Big Data Analytics Attack Detection for Critical Information Infrastructure Protection

Floris Stouten

Information Security, masters level
2016

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
‘Big data analytics attack detection for Critical Information Infrastructure Protection’

Floris Stouten (flosto-3)
06-10-2016
ABSTRACT

Attacks on critical information infrastructure are increasing in volume and sophistication with destructive consequences according to the 2015 Cyber Supply Chain Security Revisited report from ESG recently (ESG, 2015). In a world of connectivity and data dependency, cyber-crime is on the rise causing many disruptions in our way of living. Our society relies on these critical information infrastructures for our social and economic well-being, and become more complex due to many integrated systems.

Over the past years, various research contributions have been made to provide intrusion detection solutions to address these complex attack problems. Even though various research attempts have been made, shortcomings still exists in these solutions to provide attack detection. False positives and false negatives outcomes for attack detection are still known shortcomings that must be addressed.

This study contributes research, by finding a solution for the found shortcomings by designing an IT artifact framework based on the Design Science Research Methodology (DSRM). The framework consist of big data analytics technology that provides attack detection.

Research outcomes for this study shows a possible solution to the shortcomings by the designed IT artifact framework with use of big data analytics technology. The framework built on open source technology can provide attack detection, and possibly provide a solution to improve the false positives and false negatives for attack detection outcomes. Three main modules have been designed and demonstrated, whereby a hybrid approach for detection is used to address the shortcomings. Therefore, this research can benefit Critical Information Infrastructure Protection (CIIP) in Sweden to detect attacks and can possibly be utilized in various network infrastructures.
CONTENTS

ABSTRACT ......................................................................................................................... 2
1. ABBREVIATIONS ............................................................................................................. 5
2. INTRODUCTION ............................................................................................................... 6
2.1 PROBLEM DESCRIPTION .......................................................................................... 6
2.2 RESEARCH QUESTION ............................................................................................... 7
3. SCOPE DELIMITATION AND RISKS .......................................................................... 8
4. RESEARCH METHODOLOGY ......................................................................................... 9
4.1 Activity 1: Problem identification and motivation ....................................................... 9
4.2 Activity 2: Define the objectives for a solution ............................................................ 9
4.3 Activity 3: Design and development ........................................................................... 10
4.4 Activity 4: Demonstration ......................................................................................... 10
4.5 Activity 5: Evaluation ................................................................................................ 10
4.6 Activity 6: Communication ......................................................................................... 10
5. LITERATURE REVIEW ................................................................................................. 11
5.1 Literature review method ........................................................................................... 11
5.2 Critical information infrastructure .......................................................................... 12
5.3 Big data ..................................................................................................................... 13
5.4 Big data sources ......................................................................................................... 13
5.5 Big data analytics ....................................................................................................... 14
5.6 Big data analytics similar research frameworks ......................................................... 14
5.7 Attack types on critical information infrastructure ...................................................... 15
5.8 Attack detection ......................................................................................................... 17
5.9 Attack detection challenges ....................................................................................... 20
5.9.1 Attack detection challenges pre 2010 .................................................................. 20
5.9.2 Attack detection challenges from 2010 .................................................................. 22
5.10 Research gaps ........................................................................................................... 27
6. DESIGN AND DEVELOPMENT ..................................................................................... 28
6.1 Functional requirements ............................................................................................. 28
6.2 Functional design methodology ................................................................................. 28
6.3 Kernel theory based framework ................................................................................. 29
6.4 Framework overview ................................................................................................ 30
6.4.1 Data gathering, receiving and filtering module ...................................................... 30
6.4.2 Analytics module .................................................................................................... 33
6.4.3 Data log storage module ....................................................................................... 36
7. DEMONSTRATION

7.1 Demonstration technical setup

7.1.1 ‘Data gathering, receiving and filtering’ module

7.1.2 ‘Analytic’ module Apache Storm

7.1.3 ‘Analytic’ module Apache Spark

7.1.3.1 Loading data

7.1.3.2 Machine learning data processing

7.1.3.3 Machine learning knowledge outcomes

7.1.4 ‘Data log storage’ module

7.2 Attack demonstration walkthrough (scenario) approach

7.2.1 Denial of Service (DOS)

7.2.2 Remote to User attack (R2L)

7.2.3 User to Root attack (U2R)

7.2.4 Probing attack

8. EVALUATION

8.1 Evaluation logical argumentation

9. DISCUSSION

10. CONCLUSION

10.1 Future research

11. REFERENCES
1. **ABBREVIATIONS**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APT</td>
<td>Advanced Persistent Threats</td>
</tr>
<tr>
<td>BSI</td>
<td>Bundesamt für Sicherheit in der Informationstechnik (Federal Office for Information Security)</td>
</tr>
<tr>
<td>CIIP</td>
<td>Critical Information Infrastructure Protection</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma Separated Values</td>
</tr>
<tr>
<td>DSRM</td>
<td>Design Science Research Methodology</td>
</tr>
<tr>
<td>DOS</td>
<td>Denial of Service</td>
</tr>
<tr>
<td>DDOS</td>
<td>Distributed Denial Of Service</td>
</tr>
<tr>
<td>EPCIP</td>
<td>European Programme for Critical Infrastructure Protection</td>
</tr>
<tr>
<td>ESG</td>
<td>Enterprise Strategy Group</td>
</tr>
<tr>
<td>HDFS</td>
<td>Hadoop Distributed File System</td>
</tr>
<tr>
<td>ICS-CERT</td>
<td>Industrial Control Systems Cyber Emergency Response Team</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>MSB</td>
<td>Swedish Civil Contingencies Agency</td>
</tr>
<tr>
<td>OAS</td>
<td>Organization of American States</td>
</tr>
<tr>
<td>RDD</td>
<td>Resilient Distributed Dataset</td>
</tr>
<tr>
<td>R2L</td>
<td>Remote to User attacks</td>
</tr>
<tr>
<td>PLC</td>
<td>Programmable Logic Controllers</td>
</tr>
<tr>
<td>SCADA</td>
<td>Supervisory Control And Data Acquisition</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>TF-IDF</td>
<td>Term frequency-inverse document frequency ()</td>
</tr>
<tr>
<td>U2R</td>
<td>User to Root Attack</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
</tbody>
</table>
2. INTRODUCTION

In a world of connectivity and data dependency, cyber-crime is on the rise causing many disruptions in our way of living. Our society relies on infrastructures for our social and economic well-being. These infrastructures and its dependencies form the basis in the way of living and become critical and complex. Infrastructures can be considered as complex infrastructures because it consists of many integrated system. These integrated systems can be identified based on three different levels, the agent, network and system level, combined they form a complex infrastructure (van der Lei et.al, 2010) and become critical and highly informational dependent. Such critical infrastructure includes electricity, transport systems, waste disposal and many more therefore it’s essential that they are reliable, efficient and preferably sustainable (ibid).

In 2014, the Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) reported and responded to 245 incidents across all critical infrastructure sectors (ICS-CERT, 2015). The Energy sector and the Critical Manufacturing Sector were one of the primary targets and took the highest attack vector. In terms of attack sophistication, more than half of the attacks were advanced persistent threats (APT) and in many incidents the threat actors were unidentified due to lack of monitoring and detection techniques to supply evidence. Disturbingly, this resulted in most of the 245 incidents were categorized by ICS-CERT as having an unknown access vector to what extent the adversaries had access to the compromised critical infrastructures and its information.

In terms of critical infrastructure damage, the German Federal Office for Information Security BSI reported in their annual 2014 report the attack on a German steel plant (BSI, 2015). The adversary developed a sophistication social engineering attack to gain access towards the office network. Through a series of steppingstone networks the adversary successively gained access to the steelworks production network. The adversary committed damage to the control systems leading to accumulated failures to a blast furnace which resulted in sincere damage to the system.

Attacks on critical information infrastructure are increasing in volume and sophistication with destructive consequences according to the 2015 Cyber Supply Chain Security Revisited report from ESG recently (ESG, 2015). For a minimum of one third of the critical infrastructure organizations, the impact of incidents led to the disruption of critical business processes and applications and breach of valuable confidential data. These figures support the Trend Micro and the Organization of American States (OAS) report from 2015 (Trend Micro and Organization of American States, 2015). According to this report, attacks had a 43% increase over the past year, most of attacks were to steal, to destruct information and shutdown networks. Attack methods involved phishing, exploiting unpatched vulnerabilities and DDOS attacks.

Apparently with the rise of the Internet Of Things (IoT), increase of mobile and user connectivity and the various types of threats and vulnerabilities, the Internet and its infrastructures become more complex and vulnerable to attacks. How can we detect these attacks to design the appropriate Critical Information Infrastructure protection (CIIP)?

2.1 PROBLEM DESCRIPTION

Since attacks on critical infrastructures can have a devastating impact, the types, paths and the patterns of attacks must be determined to detect and control future events. By using the game theory, interaction of attacks can be modeled based upon an attacker and defender scenario. Each attacker agents determines is best strategy to attack the defending agent in the most effective way (Bier and
To support the strategic plan in line with the European Programme for Critical Infrastructure Protection (EPCIP) to implement measures before, during and after disruptions (ibid).

To support the strategic plan and by utilizing Big data analytics, attacks can be detected on an earlier stage (before) and during attacks to these societal sectors which provides better protection for a critical infrastructure.
information infrastructure. In addition, to the best of our knowledge there is scarcity of academic research that has been performed to use Big data analytics for threat or attack detection for a critical information infrastructure. To research this gap, the following research question is defined.

How can big data analytics be utilized in a designed framework to detect attacks for a Critical Information Infrastructure?

The purpose of this research is to contribute current knowledge base by designing an IT artifact in a form of a framework based on big data analytics to address the current intrusion detection shortcomings for a Critical Information Infrastructure. In addition, this framework will provide a comprehensive overview to ensure big data analytics will benefit CIIP in Sweden to detect attacks and to be utilized in various network infrastructures.

3. SCOPE DELIMITATION AND RISKS

The scope of this research will be determining the shortcomings of current detection solutions (includes detection models, theories, concepts, methods), and determining a possible solution to the shortcoming with the usage of big data analytics technology whereby a framework is designed. Since the framework requires to be utilized in various network infrastructures, commercial products will be excluded in this research. Furthermore, the created framework will provide a baseline artifact, specific details will be possibly excluded. Additionally, the used datasets for testing could possible deviate from the big data definitions or properties, or that the datasets is not tested according to Big O Notation.

Possible risks for this project could be the late setup and delivery of the demonstration environment (resources). Other possible risks would be the possible complex setup of the open source big data analytics environment, and the complexity in detection algorithms for big data analytics. Furthermore, the evaluation process could take up a lot of time to meet the objectives of the designed artifact if not controlled well.
4. RESEARCH METHODOLOGY

The objective of this research is to develop a framework artifact to solve the identified problems. Currently no framework exists in research to detect attacks for a critical information infrastructure with the use of big data analytics. To build the required framework artifact, it requires a comprehensive research methodology to determine the current available detection models, theories, concepts, methods and its shortcomings for detection in order to develop a future framework which can detect better attacks to a Critical Information Infrastructure. The chosen research methodology will be based on the Design Science Research methodology (Peffers et al., 2007) which contains practical elements of the Hevner et al. (2004) Design Science approach. Design Science is used to design and evaluate IT artifacts to solve a certain problems organization faces (Hevner et al., 2004). For this research we will choose the Design Science Research Methodology (DSRM) which incorporate 6 activities (Peffers et al., 2007) as listed below.

With the use of this DSRM methodology, the research question will be answered and supported by a completed thesis which incorporate the below 6 activities and the designed outcome artifact which is the framework. The DSRM approach will be based on an iterative process, whereby first the problems will be identified and motivated to get an in depth understanding of the current shortcomings for attack detection (solutions). After the problem phase, the objectives for the artifact will be defined which incorporates the found shortcoming solutions. Furthermore, the objectives will be a designed and developed actual framework artifact for the solution. The actual designed framework will be demonstrated and evaluated. In the evaluation process the designed artifact will be analyzed and improved to match the solution objectives as required. The outcome of the communication activity will be a thesis report to add new knowledge to this research area and to answer and support the research question.

In detail, the following 6 (DSRM) activities will be incorporated for the development of the framework.

Activity 1: Problem identification and motivation.
Activity 2: Define the objectives for a solution.
Activity 3: Design and development.
Activity 4: Demonstration.
Activity 5: Evaluation.
Activity 6: Communication.

4.1 Activity 1: Problem identification and motivation
Within the problem formulation stage, current research for attack detection methods, theories, concept and models will be analyzed to determine their weaknesses, challenges and open issues for the protection of a Critical Information Infrastructure. It includes activities such as determining the types of known attacks, detection methods and the techniques used. Various sources will be utilized, ranging from commercial to academic sources.

4.2 Activity 2: Define the objectives for a solution
The outcomes of the previous activity is used to collect and to determine the objectives for the artifact framework (solution) to detect attacks on the Critical Information Infrastructure. In the previous problem section some of the objectives have been highlighted, the framework should be adaptable to detect new attacks.
4.3 Activity 3: Design and development
The framework will be designed and developed based on the previous objectives. It will be developed and designed containing of key elements for gathering and preprocessing data obtained from multiple sources to perform analytical querying on datasets to fill in the gap of current intrusion detection shortcomings. The outcome of this design and development phases will an artifact (framework) based on Big Data Analytics and include functional designs containing of high level functions (Microsoft Word, Visio) to reflect the objectives. Furthermore, this framework will be made available to the general public to share knowledge, and to possible incorporate it into their own critical infrastructure protection setup. By using the future framework organizations could benefit to improve attack detection.

4.4 Activity 4: Demonstration
The framework will be demonstrated with in an experimental proof setup which will be based on the previous design and development requirements. The framework (simulation) will be based on a virtual environment containing of big data analytics technology, attacker and defender virtual machines where a light-weight evaluation (Peffers et al., 2007) walkthrough is performed, e.g. data will flow through the framework and it is being processed to show how attack detection is performed. Different attack demonstration use case (scenarios) will be considered to demonstrate the designed artifact.

4.5 Activity 5: Evaluation
The logical proof of the artifact of the framework will be analyzed and observed whether the objectives are achieved. The usability and the results will be compared to the activity 1 outcomes. Iterative improvements can be made to the artifact to achieve the objectives during this process. The outcomes of attack demonstration use cases (scenarios) will provide evidence whether these objectives are met.

4.6 Activity 6: Communication
The thesis outcomes will share the results of the performed research. New knowledge for this research area will be developed including the designed artifact, and it supports the answers for the research question.

