Jaakko Nurmi

IMPLEMENTATION OF ARTIFICIAL INTELLIGENCE-BASED NETWORK AND SECURITY MONITOR

Master's Thesis

Master of Engineering, Cybersecurity

2020

South-Eastern Finland
University of Applied Sciences

South-Eastern Finland University of Applied Sciences

Author (authors)	Degree title	Time
Jaakko Nurmi	Master of Engineering, Cybersecurity	December 2020

Thesis title

Implementation of Artificial Intelligence-based Network and Security Monitor

84 pages
0 pages of appendices

Commissioned by

South-Eastern Finland University of Applied Sciences Ltd.

Supervisor

Vesa Kankare

Abstract

This thesis main topic was to develop a prototype of a real-time, programmable and modular platform for network monitoring purposes by using agile software development methods. On the platform, an artificial intelligence-based data analysis processes for detecting a change in network behaviour and methods for automatic data enrichment were implemented.

The theory part contains a discussion about the key methods and techniques which was utilized in the development process and simplified operation principles of each developed process. Some developed processes were tested practically to evaluate the problems in the processes.

Modules for automatic processing and data analysis were also developed. These modules can be connected in case it is needed.

The most important data collection methods were benchmarked to detect problematic situations in the operation in different realistic situations. With the perception from the benchmark test, the problematic parts of the data collection were discovered and proposals for the solution were made which could be developed and tested in the next iterations of the development process.

Working Artificial intelligence-based detection and data enrichment methods were created. The results of the thesis allow multiple continuous research and development projects related to data collection and data analysis with statistical and artificial intelligence-based methods.

Keywords

Artificial Intelligence, network monitoring, anomaly detection

CONTENTS

1 INTRODUCTION

The amount and versatility of different devices connected to the networks have increased significantly in ten years. Cisco VNI has estimated that devices and connections per capita will grow from 2.4 to 3.6 between the years 2017 and 2019 (“Cisco Visual Networking Index,” 2019). More than ten years ago, most devices connected to the network were mostly traditional workstations and servers. In the 2010s amount of smart appliances such as smartphones & tablets, TV’s and different IoT Devices has increased Significantly (Hetting, 2019)

The evolution of the IT-industry has caused the trend in Cybersecurity to be changed. In the Solutions Review article that was written by Ben Canner in autumn 2019, the five most common attack vectors in endpoint security were employees, mobile devices, IoT, endpoint ports and applications. (Canner, 2019).

The employees or members of the organization are one of the largest attack vectors. With a correct education, the risk that an organization gets compromised because of its employees can be reduced, but the attitude of different humans may cause difficulties for the educational process. The employees or members must see security as a component of their work process, not an obstacle. (Canner, 2019).

Another growing trend in organizations is BYOD (Bring Your Own Device) culture. The culture gives benefits for both employers and employees. Employees feel comfortable when using their own devices at work and employers might be happy that there is no need for the acquisition of dedicated workstations which costs money. In most of the cases, the employer does not have control of the security of the employee’s BYOD devices which leads to devices can be unmonitored for long periods which allows data transfers in and out from the device without any regulations. (Canner, 2019).

The third growing trend in attack vectors in the organization is the IoT, Internet of Things. IoT devices do not usually include any security protection and after the installation of the device, it is forgotten causing a blind spot in the network, which an attacker can penetrate. (Canner, 2019).

The Enterprise Strategy Group has mentioned 5 top challenges in threat detection and response. Their recent research had 379 respondents mostly in the fields of cybersecurity and IT-professionals. 36% of the respondents said that their teams have spent most of the time to address high priority issues causing a stop for the evolving of strategy and process improvement. 30% said there are one or several blind spots. 26% said that their threat detection and response is anchored by manual processes that reduce their ability to keep up with the threats. (Oltsik, 2019)

Meanwhile, an increasing number of different smart devices and other devices connected to the network have helped people in their lives, the number of attack vectors has increased and threats against, cyber-attack have arisen. While one or more of the employees might make the mistake someday and there might not be a legitimate possibility for employers to install required security software into employee's device, there aren't many choices as the solution.

An anomaly-based network traffic monitoring solution may help in the detection of unwanted data transfers inside the organizational networks. Because almost all malware or intruders cause network traffic at some point of their session, the network traffic monitoring and analysis is an ideal source for information about the health of the network or specific device.

With the rise of computational power in the 2010s, artificial intelligence technologies have started evolving a lot. In 2018, artificial intelligence was already working in multiple appliances, such as real-time language translation, chatbots, deep learning and neural networks, autonomous systems, face recognition. At the same, time there have been development appliances such as

AI-assisted healthcare, cognitive cybersecurity, self-driving cars. (Lehto et al., 2018)

Around 2018 the artificial intelligence, AI in cybersecurity has risen into the discussion and it has proved to give many improvements to threat detection. Because the number of cyber-attacks has grown in volume and complexity, AI is usually helping under-resourced security teams to be ahead of the threats. Analysis done with the help of AI allows security analysts to respond to threats up to 60 times faster (IBM, 2019a).

1.1 Thesis topic and limitations

In this thesis, the main topic is to implement the prototype of a real-time network and security monitoring system or process which gives a lot of visibility from the network, its systems, its users and events with the help of artificial intelligence. All collected data is stored in the database for the required amount of time, the collected data will be enriched with other sources of data that are freely available.

The purpose of this thesis is to address the topic from the viewpoint of the technical development of the system. The theoretical part of the thesis consists of the theory about related things such as artificial intelligence, network monitoring and data collection. The topic is very wide and allocated time resources limit the project into data gathering and storing, anomaly detection processing and analysis.

The main focus of the project is to put in anomaly detection from the network traffic. The other data sources are initially planned to be used only for data enrichment. The wideness of the topic may lead to multiple follow-up projects around the topic which can be used to expand the functionality of this system.

The system will utilize universal or standardized techniques for data collection from various sources to have an easily adaptable solution in most of the cases in case this project produces a successful-prototype of a product.

Universal techniques are defined in this thesis as a technique that is available in most of the devices and systems and the features that are available free for use.

The main goal of this thesis is to have the implemented prototype of a real-time network and security monitoring system. The designed system has been desired to be responsive, modular and easily expandable. The mentioned elements for the system might enable the continuous research development process after this thesis project.

The results of this thesis are meant to fulfill existing solutions that are available or in use in the environment, not replace any of them. Results of this project can also be linked to the planned Security Operations Center -learning environment and for future education and projects.

1.2 Background of the thesis and its commissioner

In 2015, the education in Kymenlaakso University of Applied Sciences IT-department started slowly to migrate education from data network technologies towards cybersecurity. The department has a dedicated IT-environment called ICTLAB which simulates a medium-size organization with its production network, datacenter, IT-systems, and its users. Several work-life oriented projects related to cybersecurity and data networks have been performed in this environment.

In 2017, the fusion of Kymenlaakso and Mikkeli Universities of Applied Sciences established a new South-Eastern Finland University of Applied Sciences, XAMK In the year 2018, XAMK was the second largest university of applied sciences in Finland and the largest in research and development. (“Research and Development,” n.d.).

The idea of this kind of project has been in mind from around the year 2017-2018, but because of the time spent on other projects, the implementation of it has been postponed. At the beginning of this thesis in the year 2019, the existence of commercial solutions with some similarities with this idea has been

or are going to appear in the market around 2020. The products with similarities are for example Cisco DNA Center, Stealthwatch and IBM Qradar.

1.3 Related studies and solutions

There are available several other products and services that are related to the topic and which use artificial intelligence for solving related cybersecurity problems or for finding anomalies in collected data. In the field of AI and cybersecurity, there are solutions such PatternEX AI2 developed by MIT and CSAIL, Amazon Macie, Cyberlytinc WWW-threat detection tool, CylanceProtect, Draktrace, Deep Instinct, SparkCognition DeepArmor and Vectra Network Cognito. (Schroer, 2020)

Cisco DNA Center, DNAC is a management system for the intent-based enterprise networks. It uses artificial intelligence and machine learning for monitoring the network proactively. It also helps in network optimization and troubleshooting. DNAC leans to group-based policies, micro-segmentation and AI to improve network security. (“Cisco DNA Center - Network Management and Automation,” n.d.).

IBM Qradar is a security information and event management, SIEM powered with artificial intelligence and machine learning. Like most of the SIEM solutions, it allows centralized insight for the traffic and logs that collect around the devices. It correlates and aggregates gathered data into single events to accelerate incident analysis and remediation. (IBM, 2019b).

Network architectures like Zero Trust Networks keep extended visibility of its users and devices mandatory. It helps security teams to gauge security risk related to network users and device. (Cisco, n.d.)

Passwords and other authentication methods may not be enough to ensure that the correct person is using the credentials. A user behaviour analytics is in close relationship with SIEM and it focuses to determine user activity in the IT-networks and systems to indicate anomalies in user behaviour. This might be the key thing

in the future to detect an intruder in the network. It cannot prevent an intruder from getting into the system, but it can be useful for quick detection of intrusion. (Green, 2019).

For software development, there are several libraries available that are useful for artificial intelligence and machine learning implementation. Few useful ones would be TensorFlow which is an end-to-end open-source platform for machine learning developed by Google. NumPy is the fundamental package for scientific computing for Python programming language. In the field of data collection and management, there are available solutions like ELK Stack and Splunk. (Upguard, 2020)

In the past the commissioner has attempted to implement SIEM or other security systems into the network. Most of the implementations with open source solutions have not ended in production-ready solutions due to lack of documentation or those that were never in the operational state. Also, most of the already tested systems have not included required features or have been unresponsive or hard to use.

2 RESEARCH PROBLEM, METHODS AND GOALS

This thesis project is a functional thesis which is heavily-research and development-oriented. The topic requires multi processual sub research with various research methods such as quantitative and qualitative research methods in order to be able to perform the main project and give meaningful results to the system that is being designed and developed.

Quantitative research with different observing and statistical methods is used, for example, in a network traffic analysis. Qualitative methods are used for understanding the results and the reasons for them.

A research and development model suits perfectly this thesis project because a new product or process is implemented in this thesis project. The results of this

project can also evolve after the thesis is done. The development process is done by using agile software development methods.

2.1 Research methods

Data collection, analysis, description of collected data and used research methods are defined in this chapter. Research requires multi-dimensional control of data, methods and rules. Research must always follow the rules that are defined in the branch of science. This usually makes research differ from research-oriented development work which can include research elements but not how they are understood in academic circles. (Salonen, 2013).

Quantitative research is the orientation of scientific research strategy where the research target is descriptive and analysed with statistical and numerical data. Different statistical and computational analysis methods are usually used in this kind of research. Qualitative research methods are scientific research where the quality, features and consequences of the research target are tried to be understood. Quantitative research methods are usually used as the pair with qualitative research and both methods can be used in the same research. (University of Jyväskylä, 2019).

In the Investopedia article written by Will Kenton, research and development, R&D is a term for the activities in organizations that innovate and introduce new inventions or improvements to the existing ones. The final goal of the R&D activities is usually to bring new products and services into organizations. It is usually one of the first steps in the development process. (Kenton, 2019).

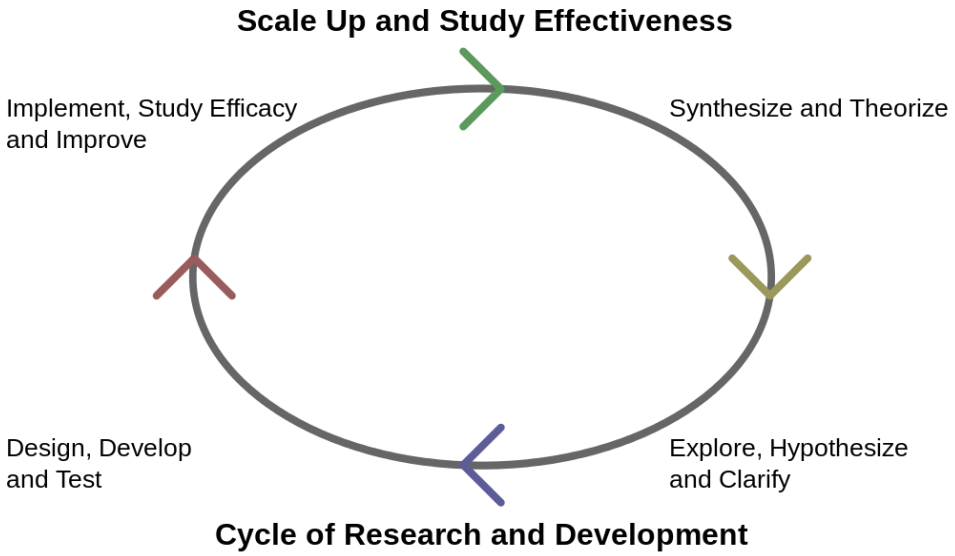

Figure 1: Cycle of research and development process (Goran tek-en, 2013).

Referring to figure 1, R&D methods are the non-stopping cycle of different processes. All research and development cycle starts from brainstorming the ideas. In this Synthesize and Theorize stage project team considers the issue and the potential of the idea for the specific industry. Each project has challenges and some of them might be impossible at the specific iteration of the cycle. (NJRD, 2019).

Explore, Hypothesize and Clarify the stage in the research and development cycle is the second step. In this step, the research or project team explores the theory and existing studies around the idea of depending on the development case, some quantitative research such as surveys or other data collection method might be required to receive enough theoretical framework for the development project. Usually, prototypes are started to be developed in this phase which enables the hypothesis and exploration. (NJRD, 2019).

The third phase is Design, Development and Test, which is the phase where the road map from the idea to the product is created. This phase is usually full of trial and error type iterations. Each one of the iterations may improve the design of the prototype products where flaws and limitations are fixed. (NJRD, 2019).

The last phase of the cycle is to Implement, Study and Improve. The product is usually still improved in the previous phase meanwhile in this phase all required criterions are checked for final approval and launch of the product. After this the development process does not end, it usually returns to the Synthesize and Theorize phase where the development project team gets ready to improve the developed product (NJRD, 2019).

Agile software development is a group of alternative methods for the traditional software development method that is called the waterfall model. The aim of the agile methods is to develop the software quicker and have a faster response to the requirements and simplifying the development process. (Abrahamsson et al., 2017)

Common things for different agile software development methods are relatively short iterations of software development. This helps to reduce risk in the development process. Each iteration consists of the same elements the usual software development project plan, requirement analysis, program design, development, testing and documentation. (Abrahamsson et al., 2017)

Software development is agile when development is done by releasing software with small changes in rapid cycles, able to make changes in the last moment, the method itself is easy to learn and to modify for specific needs. It is also common that the customer and the developer are working together. (Abrahamsson et al., 2017)

Multiple frameworks for agile software development have been created. The most common ones are Extreme Programming, Lean and Scrum. Newer agile methods consist of frameworks such as SAFe, Scaled Agile Framework. (Abrahamsson et al., 2017)

2.2 Research questions

The thesis research questions come from both a quantitative and a qualitative side. The questions are at least the following:

- Can the system setup and detection process be automated efficiently to get solutions like this quickly into production?
- How many computational resources would be in this case?
- Is it possible to detect malicious network traffic only from network flow statistics?
- What is the false positive rate when flow-based traffic statistic gathering is used as a source for anomaly detection?
- Can detection accurateness and usefulness be improved significantly by enriching the flow data with other data sources?
- How much machine learning would improve threat detection ratio and its accurateness?

2.3 Project and research objectives

The thesis is defined to be a functional research development project, which may give a large number of objectives. In this thesis primary objectives are set to the following:

- Implement the prototype of a modular framework/core for network and security monitoring systems.
- Increase the commissioner's knowledge for using Artificial Intelligence for cyber threat detection.
- Implement and/or discuss different data collection methods and review the security of the implementations of them.
- Research by observing common attack patterns from the data that was collected by the data gathering.
- Initial research and implementation of artificial intelligence and machine learning algorithms that can automatically learn normal traffic behaviour in the network and detect possible anomalies in the collected data.
- To have an initial start for Artificial Intelligence research in the field of cyber-security at the commissioner.
- To increase visibility at the commissioner's network and IT-Systems

3 THEORETICAL FRAMEWORK OF USED DATA SOURCES AND COLLECTION TECHNIQUES

3.1 Simple Network Monitoring Protocol

Simple Network Monitoring Protocol, SNMP is a versatile and relatively simple protocol that is used as a communication channel between network management stations and network elements. Management stations run monitor and control applications which are controlling and monitoring the network elements via SNMP

clients. Network elements are hosts and network equipment that runs SNMP agents. (Davin et al., 1990).

SNMP is quite an old protocol, it was originally developed in the 1980s to reduce the complexity of management of different devices. RFC 1157 document defines the goal of its design to reduce development costs for management agent software, have a functional paradigm for monitor and control operations, and be an independent protocol as much as possible. (Davin et al., 1990)

By 2020, the Simple Network Monitoring Protocol is commonly found in most of the network equipment and servers facility automation and monitoring devices.

3.2 Flow-based traffic monitoring

The network flow is a record of communications between two different points. The record is bounded by the open and close of the communication session. Flow monitoring is usually used for network traffic monitoring from the perspective of traffic analysis and bandwidth monitoring. It can track the flow of the applications and key services in all areas of the network devices, servers and links. Common network protocols for network flows are, Cisco NetFlow, IPFIX, sFlow and JFlow. (Conklin, 2018).

Flow protocols contain at least connection session information about:

- Source IP-Address
- Destination IP-Address
- Layer 4 protocol number
- Source and Destination port in the case connection session was
- OSI-layer 4 (Transport layer) or above.
- Timestamp from the beginning and end of the flow.
- Transferred packets count within the session
- Transferred bytes count within the session

(Muniz and Santos, 2017)

The data provided by flow protocols can be used as a source for anomaly detection to detect abnormal traffic patterns from the flow. In the year 2004,

Myung-Sup Kim and others in various institutes found that it was possible to detect malicious activities directly from the flow data:

- Denial of service and distributed denial of service
- Network scanning
- Network flooding
- Malicious worms that are spreading from one workstation to another.

