jamk.fi

Implementing situational awareness

Timo Hulkkonen

Master's thesis
May 2016
Technology, communication and transport
Master's Degree Programme in Information Technology

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Description

Author(s)	Type of publication	Date
Hulkkonen, Timo	Master's thesis	30.5.2016
	Number of pages 77	Language of publication: English
		Permission for web publication: x

Title of publication

Implementing situational awareness

Degree programme

Master's Degree Programme in Information Technology

Supervisor(s)

Karjalainen Mika, Hautamäki Jari

Assigned by

Vitec Health, Vitec Software Group

Abstract

Implementation of situational awareness system became an objective when Finnish national regulations for A-class systems, in the context of healthcare systems were updated. The updated criteria require IT service providers to have a capability to detect anomalies in information systems. Situational awareness was selected to meet those criteria because situational awareness can be used as a tool which helps the organization to improve its service.

The objective was to implement a cost effective situational awareness system which fulfills the national requirements and works as a tool for the IT operations team. The objective was outlined in the way that log sources were DNS, software firewall, Windows Event log and Microsoft IIS. Network devices, hypervisor and other services or applications were left out of the scope of this study.

The implementation was carried out by first analyzing different approaches in theory and then the most suitable solution, log based situational awareness, was selected. The approach was then verified with proof-of-concept environment after which the actual situational awareness system was installed and configured for receiving and normalizing logs from multiple sources. After the log flow was verified, the situational awareness system was evaluated by testing and using actual production log data.

Situational awareness has proven to be an extremely valuable tool for IT operations team to support daily monitoring and for detecting anomalies. It is not a silver bullet in any way, and it does not solve all problems; however, it brings good return for investment. The implementation project was successful within the scope of this study.

Keywords/tags (subjects)

Situational awareness, anomaly detection, information security, Windows, IIS, DNS, Firewall, Log

Miscellanous

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Hulkkonen, Timo	Opinnäytetyö	30.5.2016
	Sivumäärä 77	Julkaisun kieli: Englanti
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Implementing situational awareness

Koulutusohjelma

Master's Degree Programme in Information Technology

Työn ohjaaja(t)

Karjalainen Mika, Hautamäki Jari

Toimeksiantaja(t)

Vitec Health, Vitec Software Group

Tiivistelmä

Tilannekuvajärjestelmän implementointi asetettiin tavoitteeksi, koska terveydenhuollon tietojärjestelmien auditointivaatimukset päivittyivät ja A-luokkaan kuuluvia järjestelmiä tarjoavilla tai ylläpitävillä organisaatioilla tulee olla kyky havaita poikkeamia tietojärjestelmissä. Poikkeamia voidaan havaita muillakin menetelmillä, mutta tilannekuvasta nähtiin saatavan hyötyä palvelun parantamiseen organisaatiossa.

Tavoitteena oli implementoida kustannustehokas tilannekuvajärjestelmä, joka täyttää kansalliset auditointivaatimukset ja toimii työkaluna ylläpitotiimille. Tutkimus oli rajattu siten, että lokilähteinä käytettiin DNS-palvelua, ohjelmistopohjaista palomuuria, Windowsin Event logia ja Microsoftin IIS web-palvelinta. Verkon aktiivilaitteet, virtualisointialusta ja muut palvelut tai ohjelmat jätettiin tutkimuksen ulkopuolelle.

Implementointi aloitettiin tutkimalla tilannekuvan muodostamista teoriassa, minkä jälkeen valittiin tutkimukseen soveltuvin lähestymistapa: lokipohjainen tilannekuva. Valinta verifioitiin vielä pystyttämällä demoympäristö (proof-of-concept). Verifioinnin jälkeen asennettiin ja konfiguroitiin varsinainen tilannekuvajärjestelmä. Järjestelmän toimivuutta arvioitiin sekä testaamalla että analysoimalla todellisia tilanteita tuotantoympäristön lokeista.

Toteutettu tilannekuvajärjestelmä on todettu erittäin hyödylliseksi ja arvokkaaksi työkaluksi sekä tukemaan palvelinympäristön valvontaa että poikkeamien havainnointiin. Tilannekuvajärjestelmä ei ole täydellinen eikä sillä voi varmasti havaita kaikkia mahdollisia poikkeamia, mutta se antaa kuitenkin huomattavaa lisäarvoa ja parantaa havainnointikykyä merkittävästi. Implementointiprojekti oli onnistunut määritellyn rajauksen puitteissa.

Avainsanat (subjects)

Tilannekuva, poikkeama, tietoturva, Windows, IIS, DNS, palomuuri, loki

Muut tiedot

CONTENTS

ACRONYMS AND ABBREVIATIONS

CSV	Comma separated value
DNS	Domain Name System
DSRM	Design Science Research Methodology
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IAAS	Infrastructure as a service
IOC	Indicator of compromise
IP	Internet Protocol
IPv4	Internet Protocol version 4
IT	Information Technology
LTS	Long-term Support
NLA	Network-Level Authentication
OS	Operating System
OWASP	The Open Web Application Security Project
SA	Situational Awareness
SAAS	Software as a service
SLA	Service Level Agreement
SQL	Structured Query Language
SQLi	Structured Query Language injection
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
VLAN	Virtual Local Area Network
XML	Extensible Markup Language

1 INTRODUCTION

1.1 Research objective

The purpose of this thesis was to research how to form situational awareness inside Windows Server environment and what benefits current and accurate situational awareness provides. The assigner organization, Vitec Health of Vitec Software Group, provides SaaS-solutions to Finnish healthcare organizations from IaaS-environment. The goal was to implement a situational awareness system which fulfills the information security criteria of national patient data archive for service providers, helps the organization to serve their clients better and give insight to the system administrators about the production environment.

1.2 Vitec Software Group

Vitec Software Group is an international 31-year-old IT company that was founded in 1985 in Umeå, Sweden. Vitec's vision is to be a growth company in the mature part of the software industry. Vitec's mission is to enable customers to maximize their opportunities and to develop and secure their business through business-critical software. (Vitec in brief 2016, 6)

Vitec Software Group has around 450 employees in Scandinavia and France, Vitec Health has around 50 employees in Finland and France. (Vitec in brief 2016, 7)

1.3 Scope

The thesis is outlined for implementing situational awareness from virtualized Microsoft Server environment running on VMware platform. Situational data is gathered from the operating system and application level. The situational awareness of the hardware-level, either server or network device and hypervisor-level, was left out of the scope of this study.

1.4 Structure

The thesis is structured so that the first part presents the basic information about the thesis. The second part presents situational awareness in theory and practice; situational awareness is discussed also outside information technology area. The third part is about evaluating different solutions to implement situational awareness. The fourth part discusses about the infrastructure and components of the situational awareness system. The fifth part discusses the implementation of the situational awareness system. The sixth part is about testing what the situational awareness system can express about the actions in the target environment and how it expresses it. This is presumably the most important part of this thesis because it produces the base for the results and the conclusion of this thesis. The seventh part introduces the results, and the eighth part discusses the overall conclusion. Future research is discussed in the ninth part.

1.5 Research methods

Because of the nature of this thesis, research in information systems area, common research framework used in nature sciences was not suitable here. In the early 2000s, several researchers have succeeded in bringing design science research into the information system research community. Suitable framework for design science research in information systems and a mental model or template for readers and reviewers is necessary to recognize and evaluate the results of such research. The mental model provides the context for researchers in which they can understand and evaluate the work of others. For example, if an empirical paper failed to describe how the data was gathered, the research would be considered not valid or at least incomplete. (Peffers 2007, 2-4)

A Design Science Research Methodology (DSRM) for Information Systems Research is used in this thesis as the research method. Design science aims to create results that can serve human purposes whereas natural and social sciences tries to understand reality. (Peffers 2007, 4)

The DSRM consists of six main activities which are illustrated in Figure 1. The activities are explained in more detail in the following chapters. The following chapters also describe how the activities reflect in this thesis and its implementation.

The DSRM describes a sequential order for activities, however, there is no expectation that the researcher should proceed in the sequential order from activity one to activity six. Depending on the research case at hand one can start with almost any step. In this thesis the approach used in actual implementation is an objective-centered solution where the process is started with activity two. The objective was triggered by Finnish national regulations that demand healthcare system service providers the ability to detect anomalies in information systems. (Peffers 2007, 14)

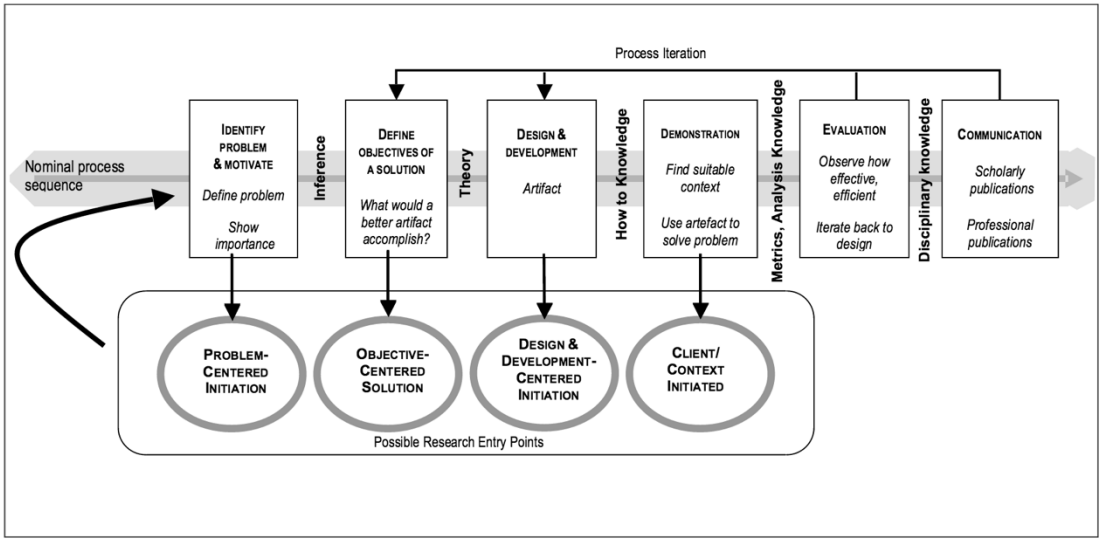

Figure 1. Design Science Research Methodology Process Model (Peffers 2007, Figure 1.)

1.5.1 Identify problem and motivate

Sequentially the first activity is to define the specific research problem and justify the value of a solution. This definition will be used to develop a concrete solution. The value of a solution should also motivate both the researcher and the target audience. (Peffers 2007, 12)

In this thesis the identified problem was that there was a need, emerging from mandatory Finnish national regulations, for the capability to detect anomalies in information systems. The regulation state that an information system has to have a way to detect anomalies in use of system. The requirement is criteria 43 in Information security requirements for A-class systems (Tietoturvavaatimukset A-luokkaan kuuluville järjestelmille ja järjestelmien käyttöympäristöille 2015, 18).

One way to detect anomalies is to use a situational awareness system. Situational awareness is a tool that can also be used for other activities than detecting anomalies. Those activities are discussed in chapter 2.3.4. Because of the many uses of situational awareness system and the mandatory regulations, motivation was easily found.

1.5.2 Define objectives of a solution

The second activity is to define the objectives for a solution. The possible and feasible objectives should be inferred from the problem definition. The objective can be either a better solution than the previous ones or a new solution. (Peffers 2007, 12)

The objective was to implement a situational awareness system in IaaS-based Windows Server environment since no situational awareness solution existed.

1.5.3 Design and development

The third activity is to design and develop the solution. In this step the solution is designed, its functionalities and architecture are decided and the actual implementation takes place. Theory is used to support the move from objective to design and development. (Peffers 2007, 13)

The design and development of the thesis were completed in chapter 4.

1.5.4 Demonstration

The fourth activity is to demonstrate the use of the solution to solve one or more problems. Demonstration could involve experimentation, simulation, case study, proof or other suitable activity. (Peffers 2007, 13)

The demonstration activity in this thesis was completed in chapter 5, where proof of concept environment was built at first using the same principles and methods as in the actual production environment.

1.5.5 Evaluation

The fifth activity is to evaluate, observe and measure how well the solution resolves the problem. (Peffers 2007, 13)

The solution evaluation is discussed in chapter 6 so that the solution was used to visualize anomalies in the production environment. The solution was tested in the test environment to find out how different kind of activities would be detectable with the solution.

1.5.6 Communication

The sixth and last activity is communication, which can be structured as this process or it can use the common structure of an empirical research process (problem definition, literature review, hypothesis development, data collection, analysis, results, discussion, and conclusion). (Peffers 2007, 14)

This thesis uses a structure that combines the nominal structure and the empirical research process.

2 SITUATIONAL AWARENESS THEORY

2.1 History of situational awareness

Although the term situational awareness is fairly recent, the concept's roots are in the history of military theory. The importance of situational awareness was developed during the conflicts in Korea and Vietnam in the 1990s. In command and control situational awareness is required for effective and correct decision-making. Implementing and keeping constant situational awareness is difficult and requires great effort to analyze the incoming information in a rapidly changing environment. (Oosthuizen 2015, 4-5)

To help forming and keeping constant situational awareness, a model called OODA Loop can be used. The OODA Loop was designed to help to quickly assess every situation. (McKay 2015)

OODA Loop model was developed by the US Air Force fighter pilot and military strategist John Boyd in the 1950s. The model is visualized in Figure 2. The steps of OODA Loop are Observe, Orient, Decide and Act. When used in decision-making, it is said that the person who can cycle through the steps fastest is the winner, whether the situation is head-to-head competition, like air-to-air combat in war, a violent confrontation in a parking lot between two men, or a political situation, or an information system incident. Constant situational awareness helps one to act on correct information. (Kelly M. 2014)

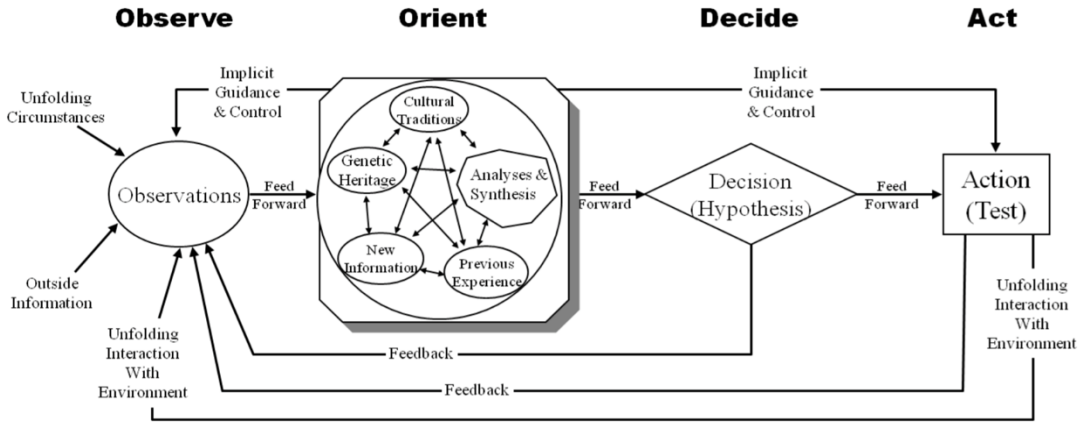

Figure 2. OODA Loop by John Boyd (Richards 2012, Figure 2)

John Boyd described Observation in one sentence as the act of sensing yourself and the world around you. Sensing, by definition, is the use of some sort of internal organ or technology - e.g. touch, taste, smell, sight, and hearing or a radar, infrared, communications intercept, etc., through which one receives stimuli from the external environment (Tremblay 2015, 7)

Figure 2 shows that observation step in OODA loop has multiple inputs. Boyd characterized these inputs as unfolding circumstances, outside information, one's unfolding interaction with the environment as well as two distinct types of internal feedback loops. (Tremblay 2015, 8)

Observe-step is about gathering data on what is happening in your environment. The biggest challenge in effective observation is knowing which elements of information are important to monitor and how to apply correct filters to them. For example, in retail sales good information to observe are customer renewal rates, the length of engagement, return rates and customer support rates. (Williams 2013)

In information system environment observe-step is about gathering data on what is happening. Data gathering can be implemented either by analyzing log files or via dedicated monitoring solutions, which for example record the actual network traffic to and from monitored target. The key challenge is to have the right amount of information and know which information is important.