This DSRM methodology would benefit the quality of the designed artifact due to the evaluation activity and its iterative process. Furthermore, this methodology is known for a design perspective (artifact) approach to solve certain problems an organization faces, whereby possible scope creeps are limited because of the ‘Problem identification and motivation’ and the ‘Objective’ activities.
5. LITERATURE REVIEW

This chapter contains the literature review analysis. At first the literature review method is explained, following a cycle processed approach to review the various literature themes to determine the research gaps for the framework artifact solution.

5.1 Literature review method

The literature review method is based upon the Baker (2000) framework for performing a literature review. The framework consist of 5 main phases, following a cycle processed approach where initially a review scope is determined, followed by conceptualizing the research topic, performing a literature search using various knowledge base sources, where the literature is reviewed, analyzed and synthesis to form the research agenda.

The review scope has been focused on previously performed research outcomes applicable to the conceptualized theme based approach generated from the research question and research area. Within the review analysis process for synthesizing the results, the main goal was to integrate the results based a neutral available representative perspective to overcome possible shortcomings in the review coverage and research gaps since several concepts or themes are exhaustive with extend of knowledge due to intensive long era of performed research by other researchers.

The review scope and the conceptualization was formed by the interest of big data analytics to possible extend current knowledge for improvement in intrusion detection solution shortcomings applicable to critical information infrastructures. From that perspective, the research question was formulated which was conceptualized by the following themes or concepts in the literature search process.

<table>
<thead>
<tr>
<th>Themes/concepts</th>
<th>Research question</th>
<th>Research topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Big data) analytics</td>
<td>How can big data analytics be utilized in a designed framework to detect attacks for a Critical Information Infrastructure?</td>
<td>Big data analytics attack detection for Critical Information Infrastructure Protection</td>
</tr>
<tr>
<td>Big data framework</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intrusion detection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attack types</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Critical information infrastructure</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the literature search process the themes were transformed into logical keyword queries where the following knowledge base sources and initial search criteria’s where used.

<table>
<thead>
<tr>
<th>Knowledge databases</th>
<th>Search criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lulea University database</td>
<td>Primary academic journals, secondary conference materials.</td>
</tr>
<tr>
<td>Sciedirect</td>
<td>Publication dates: 2010 - 2016</td>
</tr>
<tr>
<td>Web of science</td>
<td></td>
</tr>
<tr>
<td>Scopus</td>
<td></td>
</tr>
<tr>
<td>Google scholar</td>
<td></td>
</tr>
</tbody>
</table>

During the initial search outcomes and narrowing down the potential papers, the titles were collected and placed in a general overview where details such as abstract, year of publication, number of and usability was registered. Throughout the literature search and analysis process the paper references were analyzed (backward searches) to gain more knowledge and to determine the usability and quality. Furthermore, with the use of Google scholar forward searches were performed for new findings and the ‘similar article’ option was chosen.

During this process a conclusion was drawn that the concepts intrusion detection, attack types and
critical information infrastructure had good papers before publication dates 2010 and several researchers tried to improve previous findings in the last past years. For this process a hurdle needed to be taken due to large number of available papers, where in the end the number of cited sources formed the initial baseline, but those paper could use review papers or older books which makes it a long and difficult analysis process.

An outcome during the literature review are the used technology definitions in research. Good definitions in research for big data or for a critical infrastructure definitions are open issues. In the field of attack types taxonomies that could be generally applied to critical information infrastructures is another open research issue. Various taxonomies exist and researchers tried to close this gap, but a common attack type taxonomy framework which can be generally applied does not exist.

Furthermore, during the analyses and process for intrusion detection, it was clear from the start that this is an enormous research area and it dates back to the 1980’s. To gain more insights several review papers were analyzed to determine a common ground for the types of intrusion detection solutions and techniques, but this resulted only a common ground for the two main general detection technique anomaly and misuse techniques. A general merged taxonomy framework for intrusion detection techniques, methods, algorithms and classification is greatly demanded.

In the end a merge with forward and backward searches restricted by time only empirical based research journals was chosen for the detection theme. After synthesizing the results the following number of papers were used for the defined themes where mostly primary academic journals, some conference papers and industry papers were used.

<table>
<thead>
<tr>
<th>Themes/concepts</th>
<th>Nr of used papers</th>
<th>Nr of found available papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Big data) analytics</td>
<td>10</td>
<td>32,000 +</td>
</tr>
<tr>
<td>Big data framework</td>
<td>12</td>
<td>30,000 +</td>
</tr>
<tr>
<td>Intrusion detection</td>
<td>34</td>
<td>285,000 +</td>
</tr>
<tr>
<td>Attack types</td>
<td>13</td>
<td>8,000 +</td>
</tr>
<tr>
<td>Critical information infrastructure</td>
<td>4</td>
<td>3,000 +</td>
</tr>
</tbody>
</table>

Outcome in the following section shows that over the past decade researchers have found several solutions to overcome the shortcomings for the detection and prevention of attacks that could harm a critical infrastructure which are discussed in the following sections.

5.2 Critical information infrastructure

As early stated in the introduction section, critical information infrastructures can contain valuable information and can consist of many integrated system (van der Lei et.al, 2010) residing in many industrial sectors (Singh, Gupta and Ojha, 2014). These infrastructures form the basis for our modern society and according to Ten, Manimaran and Liu (2010) and therefore they form the backbone to our society. Even though references are made to supervisory control and data acquisition (SCADA) systems (ibid), critical information infrastructures not only relate to SCADA systems for power, water and other systems as stated by Ten, Manimaran and Liu (2010). According to the definition of the European Union (European Union, 2008 p.77) for critical infrastructure; ‘critical infrastructure’ means an asset, system or part thereof located in Member States which is essential for the maintenance of vital societal functions, health, safety, security, economic or social well-being of people, and the disruption or destruction of which would have a significant impact in a Member State as a result of the failure to maintain those functions’.

Therefore, critical information infrastructures can be datacenters, banks, public transport, telephone
networks and the Internet itself as well depending on definition of critical information and the level of society dependency as shown in the research performed by Singh, Gupta and Ojha (2014) in a country such as India.

5.3 Big data

Big data can be summarized as large and fast growing volumes of any type of structured, semi-structured or unstructured data coming from different sources that is too large to be processed by traditional technologies (Kshetri, 2014; Raghupathi and Raghupathi, 2014; Kaisler et al., 2013). According to Kaushal, Khan and Kumar (2015, p.123), Kaisler et al (2013,p.996) and Russom (2011, p.6) big data can be characterized, combined and summarized to the following five V’s:

- **Data volume.** The amount or size of data that gets quantified or is available in a big data environment for an organization. This includes structured, semi-structured or any unstructured types of data which can reside in different sources and formats.
- **Data velocity.** The speed of how fast data is begin generated and processed in a big data environment. Data can be processed in batches, streamed, in near and real time.
- **Data variety.** Data can contain in many different forms and types. These data types can represent text, imagery, video, audio and therefore can be structured like a relational database, unstructured like video and images or semi structured data like XML and JSON files.
- **Data complexity.** Data from various sources can contain of various data types which requires data management due to the interconnections and linkages between the data, and it can be therefore a complex process to manage it in order to gain value from it.
- **Data value.** Data used or processed in a big data environment has a certain value of usefulness to an organization. Logical decision will be required based on the data outputs.

5.4 Big data sources

Data can exist in many forms and be stored and processed in different ways. In order to clarify the possible big data sources the following data source types of big data can be defined (Kaisler et al., 2013; Chen, Chiang and Storey, 2012; Russom, 2011):

- **Human generated data.** Information generated by human activities, stored and digitalized ranging from devices such as mobile phones, laptops, computers and servers. Human sourced information can be categorized as semi-structured and unstructured data. Examples are social network activities, blogs and commenting, videos and pictures, internet searches, personal documents and e-mails.
- **Operational generated data.** Data generated by the organization on their systems for any operational activities to service their business goals and objects. Such data resides in relational database systems and are structured. Examples are customer records, medical records, banking transaction records, CRM, ERP systems and mainframe applications.
- **Machine generated data, Internet of Things.** Data generated by sensors and other machines to measure and record certain physical events. Examples are weather sensors, GPS systems, traffic sensors, system, security and application logs.

Attack data that gets generated comprises of human generated information and machine generated data where operational data is obtained during a successful attack of critical infrastructure.
5.5 Big data analytics

Big data analytics enables processing the big data sources with techniques such as text mining, machine learning for data classification, clustering and correlating data and visualizing the outcomes of the processed data (Kaisler et al., 2013; Chen, Chiang and Storey, 2012). Therefore big data analytics technology can be used for many purposes to gain knowledge from big data.

Big data analytics powerful features enables self-organizing networks to deliver the required resources to mobile phone users and prepares for future 5G networks (Imran and Zoha, 2014). Sadasivam et al (2016) used big data analytics to detect fraud by processing financial annual reports to achieve high detection accuracy on fraud detection and reduction in time. Other research has been performed by Srivastava, U. and Gopalkrishnan, S., (2015) to utilize big data analytics to process financial banking data for determining customers financial spending pattern and profiles customers and other uses.

Due to its large and fast data processing capabilities, big data analytics tools can query large data sets, which enables possible attack detection to critical information infrastructures with the use of a big data technical environment to process possible attack data to gain outcome knowledge.

This big data environment consists of an architectural framework with technical elements which collects the appropriate data from the different sources like operational and machine generated data (Zhao et al., 2014; Raghupathi and Raghupathi, 2014). Distributed data processing platform are used, to process the large gathered data which can make use of distributed database systems like NoSQL databases. Within the data processing platform, other technologies enable technical operations to filter, aggregate, indexing, transforming data to process the data or preprocess the data towards a certain data schema or forward data to a destination target like a relational database store or a HDFS (Raghupathi and Raghupathi, 2014). Within the data processing process, data can be associated as a certain type or identity, be clustered with other similar data types and or be classified to be identified. On top of the storing and data processing platform, an analytics engine will perform queries on the large processed data to derive the outcome knowledge to determine possible attacks to critical information infrastructures.

5.6 Big data analytics similar research frameworks

In the current research field of big data analytics, researcher have used and analyzed this technology in many fields to process large datasets to process outcome knowledge. In terms of developing a big data analytics framework artifact to detect attacks, Singh et al.(2014) developed a big data analytics framework to detect peer to peer botnet detection based on machine learning detection technique. This research emphasizes on botnet detection and not purely based on attack patterns or attack detection such as discussed in the earlier sections. In their setup they used known botnet attack capture files such as the Conficker worm from 2008 and Zeus trojan from 2007, and tested the detection rate based on the traffic classes ‘Malicious’ and ‘Non-malicious’. Even though high true positive rate was achieved based on these traffic samples, contributions for future research requires to overcome issues as packet drops during data processing, detection of dormant and low traffic botnets activities which are categorized as stealth communications. Possible other shortcomings for this research would be the use of the fixed data samples for detection testing, unknown or real attack scenarios with the use of dynamic mixed traffic. Likewise attacks can be in many forms as indicated in this literature review section, and be part of the botnet collective and the communication methods could silently blend in with non-malicious traffic.
Although, limited research has been performed on framework development for big data analytics for use of attack detection, currently the following frameworks are found in research that addresses the following big data analytic framework topics.

<table>
<thead>
<tr>
<th>Researchers</th>
<th>Framework Topic and usage</th>
</tr>
</thead>
</table>

Current research emphasizes that big data analytics and framework development increased in research popularity to possible address the big data challenge such as large and fast increasing datasets to develop useable outcome knowledge in many industries sectors.

**5.7 Attack types on critical information infrastructure**

Attacks can be very sophisticated according to the earlier reports (BSI, 2015; ICS-CERT, 2015; Trend Micro and Organization of American States, 2015). Complexity of the infrastructure also plays a key
role to the problem of attack detection because of the human factor involvements which dynamically changes the behavior of traffic flows. During an attack the attacker and defender considers possible actions which influences the behavior and the detection possibilities for modeling (Moayedi and Azgomi, 2012).

Nevertheless, based on the Kjaerland (2006) taxonomy for classify attacks, Miller and Rowe (2012) performed a survey on the past critical infrastructure incidents. In their research analysis for critical infrastructure incidents based on the timeframe from earlier 80's till 2012, the methods of attack operation showed high factors in root compromise, Trojans, user compromise and misuse of resources. Furthermore the impact showed a high outcome of disruptions and data disclosures.

In order to detect possible attacks against embedded control systems in power grids, Reeves et al. (2012) performed research regarding host based intrusion detection based on the Autoscopy system for power grid systems. In their approach, attacks related to the protective solution were data modification, program circumvention and process hijacking as the biggest attack types which supports the research performed by Coppolino and Romano (2014) by analyzing the software security vulnerabilities in power grids. In their vulnerability analysis of smart grids software, the tested power grids concluded vulnerabilities in communication, weak user password management, weak security measures for input validation at the application level and the support of legacy systems which increases the attack vector for inside attackers to compromise the system. Outcome of possible attacks included SQL injection, man in the middle attacks for data modification and eavesdropping and brute force password attacks.

Another form of attack which can sincerely disrupt a critical infrastructure is a distributed denial of service (DDOS) attacks to exhaust critical system resources. Genge and Siaterlis (2013) performed analysis of the DDOS attacks on multiprotocol label switching (MPLS) networks used by critical infrastructures. Outcome showed that DDOS attack have disruptive effects on the critical infrastructures evidently with use of powerful routers. Apart from creating large volume of data to overwhelm the critical infrastructure, Schuett, Butts and Dunlap (2014) enabled to exploit and modify legitimate programmable logic controllers (PLC) firmware and to install these successfully on PLC devices. This research shows that adapted firmware could be installed to trigger a time-based DOS on the operation system and a remotely triggered DOS with use of a remoted submitted external command. Outcomes are unusable PLC systems which interrupts the working of the critical infrastructure components.

Furthermore, wireless networks are possible being used for critical infrastructures monitoring due to for cost decision purposes (Buttyán et al., 2010). With the use of wireless technology wireless attack types are possible. Attack examples considered by Buttyán et al. (ibid) would be wireless frequency jamming to disrupt the wireless communication, eavesdropping wireless connection and replay and injecting of malformed data.

All these previous listed attack types are in line what other researchers performed analysis on the security solutions like intrusion detection systems to detect these attacks for impact minimization. In terms of attacks type detection, Cazorla, Alcaraz and Lopez (2015) grounded their critical infrastructures detection solution on the past Stuxnet worm attacks, Trojan (Duqu) to steal valuable information, the Nitro attacks to steal secrets from the Chemical industry with the use of a targeted email attack to infect computers with botnet capable features and the attack on decoyed water plant Honeypot in the U.S.

