. Configuration....43
    3.3. Solutions selected for the monitoring system....44
4. Implementation....46
    4.1. Infrastructure overview....46
    4.2. Centralized logging and event correlation infrastructure....48
        4.2.1. Certificate authority....48
        4.2.2. Syslog server....49
        4.2.3. Syslog client....50
        4.2.4. Application specific log collection....51
        4.2.5. Event Correlation framework....52
    4.3. Log pattern analysis and event generation....54
        4.3.1. Authentication failures....54

4.3.2. Web application injections....57
4.3.3. DNS server events....58
4.3.4. Attack events....60
5. Summary....61
Sündmuste haldamise ning aktiivse kaitse raamistik väikeettevõttele....62
List of References....63
Appendices....68
Appendix 1. Syslog daemon compilation options....68
Appendix 2. Unfiltered TCP syslog server configurations....70
Appendix 2.1. NXLog....70
Appendix 2.2. Rsyslog 5.8....71
Appendix 2.3. Rsyslog 7.6....71
Appendix 2.4. Rsyslog 8.1....71
Appendix 2.5. Syslog-ng 3.3....72
Appendix 2.6. Syslog-ng 3.5....73
Appendix 3. Filtered TCP syslog server configurations....74
Appendix 3.4. NXLog....74
Appendix 3.2. Rsyslog 5.8....75
Appendix 3.3. Rsyslog 7.6....76
Appendix 3.4. Rsyslog 8.1....76
Appendix 3.5. Syslog-ng 3.3....77
Appendix 3.6. Syslog-ng 3.5....78
Appendix 4. Test termination rule....79

List of Diagrams

Diagram 1. BSD syslog format.....20
Diagram 2. SEC configuration flow.....43
Diagram 3. NXLog pm_evcorr configuration flow.....44
Diagram 4. Event generation flow.....52
Diagram 5. SEC ruleset framework.....53

List of Tables

Table 1. Syslog daemon feature comparison.....29
Table 2. BSD message processing with 8 CPU cores.....37
Table 3. Filtered BSD message processing with 8 CPU cores.....37
Table 4. BSD message processing with 2 CPU cores.....38
Table 5. Filtered BSD message processing with 2 CPU cores.....39

1. Introduction

Over the course of last decade, the world has seen a massive rise in the consumption of and dependence on various electronic services. Services like simple web pages, electronic mail, e-commerce applications and complex information exchange pathways. They have become both tools of trade and integral part of everyday lives for large part of modern population who are often referred to by information technology professionals as end-users.

The complexity of actual solutions used to create and maintain those services is often visible only to the professionals who are tasked with managing them. Combination of high demand for novel IT solutions and the trend of workforce consolidation [1] often results in overburdening of said professionals who are forced to prioritize the incidents that have to be resolved at any given time. Network connectivity can often be compared to a double-edged sword – services have to be visible over network to provide functionality to end-users but the same channel exposes them to increasing amounts of threats from cyber criminals [2].

Conducting attacks, both automated and manual, can require very little effort [3]. On the other hand doing that in a way that leaves no footprint is difficult. The functionality to store entries of system events in log files is an integral part of operating systems (OS) and programs, which the author shall refer to from hereon as daemons, [4] used to maintain services. Some attacks manifest themselves through one specific event and are thus easy to recognize. However, other attacks might produce more complex patterns of multiple events which span over time, making them harder to detect and requiring event correlation tools. In research literature, event correlation is defined as an interpretation procedure where a new meaning is assigned to a group of events in a predefined time window.

Event correlation is a feature of several Security Information and Event Management (SIEM) systems which can be licensed from market leaders such as IBM/Q1 Labs, HP-ArcSight, McAfee/Nitro, Splunk, LogRhythm or RSA [5]. While clearly beneficial for compliance reasons, SIEM systems can also be criticized for heavy resource requirements, both from hardware and manpower, and highly commercialized nature [6]. Implementation of fully fledged SIEM system would require additional investment in the terms of administrator working hours – management of any new system introduced into existing infrastructure would have to be delegated to someone. Also, resources used to maintain logging infrastructure may be limited by the amount of CPU cores, RAM and disk space to be allocated or simply by the age of hardware. Since commercial SIEM systems are not affordable to many small to medium sized organizations, several open-source log management solutions, such as Logstash, Graylog2 and Kibana, have emerged recently.

The thesis is written from a small business perspective where the operational aspects of complex IS infrastructure are often delegated to small crews of few individuals who must be able to utilize and optimize the tools, and knowledge, they have at their disposal to develop pragmatic security solutions. This must be done without compromising the ability to carry out day-to-day tasks. Since the deployment of a commercial SIEM system is too costly for small IT organizations, the author shall not use these solutions. Instead, the author shall focus on lightweight (low resource requirements, already present in OS deployments or focused on providing specific functionality) open-source tools.

System administrators have a simple principle – any action that has to be carried out more than once on a routine basis should be automated. If the attacks can be carried out in an automated manner, why not automate the defense as well? The knowledge to identify attack patterns from log files already exists for many system administrators. Even visual confirmation of certain log entries can be used for that purpose. The caveat lies within two aspects:

- Logs are often only collected on local server with no centralized logging infrastructure in place;
- Incident handling via log verification is done manually, especially in smaller companies which lack automated incident handling and response tools.

Manual verification means that events are often discovered too late, the damage could already be done. Even if the relevant entries are clearly distinguished within log file, the distributed nature of modern IT services can cause too much information fragmentation for administrator to handle.

Open-source log management solutions often focus on log visualization and long term storage aspects which is the reason why they commonly utilize database back-end (Elasticsearch, MongoDB or some SQL-based DBMS) [7]. Automated defense based on event messages, on the other hand, requires real-time event correlation which is not directly related to storage aspects of log management. Even when utilized, event correlation within existing solutions mainly serves as a tool for generating notifications while response process is left to the system operator.

In the following subsections, the author will define the scope of the thesis, describe its main contributions, and set the outline for the thesis.

1.1. Problem statement and contribution of the thesis

The problem addressed within the thesis is the lack of infrastructure solutions which could automate attack pattern identification and responsive actions. While several IDS and IPS tools exist, they mainly focus on packet inspection on the network level, and lack ability to cross-correlate multiple attack patterns. Log management solutions serve mainly as a visualization and storage tools that can be used for data mining or alert generation. With this in mind, contributions of this thesis are the following:

- Implementation of centralized logging infrastructure for Linux based servers within small company environment;
- Comparative analysis of widely used log collection daemons that utilize syslog protocol;
- Comparative analysis of event correlation solutions;
- Development of event correlation rules for attack patterns within log files, with the purpose to defend commonly used infrastructure daemons;
- Implementation of real-time active defense against common malicious patterns;

Also, another major contribution of the thesis are publicly available configuration files of solutions within the presented scope, implementation instructions, helper scripts and event correlation rules.

1.2. Outline

Chapter 1. provides information on problem background while also identifying specific issues that are to be addressed in the upcoming chapters.

Chapter 2. presents an overview of existing infrastructure defense solutions and previous scientific research.

In Chapter 3. the author will test and compare log collection and event correlation solutions which are described briefly in Chapter 2.

Chapter 4. describes the implementation logic of centralized monitoring and correlation system which is based on components chosen as a result of Chapter 3 comparative analysis.

Results with possible improvements will be summarized in Chapter 5. Technical configuration files and test output data of Chapter 3. comparison shall be presented in the

Appendices section. Implementation guidelines, scripts and event correlation rules are publicized and maintained by author in [54].

1.3. Acknowledgments

The author would at this point like to expand his gratitude to those who supported him. To board members of Spin TEK AS, who provided the opportunity to tackle such problem. To his friends Anti Räis and Tanel Liiv, who were pressing the buttons that generated his log data. Anti in particular, for authoring the VividSEC plug-in for Sublime text editor, and all the constructive conversations while consuming countless gallons of alcoholic beverages with the author. Last but by no means least to Risto Vaarandi, for creating SEC event correlation tool and mentoring the thesis.

2. Overview of existing solutions and related work

This chapter reviews existing work on intrusion prevention and security monitoring solutions. Important aspect to keep in mind is the fact that fast development of cyber security field can render previous works incomplete or ineffective. Therefore, this chapter does not attempt to cover all academic work and industrial solutions of the last 15-20 years, but will rather focus on recent and most influential work.

2.1. The roles of IDS and IPS in evolving IT landscape

The automated threat detection and incident handling is usually carried out by dedicated Intrusion Prevention Systems (IPS) that differ from passive Intrusion Detection Systems (IDS). IDS is a software solution that detects attacks on information system, usually with the intent to generate alerts, and notify the persons who are responsible. IPS is a solution that, in addition to detecting abnormal traffic, has the ability to take automatic action against the possible attack. Human element might fail to notice monitoring alerts from IDS or simply not be able to respond in time, thus giving the attacker significant advantage.

As discussed in the SANS institute white-paper written by Doug Brown [9], active response versus passive notification is not the only aspect that differentiates IPS from IDS. IPS systems must also employ lower tolerance for false positives, as the repercussions are far more severe in those cases. Therefore IPS cannot simply be described as an "IDS that blocks traffic". On technical level, both solutions generally rely on network packet inspection and incorporate aspects of one another. For example, IPS should feed events back into the system in order to allow debugging of signature accuracy. While the article

focuses mainly on HP TippingPoint IPS, it also makes a convincing argument to separate the two solutions [9].

IPS duties are often delegated to application aware firewalls, but their ability to understand traffic can be circumvented by knowable attacker, or even IT professionals who are responsible for implementation. As presented within the white-paper by Justin Crist, systems deployed in haste can often cause confusion within IDS or IPS systems [10]. This confusion manifests itself within false positives that can occur because application developer was not aware of particular RFC standard. IPS systems can automatically block traffic that does not correctly adhere to particular standards. Another example would be configuring well-known services on non standard network ports. IPS might flag traffic destined to such ports as malicious, which could result in self-inflicted Denial of Service (DoS). Also, many firewalls employed by small companies often rely on simple open-source Linux netfilter-based access control, rather than deep packet inspection. The latter is an exclusive feature of commercial firewalls from Cisco, Juniper, HP and other vendors.

Another interesting aspect of the research paper [10] is the focus on web applications, importance of which can be summarized with the statement *"All organizations which maintain a web presence are at risk of being attacked"*. OWASP community maintains overview of most common vulnerabilities within web applications [36] to raise awareness among application developers. A weakness within web application can result in significant business impact if a successful attack were to be carried out against it.

According to Gartner [38], the leading IPS vendors in 2013 were SourceFire (acquired by Cisco), McAfee and HP. Cisco and IBM were listed as main challengers while Stonesoft (acquired by McAfee) and Radwere were listed as visionaries. HP is the maintainer of TippingPoint IPS product, which is described in [9]. McAfee produces the Network Security Platform with models that range from 100 Mbps to 40 Gbps of throughput [38]. SourceFire is also the commercial manager of Snort, the most widely used open-source Network IDS (NIDS) in the world, which can also function as an inline IPS when properly configured. "*Proper configuration"* means that sensor software must have access to raw

stream of network packets [39]. Comparison between Snort and Suricata, two competing open-source IDS solutions, was carried out by White, Fitzimmons and Matthews [40]. Comparison was based on performance, which resulted in Suricata out-performing Snort. This was due to the fact that Snort was not able to utilize multiple CPU cores.

Any monitoring system that relies on network packet analysis has several drawbacks with most severe being inability to analyze encrypted traffic. Web applications are commonly secured with TLS/SSL to enable end-to-end transport encryption between server and client. NIDS systems need to have access to private key used for encryption, or be placed before the encryption endpoint. Yamada, Miyake, Takermori, Studer and Perrig proposed a method to analyze encrypted traffic by measuring the transferred data size and timing. [41] They conclude that the method can be used to "*detect several types of web attacks"*, but "*conditions are much harder than conventional anomaly detection"*. They also point out that those conventional methods must *"archive thousands of Web pages beforehand and compare captured packets"* [41]. Overall, the application of proposed methods would be justified only for internet service providers or governmental institutions. Snort manual suggests analysis of encrypted traffic to be disabled, since it unnecessarily consumes resources, and can only generate false positives [42]. Another caveat to consider is the fact that data-center management is often outsourced to internet service providers (ISP). Implementation of any networking equipment, for example application aware firewalls and switches with port mirroring capability, might not be technically possible.

Since attack footprints can be found within log files, Linux system administrators often use a tool called Fail2ban. It functions as a host IPS (HIPS) by monitoring local log files for repeated access failures from distinct remote IP address, and setting up a Iptables DROP statement for that distinct IP once failure threshold has been reached. While highly useful, it must be configured on each host separately, and lacks the ability to detect complex attack patterns. Therefore it would not be sufficient deterrence against advanced targeted attacks. Another similar tool is DenyHosts, which only thwarts brute-force attacks against SSH servers.

2.2. SIEM systems

A general overview of Security Information and Event Management (SIEM) systems can be found within the SANS Institute white-paper by ISACA work-group, [11] which explains the nature of the system as combination of two technologies: Security Event Management (SEM) and Security Information Management (SIM). The respective purpose of those technologies are real-time and historical analysis of monitoring data, for identification of security incidents. The detailed reports provided by SIEM systems also provide information needed to demonstrate compliance during audits [11].