Log, in the context of information systems, means a set of records written in timely manner about events happening in an information system. Log records usually report what has happened, when it happened and by whom it was done if there is a user or source context available. Logs can be classified by their meaning to, for example usage log, error log and change log. In the context of implementing situational awareness system, the focus will be on usage and error log. (Viestintävirasto 2016)

According to Boyd all observations would be meaningless without the context of orientation. Orientation both shapes observations and is the lens through which one makes sense of observations. (Tremblay 2015, 11)

Orientation-step is about calculating indicators and configuring alerts from the observed information and forwarding these indicators and alerts to decision makers. The observed information will be put to a context in this phase with the past experiences in the environment where the information comes from. With calculated indicators it is possible to predict what to expect next. (Williams 2013)

OODA Loops observe, and orientation steps are what situational awareness is about - gathering data and putting it to context.

2.2 Definition of situational awareness

Situational awareness is defined, by the mother of the theory Mica Endsley, as a "*perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future*". At the simplest the definition points to "*knowing what is going on*". Figure 3 illustrates Endsley's model about situational awareness. Situational awareness provides a basis for making decisions and evaluating the performance of actions. (Endsley 1995)

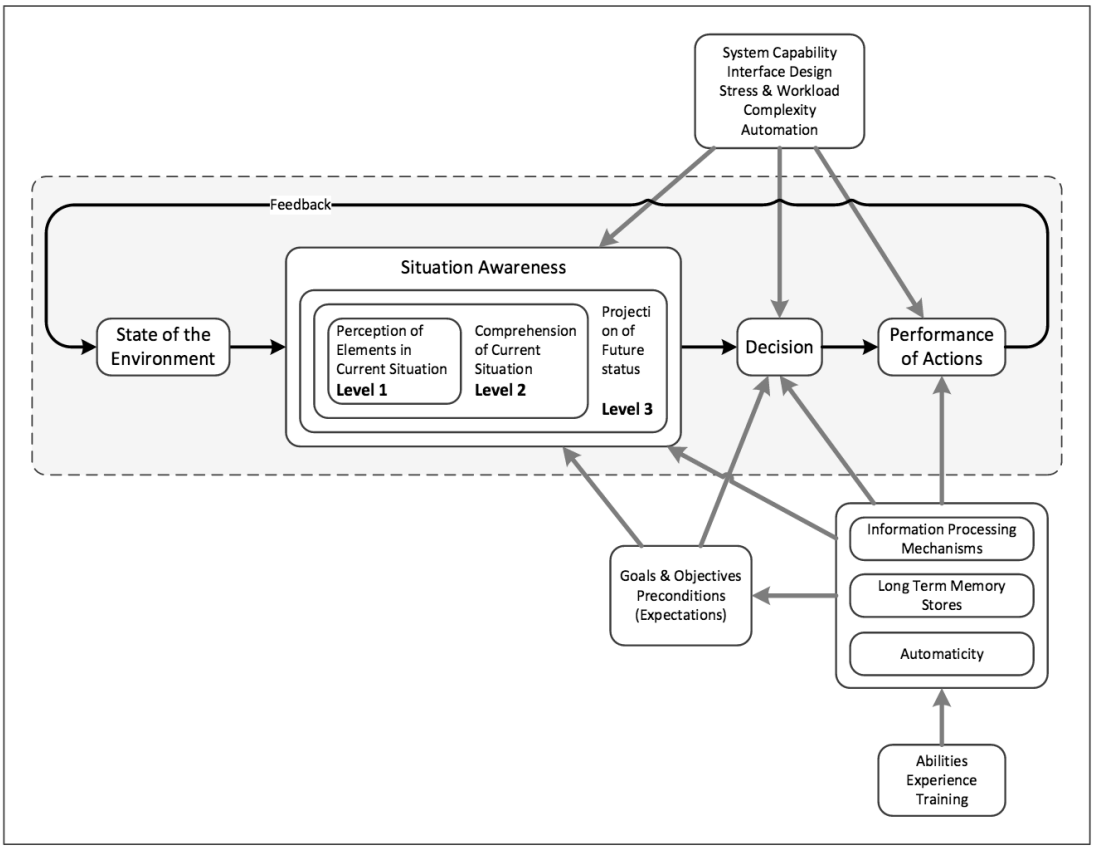

Figure 3. Endsley's model for situational awareness (Oosthuizen 2015, Figure 2)

2.3 Situational awareness examples in practical contexts

2.3.1 Aircraft and air traffic control

Aviation industry is the area which has perhaps the longest history with situational awareness. Situational awareness was recognized as a crucial asset in the World War I. Since then having SA has grown in importance also in both civil and commercial aviation. The safe operation of the aircraft in flight environment is highly dependent on a current assessment of the changing environment, including, for example weather conditions and other aircraft. (Endsley 1995)

Aircraft and air traffic control are closely connected. Air traffic controllers are responsible of managing an ever-increasing numbers of aircrafts landings and takeoff operations in tight schedules. The operators' job relies on accurate SA in a rapidly changing, three dimensional environment. (Endsley 1995)

2.3.2 Automation systems operators

Operators of automation systems, for example in manufacturing plants, refineries or nuclear power plants, must also rely on up-to-date information about situation parameters to manage automation systems effectively and safely. (Endsley 1995)

2.3.3 Tactical and strategic systems

Firefighters and police rely on SA to make correct decisions. They must have present knowledge of a current situation to determine to best course of action. Inaccurate or incomplete SA in these environments can lead to loss of life. (Endsley 1995)

2.3.4 Information systems

Operators of information systems are in a similar situation as are operators of automation systems. They must rely on up-to-date information about an environment's parameters to manage information systems effectively and safely. Outside of life-supporting devices actions based on inaccurate or incomplete SA with information systems will not lead directly to loss of life so in information systems context safety is usually interpreted as data safety.

Professionally managed information systems management activities are usually based on some IT service management framework, ITIL or ISO for example. Working according to the guidelines of IT service management frameworks requires the usage of some sort of OODA loop. ISO 20000 specifies a PDCA (Plan, Do, Check, Act) model, which is illustrated in Figure 4.

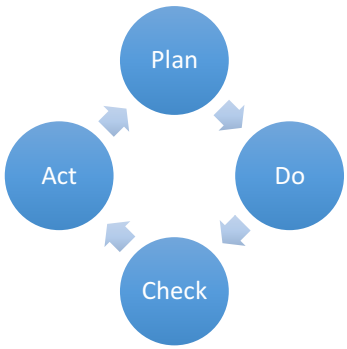

Figure 4. PDCA-model.

Both OODA and PDCA have some similarities; however, they also have slightly different approach to their use as a tool for management. OODA presents a more strategic and theoretical approach whereas PDCA has a more tactical and operative approach.

Having up-to-date situational awareness from information systems helps with PDCA model in multiple stages of IT service management. Following subchapters discuss the stages, how up-to-date SA is an asset in each stage. (Buchsein & Dettmer 2008)

2.3.4.1. Troubleshooting

In a troubleshooting context up to date situational awareness can help by providing a view to a system more as a whole than a single log file or error message. It is also possible to spot emerging troubles before they disrupt production.

Situational awareness can be presented on a dashboard which shows blocked traffic from firewall logs, different kinds of errors and warnings around the information system and a histogram of average response time of applications.

Situational awareness can help with both proactive and retroactive troubleshooting.

2.3.4.2. Change management

In the change management context situational awareness can help by giving feedback about change impact. For example, what kind of impact planned configuration change or bug fix has in system as a whole.

2.3.4.3. Capacity planning

In the capacity planning context situational awareness can help by giving information about server activity, used bandwidth and usage trends. One can use that information to predict the required capacity by extrapolating current trends.

2.3.4.4. Information security

In the information security context situational awareness can help by giving information about reconnaissance scanning, brute forcing, spidering, anomalies and many other indications of compromise or suspicious activity. Up to date situational awareness makes a huge improvement in an organization's ability to detect intrusion.

3 EVALUATING OPTIONS

3.1 Different methods for gathering data

Because situational awareness is about gathering data and putting it to context, the methods for doing that must be evaluated. In practice data can be gathered from networked information systems with three different methods. The methods are network, agent and log based. Following subchapters will discuss the methods, how each one works and each one's restrictions and benefits.

3.1.1 Network based approach

In the network based approach information system's network traffic is analyzed by either mirroring the traffic to a situational awareness instance or an instance can be installed in-line so that all traffic goes through the in-line-instance. An instance can be a device, an appliance or a server. Principals of gathering data via both methods are visualized in Figures 5 and 6.

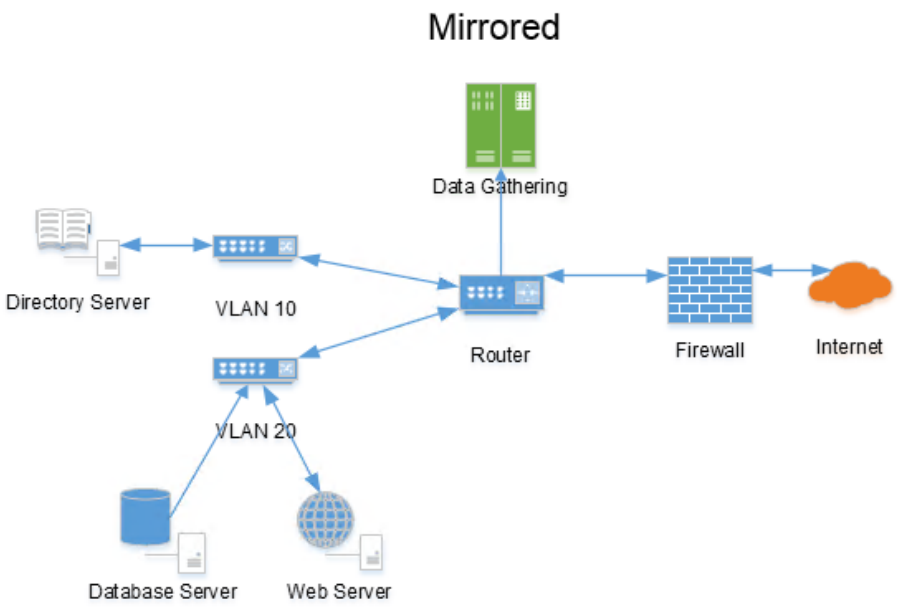

Figure 5. Data gathering by traffic mirroring.

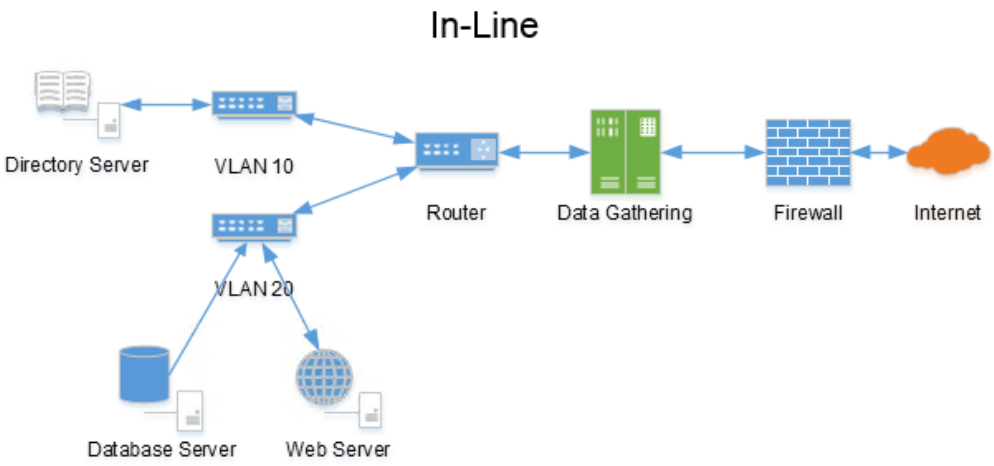

Figure 6. Data gathering by in-line solution.

For example, Riverbed SteelCentral NetShark is a solution which works with mirrored traffic. (SteelCentral NetShark Data Sheet 2015)

3.1.2 Agent based approach

Another way of gathering data from an information system is to gather data with agents. In this approach each device which is wanted for assisting forming the situational awareness has to have an agent installed which monitors the information system. The agent's gathered information is then either pushed to a situational awareness instance or pulled from agents by the instance. The principal of gathering data via agent is visualized in Figure 7.

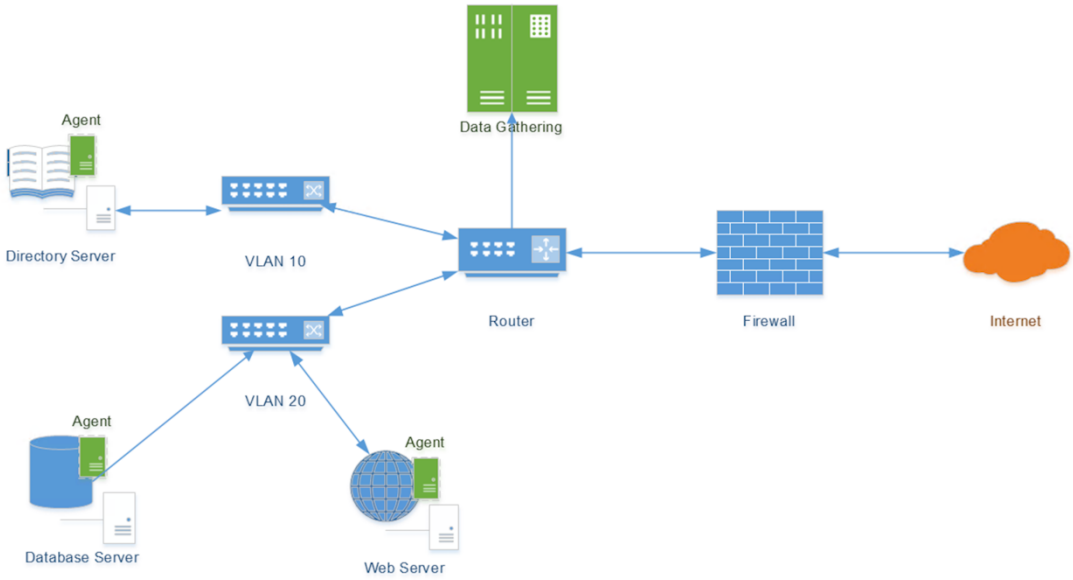

Figure 7. Data gathering by agents.

For example, Riverbed SteelCentral AppInternals is a solution which works in this way. (Hodgson 2015)

3.1.3 Log based approach

The third way of gathering data from an information system is to gather data from logs. In this approach each device and application which one wants to help with forming the situational awareness has to have logging enabled. Logs have to be monitored somehow and usually pushed or pulled to a centralized location for analysis and visualization. The principal of gathering data from logs is illustrated in Figure 8. Because of multiple log sources and different activity levels of log sources, it is more practical to push logs from source to central location than to pull all logs from one central location. When logs come from multiple sources the data must be normalized to help with the storing and visualizing.

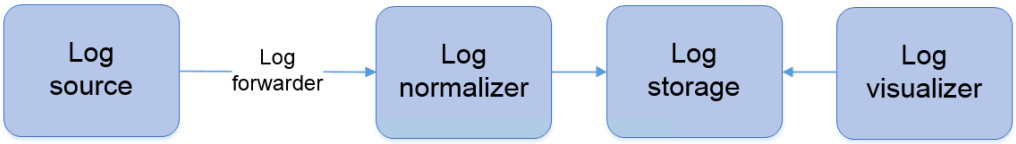

Figure 8. Log collecting and visualizing principal

For example, ELK stack is a solution which works in this way. (The Elastic Stack 2016)

3.2 Comparison of methods

Each method has its own characteristics and following subchapters discuss where the different methods differ, what they have in common and evaluate the advantages and disadvantages of each method.