Other intrusion detection researched performed related to critical infrastructure, used Remote to Local (R2L) attack samples to test the effectiveness of their intrusion detection system solution (Masduki et al., 2015). These Remote to User attacks (R2L) type of attacks exploit remotely vulnerabilities over a
network to obtain higher privileges to compromised the system. Furthermore, many intrusion detection researchers use predefined attack dataset to detect the effectiveness of their intrusion detection solution (Narsingyani and Kale, 2015; Hoque et al 2012) with the help of the KDD CUP 99 dataset, which contains of the attack classifications Denial of Service (Dos), Remote to User attacks (R2L), User to Root Attack (U2R) and probing (KDD cup 1999 data).

Therefore to summarize the previous knowledge, attacks can be classified into several classifications and be differentiated in external and internal, each having a certain level of impact and detection challenges. Based on previously elaborated sources, the different attack types and scenarios for critical infrastructures can be concluded as:

<table>
<thead>
<tr>
<th>Attack classification</th>
<th>Source attack</th>
<th>Explanation</th>
<th>Past literature attack examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denial of Service (DOS)</td>
<td>Internal / External</td>
<td>Resource exhaustive attack, where the target is overwhelmed or flooded with data.</td>
<td>Misuse of resources. Distributed denial of service (DDOS) attacks. Wireless frequency jamming. Stuxnet worm.</td>
</tr>
<tr>
<td>Probing</td>
<td>Internal / External</td>
<td>Reconnaissance attack, whereby information and vulnerabilities are passively and actively gathered about the target</td>
<td>Trojans. Stuxnet worm Trojan (Duqu). Nitro attacks.</td>
</tr>
</tbody>
</table>

5.8 Attack detection

Intrusion detection has a long history in the era of computer and network security for detecting security attacks (Anderson, 1980). Throughout these years several detection techniques and models have been introduced and studied to keep up to pass with the increased attack vector, while providing possible intrusion detection solutions based on two types, network and host based. Host based IDS is when the detection utility has been deployed on a local machine to detect possible attacks toward local services and the application level, whereas network based intrusion detection systems detect attacks on network level by monitoring traffic generated by hosts. Within this intrusion analysis process, data
gets analyzed, logged and processed real-time by a certain detection method. Furthermore, intrusions can be detected passively or prevented actively during the analysis process depending on the used technology and operational mode (Scarfone and Mell, 2007). In the process of attacks detection techniques, intrusion detection systems can be supervised or unsupervised learned. Supervised learn is based on training data which contains of simulated attacks like Juanchaiyaphum et al. (2015) to learn the system attack classes that must be detected, unsupervised learning is where no knowledge based training is used or technically not possible to classify the potential attack classes or groups as an outcome (Om and Kundu, 2012). These detection classifications of attacks during the detection analysis outcomes are defined in research to the following outcome classes (Sampat and Sonawani, 2015; Prakash and Rajendra, 2014) shown in the following confusion matrix.

<table>
<thead>
<tr>
<th>Normal data</th>
<th>Attack data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>False negative (FN)</strong></td>
<td><strong>False positives (FP)</strong></td>
</tr>
<tr>
<td>Number or percentage of intrusive attack data which get classified as normal non attack data incorrectly.</td>
<td></td>
</tr>
<tr>
<td><strong>True negative (TN)</strong></td>
<td><strong>True positives (TP)</strong></td>
</tr>
<tr>
<td>Number or percentage of normal non attack data classified as non-normal attack data correctly.</td>
<td></td>
</tr>
</tbody>
</table>

Intrusion detection solution main goal is a high accurate detection for attacks by having a very low outcome of false positives and false negatives based on actual traffic non simulated traffic.

In research to reach the previous goals, a generalized distinction is made between several detection methods and techniques where hybrid solutions (Om and Kundu, 2012) are being used as well. The hybrid solutions use data mining techniques and data clustering approaches which are discussed in this section. To summarize main core detection methods and techniques are the summarized into the following detection process.

- **Misuse detection or knowledge-based detection** is based on pattern matching for detecting attacks. It relies on a knowledge database containing of unique classified patterns that reflect specific attacks (Chebrolu, Abraham and Thomas, p.289, 2005; Prakash and Rajendra, p 7184, 2014). Whenever the data matches the signature a true positive event will be generated and identified as an attack.

- **Anomaly or behavior-based detection** is based on traffic deviations or traffic profile differences (probability, statistics) for detecting attacks. Whenever a traffic profile is different or deviates, a true positive event is identified as an attack (ibid).

- **Machine learning for intrusion detection** is a technique where self-improvement is enforced by supervised or unsupervised the system over a certain period of time with the use of certain algorithms for future new types of attack detection (Mohamad et al.,2015; Wang et al., 2010). Machine learning methods for intrusion detection have increased in popularity to overcome the shortcomings of the previous detection methods.

Furthermore in research the following techniques, modeling, methods and data classifications are known for attack detection.

Data mining techniques are being used for intrusion detection solutions for attack model creation for better attack classification and detection (Duque and Omar, 2015). With the use of data mining
techniques, data can be analyzed and modeled to identify possible new attack patterns, interrelates types of data relations and structures to improve a knowledge detection database which benefits intrusion detection solutions when integrated. Associated rule mining algorithm (Devaraju and Ramakrishnan, 2015) is common example of a data mining techniques which is used to improve intrusion detection for outlier issues. Outliers in the data analysis process could indicate possible unknown or undetected previous attacks with current detection techniques.

Rule based intrusion detection is based on pattern matching misuse detection or rule based behavior-based detection. Rule based pattern detection also known as expert systems, relay on certain sets of patterns that are classified or identified (Prakash and Rajendra, p. 7184, 2014). Rule based behavior detection is based on a certain threshold condition, profile or logical conditions classifiers for triggering (Prakash and Rajendra, p. 7185, 2014; Farooqi et al., p. 914, 2013).

Clustering based network intrusion detection, uses clustering basis algorithms to (pre)process data into groups of similar data types (Wei et al., 2014). Clustering algorithms such a k-means (Om and Kundu, 2012) and Fuzzy c-means (Wang et al., 2010) will process the data to form these clusters where other functionalities and techniques classifies those groups to build a possible anomaly detection model. Clustering is based on unsupervised learning (Om and Kundu, 2012) or supervised learning based on training data (Juanchaiyaphum et al., 2015) and is used in hybrid intrusion detection solutions to improve the shortcomings in misuse and anomaly based solutions (Juanchaiyaphum et al., 2015; Om and Kundu, 2012).

Bayesian Network, in intrusion detection also simplified applied as Naïve Bayes (Amor, Benferhat and Elouedi, p. 421, 2004) is a classification type, or models information to detect possible attacks with the use of a (graphical representative) relationships between nodes (Kruegel et al., p. 15, 2003) that outcomes in a probability calculation. Bayesian Network is also used in Machine Learning intrusion detection solutions or in data mining techniques (Panda, M., & Patra, M. R, p. 258, 2007). Outcomes of research by Kruegel et al. (2003) based on MIT Lincoln Labs 1999 data set shows that still false positives were generated even after classifying known attacks for the Naïve bayes solution. Furthermore, by using Naïve Bayes outcomes shows that performance is faster than the decision trees approach (Amor, Benferhat and Elouedi, 2004).

Support vector machine (SVM), a machine learning algorithm is used in intrusion detection to classify the types or models information to detect possible attacks (Mulay, Devale and Garje, 2010; Mukkamala, Janoski and Sung, 2002). Data will be classified into two forms, which is normal traffic and attack traffic. Large volumes of data can be processed by intrusion detection solutions based on the use of the support vector machine algorithm. Furthermore research shows (Mukkamala, Janoski and Sung, 2002) that SVM’s and neural networks shows high accurate detection results for trained DARPA data and shorter retraining time, but classifications is only based on binary outcomes is a shortcoming for the differentiating between the different types of attacks.

Neural Networks, a machine learning algorithm also defined as an artificial neural network enables to classify and identify attacks based on adaptive learning similar to the human neural brain. Multiple connected neurons in layers process the attack data, where data is supervised or unsupervised processed through the system. Based on the calculation sum outcomes for each node, the weight of each node is self-adjusted to match the supervised expected outcomes (Al-Jarrah and Arafat, 2015; Branitskiy and Kotenko, 2015). Outcome by Al-Jarrah and Arafat (2015) shows that their neural networks intrusion detection trained setup detected attacks faster than other rule based intrusion detection systems.
Decision Trees, in intrusion detection is used to classify types or group similar classes of information which is also used in data mining techniques. Flows of information is transformed into tree structure containing of a root node and its leaf attributes in the outcome classification process (Kumar, Hanumanthappa and Kumar, 2012). Decision Trees have been successfully applied to machine learning SVM’s (Mulay, Devale and Garje, 2010) as well and to the intrusion detection research performed by Kumar, M., Hanumanthappa, M. and Kumar, T.S. (2012) to process large sets of data for real-time intrusion analysis based on the decision tree algorithm detection. Also random forest algorithm a division of decision trees used supervised has been used as data mining techniques effectively, to detect patterns and to build the rules for a misuse and behavior based intrusion detection system (Zhang, Zulkernine, and Haque, 2008).

Fuzzy Logic, also used by machine learning techniques is used in behavior-based detection for attack classification and detection. Data sets can be fuzzy-learned to generate fuzzy chained logical classifications rules for intrusive behavior (Shanmugavadivu and Nagarajan, 2011), data mining outcomes or preprocessed data outcomes are used to develop fuzzy logic classification rules for intrusion detection (Prakash and Rajendra, 2014), or clusters of data types are generated based on supervised learning whereby fuzzy logic can classify whether an attack pattern can be detected or identified (Selman, 2013).

Genetic based intrusion detection, based on its genetic algorithm properties, is where supervised trained data is preprocessed and generated to form a factious population data set. During the preprocessing phase the data is transformed towards genetic data properties structure containing of ‘chromosomes’. In the detection phase, a genetic algorithm will create genetic based rules which are tested on the trained data set to determine possible attacks (Narsingyani and Kale, 2015; Kharche and Patil, 2014; Hoque et al., 2012).

Immune system based network intrusion detection, is based on the similar working of the human immune system. Within the process of the antibodies creation, a negative selection process takes place for newly generated antibodies. This process ensures illumination of antibodies which possible attack own cells. With the use of this immune algorithm, patterns will be random generated from data and compared to each other. Similar outcome patterns will not become a detection profile during the detection phase (Kim and Bentley, 2001) to detect possible attacks.

5.9 Attack detection challenges

In the previous section several detection techniques have been summarized to determine the possible detection solutions for attack detection to minimize the impact possible attacks. Since various and granular solutions exist, the following outcomes have been analyzed segmented in two main time periods pre 2010 and from 2010 till the year 2015.

5.9.1 Attack detection challenges pre 2010

Below are the listed summarized empirical research performed from the years 2001 till 2008. Outcome shows attack detection research emphasizes trying to find hybrid solutions and detection classification.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Detection technique approach</th>
<th>Improvement solution or research goal</th>
<th>Research outcomes</th>
<th>Research future suggestions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Kim and Bentley, 2001)</td>
<td>Artificial immune system</td>
<td>Not listed</td>
<td>Infeasibility negative selection algorithm, scaling problem</td>
<td>Research the effectiveness negative selection algorithm</td>
</tr>
<tr>
<td>Remarks</td>
<td>(data sets) handling real network traffic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Remarks</strong></td>
<td>Creation of detectors for attack detection takes enormous CPU utilization and computing time.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Kruegel et al., 2003)</td>
<td>Bayesian event classification, Bayesian Networks</td>
<td>Reduce false positives for misuse and anomaly based intrusion detection solutions</td>
<td>Reduction of false positives. Half false positives compared threshold based naïve schemes</td>
<td>Not listed</td>
</tr>
<tr>
<td><strong>Remarks</strong></td>
<td>Training detection models required (2 weeks). Reconnaissance network scans and port sweeps not detected, access control policy violations not detected.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Amor, Benferhat and Elouedi, 2004)</td>
<td>Naïve Bayes and decision trees</td>
<td>Effectiveness of Naïve Bayes networks versus Decision Tree (Machine learning)</td>
<td>Decision Tree better results than Naïve bayes. Computational wise naïve bayes far less intensive than decision tree for learning and classifying.</td>
<td>Not listed</td>
</tr>
<tr>
<td><strong>Remarks</strong></td>
<td>Not 100% detection score for both. Decision trees requires more computing resources. Decision trees better detection results. Learning is required. 10% of the whole KDD’99 dataset was used. Simulated traffic.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Chebrolu, Abraham, and Thomas, 2005)</td>
<td>Bayesian networks, Classification and Regression Trees (CART) and combination</td>
<td>Intrusion detection improvement with hybrid approach with data mining</td>
<td>Normal traffic, Probe, DOS 100% accuracy. U2R and R2L with 84% and 99.47%</td>
<td>Better detection for user to root attacks</td>
</tr>
<tr>
<td><strong>Remarks</strong></td>
<td>Data mining is semiautomatic, requires ‘manual’ adjustments for new attack patterns. Simulated traffic. Lower detection of user to root attacks.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yang, Usynin, and Hines (2006)</td>
<td>Statistical probability ratio tests (Anomaly based)</td>
<td>Increase of attacks</td>
<td>Detection of anomalies. Insider attackers harder to detect</td>
<td>Further development of attack detection for insider attacks and create optimal intrusion detection system indicators.</td>
</tr>
<tr>
<td><strong>Remarks</strong></td>
<td>Only DOS was tested. Simulated traffic.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Panda and Patra, 2007)</td>
<td>Naïve bayes algorithm</td>
<td>Detection shortcoming new intrusions. Human involvement for classification. Solve with the</td>
<td>95% detection rate with 5% false positive. Neural network based approach detection is higher, less time consuming. Creating the model is faster</td>
<td>Reduction of false positives by using Bayesian network for classification.</td>
</tr>
</tbody>
</table>
data mining algorithms
 naïve bayes

but generates more false positives.

**Remarks**
1.89 seconds to build detection model with simulated data. 10% of the KDD’99 dataset used. Preprocessing the data is required. Naïve Bayesian for classification is restricted by two classes, not detailed (multiple) classes for modeling as with Bayesian network. Simulated traffic.