White-paper [11] proceeds to present general overview, business benefits and potential risks involving SIEM systems, which exemplifies their primary focus – data collection, aggregation, analysis and reporting. While SIEM systems provide several benefits, the drawbacks also have to be taken into account. A survey conducted in 2013 by alQnetworks [6] revealed that *"managing the complexity of the product is considered the biggest headache when using SIEM, followed by lack of trained personnel to manage the product and lack of integration with other products".* The facts are clear – of the responder pool nearly 31 percent would prefer to replace their existing SIEM solution for better cost savings; 25 percent have invested more than month in professional services since the implementation of their current SIEM solution; and 52 percent require two or more full-time employees to manage the chosen solution [6].

Today there are number of commercial SIEM solutions in the market, such as ArcSight, Juniper, STRM, Envision; etc [5]. However due to their cost many smaller companies are using open-source SIEM or log management systems. One such is OSSIM which is available in the form of virtual appliance.

In the case of OSSIM, event data can be collected via dedicated agent or directly from log daemons on client side. Primary method of intrusion detection is carried out by Snort IDS while the proactive value is the output of event alerts that can used by developers to harden web applications against specific risks [10]. Therefore, it can be assumed that open-source

SIEM systems share the deficiencies of conventional IDS solutions, because they mainly focus on presenting IDS alerts on central dashboard.

The same functionality can be achieved by outputting IDS alerts as event messages, and presenting them with any open-source log management and parsing solution. Enterprise log search and archive (ELSA) is a centralized framework built on Syslog-ng, MySQL and Sphinx full-text search [43]. Graylog2 is a Java-based log management server that collects remote log messages via input listeners, and stores them within Elasticsearch indexing engine. Log messages can then be searched and analyzed via web interface [44]. Kibana is a Java-based web interface that presents log messages that are stored within Elasticsearch. Log messages can be forwarded to back-end database by collector parsing engine, such as Logstash, that formats messages by field [45]. Other, more simplistic, solutions are web applications written in PHP, for example php-syslog-ng and LogAnalyzer. Such solutions present log data that is stored within SQL server. In depth analysis of log management tools was carried out within the 2013 Master's thesis of Artyom Churilin [53]. The work compares Graylog2, Kibana and ELSA, and illustrates implementation of such technologies.

2.3. Syslog protocol

When an event, which can be explained as a change in the system state, occurs, a relevant component within that system can output an event message. Event logging is a procedure of writing that event message to a local or remote data storage, referred to also as event log. Event log messages on UNIX-like platforms commonly adhere to BSD syslog protocol which was developed by Eric Allman during the 1980s for the *sendmail* program, and is considered to be *de facto* protocol for event logging [12].

RFC3164 defines the standards and implementation best practices for BSD syslog protocol [13], which are outlined in the following list:

- Any process on any device might generate an event message;
- As a general rule, many devices can send messages to relatively few collectors;
- Syslog uses UDP as underlying transport layer mechanism;
- UDP port 514 has been assigned to syslog, with recommendation to use UDP port 514 as sender source;
- Syslog message is composed of PRI, HEADER and MSG parts;
- PRI (priority) is a combination of Facility (assigned numerical value to operating system daemons and local channels) and Severity;
- HEADER contains time-stamp and indication of sender host name or IP address;
- MSG contains information regarding the process that generated the message.

Diagram 1 illustrates the syslog message format and priority calculation formula. Header part is a combination of time-stamp and host name while message content is arbitrary text.

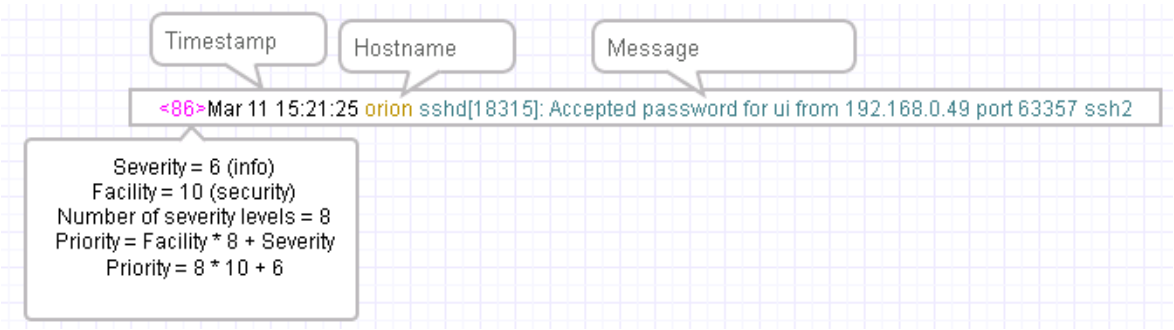

Diagram 1. BSD syslog format

The protocol is also covered in depth within *"The Ins and Outs of System Logging Using Syslog"*, which provides a detailed overview of syslog protocol, and emphasizes shortcomings in message transmission [14]. Messages are sent to remote hosts via stateless UDP protocol which provides neither confidentiality, authentication nor integrity support, leaving BSD syslog protocol susceptible to Man-in-the-Middle (MitM) attacks. As a consequence, messages transferred via RFC3164 are not valid forensic evidence because message integrity cannot be proven.

For this reason RFC3164 was rendered obsolete by RFC5424, commonly referred to as "the IETF syslog protocol", which specifies the standard to *"separate message content from message transport while enabling easy extensibility for each layer"* [15]. Primary additions include extensions for presentation of vendor specific values, and description of minimum requirements for message transport. All implementations must support TLS based transport (as described in RFC5425 [16]) while also maintaining support for legacy UDP (as described in RFC5426 [17]). TLS, specified to use port 6514/tcp, is considered to be recommended choice to ensure message confidentiality, integrity and peer authenticity. [15] The use of TCP protocol is described in RFC6587, published in 2012, which aims retroactively standardize various TCP implementations. While the document outlines the fact that functional TCP implementation have been used for several years despite the lack of standardization, RFC5425 remains as the recommended guideline to follow when presented with a choice of transport layer protocol [18].

2.4. Log collection software

UNIX syslogd has been the daemon of choice for collecting syslog messages in BSD format that, in addition to reading local file system socket and storing messages in text files, can send and receive messages using UDP port 514 [20]. In order to address the deficiencies in of syslogd, two competing open-source syslog daemon solutions have emerged – Syslog-ng and Rsyslog. Both solutions provide support for standards described by RFC5424-5426, and advanced functionality (for example storing messages in database systems). NXLog is another open-source syslog implementation that has emerged in recent years.

An overview of popular syslog daemons and log management tools was written by Vaarandi and Nizinski. [7] The paper presents performance tests of syslog collection daemons Rsyslog, Syslog-ng and NXLog. Rsyslog, which evidenced superior performance to other daemons, was also compared with Logstash in a similar test [7]. It should also be taken into account that a blog post by Zoltán Pallagi, a senior member of Syslog-ng team,

attempted to disprove the performance results presented within the research paper. In the post he claims that *"using the Syslog-ng basic configuration, as was done in this study, can produce misleading results";* [19].

Another work that focused on log infrastructure performance was written by David Lang, which documents the choices taken and lessons learned within Digital Insight (currently acquired by Intuit) [21]. Rsyslog was the chosen solution over commercial logging products due to financial and compatibility reasons, and Splunk was used for log indexing. David Lang has also published a presentation paper which covers extensively the configuration syntax and filtering options present in Rsyslog [46].

2.5. Event Correlation

Event Correlation can be defined as relating any number of events via patterns [37], and illustrated with quote *"By aggregating a set of events, one new event may be created containing a better information quality. This reduces the effort for (human) analysts.";* written by Zirkel and Wirtz in the context of problem management [24]. Another article was written by Christopher Crowell, illustrating the use of event correlation for root cause analysis in problem solving [25]. General event correlation concepts can easily be adopted to log data monitoring.

Different techniques of intrusion detection, response options and complex event correlation were written by Mustafa Toprak within his Master's thesis [22]. Practical importance of log correlation is covered in 2003 paper by Abad, Taylor, Sengul, Yurcik, Zhou and Rowe, in which a list of UNIX vulnerabilities is analyzed to find potential intrusions [23]. A SANS Institute white-paper provides an overview about creating custom SIEM applications. Event correlation was presented as one of the key components with Simple Event Correlator (SEC) being the event correlation solution which was directly referenced [26].

SEC is a lightweight open-source event correlation engine written in Perl, which is able to read input from files, and generate events from predefined rules by utilizing Perl regular expression engine. An overview of motivations for creation, event correlation concepts, and adoption within industry, is presented within the PhD thesis of Risto Vaarandi, the author of SEC [27]. However, a large part of thesis is composed of now outdated functionality description. The most up to date version can be found on SEC manual pages [28]. The paper by Vaarandi and Grimaila [29] highlights the capability of this tool to identify information security related events within real world environments. Several technical examples are presented to illustrate this point. A research paper written in 2011 by Myers, Grimaila and Mills [30] utilized this tool for the following reasons:

- Easy configuration thanks to standard format and available documentation;
- SEC is used in many organizations, giving credence to its effectiveness.

Research is concluded with the statement that, while security value is proven, there is a lot of untapped potential in log analysis as it is done currently (due to inadequate implementation techniques) [30]. Similar subject was covered within 2010 Master's thesis of Justin Myers that analyzed the use of SEC, in distributed manner, as an alternative to centralized SIEM solutions to identify malicious insider threats [31]. Event messages from different platforms can be read cross-correlated. Also, the use of SEC for the detection of security incidents within Windows workstations was outlined in a research paper that was published by SANS institute in 2013. Practical value of the paper is enhanced by the fact that it also contains rules for real time detection of several abnormal events [32].

3. Comparative analysis of open-source log collection and correlation tools

In this thesis, the author proposes the implementation of a centralized logging infrastructure with event correlation functionality, in order to create a real-time monitoring system with automated response capability. Based on literature review in Chapter 2, the critical components of such system would be syslog collector daemons and SEC.

In order to select the best software for the monitoring system, the author has conducted performance tests and made analytical comparisons between syslog daemons and correlation solutions. Previous comparisons of open-source log collection tools have been evidently made on *ad-hoc* basis with the most recent comparison, presented in Chapter 2 [7], being limited by scale and is somewhat outdated. In previous works by other researchers, functionality was briefly described, and configuration syntax was presented mainly as illustration for performance tests, the latter being subject to criticism. Please note that functionality and configuration nuances are critical factors for administrators who are tasked with continuous system maintenance. Therefore, the current chapter aims to provide an up-to-date and expanded comparison, which also takes into account small company background and event correlation context.

Based on performance tests and analytical comparisons from this chapter, components are selected for implementing the centralized logging and event correlation solution. The practical implementation and SEC ruleset development aspects of this overarching solution will be presented in Chapter 4.

3.1. Log Collection tools

As presented in Chapter 2, two competing syslog daemons on Linux and other UNIX-like platforms are Rsyslog and Syslog-ng. NXLog, while not as widely adopted as other solutions, is another highly capable log collection daemon that should be taken into account. Several different versions of Rsyslog and Syslog-ng have been compared since these syslog servers are widely used, and as of April 2014, selected versions are most commonly used. Following software is chosen for comparison:

- Rsyslog 5.8 [47];
- Rsyslog 7.6 [48];
- Rsyslog 8.1 [49];
- Syslog-ng 3.3.4 [50];
- Syslog-ng 3.5.4.1 [51];
- NXLog 2.7 [52].

Open-source log parsing tools are outside the scope of thesis. Logstash, for example, is a Java based log parser that can gather its data from existing syslog daemons, or via custom Logstash agent. Such software would effectively only add to the complexity of implemented solution, since they do not provide any benefits over using only syslog daemon.

The operating system used for implementation is Ubuntu Server 12.04 LTS (14.04 release date is within the time-frame of writing but will not be implemented yet) which maintains backward-compatible versions of Rsyslog 5.8 and Syslog-ng 3.3 within its official software repository (support ends in year 2017). System administrators often prefer to use older stable software versions with guaranteed Long Term Support (LTS) over newer versions, which would provide better functionality. As such, those older stable versions will also be included to comparison. Rsyslog also provides development version 8.1 that promises performance improvements over previous iterations, therefore this version is also included into performance tests. NXLog binary package is not maintained by Ubuntu developers (it cannot be installed via package manager), and only latest version is provided as binary

installer by NXLog developers. Therefore only latest NXLog version is included.

3.1.1. Functionality

Table 1 presents a detailed feature comparison of commonly used releases of Rsyslog, Syslog-ng and NXLog. This table is partly based on comparison between Rsyslog and Syslog-ng that was originally created in 2008 by Rainer Gerhards and was updated in 2012 to reflect changes in Rsyslog [33]. The article is licensed under GNU GPL version 2. Apart from the info from [33], the author has included detailed feature comparison data for Syslog-ng and NXLog in Table 1. Some functionality, such as Enterprise Features, are omitted from Table 1 to reflect the thesis background (open-source centric small business). Some other fields have been expanded to reflect new technology support and features unique to Syslog-ng or NXLog. Since several Rsyslog 8.1 intended modules are not operational, this development version is not included within the functionality analysis.