(Myung-Sup Kim et al., 2004)

The data might also help you to answer questions like:

- Which device is executing banned application such as BitTorrent
- Who is hogging all the bandwidth and slowing down the connection
- Where a hacked system was connected during an infection.

(Petryschuk, 2019)

Flow-based traffic inspection can be powered by anomaly-based detection. Anomaly-based detection is usually used in the process to detect new types of attacks in the networks. An ideal approach is to use artificial intelligence and machine learning techniques to record the baseline of the network traffic flow and compare it to the current flow to detect anomalies in the network. (Sadek et al., 2013)

3.3 Packet-based traffic monitoring

The packet-based traffic capture process allows the inspection of each packet with its contents that was received. The capturing can be established by tapping the cable or fiber or by listening to wireless networks. While previously mentioned methods are used mainly by intruders, it is also possible to use a port mirroring feature in various switches. The port mirroring feature in the network equipment forwards all packets from the monitored interface into the mirrored interface. (Davidoff and Ham, 2012)

This method gives the possibility to inspect the contents of each packet, which allows a signature-based detection of attacks. The signature-based detection compares the contents of the packet to a hash or pattern that is stored externally.

The signature-based attacks can be only detected if they are known, which means that it does not allow the detection of new types of attacks. (Ďouligeris and Serpanos, 2007).

3.4 Flow versus packet-based traffic information collection

Referring back to chapter 3.2, where it was mentioned that the flow record contains only specific statistical information from the traffic flow, not any of the payloads of the recorded flow sessions compared with the packet-based where the full content of a packet is included. Packet-based captures are precise in terms of resolution while flow records are only statistical. This leads to packet-based inspection to require much more computational power compared to flow-based inspection. However, both methods have their designated use cases and both methods could be used simultaneously. (Endace, n.d.)

Port mirroring is a common source of traffic for the packet capture process in the networks. It is difficult and expensive to implement efficiently on larger networks because traffic volumes are higher. Higher traffic volumes can cause the mirror interface to be saturated and some packets will not be received in the packet capture. This may lead to signature-based attack detection not to detect the attacks. (Wilson, 2019)

While most of the relevant information is encrypted with SSL in the packets that are transferred through the networks. This means that packet-based traffic collection does not give more information directly about the contents of packets either protocols that are above layer 4 in the OSI-model. While traffic encryption keeps files transferred through the network safe, it also allows the malware to have safe connections which create challenges for signature-based attack detection. (Anderson et al., 2016)

3.5 NetFlow protocol

NetFlow is a protocol that is developed by Cisco Corporation for recording metadata about the IP-traffic. Typical sources of flow information are network

devices such as routers and switches but also other appliances such as servers, firewall and virtual machines. IPFIX and sFlow are the most important variants of the NetFlow. (Kentik, n.d.) In table 1, a vendor of each flow protocol is listed and shortly compared against NetFlow protocol.

Protocol	Vendor	Record / Protocol Features
NetFlow V5	Cisco	A fixed record including IPV4 features, bandwidth and timestamp for example
NetFlow V9 / Flexible NetFlow	Cisco	Template-based with up to 104 standardized features, including relevant L3 and L4 features, bandwidth, Application ID, Protocol, Timestamps
IPFIX	IETF Standard RFC 7011	Based on IPFIX, support vendor-specific and variable-sized features such as HTTP URL
J-Flow	Juniper	Juniper implementation of NetFlow 5,8 and 9. Cross compatible with the corresponding version of NetFlow
NetStream	Huawei	Mainly detailed data based on resource usage including bandwidth, IP addresses, Time, ToS, Application
cflowd	Alcatel-Lucent / Nokia	Functionally equivalent to NetFlow with multiple NetFlow version from 5 to 9 and IPFIX
RFlow	Ericson	NetFlow 5 Based
sFlow	InMon Corporation, sFlow consortium	Sampled NetFlow, sampled record containing full packet with packets header.

Table 1: Short comparison of different flow protocols.

NetFlow implementations have a specific architecture that can be seen in figure 2. Network equipment sends NetFlow packets towards the collector and sends them to an analyzer. (Davide, n.d.)

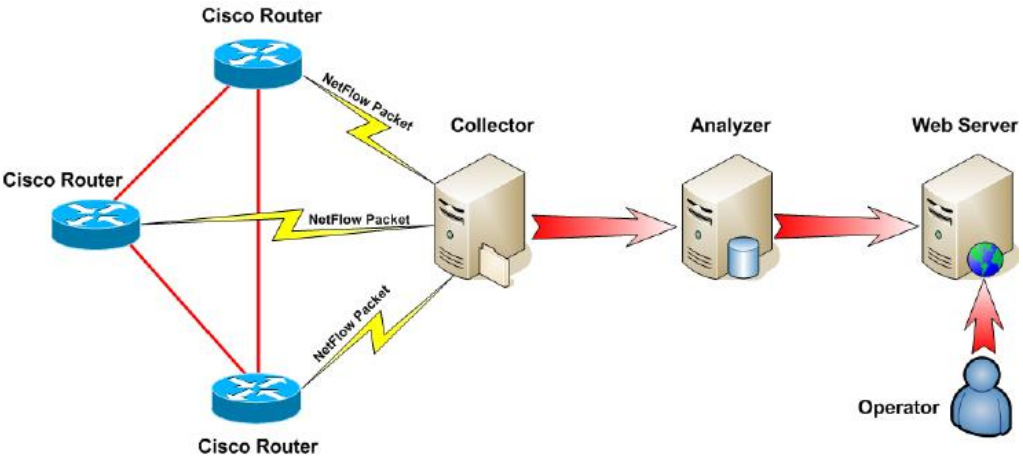

Figure 2: Typical NetFlow architecture (Davide, n.d.)

NetFlow protocol has nine versions but only two of them are in wide use, version 5 and version 9. NetFlow version 5 is supported in the large numbers of network gear also from different vendors. Version 5 is limited only to IPV4 fields. (Muniz and Santos, 2017)

NetFlow version 9 is a template-based implementation of NetFlow protocol which supports hundreds of features in the flow record compared with older versions, which supported only specific fields. Version 9 allows more flexible flow records because. To parse the NetFlow record at the collector, the template must be received. Figure 3 presents the structure of the NetFlow version 9 template FlowSet. (Muniz and Santos, 2017)

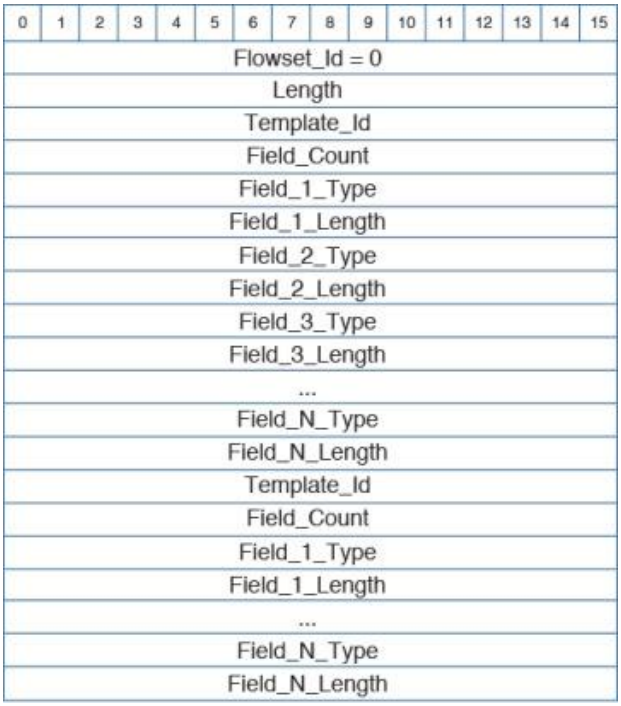

Figure 3: Structure of NetFlow v9 template FlowSet. (Muniz and Santos, 2017)

IPFIX, IP Flow Information Export is the variant of NetFlow protocol that is based on NetFlow version 9. It is standardized by IETF in RFC7011 IPFIX nearly similar to NetFlow Version 9, but it has two large features that are not included in NetFlow Version 9. The first of the differences is that the IPFIX allows a VendorID field to be specified which allows exporting proprietary features that are not part of the standard. Another feature is that IPFIX allows variable-length fields that allow exporting URLs or messages for example. (Trammell and Claise, n.d.). IPFIX is also known as NetFlow Version 10.

sFlow is the variant of flow protocol that is meant for high-speed networks. Compared with NetFlow protocol it relies on sampling packets instead of connection sessions. In practice, it works by selecting random packets from incoming interfaces and send samples to the collector. (Panchen et al., n.d.)

sFlow uses two sampling mechanism, packet-based sampling and time-based sampling. Packet-based sampling samples one packet from the specified number of packets. Time-based sampling samples interface statistics. (Panchen et al., n.d.)

The architecture of sFlow and sampling techniques provides continuous network-wide traffic monitoring. The designed addresses the problems that are associated to monitor network traffic at gigabit speeds and faster and scaling to manage thousands of agents from single points. (Panchen et al., n.d.)

In the case of this thesis, the NetFlow version 9 was selected as a primary format for traffic flow source because it provides the most accurate format. It is important to receive all information to gather enough and accurate information for the detection. Sampling mechanisms may cause that important packets are not collected, which may lead to problems in anomaly detection. Also, the developers of Flowmon software rely on NetFlowv9 or IPFIX formats because those prove the most accurate and comprehensive information from high-performance network monitoring appliances. (Listvan, 2018)

3.6 Centralized log collection and management

Centralized log management and log collection is a technique where logs from multiple sources are stored in a centralized location. It enables an ability to search quickly relevant events from various logs and allows a single point for the log retention period. It also enables a single point to define different alert patterns for specific events. (Morgan, 2016).

Security of centralized log management is one minor thing of concern in this thesis. Referring to NIST report SP 800-92, Traditional Syslog implementations are connectionless which makes them unreliable and the server does not perform any access control. The possible attacker may also monitor logs that are transmitted over the network and gain sensitive information. (Kent and Souppaya, 2006).

RFC 3195 is a standard for improving the confidentiality, integrity and availability of the logs. It defines that connection-oriented transmission protocol must be used as a connection protocol for ensuring the log message reaches their destination. TLS protocol is recommended to be used to ensure the

confidentiality of the logs. To ensure integrity, RFC 3195 recommends the message digest algorithm to be used. (Kent and Souppaya, 2006)

4 THEORETICAL FRAMEWORK OF DATA PROCESSING AND ANALYSIS METHODS

4.1 Data enrichment

Single heaps of collected data around the systems may not help to discover the overall view of the events and the relationship between them. Collected data is usually dumped into the database. This data is called raw data and it is not often used in wide contexts. Data enrichment is the way to make raw data more useful to have a deeper insight into relationships between the datasets. (Todd, 2018).

It has been found that data enrichment is the way to make event data more meaningful for threat detection, hunting and incident response. In the case of security, the typical process is to perform data enrichment by adding contextual information into the events. In the article: Data Enrichment: The Key Ingredient for SIEM Success following typically used contextual information has been defined.

- Identity context
- Assets information
- Access privileged
- Non-technical feeds
- Vulnerability context
- Social and online context
- Network maps and geolocation

(Sharma, 2019).

Enrichment of the data provides an additional viewpoint for the data, which is extremely useful in use cases in the field of cyber-security because when combining multiple data sources, the big picture of the event is generated. Security analyst Ertugrul Akbas defines several information sources that can be used in the data enrichment process in the context of cyber-security. The sources are the following:

- Operating systems
- Logged-in user
- Active Directory memberships
- Resource utilization
- Geographic location
- Associating IP addresses with users, machines, and timelines
- Tracking asset ownership
- Associating user and machine types with activities
- Correlating personal email addresses with employees

(Akbas, 2019)

4.2 Artificial intelligence and machine learning

Artificial intelligence, AI is a common term for the computer application which can execute tasks that are defined to be intelligent The definition of artificial intelligence is very wide and it varies in different magnitudes and viewpoints. (Kaplan and Haenlein, 2019). The four types of artificial intelligence are reactive machines, limited memory and the theory of mind and self-awareness. (Hintze, 2016)

In the book named Introduction to Artificial Intelligence for Security Professional, written by the Cylance data science team, three more types of artificial intelligence are defined. Those are named Artificial Narrow Intelligence, Artificial General Intelligence and Artificial Super Intelligence. (The Cylance Data Science Team, 2017)

Reactive machines are the most basic types of AI. They don’t store memories which lead to that it cannot be improved by practising. It makes its decisions on current situations that are predefined. A good example of reactive AI is a chess computer. The chess computer has defined instructions on how each of the pieces moves and knows where they are located. It does not need the memory of existing moves, because it only needs to compute the most advantageous movement for itself. (Hintze, 2016).

The limited memory machines have the capabilities of reactive machines but have a memory that allows decision-making also from the historical data.

Currently, the development of AI is in this stage and most of the AI applications are of this type. Typical use cases of this kind of AI are chatbots and self-driving vehicles. (Joshi, 2019)

Machine learning is a specific field of artificial intelligence. It consists of several methods of data science methods such as clustering, regressions or artificial neural networks.

Clustering is one of the techniques that is used in the field of machine learning. Its purpose is to group data with given rules. Generally, data points that are in the same group have similar properties or features. Unsupervised learning is one of the use cases of clustering. (Seif, 2019)

Cylance security team mentions that machine learning is raising the bar for attackers. In response, attackers are also utilizing machine learning technologies to find new ways to penetrate the target. Because of this machine learning is needed on the defensive side to maintain parity between attack and defensive trends. (The Cylance Data Science Team, 2017)

4.3 Decision tree algorithm

Decision trees are one of the most used algorithms that are used in machine learning. It mimics a human brain mostly for classification and regression-based problems. Because of its near relationship against human thinking, it is easy to understand and decision trees are the most popular machine learning algorithms because of their simplicity. (Grimaldi, 2019).

Decision trees are built from nodes, edges and leaves. Nodes are used to test the value of a certain attribute. Edges correspond to the outcome of the test performed in the node and connect the value into the next node of the leaf. Leaf nodes terminate nodes that predict the outcome of the tree. A simple example of the decision tree algorithm can be seen in Figure 4. (Chakure, 2019)

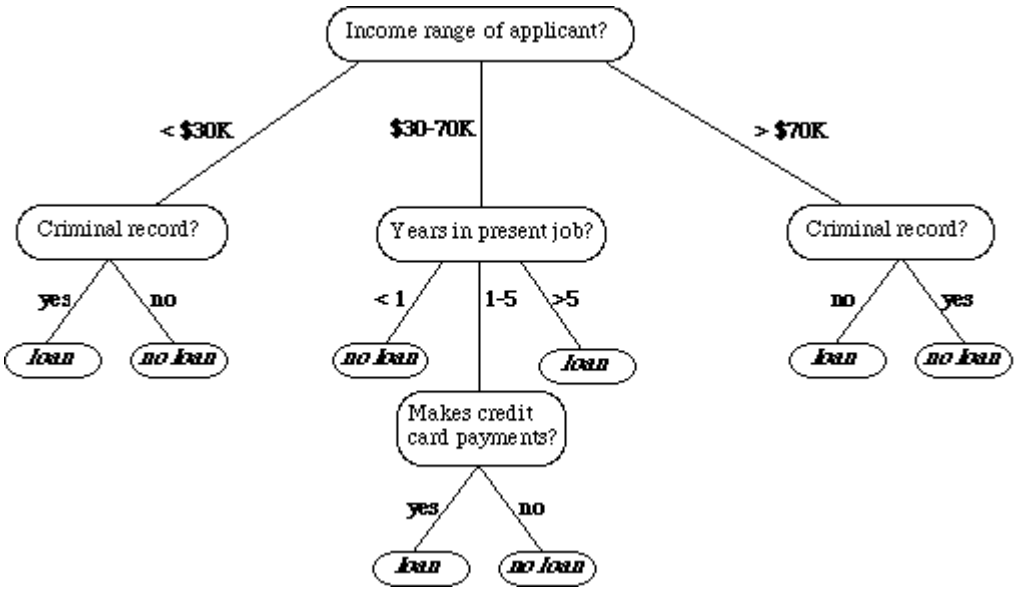

Figure 4. Example of graphical presentation of simple classification decision tree algorithm (Mady, 2018)

Referring to Figure 4. The decision tree algorithms are just a set of questions with conditional answers. In the cyber threat detection process, the decision trees might come very complex. By looking into the decision tree structure, some input values might cause unstable results. In some cases, the feedback connection back to the tree would be desired, but in theory, it might cause a never-ending loop in the decision tree.

The main types of decision trees are classification trees and regression trees. Classification trees are used for classification and output is discrete, typically yes or no. In regression trees, the target variable can take continuous values. (Chakure, 2019)

An inexpensive construct, high performance when classifying unknown data and the accuracy is comparable to other classification methods for multiple simple datasets are the advantages of the decision tree algorithm. As a disadvantage, the decision tree algorithm is easy to overfit and small changes in the training data easily result in a large change in the logic of the decision tree. (Chakure, 2019)

Decision tree algorithms are used in multiple applications in the fields of biomedical engineering, financial analysis, astronomy, system control, manufacturing and production, medicines and psychics, for example. In these fields, it has been used for things such as galaxy classification, quality control, diagnosis, cardiology and particle detection. (Chakure, 2019).

4.4 Artificial neural network

This chapter introduces the artificial neural network which is a computational model that works like the way how human brains process information. The model consists of neurons that are also called nodes. These nodes receive information from the input layer and other nodes. (ujjwalkarn, 2016)

Artificial neural networks consist of three types of layers, an input layer, hidden layers and an output layer. The input nodes provide information for the neural networks. At the input layer, no computation is performed, it is left for the rest of the layers. (ujjwalkarn, 2016)

The nodes in hidden layers perform the computation of the data provided by the input layer. Nodes at this layer do not have a direct connection to the outside. The artificial neural network model can have multiple hidden layers which enable the deep learning process. (ujjwalkarn, 2016)

Nodes in the hidden layer get their values by the activation functions. Activation functions are like a mathematical gate in the current node between the input and output of it.