3.2.1 Differences between methods

The differences between the three methods depend on the selected solution. The amount of data available to be gathered for situational awareness is different, network-based solutions have the least amount of data at their disposal whereas log-based solutions can have the greatest amount of data at their disposal and agent-based solution is situated in between the other two.

3.2.2 Similarities between methods

Each of the three methods shares in common that everyone is able to gather data from what is available for them. All solutions can have automatic correlation and alarms. The data gathered can be visualized no matter what approach is implemented. Depending on the selected solution, commercial products and support are available.

3.2.3 Advantages per methods

The advantages of the network-based approach are that it is easy to implement and there are not many moving parts. The ease of implementation is because the data that is gathered and analyzed, TCP and UDP packets, is strictly standardized. Traffic mirroring is also very easy to implement in virtualized environments because the network devices are also virtualized; thus, traffic mirroring does not need any new hardware. Configuration of virtualized network devices is quite straightforward because the configuration can be managed through one interface. With the network-based approach situational awareness system can be built quite fast because there are "bolt-on" solutions. After the network traffic is either routed through or mirrored to the instance, the network-based approach does not require any other installations or configurations to the target environment.

Network-based approach can gather data from every device, system or solution attached to a network, also proprietary in-house applications or solutions as long as the traffic is either routed or mirrored to the network-based solution.

The advantages of agent-based approach are that it has better visibility than the network based approach. Visibility is better because the agent can gather data inside a server about the utilization of server's resources or error messages. The implementation with agent-based approach is quite straightforward because the information sent from the agent to the situational awareness system is managed by the vendor.

The advantages of the log-based approach are that it can have the best visibility because every operating system, application, device and appliance supports logging to some extent. There are also open source solutions so log-based approach does not have mandatory expenses, it can be built on the existing hardware. Log-based approach can be used to gather data from in-house solutions or application as long as logging is implemented in them.

3.2.4 Disadvantages of each method

The disadvantages of the network-based approach are that it has a limited ability to gather data and there are only commercial solutions available so this method has mandatory expenses. The limited view is because the network-based approach can gather data only from the traffic that it sees. Visibility can be limited because of the network hierarchy or because the interesting information never travels on the network. Interesting information that never travels on a network can be, for example a blocked connection attempt in software firewall, failed local login attempt or an error message in system logs.

An example of limited visibility because of network hierarchy and location of data gatherer is illustrated in Figure 9. When information has been gathered with in-line solution, the green device after perimeter router, internal traffic, orange line between servers is outside of its reach.

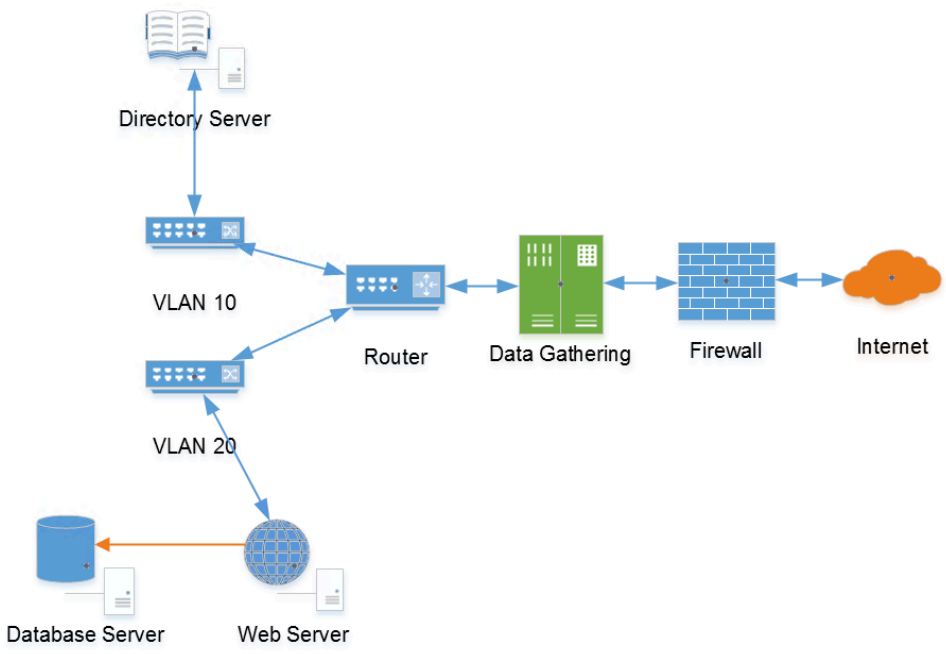

Figure 9. Limited visibility of network-based appliance.

The disadvantages of the agent-based approach are that there is usually no support for proprietary in-house solutions, every device or server must have

an agent installed. Agent installation might not be possible if for example there is no support for operating system in use.

The disadvantages of the log based approach depend on the selected solution, however, universally the only downside is that the log-based approach requires the greatest amount of work to get up and running. If an open source solution is selected there might not be commercial support available. The log-based solution is usually the most complex and has more moving parts than the other approaches.

3.3 Selecting the solution

After different approaches were studied in theory there was a demonstration scheduled with the assigner company's IaaS-vendor. The IaaS-vendor collaborates with Bemecon Solutions which provides solutions from Riverbed Technology.

There was a demonstration about Riverbeds products, they have both network and agent based solutions and also pure web application analytics solution. The demonstrated solutions were SteelCentral AppInternals, Web Analyzer and NetShark. All of these commercial solutions would have been relatively easy to implement; however, there were some clear disadvantages why it was chosen not to proceed with any of the demonstrated solutions.

SteelCentral Web Analyzer is aimed at analyzing only web applications. Its disadvantages were that it would not give any overview to the servers; it is based on injecting JavaScript-code to users' browsers, and the pricing was quite steep. Using shared instance, the price was near manageable; however, regarding the nature of the provided services, patient records, shared instance and injecting JavaScript to clients did not sound acceptable. On top of that, only situational awareness would have been about the web application.

SteelCentral AppInternals is an agent-based solution. It is able to dig in to the applications or services running in a server, for example visualizing web

application's transactions to SQL-query or .NET method level. The major disadvantage of AppInternals was the too high price.

SteelCentral NetShark is a network-based solution. It is able to form situational awareness from the network traffic which is available to it. The biggest disadvantages of NetShark were the high price, limited visibility of situational awareness and the need to decrypt the traffic for analysis.

For evaluating the log-based solution, a proof of concept environment was built using open source products called ELK stack. With the proof of concept environment all of the advantages and disadvantages discussed in previous chapters were noted accurate.

Because all of the demonstrated commercial solutions were too expensive and had clear disadvantages it was decided to proceed with the log-based approach with open source products. The selected log-based solution is called ELK stack. ELK stack is formed by combining Elasticsearch, Logstash and Kibana. ELK stack and the other needed components are discussed in the next chapter. The only disadvantages of the selected solution, based on the proof of concept environment, were that there is slightly more work than in commercial solutions, there is no automatic correlation, and the alerts require expensive subscription from Elastic, the company behind Elasticsearch. The biggest advantage of the selected solution was the price - it was free because of open source components, and there was a possibility to form most comprehensive situational awareness.

4 SYSTEM INFRASTRUCTURE

4.1 Situational awareness system, main components

The following chapters discuss the main components of situational awareness system, ELK stack, infrastructure. ELK stack infrastructure is illustrated in Figure 10. The infrastructure figure contains also optional utilities used in this implementation, NGINX and NXLog, which are discussed in detail later in chapter 4.2.

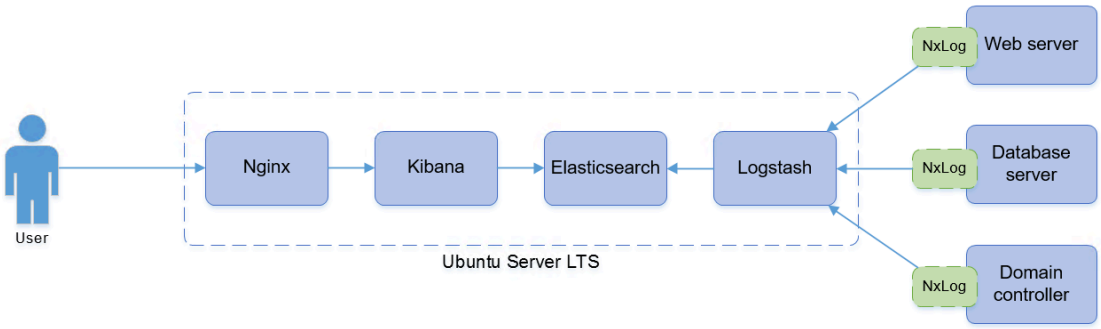

Figure 10. ELK Stack infrastructure.

4.1.1 Ubuntu

Ubuntu is a Debian-based Linux operating system by Canonical. In this thesis Ubuntu Server LTS is used as a base operating system for situational awareness system.

4.1.2 Logstash

Logstash is a product of Elastic and it is an open source data collection engine with real-time pipelining capabilities. Logstash is capable of dynamically unifying and normalizing diverse logs to multiple destinations. It can also enrich or transform logs with a broad array of different kind of plugins.

Elastic (Logstash Introduction 2016) states that "*The better the data, the better the knowledge. Clean and transform your data during ingestion to gain near real-time insights immediately at index or output time.*"

In this thesis Logstash is used for both normalizing and enriching data. For example, log event time is normalized, log sources are typed, unnecessary fields are dropped and IP addresses can be enriched with reverse DNS queries.

4.1.3 Elasticsearch

Elasticsearch is also a product of Elastic and it is a highly scalable open source full-text search and analytics engine. Elasticsearch allows one to store, search, and analyze big volumes of data almost in real time.

In this thesis Elasticsearch is used to store and search normalized and enriched data coming from Logstash.

4.1.4 Kibana

Kibana is also a product of Elastic and it is an open source analytics and visualization platform designed to work with Elasticsearch. Kibana can be used as user interface to search, view and interact with data stored in Elasticsearch indices. One can easily perform advanced data analysis and visualize the data with Kibana. Elastic (Kibana Introduction 2016) states that "*Kibana makes it easy to understand large volumes of data. It's simple, browser-based interface enables you to quickly create and share dynamic dashboards that display changes to Elasticsearch queries in real time*."

In this thesis Kibana is used as a browser-based user interface to create a dynamic dashboard from the data.

4.2 Situational awareness system, utilities

The following chapters discuss optional utilities used with situational awareness system.

4.2.1 NGINX

NGINX is a multipurpose proxy server originally written by Igor Sysoev. In this thesis NGINX is used as a HTTP proxy to forward traffic from HTTP default port, TCP 80 to Kibana's port TCP 5601. (About NGINX 2016; Getting Kibana Up and Running 2016)

4.2.2 Elasticsearch Curator

Elasticsearch Curator is a product of Elastic but it was originally published by Jordan Sissel with the name clearESindices.py. After Sissel was hired by Elasticsearch the tool got its current name. Curator is a tool which is used to manage Elasticsearch indices. (Curator readme 2016)

In this thesis Elasticsearch Curator is used to clear logs from Elasticsearch after retention time.

4.2.3 NXLog Community Edition

NXLog Community Edition is a product of NXLog, it is an open source, high-performance, multi-platform log management solution which is aimed at solving typical log processing tasks such as forwarding, filtering and classification for example. NXLog can collect logs from various file formats, receive logs from network. It supports common platform specific sources for example Windows Event log, Linux kernel logs and syslog. (NXLog Community Edition 2016)

In this thesis NXLog is used as a log forwarder in Windows Servers. NXLog is configured to forward both file-based logs and event logs to Logstash in structured format where possible.

4.3 Production environment

Simplified production environment infrastructure, regarding the scope of this thesis, is illustrated in Figure 11. The whole environment consists of dozens of servers, however, the main components and the structure are the same.

The operating systems are either Windows Server 2008R2 or 2012R2, the web server is Microsoft IIS and the software firewall is Windows Firewall. Logs are forwarded with NXLog CE. The database server is illustrated in Figure 11, however, nothing database specific is used as a log source within the scope of this thesis.

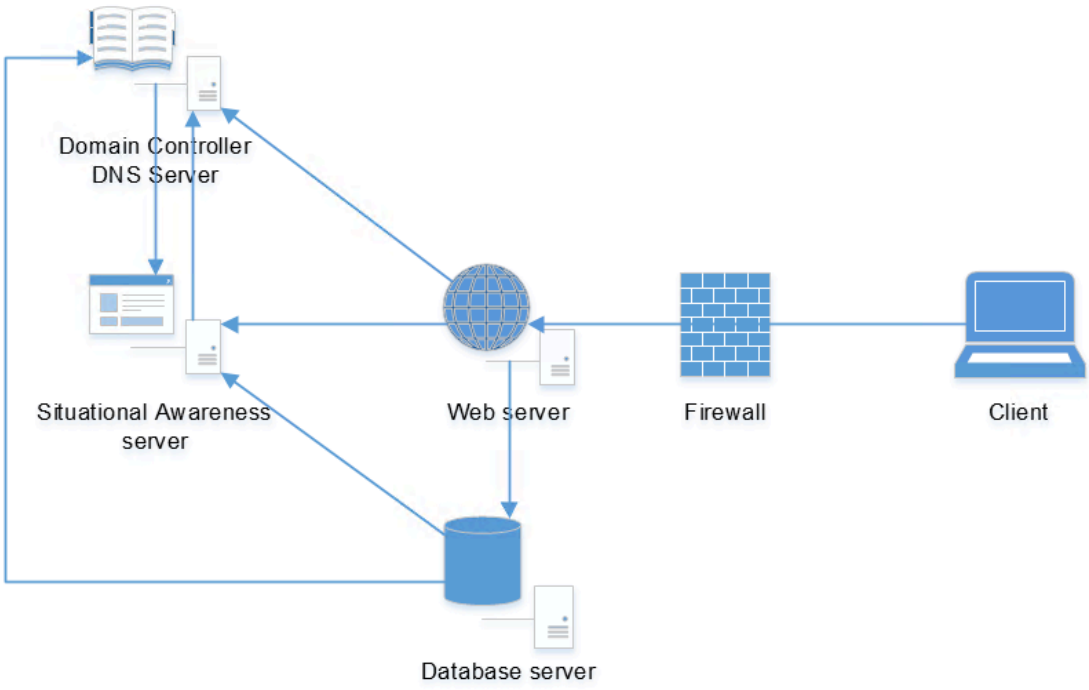

Figure 11. Production environment infrastructure

5 IMPLEMENTATION

5.1 Production environment

The situational awareness is formed with log data collected from multiple Microsoft applications used in the production environment. The only implementation required in the current production environment is installing NXLog CE and configuring both NXLog and log sources. The following chapters discuss the data source configurations per log source.

5.1.1 Microsoft IIS

The web server used in the production environment is either IIS 7.5 or IIS 8.5. There is no difference in logging features between those versions. IIS logs to a text file using W3C Extended log format which contains delimiter-separated values. The logged fields are listed in Table 1.

Table 1. IIS log fields

Field	Description
date	the date on which the request occurred
time	the time, in UTC, at which the request occurred
c-ip	the IP address of the client that made the request
cs-username	the name of the authenticated user who accessed your server
s-computername	the name of the server on which the log file entry was generated
s-port	the server port number that is configured for the service
cs-method	the requested action, for example, a GET method
cs-uri-stem	the Universal Resource Identifier, or target, of the action
cs-uri-query	the query, if any, that the client was trying to perform
sc-status	the HTTP status code
sc-substatus	the HTTP substatus code
sc-win32-status	the Windows status code
sc-bytes	the number of bytes that the server sent
cs-bytes	the number of bytes that the server received
time-taken	the length of time that the action took in milliseconds
cs(User-Agent)	the browser type that the client used
cs(Referrer)	the site that the user last visited. This site provided a link to the current site

5.1.2 Windows Firewall

The servers are configured to filter and log the filtered traffic with a Windows Firewall. The traffic is filtered both inbound and outbound. The Windows Firewall logs to a text file using delimiter-separated values. The logged fields

are listed in Table 2. The Windows servers are configured to use only IPv4. Because the firewall does not enable the configuring of logged fields, unnecessary fields will be dropped during normalization. Unnecessary fields are marked red in Table 2.