Feature (community/ open-source version)	**Rsyslog 5**	**Rsyslog 7**	**Syslog-ng 3.3**	**Syslog-ng 3.5**	**NXLog 2.7**
Input Sources (Native)	**Rsyslog 5**	**Rsyslog 7**	**Syslog-ng 3.3**	**Syslog-ng 3.5**	**NXLog 2.7**
UNIX domain socket	+	+	+	+	+
UDP	+	+	+	+	+
TCP	+	+	+	+	+
RELP	+	+	-	-	-
RFC3195/BEEP [34]	+[1]	+[2]	-	-	-
Kernel log	+	+	+	+	+
File	+	+	+	+	+
Message generator	+[3]	+	+	+	-
Windows Event Log	-	-	-	-	+

1 Only present in IPv4 version
2 Only present in IPv4 version
3 Available since version 5.5

Network	Rsyslog 5	Rsyslog 7	Syslog-ng 3.3	Syslog-ng 3.5	NXLog 2.7
TCP	+	+	+	+	+
GSS-API	+	+	-	-	-
ACL	+	+	+	+	-
UDP	+	+	+	+	+
RELP	+	+	-	-	-
Compression[4]	+	+	-	-	-
RFC3195	+[5]	+[6]	-	-	-
TLS	+	+	+	+	+
RFC5424-5426	+	+	+	+	+
IPv6	+	+	+	+	+
Send SNMP traps	+	+	-	-	-
Preserve original host-name	+	+	+	+	+
Filtering	**Rsyslog 5**	**Rsyslog 7**	**Syslog-ng 3.3**	**Syslog-ng 3.5**	**NXLog 2.7**
Facility and priority	+	+	+	+	+
Host-name	+	+	+	+	+
Application name	+	+	+	+	+
Message contents	+	+	+	+	+
Sender IP	+	+	+	+	+
Sub-string filtering	+	+	+	+	+
Boolean algebra in filters	+	+	+	+	+
Reusable filters	-	-	+	+	+
String expressions	+	+	+	+	+
POSIX Regular Expressions	+	+	+	+	+
Perl Compatible Regular Expressions	-	-	+	+	+
Discard messages	+	+	+	+	+
Event Correlation	-	-	+	+	+

4 Compression of messages during transport
5 Only present in IPv4 version
6 Only present in IPv4 version

Output	Rsyslog 5	Rsyslog 7	Syslog-ng 3.3	Syslog-ng 3.5	NXLog 2.7
MySQL	+	+	+	+	+
PostgreSQL	+	+	+	+	+
Oracle	+	+	+	+	+
SQLite	+	+	+	+	+
Microsoft SQL	+	+	+	+	+
Sybase (Open TDS)	-	+	-	-	+
Firebird/Interbase	+	+	-	-	+
Ingres	+	+	-	-	-
mSQL	+	+	-	-	-
MongoDB	-	+	+	+	+
Elasticsearch	-	+	-	-	-
CSV	-	+	+	+	+
XML	-	+	-	-	+
JSON	-	+	+	+	+
STOMP	-	-	-	+	-
SMTP	+	+	-	+	-
GELF	-	-	-	-	+
Platform support (native)	**Rsyslog 5**	**Rsyslog 7**	**Syslog-ng 3.3**	**Syslog-ng 3.5**	**NXLog 2.7**
Linux (Generic)	+	+	+	+	+
Windows	-	-	-	-	+
Configuration	**Rsyslog 5**	**Rsyslog 7**	**Syslog-ng 3.3**	**Syslog-ng 3.5**	**NXLog 2.7**
Include from file	+	+	+	+	+
Include from directory	+	+	+	+	+
Extensibility	**Rsyslog 5**	**Rsyslog 7**	**Syslog-ng 3.3**	**Syslog-ng 3.5**	**NXLog 2.7**
Module support	+	+	+	+	+
Third-party input plug-ins	+	+	+	+	+
Third-party output plug-ins	+	+	+	+	+

Other	Rsyslog 5	Rsyslog 7	Syslog-ng 3.3	Syslog-ng 3.5	NXLog 2.7
Dynamic output file naming	+	+	+	+	+
Control over log output format	+	+	+	+	+
RFC3339 [35] support	+	+	+	+	+
Log files larger that 2GB	+	+	+	+	+
Log rotation	+	+	+	+	+
Script triggering	+	+	+	+	+
Pipe messages to continuously running program	+	+	+	+	+
Multi-threading	+	+	+	+	+
Limit repetition	+	+	+	+	+
Multiple actions per filter	+	+	+	+	+
Web interface[7]	-	-	-	-	-
Flow-control	+[8]	+	+	+	+
DNS cache	-	+	+	+	-

Table 1. Syslog daemon feature comparison

All subjects of comparison provide quite similar feature set, with clear advances per version update. Table 1 may contain some inaccuracies, for example use of RFC3195 has been depreciated in favor of RFC5424. Software may still support it but documentation is limited (Rsyslog documentation points out that the use of RFC3195 transmission protocol is only supported over IPv4 due to lack of demand for IPv6 support). In the case of database back-ends, generic drivers could be used to connect to SQL servers which were not listed in official documentation. If functional support exists but is not represented in documentation, then author did not include it within Table 1.

7 While not a built-in feature, several log management solutions provide this functionality
8 Available since version 5.7

Rsyslog 7 is the only solution that allows log messages to be saved directly in Elasticsearch indexing back-end. This would provide significant advantage if, additionally to real-time monitoring, the solution would later be expanded to provide long term log storage functionality. Some improvements of Rsyslog version 7 over version 5 include support for additional log parsing options (including XML, CSV and JSON) and support for MongoDB back-end. All included versions of Rsyslog can also transport event messages via custom Reliable Event Logging Protocol (RELP), but danger of vendor lock-in when support for universal RFC5424 exists in every compared syslog daemon would be a deterrent against using it. Syslog-ng and NXLog allow reusable filters to be written, which is an advantage over Rsyslog. Syslog-ng 3.5 also provides support for STOMP protocol which would be beneficial if custom log applications were to be written for centralized log monitoring and correlation solution. This is outside the scope of current thesis.

The most beneficial features of NXLog are existence of native binary package for Microsoft Windows operating systems, support for Windows EventLog format and event correlation module. Log collection from Windows platform is outside the scope of this thesis, but since such servers exist within the infrastructure of Spin TEK, the feature cannot be ignored. NXLog also supports Graylog Extended Log Format (GELF) message output which could be useful if Graylog2 log management solution would later be incorporated to the monitoring system. Use of universal syslog protocol would still be preferable in that scenario, as it is supported by Graylog2. Syslog-ng supports event correlation but, unlike SEC and NXLog, does not contain different rule types (one or more patterns within configured time window can be identified, which would result in generation of event message). Therefore correlation functionality is too limited when compared to SEC.

Every compared solution supports RFC5424 standard. Because sender and receiver only have to understand common message format and transportation mechanism, it can be concluded that monitoring solution is not limited to only one syslog daemon across multiple infrastructure devices.

3.1.2. Configuration

This chapter provides an overview of Rsyslog, Syslog-ng and NXLog configuration syntax. Two sample scenarios were chosen by author – simple and complex. Simple scenario illustrates basic message forwarding functionality. Complex scenario reflects the need of Spin TEK AS to manage web server logs. In simple filtering scenario, all messages must be forwarded to port 6514/tcp on remote host while messages with *mail* syslog facility value are also stored within text file on local device. In complex filtering scenario, Apache2 web server access and error logs must be stored within separate text files.

Rsyslog was written to be a descendant of legacy syslogd, which means that configuration follows the same principles. *Selectors* filter the log messages by *facility* and *priority* while *Action* specifies where the message should be sent to or displayed at. This includes local file storage, "pipelines", terminals, users or remote hosts. Simple filtering with Rsyslog is visually clean.

```
$ModLoad imuxsock
$ModLoad imklog
mail.*      /var/log/mail
*.*         @@192.168.0.180:6514
```

Additional functionality in Rsyslog includes numerical and textual keywords for syslog protocol elements, for example *facility, message* and *severity*. Complex filters can be created by combining these keywords with evaluation operators like *contains, startswith* and *endswith*. The following Rsyslog example illustrates the complex filtering scenario.

```
$ModLoad imuxsock
$ModLoad imklog
if $syslogtag contains 'apache' \
      and $syslogpriority == 3 then   /var/log/apache2/syslog-error.log
if $syslogtag contains 'apache' \
      and $syslogpriority != 3 then   /var/log/apache2/syslog-access.log
```

Rsyslog does not support reusable filters – *syslogtag* evaluation is combined with priority evaluation, but former has to be redefined for every filter. A better way to achieve this would be the use of if-else statements.

```
if $syslogtag contains 'apache' then {
      if $syslogpriority == 3 then
            action(type=”omfile”           file=”/var/log/apache2/syslog-
error.log”)
      else
            action(type=”omfile”           file=”/var/log/apache2/syslog-
access.log”)
}
```

Even when documented, long configuration files containing such filters can be difficult to manage. Rsyslog introduces incremental extensions to configuration options per version update, which can be illustrated by the fact that multiple distinct syntax versions are supported in parallel (“old” and “new” versions, as described in [46]). Common practice is to utilize file inclusion to maintain manageability of large configuration sets. Files with *“.conf”* suffix, placed in /etc/rsyslog.d/ folder, are extensions of main configuration file. Syslog-ng uses a custom configuration syntax that divides directives into logical blocks. The following configuration directives realize the simple filtering scenario.

```
source s_src {system(); internal();};
filter mail_all {facility(2);};
destination d_remote {tcp(“192.168.0.180” port(6514););};
destination d_file {file(“/var/log/mail”);};
log {source(s_src); destination{d_remote};};
log {source(s_src); filter(mail_all); destination(d_file);};
```

When compared to Rsyslog, simple filtering directives are more complex. This is due to the fact that Syslog-ng configuration blocks are designed to be invoked at will. Both *log* directives in the example use the same *source* and only small modifications are needed within second *log* directive (addition of *filter* and change of *destination).* Following Syslog-ng configuration realizes the complex filtering scenario.

```
source s_src {system(); internal();};
filter apache {program(“apache”);};
filter error {level(3);};
filter access {not filter(error);};
destination d_file_error {file(“/var/log/apache2/syslog-error.log”);};
destination d_file_access {file(“/var/log/apache2/syslog-access.log”);};
log {source(s_src); filter(apache); filter(error);
      destination(d_file_error);};
log {source(s_src); filter(apache); filter(access);
      destination(d_file_access);};
```

The overall level of complexity stays nearly the same when compared to simple Syslog-ng example. Each filter is created once and invoked when needed, which is similar to functions in programming languages. The benefits of procedural approach is universally accepted by IT specialists – the ability to trace configuration flow, especially by someone who has not written the original directives, outweighs the initial learning curve. Syslog-ng configuration blocks can be written in one line or separated by newlines and white-spaces.

NXLog uses Apache-style configuration syntax that contains one directive per line ("newline" is evaluated as delimiter) while directives are grouped into configuration blocks. Anyone who has configured Apache web server or XML documents should have little trouble adjusting to log directives. The following code is equal to previous simple examples.

```
<Input unix>
      Module      im_uds
      uds         /dev/log
      Exec        to_syslog_ietf();
</Input>
<Input kern>
      Module      im_kernel
      Exec        to_syslog_ietf();
</Input>
<Output tcpout>
      Module      om_tcp
      Host        192.168.0.180
      Port        6514
</Output>
<Output mail_file>
      Module      om_file
      Exec        if $SyslogFacilityValue != 2 drop();
      File        "/var/log/mail"
</Output>
<Route 1>
       Path unix,kern  => tcpout,mail_file
</Route>
```

Syntax is clear and flexible, but basic filtering example requires a large amount of configuration when compared to Rsyslog. It must also be stated by author that clear overview of simple filtering by syslog facility was surprisingly difficult to find in official documentation. NXLog configuration block invocation approach is similar to Syslog-ng,

but the latter provides better documentation of variables supported within modules. The following NXLog directives are functionally analogous to complex filtering examples.

```
<Output apache_error>
        Module  om_file
        Exec    if $SourceName != "apache" drop();
        Exec    if $SyslogSeverityValue != 3 drop();
        File    "/var/log/apache2/syslog-access.log"
</Output>
<Output apache_access>
        Module  om_file
        Exec    if $SourceName != "apache" drop();
        Exec    if $SyslogSeverityValue == 3 drop();
        File    "/var/log/apache2/syslog-access.log"
</Output>
```

Author has omitted *input* and *route* definitions in this example. Inverse *drop()* filters are used because they provide performance benefits over deprecated *pm_filter* processor module. Any message that matches conditions in *Exec* directive will be not be processed. NXLog allows *Exec* directive to be used in any configuration block; *Input, Processor* or *Output.* To reduce redundancy, the *SourceName* directive could be moved to separate *Processor* which would also mean that separate *Route* has to be created.

Overall, the benefits of Rsyslog are the most apparent when presented with simple configuration scenarios. When configured with high number of complex filters, proper documentation and configuration file version management procedures should be in place to ensure manageability of monitoring system. Syntax is flexible, but fragmented in recent software release versions. Anyone who does not have extensive experience with Rsyslog would experience difficulties in ensuring continuous monitoring system maintenance.