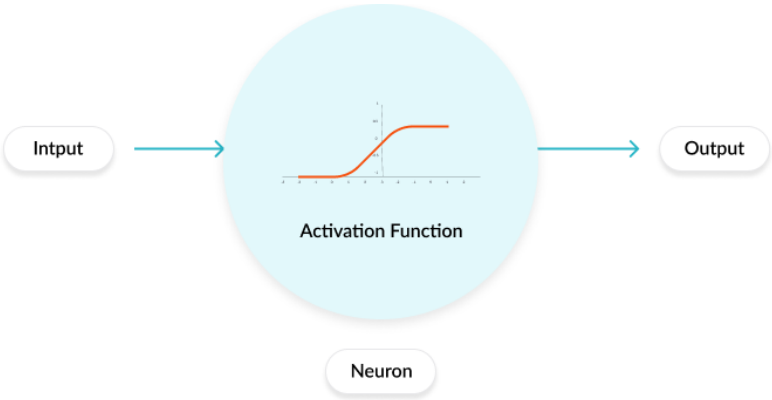

Figure 5: Activation function in the node. (Missinglink AI, n.d.)

There are several activation functions available. The activation function can be set in three categories: linear and non-linear activation functions and derivatives or gradients of activation functions. (Missinglink AI, n.d.)

Binary step function and linear activation function is included in the category of the linear activation function. In a binary step function, if the value of the input is above or below a defined threshold the node is activated and sends exactly the same signal to the next layer. Linear activation function multiplies the weight for each node and creates output that is proportional to the input signal. (Missinglink ai, n.d.)

Non-linear activation functions are used in modern neural networks. These functions utilize complex mapping between the nodes of the network. Common non-linear functions are Sigmoid, Hyperbolic Tangent, Rectified Linear Unit and Softmax. (Parmar, 2018)

Loss functions define how the specific algorithm models the given data. The machine learns by the means of a loss function. Any of the loss functions give one size fits all. The selection of the loss function depends upon the learning that is performed. (Parmar, 2018)

These loss functions can be set in two categories, regression losses and classification losses. Classification losses are used to predict the output of a set

of output values that are categorized. Detecting handwritten alphabets from the given dataset and fitting them into the corresponding category for example. Regression losses are used when predicting future values from datasets. (Parmar, 2018)

5 DETECTION OF MALICIOUS TRAFFIC WITH ANOMALY DETECTION

The versatility of ways how malware or even legitimate software communicates through the network makes the task of detection of malicious traffic difficult. Known malware can be detected by comparing their signatures with known ones. However, the signature-based detection would not work in the case where the malware and command and control channel is designed for a unique target or it is more advanced and it mutates the way of the communication. New and upcoming solutions are more based on anomaly detection by using advanced analytic methods. (Luis and Franklin, 2019)

The malicious anomalies can be found by look for the following things from the network data:

- Is the protocol expected to see in the current environment?
- Are there unrecognized port numbers in use?
- Is the bandwidth caused by the protocol or destination normal?
- Does the pattern of the traffic or protocol look like it should?
- Are there suspicious destination and source IP combinations?
- Is the behaviour of the protocol different when compared with the other traffic using similar ports?
- Is there application-level detection that shows the application in the port where it does not belong?
- Are there some protocols that communicate periodically?
- Is there traffic that looks proportionately the same in the chain of connections in the same time window?

(Luis and Franklin, 2019)

Above mentioned questions just present the basis of the detection of malicious traffic. Each of the questions might give sub-questions after the answers have been given. Any of the questions does not give answers is the communication legitimate or malicious, because many legitimate communications through the network might give positive answers to the multiple questions which might result

in false-positive results. While there is no direct answer to the question, is the traffic malicious or not, the enrichment of the data would help to gain more resolution to filter out the sure false-positive result and reducing the number of anomalies required to handle manually.

There are various indicators for detection which can be used to detect and classify malicious events in the network by using the network traffic flow information.

5.1 Indicators of malware command and control communication in network level

At the level of network communication, malware communication looks like all other packets. There are numerous ways of how malware can communicate at the network level. This chapter only deals with the signatures that have been researched or utilized in this thesis.

Malware typically uses a particular protocol which behaviour differs from a legitimate one. The content of the packet may differ from legitimate traffic. The duration of the connection session might be longer than in legitimate traffic. Detection of this kind of behaviour can be detected by creating signatures from the legitimate traffic and compare incoming traffic records to them. (Gardiner et al., 2014)

Newer techniques of implementing command and control channels having an ability to hide the detection by causing a little traffic as possible or hide inside the legitimate traffic. This kind of traffic also has specific behaviour, which can be detected by looking at the length of the session. For example, typical HTTP sessions are short and generate many flows to a single destination or multiple destinations. Malware beacon communication typically spreads over a large period. (Soniya and Wilscy, 2016)

Another way to detect beacons is to check the uniformity of communication. Usually, malware beacons exhibit regularity in the packet count and interval in the

specific time window. Kolmogorov-Smirnov test is a method for calculating a maximum difference between empirical distribution and a known cumulative distribution from the list of data points. (Soniya and Wilscy, 2016)

A more advanced method from the command and control channel is to use application-level protocol for communication to hide from the network intrusion detection systems. One of them is called DNS beacon which data travels inside the DNS queries. The DNS beacon can be detected by tracking the egress DNS queries and their interval. (Sheridan and Keane, 2015)

5.2 Indicators of network and port scan

A Network and port scan are early indicators of the attack. The scanning helps an intruder to enumerate the hosts in the network. There are multiple variants of a different kind of scan having different behaviour in the view of network traffic.

Typical scanning activity can be detected relatively easily because it creates lots of noise which can be seen in the metrics such as created network flows per host or created different networks flows per host. Indicators of this kind of activity can be achieved by tracking the number of connections created by a single host in the network. (Ring et al., 2018)

Detection of slow-scan methods is more challenging because networks typically have a high amount of noise generated by other activities which allow slow port scans to hide into other traffic. Typically, the scanning applications loop through a specific network range which causes a failed connection establishment in the case there was no Host IP address or specific port up. (Ring et al., 2018) For example, in TCP SYN scan, which is the most common TCP port-scanning method, failed connection establishment indicators are the following:

- There are no traffic flows in the reverse direction
- The TCP RST flag is not set
- The number of packets to the destination is low
- The number of failed connection establishments of the source IP for a specific time window is higher than in other hosts.

The typical behaviour of TCP protocol causes at least SYN, ACK flag pair set in return traffic in case the connection was successfully established. In case there was no listener attached to the destination port there will not be return traffic. The TCP connection is terminated with RST flag. (Postel, 1981) Looking at the number of failed connections caused by a single host is a good metric because typically the connections are successfully established if the host is up.

6 OVERALL PICTURE OF THE IMPLEMENTATION

A prototype implementation of the system was created into the ICTLAB research and education environment of South-Eastern Finland University of Applied Sciences. The environment consists of Active Directory domain controller, firewall, a core switches Cisco 4500-X and various amounts of different kinds of services and servers.

Figure 6 presents the simple logical view of the system. It consists of different sources of data, such as firewalls, servers and network equipment. All this data is gathered into a centralized data collection server which has listeners for each data source for receiving. The network flow information data is generated by a packet capture device that is attached with 2x 10Gbit/s connectivity to each of the 4500-X switches.

The solution allows a hybrid model of packet-based and flow-based inspection. A full packet with the payload can be stored in the packet capture device RAM-memory to allow to return it for deep packet inspection purposes in case the anomaly detection process finds something suspicious. However automatic method for this use case was not implemented within this thesis.

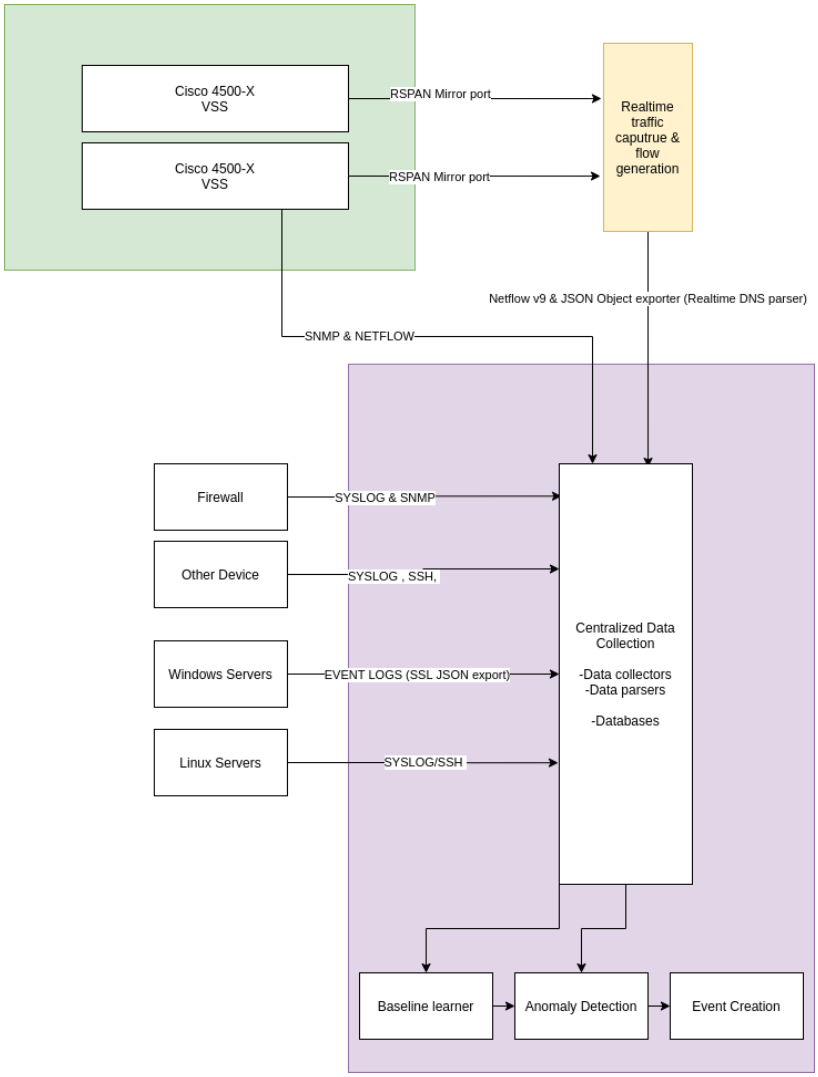

Figure 6: Overall picture of the system

In the original plan, it was supposed to receive the NetFlow records from the core switches of the network directly without the dedicated packet capture solution. VSS, Virtual Switching System feature that is in use in the core switches caused problems for the integrity of the NetFlow data. About 50% of the traffic flow information was newer exported by the Cisco 4500-X switches. The reason for that was that the switch which was in the standby state was not able to export the flow records of the traffic while another switch was active.

Centralized data collection is one of the core parts of this thesis. It consists of data collectors and processors. The collected data is used to create a baseline

model of the data and detect anomalies from the collected data against the baseline model and create events from the detected anomaly.

7 IMPLEMENTATION OF DATA COLLECTION PROCESSES

The implementation of the data gathering process is the core part of this thesis. Data gathering techniques were selected to be universal or freely available to allow implementation without obstacles caused by financial resources. The data gathering processes were programmed by using Python3. Each of the processes was programmed as modules, which allows easy reuse of code in other projects.

For this thesis project, four data collection techniques were implemented. The methods or techniques are the following:

- Real-time network traffic capture
- Microsoft Windows Event Log forwarder
- Syslog server
- SSH stream reader

Design each of them tries to follow similar design with each other and confidently, integrity and availability in the case that is reasonable, either possible to implement with the corresponding technology. Each of the datasets that are output from the module is in JSON format, JavaScript Object Notation format.

In Figure 7, the operation of the data collection module is presented as a diagram. The green area is the main application, where the module has been linked. The blue area is the Python collector module. The red area is the worker thread inside the module.

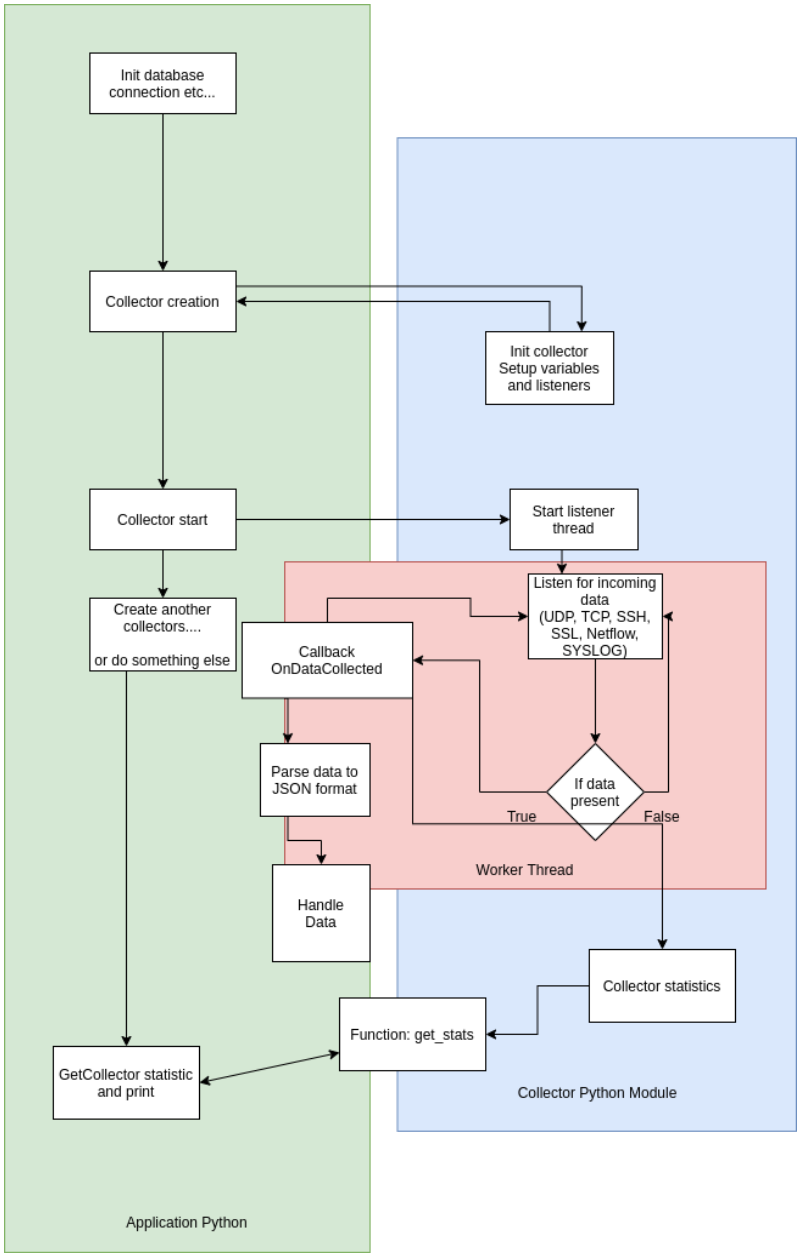

Figure 7: The design of the data collection module.

In the collector initialization phase, the required variables are set for the module. After that, the start call initializes the worker thread for the module and it becomes active. Each of the collectors has collector statistics, which stores the numerical statistics of the collector. This statistic can be used as data for EPS (Events Per Second) metric or calculating the latency of the data collector.

The operation of the worked thread is relatively simple: Receive data and fire callback functions on different events and update collector statistics. Parsing the

collected data in the case that is required to do and other data handling is also done in the worker thread. In the case the ingress volume of the data collector process is high and the data handling process is relatively heavy, it is recommended to separate the data handler to another thread to prevent the saturation of the collector.

This design prevents other collectors from blocking other collectors and the system operates in non-blocking mode. The modular design also allows easy integration to the other implementations of AI processes or other data science projects relating to the topic.

7.1 Real-time network traffic flow capture process and flow collector

Network flow information is a valuable source of data to improve the visibility of the networks. Due to the problems for implementing NetFlow records and exporters into the two Cisco 4500-X Switches, this data collection technique required a different approach to get traffic flow information that is accurate and real-time. This can also be implemented into other vendor hardware, which supports port mirroring. Figure 8 presents the implementation.

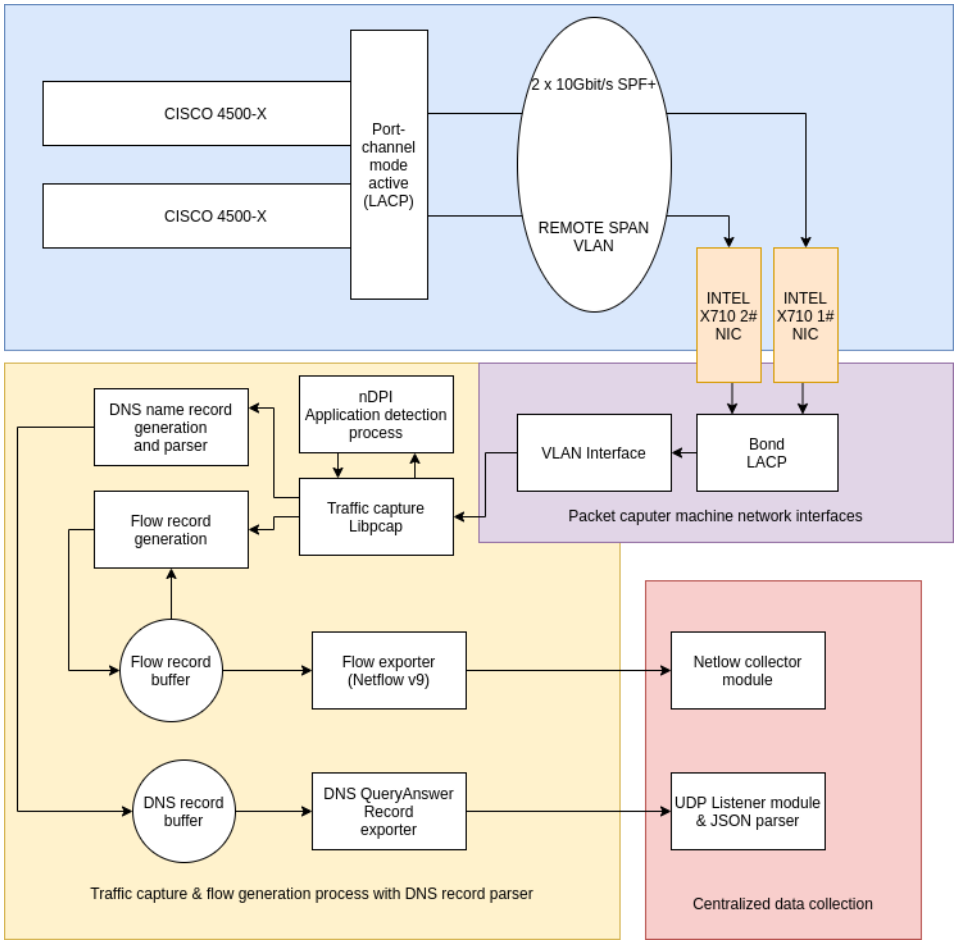

Figure 8: The operation of the traffic capture process.