Table 2. Windows Firewall log fields

Field	Description
date	Date formatted in YYYY-MM-DD
time	24 hour time formatted in HH:mm:ss using OS time.
action	Action for data packet, DROP or ALLOW
protocol	Protocol for data packet, TCP, UDP or ICMP
src-ip	Source IP-address
dst-ip	Destination IP-address
src-port	Source port
dst-port	Destination port
size	Data packet size in bytes
tcpflags	Information about TCP control flags in TCP headers
tcpsyn	TCP sequence number in the packet
tcpack	TCP acknowledgement number in the packet
tcpwin	TCP window size, in bytes, in the packet
icmptype	Information about the ICMP message
icmpcode	Information about the ICMP message
info	An entry that depends on the type of action that occurred
path	Direction of the data packet, RECEIVE or SEND

5.1.3 Microsoft DNS

All components of the internal infrastructure are configured to use DNS from the organization's domain controllers. Basic DNS server installation and configuration are not within the scope of this thesis. Before Windows Server 2012R2 the only way to log DNS queries was through debug logging. From 2012R2 on there is also a feature called Analytic log for DNS. If DNS performance is critical and the DNS server is under heavy load, then Analytic logs should be a better choice because Analytic logs has negligible performance impact according to Microsoft.

In the organization's production environment, the DNS is only in internal use and the server environment is relatively small, only few hundred servers, either one of the logs could be used. Because debug log is flat file based, it is easier to forward to Elasticsearch, so it will be the choice. The default location of the debug log is C:\Windows\System32\dns\dns.log. The logged fields are listed in Table 3. The field description is left empty if documentation was not

available and description was not possible to derive by observing log events. Because the DNS server does not enable the configuring of logged fields, unnecessary fields will be dropped during normalization. Unnecessary fields are marked red in Table 3.

Table 3. Microsoft DNS log fields

Field	**Description**
Date	Date using current regional format
Time	Time using current regional format and time zone
Thread ID	-
Context	-
Internal packet identifier	-
UDP/TCP indicator	Indicates whether UDP or TCP was used
Send/Receive indicator	Indicates whether logged packet was sent or received
Remote IP	Source address of the dns query
Xid (hex)	-
Query/Response	Indicates whether packet is query or response
Opcode	DNS query type e.g. Standard query or update
Flags (hex)	Response type as hex e.g. authoritative answer
Flags (char codes)	Response type flag e.g. authoritative answer
Response code	Response code e.g. successful query
Question type	Type of DNS record queried e.g. A = host record
Question name	Actual DNS record queried e.g. (5)time1(5)mikes(2)fi(0)

Debug logging is enabled through DNS Manager. In the scope of this thesis DNS logs are used to form situational awareness, both incoming and outgoing DNS query requests and responses are of interest for the thesis. The logging configuration is illustrated in Figure 12.

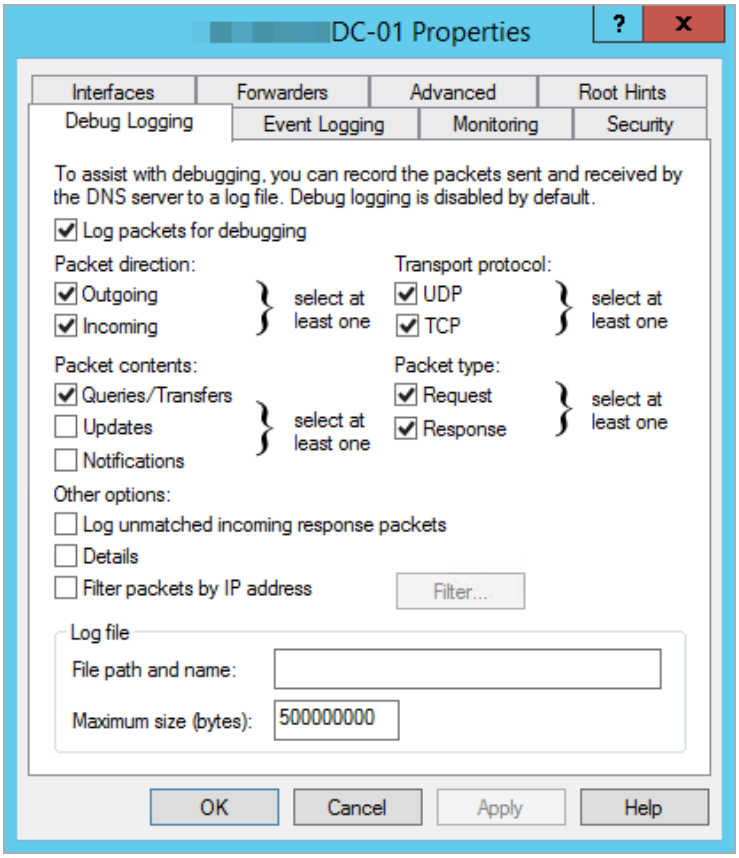

Figure 12. DNS logging configuration.

5.1.4 Windows Event Log

Windows logs most of its internal events to Event log. The verbosity and events that will be logged can be configured with either group policy objects or local security policies. Windows Event log consists of four categories: system, security, application and setup. In this thesis server event logging is configured using group policy objects and configurations are based on Center for internet security configuration benchmark for Windows 2008R2.

Security log contains audit events and they are classified as successful or failed depending on the event, such as whether a user trying to log on to Windows was successful. Security-log is for system use only.

Application log contains events logged by applications or programs. Events are classified as error, warning and information, depending on the severity of the event. Application-log is open to user and third party applications.

System log contains events logged by Windows or Windows system services. The events are classified as error, warning and information, depending on the severity of the event. System-log is for system use only.

Setup log contains events related to Windows components, Windows updates and application setup. In the production environment log events are mainly about security updates.

In the scope of this thesis the focus is on the security and system logs.

5.2 Installation of ELK stack

The actual installation of ELK stack is quite well documented online. There are package repositories available to all necessary components so there is no need to do much manual work installing the components. The package repositories help with keeping the components up to date. The installation was based on Digital Ocean's article "*How To Install Elasticsearch, Logstash, and Kibana (ELK Stack) on Ubuntu 14.04*". The only deviations from the article were that Kibana was version 4.5 and Elasticsearch was version 2.3 so the repository sources were modified accordingly. The operating system used was Ubuntu Server 14.04 LTS.

The following chapters discuss the getting of data from different sources normalized and forwarded to Kibana.

5.3 Data normalization

Before the heterogeneous data from multiple sources can be indexed and analyzed, there is a need for data normalization which is done by manipulating data. Normalization by definition means to remove redundancy and to make log event regular and consistent.

The manipulation can be made with either NXLog before forwarding or with Logstash just before indexing. Both components are used to get the data

normalized. In the normalization phase correct data types have to be assigned for each field because by default Elasticsearch will use string.

Log data from different sources will be in multiple different formats so each format has to be recognized and manipulated accordingly. In this thesis log source differentiation is made by tagging each log event based on which TCP port is used to record the log event. Logstash input configuration is presented in Appendix 1. After the log event is normalized, it will be forwarded to Elasticsearch. Logstash output configuration is presented in Appendix 6.

5.3.1 IIS

IIS log data structure is space delimited and log event data will be structured with NXLog before forwarding. With IIS only interesting fields are logged so there is no need to discard any data. The only manipulation made with Logstash is to match the event timestamp and extract user agent information from cs(User-Agent) field.

Example event from log file, address details removed

```
2016-05-07 20:38:51 <hostname> POST /<application>/<filename>.aspx - -
88.114.x.x
Mozilla/5.0+(X11;+Linux+x86_64;+rv:45.0)+Gecko/20100101+Firefox/45.0
https://<hostname>.<domain>.fi/<application>/<filename>.aspx
<hostname>.<domain>.fi 200 0 0 1130 825 327
```

Both NXLog and Logstash configurations regarding IIS are presented in Appendix 2. Example event in Kibana after normalization is illustrated in Figure 13.

Figure 13. Normalized event.

5.3.2 Windows Firewall

The structure of Windows Firewall log is space delimited and log event data will be forwarded as is. The structuration is made with Logstash. Windows does not have any configuration about the logged fields in the firewall so Logstash is used to discard unnecessary fields from log and match event timestamp.

Example event from log, source address details removed.

```
2016-04-17 18:48:15 DROP ICMP 10.x.x.x 216.58.209.131 - - 0 - - - - 8 0 -
SEND
```

Both NXLog and Logstash configurations regarding firewall are presented in Appendix 3. Example event in Kibana after normalization is illustrated in Figure 14.

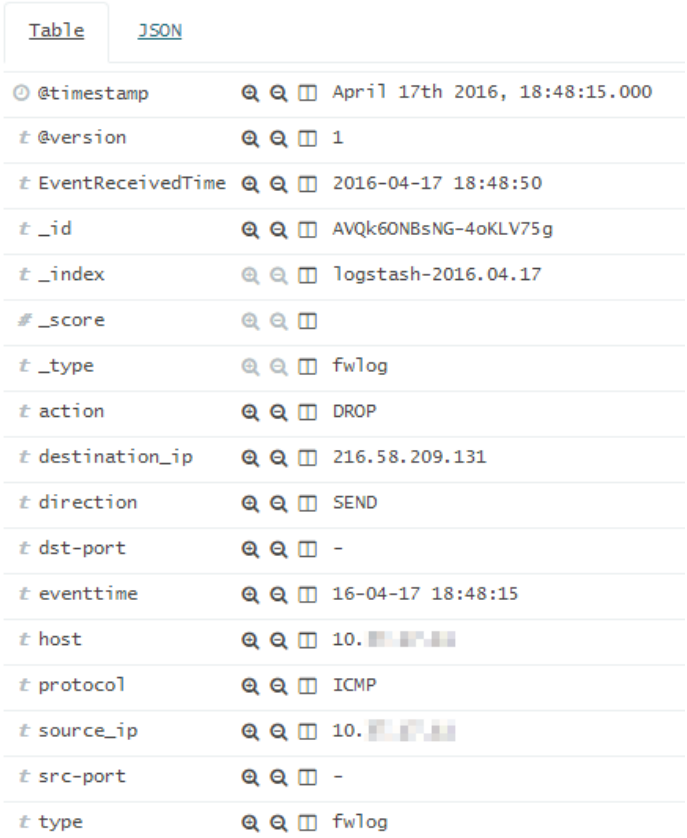

Figure 14. Normalized event.

5.3.3 DNS

The structure of DNS debug log is of fixed width and will be forwarded as is and structured and manipulated with Logstash before indexing. Windows does not have any configuration about the logged fields in DNS debug log so Logstash is used also to discard unnecessary fields and match event timestamp.

Example line from log, source address details removed

```
15.5.2016 18:28:36 07F4 PACKET  0000000001D97A90 UDP Rcv 10.40.x.x    b9b5
Q [0001   D   NOERROR] A      (4)ldap(6)fineid(2)fi(0)
```

Normalization is made in four steps, the first data will be split into fields, the second unnecessary data will be discarded using previously defined fields, the

third domain name query is converted to its normal format, separated with dots, and lastly, the source DNS server hostname is added based on IP address.

Both NXLog and Logstash configurations regarding DNS are presented in Appendix 4. Example event in Kibana after normalization is illustrated in Figure 15.

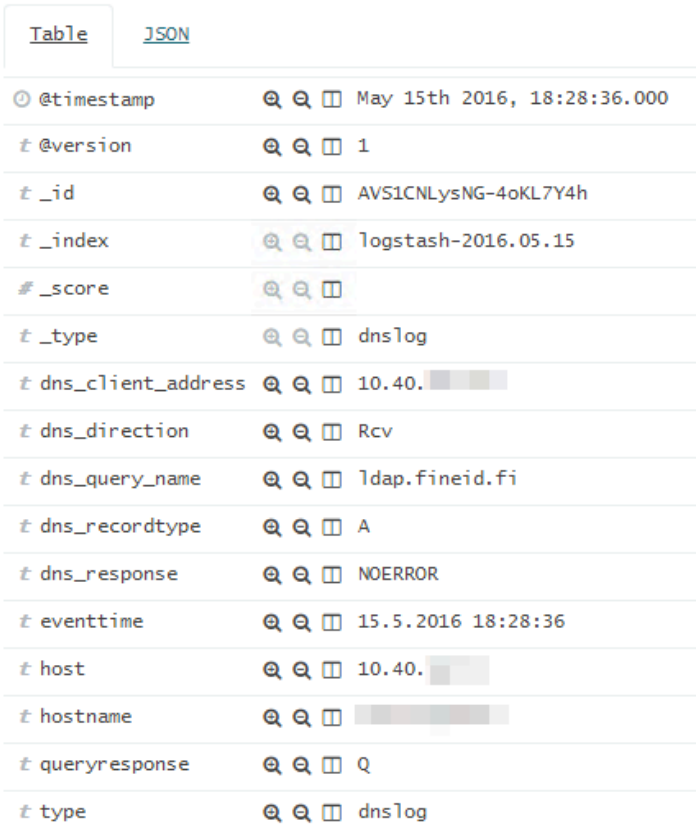

Figure 15. Normalized DNS log event.

5.3.4 Event log

Event log is a centralized location for structured logs in Windows. Event log infrastructure was completely revamped in Windows Vista. Since Vista event logs are written in XML format. (Microsoft, 2014)

Event logs can be forwarded as they are, because XML is already a structured format. Event log contains quite an amount of unnecessary information so normalization is required with Logstash before indexing.

Example event from log in XML format is presented in Appendix 7.

Both NXLog and Logstash configurations regarding event log are presented in Appendix 5. Example event in Kibana after normalization is illustrated in Figure 16.

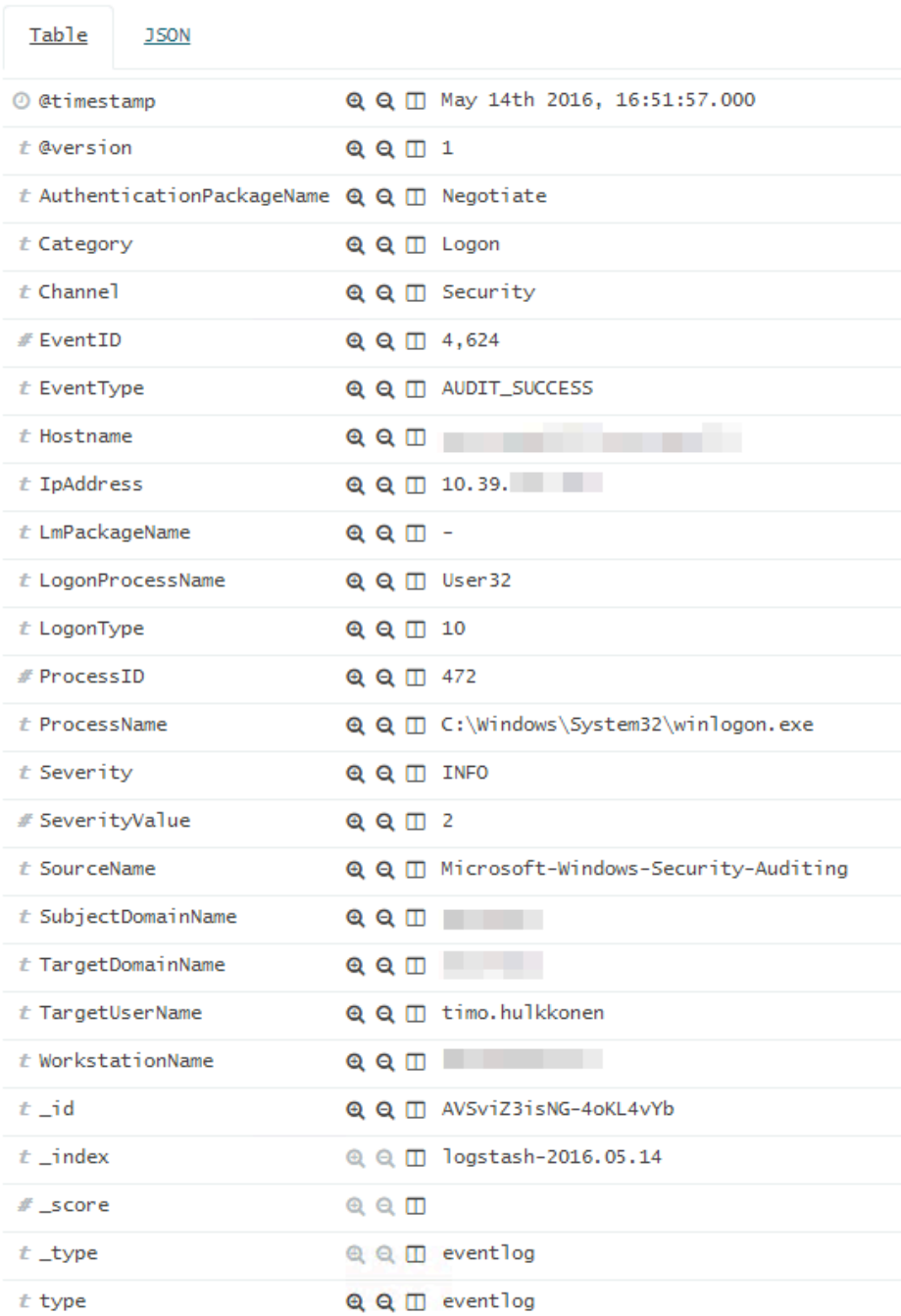

Figure 16. Normalized login event.