(Zhang, Zulkernine and Haque, 2008)

| Random-forests | Detection shortcoming of rule based intrusion detection systems that miss out new intrusions. Time and known attack recognition for detection rules creation. | Improved and higher detection than other unsupervised anomaly detection solutions. Detection decreases upon increasement of attack data. Outlier detection decreases when more attack data is used or minor differences in attack data | Use of clustering algorithm to overcome the shortcoming of the research outcomes. |
| Remarks | Increase of volume of normal or attack data has impact on the detection performance. Detection issues minor differences in data. Requires training. Simulated traffic. |

5.9.2 Attack detection challenges from 2010

Below are the listed summarized empirical research performed from the years 2010 till 2015. Outcome shows attack detection research emphasizes more on machine learning, data mining and clustering including hybrid solutions containing of misuse and anomaly based intrusion detection.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Detection technique approach</th>
<th>Improvement solution or research goal</th>
<th>Research outcomes</th>
<th>Research future suggestions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Mulay, Devale and Garje, 2010)</td>
<td>Support vector machine and decision tree</td>
<td>Decreasing training and testing time. Improve efficiency, solving issues with classification.</td>
<td>Better outcomes by merging support vector machine and decision tree</td>
<td>Finalizing the results.</td>
</tr>
<tr>
<td>Remarks</td>
<td>Speed issues with SVM for large datasets due to computational requirements. Simulated traffic.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Wang et al., 2010)

<p>| Fuzzy clustering and Artificial Neural Networks (FC-ANN) | Improve detection for low-frequent attacks. Improve false positive rate, detection stability | Fuzzy clustering and Artificial Neural Networks has on average more precision compared to BPNN, decision tree, the naïve Bayes. Better low frequent attacks detection | Number of clusters for fuzzy classification is an open issue (Has impact on probes, R2L and U2R attacks). Other data mining techniques as SVM, evolutionary computing, outlier detection is suggested. |
| Remarks | Speed issues with SVM for large datasets due to computational requirements. Simulated traffic. |</p>
<table>
<thead>
<tr>
<th>Remarks</th>
<th>ANN requires learning to generate models. Training time of FC-ANN is huge (2125 seconds) compared to decision tree (2,68 seconds) and Naïve Bayes (1,93 seconds). Detecting Probe attacks performance is weaker. R2L and U2L is detected better with FC-ANN. High computational requirements is demanded. Simulated traffic.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Shanmugavadivu and Nagarajan, 2011)</td>
<td>Fuzzy logic</td>
</tr>
<tr>
<td>Remarks</td>
<td>10 % of the dataset was used for training and testing. Without data mining, fuzzy rules manually creation is an extensive workload with large datasets. Simulated traffic.</td>
</tr>
<tr>
<td>(Hoque et al., 2012)</td>
<td>Genetic algorithm</td>
</tr>
<tr>
<td>Remarks</td>
<td>DOS detection was high. Simulated traffic.</td>
</tr>
<tr>
<td>(Kumar, Hanumanthappa and Kumar, 2012)</td>
<td>Decision tree algorithm (anomaly and misuse detection)</td>
</tr>
<tr>
<td>Remarks</td>
<td>Decision trees requires pre-classified dataset for learning and categorizing behavior changes and patterns. Required to learn the system for classifying attacks (rule creation). Performance in trees building can be increased by using boosting, but fails if training data contains noise, e.g. high traffic loads. Simulated traffic.</td>
</tr>
<tr>
<td>(Om and Kundu, 2012)</td>
<td>Hybrid system anomaly intrusion (k-Means clustering, K-nearest neighbor, naive Bayes)</td>
</tr>
<tr>
<td>Remarks</td>
<td>Requires training even though hybrid system of misuse and anomaly based detection. Detection not 100 %. DOS traffic contains of 71% of dataset, second lowest in results. Data misclassification in anomaly based methods known issue. Simulated traffic.</td>
</tr>
<tr>
<td>(Selman, 2013)</td>
<td>Fuzzy Logic.</td>
</tr>
<tr>
<td>Remarks</td>
<td>Requires to be trained. The number of clusters and data overlapping has impact on the detection rate. Simulated traffic.</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>(Kharche and Patil, 2014)</td>
<td>Genetic algorithm, data mining method of fuzzy logic (class association rule mining)</td>
</tr>
<tr>
<td>Remarks</td>
<td>Crisp data mining better than fuzzy data mining. False positive and false negative rate was lower as well (contradicts researcher outcomes). Sharp boundary (data overlaps) problem exist with crisp data mining methods, normal traffic can match intrusion traffic because of minor differences. Better detection for anomaly detection with use of large mixes of normal rules instead of specific rules. Simulated traffic.</td>
</tr>
<tr>
<td>(Prakash and Rajendra, 2014)</td>
<td>Genetic-Fuzzy Classification</td>
</tr>
<tr>
<td>Remarks</td>
<td>Requires training. Lower detection for R2L and USR type of attacks. 10% of test set was used. Randomly attacks were chosen from the set. Simulated traffic.</td>
</tr>
<tr>
<td>(Wei et al., 2014)</td>
<td>Clustering analysis algorithm k-means</td>
</tr>
<tr>
<td>Remarks</td>
<td>10% test set used. Mixture of traffic shows lower detection rate.</td>
</tr>
<tr>
<td>(Al-Jarrah and Arafat, 2015)</td>
<td>Neural Network classification</td>
</tr>
<tr>
<td>Remarks</td>
<td>Test was only based on probes and reconnaissance attacks, not R2L and U2R. System was optimized for these attacks, number of sessions unknown. Simulated traffic.</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>(Branitskiy and Kotenko, 2015)</td>
<td>Neural, Immune and Neuro-Fuzzy Classifiers</td>
</tr>
<tr>
<td>Remarks</td>
<td>Attack recognition improves over time (machine learning) but requires time and training. Shows relationship with connections increase versus detection rate decrease. Attacks still get bypassed. Neuro-Fuzzy Classifiers long learning, due to complexity of calculations (performance). Neural network best rate of pattern recognition. Immune detectors can modify their structure in response, detection increased over time. Simulated traffic.</td>
</tr>
<tr>
<td>(Devaraju and Ramakrishnan, 2015).</td>
<td>Data Mining Algorithms</td>
</tr>
<tr>
<td>Remarks</td>
<td>10% of dataset was used. No 100% detection score. False positives were high for IP sweeps (reconnaissance) and brute force password attacks. Simulated traffic.</td>
</tr>
<tr>
<td>(Duque and Omar, 2015)</td>
<td>Data Mining Algorithms (k-means)</td>
</tr>
<tr>
<td>Remarks</td>
<td>Issues with obtaining the correct number of clusters for optimal detection and false positives in a real network environment. Simulated traffic.</td>
</tr>
<tr>
<td>(Mohamad et al., 2015)</td>
<td>Hybrid machine learning, K-means clustering and support vector machine classification</td>
</tr>
<tr>
<td>Remarks</td>
<td>Dynamic data requires preprocessing data (normalization). Noisy (mixed) data has impact on the learning algorithm. Simulated traffic.</td>
</tr>
<tr>
<td>(Juanchaiyaphum et al., 2015)</td>
<td>Data Mining Techniques, K-Means clustering. Decision tree (anomaly and misuse detection)</td>
</tr>
<tr>
<td>Remarks</td>
<td>Complex attacks still an issue. Training sets is of high quality (filtered, optimized). Computation time and resources increases for such solutions. Training phase is required for model creation. Preprocessing required for anomaly and misuse detection module. Simulated traffic.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>(Narsingyani and Kale, 2015)</td>
<td>Only DOS attack was tested. Increase of the number of rules improves true positives but increases false positives a lot. Number of rules has impact on resource utilization. Simulated traffic.</td>
</tr>
<tr>
<td>Remarks</td>
<td>Requires normalization of the training set. Simulated traffic.</td>
</tr>
</tbody>
</table>
5.10 Research gaps
Currently no big data analytics framework exists that tries to provide a detection solution for the current attacks on critical information infrastructures. Furthermore, current intrusion detection solutions based on anomaly, misuse or hybrid approaches still faces false positives (misclassifications) issues and false negatives due to attack detection shortcomings based on simulated attack traffic that rely a single detection source, where other issues such as utilization of resources, time, workload for ruleset development needs to be overcome as well.
6. DESIGN AND DEVELOPMENT

This chapter will set the required functional requirements for the solution. Based on the functional requirements for the solution, the framework artifact will be designed and developed accordingly.

6.1 Functional requirements

The framework development must incorporate and support the functional and fundamental requirements to overcome the shortcomings of the determined gaps in the previous research section. The requirements for the framework development are defined as the following.

- R1 – Provide attack detection based on open source big data analytics techniques and elements
- R2 – Provide a solution to improve false positives and false negatives for attack detection

For this research, attack detection is defined as when thresholds for detection rules are triggered or whenever there is a clear distinguish in data offsets such as peaks which results in outliers and classification differences.

In future research the following requirements which were found during the literature review process, can be considered. These are excluded for this framework design.

- Incorporate and process live dynamic traffic and non-simulated attack traffic for performance detection measurements, and emphasizing on adaptive detection techniques for detection and traffic patterns offsets
- Resilience against large amount of attack training dataset, normal traffic, noisy and mixture (overlapping) traffic to preserve attack detection functionalities
- Use of live dynamic traffic
- Limiting high utilization of resources (system, computational, time, human, training) without data loss during processing.
- Include multiple detection and attacked data sources for the detection of slow, complex, coordinated attacks and unauthorized technical compliance policy changes
- Limit manual detection rule creation and verification, define automated detection profiles preset creation for large datasets, Limit training, testing time and resources for the detection functionality

For this research a framework will be developed in accordance with the definition of Cunningham et al (2002, p.169), where the framework comprises of a set of technical and functional building blocks, layers and standardized elements which interrelate with each other serve the previous listed scope in order for the framework to be reused, extended and customized for future research. Moreover, the framework will be designed on an abstract design where specific and detailed functionalities and descriptions are excluded from the artifact design scope.

6.2 Functional design methodology

For this design research in order to meet the research goals, apart from using the Design Science Research Methodology (DSRM) which incorporate 6 activities (Peffers et al., 2007), the design and development until its evaluation phases will make use of the kernel theory and a structured design evaluation process.

The kernel theory is used for the construction of the initial functional design from a scientific perspective (Hevner et al., 2004, p. 96) where a structured design (evaluation) search process iteratively improves the artifact during its construction phase (Hevner et al., 2004, p. 88). During the
evaluation process the design development follows the main goals from Venable, Pries-Heje and Baskerville (2012, p.427), which are rigorous, efficiency and ethics. Throughout the design, diagrams be used to demonstrate the process flows for the design outcomes.

6.3 Kernel theory based framework

From a kernel theory point of view, enabling attack detection on a large scaled critical infrastructure, the requirements for the framework must generally comply to processing various complex data sources and types which can produce large and fast volumes data within a short period of time, which requires to be processed to gain analytics outcomes.

These big data requirements must be fulfilled additional to the previously listed requirements outcomes to meet the functional requirements for the research gap. Therefore, for any big data framework setup based on the knowledge gained from similar the big data analytics research frameworks, the framework for attack detection must incorporate and align with the current available open source big data technology, which includes the following core functions.

- Data gathering and receiving module. Required for collecting the human generated data, operational generated data and machine generated data
- Data processing module. Required for filtering, aggregating, normalizing and moving the collected data.
- Data storage module. Required for storing data that is receive by the log processing module, and to be possibly used for storage after being processed by the analytics engine
- Analytics engine module. Processes data to gain the outcome knowledge, where the data is processed in batches, near real-time or real-time depending on the requirements.

Consequently for the kernel theory based framework approach, the following core functions for the framework can be represented and summarized by the following diagram.
6.4 Framework overview
For this design the following framework overview has been designed, which is discussed in this section to find a solution to the found shortcomings.

<table>
<thead>
<tr>
<th>Analytics engine</th>
<th>Realtime</th>
<th>Near realtime</th>
<th>Batched</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine learning</td>
<td>Processing</td>
<td>Data - Log storage</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data - Log gathering / receiving / filtering</td>
<td></td>
</tr>
</tbody>
</table>

6.4.1 Data gathering, receiving and filtering module
In order to provide attack detection, the framework depends on a framework element that receives, gathers data and processes from distinctive sources such as human generated, operational and machine generated data. From the perspective of an attacker, whether the attacker resides within the critical infrastructure or outside, these activities must be logged. Critical infrastructure resides primary on IP-flow network traffic which requires to be collected and processed within the framework. Depending on these information flows, these must be processed in order to detect attacks. Referencing toward the critical infrastructure architectural examples in the attack types on critical information infrastructure sections, the minimum requirements for obtaining the required network data sources would be endpoints and core components within the critical infrastructure that processes main information flows generated by human activities for their operational tasks and reflects the machine behavior in terms of data outputs depending on the audit and log configurations.

Therefore from a network perspective, in general these traffic flows passes through core network elements such as firewalls, routers and proxies, switches and other components and this activity must be collected via various ways depending on their technical capability. Furthermore is it vital to collect the (audit) log activities from authentication components within the critical infrastructure to determine whether complex or slow attacks have been successful or failed and whether users had the authorization to performed these activities. Likewise, patterns of normal and abnormal behaviors of network flows must be learned and determined to minimize false positives. Therefore these data sources must be collected and combined for correlation purposes to conclude in the analytic outcome phases whether an attack was successful in the end. Within this design approach, in order to form a minimum baseline of components to detect attacks, the following can been set as an example baseline for the data sources which can improve the detection process within this framework.

- Core and gateway routers, every traffic flow will be processed via routers within a network and can restrict traffic flows
- Firewalls, similar to routers for processing traffic flows. Possible restrictions are set to filter out malicious activities and attacks at a more granular level for content inspection
Proxies. Processes application level user and machine activities such as web browsing activities

DNS servers, processes name serving requests for domain name to ip address translations and vice versa. Internal infected machines generate valuable DNS log entries for attack detection process

Authentication and authorization components. Processes and restricts authentication requests for access privileges. Authentication allows and denies are of interests during the analytic process.

As the open source requirement applies to this design, data audit logs gathering or receiving can be achieved depending on the technical capabilities of the component and the requirement earlier set depending on the type of attacks detection that are in scope. As the type of attack varies, a large volumes DOS attacks, or complex attacks that inquires data over a longer period of time to be detected. In such situations it is important that the data source log requirement of integrity and availability must be met.

Furthermore during attacks, audit data generation by the component can overwhelm the data log receiving component or impacts the inability to retrieve the logs at the source or in worst case log generation is automatically switched off or manual halted for the component upon a certain performance threshold. This implicate that the type of attack detection, influences the requirement for this part of the design development as its impact the data requirement in its analytic processing phase as well.

To form the minimum baseline requirements in data gathering and receiving framework element, the following functional criteria’s must be considered in the design of this component and are related to durability and a certain fault tolerance level to meet the defined requirements.

- Data access and generation functionality. The required data source components must have built in functionalities to generate audit-data-logs based on the performed activities. These audit logs can be limited or restricted in its generation. The design must overcome this issue by introducing a method to receive or pull the data from various sources. Furthermore, network sensors with capture extraction functionalities can be used to capture each packet to possible try to fill in the gap for a the detection use case whenever there is a restriction in obtaining the data.