The traffic is captured from the ICTLAB core switches, where all the network traffic passes. Traffic is exported by using port mirroring technology called Remote Switch Port, RSPAN. Each of the switches has a single 10Gbit/s configured as a mirror interface where all data incoming to other interfaces are forwarded. The physical mirror interface is bonded with Link Aggregation Protocol with other mirror ports. The bonded interface is configured for a dedicated Virtual Local Area Network, VLAN.

This solution was practically the only working method to capture all traffic going through the two Cisco 4500-X switches without issues in this case. Mirroring is performed from channel-group interfaces instead of a single interface because Cisco 4500-X did not allow port mirroring for single interfaces that were a member of a channel-group.

A dedicated computer was used for the traffic capture process. The traffic capture computer specifications are the following:

- CPU: Intel I7-4790 3600Mhz
- 4 cores with 2 threads in each core
- Memory: 32 GB
- 2x 10Gbit/s Intel X710 Ethernet Adapters

The traffic capture program was written in C-programming language and it utilizes Libpcap library for packet capture features and nDPI library for deep packet inspection processes which are used to identify the application of the flow. With the specs mentioned above, the capture system should be able to handle 28Gbps of data, if every single bit in the packet is being processed and multi-threading is utilized for the packet capture process. In practice, the implementation should be able to handle a lot more because mostly the headers of each packet are processed.

nDPI is an open-source and extensible deep packet inspection library. It allows application-layer detection for the packet capture processes, which does not depend on which port the application uses. It enables the detection of sub-protocols by using string-based matching, which utilizes Aho-Corasick algorithm for matching thousands of sub-strings efficiently. In case, the deep packet inspection process fails nDPI tries to guess the protocol by using a heuristic. nDPI can detect and classify up to 246 different protocols. (nDPI project, 2020)

In the traffic capture phase, the packet is captured and a traffic flow record is created. The flow record is created from IP, ICMP, TCP and UDP traffic. The flow record consists of the following features with size:

- Source IP address, 4 Bytes
- Destination IP address, 4 Bytes
- IP Protocol number, 1 Byte
- IP Identification, 2 Bytes
- Source MAC Address, 6 Bytes
- Destination MAC Address, 6 Bytes
- Ethertype, 2 Bytes
- TCP/UDP source port, 2 Bytes
- TCP/UDP destination port, 2 Bytes

- TCP Flags, 1 Byte
- Number of bytes, 4 Bytes
- Number of packets, 4 Bytes
- Application ID, 2 Bytes
- Unix timestamp of first packet, 4 Bytes
- Unix timestamp of last packet, 4 Bytes

The total size of each flow record is 48 Bytes. Depending on an installation, Ethernet layer features, such as the MAC addresses can be unnecessary in the flow record. After the creation of the flow record, it is pushed into a circular buffer to wait for the export to a NetFlow collector. In the case the buffer already included a record with the same IP-addresses and ports, the new flow record is summed with the existing ones stored in the buffer.

NetFlow exporter is executed in the dedicated thread. It combines flow records for efficient export to the collector. Exporter pops flow records from the circular buffer that has been stored in the buffer for a defined amount of time. The time is by default one second. This feature with the seek function is used to combine multiple packets of the same connection into a single record. This reduces the number of records exported towards the collector. The amount of time can be reduced to have a higher resolution of flow data and reducing the latency between the traffic capture and flow collector.

The seek function for circular buffer operates by seeking flow records that match with source and destination IP-address and ports. The seek starts from the tail of the circular buffer and when the head of the circular buffer has been reached the seek is ended. This reduces execution time especially in large buffers because only the used part of the buffer is searched. However, the current implementation of the seek function algorithm is not optimal in a case where circular buffer usage is high. This thing can be seen in tests which results are mentioned in chapter 9.1.

The developed capture system has few variables that can be tuned to make the capture system to fit the environment. These variables are the following:

- Flow record cache size
- Flow record cache expiration time
- Flow export rate

Flow record cache size defines the number of flow records that can be stored in the cache. By default, the size of the buffer is 10000 records. In practice, it means that there can be up to 10000 concurrent connections within the time specified in the flow record cache expiration time. The flow record cache expiration time variable defines the time how long the record stays in the buffer until it can be exported. Flow export rate defines the interval of how often flow records are sent to the collector.

Flow export rate defines that how often NetFlow exporter thread pops flows from the circular buffer for sending them to the collector. By default, the NetFlow exporter iterates every thousand microseconds and packs the flow records into optimal packet size, which is by default the same as Message Transfer Unit, MTU and sends them by using User datagram protocol. The value of MTU is by default 1500 Bytes.

The formula for the theoretical maximum amount of exported flow records per second is:

$$\frac{MTU}{Size\ of\ flow\ record} * \frac{1\ s}{Export\ Inteval\ (\mu s)}$$

Which is with default values:

$$\frac{1500\ Bytes}{48\ Bytes} * \frac{1000000 \mu s}{1000\ (\mu s)} = 31250 \text{Flow records / second}$$

This value can be improved in the limits of hardware by reducing the size of the record, stripping out unnecessary features, increasing exporter polling interval and increasing MTU of the network. Gigabit Ethernet connection should be able to deliver ~81k full-sized packets per second when MTU is set to 1500. (Cisco

Security, n.d.). This means that the theoretically Gigabit Ethernet allows transmitting 2.4 million flow records per second. However, other computational limitations may limit this number drastically.

Because NetFlow version 9 uses UDP transport layer protocol, high export rates would require an error-prone network or data path to ensure maximum integrity of the data. Also, the NetFlow collector must be implemented optimally to be able to receive and handle a high number of UDP datagrams and parse flow records from the datagrams efficiently.

At the collector end, in the case where template FlowSet does not exist, it waits for the exporter to send a template frame for the collector. Depending on the device this usually takes a few minutes. Unexceptionally the NetFlow exporter in the developed traffic capture system for the thesis purposes does not have a template export feature, which means that the template is hand-made on the collector side. When using the module with a standard NetFlow exporter, the automatic template collection process works correctly.

NetFlow version 9 protocol does not support any encryption which practically means that the NetFlow data travels as plain text through the network. The superior of the NetFlow Version 9 is IPFIX protocol, which provides encryption support and multiple other improvements such as variable-sized fields and enterprise-defined fields. IPFIX is an extended version of NetFlow v9 and it is standardized by IETF. (Zseby et al., n.d.)

The IPFIX protocol was not utilized in the project because the IPFIX records did not work in Cisco 4500-X which was the original plan for the source of the network traffic flow. After updating the 4500-X the IPFIX started to operate correctly, but the main problem of missing flow records caused by the VSS system was not fixed. Before the update, the NetFlow version 9 collector was already implemented.

Performance of the collector in the test environment could be improved by running two instances of the capture process, each for a single capture interface without the interface bonding. Running dedicated capture processes for each interface multiples the performance. Also, the Linux network stack with bonding and VLAN interface may cause a bottleneck for the capture process.

Another way to improve performance is to increase the number of worker threads for the packet capture process. Using multiple processes or threads may cause misordering in the flow records, but practically that should not be a problem for the data use case in this thesis.

An extra function in the packet capture process was also developed. The function is a DNS query-response pair parser and exporter. It exports DNS queries and answers in JSON format to the centralized data collector. This record contains the following variables:

- Querier IP
- DNS name
- Resolved IP address
- Nameserver
- Timestamp

This data is useful for determining the hostnames behind the IP-addresses by enriching this data with the flow record. However, the feature is limited to DNS A queries so the function is not capable for full DNS logging at this stage.

7.2 Microsoft event log forwarder

The event log of the Microsoft Windows Server Active Directory provides valuable information about the users of the network and different security events such as successful and unsuccessful logins on the workstations which are set up into the domain, but also for the devices, which logon and authentication processes are set up to use RADIUS protocol.

The practical implementation to forward Microsoft Server Event Logs tried to keep simple and follow best practices in communication security. The

implementation consists of PowerShell script which initiates TCP socket with TLS 1.2 context, reads new events from the security event log, converts it to JSON format and forwards them to the centralized log management system through the SSL protected socket.

In the centralized data collector, a simple Microsoft event log receiver over the SSL python module was implemented. Because the log was possible to convert directly into the JSON format in the PowerShell environment, no external processing was not required to do.

This solution is quite useful because the data channel is encrypted and the server is authenticated with an SSL certificate. The solution allows also a client certificate to be used to authenticate the client. This can be used to prevent external sources from spoofing messages into the log. The solution itself is quite universal and easily adaptable because it does not need any external programs to be installed into the Windows Server. The only thing that needed to run is the PowerShell script at the Windows Server.

7.3 Syslog server

A simple Syslog server module was implemented into the centralized log collection system. Practically It is a simple UDP listener that extracts the facility code and severity number out from the PRI number, which is located in the front of each message. After that, facility number, severity code, timestamp and message are converted into the JSON format and forwarded further in the process. However, parsers for each type of log messages are required to be implemented to use the data in the log messages directly in the processes.

The security of Syslog-protocol is quite poor because quite a high number of devices do not support TLS or DTLS features in their Syslog implementations. Syslog itself is quite a good source of data because it is widely used. Due to the lack of security features in Syslog implementations, those were not implemented for this server module in this project. The Syslog communications were put on a private subnet in the test environment.

7.4 SSH stream reader

The SSH stream reader was created for fetching logs from the servers and services which do not provide standardized log format and forwarding method. The solution is relatively simple, it opens an SSH connection into the server and starts reading a log file. Once a new line appears it is forwarded to the corresponding parser, which parses defined the entries of the line. From the entries, a JSON object is created and pushed into the database.

The solution is relatively usable because the SSH communications are strongly encrypted by default. SSH provides also authentication for communication.

7.5 Selection of database

NoSQL database MongoDB was chosen as the main database for all the collected data. Originally, it was decided to utilize the MariaDB SQL Server as the main database. During the project, it was noticed that the content of the data objects differs a lot from the others which would have required a more specific database schema design. MongoDB does not require a specific database design, because it creates the schema of the database dynamically.

The main differences between MySQL and MongoDB are the following:

- MySQL represents the data in tables and rows. MongoDB represents data as JSON Documents.
- MongoDB does not require the definition of the schema of the database
- JavaScript is used as Query Language in MongoDB, whereas MySQL uses Structured Query Language
- MongoDB can handle large unstructured data efficiently

(Guru99, n.d.)

The reason for handling large numbers of unstructured data and there is no need for schema definition makes the choice ideal for development projects using agile software development methods. It also integrates perfectly with the framework, because it uses JSON as a document format, which the framework uses as well. The model of how MongoDB operates also allows a flexible horizontal scaling of

the database, which is required if a completely new kind of data is stored in the database.

MongoDB Inc. defines the six best practices steps for improved security. These steps are the following:

- Create separate security credentials
- Use Role-Based Access Control (RBAC)
- Limit database connections
- Encryption the data
- Extra encryption for sensitive data
- Audit and logging

By default, MongoDB does not have authentication or other security features enabled. Role-based access control allows authorization for different roles such as an application server or a database administrator. Roles can be customized to fulfil the needs of particular teams and functional areas. Limiting connections to the database reduces the risk of intrusion and data theft. MongoDB Inc. defines the white list of allowed IP-address which are allowed to connect into the database. (MongoDB Inc, 2020)

Data encryption makes data unreadable for users who do not use keys. The Community version of MongoDB does not support encryption. Alongside the encryption, MongoDB supports client-side field-level encryption that ensures only relevant parties have access to read. The audit log keeps track of database operations and changes to answer questions about what changes and when those changes were done. The enterprise edition of MongoDB enables extra features such as LDAP (Lightweight Directory Access Protocol) and Kerberos authentication. (MongoDB Inc, 2020)

7.6 Risk management of the data collection

The first development iteration was mostly prototyping the system. Because of this, the used solutions may include risks that must be taken when the system is taken into production. This chapter introduces the initial risk management for some recognized risks. Table 2 represents some of the recognized risks, consequences, controls and mitigation methods.

Feature	Risk	Consequences	Control & Detection	Mitigation
Syslog message collector	MiTM attack	Attacker gains visibility of the system logs in realtime. Attacker has possibility to inject spoofed log messages into the data collector.	Syslog messages communicates trough private network. Monitor activity of this network	Enable encryption and authentication for the communication. If device or system does not support them, provide secure tunnel interface for the communications. Mitigate MiTM attack vectors
	Hijack of device sending Syslog	Attacker has possibility to inject spoofed log messages into the data collector.	Monitor activity and detect anomalies in the log events	Enable encryption and authentication for the Syslog communication. If device or system does not support them, provide secure tunnel interface for the communications. Mitigate MiTM attack vectors
	Communication gets disrupted or blocked	Log messages are not received at the data collector. Causes blind spot of the specific data source	Monitor data collector activity and alert if activity is reducted.	Cache messages in the device until collector acknowledges that the log messages have arrived. Enable redundant communication channel
NetFlowv9 Collector and traffic capture appliance	MiTM attack or Hijack of traffic capture device	Attacker gains visibility of all network flows. Attacker has possibility to spoof flow records to obfuscate the security team	Use dedicated nic and network for the data transmit to the collector. Monitor activity of the collector and detect anomal network activities	Change protocol to IPFIX and enable authentication and encryption for the flow record. If this is not possible, provide encrypted tunnel interface for the communications. Mitigate MiTM attack vector
	Communication or traffic capture gets disrupted or blocked	Immediate blind spot in the data collection. Intrusion is not detected. Increased Latency in collected data	Monitor data collector activity and alert if activity is reducted from the normal.	Enable redundant communication channel. Cache flow records for specific amount of in order to fetch them later when the communication operates again.
Microsoft Event Log Collector with TLS	MiTM attack or Hijack of traffic capture device			
	Communication or traffic capture gets disrupted or blocked	Microsoft events not getting transmitted	Monitor data collector activity and alert if activity is reducted from the normal.	Enable redundant communication channel.
Database	Data breach	Outsider has all the collected data	Detect abnormal traffic from the database	Limit the database connections only for the IP addresses that is require. Encrypt the sensitive data
	Database corruption or hard disk crash	The data is probably lost	Keep track on disk usage,activity and health.	Use redundant disk arrays or hyper converged file system

Table 2: Table of the risk management plan of the data collection.

8 IMPLEMENTED MACHINE LEARNING AND ENRICHMENT FEATURES

During the thesis, there were few attempts to implement machine learning features to process the data automatically and find unseen things from the collected data. Two of them, decision tree-based anomaly traffic detection and regular traffic detection modules were successfully tested and the methods were proved. One artificial neural network-based traffic classification was only tested as a proof of concept, but the result of it did not offer anything useful at that time.

8.1 Network behaviour model and anomaly detection module

An anomaly traffic detection module was created to detect anomalies from the change of network behaviour by comparing them to the pre-built model of the traffic. A decision tree-based supervised learning method is used to generate the model from the flow records of the network traffic. The model of the structure is hierarchical, which allows hierarchical comparison between the data and the model. The solution allows optimal comparison against the model which improves the performance of the detection process. High performance of the comparison process is important because the amount of incoming traffic to the detection module can be high which may lead to an increased latency of the detection process.