5.4 Dashboards

By default, Kibana does not have any dashboards. One has to think what kind of information is possible to be extracted from the log data. For example, IIS logs contains a field called sc-bytes, which is the amount of bytes sent from the server to client. Using this field, it is possible to visualize the used bandwidth when summing sc-bytes from log events and distributing it to the timeline. Another interesting field is time-taken, which is the amount of milliseconds each HTTP request took from the server to complete. Calculating and visualizing an average response time gives quite a good insight about the application's or web service's performance in the scope of server-side actions.

5.4.1 IIS

IIS, which is Microsoft's web server, is a platform for the web application and it makes actions regarding to clients' HTTP requests. Normalized log event is described in chapter 5.3.1. Using the information from normalized events, it is possible to build dashboards which will visualize, for example:

- server activity by counting sum of http requests in a time period
- client user experience by counting average response time in time period
- client browser shares by counting shares from user agent information
- used bandwidth by counting sum of sc-bytes in a time period
- error activity by counting and grouping sum of http status codes over 399
- effectivity of client-side caching by counting shares of http status codes with static files

A dashboard from an actual production environment from a dedicated client environment is illustrated in Figure 17. The dashboard shows four different charts from logs; top left corner is bandwidth, top right is average response time, bottom left is http request status codes and bottom right is http error code count. Each value is within five-minute time period. The charts in dashboard show normal activity regarding the visible timeline which is Monday from 6:00 to 20:00 in the example.

Figure 17. Actual dashboard from production environment.

5.4.2 Firewall

Firewall is a critical component of modern information systems which helps to restrict both incoming and outgoing traffic. Normalized log event is described in chapter 5.3.2. Using information from normalized events, it is possible to build dashboards which will visualize e.g.

- servers' denied traffic activity by counting sum blocked traffic in a time period
- top blocked destination ports
- top blocked destination IP addresses

An actual production environments dashboard is illustrated in Figure 18. Logs are collected at the moment from one internal server. The dashboard shows three different charts and tables from logs. Charts and tables in dashboard show normal activity regarding the visible timeline which is Monday from midnight to Sunday 23:00 in the example. There are two visible spikes on Wednesday and Saturday exactly at 6:40. These anomalies are discussed in chapter 6.2.

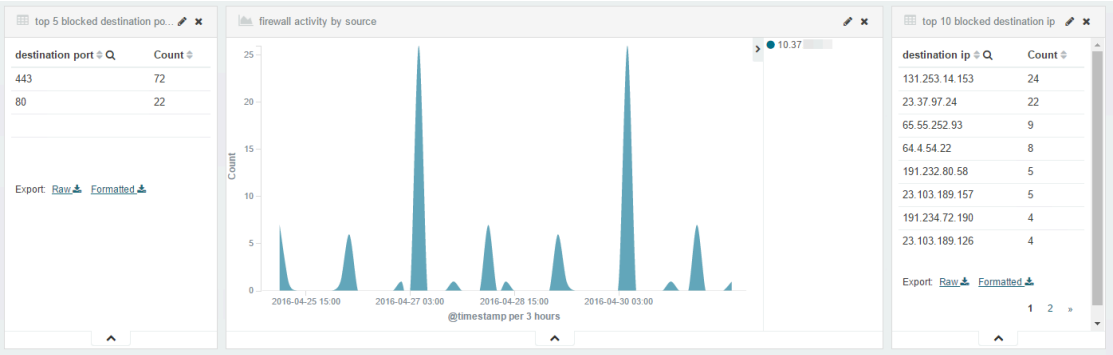

Figure 18. Actual firewall dashboard from production environment

5.4.3 DNS

DNS is a base of modern networks which helps to map server names and addresses to actual IP addresses. Normalized log event is described in chapter 5.3.3. Using information from normalized events, it is possible to build dashboards which will visualize

- total DNS request count in a time period
- external DNS request count in a time period
- top DNS client requests by counting shares from source IP address
- top target domain names for DNS queries
- target domain names where DNS record was missing

An actual production environment's dashboard is illustrated in Figure 19. Logs are collected from all internal DNS servers. The dashboard shows five different charts and tables from logs. Charts and tables in the dashboard show normal activity regarding the visible timeline which is from Saturday midnight to Tuesday 23:00 in the example.

Figure 19. Actual DNS dashboard from production environment.

5.4.4 Event log

Event log is a centralized location for Windows logs. Normalized log event is described in chapter 5.3.4. Using information from normalized events, it is possible to build dashboards which will visualize

- successful login counts per logon type and server in a time period
- failed login counts per logon type and server in a time period
- error event count per server in a time period
- warning event count per server in a time period

An actual production environment's dashboard is illustrated in Figure 20. Logs are collected at the moment from few development environment servers. The dashboard shows four different charts which show normal activity in every other chart except system errors chart regarding the visible timeline which is from Saturday midnight to Tuesday 23:00 in the example. The system error chart shows a high amount of errors on Monday around 02 at a.m. which is discussed in chapter 6.4

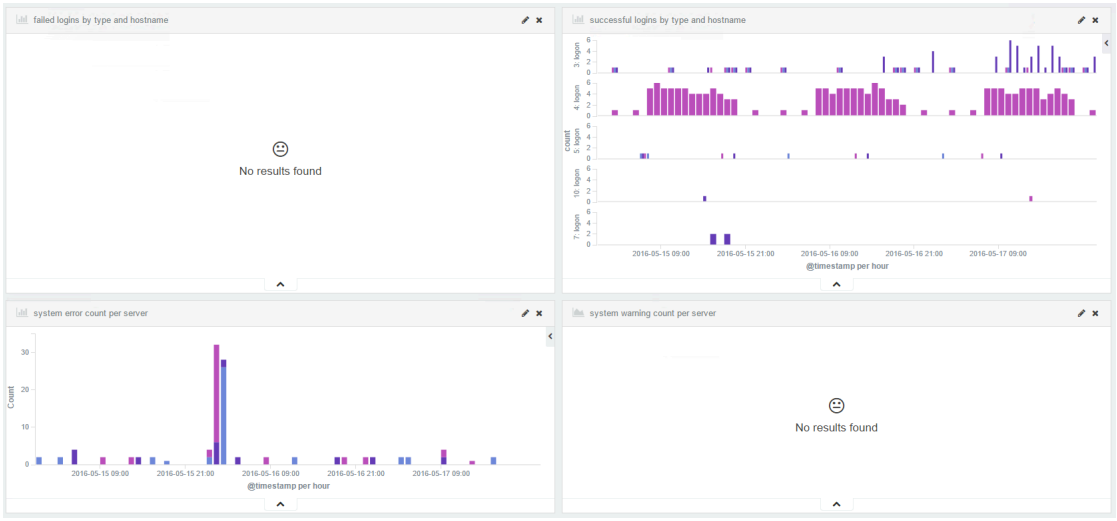

Figure 20. Actual event log dashboard from production environment.

5.5 Data retention

By default, ELK stack stores all data indefinitely and it is not usually possible because in production environments the data amounts are generally large and the required storage costs would rise indefinitely. Therefore log data usually has a limited retention time. Depending on the log data, there might be

regulations that direct the required retention time. In the scope of this thesis there are no regulations so retention time can be defined using common sense and planning what is required from the SA system in question.

The busiest single web server produces up to 400 megabytes of log per weekday. All DNS servers produce around 50 megabytes of log per day. At the moment, the log retention time is one month, so the required storage is few dozen gigabytes, when there are only few web servers forwarding logs.

Data retention is managed using Elasticsearch Curator which was discussed in chapter 4.2.2. Curator deletes indices using the provided parameters. In the scope of this thesis the indices are deleted after retention time which is 31 days. Curator is executed using following command:

```
curator delete indices --older-than 31 --time-unit days --
timestring %Y.%m.%d
```

The execution of Curator is automated using Cron. Scheduling is configured by opening Cron by executing `sudo crontab -u logstash -e` and adding following line to Cron which executes Curator daily at 04:30:

```
30 04 * * * /usr/local/bin/curator --logfile
/var/log/logstash/curator.log delete indices --older-than 31 --
time-unit days --timestring \%Y.\%m.\%d
```

6 TESTING SITUATIONAL AWARENESS SYSTEM

The following chapters discuss how the situational awareness system was able to visualize different kinds of activity, what the normal user activity looked like, was there useful information available and is it possible to detect different kinds of information security related activities. The results are discussed per log source.

6.1 Web server (Microsoft IIS)

To see and verify the value of situational awareness, the following tests were made regarding web server logs - vulnerability assessment and spidering with Burp Suite, reconnaissance scan with DirBuster. The tests were made in the test environment without other users using software. Another method for verification is to look for characteristics in the visualization of the production environment and to compare the visualization of the normal situation to the visualization from a situation where there is an active issue with, for example, performance.

Figure 21 is from the organization's busiest production environment and illustrates the actual server-side performance issue beginning at 11:10 and resolving at 12:05. During the performance issue the average response time shows a clear spike and about three times greater average than in a normal situation. With the current situational awareness data this issue was noticed by the operations team before the customer called the helpdesk. Situational awareness helped to see the impact of resolving the performance issue, and the average response time returned to normal.

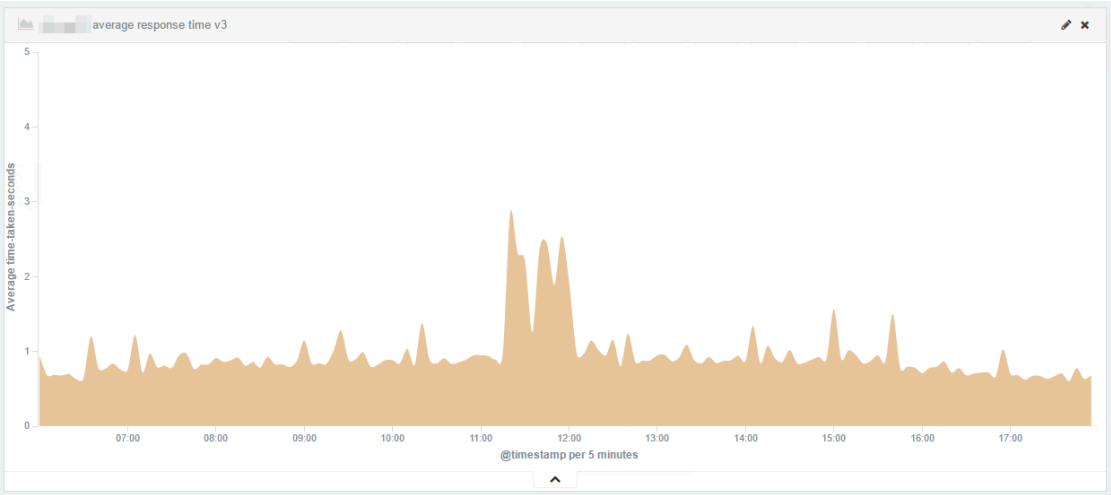

Figure 21. Actual dashboard from production with performance problem.

With visualized logs it is easy to see usage trends in the environment. Figure 22 illustrates one week from Monday to Sunday, and it can be seen that the customer's normal pattern for usage is from Monday to Friday between 7:30 and 15:30, Monday being the busiest. Outside the customer's usage the bottom noise comes from automatic monitoring which is a set of scripts which tests the basic features and records the response times. The results from that monitoring are used for calculating SLA levels regarding availability and performance.

With the chart in Figure 22 it is easy to calculate the used average bandwidth for this application - 150MB / 5 minutes / 60 seconds = 500kB/s. In the example production environment there are hundreds of active users.

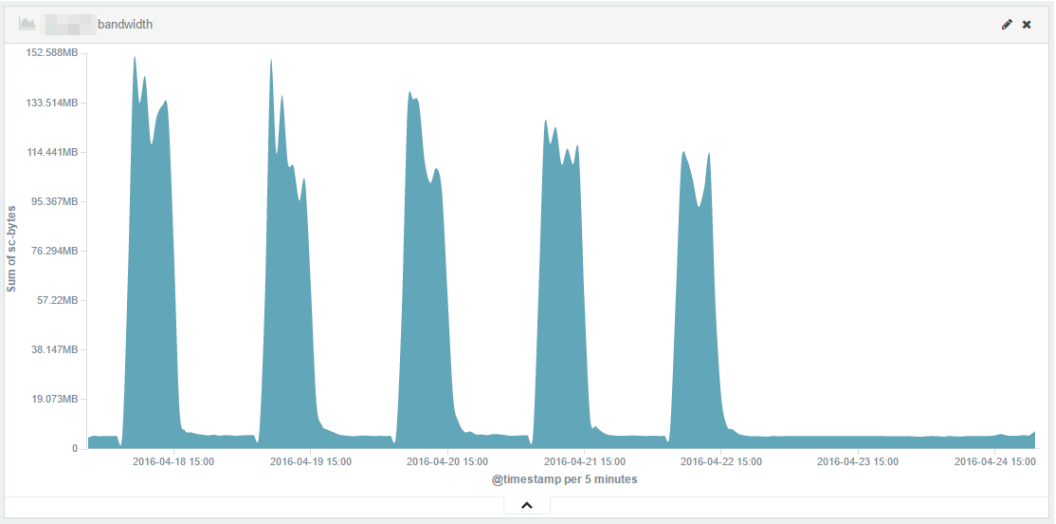

Figure 22. Actual chart from production showing activity trends.

The following figures illustrate how different information security related activity can be detected in situational awareness charts. Vulnerability assessment with automated tools such as Burp Suite is quite noisy, and it is relatively easy to identify. Detection methods depend on what HTTP method the vulnerability assessor uses. Vulnerability assessment using HTTP GET can be detected observing query parameters. If HTTP POST method is used, the assessment has to be detected by other means because HTTP POST body is not recorded to log.

The testing was made in a test environment without other users and the application was used before starting the scan. The active scan was started at 20:01 as illustrated in Figure 23. At first the HTTP request count alone is not suspicious, because the active scan generates only few requests per second by default whereas the production environment can have thousands of requests per minute.

The usage of automated tool such as Burp's Active Scan can be identified usually by observing HTTP error rate. Automated tools usually cause HTTP errors which can easily be detected at 20:06 in Figure 23. When using Burp Suites active scan, the error rate can go up to 100 per minute where in production the average error rate is one per minute.