- Data integrity and availability. High volumes of data during attacks still must be captured setting a high integrity dependency on the transport. Long slow and complex attacks sets a high dependency on its data availability (retention). Availability and integrity of the data meets a lot of the requirements and makes use of the log storage component apart from the scalability requirement in a dynamic environment. Certain attacks can be detected based on a minimum set of information with the available data set. Other attack scenarios require the full sets of packets for inspection for attack detection and requires to be stored or reused for future analysis in the log storage component within the framework, making log delivery and receiving a critical element.

- Data filtering. Data must contain the events of interest for processing by the analytic engine, others requires to be filtered out not to overwhelm the analytic engine or data storage component with unusable log data during processing or whenever high volume peaks occur.

- Data transforming. Data sources have a large variety of structures based on various data models or schemas which possible requires to be transformed to a certain data model standard
to be processed or data requires to be aggregated during processing. Data variety and data complexity requires to be overcome during log processing.

- Data velocity. The requirements must meet the analytical processing requirements for batched, near real-time and real-time attack detection to specific attacks. Certain attacks require real-time detection, others can be better detected via batch or near real-time processing requirements which impacts log quality delivery and receiving functionality at any data velocity, receiving or forwarding rate.

To combine the data sources and the previous listed requirements, the following below diagram is represented as an example for a critical infrastructure setup for this framework. The blue lines are the network connections for the normal network activities and the red lines are the data gathering – receiving connections for the attack detection framework for the source data collection, technically these can be a dedicated logical or physical network interface.

This part of framework will be primary based on existing open source technology which are combined to fulfill the requirements by using technology such as Logstash, Apache Flume and Apache Kafka. Besides this existing technology, the underlying technology for the data gathering and receiving can be a mixture of known data extracting and transport protocols such as syslog, dumpcap and tcpdump for packet capturing which can be outputed towards local log files, which then can be forwarded towards Apache Flume and Logstash for data filtering and transformation. Whether or not to use certain technology for this framework element depends on the support for the specific data source in the data normalization process, the data reliability requirement and analytics processing requirement. For example, Apache Kafka includes connectors to import an entire database as a source, but can’t normalize or filter out certain data during processing unless the data is first forwarded to Apache Flume for filtering and transforming, which has the capability to forward the processed data back towards Apache Kafka. Without any filtering or transforming the data sources, the framework will be overloaded with unnecessary data and impact the processing power and attack detection process. Apache Kafka is a very robust technology, provides high data durability for slow and complex attacks and offers fault-tolerance even at very high volume data peaks during large volumes dos attacks, and is
very fast and scalable due to seamless cluster support which meets the requirements as an input source for the analytics engines for the attack detection. Apache Kafka has default support for Apache Flume and Logstash in the data transport connectivity phase.

Apache Flume on the other hand, integrates very well with Apache Kafka as an input source and can import data from the data log storage framework component, has direct support for receiving syslog from many data sources for the normalization process, and can aggregate data, filter (regex extractor) and forward the output directly to Apache Storm and Spark or towards other big data log storage components. Apache Flume offers less durability for data loss because of its forwarding data mechanisms, but Flume is still chosen for the direct integration possibilities for the data source and forward capabilities towards Apache Kafka. Apache Flume does have redundancy and failover load balancing techniques by its selves but relies on knowing how much data requires to be handled. Furthermore, Apache Flume will possibly be used as a second filter and normalization mechanism to provide further clean data outcome towards Apache Kafka and the big data log storage framework component. Furthermore, Apache Flume can be used to merge similar data types like multiple webserver logs to a single data output which can be a separate feed in Apache Storm to apply logic.

Logstash has increased in popularity recent years because it is one of the core components of Elastic, formerly known as Elasticsearch, where Logstash emphasizes strongly on supporting various input data sources by default which enables less development time in the data transferring process modelling. This enables better integration for processing packet capture data. Logstash has many built in data filtering and transforming (normalization) capabilities including data aggregation or functions to anonymize certain data sections with data and can output directly towards Apache Kafka and Apache Flume. Logstash excese Apache Flume in many cases, but Logstash does not (currently) directly integrate with Apache Storm or Spark as an output. Therefore Logstash will possible be used for the data aggregation and normalization process to filter and transform the data from the data sources towards the appropriate data models as a source input to Apache Flume and Kafka.

To conclude, the design decisions for this framework component can be summarized as:

<table>
<thead>
<tr>
<th>Involved technology</th>
<th>Outputs to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data access and generation functionality</td>
<td>Primary Apache Flume, secondary Logstash</td>
</tr>
<tr>
<td>Data integrity and availability</td>
<td>Apache Kafka, Apache Flume, secondary Logstash</td>
</tr>
<tr>
<td>Data filtering</td>
<td>Primary Apache Flume, secondary Logstash</td>
</tr>
<tr>
<td>Data transforming</td>
<td>Primary Apache Flume, secondary Logstash</td>
</tr>
<tr>
<td>Data velocity</td>
<td>Apache Kafka, Apache Flume, secondary Logstash</td>
</tr>
</tbody>
</table>

### 6.4.2 Analytics module

Decisions for choosing the design issues in the analytics part of the framework is initially based on the following.

- Data sources can detect a certain attack based on its detection functionality (capability) and the attack event activity details is logged
- Data sources don’t have attack detection techniques and log outputs, or are misconfigured for suitable attack detection and logging

In all cases, the data is collected and logic is applied in its analytical phase to improve attack detection for critical infrastructures to fill in the functional requirements gap. Attacks can vary in types as concluded in the types of attack section for critical infrastructure. To summarize, these attacks can contain of high data volumes, be slow or very complex which has impact on the detection requirements in terms of data processing, required data sources and volumes of data usage for detection. Hence, attack types have impact on the detection method for this framework, therefore this design decision is based upon a hybrid model for attack detection based on two core analytics component, Apache Storm and Apache Spark for more detailed attack detection. Both technologies are open source and can be designed to provide attack detection for the found shortcomings. Therefore, based on the hybrid chosen analytics technology, attacks will be either identified by the event outcomes generated and detected by the used log sources, or detected by the analytics process itself because of a creation detection ruleset on top of the processed event stream. Within this setup, Apache Storm can trigger on event outcomes and with use of a detection ruleset to improve detection. Secondly, Apache Spark will possible process the outcomes and apply more detailed analysis with the use of additional machine learning modules for better detection. The outcome of Apache Spark could be used to improve the Apache Storm ruleset for detection, and improve attack detection in general like a feedback loop for continuous improvements.

Moreover, these decisions for using the hybrid approach is based on the following attack classifications chosen use cases and explanations, which are explained in greater detail in this chapter:

<table>
<thead>
<tr>
<th>Attack type use case</th>
<th>Example</th>
<th>Impact</th>
<th>Requires technology, design decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denial of Service (DOS)</td>
<td>Resource exhaustive attack, where the target is overwhelmed or flooded with data</td>
<td>Requires to process a lot of data within a short time frame, where short time attack detection is desirable for protection. Correlation possibly required for attack source distinction.</td>
<td>Apache Storm, provides real-time data processing at high velocity with enormous datasets</td>
</tr>
<tr>
<td>Remote to User attacks (R2L)</td>
<td>Remote attempts to tamper, inject, replaying, spoof for privileged escalation attempts to gain control of the target or misuse its functions.</td>
<td>Requires several data sources, possible with correlation and can be complex and the attack can be slowly executed over a longer time frame. More detailed analysis, data and time is required.</td>
<td>Apache Spark with support of machine learning modules of Spark for batched and near real-time data processing for more detailed attack analysis.</td>
</tr>
<tr>
<td>User to Root Attack (U2R)</td>
<td>Local attempts to tamper, inject, replaying, spoof for higher privileged escalation attempts to gain control of the target or misuse its</td>
<td>Similar to R2L attacks</td>
<td>Similar to R2L attacks</td>
</tr>
</tbody>
</table>
The other component which is used in this part of the design is Apache Spark. Apache Spark is a technology that processes data near real-time based on a batched approach. Like Apache Storm, Apache Spark is fully compatible with other framework components and includes all features to prevent any data loss during processing. Due to its compatibility, data sources can be processed directly from Apache Kafka and Flume and can make use of the data storage framework elements HDFS and the NoSQL database Hbase. In our setup the data processing framework will forward all the processed data towards the HDFS for storage. Apache Spark will process this data from the HDFS, and optionally will use the data processed by Apache Storm via the HBase database. Furthermore, in our setup the Apache Spark machine learning algorithm module will be used to better detect attacks for anomaly detection. These outcomes will be stored on the HDFS and or the HBase database.

In our setup for the attack detection use cases for Apache Spark, we primary choose the unsupervised Machine Learning techniques for attack improvement detection. These machine learning techniques will process various data sources based on a controlled experiment, where attack scenarios in the form of use cases will be developed. In our solution, the defined machine learning techniques will be used to triangulate and combine a knowledge stack outcome for attack detection. Datasets will be processed by each machine learning technique and compared like knowledge aggregation. With this type of approach, outliers can be detected and outcomes of classifications (differences) can be shown.

<table>
<thead>
<tr>
<th>Machine Learning techniques</th>
<th>Design decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term frequency-inverse document frequency</td>
<td>To analyze the dataset which attacker or data</td>
</tr>
</tbody>
</table>
Floris Stouten, flost-3

<table>
<thead>
<tr>
<th>Method</th>
<th>Technique</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF-IDF – Text mining</td>
<td>within a dataset appears to be frequent. During a DOS attack an attacker could create a high data frequency. Low frequency could indicate a slow complex attack to blend in with normal traffic.</td>
<td></td>
</tr>
<tr>
<td>Word2Vec – Text mining</td>
<td>To determine specific patterns for normal behavior patterns and anomaly patterns.</td>
<td></td>
</tr>
<tr>
<td>FP-growth – Pattern mining</td>
<td>FP-growth enables to identifies the frequent items within a dataset, to effectively show the number and type of occurrences. Outcomes can be used to decrease false positives or to point to outliers.</td>
<td></td>
</tr>
<tr>
<td>K-Means – Clustering</td>
<td>Clustering data based on similarity. Outcomes results in possible outcomes in normal and anomaly based data. In our setup we will use the streaming version.</td>
<td></td>
</tr>
</tbody>
</table>

More details of this approach is explained in the demonstration chapter.

6.4.3 Data log storage module

Since the framework requires to perform analytical processing of the data sources, the outcome knowledge requires to be stored for analytical reuse or used as an input factor for the analytics engines data processing process. In detailed, the data log storage within this framework will be mainly used as a three way function. Primary the outcome of the first stage Apache Storm analytics will be stored as outcome knowledge. Secondly the framework module ‘Data gathering, receiving and processing’, sends the data as an additional data feed as raw or normalized data towards the data log storage module for data processing. The Apache Spark analytics engine will process this data for more detailed analytics tasks for attack detection and its improvement. Thirdly, the outcome knowledge of the Apache Spark analytics engine requires to be stored for future use, similar like the Apache Storm analytics outcomes.

Considering the previous functions, the following design requirements must be met in the design. The data log storage module must be fully integratable with the analytics and the log processing module. The outcome knowledge of both analytics engines must be compatible and stored on the data log storage framework element.

In our design we choose between a direct raw feed from the ‘Data gathering, receiving and processing’ module to potentially process the data for future use. Therefore in our setup the known Hadoop Distributed File System (HDFS) setup is chosen for storing large raw datasets and normalized data, where the HDFS can be used additionally as an input data source for processed data (outcome knowledge) during the analytics process for combining data in the correlation process for attack detection.

The HDFS technology is completely compatible with other framework used technologies Apache Storm, Apache Spark and Apache Kafka. Within the design, the HDFS is not used by the Apache Storm as an input functionality, while it possible impacts the processing time (latency) in the real-time analytics process, but this will be a considered option for Apache Spark, since its technology is based on more batched approach for processing (near real-time) data.

Another decision for using the HDFS option in this design is because its known robust technology to store huge amounts of data spread over various systems, as it can makes use of a cluster based setup. HDFS enables a high availability cluster technique with the use of the Quorum Journal Manager to support the requirements for zero data loss. This cluster based setup enables the functionality to simply add new members to the cluster to meet the data storage requirements, besides offering the capability
for still being a very fast data storage and data usage for data input to meet the processing power within the framework. Furthermore, the HDFS is chosen because the most of the data can be spanned over multiple files and can be better managed when a directory structure is used instead of object stores. Visibility of (processed) data sources is very important as it will impact the logic for attack detection during the analytic building phase.

To meet the requirements for this design, and for the real-time analytic engine to provide future better attack detection and reduce false positives, a NoSQL database is also required within the design. With the use of a NoSQL database, processed data can be stored and retrieved at a very fast speed with low latency for the attack detection.

Therefore in this design, the outcome knowledge will be stored in a NoSQL database from the Storm analytic technology. The setup of the NoSQL database structure will be based on specific outcome knowledge tables, where data within the tables will be queried and joined into additional tables for possible future correlation purposes in attack use cases.

In our setup the Hbase NoSQL (wide column store) database is chosen based on our previous requirements. Hbase is compatible with the used analytics engines as an input and output source and is very fast with low latency properties. Hbase supports HDFS and can be distributed and scalable over various systems. One additional property is capabilities of being immediate consistent during data processing to meet the requirements and supports server-side scripting like stored procedures. With the use of stored procedures, additional triggers can be created to clone data real-time from tables for other uses. The NoSQL Cassandra database is excluded from the design due to the lacking of HDFS support, not being immediate consistent and supporting server-side scripting, even though is it faster for data processing with lower latency according to several sources and has the ability to query data based on a SQL structure.
7. DEMONSTRATION

This chapter will demonstrate the designed framework based on the earlier defined modules to incorporate the defined requirements based on a light-weight evaluation (Peffers et al., 2007) walkthrough approach. Data (logs) will be captured and processed through the whole framework for attack detection and the outcomes are stored in the data log storage module, to demonstrate outsider and insider attacks. Process diagrams will be used to demonstrate each framework module.

7.1 Demonstration technical setup

Our proof of concept has been developed on a virtual environment based on VMware technology. The proof of concept host machines created in a this setup, all hosts are built on the open source Ubuntu 16.04 LTS 64-bit operating systems each assigned specific amount of disk space and memory. All systems, except for the open source pfSense firewall and the webserver used as an attack target located in a demilitarized zone, are assigned 16GB of memory with a minimum of 64GB disk space.

As a summery the following components software versions are used in our setup.

- Apache Storm 0.10.1
- Apache Spark 1.6.1
- pfSense 2.3-Release (snort ids enabled)
- Apache Kafka 0.10.0.0
- Apache Flume 1.6.0
- Hadoop 2.7.2 (HDFS)
- Logstash 2.3.2
- Hbase 1.1.2

7.1.1 ‘Data gathering, receiving and filtering’ module

The walkthrough process starts with the ‘Data gathering, receiving and filtering’ module. Data is captured with the packet capture host or received from other component like the pfSense firewall.