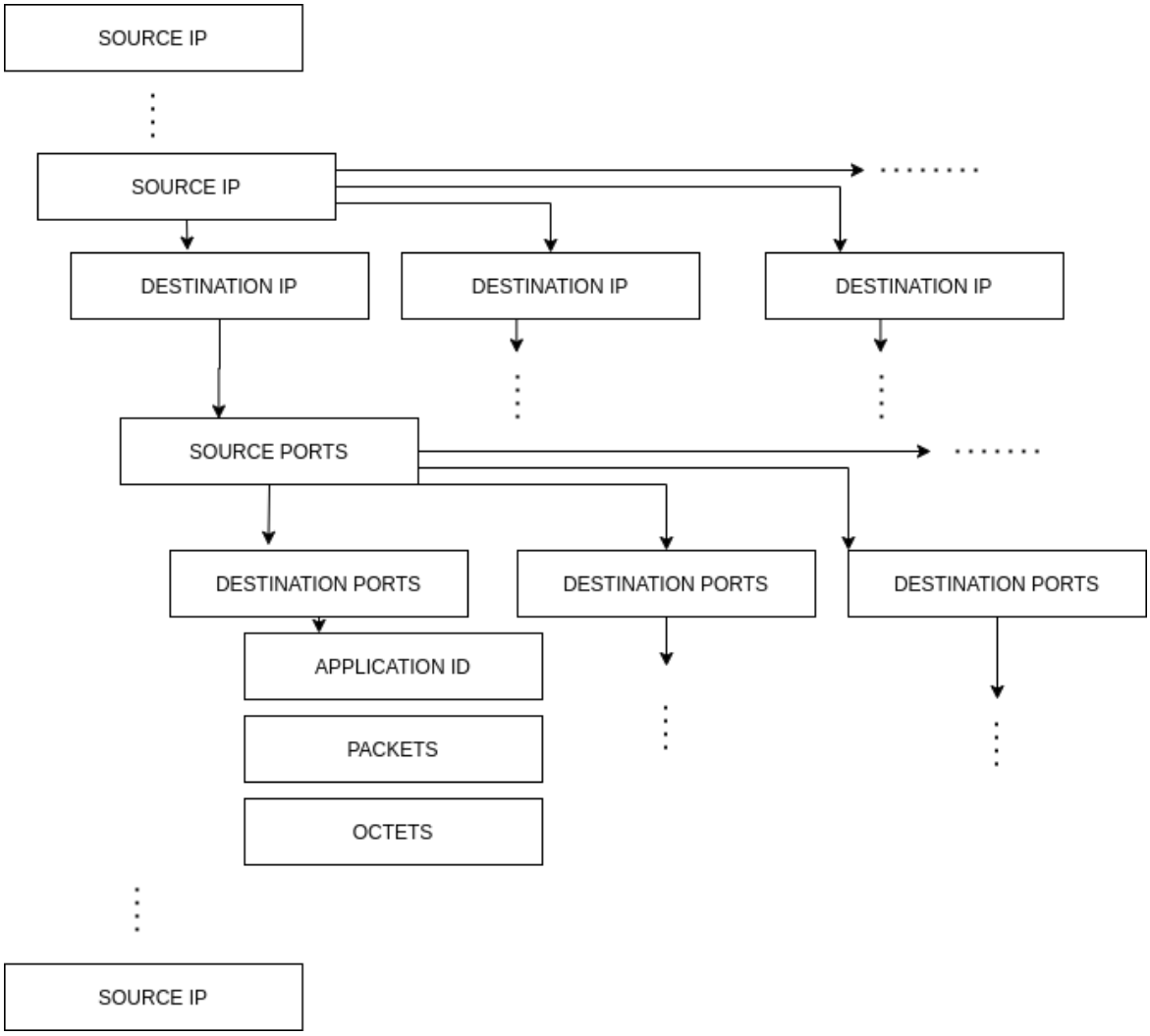

Figure 9: The hierarchical traffic model of the traffic record.

In figure 9, the content of the model is explained. The model is a simple hierarchical list of traffic flows from the Source IP to the Destination IP with their corresponding source port and destination ports. At the bottom of the hierarchy, the application identification number and number of packets and bytes are stored. The comparison of the incoming flow against the hierarchical model is based on a decision tree algorithm. Figure 10 present an algorithm for the comparison process.

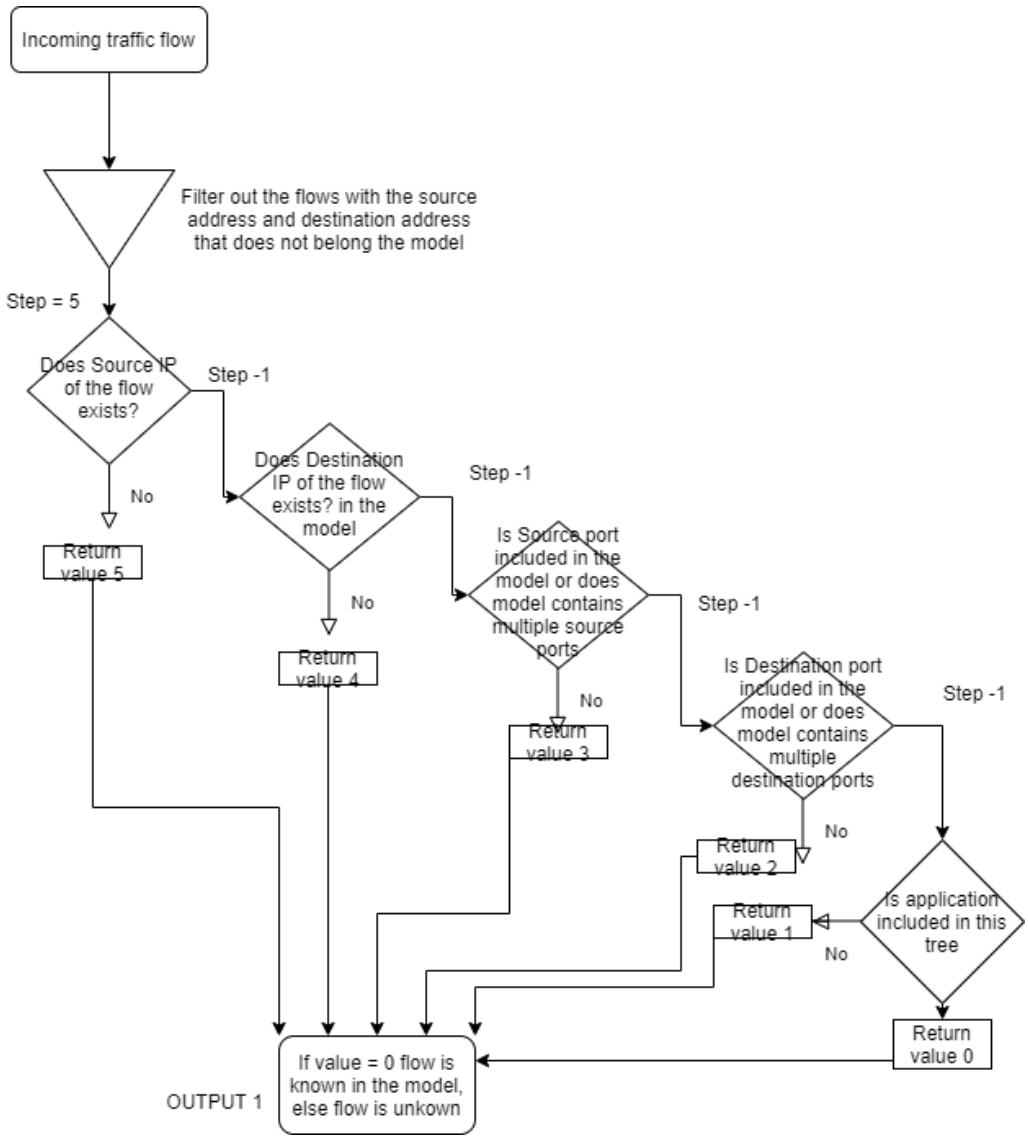

Figure 10: Simplified Logical presentation of the comparison algorithm.

The algorithm is relatively simple. As seen in figure 10, the tree provides information of the reason via return values alongside the information was the flow anomalous or not. The return values can be defined as the following:

- 0: The flow had no significant difference against the model
- 1: The application was different than in the model.
- 2: The destination port was different and the model contained only one destination port. Did the direction of the connection change?
- 3: The source port was different and the model contained only one source port. Did the direction of the connection change?
- 4: The destination IP where Source IP sent the packet, did not exist in the model. A new connection to a new destination was caused by the source.
- 5: The source IP did not exist in the tree. A completely new source of traffic appeared.

In practice, return values 3 and 2 should never appear unless there is a change in the direction of the connection. In case the return value is 1 there was a change in the application. This condition triggers in the case where the destination runs HTTP server, but the source uses a telnet client without headers or performs TCP SYN scan against the port for an example.

8.2 The regular traffic detection module

The regular traffic detection module was developed to track and detect regular or periodic connections from the flow records. This module is used to find possible command and control channels that are used by malware. This is based on an automatic observation of traffic flows with the same source and destination and time between them. The method requires at least three flow records to determine the flow interval. This interval value can be used to predict the next occurrence of the flow.

In figure 11, the operation of the module is shown. It practically keeps track of flow intervals when three or more flows with the same time difference between the flows it calls a callback function for further processing and saves it to an output record for post-analysis.

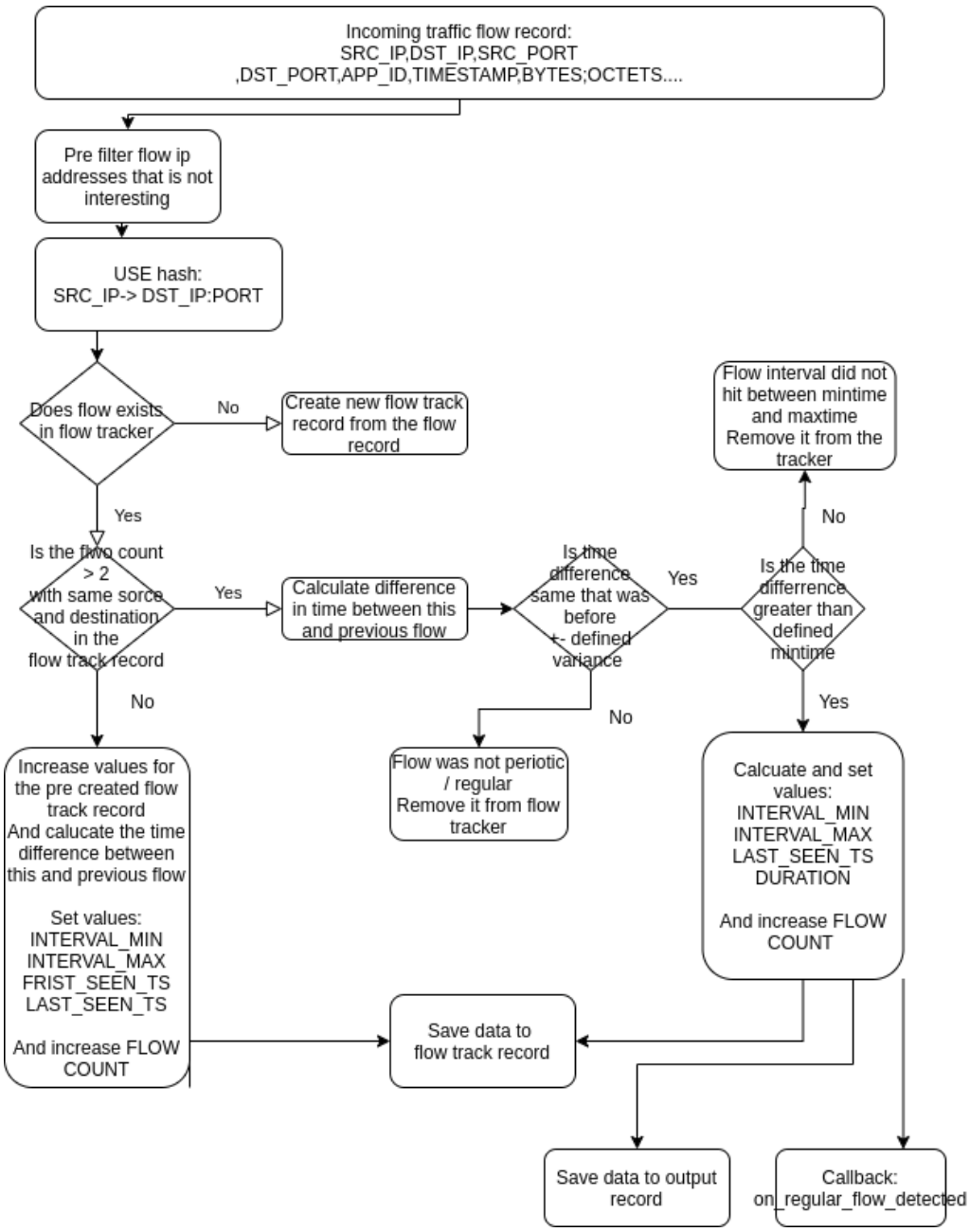

Figure 11: The simplified presentation of the operation of the module.

The regular traffic detection module takes four definable variables inside which are the following:

- Minimum allowed interval
- Maximum allowed interval
- Maximum allowed variance interval
- Minimum required amount of flows

Allowed interval variables practically filter out unwanted results with the specific interval from the output. The variance variable defines the amount allowed jitter in the interval value. The minimum amount of required amount of flows variable defines a number of the flows that are required for determining if a specific flow regular. The minimum value for this variable is three. The regular traffic detection module outputs the following variables alongside variables defined in the flow record:

- Minimum interval
- Maximum interval
- First occurrence time
- Last occurrence time
- Flow duration
- Total bytes
- Total packets
- Predicted next occurrence time of the flow

Figure 12 presents the detection method in a time sequence chart. Green transparent bars present the detected regular flows caused by the flow records of a possible malware beacon. The transparent blue bar presents detected regularity in the WWW-browsing 1. Both of the bars visualized the prediction of the next occurrence time of the flow where small variance in the flow next appear time is allowed.

The small blue transparent bar visualizes a situation, where the flow was predicted to appear, but it did not. In this case, the regularity of WWW-browsing 1 stopped. This is typical behaviour for the traffic where a user loads a web page and some kind of tracker or an API connection stays active until the user closes the web page. This is a typical cause of false-positive results for regular traffic detection methods like this. The minimum interval was set to three seconds in this example, which filters out the transmission caused by WWW-browsing 3 and 2.

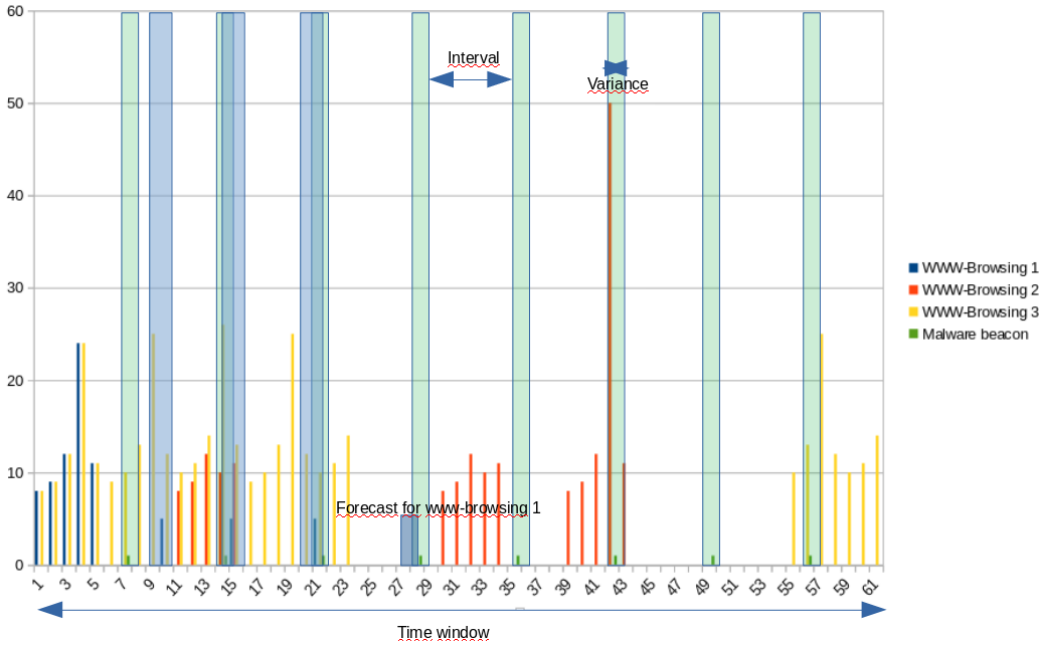

Figure 12: An example of data shown from one minute. X-axis = Time, Y-axis is the amount of data transmitted.

Depending on the environment where this method is used, the method causes a lot of false-positive results caused by protocols such as Network Time Protocol and the behaviour of applications and services such as Web-trackers, Windows Update Services or cloud applications such as Microsoft Teams.

The hour of flow records may cause thousands of results when those are driven through this process for example. When the minimum time is set relatively low, under ten seconds, for example, the amount of false-positive results increases because a lot of normal traffic starts to hit against the detection rule. In the case where the allowed variance is set to high, to minutes, for example, more false-positive results are caused by activities such as random web browsing.

For the false positive problem, some filtration for the results can be done and the amount of the results might be reduced significantly. Some applications and ports such as Network Time Protocol and port 123 causing false-positive results can be filtered out easily for example. Also, the results that are not present in the whole given time window could be reduced.

The process of the detection module give also a useful method to detect if legitimate but regular connections such as SNMP poll or periodic data transmission from IoT Device communication suddenly disappear or change their transmit interval. These events might mean that the device is broken or it is malfunctioning. In case it comes back online, it might mean that the device has been stolen and the software or the content of it has been modified.

8.3 Automatic data enrichment module

The prototype version of the automatic data enrichment module was implemented to find relations between multiple JSON objects. The enrichment method works similarly with the Laravel Eloquent Relationships. Laravel Eloquent Relationships method provides a powerful method for chaining and querying different objects together. (Laravel LCC, 2020). Relational database systems, such as MySQL provides similar functionality with JOIN commands.

The operation of this module is relatively simple. It loops through the object keys and compares them into values in other objects and creates links between the objects. The process creates a new object which can be iterated again through the process.