On the other hand, Syslog-ng provides a structured approach as the only option. The initial learning curve is more difficult when compared with Rsyslog, but configuration logic is easier to trace. Syslog-ng learning curve also becomes less intensive after basic concepts have been introduced to system administrator.

NXLog provides structured approach that is nearly as good as Syslog-ng. Anyone familiar with Apache configuration or XML will have little trouble with initial understanding of

syntax. Initial understanding can be easier to attain when compared to Syslog-ng, but latter does provide better documentation. This does not imply that NXLog documentation is bad, it simply requires more manual search than Syslog-ng.

3.1.3. Performance

In this section, the author presents the results of performance tests that were conducted on selected syslog daemons. Most modern CPU-s provide enough computational throughput to process a large amount of log messages, but resources for practical implementation are limited. A finite number of logical CPU cores can be allocated to virtual machines. Processing efficiency is an important factor, since physical CPU cores are shared between multiple virtual machines.

Log monitoring and event correlation solution do not require memory intensive indexing solutions. Event messages are stored in text files that can easily be compressed during process known as “log rotation”. Therefore memory and disk space are not included as comparison criteria.

The tests that are presented in this section measure the usage of CPU time (in seconds), elapsed wall-clock time (in seconds) and CPU utilization (in percentage). Measurements are designed to reflect the resource consumption and throughput of the tested syslog server (for example, if the server is able to handle 1 million messages in 5 seconds, it's throughput is 200,000 messages per second). Following central testing method is used to reflect realistic environment with maximum accuracy:

1. Central server node receives log messages from 5 remote host nodes;
2. 2 CPU cores are allocated to remote host nodes;
3. Syslog daemons are compiled with configuration options that are presented in Appendix 1;
4. Log generation script (presented in [54]) is deployed on each host;
5. 4 parallel script executions are carried out on each host;
6. Each script generates 300 000 messages;

7. Syslog daemon is executed with Linux *time* utility to measure the throughput for processing 6 000 000 messages from 20 parallel generators.

The *time* utility also measures CPU time when syslog daemon is activated but no messages are yet received. Therefore syslog daemon initialization, log generation on remote hosts and syslog termination must be synchronized.

1. Clocks of each node are synchronized from private NTP server using *ntpdate* command;
2. Linux *crontab* is used for parallel initialization of syslog daemon with *time* command, and log generator scripts on remote hosts;
3. Last log message sent from generator scripts contain string "DONE";
4. A syslog daemon filters messages containing that string to separate log file;
5. After 20 occurrences of that string have been detected, a syslog daemon is terminated with *pkill* command;
6. Measurements from *time* command are appended to text files once syslog daemon is terminated.

Parallel execution via *crontab* avoids the "Firing squad synchronization problem", as alternative method of using SSH to sequentially initiate log generation would introduce delay between each script execution. Table 1 presents the results of first benchmark in which no complex filtering is configured and 8 CPU cores are allocated to central server. Note that CPU time is calculated across multiple threads, allowing it to exceed wall-clock time of benchmark duration. Utilization percentage, presented by GNU version of *time* utility, is calculated by dividing CPU time with wall-clock time. Arithmetic average values of 3 test iterations are presented in Tables 2-5. Comprehensive output data of each test is uploaded to [54]. Daemon configuration directives are presented in Appendix 2.

Average	Rsyslog 5	Rsyslog 7	Rsyslog 8	Syslog-ng 3.3	Syslog-ng 3.5	NXLog 2.7
CPU time (seconds)	25,56	20,97	20,25	61,86	58,3	428,34
Wall-clock time (seconds)	23,05	7,95	9,7	13,21	12,3	168,5
CPU utilization (percentage)	111	264	215	486	472	254

Table 2. BSD message processing with 8 CPU cores

Rsyslog 7 is the most efficient syslog for unfiltered log message processing. Version 7 also displays ability to use multiple threads while maintaining CPU efficiency. Version 8 is under development, so lower utilization percentage may be due to software instability. All Syslog-ng versions are less efficient than Rsyslog, but former display higher utilization of multiple CPU cores. Syslog-ng version 3.5 displays highest CPU utilization with resulting throughput that is slightly weaker than Rsyslog version 7 and 8. NXLog performance is the weakest. Table 3 presents second benchmark which is conducted with complex filters. Each priority level is stored to different file that is configured using dynamic log file naming, essentially forcing the server to sort messages into 192 different log files. Daemon configuration directives are presented in Appendix 3.

Average	Rsyslog 5	Rsyslog 7	Rsyslog 8	Syslog-ng 3.3	Syslog-ng 3.5	NXLog 2.7
CPU time (seconds)	86,75	105,97	86,15	112,65	107,09	1021,21
Wall-clock time (seconds)	85,55	82,47	76,55	21,65	20,75	366
CPU utilization (percentage)	101	128	112	520	516	278

Table 3. Filtered BSD message processing with 8 CPU cores

Surprisingly, Rsyslog seems to lose multithreading ability when filtering messages to different files. Version 7 is also less efficient compared to version 5. Both versions of Syslog-ng, on the other hand, are able to effectively use CPU time of 5 cores to process messages slightly faster than in tests displayed in Table 2. Rise in utilized CPU time is also smaller compared to Rsyslog, which seems to be four times less efficient than without filtering. NXLog filtering seems to be extremely inefficient. Table 4 presents benchmark results with 2 allocated CPU cores and no complex filtering. Daemon configuration directives are presented in Appendix 2.

Average	Rsyslog 5	Rsyslog 7	Rsyslog 8	Syslog-ng 3.3	Syslog-ng 3.5	NXLog 2.7
CPU time (seconds)	22,09	12,46	12,82	42,92	37,82	218,55
Wall-clock time (seconds)	20,38	9,12	9,17	29	26,89	140,92
CPU utilization (percentage)	108	136	139	147	140	155

Table 4. BSD message processing with 2 CPU cores

Results are analogous to first test with execution times that are nearly identical for all syslog daemons. Rsyslog 7 is the most efficient syslog daemon. Syslog-ng exhibits higher utilization on multiple threads, while all Rsyslog versions are more efficient in terms of CPU usage. and Table 5 presents the results of benchmark with 2 allocated CPU cores and complex filtering configuration. Daemon configuration directives are presented in Appendix 3.

Average	Rsyslog 5	Rsyslog 7	Rsyslog 8	Syslog-ng 3.3	Syslog-ng 3.5	NXLog 2.7
CPU time (seconds)	85,49	94,77	80,67	69,92	62,91	585,91
Wall-clock time (seconds)	87,03	89,72	77,11	45,03	41,3	318,86
CPU utilization (percentage)	98	105	104	155	152	183

Table 5. Filtered BSD message processing with 2 CPU cores

Compared to 8 CPU core results displayed in Table 3, Rsyslog all versions maintained similar performance while Syslog-ng both versions suffered throughput drop of nearly 2 times. Syslog-ng still processed messages faster than other daemons and exhibited most efficient CPU usage. NXLog completed filtered tests with 2 CPU cores faster than with 8 CPU tests. This appears to be an anomaly with only possible explanation being higher CPU utilization per CPU core – average 183 percent from maximum 200 were used against previous 278 percent from maximum 800. As such, utilization of multiple threads seems to cause anomalies in message processing time for NXLog.

Preliminary TLS tests were conducted. However, Rsyslog versions 7 and 8 caused network stream interruption, which terminated message generation before test was complete. Addition of encryption does not cause noticeable increase in CPU load. This is due to the fact that daemons use external TLS libraries to create encrypted tunnel while message processing logic remains unaltered. For that reason, the author did not include average results of those tests in this section.

Overall, Rsyslog 7 proved to be most efficient when no complex filtering is in place while Syslog-ng 3.5 provides better performance reliability and scalability with multiple CPU cores.

3.2. Comparison of SEC and NXLog

Functionality analysis revealed the fact that NXLog provides *pm_evcorr* processor module that is directly inspired from SEC. Please keep in mind that NXLog is a fully-fledged log daemon while SEC is designed to be a monitoring tool that can be integrated into existing logging environments. Performance benchmarks revealed NXLog to be the most inefficient of selected syslog daemons. If the event correlation module were to provide competitive feature set to SEC, then NXLog could still be considered as viable option for central syslog server. SEC could then be omitted from overarching log monitoring and correlation solution, which would provide clear benefits in terms of configuration management and performance (NXLog is written in low-level C programming language while SEC is written in Perl). The following sub-section will present analytical comparison of functionality and configuration options between these event correlation tools.

3.2.1. Functionality

The most popular open-source event correlation tool is Simple Event Correlator [28]. It is a Perl script that processes input messages according to defined rules, generates event on positive match and carries out actions after event has occurred. Event messages are matched by patterns based on strings or Perl regular expressions. Rules are defined within configuration files that are passed to SEC on program initialization with --*conf* parameter. The latest version, 2.7.5, supports following rule types:

- Single;
- SingleWithScript;
- SingleWithSuppress;
- Pair;
- PairWithWindow;
- SingleWithThreshold;
- SingleWith2Threshholds;
- EventGroup;

- Suppress;
- Calendar;
- Jump;
- Options.

Single rule matches event according to defined regular expression and executes action. *WithScript* variation forks a process to execute external program while existing *contexts* are written to standard input of that program. *SingleWithSuppress* variation filters repeated events in specified time window, which avoids overloading system with excessive amount of events and actions. *SingleWithThreshold* does not take action until certain number of matches have occurred in set amount of seconds (specified as integer in *Window* directive) while *SingleWith2Threshholds* executes a second pattern matching after first action has been taken. For example, a rule could be written to block offending IP if password brute-force attack is detected (5 log-in attempts in 1 second, obviously not a human error) and remove that IP from blocked hosts if second threshold is met (0 attempts in 60 seconds). If threshold is not reached by the end of time-frame, the time window is pushed forward without extending it. Occurrences of positive pattern matches which are older than window are dropped from match count. This is referred to as “sliding window”.

PairWithWindow rule matches two events. If event B does not occur after positive match of event A, the first action is taken. If event B occurs, a second action is taken instead. Two events must occur within defined *Window* which is optional attribute of *Pair* rule and mandatory attribute of *PairWithWindow* rule. *Pair* rule logic is similar, but first action is taken immediately after first event has occurred. *EventGroup* rule functions similar to *Pair,* but multiple events can occur within defined time window in any sequence. Rule matches repeated occurrences of patterns P(1),...P(n) to identify events E(1),...E(n). If N event matches occur within T seconds, an action is taken. Sliding window is taken into account.

Final set of rules provide configuration flexibility. *Suppress* rule matches events without executing any action, and prevents them from being matched by later rules in the same configuration file. This allows exceptions to be made to certain occurrences of later rules,

for example HTTP POST queries from company internal network would not be matched. Same pattern from remote IP address would generate an event which could be used to block that IP. *Calendar* rule executes action when defined *time* matches system clock, providing analogous functionality to UNIX cron daemon. The practical value of such functionality would be periodic processing of information collected by other rules. Instead of forwarding that information into external buffer which could then be accessed by scripts executed by crontab, the same operation can be carried within single configuration set. This would reduce complexity within SEC integration. *Jump* rule matches patterns and delegates processing of those patterns to files defined within configuration set. *Options* rule defines the configuration set into which the configuration file is added. Thus it would be possible to create hierarchically structured configuration.

NXLog version 2.7 provides module *pm_evcorr* that is inspired by SEC [52]. Events are processed by *processor* and forwarded to NXLog *output* module. As of April 2014, the following rule types are supported:

- Simple;
- Suppressed;
- Pair;
- Absence;
- Thresholded;
- Stop.

Simple is equal to NXLog *Exec* statement that matches a pattern from syslog variable. *Suppressed* matches first occurrence of pattern and ignores consecutive iterations within configured *Interval* similar to *SingleWithSuppress* SEC rule. *Pair* rule requires *TriggerCondition* and *RequiredCondition* to be evaluated TRUE within set *Interval* for *Exec* statements to be executed. Unlike SEC version of *Pair* rule, NXLog does not carry out action when only first condition is met. *Absence* rule only executes action if *RequiredCondition* is not evaluated as TRUE. Therefore two rules have to be written to achieve the same functionality as SEC. *Thresholded* is analogous to *SingleWithThreshold* rule while *Stop* is analogous to *Suppress*. All rules except *Simple* and *Stop* support the use

of contexts similar to SEC. NXLog, unlike SEC, allows *TimeField* to be specified, allowing offline log analysis to be conducted. Several complex options like multiple thresholds or matching of asynchronous events are not present in NXLog. Offline event generation could also be conducted with SEC by using UNIX "pipe" to read lines from log file. This is because syslog time-stamp value is irrelevant in the event correlation logic (the time of receiving the message according to system clock is used).

While *pm_evcorr* module does provide some functionality that is already present in SEC, the overall feature set is weak. Therefore NXLog 2.7 cannot compete with SEC 2.7.5 in event correlation functionality.

3.2.2. Configuration

This section provides an overview of SEC [28] and NXLog *pm_evcorr* [52] module configuration syntax. SEC can be configured to read log messages from external source, match events by regular expression and take actions upon positive match.