This module has two main inputs and two outputs which is shown is the below process diagram where all the used technology is merged.

In our design for the packet capture host we chose to use the tcpdump utility configure to output directly towards a file without the buffering option to fulfill the real-time requirement as much as possible.
In tcpdump default mode settings, captured data is buffered and batched which is not ideal for (near) real-time data processing. Furthermore tcpdump is configured to output to full timestamp option and to assign each captured packet log line numbered to verify possible incompleteness (incremental) during the data processing phase. The packet capturing file is monitored by logrotate utility with a specific assigned policy to control the log file output. In our lab setup we used a ‘minimum’ logrotate policy which has a size rotate of 100 Mb, 5 file rotations with a maximum of 100 days for archiving purposes. Whenever the size or condition is met, the log file is preserved as much as possible.

To push the logs towards the ‘receiving and filtering’ component the rsyslog utility is used. The rsyslog option was set to use the TCP option to prevent loss of logs during transport which has retransmission capabilities for data transport loss prevention. To identify the packet capture host as an unique log source host, a unique tag ‘packetcapture’ is added to each log line to keep track of data throughout this framework.

In our design, the packet capture hosts syslog outputs towards Apache Flume. Apache Flume is configured in our setup to listen on two specific ports (UDP and TCP based) for incoming data, each having its own channel for processing and temporary buffering the received logs. UDP is used for incoming pfSense firewall syslog traffic. In our case, the stateless UDP protocol for the firewall log delivery is chosen because the network path is a direct one on one connection which minimizes the risk of any data transport loss. The TCP listener is used for the packetcapture and for the Webserver logs.

For the used channels, which stores temporary the logs in a memory buffer, the option for in memory is chosen instead of temporary storing logs in local system files. Used setting is a minimum of 100.000 events for buffer capacity and processing at once, to handle bursts of logs which can be generated during network attacks. The memory storage decision is chosen to fulfill the real-time processing property and earlier requirements in the previous chapter, but by accepting the possible loss of logs whenever the Flume agent process cripples.

Apache Flume in our setups outputs the logs towards three output destinations, Apache Kafka, Hadoop HDFS (log storage component) and optional towards the local file system.

For the log outputs, Apache Flume is configured to use streams (with integrity checks) the logs towards a specific Apache Kafka topic and Hadoop HDFS directory. For the HDFS output, the logs are processed in batches of 5 minutes and is primary used by Apache Spark for processing to detect more complex attacks. The local file output option within this design, is chosen for future possible log replay which can be done via Apache Flume itself or even by Logstash. Furthermore, the output structure is based on comma separated values (CVS) to structure each log line, Json file output structure was considered but left out of scope.

Logstash in this setup is used as a secondary source for processing and filtering logs as described earlier. In this setup Logstash is configured to receive the inputs for the pfSense firewall and secondary the Webserver Apache and system logs. On the pfSense firewall we enabled the Snort IDS functionality to enrich the visibility of possible attacks for future analytics use, which are parsed by Logstash to clean and process this data within the framework.

Logstash outputs primary towards Apache Flume (TCP listener) and secondary towards Apache Kafka as a backup and for possible future use. The design decision for this is to prevent multiple log flows throughout the framework and have Apache Flume as the primary input source for Apache Kafka to keep control and overview.

Apache Kafka is designed in this lab setup to have a single topic. A topic is the log channel where the
logs are queued to be read by Apache Storm for processing. In our setup we choose to accept the default queue capacity size of 100 Mbyte, since we use not many data sources as an input. It is possible to create multiple topics for each log type or purpose, depending on the usage of Apache Storm for attack detection and the usage of the framework.

7.1.2 ‘Analytic’ module Apache Storm

The analytics module is based upon a hybrid model for attack detection and consist of Apache Storm and Apache Spark. For the proof of concept to detect real-time attacks, a Apache Storm topology was built based on java programming technology, where Apache Storm was installed in local cluster mode setup.

A topology defines a set of tasks with the use of spouts and bolts to process data that are jointed together. Spouts are used to retrieve or receive data in the form of tuples, were bolts are used to perform logically computations with tuples and can output to other bolts or assets like databases, HDFS and many others.

In our setup the following topology was built containing of the following elements.

- Kafka spout, used to read the data from the Kafka component topic
- Parsing bolt, to structure and split the data
- Detection bolt, to apply logic to in the form of detection
- HDFS bolt, to store the processed data into the HDFS framework component as an output
- Hbase bolt, to store the processed data into the Hbase framework component as an output

This topology is shown with the following diagram. Its structure is based on a transactional topology to limit event duplication issues upon failures.

The topology runs continuously until the topology is paused or removed within Apache Storm. The Kafka spouts monitors the Kafka topic for available data, in our setup the topic is defined as ‘incoming’. The parsing bolt was configured to split up and clean up the tuples for the detection bolt as shown in the below figure 1.
In our setup we used two types of parsing bolts. Primary the simple (figure 1) but effective parser that splits the tuples into words on each space, and secondary a more complex (figure 2) parsing bolt for splitting up based on ‘combined word sections’. Words are for example within a tuple (log line) the source and destination ip addresses, source and destination port and such information.

After splitting up the tuples into words in the topology setup with the parser bolt, the words are being index into a map. The map contains of each split word with the number of occurrences. At this stage logic can be applied in the form of a threshold ‘count’ ruleset condition for attacks such as Denial of Service (DOS). Logic is in the form of ‘if’ ‘then’ statements as shown in figure 3.
Whenever the condition is met, the output will be forwarded to the HDFS and the Hbase bolts to store the processed results in the data-log storage components HDFS and Hbase.

### 7.1.3 ‘Analytic’ module Apache Spark

In our lab setup Apache Spark is installed similar like Apache Storm in a local cluster mode setup. The developed programs can be submitted to run continuously to process data for analytics purposes. In our setup we configured Apache Spark to process the data stored from the HDFS, and use Python as the preferred programming language for analytics computations. We choose the Python programming language instead of Scala due to its shorter code.

In our framework, data requires to be processed by Apache Spark is supplied by Apache Flume as a direct log stream. This log stream is saved into text files in a specific directory on the HDFS environment. We choose to have Apache Flume write text which is based on comma separated log lines and have file sizes of 5 minutes as shown in the below figure 4 for obtaining overview instead of processing many small files.

```
#Sink to HDFS
agent.sinks = x1
agent.sinks.x1.type = hdfs
agent.sinks.x1.channel = cl
agent.sinks.x1.hdfs.path = hdfs://172.16.1.100:9000/flume/k7/km/kd/kH
agent.sinks.x1.hdfs.rollInterval = 600
agent.sinks.x1.hdfs.fileType = DataStream
agent.sinks.x1.hdfs.writeFormat = Text
agent.sinks.x1.hdfs.filePrefix = events-
agent.sinks.x1.hdfs.round = true
agent.sinks.x1.hdfs.roundValue = 10
agent.sinks.x1.hdfs.roundUnit = minute
agent.sinks.x1.hdfs.useLocalTimeStmp = false
```

Spark monitors the specific directory for any new files updates. If you require to have output processed on each 5 minutes exactly, a move function in the HDFS is required with a cron job
(automatic scheduled script) or a thread sleep function (Thread.sleep()) within the Spark program whenever you submit the Spark program towards the local cluster.

The analytics data processing by Apache Spark starts whenever the files are stored in the HDFS directory. In our setup we created four machine learning programs in the below process diagram. These four programs form the core of detection process for better attack detection.

These machine learning techniques reside in the general available library. Apache Spark evolved very fast the last year, which impacted the use of the type of the machine library for our design. Spark differentiates machine learning into two libraries ‘Mlib’ and ‘Ml’. The difference between these are the flexibility of using and processing data. The ‘Mlib’ is the older type which requires a resilient distributed dataset (RDD) type, which is more ‘static’ in terms of dataset manipulation but still very powerful to obtain fast results whenever the dataset is in a ‘ready state’ for processing. The ‘Ml’ library uses DataFrames for data usage. With the use of DataFrames, the data gets placed into columns and rows like tables. This enables a lot of flexibility in terms of filtering, merging, querying and exporting. Whenever your data is not ready for processing, you still have a lot of flexibility with DataFrames to filter the dataset during data processing process.

To conclude, we used the following machine learning library types for the four programs:

<table>
<thead>
<tr>
<th>Machine Learning techniques</th>
<th>Machine learning library type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term frequency-inverse document frequency (TF-IDF)</td>
<td>ML</td>
</tr>
<tr>
<td>Word2Vec</td>
<td>Mlib</td>
</tr>
<tr>
<td>FP-growth</td>
<td>Mlib</td>
</tr>
<tr>
<td>K-Means</td>
<td>ML</td>
</tr>
</tbody>
</table>

Additionally with the use of the Spark SQL library, you can query datasets with SQL statements which can provide another level of supportive evidence for the machine learning results.
7.1.3.1 Loading data
In order to start processing data for four machine learning programs, data requires to be loaded. In our approach we used two methods during this part of the design. The default available SQLContext method via the import functionality, and we used the open source Databricks external third party library. Example codes are shown below.

Example code SQLContext:

```python
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
training_set = sc.textFile("Hdfs:/... or file:///... ").map(lambda l:l.split("\"")).cache()
LabeledDocument = Row("id", "text", "label")
training = training_set.map(lambda l: LabeledDocument(int(l[0]),l[1],int(l[2]))).toDF()
```

Example code Databricks (package com.databricks:spark-csv_2.10:1.4.0):

```python
df =
sqlContext.read.load('file:///opt/datasamples/test.csv',format='com.databricks.spark.csv',header='false',
inferSchema='true')
```

The difference between the two is that Databricks can determine automatically the schema for the log lines in most cases and loads it directly to a DataFrame. Additionally with the use of the Databricks, you can easier change the column data types to string, integer etc.

7.1.3.2 Machine learning data processing
After loading and processing the data into a schema, the machine learning techniques can be applied. It is briefly described in the following sections.

Term frequency-inverse document frequency (TF-IDF)’ machine learning technique

```python
tokenizer = Tokenizer(inputCol="text", outputCol="words")
wordsData = tokenizer.transform(training)
hashingTF = HashingTF(inputCol="words", outputCol="rawHashing", numFeatures=50)
featurizedData = hashingTF.transform(wordsData)
idf = IDF(inputCol="rawHashing", outputCol="features")
idfModel = idf.fit(featurizedData)
rescaledData = idfModel.transform(featurizedData)
rescaledData.select("id", "features").show()
```

The tokenizer is responsible for splitting up all the words within the column ‘text’. For the processing it is important to set data type to string and have more than one word. Secondly the HashingTF function creates from each row a vector where the idf function ‘scales’ the results to same sizes vectors in order to calculate which word is frequent or infrequent for the comparison.

Word2Vec machine learning technique
For the machine learning Word2Vec technique, the below code snipper shows the relationship of an ip address towards other ‘words’, results are shown in cosine numeric outcomes.
In this below example, the ip address is the attacked server in our lab setup. The MLlib library was used over the ML for the quick effectiveness, no DataFrames are required to gain some advantage.

```python
word2vec = Word2Vec()
model = word2vec.fit(training_set)
synonyms = model.findSynonyms('172.16.2.1', 20)
for word, cosine_distance in synonyms
```

**FP-growth machine learning technique**

Similar to Word2Vec ML decision, FP-growth is very effective is terms of output for determining the frequency for ‘words’. The frequency of words can be modeled with the following code where increasing the number of partitions processes the data faster in our setup. The advised number of ‘numPartitions’ is based after several tests, the higher the better results which also counts for the ‘minSupport’ value which requires to be 0.0.

```python
model = FPgrowth.train(transactions, minSupport=0.0, numPartitions=1200)
result = model.freqItemsets().collect()
```

**K-Means machine learning technique**

For using K-Means with the ML library, it is a requirement to convert the input of an integer to a double before creating vectors of the doubles. Any text which contain a dot or other structure characters must be filtered out first. In the below code the number of clusters and seed have been set to 100 and the classification ‘relationship’ between source port and destination port is used.

```python
LabeledDocument = Row("id", "sourceip", "destinationip", "sourceport","destinationport", "length")
training = training_set.map(lambda l:
LabelledDocument(int(l[0]),l[1],l[2],int(l[3]),(l[4]),int(l[5]))).toDF()
df2 = training.select('sourceport','destinationport')
vectorAssembler = VectorAssembler(inputCols=["sourceport", "destinationport"],
outputCol="features")
converttodouble = [col(c).cast("Double").alias(c) for c in vectorAssembler.getInputCols()]
df2 = df2.select(*converttodouble)
df = vectorAssembler.transform(df2)
kmeans = KMeans().setK(100).setSeed(100)
model = kmeans.fit(df)
centers = model.clusterCenters()
```

Increasing the clusters and seed improves the result, an optimal setting must be found during this process.

### 7.1.3.3 Machine learning knowledge outcomes

For the processed knowledge outcomes, the results where outputted towards files or printed toward the screen. In general two main methods were used to output to files, where Databricks is in favor due to its effectiveness.

```python
def outputtoCSVLine(data):
    return ','.join(str(d) for d in data)
lines = result.map(outputtoCSVLine)
lines.saveAsTextFile('hdfs://…/output.csv')
rescaledData.select("id","features").write.format("com.databricks.spark.csv").save("/opt/outcome/tfidf/ml/slowlorusattacksinglesourcepacketsize-idf.csv")
```
7.1.4 ‘Data log storage’ module
The data log storage contains of two main elements. The Hadoop HDFS and the Hbase database. In our setup these two elements only have the purposes of storing data. In the design for the lab setup HDFS and Hbase are fully integrated, where the Hbase data gets stored within HDFS environment.

```
<property>
  <name>hbase.rootdir</name>
  <value>hdfs://logstorage.lab.local:9000/hbase</value>
</property>
```

The reason for this integration, is to make use of the robustness of HDFS. A HDFS has the capability to easily expand its cluster by adding new hosts, even though in our setup we have chosen to configure the HDFS as a pseudo distributed single node cluster, where the data get stored on a single host as shown in the below HDFS configuration snippet.

```
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/opt/hadoop/hdfs/namenode</value>
</property>
<property>
  <name>dfs.namenode.data.dir</name>
  <value>file:/opt/hadoop/hdfs/datanode</value>
</property>
```

7.2 Attack demonstration walkthrough (scenario) approach
In order to provide detection within the framework, whether it can successfully detect attacks and fulfill the requirements, a controlled experiment for the attack scenarios have been developed based on several attack use cases. The chosen use cases for the attacks have been determined on the literature review analysis outcomes.