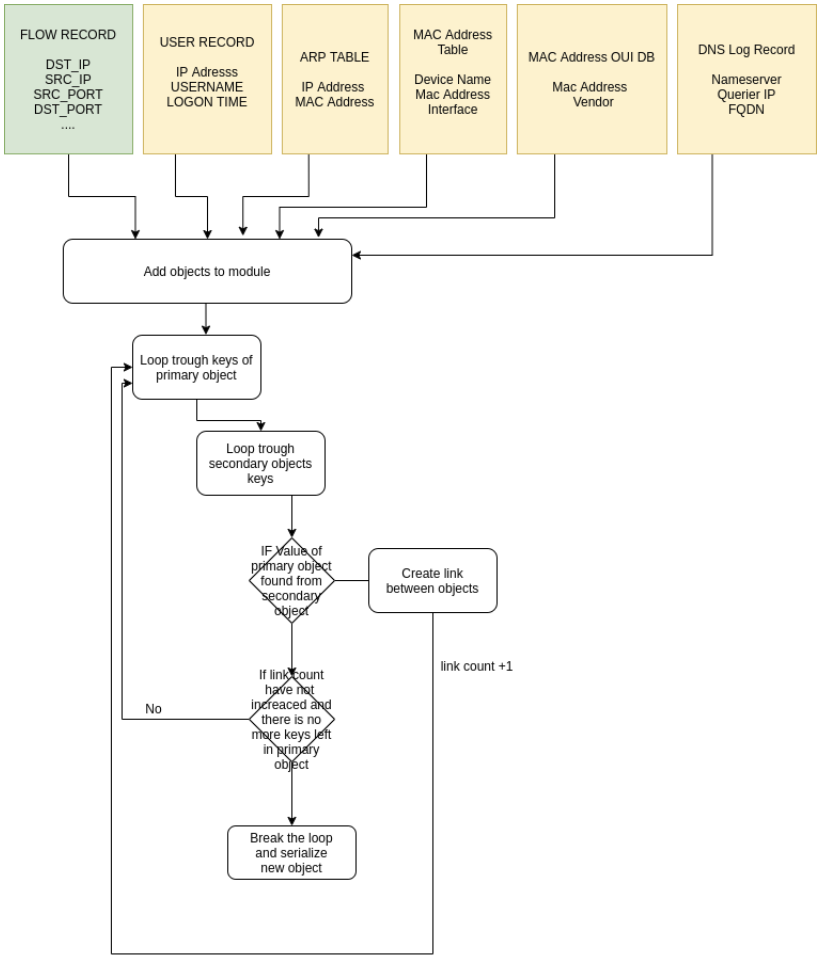

Figure 13: Operation of the process in a simple format. The Green box is the primary object and the yellow ones are secondary objects.

The process shown in figure 13 can be run in sequential iterations where previously linked objects are included in the primary object.

In figure 14, the example result of the enrichment process is presented in a graphical format. In this example, a flow record is set as the primary object. In the first enrichment iteration, a username and MAC address is found from the records by comparing a flow Source IP address to other objects. By comparing Destination IP address to various objects, the information of the Geological location, DNS name, a Network block owner and persistence in Open Threat exchange could be found.

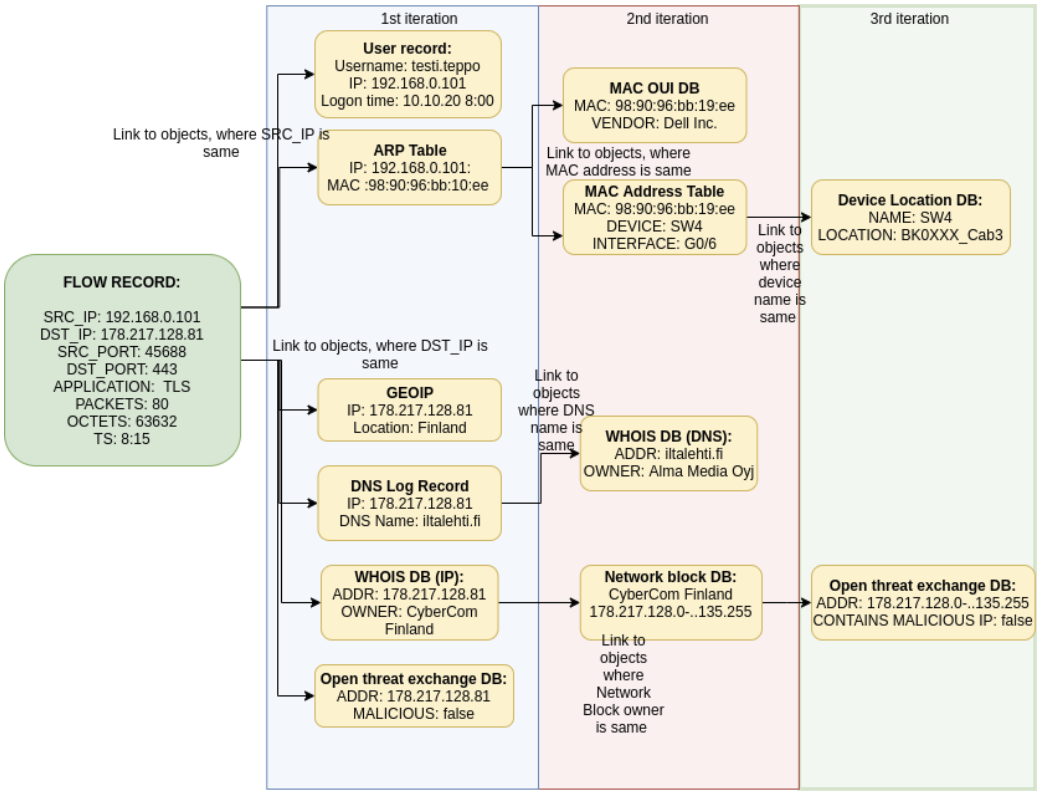

Figure 14: Simulated hierarchical result of the enrichment process.

During the second and the third iterations, newly found variables are iterated again against the same objects and new information about the source IP is found: Vendor of the workstation, the network equipment where the device is connected and its interface. On the destination IP side, information about the owner of the DNS name and information of the network block for an example.

The process could be started from anywhere. Setting a user record as the primary object, the output contains everything related to the user: IP addresses, accessed websites and used devices for example. For another example where DNS log record is set as the primary object, from the output, which users have accessed into this website related to the DNS log record can be determined.

The solution for this case is not free from problems. Without filtering or giving rules for skipping objects or variables the algorithm starts to loop for a long time and the hierarchical result is massive containing a lot of information that may not be relevant to the case. For example, if a flow record is compared with another flow records, the created link will be found between all of them, because there

are multiple records, containing the same ports, the same addresses and the same transmit time for example. The process is computationally quite heavy, especially if multiple iterations are used and a volume on the input side is high.

8.4 Flow classification with artificial neural network

Traffic flow classification with artificial neural networks was tested for the further implementation of the traffic behaviour classification process. Due to the time taken in model compilations, the complete implementation of this process was moved to the next development cycle. The method uses supervised learning with predefined datasets. The process automatically classifies the flow behaviour to categories such as:

- HTTP Large activity
- HTTP Small activity
- SSH small activity
- SSH large activity
- SNMP poll
- ICMP echo/reply

The test implementation of flow classification by using an artificial neural network was created with Keras library, which is a versatile and simple deep learning module for Python programming language. Keras library is a user-friendly interface for the TensorFlow library.

src port	dst port	octets	packets	app_id	output
49141	22	120	2	(ssh)	1 (SSH small activity, input)
49141	22	140	2	(ssh)	1 (SSH small activity, input)
49141	22	400	3	(ssh)	1 (SSH small activity, input)
22	49141	500	3	(ssh)	2 (SSH small activity, output)
22	49141	620	2	(ssh)	2 (SSH small activity, output)
22	49141	700	3	(ssh)	2 (SSH small activity, output)
48134	443	6000	5	(tls)	3 (TLS small activity, input)
48134	443	12498	8	(tls)	3 (TLS small activity, input)
48134	443	13311	10	(tls)	3 (TLS small activity, input)
443	48134	64234	80	(tls)	4 (TLS large activity, output)
443	48134	341241	50	(tls)	4 (TLS large activity, output)
443	48134	123141	30	(tls)	4 (TLS large activity, output)

Table 3: Short example of contents of dataset

Table 3 presents a short example of learn dataset with five inputs at the input layer and four output values at the output layer. The actual size of the dataset was much larger, containing thousands of rows. The input layer consists of source and destination ports, the number of octets and packets transmitted and the application id.

The type of model shown in figure 15 is sequential. The hidden layer contains a dense model with seven nodes in the hidden layer using the rectifier function as the activation function. Categorical cross-entropy was used as the loss function. With forty iterations the model accuracy was around 82.8%.

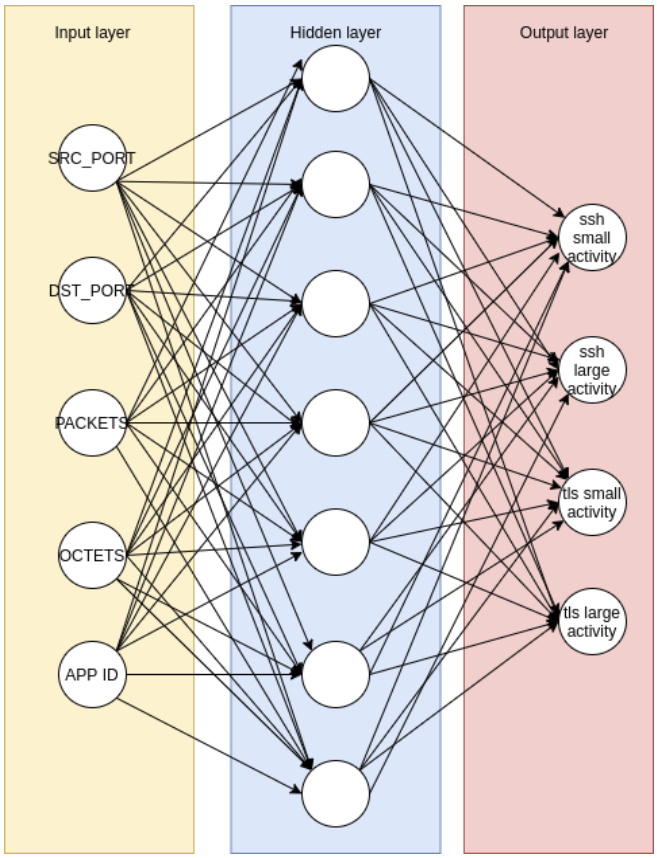

Figure 15: Example graphical presentation of used model of artificial neural network.

The output classification is practically the classification of the behaviour of the flow. These behaviour classifications are linked to the source to destination combo of traffic flow during a learning period. This leads to flow such as

192.168.100.101 to 192.168.100.105 gets the flag of SSH small activity classification for an example.

After the learning period, a flow of the source to the destination combo is predicted towards the model and in the case, the prediction output, the classification of the flow, is not included in the record of the source to destination, an alarm is raised because of unexceptionally behaviour.

The traffic flow classification could also be used with the method present in the regular traffic detector module. The combo of the output of these methods could be useful to determine the type of flow and monitor the anomalies in that. Because of the time generating the training dataset and compiling the model more detailed.

9 SYSTEM AND FEATURE BENCHMARK TESTS

In this chapter, some benchmark tests are performed for evaluating the implemented system and its features. The test environment, ICTLAB, consists of:

- 60 Users (COVID-19 situation limits users at the lab environment)
    - In normal situations, the number is around 180 different users in one week.

  150 Workstations, internal services and servers
- Firewall
- Enterprise Local Area Network, with 10+ switches
- Microsoft AD as a Domain Controller

Due to the privacy of the other users in the ICTLAB network DNS Logging and Active Directory Security Event Logging were disabled during the tests. Those two sources would give a good view of what users are doing in the network but it is not the main purpose of these tests. Enriching IP addresses with DNS record from the DNS log and enriching the user information from the Event log gives a good sight of who is accessing a specific website site and when for example.

Due to privacy and security reasons, the dataset with IP-addresses is not shown in this report. Contents of the datasets are only meant for descriptive use.

9.1 Stress test of data collection

In this test, the system was shortly proofed against a high number of events and collection time and accuracy were measured. The tests were performed with two dedicated servers with 20Gbit/s connectivity communicating through the network. The NetFlow collector was used as the collector because it has the highest volume when compared to other collectors.

During the tests, 50000 echo/reply packets were sent in flood mode from the server to another server to detect possible packet loss at the collector. Each of the saturation was executing one minute before sending the test packets.

9.1.1 Test 1: Saturating the collector with large transmission

In this test, it was tried to saturate the network interfaces of the packet capture device with large file transmission through the network. Two connections between the mentioned servers totalling up to 14.9Gbit/s of network traffic. The traffic was generated with the Iperf3 application by generating two parallel TCP streams between two servers.

During the test, there was no noticeable change in the CPU usage of the packet capture process. Amount of the captured packets varied from 60k to 75k packets during the test. On the collector side, there was no change in collector latency and there was no increase in the flow rate. The detected packet loss of the ICMP echo reply test was 0%.

9.1.2 Test 2: Saturating the collector with a high number of packets from a high number of connections

In this test, it was tried to saturate the network interfaces of the packet capture device with a high number of concurrent connections. The traffic simulating concurrent connections was generated by using Hping3 application which was initialized to flood TCP packets to the destination from random source IP-addresses generating up to 250 000 packets per second.

On the collector side, the amount of received flow rate was increased to ~2000 flows per second which is a relatively too low amount when compared with the packet incoming rate. None of the test packets was detected on the collector side. Because of this behaviour, a more detailed test was performed to find the reason for this problem.

The same test was repeated with different size of the circular buffer for stored flow records. The size of the buffer was changed in every test iteration to detect that does the size of the buffer affects the packet loss rate. Flow cache expiration timeout was also reduced from 1000 milliseconds to 200 milliseconds. Table 4 presents the packet capture result of the test with different sizes in flow record buffer.

Circular Buffer Size	AVG Measured Capture packet rate	Flow export rate	Latency (sec)	AVG Measured Loss %
100000	12154	6457	3	97
50000	13173	7121	2	91
25000	25127	7502	4	82
12500	39625	6369	2	79
6000	63201	6402	1	74
3000	98707	8004	1	60
2000	104275	6382	1	52
1500	119411	6400	0	45
1000	152146	5000	0	44
768	170183	3800	0	60
500	184244	2400	0	72
256	183455	1200	0	80
128	191214	650	1	95

Table 4: The result of the same test with different buffer sizes.

Referring to table 4, the packet capture rate efficiency with different buffer sizes can be seen. A smaller buffer size allows a higher number of incoming packets. Buffer sizes from 1000 to 3000 allow the lowest loss in the detection in the situation where there are multiple connections flooding packets. From 100000 to 2000 buffer sizes the loss was caused by packet loss at the capture interfaces. The packet loss was caused by the circular buffer seek function which execution takes a relatively long time in the case the buffer is heavily populated with

records. This behaviour practically blocks the capture until the seek function is executed completely.

The loss that was caused in the situations where buffer sizes were under 2000 was caused by the fact that there was no room for all flow records in the buffer. This also explains the slow down at the flow export rate.

In the next test, the highest amount of packets in different connections was found to find the limits of the capture system. The flow record buffer was set to 10000 to have enough room for the flow records. The amount of sent packets was raised in steps. The results of this test can be seen in table 5.

Ingress packet count (different connections)	AVG Flow export rate (records)	Latency (sec)	AVG Measured Loss %
4104	640	0	0
6200	1437	0	0
7100	2044	0	0
9120	3982	1	0
19878	6521	2	0
24512	7995	30	10

Table 5: collector test results with a high number of connections where the packet count was raised slowly.

The result of the test shows that the current system can handle up to 20000 packets from different connections with stable latency. Packet counts over 20000 caused a slow continuous rise in latency, which was caused when the flow record buffer started to fill up. The measured loss was caused when the flow record buffer was filled up and there was no room for new flows.

9.1.3 Test 3: Saturating the collector with a high number of packets from a single connection

Due to the results of the previous collector saturation tests, the behaviour of the collector was tested with a high number of packets from a single connection. The

test was performed with the same parameters as in test number two. Table 6 presents the results of this test. The result of the test shows that the high packet count is not itself the reason for the loss in the flow records.

Ingress packet count (single connection)	AVG Flow export rate (records)	Latency (sec)	AVG Measured Loss %
5840	450	0	0
17421	471	0	0
26312	441	0	0
112105	434	0	0
144145	432	0	0
220211	436	0	0

Table 6: collector test results with a single connection and a high number of packets.

9.2 Anomaly detection in network behaviour tests

In this test, the network behaviour anomaly detection module was tested in two network environments. The test was run for a month to detect a possible amount of false-positive results. The detection was tested by opening random connections to the host in the network where the model was applied. This simulates the unwanted anomalous behaviour that happens in the network.

The model was first applied to a network management network that contains the management interfaces of the network equipment and some operational technology of the test environment. The model was built from one week of traffic flow records coming in and out from the network management network. Traffic in the management network is known and there should not be anything anomalous. After one month of testing and after filtering out known management traffic caused by me, the results showed only the anomalies that were generated as a test.

The same method was tested by creating a model from one of the classroom traffic going in and out from the network and comparing traffic related to the classroom network against the model. After one hour of testing, there were nearly seven hundred false-positive results which were caused by the workstation within the network. The test was cancelled after that.

9.3 Regular traffic detection

Because of the large amount of malware command and control channel beacons to home regularly. In this test, the performance of regular traffic detection was tested in a few different cases.

A Kali Linux with Cobalt Strike application and normal Windows 7 machines were placed in the ICTLAB network. The Kali Linux simulates the attacker and Windows 7 the victim. The malware was generated in the cobalt strike and it was injected into the victim machine. Both of the machines were set up into the Virtual Laboratory System which allows the remote use of the machines and possible isolation from the ICTLAB network.

The test goal was to find a command and control channel of the malware that was generated with Cobalt Strike and injected into Windows 7 in different situations. The test utilizes the implement regular traffic detection python module, flows per host gauge and simulated decision trees.

Test numbers four and five were performed first because the beacon was running already with the correct parameters and the system had gathered data for the testing. The numbers four and five also includes the more detailed steps of the analysis and filtration process. The rest of the tests were performed with similar methods. The only difference in the quality of the output of the regular traffic detector module was the lack of the output of application detection.

9.3.1 Test 1: Malware beacon interval 60 seconds, jitter 0%

In this test, a beacon call home interval value was set to one minute and jitter was set to 0%. The test simulated the case where malware was active and it was communicating with its host.

In step one, 1 hour of NetFlow data was fetched from the database and regular traffic detection was performed in a regular traffic detection module. Fetching and processing one hour of NetFlow data took 19 seconds which contained 465667 flow records.

The parameters for the module were set to the following:

- Minimum interval 10 sec
- Maximum variance between intervals 2 seconds

With these settings, 1283 regular traffic flows were detected. A relatively small minimum interval caused a lot of false-positive results because of a regular web browsing activity, WWW-trackers and protocols such as SNMP, ICMP and NTP for example.