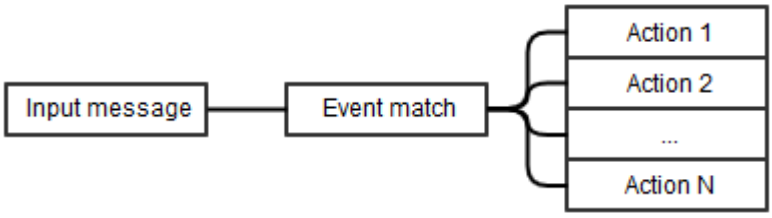

Diagram 2. SEC configuration flow

As illustrated in Diagram 2, SEC makes it trivial to define correlation rule and multiple actions that should be taken upon match. NXLog *pm_evcorr* module must be defined as *processor* directive. Diagram 3 illustrates configuration flow on NXLog modules.

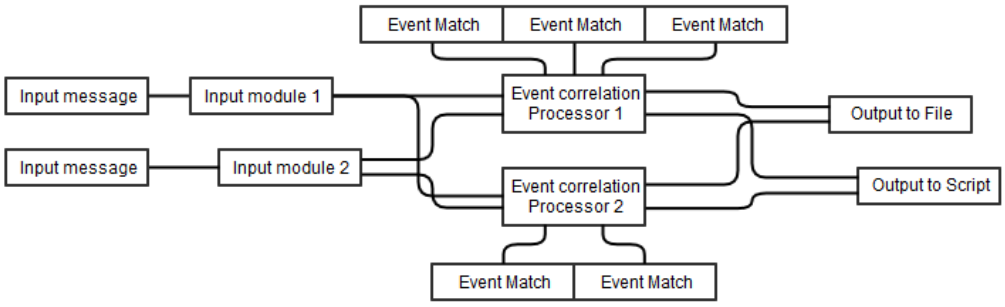

Diagram 3. NXLog pm_evcorr configuration flow

NXLog event correlation rule configuration complexity is higher because SEC is designed to read input from existing syslog infrastructure while NXLog consolidates log collection and event correlation.

As evidenced by Diagram 2 and Diagram 3, utilization of single solution would not reduce configuration complexity. This is due to the fact that log collection and event correlation remain different functional entities. On the contrary, consolidation of two functions into single daemon could have adverse effect on manageability. Changes in one can directly cause problems in second.

The author shall now proceed by concluding the performance tests and analytical comparison, and choosing monitoring solution elements.

3.3. Solutions selected for the monitoring system

The following subsection will conclude the performance tests and analytical comparison, and present the components which were selected for the monitoring solution. Every syslog daemon can be utilized within heterogeneous network environment. Rsyslog version 7 would be the best choice when no complex configuration or message filtering is required, while version 5 could also adequately handle the task. For example, on syslog clients or

relay servers. This is due to the fact that the best course of action when implementing centralized logging environment is to collect all log data from local device and forward that data to central server. Complex filters would be configured on that server. Any compared version of Syslog-ng would be an excellent choice for that task. Processing time reliability, scalability across multiple CPU cores, dynamic filtering efficiency and good configuration syntax makes it the best choice for central log collector. The only downside when compared to Rsyslog and NXLog is lack of support for Elasticsearch and GELF protocol which would make future addition of log analysis tool more difficult.

NXLog, while providing excellent feature set, proved to be underwhelming in performance. Configuration syntax is structured better than Rsyslog but is outclassed by Syslog-ng. At the time of writing, the feature set of NXLog *pm_evcorr* module is lacking compared to SEC. If more rule types would be introduced, then NXLog could become viable alternative to Syslog-ng solution. NXLog message processing would have to be optimized in this case because main benefit of choosing *pm_evcorr* would be performance benefits from utilization of multiple CPU threads, and the fact that NXLog is written in C while SEC is written in Perl. Replacing Rsyslog or Syslog-ng central server with NXLog at current time would mean severe performance drop in message processing. Despite not being chosen as element of log monitoring solution, NXLog could still be utilized as log collector from Windows hosts once they are integrated into the monitoring system.

4. Implementation

This chapter describes the implementation of centralized logging infrastructure with real time event correlation functionality that is capable of automating defensive actions against common attack patterns. Firstly the author shall give an overview of services within the infrastructure, then the chapter will proceed to describe the actions needed to interface existing Linux daemons with syslog infrastructure, and finally an analysis of log trails from common attacks along with possible countermeasures will be conducted.

4.1. Infrastructure overview

This thesis is written from a perspective of small company that provides variety of IT services. More specifically, the two currently employed administrators are directly responsible for the continuous operation of following customer facing systems:

- Domain Name System;
- E-mail;
- Web hosting;
- Hosting of internally developed web services.

Spin TEK AS is an authorized second level domain (SLD) registrar of Estonian top level domain (TLD) with “dot EE” suffix. Other SLD-s are also registered via external service providers. On technical level those zones are published via primary and secondary BIND9 DNS servers. Due to legacy reasons those servers are at current time configured to allow unrestricted recursive name resolution which exposes them to DNS amplification and “cache poisoning” attacks [55]. This configuration, commonly referred to as “Open

resolver", is currently accepted risk by management.

Company provides electronic mail service on two mailbox servers. Both are based on Linux Postfix mail daemon with web interface, and allow e-mails to be downloaded via Dovecot POP3 daemon. First is legacy server with Openwebmail interface that has proven to be difficult to deprecate due to large number of client mailboxes and mailing lists. An intrusion into arbitrary device within trusted network could be escalated to send a flood of mass emails from that server [56]. Second server is configurable via ISPConfig3 interface – customers are provided an administrative account which allows them to manage user mailboxes for their organization. E-mails can be read with Roundcube web interface, or downloaded via Dovecot daemon which also provides IMAP support.

Four servers are used to host client web sites on LAMP (Linux, Apache, MySQL, PHP) stack. Three of them are in the process of deprecation, but continue to provide services due to legacy reasons. Clients can upload code via PureFTP or Vsftp daemon and manage their databases with PhpMyAdmin interface. Web application security risk is high due to the fact that provided PHP version is deprecated but functionality of applications depends on it. Internally developed web services are hosted on separate servers from standard web hosting, but are also written in PHP scripting language. Back-end is stored within PostgreSQL databases.

In addition to external services, the administrators are also responsible for the management of Windows domain controller, Terminal servers, VmWare ESXi virtual machine hosts and Samba file server. Every Linux server is also configured with SSH and SNMP daemons for management purpose. To consolidate resources, several small companies employ outsourcing business model which can be abstractly divided into three categories:

- Physical devices are hosted in server rooms where networking aspects are not controlled by device owner;
- Virtual servers are acquired from cloud providers;
- Combination of two with virtualized server infrastructure on physical devices that are hosted within service provider compound.

The employer of author currently uses the third option for the hosting of their production infrastructure. Servers are physically located at server rooms of a major ISP. Central firewall in this case is simple Linux server on legacy hardware that is used as default gateway and traffic filter for Linux based virtual machines. Network switches belong to the Spin TEK, and are separated from other ISP infrastructure by central firewall, but lack advanced port mirroring capability that would allow for network traffic monitoring.

A common approach to managing security risks and legacy systems within production environments is “Defense-in-depth”, which dictates a layered security mechanism. If one security layer fails then others can still protect the system [57]. The ability to collect information from multiple hosts and utilize real-time event filtering would serve as another layer within existing defense strategy. IDS systems, like Snort, have the ability to output alerts to syslog, allowing them to be later interfaced with existing logging infrastructure. Author will now proceed to describe the centralized log monitoring infrastructure.

4.2. Centralized logging and event correlation infrastructure

This chapter describes the implementation of centralized logging infrastructure with real-time event correlation functionality. Remote hosts will be configured to forward everything to central server, any message filtering will be configured there. This is to reduce the management overhead of syslog daemon configurations.

4.2.1. Certificate authority

In order to assure the confidentiality and integrity of transmitted event messages, both syslog server and client must be configured to use authenticated TLS transport functionality in accordance with RFC5424-5425 [15][16]. A common caveat in syslog TLS configuration is the lack of mandatory authentication which leaves messages vulnerable to MitM attacks, and can allow unauthorized devices to send event messages to central server.

An example MitM attack scenario would be “ARP poisoning” where all traffic destined to syslog server is sent to malicious server within the network instead. That server could intercept the messages and forward them to actual syslog server, potentially in modified form. This can be mitigated by using *peer-verify* directive for Syslog-ng or *$ActionSendStreamDriverAuthMode* for Rsyslog. The latter also allows common name (CN) field within certificate to be checked with *x509/name* directive. CN field of peer certificate must match Fully Qualified Domain Name (FQDN) within configuration file in addition to certificate being signed by trusted party, otherwise the connection will be rejected.

Internal CA with self-signed certificates is used since chain of trust between peers has already been established. Server and clients are part of common infrastructure and managed by same administrators. Key and certificate are generated on central server and then distributed between nodes. OpenSSL utility was used because certificate generation scripts were already available from previous project. Alternatively, GnuTLS provides *certtool* utility that consolidates actions needed to configure internal CA, but it proved to be more complex to use for batch certificate generation than OpenSSL. CA creation technical documentation is provided in [54].

4.2.2. Syslog server

Syslog-ng 3.3 was chosen as syslog daemon for central log server, since as argued in Chapter 3, its configuration language is powerful and flexible, allowing for the creation of complex configurations which are easy to read and maintain. Furthermore, although Rsyslog is the most efficient syslog server in terms of consumed CPU time, Syslog-ng provides adequate performance and scales well to multiple CPU cores. Finally, the author plans to use dynamic logfile names extensively in configurations, and Syslog-ng provided the best performance in terms of wall-clock time for this particular scenario (in other words, Syslog-ng offered the best message throughput). Following filtering logic is used:

- All log files are stored under single directory "*/var/log/server/*";
- Log file name format is "*$facility_name.$severity_name.log*";
- *Daemon* and *Local* channels are used for any program that does not have a dedicated syslog facility, therefore those logs are stored within sub folder named "*/var/log/server/$facility_name*";
- *Daemon* and *Local* log file naming format is "*$application_name. $severity_name.log*".

Although Syslog-ng 3.5 is an improvement over version 3.3, it is not included within Ubuntu Server 12.04 LTS repositories, so the author decided to use standard Syslog-ng 3.3 that is provided with Ubuntu 12.04. Upgrade to server version 14.04 with Syslog-ng version 3.5 would be trivial since, aside from event correlation toolset, no custom software will be installed on the syslog virtual machine. Ubuntu provides *do-release-upgrade* command that automates distribution upgrade process. Syslog-ng server technical documentation is provided in [54].

4.2.3. Syslog client

Every syslog daemon discussed in Chapter 3. is RFC5424-5425 compatible which means that any of these daemons can be used on the client side. No complex configuration is required, since client is tasked with establishing authenticated TLS encryption tunnel to central server, and forwarding all messages from local syslog channels there. Rsyslog version 5 is already packaged to most servers that are to be interfaced with logging infrastructure while performance tests proved that version to be adequate for the task at hand. Legacy syntax is used to facilitate the use of single configuration template across multiple Rsyslog versions. Syslog-ng was used on clients where Rsyslog installation proved to be problematic (technical difficulties in attaining operational status, most likely a fault in OS deployment). Syslog client deployment guide is provided in [54].

Implementation process revealed that some legacy devices are unable to utilize RFC5425 TLS standard. Syslog daemons are capable of creating network listeners on multiple ports allowing parallel collection of encrypted and unencrypted message streams. To ensure message confidentiality and integrity, a private management network should be used. If the device can be configured with encrypted VPN tunneling, such as OpenVPN or Stunnel, then BSD syslog traffic could be forwarded through that channel instead. An example OpenVPN configuration that enables the use of point-to-point encrypted tunnel between 2 hosts is provided in [54].

4.2.4. Application specific log collection

Most daemons log event messages via syslog, but some write messages directly into custom text file. Others might require additional configuration to ensure that relevant event messages are generated for audit purposes. In accordance with infrastructure overview within Chapter 4.1. the following application logs must be collected:

- BIND9;
- Postfix;
- Dovecot;
- Roundcube;
- ISPConfig3;
- Apache2;
- MySQL;
- PostgreSQL;
- Samba;
- SSH;
- Net-SNMP;
- Pure-FTP;
- Vsftp.

BIND9 query and zone transfer logs, which can be configured via *logging* clause, provide

the most information regarding possible attacks. Roundcube and ISPConfig3 log user authentication to files that can be collected using Rsyslog or Syslog-ng file input module. Apache2 can be configured to forward error logs to syslog but that functionality is missing from version 2.2 access logs. Log files are usually configured separately for each virtual host on server. Collection with file input module can be prone to configuration errors since each log file collection must be configured separately, but virtual host directives allow log messages to be "pipelined" to Linux *logger* utility. Samba file server allows audit logs to be generated that store information regarding file access and modification. Postfix, Dovecot, SSH and Net-SNMP utilize syslog by default. Pure-FTP logs entries by default to syslog while this functionality must be enabled for Vsftp. Vsftp authentication can also be audited via *authpriv* facility. SQL servers are currenlty left outside the scope of this thesis due to the fact that they can only be accessed externally via Apache web server. Technical documentation regarding log collection from listed daemons presented in [54].