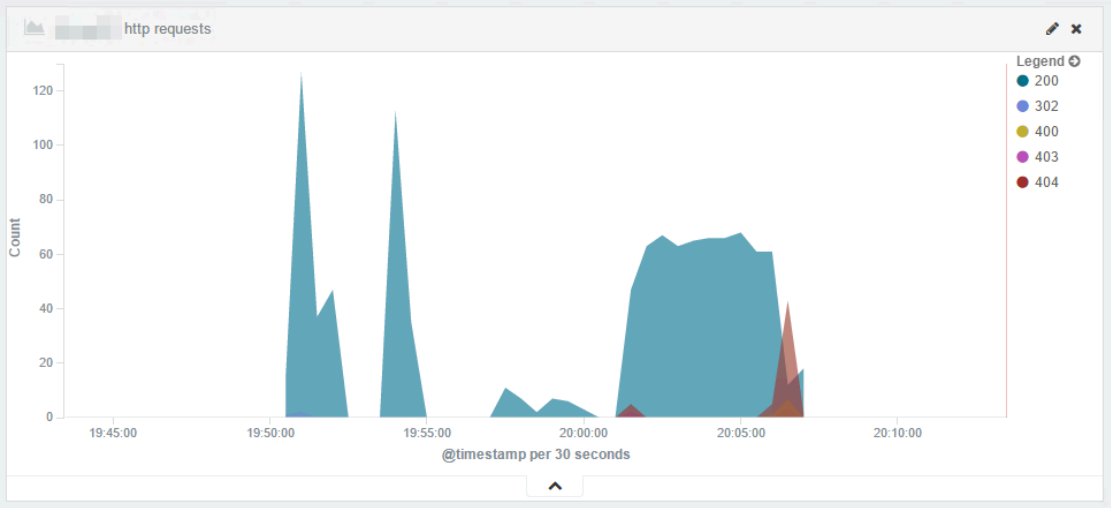

Figure 23. Burp Suite Active scan in IIS logs.

Spidering, which means crawling web application by analyzing page content and submitting forms with automated tool such as Burp Suite is also relatively noisy. Spidering was started at 20:39 as illustrated in Figure 24, and it can be identified by observing HTTP error rate starting immediately at 20:39. The error rate is around 20 per minute, where in production the average error rate is one per minute.

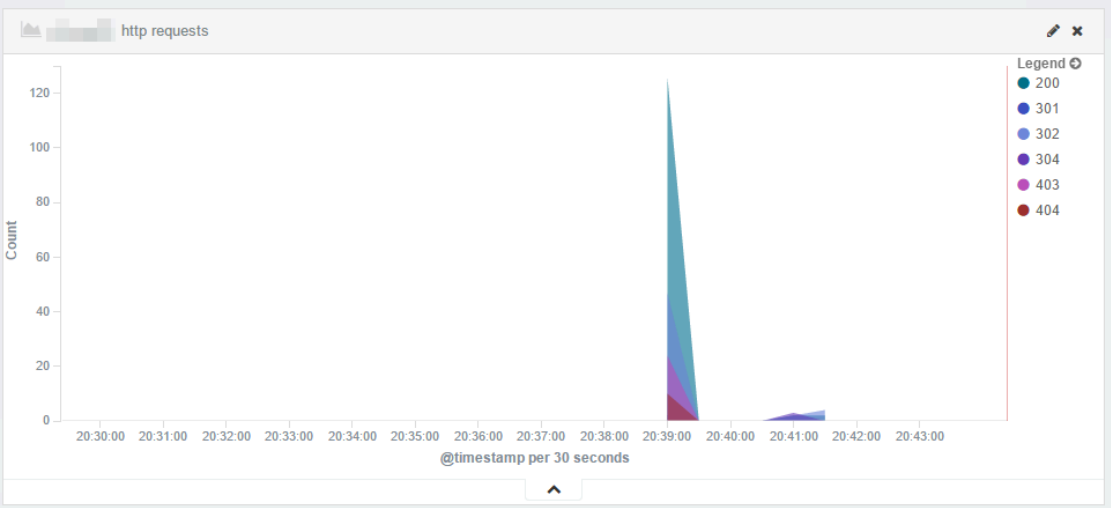

Figure 24. Burp Suite spidering in IIS logs.

Reconnaissance scan with for example DirBuster is extremely noisy and it can be identified observing http request activity and http error activity starting at 13:10 in Figure 25. In this example DirBuster was run with default settings and with the smallest directory list which is included in DirBuster. DirBuster generates around 40.000 requests in five minutes. If that is compared to the busiest environment, which has just below 20.000 requests in five minutes, reconnaissance scan using DirBuster with default settings is very easy to detect by observing the request count. One could try to do reconnaissance scan more discreetly by limiting the request count per second, however, this kind of reconnaissance scan, by nature, is easy to detect because most of the requests have HTTP 404 as a response, which is very rare in production environment.

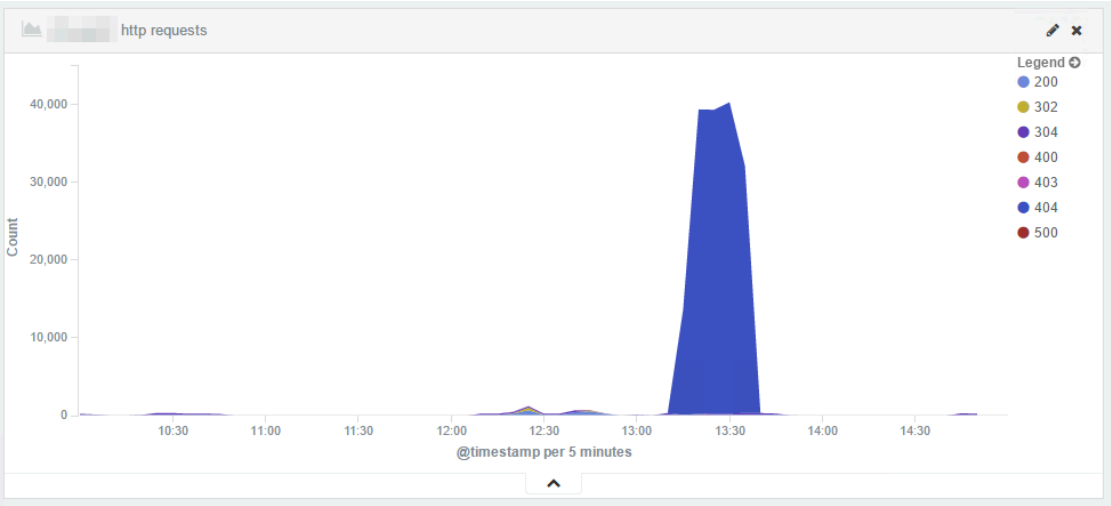

Figure 25. DirBuster scanning in web server logs.

One of the most critical vulnerabilities in web applications is SQL injection according to OWASP top 10. If SQLi vulnerability exists in applications, it is possible for threat agent to compromise the whole application and its data. Figure 26 illustrates data exfiltration from application database via SQLi vulnerability with a tool called SQLmap. The application did not have any other users at the time of testing. Exfiltration can be made in three different ways, using application's successful or erroneous response when exploiting normal SQLi or using database's time delays when exploiting blind SQLi. When exfiltration is made using the application's error messages, it is relatively noisy and can be identified observing http response codes at 21:10. Data exfiltration tests are also identified at 21:25 and 21:50. If exploiting is made without causing error messages and using HTTP POST method, it stays relatively well under the radar and it is much harder to identify.

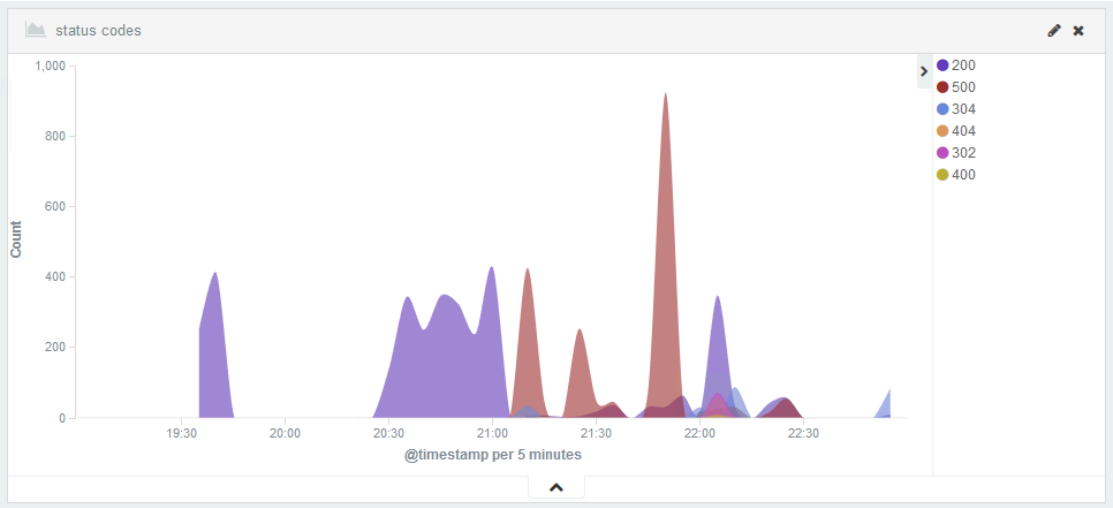

Figure 26. Data exfiltration via SQLi in web server logs.

6.2 Firewall (Windows Firewall)

To see and verify the value of situational awareness, the following tests were made regarding firewall logs - intentional misconfiguration trying to use a port which is not allowed, and port scanning. Another method for verification is to look for characteristics in the production environment's visualization and compare the normal situation's visualization to the visualization from situation where there is an anomaly and to find an explanation for it.

Figure 27 illustrates how the tested activity shows in situational awareness charts. The outgoing access attempts were made using HTTP and HTTPS. Access attempts can be identified at 17:25 in Figure 27. The outgoing connection attempts were tested using also nslookup and connection attempts can be identified at 17:35. When comparing these charts to the actual production environment activity, normally blocked outgoing connection attempts occur 2-3 times per hour per server at the peak times, for example Windows 2012R2 tries to send telemetry information to Microsoft. Unusual connection attempt count is at least twice over the average so they are quite well distinguishable. When suspicious activity is detected, necessary information for troubleshooting or investigation is available in the log event.

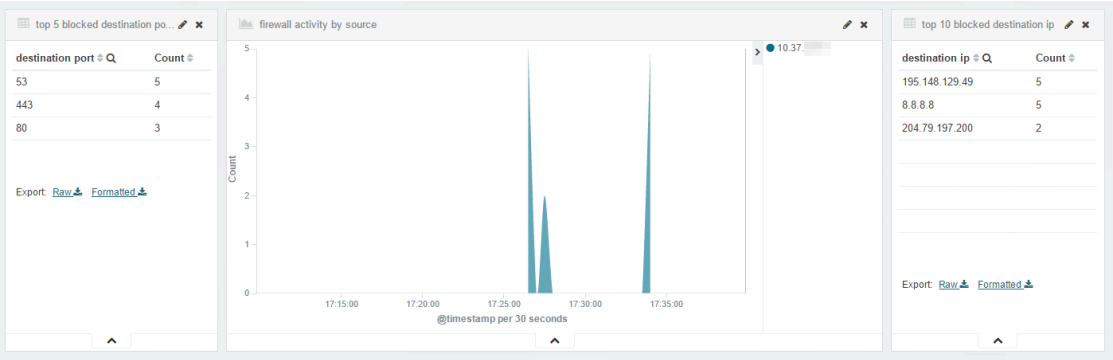

Figure 27. Blocked connection attempts from firewall logs.

In chapter 5.4.2 two anomalies were identified in Figure 18. When those spikes were investigated using destination IP addresses they were Windows 2012R2 trying to connect to Microsoft using addresses go.microsoft.com, statsfe2.update.microsoft.com and fe2.update.microsoft.com. According to Microsoft community, those addresses are used by Windows Update. The server environment is configured to download and report installed updates to internal Windows Server Update Services -server, so there is no need to allow reporting to Microsoft.

The incoming access attempts were tested using nmap as a port scanner. Firewall, by design, should log every connection attempt if configured accordingly. During testing it was noticed that there is either a bug or design flaw in Microsoft's Windows Firewall which logs dropped connection attempts only if there is active service bound and listening to port. This behavior was tested with fully patched Windows Server 2008R2, 2012R2 and 2016 Technical Preview 4. Microsoft's support was contacted through Twitter and they suggested writing a question to their community support forum. The question was posted April 11th and it has not been answered yet. Finnish Communications Regulatory Authority was contacted regarding this issue on April 24th and they forwarded the question to Microsoft. There has been no reply regarding this issue to date. Port scanning is impossible to be detected using log based situational awareness if the log source is Windows Firewall.

6.3 DNS (Microsoft DNS)

To see and verify the value of situational awareness, anomaly detection capability was tested with intentional misconfiguration requesting missing DNS record. Another method for verification is to look for characteristics in the production environment's visualization and compare the visualization of a normal situation to the visualization from a situation where there is anomaly and to find an explanation for it.

Figure 28 is from the production environment and illustrates an anomaly in Tuesday night at 23:00. The anomaly was quickly resolved, it was caused by Windows' security update installation, which is scheduled to start at 23:00. During the installation start DNS query count shows a clear spike which is more than twice over the average in normal situation.

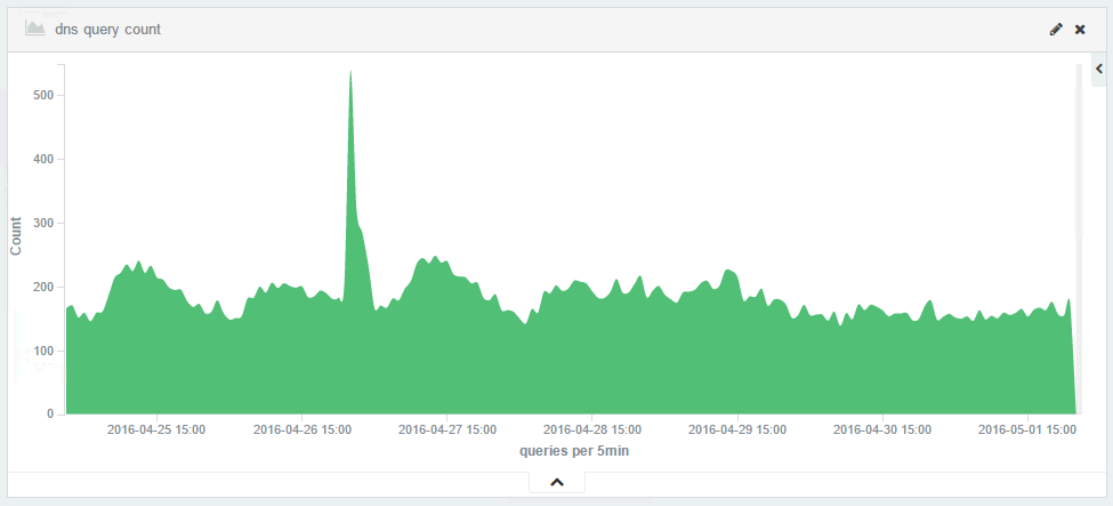

Figure 28. Windows security update installation in DNS logs.

Figure 29 illustrates what using non-existent address looks like, the response from DNS server is always NXDOMAIN whether the cause is missing non-existent DNS record or a typo in configuration because the DNS server cannot make a difference. Test configuration used address "*testing.for.thesis*" which does not exist. When a DNS query has NXDOMAIN as its response, maintenance team can easily check that DNS request's source and it guides the troubleshooting immediately to the correct server. Single erroneous DNS query produces four failed DNS responses because DNS client queries both A

and AAAA types of records and if that is not successful, it tries to append active directory's domain as a suffix.

Failed wpad and isatap queries shown in Figure 29 are Windows' default features, Web Proxy Autodiscovery Protocol and Intra-Site Automatic Tunnel Addressing Protocol. Neither one of them is configured.