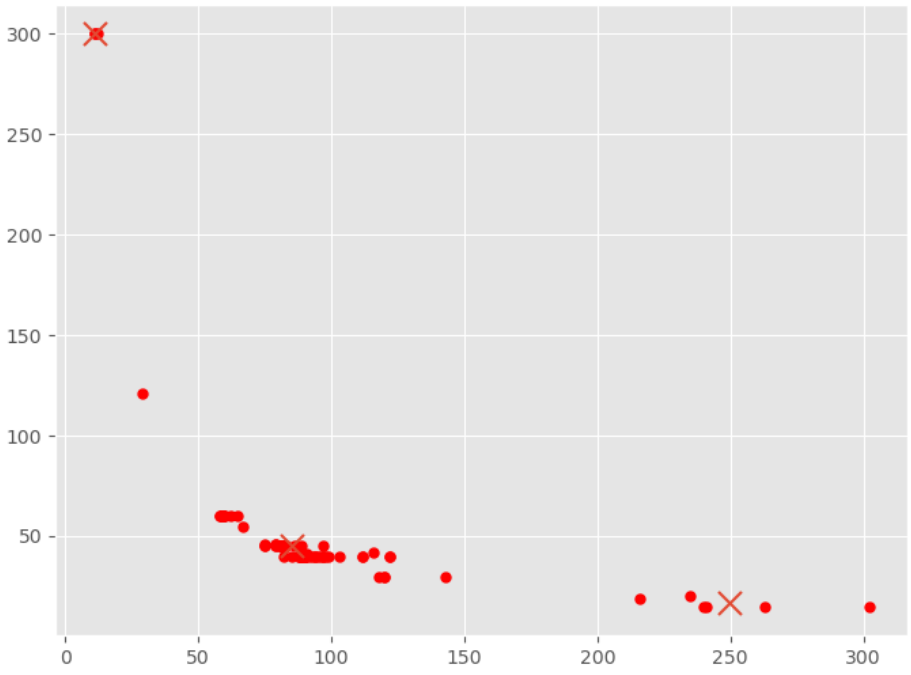

Figure 16: Scatter of all detected regular flows. X = Amount of flows Y = Interval. 1283 Data points.

In figure 16, the scatter of regular flows in a given time window, a cluster of regular flows can be seen at the point or near where also the command and control channel of the malware is located. It is possible to perform some filtration for the data. Cross marks in the graph present the centre of the cluster of the multiple points. The mark does not have any actual meaning in these tests.

In the next step, all traffic flows that were not present in the whole time window were removed. These flows were caused by transient WWW-browsing for example. Also, the port filter was enabled to filter all ports, except 53(DNS), 80(HTTP) and 443(TLS/HTTPS) away from the dataset.

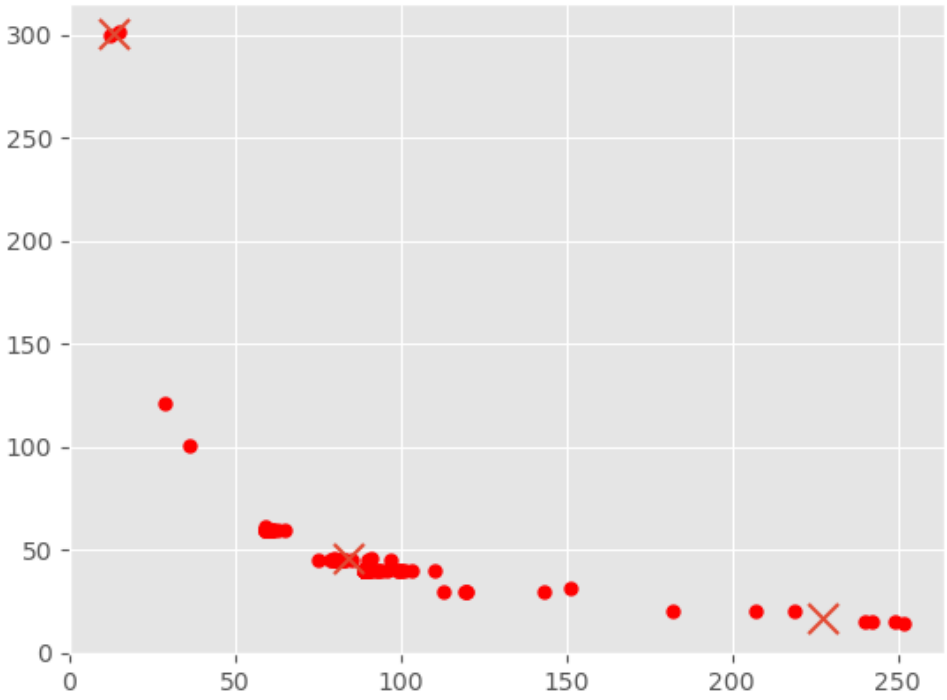

Figure 17: Scatter of regular flow after filtration. X = Amount of flows Y = Interval. 111 Datapoints.

The filtration did not remove the noise around the command and control channel significantly. After a more detailed look at the data, most of the noise was caused by Microsoft 365, Microsoft Teams, Windows Telemetry Services and Skype which all are used in the workstations located in the environment. Due to the lack of an automatic filtration of IP-addresses with a good reputation, those cannot be automatically filtered away in this stage. By filtering them out manually the noise disappears around the command and control channel.

The same test was also performed with 12 and 24 hours of flow records. With 12 hours, the dataset size after filtration was reduced from the previous 111 data points to 36 data points. The 24 hours reduced the dataset size to 34 data points.

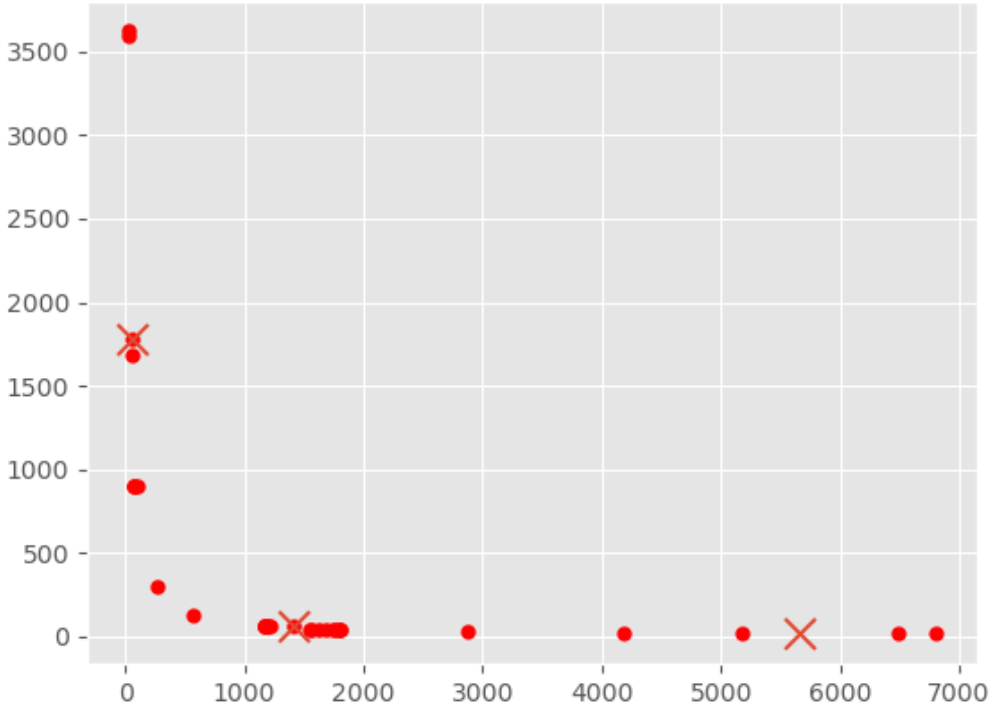

Figure 18: Scatter of all filtered regular flows generated from 24 hours of flow records. X = Amount of flows Y = Interval. 34 Datapoints.

A larger time window reduced the noise generated by Microsoft services in this window because some of the workstations have been powered off during the night time. However, the automatic determining is the flow caused by malware or not, is not possible without automatic data enrichment processes and closer observation of the results.

Figure 19 shows that there were similar regular flows when compared with the real command and control channel. The row marked with red text was the traffic caused by the command and control channel of the malware. The rows in the greyed area were proved to be false-positive results because of jitter. The variance was allowed to be two seconds from the calculated interval. Regular traffic flows with large jitter could be filtered away, because only the regular flows with the accurate intervals were wanted to be found.

	Flowcount	Interval	Jitter	Appear % in timewindow	INTERVAL_MIN	INTERVAL_MAX	DURATION
443(TLS)	1169	60	0	100	60	60	70200
80(UNKNOWN)	1170	60	0	100	60	60	70200
9:443(UNKNOWN)	1169	60	0	100	60	60	70200
.82:443(TLS)	77	900	0	99	900	900	69300
05:443(UNKNOWN)	1559	45	1	100	46	45	70243
2:443(MICROSOFT)	1755	40	1	100	41	40	70203
:80(HTTP)	1744	40	1	100	41	40	70223
:443(TLS)	1182	60	1	101	61	60	70199
7:80(HTTP)	1221	60	1	105	61	60	70201
.163:443(GOOGLE)	577	121	1	100	121	120	70118
:443(UNKNOWN)	264	300	1	113	301	300	69953
.82:443(TLS)	79	900	1	102	900	899	70200
53(DNS)	26	3600	1	134	3601	3600	64800
.203:443(UNKNOWN)	1798	40	2	103	41	39	70251
.201:443(UNKNOWN)	1680	40	2	96	41	39	66353
9:443(UNKNOWN)	1170	60	2	100	61	59	70201
6:443(UNKNOWN)	1169	60	2	100	61	59	70143
:443(UNKNOWN)	1165	60	2	100	61	59	69902
6.189:443(GOOGLE)	5170	15	2	111	15	13	69338
:53(DNS)	21	3623	2	109	3624	3622	69777
.142:443(GOOGLE_DOCS)	6490	15	4	139	15	11	69942
1:443(OFFICE_365)	4177	20	10	119	21	11	70099
9:443(TLS)	1209	60	16	104	61	45	70249
:53(DNS)	6810	14	19	136	30	11	66790
2:443(SKYPE)	1789	40	21	102	41	20	70208
6:443(TLS)	1545	45	31	99	61	30	70192
.140:443(TLS)	1618	45	31	104	47	16	69296
:53(OFFICE_365)	1798	45	33	116	46	13	60944
1:443(OFFICE_365)	2878	30	35	123	46	11	70251
6:443(TLS)	1414	60	38	121	61	23	70253
9:443(TLS)	55	1680	841	132	1681	840	58142
:443(TLS)	101	900	855	130	900	45	52222
:53(STEAM)	73	900	870	94	900	30	32045
3:53(APPLE)	51	1778	1186	130	1778	592	62420

Figure 19: Datasets from 24 hours of data after all filters has been applied.

Application detection has worked fine for some flows. The records that the application has determined as unknown were manually checked and those belonged to some of the previously mentioned Microsoft services. The reason why the application has not been recognized is that the flow has been ongoing for a long time and nDPI library can't determine the application in the middle of the TLS conservation because the content of it is encrypted.

9.3.2 Test 2: Malware beacon interval 60 seconds, jitter 99%

In this test, the call home interval of the beacon was 1 minute and jitter was set to 99%. The test simulated the case where malware is active and it is communicating with its host with a large jitter to obfuscate the detector. The test is similar to test 1.

Parameters for the module were set to the following:

- Minimum interval 0 sec
- The maximum variance between intervals of 100 seconds

Figure 20 shows the results after the same filter that was used in test number one.

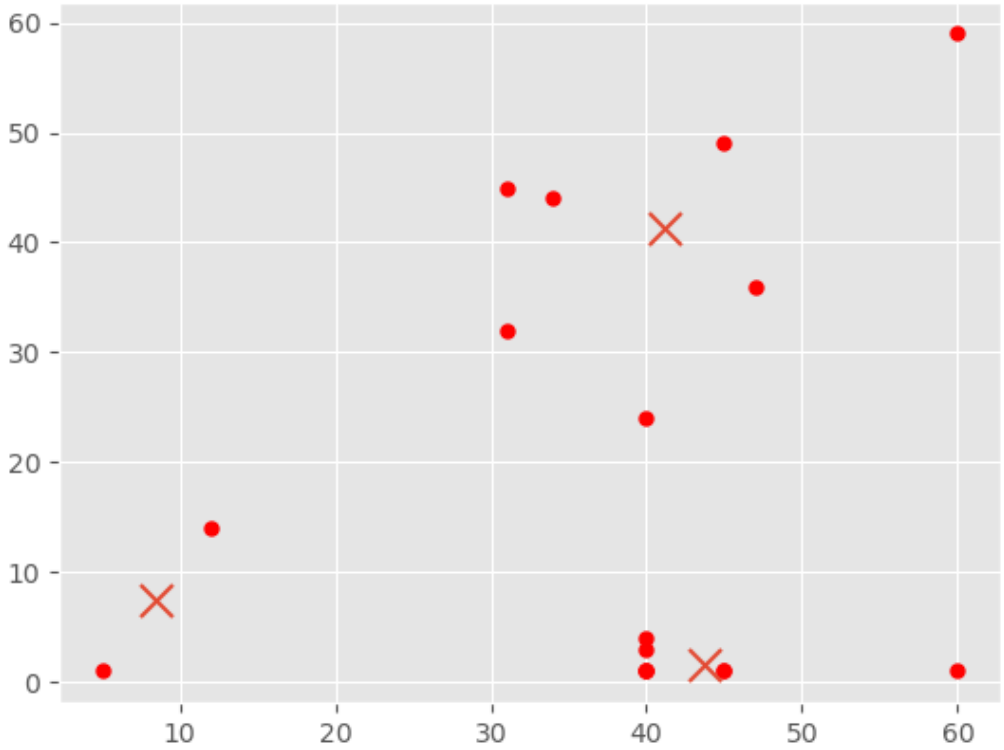

Figure 20: Scatter of all filtered regular flows generated from 1 hour of flow records. X = Interval Y = Jitter. 34 Datapoints.

In this case, the axis in the scatter in figure 20 was set to be Interval and Jitter, because in this test it was looked to find regular flow which jitters. As can be seen, there are only several flows which jitter is more than 10 seconds. Figure 21 shows the content of the filtered datasets.

A	B	C	D	E	F	G	H
	Flowcount	Interval	Jitter	Appear % in timewindow	INTERVAL_MIN	INTERVAL_MAX	DURATION
43(TLS)	65	60	59	108	60	1	3563
0:443(UNKNOWN)	91	45	49	114	50	1	3571
443(TLS)	147	31	45	127	46	1	3583
443(OFFICE_365)	112	34	44	106	45	1	3560
7:443(UNKNOWN)	91	47	36	119	49	13	3555
80(HTTP)	87	31	32	75	60	28	3573
1:443(UNKNOWN)	89	40	24	99	41	17	3540
2:443(GOOGLE_DOCS)	288	12	14	96	15	1	3588
6:443(UNKNOWN)	89	40	4	99	41	37	3560
6:443(UNKNOWN)	89	40	3	99	41	38	3560
OGLE)	718	5	1	100	6	5	3594
:443(UNKNOWN)	89	40	1	99	41	40	3566
443(UNKNOWN)	89	40	1	99	41	40	3564
:443(UNKNOWN)	79	45	1	99	46	45	3559
443(UNKNOWN)	79	45	1	99	46	45	3556
:443(UNKNOWN)	59	60	1	98	61	60	3553
)(HTTP)	88	40	1	98	41	40	3543
JNKNOWN)	720	5	0	100	5	5	3600
JNKNOWN)	720	5	0	100	5	5	3600
443(UNKNOWN)	90	40	0	100	40	40	3600
443(UNKNOWN)	60	60	0	100	60	60	3600
3(TLS)	719	5	0	100	5	5	3595
JNKNOWN)	719	5	0	100	5	5	3595
JNKNOWN)	719	5	0	100	5	5	3595
443(UNKNOWN)	89	40	0	99	40	40	3560
443(UNKNOWN)	79	45	0	99	45	45	3555
443(UNKNOWN)	59	60	0	98	60	60	3540
43(UNKNOWN)	59	60	0	98	60	60	3540
'UNKNOWN)	59	60	0	98	60	60	3540
443(UNKNOWN)	59	60	0	98	60	60	3540
443(UNKNOWN)	59	60	0	98	60	60	3540
443(UNKNOWN)	59	60	0	98	60	60	3540
3(TLS)	59	60	0	98	60	60	3540

Figure 21: The rest of the dataset of test 2 after filtration

In the data set shown in figure 21, the red coloured text was the command and control channel of the malware. Greyed out area can be post filtered away because in this test regular flows whose jitter value was higher than 10 seconds were looked at. The rest of these results also contained the same false-positive results caused by Microsoft services than in test 1.

9.3.3 Test 3: Malware beacon interval 600 seconds, jitter 99%

In this test, the call home interval of the beacon was set to 10 minutes and jitter was set to 99%. The test simulates the case where malware calls rarely home and uses large jitter in intervals to obfuscate the detector. In this case, the command and control channel cause very low amounts of traffic in the network.

Parameters for the module were set to the following:

- Minimum interval 60 sec
- Maximum variance between intervals 1000 seconds

Figure 22 shows the results of the tests after all filters were enabled. The test result contained a high amount of false-positive results because a lot of legitimate regular connections happens in the range of intervals from 60 to 600 seconds.