4.2.5. Event Correlation framework

SEC 2.7.5 [28] was chosen as event correlation engine for its wide acceptance in industrial and academic communities. Diagram 4 depicts a centralized logging infrastructure with event correlation engine that processes input from log files on central server.

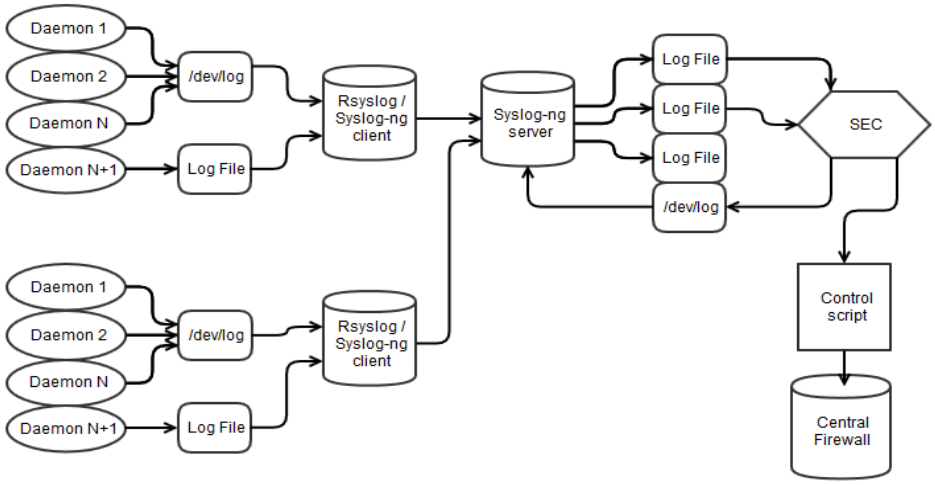

Diagram 4. Event generation flow

As illustrated in Diagram 4, every event match is fed back to the system via syslog to provide audit trail, and forwarded to control script that can be used to automate defensive actions. This allows SEC to combine passive IDS and active IPS functionality, while the use of external script allows for error handling and functionality additions without complicating rule configuration (OS commands could alternatively be written in SEC rules). This would allow better control over active defense components. Since logs are sorted by syslog facility and priority, any log file that is known to contain no interesting event messages can be ignored to avoid redundant event processing. If log management solution such as Graylog2 or Kibana were later to be added into the system, then SEC alerts could be presented on central dashboard. Diagram 5 depicts SEC configuration logic.

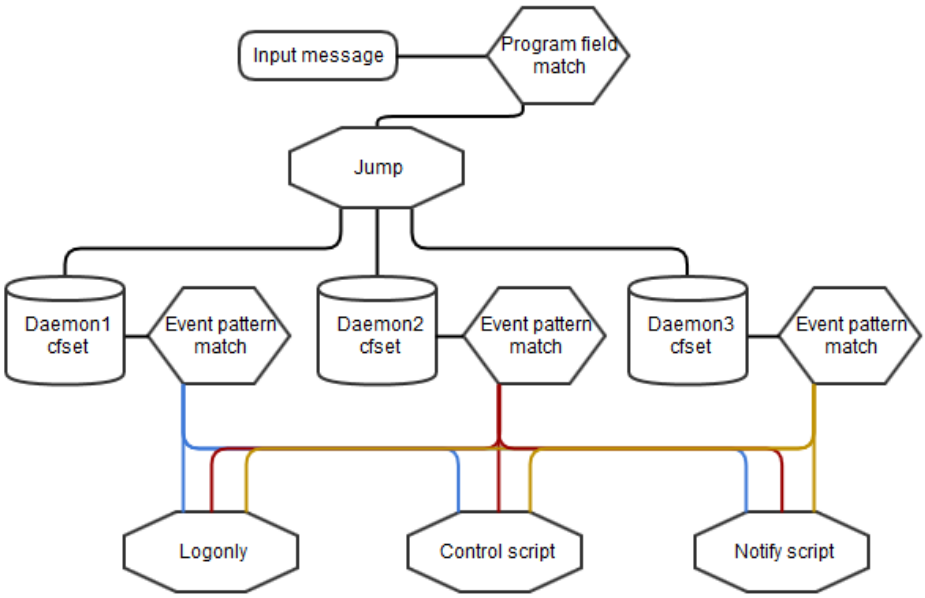

Diagram 5. SEC ruleset framework

As displayed in Diagram 5, combination of SEC *Jump* and *Options* rules will be used. The solution is designed to provide real-time log monitoring on per-daemon basis – each Jump rule matches one particular daemon name in incoming log messages, and directs matching messages to relevant SEC rules for that specific daemon name. This hierarchical rule arrangement allows for saving substantial amount of CPU time, since the matching of irrelevant rules for other daemons is never attempted. Furthermore the same hierarchical

arrangement can be used to match event types that are described in Chapter 4.3. The data fields that are extracted from event message are passed as arguments to script that takes corrective actions on affected host or central firewall via SSH channel. An optional notification script can also be used as redundant alert channel. The logic behind SEC rules shall be covered within the next section.

4.3. Log pattern analysis and event generation

In this chapter, the author shall analyze attack footprints within log data to create SEC rules for filtering of common attacks against daemons described within Chapter 4.1. During the following discussion, the author uses log data of Spin TEK AS where a session of penetration tests was conducted against selected web applications. These tests generated event messages which would then be used by author to test regular expressions in SEC rules.

Due to the scope of thesis and time limitations, a selection of attack vectors was made by author for creation of initial SEC rules. This includes handling of authentication attack, web application injection and DNS server events. Up to date set of rules and control scripts are maintained by author in [54].

4.3.1. Authentication failures

A common forensic trail within log files are exhaustive searches for valid credentials. In those cases an attacker attempts to gain access to system by trying different user name and password combinations. Whether this is conducted by random generation (“brute force” attack) or by existing word-list (dictionary attack) is irrelevant from the log view – take the following Pure-ftp log record as an example:

```
Apr 17 03:17:50 <hostname> pure-ftpd: (?@<REMOTE_IP>) [WARNING]
Authentication failed for user [<user>]
```

Any system daemon with authentication functionality can produce log records of these events. One could create correlation rules for each system daemon or, to keep SEC configuration files at manageable levels, convert all different authentication failure events into a single canonical format, and write more complex event correlation rules for this format since peculiarities of each individual message are irrelevant once the same processing scheme is applied for all messages. In SEC, this functionality can be achieved by matching events of interest, and generating synthetic events in a canonical format. The following example rule generates a synthetic event for *pure-ftpd* authentication failure:

```
type=Single
ptype=RegExp
pattern=(?<server>\w+) pure-ftpd: \(\?@(?<remote_IP>\S+)\)\s+\[WARNING\]
Authentication failed for user \[(?<username>\S*)\]
desc=$+{server} | $+{remote_IP} | FTP authentication failure
action=logonly; event FTP_AUTH_FAILURE_$+{server}_$+{remote_IP}_$+
{username}
```

Synthetic events are processed by SEC similar to input event messages, allowing pattern matching to be applied on processed synthetic event rather than arbitrary log message. *FTP_AUTH_FAILURE* event can be generated by analogous rule for Vsftp failure messages, with relevant information encoded in the event text in easily extractable form. In the case of the above example, relevant information includes the name of the affected server, the IP address of the remote host, and the user name that was involved in the attempt. Additionally, *logonly* directive is used to generate log record of the matching event for debugging purposes. The simplest correlation rule to be applied to generated synthetic event would be *SingleWithThreshold* configured for small time window which might identify primitive brute-force attacks, but fails to detect botnet nodes with systematic approach – attacker can evade simple filters by limiting the attempt rate to smaller value than what is configured within the rule. Essentially, the attacker is most likely an automated botnet node that is not concerned whether goal is achieved in one minute or one year. The following rule identifies single IP address that fails to log in with five distinct user names within 900 seconds.

```
type=EventGroup
init=create USER_COUNTING_$+{remote_IP}_$+{server}
end=delete USER_COUNTING_$+{remote_IP}_$+{server}
ptype=RegExp
pattern=\S*AUTH_FAILURE_(?<server>[\w-]+)_(?<remote_IP>\S+)_(?
<username>\w+)
continue=TakeNext
context=!COUNTED_$+{remote_IP}_$+{server}_$+{username}
count=alias USER_COUNTING_$+{remote_IP}_$+{server} COUNTED_$+
{remote_IP}_$+{server}_$+{username}
desc=$+{remote_IP} | $+{server} | Authentication failures from single IP
with 5 distinct usernames
action=logonly; event ATTACK_FROM_$+{remote_IP}
window=900
thresh=5
```

Please note that for matching the daemon name, the regular expression in *pattern* field matches it with *\S* as* any non white-space string. This allows common events across multiple services, like Dovecot, SSH, Roundcube, ISPConfig3 or FTP daemons, to be handled by single rule. SEC context is created for every unique combination of remote IP, affected server and user name. Even a small number of such events can be interpreted as malicious activity. The above rule generates *ATTACK_FROM* synthetic event for a malicious remote host which can then be matched with consecutive rule that blocks the remote host. *Continue* parameter with this event handling logic must be set to *TakeNext,* otherwise SEC would stop processing the matching event once the current rule has handled it. The fact that rule has matched an event does not necessarily mean that *ATTACK_FROM* output event is eventually generated, since the correlation operation might not observe enough events for fulfilling the threshold condition in 15 minutes. For example, an attacker might attempt to use single user name against multiple system daemons. Another rule could be written for that case, which identifies remote hosts that fail to authenticate against several distinct daemons. Long term systematic attacks could alternatively be identified by creating *MULTIPLE_AUTH_ATTEMPTS* event from correlation of multiple failures over long (for example, one week to one year) time period. Each level of correlation reduces the chance of encountering a false positive, which is especially important in the context of automated incident handling.

4.3.2. Web application injections

Another group of services integrated into centralized logging and event correlation system are commonly known as web applications, which were chosen due to their popularity as a target for cyber criminals. The large amount of elements used in web application development makes identification of malicious queries difficult – technical aspects of these threats are, in large part, not standardized, and are constantly evolving. OWASP maintains a list of 10 most critical vulnerabilities in web applications with "injections" being the highest rated in the 2013 version of the document. Injection occurs when untrusted data is sent to the interpreter as part of command or query [36] which can then cause unexpected behavior in application. Malicious user can gain a certain amount of control over the web application or server. Identification and exploitation of such vulnerabilities are usually conducted using trial-and-error methodology by sending known application-breaking injections and observing the results. Several automated tools [58], such as Burp Suite, Nessus or SQLMap, can be used to speed up that process for the attacker, while specialists of this field usually prefer manual testing or use of custom tools. Both approaches can generate log trail with visible injected parameters. As such, injection attempts, successful or not, can be used on first level of correlation similar to failed authentication attempts in Chapter 4.3.1. The following SEC rule snippet identifies OS command injection commonly known as "directory traversal":

```
ptype=RegExp
pattern=(?<server>[\w-]+) apache:\s?(?<vhost>\S+)? (?
<remote_IP>(?:\d{1,3}\.){3}\d{1,3}).+(?:(?i:\.|\%2e){2,}(?:(?i:/|\\|\%2f|
%c0%af)+))+
```

Regular expression matches sequential occurrence of two or more dot symbols that is immediately followed by one or more slash or backslash, while taking into account some forms of encoding that might be used to evade filters. Upon positive match of such pattern, a synthetic event *OSCOMMAND_INJECTION_WEBAPP_* is generated with affected server, remote IP and targeted Apache2 virtual host attached as unique suffix. Injection events can then be correlated by subsequent rules. In the case of automated web application scan, the occurrence of such patterns in relatively small time-frame, no more that 10

seconds, is generally large enough to be handled by *SingleWithThreshold* rule. Attacker could reduce the number of event matches by utilizing filter evading techniques such as encoding the payload. In the process of exploitation, a small number of alerts can still be triggered that can then be cross-correlated with various information sources. The following SEC rule cross-correlates injection events with authentication failures from common remote IP address:

```
type=EventGroup2
continue=TakeNext
ptype=RegExp
pattern=\S*INJECTION_WEBAPP_(?<server>[\w-]+)_(?<remote_IP>\S+)_\S*
thresh=15
ptype2=RegExp
pattern2=\S*AUTH_FAILURE_(?<server>[\w-]+)_(?<remote_IP>\S+)_\w
thresh2=5
desc=$+{remote_IP} | $+{server} | Combined authentication and web
application attacks
action=logonly; event ATTACK_FROM_$+{remote_IP}
window=60
```

EventGroup rule is used since events can occur at any order within common time window, the exact injection type is considered to be irrelevant in the context of this rule. Offender can attempt to gain access to system via different attack vectors, if even a small number of alerts from multiple vectors are detected then probable cause has been established to interpret these messages as malicious activity directed against server. Extracted data in the form of remote IP address can then be used to block access to all systems. Other methods can include counting of distinct injection types that are detected from single IP address or, if alerts from NIDS device were to be added into log monitoring system, cross-correlation of detected injections with Snort alerts from common remote IP. The purpose would be to reduce false positives and pinpoint malicious events from general background noise. Cross-correlation with IDS alerts is outside the scope of this thesis.