For the controlled experiment attack data is generated by automated tools. The automated attack tools have been selected based on their popularity, and mostly reside in the known security penetration toolset ‘Kali Linux’. In the test setup, the webserver located on the DMZ network zone was attacked by sources within the network based on real and spoofed ip addresses. No third party data presets were used during the test, to make it realistic as much as possible in terms of live dynamic data for attack detection.

In this approach, the known KDD CUP 99 attack classification categories ‘Denial of Service’ (Dos), ‘Remote to User attacks’ (R2L), ‘User to Root Attack (U2R)’ and probing has been used for references and comparison to previous research, each having their own type of attack tools and configurations.

This is summarized into the following table.

<table>
<thead>
<tr>
<th>Attack classification</th>
<th>Type of use case attack used</th>
<th>Used utilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denial of Service (DOS)</td>
<td>Resource exhaustive attack.</td>
<td>Hping3 and local ping. Nmap script http-slowloris</td>
</tr>
</tbody>
</table>
User to Root Attack (U2R) | Local brute force password attacks. | Sucrack
Probing | Reconnaissance probe attack | Nmap port scan

For each of the attack classification categories, the outcomes are shared and includes a brief explanation of the attack details.

### 7.2.1 Denial of Service (DOS)

For this use case type, a denial of service attack consisted of three types of attacks where one was applied to Apache Storm and two towards Apache Spark.

For the first DOS attack type, the Linux Apache webserver was attacked remotely with the Hping3 utility default settings together with the local ping command line option. In this type of attack, large numbers of ICMP ping packets are generated.

The data was captured by the packet capture module and processed by the Apache Storm analytics module.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Results snippet</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Storm single trigger rule</td>
<td>172.16.2.30 Dos Attack detected packetloginfo IP 172.16.2.30 &gt; 172.16.2.1: ICMP echo request, id 7299, seq 3, Dos Attack detected</td>
<td>In this example the threshold for triggering was set to a count of minimal 100 instances of an ICMP echo request type.</td>
</tr>
</tbody>
</table>

Outcome shows (results snippet) that with a single trigger rule, an attack can be detected by counting the received ‘words’ via the threshold setting via the Apache Storm detection bolt.

For the second DOS attack type, the previous type of attack is also used for Apache Spark. The configuration of Hping utility was set slightly different with the packet length size option of 120 bytes and the usage of spoofed external ip addresses. During this test some normal network traffic from other hosts (due to captures) within the network was included as well.

<table>
<thead>
<tr>
<th>Machine Learning technique</th>
<th>Results snippet</th>
<th>Comment</th>
</tr>
</thead>
</table>
| Term frequency-inverse document frequency (TF-IDF) | 3293,"(20,[4,7,10,11,14],[0,0,0,0,0,0,2.32,39599525787793,2.317806087004401])" 3294,"(20,[4,5,7,10,14],[0,0,2.4009209937149074,0,0,0,2.317806087004401])" 3295,"(20,[4,7,10,12,18],[0,0,0,0,0,0.2,2.261159051339749,2.296560978390665])"
| Word2Vec – Text mining | 80: 0.907793745087 120: 0.8209599171 | The destination port (80, webserver) and packet size length (120) is shown. Keyword for the query was the ip address of the target host. |
| FP-growth – | FreqItemset(items=[u'80'], freq=50) | Relationship between destination port |
Floris Stouten, flosto-3

<table>
<thead>
<tr>
<th>Pattern mining</th>
<th>FreqItemset(items=[u'120'], freq=50)</th>
<th>(80, webserver) and packet size length (120) is shown.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FreqItemset(items=[u'120', u'80'], freq=50)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FreqItemset(items=[u'172.16.2.1'], freq=50)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FreqItemset(items=[u'172.16.2.1', u'120'], freq=50)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FreqItemset(items=[u'172.16.2.1', u'120', u'80'], freq=50)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FreqItemset(items=[u'172.16.2.1', u'80'], freq=50)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FreqItemset(items=[u'172.16.2.1', u'120', u'80'], freq=50)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FreqItemset(items=[u'172.16.2.1', u'120'], freq=50)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FreqItemset(items=[u'172.16.2.1', u'120', u'80'], freq=50)</td>
<td></td>
</tr>
</tbody>
</table>

K-Means - Clustering

<table>
<thead>
<tr>
<th>[ 80. 120. ]</th>
<th>[ 80. 120. ]</th>
<th>[ 80. 120. ]</th>
<th>[ 80. 120. ]</th>
<th>[ 80. 120. ]</th>
<th>[ 80. 120. ]</th>
<th>[ 80. 120. ]</th>
<th>[ 80. 120. ]</th>
</tr>
</thead>
</table>

K-Means was configured to use the target port (80, webserver) and packet size length (120) as a relation to cluster data.

The outcomes based on merged results shows an attack with a close relation of the target port and packet size (K-Means) as shown by Word2Vec and the relation towards FP-growth, which identifies this type of DOS attack. Furthermore, interpretation of TF-IDF results requires a more manual analysis approach which is elaborated in the discussion chapter.

The third DOS attack type is generated by attacking the Linux Apache webserver remotely with using the Nmap Slowlorus attack script. The purpose of a Slowlorus attack is to generate many ‘open’ webserver sessions to exhaust memory resources on the webserver. In our example the attack source has the 172.16.1.180 ip address. The following below outcomes where based on the Apache Spark analytics module.

<table>
<thead>
<tr>
<th>Machine Learning technique</th>
<th>Results snippet</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term frequency-inverse document frequency (TF-IDF)</td>
<td>92,&quot;(20,[4,8,10,12,13],[0.0195700961940972 96,0.013658748931040044,0.015625317903 080815,2.9502698994772336,0.0])&quot; 93,&quot;(20,[4,8,10,13,14],[0.0195700961940972 96,0.013658748931040044,0.015625317903 080815,0.0,2.8121195609964165])&quot; 94,&quot;(20,[4,8,10,13,16],[0.0195700961940972 96,0.013658748931040044,0.015625317903 080815,0.0,3.068052935133617])&quot; 95,&quot;(20,[4,8,10,13,19],[0.0195700961940972 96,0.013658748931040044,0.015625317903 080815,0.0,2.9880102274600806])&quot;</td>
<td>Example of results snippet.</td>
</tr>
<tr>
<td>Word2Vec – Text mining</td>
<td>80: 0.722490240586 0: 0.648848831403 172.16.1.180: -0.0467723684941 53: -0.67097751492</td>
<td>Target port (80, webserver) and packet length (0) shows near relations including the attacker ip address (172.16.1.180)</td>
</tr>
<tr>
<td>FP-growth – Pattern mining</td>
<td>FreqItemset(items=[u'80', u'172.16.2.1'], freq=505)</td>
<td>Multiple connections from the attacker source (172.16.1.180) towards the destination target</td>
</tr>
</tbody>
</table>
The outcomes based on merged results shows multiple target port activity (FP-growth) by the attacker towards the webserver, where there is a relationship between target port and length (K-Means). More effectively which is not shown in the above example is to use source and target port for clustering. Interpretation of TF-IDF results requires a more manual analysis approach which is discussed in the discussion chapter. Traffic which have similar patterns won’t distinguish that much differences in outcome results with a smaller dataset, due to less outliers.

7.2.2 Remote to User attack (R2L)
This type of attack has been performed, by attacking the Linux Apache webserver remotely with brute forcing the user accounts with the Nmap script http-brute script. In our setup the webserver main landing page was password protected with a locally created username and password. The following below outcomes where based on the Apache Spark analytics module.

<table>
<thead>
<tr>
<th>Machine Learning technique</th>
<th>Results snippet</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term frequency-inverse document frequency (TF-IDF)</td>
<td>51007,”(20,[2,7,9,10,11,13,14,16,17,18,19], [1.6086984522063472,1.599552128814294E-4,1.599552128814294E-4,0.0,1.599552128814294E-4,0.0,1.3995941199904875E-4,0.0,0.0,1.3995941199904875E-4,0.0])” 51008,”(20,[1,7,9,10,11,13,14,16,17,18,19], [2.3025451081882666,1.599552128814294E-4,1.599552128814294E-4,0.0,3.199104257628588E-4,0.0,1.599552128814294E-4,2.3025451081882666,0.0,0.0,0.0,0.0,1.3995941199904875E-4,0.0])”</td>
<td>Example of results snippet.</td>
</tr>
</tbody>
</table>
| Word2Vec – Text mining | sysadmin: 1.8853445272
guest: 1.8783376534
webadmin: 1.87466342978
user: 1.87350615934 | The list of users (sysadmin, guest, etc) involved during the attack is shown. Keyword for the query was the word 401 (webserver) |
The outcomes based on merged results shows multiple failed attempts 401 code (FP-growth) by the webserver with used users (Word2Vec) by the maintainer user to authenticate towards the root user by su. K-Means was excluded due to the requirements that ‘text’ requires to be converted to numeric values for a word such as ‘authentication’. It is not impossible but requires another level of preprocessing. Interpretation of TF-IDF results requires a more manual analysis approach which is discussed in the discussion chapter.

7.2.3 User to Root attack (U2R)

This type of attack has been performed by brute forcing the local Linux user authentication database with sudo authentication attempts. Sudo is used to control root account usage for commands, and to create more (log) visibility for system activities. Sudo enables users to only run the assigned root commands to managed high privileges for managing the system. In our setup we used the ‘Sucracker’ utility to attempt, to brute force the administrator account with a large username list file. The following below outcomes where based on the Apache Spark analytics module.

<table>
<thead>
<tr>
<th>Machine Learning technique</th>
<th>Results snippet</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term frequency-inverse document frequency (TF-IDF)</td>
<td>1119,&quot;(20,[0,2,4,6,7,8,9,10,12,15,18,19],[1.4042317010854113,0.559615787935,4227,1.4683104085746448,1.3090026861,0.0,0.945737932004561])&quot;</td>
<td>Example of results snippet.</td>
</tr>
<tr>
<td></td>
<td>1120,&quot;(20,[2,4,7,12,14,18,19],[0.559615787935,4227,1.4683104085746448,1.3090026861,0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1121,&quot;(20,[2,4,7,12,14,18,19],[1.1075809586508703,0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1122,&quot;(20,[2,4,7,12,14,18,19],[0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1123,&quot;(20,[2,4,7,12,14,18,19],[1.1075809586508703,0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1124,&quot;(20,[2,4,7,12,14,18,19],[0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1125,&quot;(20,[2,4,7,12,14,18,19],[1.1075809586508703,0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1126,&quot;(20,[2,4,7,12,14,18,19],[0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1127,&quot;(20,[2,4,7,12,14,18,19],[1.1075809586508703,0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1128,&quot;(20,[2,4,7,12,14,18,19],[0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1129,&quot;(20,[2,4,7,12,14,18,19],[1.1075809586508703,0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1130,&quot;(20,[2,4,7,12,14,18,19],[0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1131,&quot;(20,[2,4,7,12,14,18,19],[1.1075809586508703,0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1132,&quot;(20,[2,4,7,12,14,18,19],[0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1133,&quot;(20,[2,4,7,12,14,18,19],[1.1075809586508703,0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1134,&quot;(20,[2,4,7,12,14,18,19],[0.0,0.945737932004561])&quot;</td>
<td></td>
</tr>
</tbody>
</table>
The outcomes based on merged results shows multiple failed attempts (FP-growth) by the maintainer user to authenticate towards the root user by the su command. K-Means was excluded due to the requirements that text requires to be converted to numeric values. Is not impossible but requires another level of preprocessing. Furthermore, interpretation of TF-IDF results requires a more manual analysis approach which is discussed in the limitations sections.

### 7.2.4 Probing attack

This type of attack has been performed, by attacking the webserver with the use of Nmap. Nmap was configured to use five source ip addresses (172.16.1.30, 172.16.1.99, 172.16.1.180, 172.16.1.200, 172.16.1.220) as the attacker to probe the webserver target service ports. The following below outcomes where based on the Apache Spark analytics module.
Floris Stouten, flosto-53

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FP-growth – Pattern mining</td>
<td>FreqItemset(items=[u'0'], freq=310) FreqItemset(items=[u'172.16.2.1'], freq=265) FreqItemset(items=[u'172.16.2.1', u'0'], freq=260) FreqItemset(items=[u'0'], freq=260) FreqItemset(items=[u'61281'], freq=260) FreqItemset(items=[u'172.16.2.1', u'61281'], freq=260) FreqItemset(items=[u'172.16.2.1', u'0', u'61281'], freq=260)</td>
<td>Fp-growth was set to query the target hosts webserver (172.16.2.1) to determine the port activity. 61281 is the main used source port by the attackers.</td>
</tr>
<tr>
<td>K-Means - Clustering</td>
<td>[ 31425.  0. ] [ 50523.  0. ] [ 8080.  0. ] [ 3306.  0. ] [ 19035.  0. ] [ 42749.  0. ] [ 443.27777778  0. ] [ 6496.  0. ] [ 11559.  0. ] [ 56945.  0. ] [ 34703.  0. ]</td>
<td>K-Means was configured to use the target port and size of the attack packet to cluster data. TCP sync packets were logged as zero packets by pfSense.</td>
</tr>
</tbody>
</table>

By merging the results whereby each machine learning techniques looks at a specific element, the results show that probing attacks can be identified. The main attacker sources are identified (Word2Vec), where a probe can be detected due to the use of static source port (FP-growth) and the sweep of source ports (K-Means) can be detected. For TF-IDF, results shows an indication but it is further discussed in the discussion chapter.

To conclude, the previous committed test for Apache Storm and Spark shows that it is possible to detect attacks. It can clearly distinguish in the type of attacks when the results are merged. Outliers and outcome classification differences can be shown. This approach can improve false positives and false negatives issues for detection.
8. EVALUATION

For the logical evaluation for this framework, the evaluation was based on the research performed by Venable, J., Pries-Heje, J. and Baskerville, R. (2012) which evaluates a DSR design on three main elements rigor, efficiency and ethics.

Rigor is defined as whether the designed artifact is an ‘observed improvement’ for the determined gaps which are fulfilled by the requirements for the framework, secondly rigor is that the designed artifact ‘works in a real situations’. Efficiency is whether the designed artifact works within the limits of organizational resources required for the implementation such as equipment, money and time. Ethics evaluation is whether the design imposes no risks during the evaluation process for the organization, people, critical organization systems and technologies.

8.1 Evaluation logical argumentation

The first requirement for the design is to provide attack detection based on open source big data analytics techniques and elements for rigorous. Throughout the design every element was built on open source technology within the framework and various types of big data can be processed and stored. Several open source functionalities were included to preprocess data, to have the capacity to process huge amount of data, before any analytics computation function was applied where the outcome is stored in the log storage component. During this process for the design we had to go through many iterations even though we had some starting point with the use of a kernel theory. Most importantly for this design was enabling of the log flow through the various components, where one issue was found in the log storage component which is discussed in the discussion chapter, but it didn’t had much impact for the completeness of the framework development.