Failed DNS responses are extremely rare on correctly configured production environment. The cause of failed DNS responses should always be investigated and resolved.

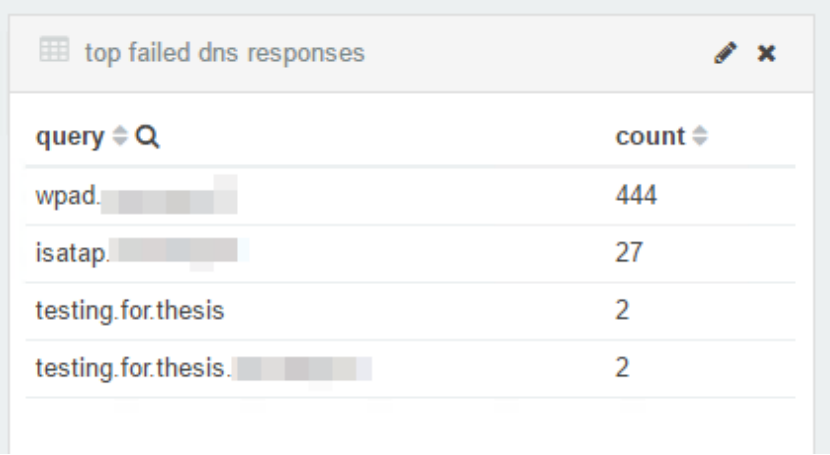

Figure 29. Missing DNS record queried.

6.4 Event log (Windows)

To see and verify the value of situational awareness, anomaly detection capability was tested by trying to log in to servers without necessary credentials, causing intentional error events and also by looking for anomalies in the actual production environment's visualization.

Figure 20 illustrated actual data from few days and it showed two high spikes of system errors. The errors were caused by something trying to connect to IIS HTTPS website using HTTP in port 443. These are extremely hard to investigate because Windows does not log the source address.

Figure 30 illustrates the system log's error messages during testing. In a normal situation the average error rate is practically near zero, so every error might be an anomaly which should be investigated and resolved. Two errors in the middle were caused by misconfiguration with NXLog during testing and starting NXLog caused an error. The last higher anomaly was caused by schannel when https website was accessed with http protocol. The first two error sets are from a different server and were caused by a client connecting using unsupported TLS cipher.

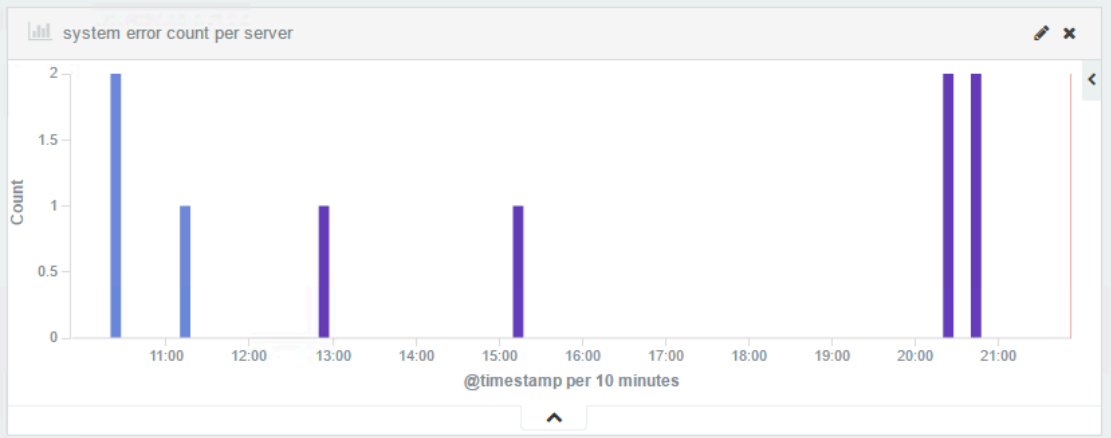

Figure 30. System errors in logs.

Figure 31 illustrates the failed logins by logon type and hostname. Different logon types are distributed to rows and hostnames are illustrated with different colors. In normal a situation the average error rate is practically near zero, so every failed login is an anomaly in theory. Sometimes authorized users happen to remember or type their passwords wrong at the first time, so few failed logins now and then are not abnormal. The failed logins were tested within two different servers and using different logon failures. The failed logon types are 3 for network and 7 for unlock. First one was accomplished by trying to open a remote desktop connection to the server. By default, the remote desktop connection would show as logon type 10, remote interactive, however, the servers have network-level authentication enabled which forces the client to authenticate with type 3 at first. Logon type 7 error was accomplished by locking the remote session and trying to unlock using a wrong password.

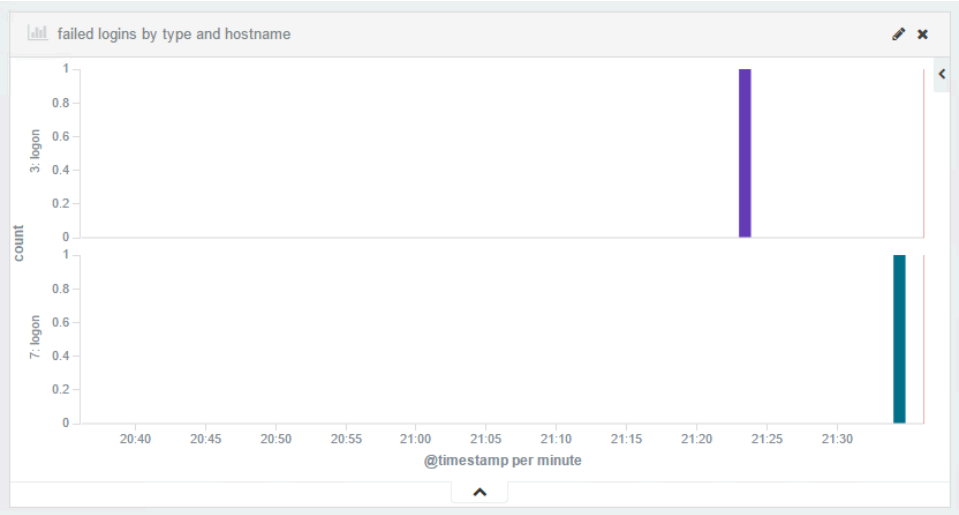

Figure 31. Failed logins visualized.

Figure 32 illustrates successful logins by logon type and hostname. Different logon types are distributed to rows and hostnames are illustrated with different colors. Because NLA is forced, successful remote desktop login produces both logon type 3 and 10 at 21:35. Regarding anomalies e.g. logon type 2, console login should never be seen after the server is installed and joined to network and domain. If logon type 2 came up, it should be investigated immediately.

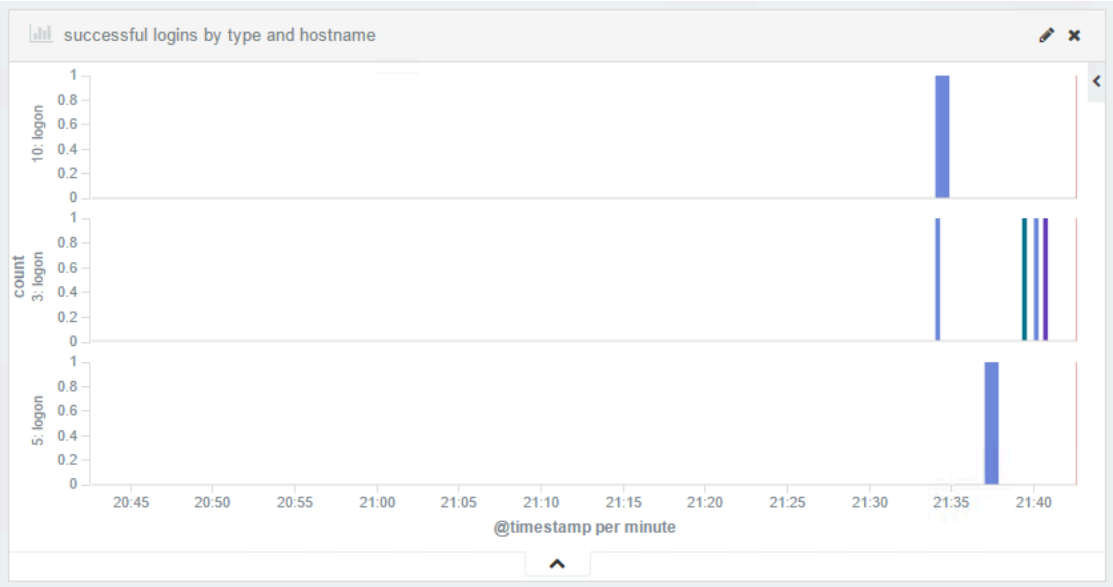

Figure 32. Successful logins visualized.

7 RESULTS

The result of this thesis was an internal situational awareness system in an IaaS server environment within the defined scope and verification that it fulfills the requirement which was capability to detect anomalies in information system. The tests and examples were chosen bearing in mind that the anomalies may be a result of a malfunction of an information system or an impact of an external threat agent.

The situational awareness system was built to our IaaS-environment using one virtual server for ELK stack and using either production or test servers as log sources. The production servers were used to observe actual situations and issues where the test environment was used to verify the capabilities of the SA system regarding malicious activity and configuration errors.

The cost of the SA system was almost zero because it was built to our dedicated environment using existing resources. No commercial licenses, support hours or products were needed.

The implemented SA system was tested and verified using common methods threat agents are using in reconnaissance scan, vulnerability assessment and exploitation phase. The benefits of SA system with configuration errors were tested making intentional mistakes in most probable locations that could affect the system.

The reliability of the used test methods differs between log sources. Situational awareness and the capability to detect anomalies from DNS logs is very accurate and reliable because of the nature of DNS protocol, it consists mainly of queries and responses. With situational awareness system DNS logs were successfully visualized in the way that most of the imaginable anomalies should be easily detectable.

In the other end of reliability is the capability to detect an anomaly from firewall logs. The detection capability was easy to test, however, the biggest problem

was a bug or design flaw in Windows Firewall's logging, it logs the rejected network packets only if there is an active service listening. This issue prevents the detection of most of the anomalies that could be detected from firewall logs, e.g. reconnaissance scan, such as port scan using nmap. In practice, Windows Firewall logs can be used at the moment only to detect configuration errors. Microsoft did not respond regarding this issue despite of several requests.

In the middle of the reliability scale in anomaly detection capability are situational awareness from IIS and event logs. Detection capability from IIS logs was tested using common tools and methods, with default options, that malicious actors are using. Not all malicious activities were easy to detect and they would be harder to detect if the malicious actor tried to keep as low profile as possible, e.g. making tests manually. In the scope of this thesis event log visualization was quite narrow, only logins, errors and warnings. Regarding the logged events, the detection capability is good and reliable but the visibility is slightly restricted by default.

8 CONCLUSIONS

The main question of this thesis was how to build situational awareness system and is it possible fulfil the requirement of detecting anomalies in information systems using situational awareness system. The thesis was scoped the way that it included only operating system and application logs hypervisor and network devices were left outside of the scope of this thesis.

The source material was relatively easy to find. There were few key publishers who were cited in many publications. After few keywords it was possible to derive principles, such as OODA loop from information technology frameworks for gaining situational awareness.

The selected research method supported this thesis well since Design Science Research Methodology for Information Systems Research aims to create results that can serve human purposes. The structure of the selected research methodology was also well suitable and logical for this kind of thesis.

The section on situational awareness theory discussed the terms, some history and principles how to form situational awareness. It was the foundation based on which different methods were evaluated in theory.

The situational awareness system was originally made to fulfill the requirements from mandatory Finnish national regulations and to be able to detect anomalies in information systems. On the other hand, it is not good to do something just to fulfill regulations. It might give a false sense of security. Situational awareness system sounded like a system which would give value to the operations team.

The implementation phase of the log normalization took several rounds of testing and iteration before the situational awareness could be formed with the required accuracy. The challenge was to know which elements of information are important to monitor and how to apply correct filters to them as stated in the observation theory chapter.

As the result of this thesis, situational awareness system was implemented and tested with different kinds of situations. Most of the tested example cases were identified as anomalies by observing situational awareness graphs. Building a foolproof system which would be able to identify every possible known and unknown anomaly is not possible at the moment. Every solution in the anomaly detection sector has to balance with false positives and false negatives, where the first one is normal activity reported as an anomaly and the latter is an actual anomaly interpreted as normal activity.

Since the SA system was deployed in the first production environment, it has proven to be extremely valuable tool. We have been able to identify production environment issues immediately as they have occurred, issues which would have been almost impossible to identify by other means than detecting anomaly. The ELK stack's current versions are relatively stable and mature and there is commercial support available with premium SLA levels. In my opinion, the objective of this thesis is met and the implemented SA system has given value to the operations team and whole organization.

9 FUTURE RESEARCH

There are some features which can be used to extend the usability of the situational awareness system created during this thesis. There is also some optimization which can be used to optimize the speed and required disk space in Elasticsearch.

The most useful feature to extend the usability of the SA system is alerting. At the beginning of this thesis there was only one option for alerting, Elastic's product called Watcher. The only downside with it is that it requires an expensive subscription. For example, a single node for production costs over 4.000€ per year and five nodes over 17.000€ per year. During this thesis Yelp has released an update to its framework called ElastAlert and it now supports Elasticsearch version 2.

According to Yelp "*ElastAlert is a simple framework for alerting on anomalies, spikes, or other patterns of interest from data in Elasticsearch*". (Running ElastAlert for the First Time 2016)

One relatively easy minor optimization in the near future would be data type optimization in Elasticsearch. By default, Elasticsearch handles integers and strings but more complex formats such as IPv4 addresses has to be configured manually in Elasticsearch index template. The correctly configured datatypes regarding IPv4 addresses enable the usage of IP address and networks based visualization in Kibana and therefore, better situational awareness.

To broaden the visibility in the situational awareness system, and to improve the organization's ability to detect anomalies, more information can be added. It can be accomplished by adding more log sources or by adding logging to important or interesting activity in current log sources. Sysinternals has released a tool called Sysmon in 2014 which is under active development and

enables monitoring and logging in a more granular way than Windows offers by default. With Sysmon it is possible to monitor and detect changes in files, e.g. hosts file changes would be extremely interesting events.

In the recent few years, artificial intelligence and machine learning have taken huge leaps. For example, few years ago there were the first automatic anomaly detection systems released to consumers which were based on pattern recognition. After that structure based anomaly detection systems appeared. Next on the line, a method called metric based anomaly detection was developed. It can take any data which has a numerical value in its timeline. These new methods can automatically baseline the given data and alert the operator about anomalies using artificial intelligence. So in the near future, the use of artificial intelligence can improve automatic anomaly detection substantially.

REFERENCES

Anicas M. 2015. How To Install Elasticsearch, Logstash, and Kibana (ELK Stack) on Ubuntu 14.04. Accessed on 14 February 2016. Retrieved from https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elk-stack-on-ubuntu-14-04

Buchsein R., Dettmer K. 2008. ISO/IEC 20000 IT Service Management - Benefits and Requirements for Service Providers and Customers. Accessed on 10 April 2016. Retrieved from http://www.eprogram.com.au/attachments/article/57/ISO20000%20and%20ITIL.pdf

Curator Readme 2016. Page on Github's website. Accessed on 8 May 2016. Retrieved from https://github.com/elastic/curator/blob/master/README.rst

Data sheet SteelCentral NetShark 2015. PDF document on Riverbed's website. Accessed on 10 April 2016. Retrieved from http://www.riverbed.com/document/fpo/0442_SteelCentral-NetShark_DS_061815AS.pdf

Data Sheet: SteelCentral Web Analyzer 2014. PDF document on WANSolutionWorks' website. Accessed on 14 February 2016. Retrieved from http://www.wansolutionworks.com/datasheets/DataSheet_SteelCentral_Web_Analyzer.pdf

ElastAlert Documentation 2016. PDF document on Read the Docs' website. Accessed on 10 April 2016. Retrieved from https://media.readthedocs.org/pdf/elastalert/latest/elastalert.pdf

Elasticsearch Reference 2016. Page on Elastic's website. Accessed on 8 May 2016. Retrieved from https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html

Endsley, M. R. 1995. Toward a theory of situation awareness in dynamic systems. Human Factors, 37 (1), 32-64.