4.3.3. DNS server events

Rules were developed to mitigate risks caused by hosting a public DNS server. For example, consider the following DNS query log message entry from BIND9 authoritative

server with split-view configuration.

```
Apr 22 06:50:34 ns3 named[20543]: queries: info: client
<REMOTE_IP>#11578: view authoritative: query: <FQDN> IN ANY +E
(<LOCAL_IP>)
```

The DNS view is named “authoritative”, which means that remote IP address is using the server to conduct recursive queries. Remote IP address represents the victim of the attack. Due to the nature of stateless UDP protocol (which the DNS queries are built upon), the actual source of malicious query can not be identified. Malicious party can forge DNS request packet with falsified source address and server responds to victim IP with answer that is several times larger than request. In large volumes this attack vector can be used to overload victims network connection with UDP packets. The purpose of reflected DNS amplification attack is to generate response that is larger than query, which is the reason why ANY record type is requested – all relevant records to the domain name are sent to victim. Under normal circumstances, a recursive query would only be conducted against specific record type, for example A, MX or NS. If large amount of ANY queries within the time-frame of 5 seconds are identified, an event will be created. Another common footprint is zone transfer attempt.

```
Apr 22 11:07:06 ns3 named[29809]: security: error: client
<REMOTE_IP>#43836: view authoritative: zone transfer '<DNS_ZONE>/AXFR/IN'
denied
```

The zone file transfer feature has been limited by system administrators, which is common practice within DNS server configuration. Any attempt from unauthorized IP address will produce an error message. These messages, while of little use on their own, can be cross-correlated with other security related log messages by using the extracted IP address. DNS zone transfer is a method for reconnaissance that is used by attackers and penetration testers. If such event were to occur within common time window with web application injections or authentication failures from common IP address, then the only logical conclusion would be malicious activity. An *EventGroup* rule similar to one shown in Chapter 4.3.2. can be used for that purpose.

4.3.4. Attack events

Once an attack has been correlated from real-time log data, the next logical step would be to take action against offending IP address. The following SEC rule matches correlated malicious events.

```
type=SingleWith2Thresholds
ptype=RegExp
continue=TakeNext
pattern=ATTACK_FROM_(?<remote_IP>(?:\d{1,3}\.){3}\d{1,3})
desc=$+{remote_IP} | Generic attack: IP added to ban list
action=logonly; event IP_BLOCKED_$+{remote_IP}; shellcmd
/opt/scripts/iptables.sh $+{remote_IP}
window=60
thresh=1
desc2=$+{remote_IP} | Generic attack: IP removed from ban list
action2=logonly; shellcmd /opt/scripts/iptables_remove.sh $+{remote_IP}
window2=60
thresh2=0
```

Malicious IP address will be forwarded to Bash script that will be executed via SSH channel on central firewall. Rule then expects second threshold to be reached. Since threshold is 0, it will always evaluate as TRUE after 60 second time period. Blocked IP address will then be removed from list of banned hosts. Synthetic event *IP_BLOCKED_* with address as suffix will also be created which allows repeat offenders to be permanently blocked by subsequent rule. SEC *SingleWithThreshold* rule was created by author for that purpose that matches 10 such events in the time period of one week. Known trusted IP addresses can be excluded from any potential actions with *Suppress* rule.

This concludes the analysis of elements in preliminary implementation. The author will now proceed to summarize the thesis.

5. Summary

As a result of this thesis, the author implemented a centralized event management and active defense framework as event message monitoring system, and created an open-source repository for distribution of Simple Event Correlator rules. Such system, which combines features from SIEM and IPS, does not require a dedicated appliance. Open-source tools like Rsyslog, Syslog-ng, NXLog and SEC are sufficient for that purpose. It is the Authors opinion that every Linux system administrator should be familiar with these tools.

Analysis of tools revealed that Syslog-ng provides the best configuration syntax and scalability with multiple CPU cores, which resulted in author selecting this software for central syslog server. Rsyslog is also very good log collection tool, and the most efficient in resource usage, but hampered by configuration syntax. NXLog is not yet ready to replace Rsyslog or Syslog-ng on Linux platform, but could become a viable competitor in the subsequent years. NXLog and Syslog-ng provide event correlation feature, but are not yet functionally ready to replace SEC.

While pilot installation of monitoring system is in use within Spin TEK AS, a large amount of work can still be conducted. More rules can be written to identify different attack vectors, and author hopes that other contributors will join the rule-set development project. Identification of malicious patterns could also benefit from application of data mining algorithms. Existing monitoring system could be expanded with NIDS event messages, and log management tool for event message storage and visualization. The overarching event management system could then provide competitive functionality to commercial SIEM solutions.

Sündmuste haldamise ning aktiivse kaitse raamistik väikeettevõttele

Magistritöö kood ITI70LT

Tudeng: Markus Kont
Matrikkli number: 121785IVCMM
Juhendaja: Risto Vaarandi, Ph.D

Resümee

Käesolev magistritöö keskendub enimlevinud logisõnumite kogumise ja korrelatsiooni tööriistade võrdlusele. Eesmärgiks on juurutada sündmuste haldamise ja aktiivse kaitse raamistik väikeettevõttele, mida oleks võimalik rakendada turbe intsidentidele reageerimise automatiseerimiseks.

Autor viis läbi kirjanduse ülevaate, analüütilise võrdluse ning jõudluse testid. Tööriistade võrdlusesse kuulusid kolm syslog formaadis sõnumite kogumise tarkvara – Rsyslog, Syslog-ng ja NXLog. Syslog-ng on parima konfiguratsiooni süntaksiga ning suudab efektiivselt filtreerida miljoneid logisõnumeid. Rsyslog on väga hea tööriist, ning kõige tõhusama sõnumite läbilaske võimega. NXLog on esile kerkiv alternatiiv eelnimetatud syslog deemonitele, mille funktsionaalsuse hulka kuulub ka sündmuste korrelatsiooni moodul, mis on otseselt inspireeritud SEC nimelisest tarkvarast. SEC on maailma kõige populaarsem sündmuste korrelatsiooni tööriist.

Töö tulemusena on Spin TEK AS nimelises asutuses võetud kasutusele kirjeldatud lahenduse pilootinstallatsioon. Autor haldab kirjutatud SEC reegleid, õpetusi lahenduse seadistamiseks ning kontroll skripte avalikus repositooriumis.

List of References

[1] Ghia, A. “Capturing value through IT consolidation and shared services”, McKinsel & Company, 2014, [http://www.mckinsey.com/~/media/mckinsey/dotcom/client_service/Public%20Sector/PDFS/McK%20on%20Govt/IT%20Challenge%20and%20opportunity/MOG7_Consolidation.ashx]

[2] “Security Threat Report”, Sophos, 2011, [http://www.sophos.com/en-us/medialibrary/PDFs/other/sophos-security-threat-report-2014.pdf]

[3] TechNet, “What is a botnet”, Microsoft, Web. May 2014, [http://www.microsoft.com/security/resources/botnet-whatis.aspx]

[4] Kerrisk, M. “DAEMON”, Linux Programmer's Manual, Web. May 2014, [http://man7.org/linux/man-pages/man3/daemon.3.html]

[5] Burnham, J. “Gartner Publishes 2013 Magic Quadrant for Security Information and Event Management (SIEM)”, Security Intelligence, Web. June 11 2013, [http://securityintelligence.com/gartner-publishes-2013-magic-quadrant-for-siem/]

[6] “elQnetworks Survey Reveals Organizations Are Suffering from SIEM Deployments”, elQnetworks, Web. March 6 2013, [http://www.eiqnetworks.com/news-events/press-releases?pr=eiqnetworks-survey-reveals-organizations-are-suffering-from-siem-deployments]

[7] Vaarandi, R. Nizinski, P. “Comparative Analysis of Open-Source Log Management Solutions for Security Monitoring and Network Forensics”, CCDCOE, 2013, [http://ristov.users.sourceforge.net/publications/eciw13-logman.pdf]

[8] TechNet, “Security Threats”, Microsoft, 2013, [http://technet.microsoft.com/en-us/library/cc723507.aspx]

[9] Brown, D. “Active security or: How I learned to stop worrying and use IPS with incident handling”, SANS Institute, 2013, [https://www.sans.org/reading-room/whitepapers/incident/active-security-or-learned-stop-worrying-ips-incident-handling-34465]

[10] Crist, J. “Web Based Attacks”, SANS Institute, 2007, [https://www.sans.org/reading-room/whitepapers/application/web-based-attacks-2053]

[11] Hernandes, J. “Security information and Event Management: Business Benefits and Security, Governance and Assurance Perspectives”, ISACA, 2010, [https://www.academia.edu/attachments/4903624/download_file]

[12] “Syslog”, Web. March 2014, [http://en.wikipedia.org/wiki/Syslog]

[13] Lonvick, C. “The BSD syslog Protocol”, IETF, 2001, [http://tools.ietf.org/html/rfc3164]

[14] Eaton, I. “The Ins and Outs of System Logging Using Syslog”, SANS Institute, 2003 [http://www.sans.org/reading-room/whitepapers/logging/ins-outs-system-logging-syslog-1168]

[15] Gerhards, R. “The Syslog Protocol”, IETF, 2009, [http://tools.ietf.org/html/rfc5424]

[16] Miao, F. Ma, Y. Salowey, J. “Transport Layer Security (TLS) Transport Mapping for Syslog”, IETF, 2009, [http://tools.ietf.org/html/rfc5425]

[17] Okmianski, A. “Transmission of Syslog Messages over UDP”, IETF, 2009, [http://tools.ietf.org/html/rfc5426]

[18] Gerhards, R. Lonvick, C. “Transmission of Syslog messages over TCP”, IETF, 2012, [https://tools.ietf.org/html/rfc6587]

[19] Pallagi, Z. “Measuring log collection performance”, Balabit, Web. January 2014, [https://pzolee.blogs.balabit.com/tag/syslog-ng/]

[20] Cristias, P. “Linux System Administration, SYSLOGD”, 1998, Web. March 2014, [http://unixhelp.ed.ac.uk/CGI/man-cgi?syslogd+8]

[21] Lang, D. “Building a 100K log/sec logging infrastructure”, LISA12, 2012, [https://www.usenix.org/system/files/conference/lisa12/lisa12-final-34.pdf]

[22] Toprak, M. “Intrusion Detection System Alert Correlation with Operating

System Level Logs", MSc thesis, 2009, [http://library.iyte.edu.tr/tezler/master/bilgisayaryazilimi/T000204.pdf]

[23] Abad, C. Taylor, J. Sengul, C. William, Y., Zhou, Y. Rowe, K. "Log Correlation for Intrusion Detection: A Proof of Concept", IEEE, 2003, [http://speech.mty.itesm.mx/~jnolazco/cursoSeguridad/IDS_logCorr.pdf]

[24] Zirkel, W. Wirtz, G. "Proactive Problem Management and Event Correlation" [http://www.ksi.edu/seke/Proceedings/seke11/10_Werner_Zirkel.pdf]

[25] Crowell, C. "Event Correlation and Root Cause Analysis", CA, 2004, [http://www.genesiscom.info/de/2/products/downloads/event_correlation_and_root_cause_analysis.pdf]

[26] Sweeny, J. "Creating Your Own SIEM and Incident Response Toolkit Using Open Source Tools", SANS Institute, 2011, [https://www.sans.org/reading-room/whitepapers/incident/creating-siem-incident-response-toolkit-open-source-tools-33689]

[27] Vaarandi, R. "Tools and Techniques for Event Log Analysis", PhD thesis, 2005, [http://ristov.users.sourceforge.net/publications/thesis.pdf]

[28] Vaarandi, R "sec", Manual pages, Web. January 2014 [http://simple-evcorr.sourceforge.net/man.html#lbAY]

[29] Vaarandi, R. Grimaila, M.R. "Security Event Processing with Simple Event Correlator", ISSA Journal, 2012, [http://ristov.users.sourceforge.net/publications/sec-issa2012.pdf]

[30] Myers, J. Grimaila, M.R. Mills, R.F. "Log-Based Distributed Event Detection Using Simple Event Correlator", Proceedings of the 44th Hawaii International Conference on System Sciences, 2011, [http://www.computer.org/csdl/proceedings/hicss/2011/4282/00/02-02-10.pdf]

[31] Myers, J. "A Dynamically Configurable Log-Based Distributed Security Event Detection Methodology using Simple Event Correlator", MSc thesis, 2010, [http://www.dtic.mil/dtic/tr/fulltext/u2/a523531.pdf]

[32] Anthony, R. "Detecting Security Incidents Using Windows Event Logs", SANS Institute, 2013, [https://www.sans.org/reading-room/whitepapers/logging/detecting-security-incidents-windows-workstation-

event-logs-34262]

[33] Gerhards, R. “rsyslog vs. syslog-ng”, 2012, Web. April 2014, [http://www.rsyslog.com/doc/rsyslog_ng_comparison.html]

[34] New, D. Rose, M “Reliable Delivey for syslog”, IETF, 2001, [https://www.ietf.org/rfc/rfc3195.txt]

[35] Klyne, G. Newman, C. “Date and Time on the Internet: Timestamps”, IETF, 2002, [http://www.ietf.org/rfc/rfc3339.txt]