For the second requirement, the main analytics modules provided detected capabilities by detecting attacks based on a ruleset within Apache Storm, and where Apache Spark was used with machine learning techniques, to distinguish on a more complex and detailed level between the different types of attacks based on the several defined use cases. This design provides a solution to improve false positives and false negatives for attack detection to form an initial ‘detection’ baseline to support this requirement. If Apache Storm is continuous detecting attacks for DDOS, and Apache Spark runs effectively with well-defined time schedule, the outcomes with its time based comparison and merged approach, can address the shortcoming false positives and false negatives because it will show the outliers and outcome classification differences as gaps between the outcomes.

However, as for any solution, this solution requires to be optimized to specific environments by improving the Apache Storm detection ruleset, and by processing and learning the outcomes from Apache Spark. In its evaluation, results requires always to be interpreted and verified because outcome of activities can deviate whenever there is change in log sources, users behavior, changes in the IT architecture. Another important fact is whenever the target organization has IT related malfunctions that outcomes in possible triggering thresholds in the detection rule triggers. The malfunction can be detected as a gap by machine learning techniques, but it is a false positive or negative must be investigated and confirmed. False positives and false negatives outcomes can be differently interpreted which is discussed in the discussion chapter. Therefore, a monitoring baseline must be continuously worked on, to be successful in attack detection.

Comparing the second rigor requirement whether this solution ‘works in real situations’, the solution was demonstrated on a virtual environment. Each framework component was installed and configured on various separated virtual machines interconnected with each other on network level. The developed
setup of the demonstration environment can be compared to a real organizational network environment. The firewall formed a segmented environment and each host was configured with two network interfaces for ‘production’ traffic usage and a management interface. The only limitation would be the usage of the Linux operating systems for organizations whenever Windows is preferred, but these open source technologies can be run on Windows, but it is uncertain whether there are limitations. However, the designed artifact does not imposes restrictions because it is flexible in scaling and can be extended with cheap hardware to improve performance.

In terms of organizational resources limits for efficiency, the designed artifact requires a learning curves in terms of technology usage when the technology or topic is new. It can be overcome by defining a specific initial small threat scope. Therefore it is advised to first start with the data gathering, receiving and filtering module and Apache Spark. This will create visibility in terms of understanding the desired network monitoring scope, and the required data log sources and for determining the filtering requirements. Apache Spark is very valuable for gaining this type of insights. This knowledge outcome is essential to determine the ruleset requirements for Apache Storm to detect denial of service attacks effectively. The scope can be further extended by adding Apache Storm in a later phase. Also by starting small, this framework design is feasible in terms of equipment, money and time. Although you require key users who are enthusiastic to support this organizational project. For the ethics requirement, the framework and the used use cases test formed no risks that could harm the organization, people or critical systems and technologies. During development, it is advised to use a separate testing environment and to receive a copy of the data log sources feeds.

To conclude the logical argumentation, the framework achieved the main requirements for the research gaps and the framework is implementable. However, further automation, refinement and development is necessary which is discussed in the discussion chapter.
9. DISCUSSION

Have we designed a framework based on big data analytics to detect attacks for an appropriate Critical Information Infrastructure protection (CIIP)? During various irritations with help of the DSRM methodology we have designed a framework that provides a baseline artifact that can be used as a starting point for future use. This research has tried to show as much possible to generate a comprehensive overview with all the involved open source technologies to detect attacks for a Critical Information Infrastructure. Nevertheless, there are still remaining challenges and limitations which are discussed in this section even though the results show that it can detect attacks, and it provides valuable outcome results for the false positive and negative challenges to possibly close such gaps.

Even though we found another framework developed by Singh et al. (2014) during the literature review process, that emphasizes peer to peer botnet detection with machine learning techniques, our framework considered a broader approach and included technology which can process real and near real-time (hybrid approach) data to detect attacks on various levels. Our research can possible complement the Singh et al. (2014) framework. In their framework they used the Mahout machine learning library integrated with Hadoop based on a batched approach on a smaller scale, and they used the Random Forest’s supervised learning. With supervised learning you have to instruct which event (log line) or pattern is malicious or not. This makes it more difficult in a dynamic (life) network environment for detecting botnets. Botnets have the capability to change its behavior to blend in according to the literature review section. During the built of the framework, we considered Random Forest’s supervised learning for the development, but we excluded it, because it requires a more manual approach to train the algorithm to classify each type of log line for an attack. In a life dynamic environment it could become a challenge, and secondly, it would have impacted the development process enormously if we had to find a solution for it.

Nevertheless, of course we have to recognize that there is always a certain baseline in attack detection with the use of supervised machine learning that could help to determine the gap in the type of attack traffic. However, there is no ‘silver bullet solution’ that works out of the box within an organization that gives all the threat outcomes with a push of a button with zero false positives and negatives.

To do another comparison of this research to examples in the literature review section, we extended the found research to provide a solution that can capture life traffic, process enormous amount of big data, can extend performance by increasing resources easily, used two types of analytics engines with the extension of a ruleset and machine learning techniques, and lastly, introduced a storage environment that acts as function for storing data and reusage of the outcome knowledge. Therefore, this research would be a possible extension to the trend of using multiple (hybrid) techniques to improve attack detection and limit the shortcomings with false positives and false negatives detection outcomes.

What remains an open discussion is the false positive and negatives detection for attacks. During the process of designing and developing the test lab virtual environment for the demonstration, we considered to use KDD cup 1999 dataset to prove its effectiveness for Apache Storm and for Apache Spark. We decided not to use it because the age of such datasets is rather old because attacks evolve and it would contradict our research if we would have used it. We considered it as a research gap in the literature review section. It is worrying fact, that older attacks could now be seen as the ‘noise of the internet’. How realistic would the outcome be for this designed framework in a real infrastructure, based on older attack datasets for its detection process verification. Secondly for Apache Spark, the
usage of the KDD cup 1999 dataset would rather test the quality of the machine learning techniques how they would effectively work.

Of course we have to recognize that for our (use case) approach and framework solution, both requires tuning and improvement and is not a silver bullet as well, but we attempted to use a more realistic approach by using ‘live’ attack data that is currently threatening the IT landscape. On the contrary and besides the previous statements, during or after an attack is it essential to know whether an attack was effective, which is explained with the following example. An attack can target a webserver successfully, and the detection solution in between could positively have identified it, but the outcome could be harmless when the webserver has protective measures, because of a smooth patch management and configuration management process. Without verifying the target outcomes, it still would be a considered a false positive.

This emphasize also the follow up for this detection framework, we require to correlate more data sources together, as we did manually for the attack scenarios outcomes, to determine whether an attack was successful. In our framework, it can be considered that Apache Storm populates an active list with bad attackers, which are queried continuously by Apache Storm and Spark for other types of detections. Anyhow, false positives and negatives detection is a difficult subject and requires more research.

Another shortcomings for this designed framework, is an unified log standard scheme for different types of (big data) log sources required for each analytics component. We developed a framework that can preprocess data at the receiving end (data gathering, receiving and filtering module), developed an additional regex Apache Storm bolt as the second filtering layer, and thirdly the data gets another round of preprocessing at the Apache Spark level.

Ideally, we require to preprocess data as much as possible at first layer with the use of Logstash, and have it ready for processing for each analytics engine. During developing the framework, this seems not possible because you would have many duplicate log feeds and various complex configuration. Therefore we chose to address these log filtering issues along the line to remain flexible for detection capabilities. Technically it is possible to have a minimal setup as long as you know the use cases and required logs. But this decision had impact on Apache Spark designed programs but it can work. Spark machine learning modules requires a lot of data filtering on various punctuations, otherwise the machine learning technique will fail. This is considered as part of the DSRM process during improvement iterations, but it would have been better to determine the actually required requirements at the start.

To address this similar issue for Apache Storm, an additional module input filtering is required for its future design improvement. This module could contain a presets of regexes for matching the log types, and to have a detection file ruleset that gets loaded via the detection bolt. This approach would be more flexible because Apache Storm requires a recompile built upon each code change. This possible solution would tighten the gap for regex challenges we had during the development for Apache Storm.

In terms of how logs are processed by Apache Storm during this design, we choose the at least once solution and not the exactly-once processing based on the ‘alternative interface ‘Trident’. The question is whether you require reprocessing of the same tuples upon failures. According to the Apache Storm documentation in some cases, dropping data with analytics seems to be acceptable. Therefore we decided not to choose for ‘Trident’ and use a transactional topology approach as recommended to prevent the possibility of duplication as much as possible during processing errors. Mostly processing errors occurs due to the lack of assigned performance resources for spouts and bolts in a topology.
Furthermore, during the Apache Storm topology development we ran into a compatibility issues with other framework components such as Hbase database. We tried several options by changing Apache Storm and Hbase version, but it didn’t outcome in a solution to output towards Hbase. Since most of this technology is built on java, finding the correct setup for all framework elements to support and align to each other can be a challenge, especially when you are trying to find a solution for the problem.

Anyhow, for this cases and issues during the design development, the DSRM method was suitable to help in this process. Unfortunately, we ran into time restrictions with all these challenges. We have to acknowledge that this research area was too enormous which impacted the development time for delivery. This is one of the reasons why we didn’t test Apache Storm for the probe attacks. But on the other hand, we found a solution with Apache Spark that outcomes in a complete workable substitution.

With regard to Apache Spark, it is a very joyful and powerful and fast technology, but it is important to know at hand what data requirements are needed for the machine learning techniques as earlier discussed. In the design process, the Apache Spark streaming option was excluded, because it would duplicate Apache Storm in terms of functionalities and does not benefit attack detection as we designed it to make outcome comparisons for false positives and negatives.

While during the design and improvement phase, one of the outcomes is to clean you data as much as possible and understand the data inputs for each of the machine learning techniques. Not everything within a log line can be used.

For example, K-Means requires numeric data, which means you need some sort of additional log translation scheme if you would like to process every ‘word’ of a log line. Additionally, a mechanism for optimal cluster sizes and seeds would benefit the results, but if you process data in fixed time batches this can be manually determined.

To consider another example, FP-growth machine learning technique, any duplication within a log line (even white spaces) cause processing errors because it violates the working of unique values per log line. Furthermore, like with the outcomes of Term frequency-inverse document frequency (TF-IDF), visualization is required to gain outcome knowledge. Similar types of data shows minimal outcome results which makes it hard to understand the outcome and more research is required for this section whether TF-IDF is effective for detection.

To conclude, in terms of main framework research improvements, I would recommend to research the effectiveness of the Apache Storm machine learning module to combine it together with the outcomes of Apache Storm machine learning modules. The results would outcome in a good comparison to measure the effectiveness of each analytic solution for the attack use cases, and will show better outliers and outcome classification differences.

For future Apache Spark research, the usage of SQL query statements functionality and to use machine learning pipelines is recommended with additional machine learning techniques. SQL queries were partially tested and outcomes are very valuable and effective for attack detection. Pipeline of multiple machine learning techniques is advised, because it enables to process a dataset with several machine learning techniques at once and it can be compared with Apache Storm machine learning outcomes. Besides, the whole framework requires to be more automated to support various different log sources (log standard scheme) for processing at the same time and ideally visualize the outcomes real-time. However, it can be concluded that it is possible to use this framework, but another round of automation, refinement and development would benefit the quality of the framework artifact.
10. CONCLUSION

With this research we have tried to find a solution to the shortcomings for intrusion detection where attacks can have a devastating impact on Critical Information Infrastructures. We have designed an IT artifact in the form of a framework, based on big data analytics technology that provides attack detection and provides a possible solution to improve the false positives and false negatives for attack detection outcomes. This research benefits CIIP in Sweden to detect attacks and can possibly be utilized in various network infrastructures.

During the literature review process, we contributed knowledge for several areas in the field of intrusion detection and critical information infrastructures. An extensive literature review was performed that contained most of the detection techniques over the past years. We found various amount of shortcomings for detection that can be used by other researchers to improve solutions. We created an overview for the limited research in framework development for big data analytics which didn’t existed. Added knowledge with a mapping to the different attack types and scenarios for critical infrastructures. We addresses the request for good definitions in research for big data and for a critical infrastructure. Further, several researchers attempted to create attack types taxonomies, but a common attack type taxonomy framework is missing that can be generally applied. Recommended is to create a general merged taxonomy framework for intrusion detection techniques, methods, algorithms and classification, that would benefit many researchers and practitioners.

In the design and development phase, we formulized the requirements for the framework to fulfill the gaps for the shortcomings. Based on the requirements, we extensively described the framework based on the three main modules containing of open source technology to process big data for attack detection and store the outcome knowledge. For practitioners, these outcomes benefit future development because of the detailed insights of knowledge sharing and the built framework. Several elements within this framework can be utilized in other research fields to possible close some research gaps.

In the demonstration section, we tested the designed framework IT artifact based on various attack scenario use cases to demonstrate the framework. In terms of examples outcomes, results show that it is possible to do attack detection. All three main modules can effectively work together to detect attacks for critical infrastructures to process big data. By using the hybrid attack detection approach with Apache Storm and Apache Spark with its time based comparison, merged approach and a ruleset, it can address the shortcoming of false positives and false negatives because it will show the gaps between the outcomes in its details. For researchers and practitioners in the field of detection, this section shows possible examples to gain possible benefits from these outcome results.

Besides the previously highlighted future improvements, this study is subjective to certain limitations. During the literature review process, the research area for detection was huge even though we spend enormous amount of effort to gain insights, possible research could exist that supersedes this research. This research topic was extensively large and most open source technologies were new for the author, possible shortcoming could therefore reside in its development, the understanding and usage of the used technologies. The previously shortcomings could have impacted the interpretation of the outcome results, but it was prevented as much as possible.
10.1 Future research
Future research as elaborated extensively in the discussion section, can be summarized briefly as by enabling machine learning for Apache Storm for the comparison outcome by Apache Spark machine learning techniques. Extending functionalities in Apache Spark to make use of the SQL query statements functions, combine multiple machine learning techniques in a pipeline approach and possible use additional supervised machine learning techniques. Additionally a standard log scheme for various data types for data processing and visualizing the outcomes would benefit the framework artifact. Lastly, another round of automation, refinement and development would benefit the quality of the framework artifact.
11. REFERENCES


Floris Stouten, flos-3


Om, H., & Kundu, A. (2012, March). A hybrid system for reducing the false alarm rate of anomaly intrusion detection system. In Recent Advances in Information Technology (RAITT), 2012 1st International Conference on (pp. 131-136). IEEE.