Event Logs 2014. Page on Microsoft's website. Accessed on 10 April 2016. Retrieved from https://technet.microsoft.com/en-us/library/cc722404(v=ws.11).aspx

Getting Kibana Up and Running 2016. Page on Elastic's website. Accessed on 10 April 2016. Retrieved from https://www.elastic.co/guide/en/kibana/current/setup.html

Hodgson J. 2015. SteelCentral AppInternals 10: The best just got a whole lot better. Accessed on 10 April 2016. Retrieved from http://www.riverbed.com/blogs/SteelCentral-AppInternals-10-The-best-just-got-a-whole-lot-better.html

Interpreting the Windows Firewall Log 2005. Page on Microsoft's website. Accessed on 24 April 2016. Retrieved from https://technet.microsoft.com/en-us/library/cc758040%28v=ws.10%29.aspx

Kelly M. 2014 Applying military strategy to high risk decision making and operational learning processes for on-snow practitioners, 1056-1059

Kibana Introduction 2016. Page on Elastic's website. Accessed on 8 May 2016. Retrieved from https://www.elastic.co/guide/en/kibana/current/introduction.html

Liite 1 Tietoturvavaatimukset A-luokkaan kuuluville järjestelmille ja järjestelmien käyttöympäristöille 2015 [Appendix 1 Data security requirements of Class A systems and system operating environments 2015]. PDF document on THL's website. Accessed on 24 January 2016. Retrieved from https://www.thl.fi/documents/920442/1449818/Liite_1_THL_Määräys_1_2015_Tietoturvavaatimukset_201501.pdf/f2817278-2843-4c2d-b038-bac88dd9691d

Logstash Introduction 2016. Page on Elastic's website. Accessed on 10 April 2016. Retrieved from https://www.elastic.co/guide/en/logstash/current/introduction.html

Lokitiedot tietoturvallisuuden tukena 2016 [Supporting information security with log data 2016]. Page on Finnish Communications Regulatory Authority's website. Accessed on 27 March 2016. Retrieved from https://www.viestintavirasto.fi/kyberturvallisuus/tietoturvanyt/2016/03/ttn20160 3040919.html

McKay B., McKay K. 2015. How to Develop the Situational Awareness of Jason Bourne. Accessed on 30 January 2016. Retrieved from http://www.artofmanliness.com/2015/02/05/how-to-develop-the-situational-awareness-of-jason-bourne/

Nginx 2016. Page on Nginx's website. Accessed on 8 May 2016. Retrieved from http://nginx.org/en/

NXLog Community Edition 2016. Page on Nxlog's website. Accessed on 24 April 2016. Retrieved from http://nxlog.net/products/nxlog-community-edition

Oosthuizen R., Pretorius L. 2015. System Dynamics Modelling of Situation Awareness.

OWASP Top 10 2013. Page on OWASP's website. Accessed on 27 March 2016. Retrieved from https://www.owasp.org/index.php/Top_10_2013-Top_10

Peffers K., Tuunanen T., Rothenberger M., Chatterjee S. 2007. A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems 24, 3, 45-78.

Richards C. 2012 Boyd's OODA Loop. PDF document on JV/M Inc's website. Accessed on 10 April 2016. Retrieved from http://www.jvminc.com/boydsrealooda_loop.pdf

Running ElastAlert for the First Time 2016. Page on Read the Docs' website. Accessed on 8 May 2016. Retrieved from http://elastalert.readthedocs.io/en/latest/running_elastalert.html

Sysmon v4.0 2016. Page on Microsoft's website. Accessed on 8 May 2016. Retrieved from https://technet.microsoft.com/en-us/sysinternals/dn798348

The Elastic Stack 2016. Page on Elastic's website. Accessed on 24 April 2016. Retrieved from https://www.elastic.co/products

Tremblay P.J. 2015. Shaping and Adapting - Unlocking the power of Colonel John Boyd's OODA Loop

Vitec in Brief 2016. PDF document on Vitec's website. Accessed on 24 April 2016. Retrieved from http://www.vitecsoftware.com/Global/Moderbolaget/Om%20Vitec/Vitec%20in%20brief%202016.pdf

Williams, D.K. 2013. What A Fighter Pilot Knows About Business: The OODA Loop. Accessed on 3 January 2016. Retrieved from http://www.forbes.com/sites/davidkwilliams/2013/02/19/what-a-fighter-pilot-knows-about-business-the-ooda-loop/

APPENDICES

APPENDIX 1 - Logstash configuration for log input from NXLog

/etc/logstash/conf.d/10-input.conf

```
input {

  # dns logs
  tcp {
    type => "dnslog"
    port => 4001
    codec => "json"
  }

  # event logs
  tcp {
    type => "eventlog"
    port => 4002
    codec => "json"
  }

  # firewall logs
  tcp {
    type => "fwlog"
    port => 4003
    codec => "json"
  }

  # iis logs
  tcp {
    type => "iislog"
    port => 4004
    codec => "json"
  }

}
```

APPENDIX 2 - Configurations for log forwarding and normalization regarding IIS

NXLog

```
C:\Program Files (x86)\nxlog\conf\nxlog.conf


# Parse the log event as csv
<Extension w3c>
    Module xm_csv
    Fields $date, $time, $s-computername, $cs-method, $cs-uri-
stem, $cs-uri-query, $cs-username $c-ip, $cs-User-Agent, $cs-
Referer, $cs-host, $sc-status, $sc-substatus, $sc-win32-status,
$sc-bytes, $cs-bytes, $time-taken
    FieldTypes string, string, string, string, string, string,
string, string, string, string, string, integer, integer,
integer, integer, integer, integer
    Delimiter ' '
</Extension>

<Extension json>
 Module xm_json
</Extension>

<Input iis>
    # custom location for IIS logs
    Module    im_file
    File    "D:\Systemlogs\W3SVC1\u_ex*.log"
    SavePos  TRUE

    # drop comment lines
    Exec if $raw_event =~ /^#/ drop();                \
       else                                           \
       {                                        \
            w3c->parse_csv();                         \
            $EventTime = parsedate($date + " " + $time);    \
            $SourceName = "IIS";                      \
            $raw_event = to_json();                   \
       }
</Input>

<Output out_iis>
    Module      om_tcp
    Host        10.37.x.x
    Port        4004
</Output>

<Route 1>
     Path       iis => out_iis
</Route>
```

Logstash

/etc/logstash/conf.d/20-iislog.conf

```
filter {
  if [type] == "iislog" {

    # remove unnecessary fields produced by NxLog
    mutate {
      remove_field =>
["port","SourceModuleType","date","time","SourceModuleName","So
urceName"]
    }

    # match timestamp with given format and timezone
    date {
      match => ["EventTime","YYYY-MM-dd HH:mm:ss"]
      timezone => "Etc/UCT"
    }

    # extract useragent information
    useragent {
      source => "cs-User-Agent"
    }

    # remove unnecessary useragent information
    mutate {
      remove_field =>
["EventTime","minor","os_name","device","cs-User-Agent"]
    }
  }
}
```

APPENDIX 3 - Configurations for log forwarding and normalization regarding Firewall

NXLog

C:\Program Files (x86)\nxlog\conf\nxlog.conf

```
<Extension json>
     Module xm_json
</Extension>

<Input firewall>
     Module          im_file
     # Default firewall log location
     File
     'C:\Windows\Sysnative\LogFiles\Firewall\pfirewall.log'
     SavePos         TRUE
     InputType  LineBased
     Exec $Message = $raw_event;
     # drop empty messages
     Exec if $Message == '' drop();
     # drop comment lines
     Exec if $Message =~ /^#/ drop();
     # format the message as json
     Exec to_json();
</Input>

<Output out_iis>
    Module      om_tcp
    Host        10.37.x.x
    Port        4003
</Output>

<Output out_debug>
     Module          om_file
     File       'C:\temp\nxlog_debug.log'
</Output>

<Route 1>
    Path        firewall => out_iis
   #Path          firewall => out_debug
</Route>
```

Logstash

/etc/logstash/conf.d/20-fw.conf

```
filter {
  if [type] == "fwlog" {
    grok {
       # match log structure using grok
      match => { "Message" => "%{DATE_EU:date} %{TIME:time}
%{WORD:action} %{WORD:protocol} %{IP:source_ip}
%{IP:destination_ip} %{USERNAME:src-port} %{USERNAME:dst-port}
%{USERNAME:size} %{USERNAME:tcpflags} %{USERNAME:tcpsyn}
%{USERNAME:tcpack} %{USERNAME:tcpwin} %{USERNAME:icmptype}
%{USERNAME:icmpcode} %{USERNAME:info} %{WORD:direction}" }
    }

    # combine date and time, remove unnecessary fields
    mutate {
      add_field => { "eventtime" => "%{date} %{time}" }
      remove_field => [
"port","date","time","SourceModuleName","SourceModuleType","Mes
sage","icmpcode","icmptype","info","size","tcpack","tcpflags","
tcpsyn","tcpwin" ]
    }

    # match datetime with given format
    date {
      match => ["eventtime","yy-MM-dd HH:mm:ss"]
    }
  }
}
```

APPENDIX 4 - Configurations for log forwarding and normalization regarding DNS

NXLog

```
C:\Program Files (x86)\nxlog\conf\nxlog.conf


<Extension json>
     Module xm_json
</Extension>

<Input DNS>
     Module          im_file
     # default dns log file location
     File       'C:\Windows\Sysnative\dns\dns.log'
     SavePos         TRUE
     InputType  LineBased
     Exec $Message = $raw_event;
     # drop empty lines
     Exec if $Message == '' drop();
     # format the message as json
     Exec to_json();
</Input>

<Output out_dns>
    Module      om_tcp
    Host        10.37.x.x
    Port        4001
</Output>


<Output out_debug>
     Module          om_file
     File       'C:\temp\nxlog_debug.log'
</Output>

<Route 1>
    Path        DNS => out_dns
   #Path        DNS => out_debug
</Route>
```

Logstash

/etc/logstash/conf.d/20-dns.conf

```
filter {
  if [type] == "dnslog" {
    grok {
      match => { "Message" => "%{DATE_EU:date} %{TIME:time}
%{WORD:dns_thread_id}
%{WORD:dns_context}%{SPACE}%{WORD:dns_packet_id}
%{WORD:dns_ip_protocol} %{WORD:dns_direction}
%{IP:dns_client_address}%{SPACE}%{WORD:dns_xid}%{SPACE}
%{GREEDYDATA:queryresponse}
%{SPACE}\[%{WORD:dns_hex_flags}%{SPACE}%{GREEDYDATA:dns_hex_fla
gs}%{SPACE}%{WORD:dns_response}\]%{SPACE}%{WORD:dns_recordtype}
%{SPACE}%{GREEDYDATA:dns_query_name}" }
    }

    mutate {
      add_field => { "eventtime" => "%{date} %{time}" }
      remove_field =>
["dns_context","Message","dns_hex_flags","dns_ip_protocol","dns
_packet_id","dns_thread_id","dns_xid","date","time","port","Eve
ntReceivedTime","SourceModuleName","SourceModuleType"]

      gsub => [
        # remove leading, trailing and inner (n)
        "dns_query_name", "^\(\d+\)", "",
        "dns_query_name", "\(\d+\)$", "",
        "dns_query_name", "\(\d+\)", "."
      ]

      # convert all queries to lowercase
      lowercase => ["dns_query_name"]
    }

    # match timestamp with given format
    date {
      match => ["eventtime","d.M.yyyy HH:mm:ss"]
    }

    # add dns servers hostname based on ip address
    if [host] == "10.37.x.x" {
      mutate {
        add_field => { "hostname" => "dnsservername1" }
      }
    }
    if [host] == "10.37.x.x" {
      mutate {
        add_field => { "hostname" => " dnsservername2" }
      }
    }
  }
}
```

APPENDIX 5 - Configurations for log forwarding and normalization regarding Event log

NXLog

```
C:\Program Files (x86)\nxlog\conf\nxlog.conf


<Extension json>
      Module xm_json
</Extension>

# Windows Event Log
<Input eventlog>
      Module im_msvistalog
      # format the message as json
      Exec to_json();
</Input>

<Output out_eventlog>
    Module       om_tcp
    Host         10.37.x.x
    Port         4002
</Output>

<Output out_debug>
      Module           om_file
      File       'C:\temp\nxlog_debug.log'
</Output>

<Route 2>
      Path       eventlog => out_eventlog
</Route>
```

Logstash

/etc/logstash/conf.d/20-eventlog.conf

```
filter {
  if [type] == "eventlog" {
    # drop unnecessary fields
    mutate {
      remove_field => [
"EventReceivedTime","IpPort","KeyLength","Keywords","LogonGuid"
,"ProviderGuid","RecordNumber","SourceModuleType","SubjectLogon
Id","SubjectUserName","SubjectUserSid","TargetLogonId","TargetU
serSid","ThreadID","Message","host","port","Status","SubStatus"
,"Opcode","OpcodeValue","SourceModuleName","Task","TransmittedS
ervices","Version","ImpersonationLevel" ]
    }

    # match timestamp with given format
    date {
      match => ["EventTime","yyyy-MM-dd HH:mm:ss"]
    }
    # remove datetime field and convert three fields to
lowercase
    mutate {
      remove_field => [ "EventTime" ]
      lowercase =>
["Hostname","TargetUserName","TargetDomainName"]
    }
  }
}
```

APPENDIX 6 - Logstash configuration for log output to Elasticsearch

/etc/logstash/conf.d/30-output.conf

```
output {
  elasticsearch { hosts => ["localhost:9200"] }
  # for debugging
  #stdout { codec => rubydebug }
}
```

APPENDIX 7 - Windows Event Log example in XML-format

```
<Event
xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
  <System>
    <Provider Name="Microsoft-Windows-Security-Auditing"
Guid="{54849625-5478-4994-A5BA-3E3B0328C30D}" />
    <EventID>4624</EventID>
    <Version>1</Version>
    <Level>0</Level>
    <Task>12544</Task>
    <Opcode>0</Opcode>
    <Keywords>0x8020000000000000</Keywords>
    <TimeCreated SystemTime="2016-05-14T13:51:57.590574300Z" />
    <EventRecordID>657937</EventRecordID>
    <Correlation />
    <Execution ProcessID="472" ThreadID="2688" />
    <Channel>Security</Channel>
    <Computer>targetcomputerfqdn</Computer>
    <Security />
  </System>
  <EventData>
    <Data Name="SubjectUserSid">S-1-5-18</Data>
    <Data Name="SubjectUserName">hostname$</Data>
    <Data Name="SubjectDomainName">domainname</Data>
    <Data Name="SubjectLogonId">0x3e7</Data>
    <Data Name="TargetUserSid">S-1-5-21-2653058874-47457212-
3613976224-1157</Data>
    <Data Name="TargetUserName">timo.hulkkonen</Data>
    <Data Name="TargetDomainName">domainname</Data>
    <Data Name="TargetLogonId">0x110ce7ba</Data>
    <Data Name="LogonType">10</Data>
    <Data Name="LogonProcessName">User32 </Data>
    <Data Name="AuthenticationPackageName">Negotiate</Data>
    <Data Name="WorkstationName">hostname</Data>
    <Data Name="LogonGuid">{33CF6AC7-0520-659A-DAF7-
EB9E856111BC}</Data>
    <Data Name="TransmittedServices">-</Data>
    <Data Name="LmPackageName">-</Data>
    <Data Name="KeyLength">0</Data>
    <Data Name="ProcessId">0xf9c</Data>
    <Data
Name="ProcessName">C:\Windows\System32\winlogon.exe</Data>
    <Data Name="IpAddress">10.39.x.x</Data>
    <Data Name="IpPort">0</Data>
    <Data Name="ImpersonationLevel">%%1833</Data>
  </EventData>
</Event>
```