[36] “OWASP Top 10 – 2013 The Ten Most Critical Web Application Security Risks”, OWASP, 2013, [http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf]

[37] Doshi, N. “Event Correlation”, 2010, Web. March 2014, [http://blogs.splunk.com/2010/09/01/event-correlation/]

[38] Hills, A. Young, G. D'Hoinne, J. “Magic Quadrant for Intrusion Prevention Systems”, Gartner, 2013, [http://www.gartner.com/technology/reprints.do?id=1-1OAVJS3&ct=131217&st=sb]

[39] Brennan, M.P. “Using Snort For a Distributed Intrusion Detection System”, SANS Institute, 2002, [http://www.sans.org/reading-room/whitepapers/detection/snort-distributed-intrusion-detection-system-352]

[40] White, J. S. Fitzsimmons, T. T. Matthews, J. N. “Quantitative Analysis of Intrusion Detection Systems: Snort and Suricata”, 2013, [http://people.clarkson.edu/~jmatthew/publications/SPIE_SnortSuricata_2013.pdf]

[41] Yamada, A. Miyake, Y. Takemori, K. Studer, A. Perrig, A. “Intrusion Detection for Encrypted Web Access”, 2007, [http://www.researchgate.net/publication/4250251_Intrusion_Detection_for_Encrypted_Web_Accesses/file/60b7d51841f19c6505.pdf]

[42] Esler, J. “SSL/TLS”, Snort manual, 2012, Web. March 2014, [http://manual.snort.org/node147.html]

[43] “enterprise-log-search-and-archive”, Web. March 2014, [https://code.google.com/p/enterprise-log-search-and-archive/]

[44] Knowlage Base, Graylog2, Web. March 2014, [http://support.torch.sh/help/kb]

[45] Kibana, Web. March 2014, [http://rashidkpc.github.io/Kibana/]

[46] Lang, D. “Log Filtering with Rsyslog”, 2013, [http://www.sclug.org/sites/www.sclug.org/files/presentations/rsyslog_filtering.pdf]

[47] “Rsyslog v5-stable documentation”, Web. May 2014, [http://www.rsyslog.com/doc/v5-stable/]

[48] “Rsyslog v7-stable documentation”, Web. May 2014, [http://www.rsyslog.com/doc/v7-stable/]

[49] “Rsyslog 8 documentation”, Web. May 2014, [http://www.rsyslog.com/doc/master/]

[50] “The Syslog-ng Open Source Edition 3.3 Administrator Guide”, BalaBit, 2014, Web. May 2014, [http://www.balabit.com/sites/default/files/documents/syslog-ng-ose-3.3-guides/en/syslog-ng-ose-v3.3-guide-admin-en/html-single/index.html]

[51] “The Syslog-ng Open Source Edition 3.5 Administrator Guide”, BalaBit, 2014, Web. May 2014, [https://www.balabit.com/sites/default/files/documents/syslog-ng-ose-3.5-guides/en/syslog-ng-ose-v3.5-guide-admin/html-single/index.html]

[52] Botyabzky, B. “NXLOG Community Edition Reference Manual for v2.7.1189”, Web. May 2014, [http://nxlog.org/nxlog-docs/en/nxlog-reference-manual.html]

[53] Churilin, A. “Choosing an Open-Source Log Management System for Small Business”, MSc thesis, 2013, [http://lab.cs.ttu.ee/dl135]

[54] Kont, M. “SagittariuSEC”, [https://github.com/markuskont/SagittariuSEC]

[55] Kak, A. “DNS and the DNS Cache Poisoning Attack”, 2014, [https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture17.pdf]

[56] Tschabitscher, H. “Spam”, Web. May 2014, [http://email.about.com/library/weekly/aa100697.htm]

[57] “Defense in depth”, OWASP, Web May 2014, [https://www.owasp.org/index.php/Defense_in_depth]

[58] Lydon, G. “SecTools.Org: Top 125 Network Security Tools”, Web. May 2014, [http://sectools.org/]

Appendices

Appendix 1. Syslog daemon compilation options

```
# Rsyslog 5.8
./configure \
        --prefix=/opt/rsyslog5 \
        --enable-extended-tests=yes \
        --enable-imptcp \
        --enable-imttcp \
        --enable-impstats \
        --enable-omstdout \
        --enable-imfile \
        --enable-mmsequence \
        --enable-kmsg \
        --enable-gnutls
# Rsyslog 7,6
./configure \
        --prefix=/opt/rsyslog7 \
        --enable-extended-tests=yes \
        --enable-imptcp \
        --enable-imttcp \
        --enable-impstats \
        --enable-omstdout \
        --enable-imfile \
```

```
        --enable-mmsequence \
        --enable-kmsg \
        --enable-gnutls
#Rsyslog 8.1
./configure \
        --prefix=/opt/rsyslog8 \
        --enable-extended-tests=yes \
        --enable-imptcp \
        --enable-imttcp \
        --enable-impstats \
        --enable-omstdout \
        --enable-imfile \
        --enable-mmsequence \
        --enable-kmsg \
        --enable-gnutls
#Syslog-ng 3.3.4
./configure \
        --prefix=/opt/syslog-ng_3.3.4 \
        --enable-ssl=yes
#Syslog-ng 3.5.4.1
./configure \
        --prefix=/opt/syslog-ng_3.5.4.1 \
        --enable-ssl=yes
#NXLog 2.7
/configure \
        --prefix=/opt/NXLog2.7
```

Appendix 2. Unfiltered TCP syslog server configurations

Appendix 2.1. NXLog

```
LogFile /var/log/nxlog.log
LogLevel INFO



<Extension _syslog>
        Module      xm_syslog
</Extension>
<Input in>
        Module      im_tcp
        Host        0.0.0.0
        Port        514
</Input>
<Output out_done>
        Module      om_file
        Exec        if $raw_event =~ /^(?:(?!DONE).)*$/ drop();
        File        "/var/log/done.log"
</Output>
<Output out>
        Module      om_file
        File        "/var/log/test.nxlog27.log"
</Output>
<Route route1>
        Path        in => out,out_done
</Route>
```

Appendix 2.2. Rsyslog 5.8

```
$ModLoad imuxsock # provides support for local system logging
$ModLoad imklog   # provides kernel logging support (previously done by rklogd)
$ModLoad imtcp
$InputTCPServerRun 514
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
$RepeatedMsgReduction off
*.*                     /var/log/test.rsyslog5.log
if $msg contains 'DONE' then /var/log/done.log
```

Appendix 2.3. Rsyslog 7.6

```
$ModLoad imuxsock # provides support for local system logging
$ModLoad imklog   # provides kernel logging support (previously done by rklogd)
$ModLoad imtcp
$InputTCPServerRun 514
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
$RepeatedMsgReduction off
*.*                     /var/log/test.rsyslog7.log
if $msg contains 'DONE' then /var/log/done.log
```

Appendix 2.4. Rsyslog 8.1

```
$ModLoad imuxsock # provides support for local system logging
$ModLoad imklog   # provides kernel logging support (previously done by rklogd)
$ModLoad imtcp
$InputTCPServerRun 514
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
$RepeatedMsgReduction off
```

```
*.*                 /var/log/test.rsyslog8.log
if $msg contains 'DONE' then /var/log/done.log
```

Appendix 2.5. Syslog-ng 3.3

```
@version: 3.3
@include "scl.conf"
options {threaded(yes) ; log_fifo_size(1000);};
source s_local {
        system();
        internal();
};
source s_syslog { tcp(ip(192.168.0.75) port(514) max-connections(25) log_iw_size(2500)
log_fetch_limit(2500));};
destination d_local {
        file("/var/log/test.syslogng3.3.log");
};
destination d_local_done {
        file("/var/log/done.log");
};
filter f_done {
      message("DONE");
};
log {
        source(s_syslog);
        source(s_local);
        destination(d_local);
};
log {
        source(s_syslog);
        filter(f_done);
```

```
        destination(d_local_done);
};
```

Appendix 2.6. Syslog-ng 3.5

```
@version: 3.5
@include "scl.conf"
options {threaded(yes) ; log_fifo_size(1000);};
source s_local {
        system();
        internal();
};
source s_syslog { tcp(ip(192.168.0.75) port(514) max-connections(25) log_iw_size(2500)
log_fetch_limit(2500));};
destination d_local {
        file("/var/log/test.syslogng3.5.log");
};
destination d_local_done {
        file("/var/log/done.log");
};
filter f_done {
      message("DONE");
};
log {
        source(s_syslog);
        source(s_local);
        destination(d_local);
};
log {
        source(s_syslog);
        filter(f_done);
```

```
    destination(d_local_done);
};
```

Appendix 3. Filtered TCP syslog server configurations

Appendix 3.4. NXLog

```
LogFile /var/log/nxlog.log
LogLevel INFO
<Extension _syslog>
   Module      xm_syslog
</Extension>
<Input in>
   Module      im_tcp
   Host        0.0.0.0
   Port        514
   Exec        parse_syslog_bsd();
</Input>
<Input unix>
   Module      im_uds
   uds         /dev/log
</Input>
<Processor filter>
   Module pm_filter
   Condition $raw_event =~ /DONE/
</Processor>
<Output out_sev0>
   Module      om_file
   Exec        if $SyslogSeverityValue > 0 drop();
   File        "/var/log/test.nxlog27.$SyslogSeverityValue.log"
```

```
</Output>
<Output out_done>
   Module      om_file
   Exec          if $raw_event =~ /^(?:(?!DONE).)*$/ drop();
   File        "/var/log/done.log"
</Output>
<Output out>
   Module      om_file
   File        "/var/log/test.nxlog27" + ".facility" + $SyslogFacilityValue + ".severity"  +
$SyslogSeverityValue + ".log"
</Output>
<Route route1>
   Path        in => out,out_done
</Route>
```

Appendix 3.2. Rsyslog 5.8

```
$ModLoad imuxsock # provides support for local system logging
$ModLoad imklog   # provides kernel logging support (previously done by rklogd)
$ModLoad imtcp
$InputTCPServerRun 514
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
$RepeatedMsgReduction off
if $msg contains 'DONE' then /var/log/done.log
$template DynaFile,"/var/log/test.rsyslog5.facility.%syslogfacility%.severity.
%syslogseverity%.log"
*.* -?DynaFile
```

Appendix 3.3. Rsyslog 7.6

```
$ModLoad imuxsock # provides support for local system logging
$ModLoad imklog   # provides kernel logging support (previously done by rklogd)
$ModLoad imtcp
$InputTCPServerRun 514
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
$RepeatedMsgReduction on
$template DynaFile,"/var/log/test.rsyslog7.facility.%syslogfacility%.severity.
%syslogseverity%.log"
*.* -?DynaFile
if $msg contains 'DONE' then /var/log/done.log
```

Appendix 3.4. Rsyslog 8.1

```
$ModLoad imuxsock # provides support for local system logging
$ModLoad imklog   # provides kernel logging support (previously done by rklogd)
$ModLoad imtcp
$InputTCPServerRun 514
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
$RepeatedMsgReduction off
$template DynaFile,"/var/log/test.rsyslog8.facility.%syslogfacility%.severity.
%syslogseverity%.log"
*.* -?DynaFile
if $msg contains 'DONE' then /var/log/done.log
```

Appendix 3.5. Syslog-ng 3.3

```
@version: 3.3
@include "scl.conf"
options {threaded(yes) ; log_fifo_size(1000);};
source s_local {
        system();
        internal();
};
source s_syslog { tcp(ip(192.168.0.75) port(514) max-connections(100) log_iw_size(2500)
log_fetch_limit(2500));};
destination d_local {
        file("/var/log/test.syslogng3.3.facility.$FACILITY_NUM.severity.
$LEVEL_NUM.log");
};
destination d_local_done {
        file("/var/log/done.log");
};
filter f_done {
       message("DONE");
};
log {
        source(s_syslog);
        source(s_local);
        destination(d_local);
};
log {
        source(s_syslog);
        filter(f_done);
        destination(d_local_done);
};
```

Appendix 3.6. Syslog-ng 3.5

```
@version: 3.5
@include "scl.conf"
options {threaded(yes) ; log_fifo_size(1000);};
source s_local {
        system();
        internal();
};
source s_syslog { tcp(ip(192.168.0.75) port(514) max-connections(100) log_iw_size(2500)
log_fetch_limit(2500));};
destination d_local {
        file("/var/log/test.syslogng3.3.facility.$FACILITY_NUM.severity.
$LEVEL_NUM.log");
};
destination d_local_done {
        file("/var/log/done.log");
};
filter f_done {
       message("DONE");
};
log {
        source(s_syslog);
        source(s_local);
        destination(d_local);
};
log {
        source(s_syslog);
        filter(f_done);
        destination(d_local_done);
```

};

Appendix 4. Test termination rule

```
#Usage:
#sec --detach --conf=/opt/sec-2.7.5/conf.sec --input=/var/log/done.log
type=SingleWithThreshold
ptype=RegExp
pattern=DONE
desc=Test done
action=shellcmd (pkill "rsyslog|syslog-ng|nxlog" && sleep 1 && rm /var/log/test.*)
thresh=20
window=60
```