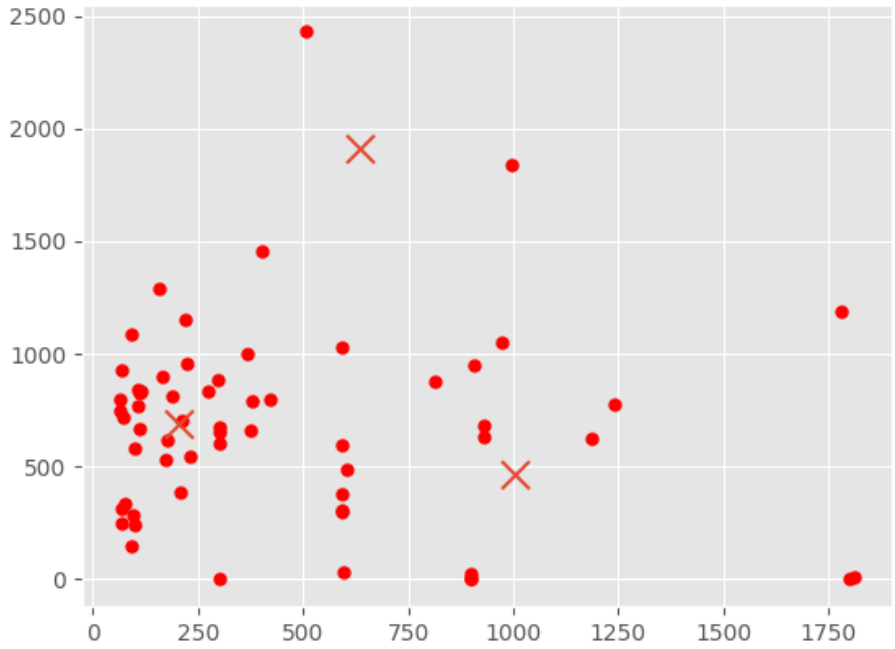

Figure 22: Scatter of all filtered regular flows generated from 4 hours of flow records X = Interval Y = Jitter. 60 Data points.

Due to a high number of false positives, post-filtration of the hand was performed. Figure 23 shows the rest of the dataset after post-filtration.

.82:443(TLS)	16	900	0	100	900	900	14400
3(SKYPE)	47	380	791	124	863	72	14216
162:443(UNKNOWN)	51	300	5	106	301	296	14109
1:53(DNS)	172	98	581	117	642	61	14048
3(OFFICE_365)	43	300	604	90	721	117	13958
57:80(HTTP)	25	595	31	103	599	568	13948
63:53(DNS)	32	420	799	93	886	87	13798
:53(DNS)	36	507	2430	127	2491	61	13691
8:53(APPLE)	28	592	303	115	596	293	13666
.100:53(APPLE)	24	593	377	99	622	245	13666
(DNS)	16	929	686	103	949	263	13547
10:53(GOOGLE_SERVICES)	225	95	285	148	346	61	13520
.82:443(TLS)	15	900	0	94	900	900	13500
).235:443(MICROSOFT)	14	996	1842	97	1956	114	13359
(DNS)	63	212	701	93	762	61	13303
65:53(OFFICE_365)	36	403	1455	101	1551	96	13123
1:53(APPLE)	28	594	306	116	598	292	13075
3(OFFICE_365)	239	67	248	111	310	62	13044
2:53(DNS)	17	972	1052	115	1247	195	13041
54:53(APPLE)	16	1185	622	132	1213	591	12478
32:53(OFFICE_365)	194	67	312	90	374	62	12443
2:53(DNS)	135	101	239	95	300	61	10991
45:443(MICROSOFT)	18	815	880	102	960	80	9244
.80:443(APPLE)	12	1779	1192	148	1779	587	8922
:53(DNS)	38	374	659	99	901	242	8475

Figure 23: The rest of the dataset.

Like in the previous test here the content of the greyed area was filtered out manually. Those could be filtered because the top-level application was known or the flow has not been present in the whole time window.

9.3.4 Test 4: Malware beacon interval 600 seconds, jitter 10%

In this test, the call home interval of the beacon was set to 600 seconds and jitter was set to 10%. The test simulated the case, where malware calls rarely home and uses jitter in transmit intervals to obfuscate the detector. In this case, the Command and Control channel caused very low amounts of traffic in the network. Because this test was performed first this test contains more detailed steps of the analysis process.

In step one, four hours of flow records were being fetched from the database and regular traffic detection was performed in a regular traffic detection module.

Parameters for the module were set to the following:

- Minimum interval 60 sec
- Maximum variance between intervals 600 seconds

The regular traffic detector detected 777 different regular traffic flows, which have been ongoing in a given time window. Figure 24 presents the scatter of all detected regular flows.

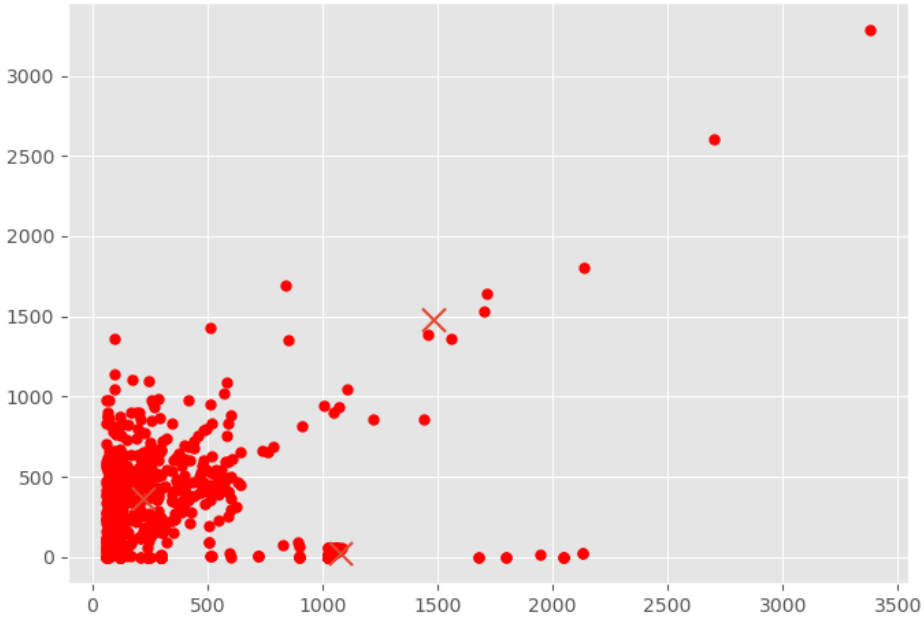

Figure 24: Scatter of all detected regular flows. X = Interval, Y = Jitter. 777 data points

The cluster at the down left corner in figure 24 was mostly caused by transient regular transmissions caused by actions such as WWW-browsing or AJAX-requests. These transients were caused because the variance in the regular traffic detector was set to relatively high, which means that even unregular flows that hit in the window, were also put into the dataset.

In this case, it was known that the malware calls home every 600 seconds with 50% jitter. Referring to the scatter in Figure 24, the point of malware traffic was practically hidden in the noise, but in the case where the parameters of the malware beacon are not known, it could be any of the points outside or inside the cluster. While it is nearly impossible to find the Command and Control channel of the malware from the current data, it was filtered.

In the second step, all detected regular traffic flows that were not present the whole time window was reduced. This reduces transient activities such as WWW-browsing out from the data, for example. The filter in this example was configured

by the following: flows that exits at least 90% of the given time window was kept in the dataset. The result after the second step can be seen in figure 25.

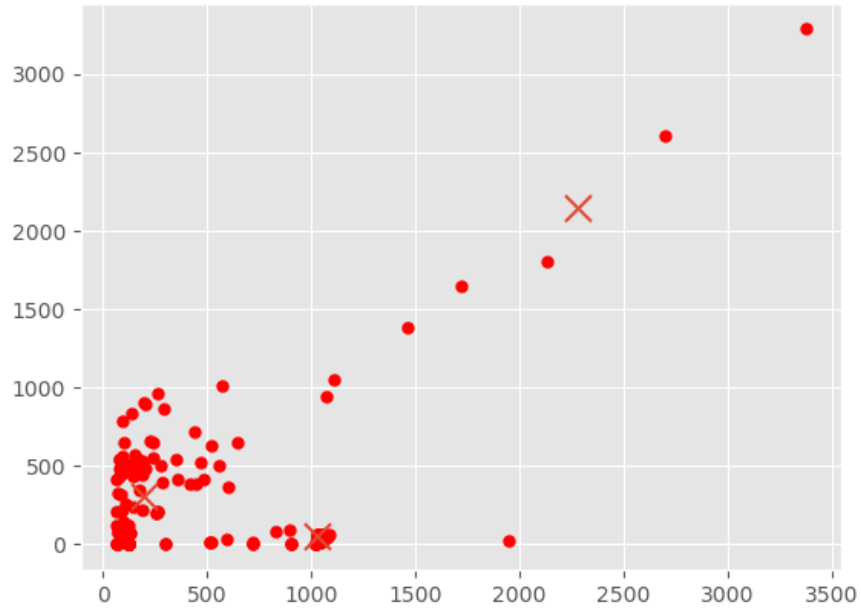

Figure 25: Scatter of filtered regular flows. X = Interval, Y = Jitter. 169 data points

In the scatter that is shown in figure 25, can be seen that there were still too many false-positive results that were required to be handled manually. In the case where the firewall is properly configured to allow only certain ports to communicate outside, the malware beacon may also use these ports. In step three all ports, except 53(DNS), 80(HTTP) and 443(HTTPS) are filtered out. The results of test three can be seen in figure 26.

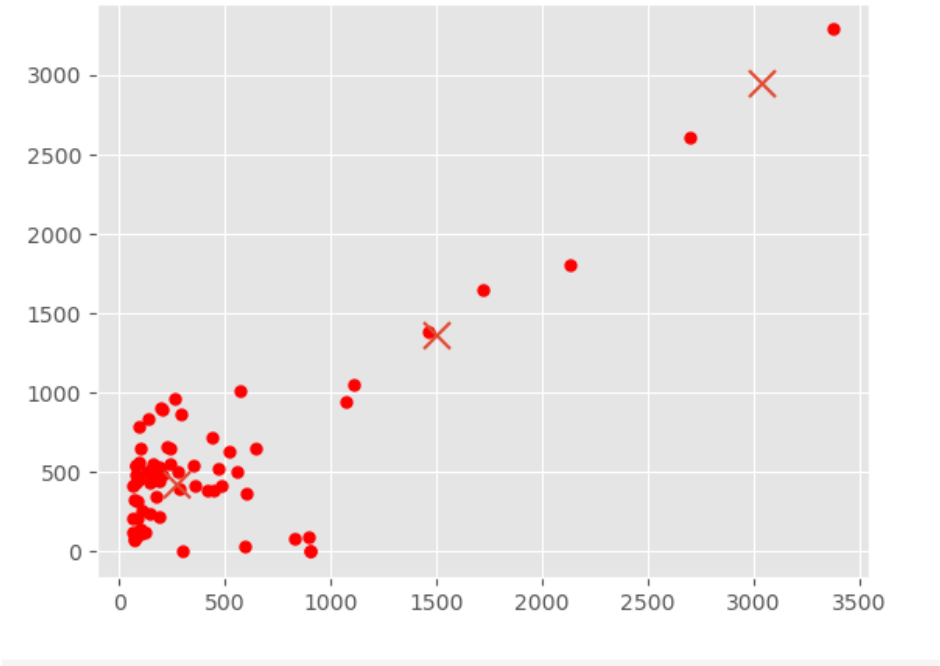

Figure 26: Scatter of regular flows where the port filter has been applied. X = Interval, Y = Jitter. 69 data points.

After the port filtration, the amount of data could be handled manually in a reasonable amount of time. Because it was known or expected that a beacon command and control traffic has been present for the full-time window, all flows that have been started in the time window can be filtered out. Figure 27 shows the results after the regular flow that was not present in the full-time window was removed.

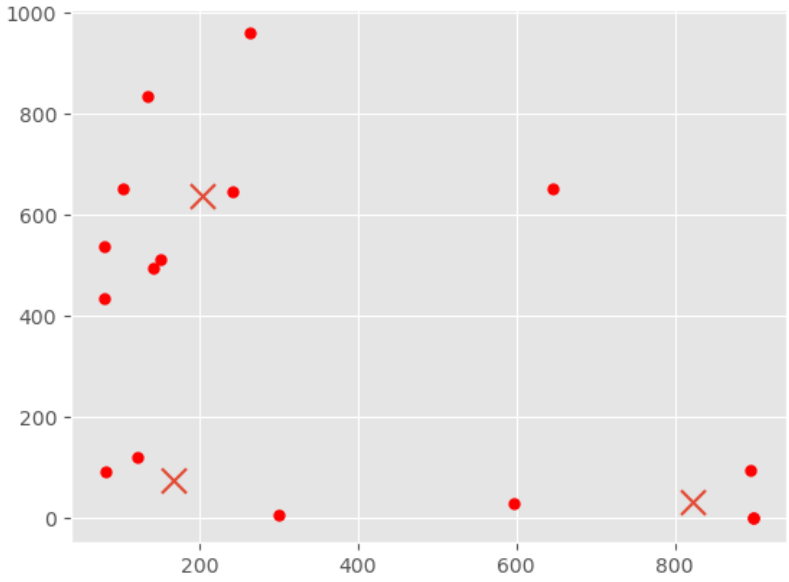

Figure 27: Scatter of regular flows where all started flows in the time window has been filtered. X = Interval, Y = Jitter. 16 data points.

Filtering flows that were not present in the whole time window reduced the number of results in the dataset. The amount of results left in the datasets is shown in figure 28.

.163:443	117	121	122	98	242	120	14277
2:53	67	241	646	112	721	75	13948
0:53	184	82	93	105	162	69	13556
	215	79	436	118	498	62	14136
225:443	53	301	8	111	302	294	14112
65:53	154	104	653	111	714	61	13609
.223:53	141	142	494	139	555	61	14086
.67:53	234	79	538	128	599	61	13990
1:53	30	646	653	135	921	268	14113
.200:53	113	151	512	118	573	61	13831
.86:53	80	263	959	146	1020	61	14232
57:53	25	597	29	104	597	568	14006
3	112	135	835	105	897	62	13684
	15	895	94	93	948	854	13502
.82:443	15	900	0	94	900	900	13500
.82:443	15	900	2	94	901	899	13500

Figure 28: The rest of the data set, the red coloured text is the control channel.

The beacon used port 53 (DNS) even though it runs by using TCP protocol and HTTP protocol inside it. This was caused by an accident in the configuration of the beacon. The accident does not cause a significant difference in the test results. In case the beacon has used port 80(HTTP), port 53(DNS) would have

been filtered out. All other flows in port 53 are clear false positives or unclear situations due to the nature of DNS protocol inside the environment where all queries use the same DNS Servers and port. Because the Regular traffic detector hashes flow with the following key: SRC_IP->DST_IP:DST_PORT, it sums all DNS queries caused by the host under the same hash.

If port 53 had been filtered out from the results and beacon would have used TCP ports 80 or 443, there would have been five flows left more for analysis. To determine which of them is the malicious flow, information from DNS log, WHOIS services or IP reputation databases can be utilized to determine if the IP-address of the flow is known to service or something else for example.

9.3.5 TEST 5: Malware beacon interval 4000 seconds, jitter 50%

In this test, the call home interval of the beacon was set to 4000 seconds with a 50% percentage of jitter. The test was practically the same as test 4, but with different values. The most significant difference, when compared with test number 4, is that the malware beacon was active in night hours which caused some reduction in the noise. Also, the port of the beacon listener was configured to 80 which is supposed to be also in test 4 as well.

Parameters for the module was set to the following:

- Minimum interval 60 sec
- Maximum variance between intervals 3000 seconds

These options resulted in a high number of false positives. The results can be seen in figure 29.

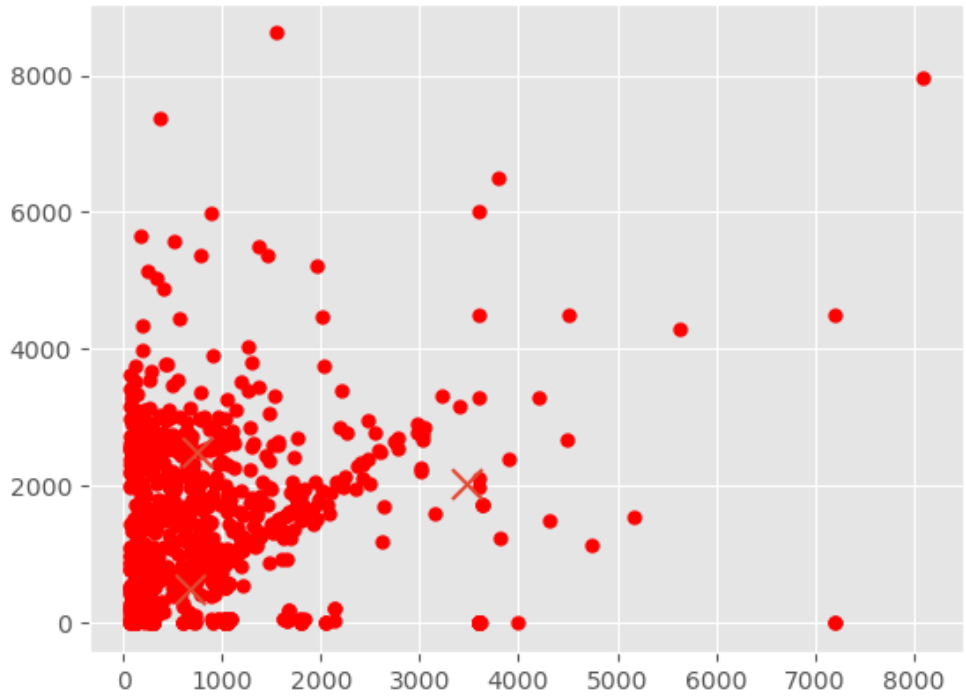

Figure 29: Scatter of regular flows, unfiltered. X = Interval, Y = Jitter. 16 data points: 1364.

Like in test 4, high variance caused a high number of false positives which must be filtered away to have fewer results for manual handling. Also, the relatively long time window showed the false positives caused by activities such as a WWW-browsing or Network Time protocol.

In the second step, all detected regular traffic flows that are not present in the whole time window were reduced. This reduces transient activities such as WWW-browsing out from the data, for example. The filter in this example was configured as following: flows that exits at least 90% of the given time window are kept in the dataset. Also, port filters were enabled for ports 80 and 443. The result after the mentioned filters can be seen in figure 30.

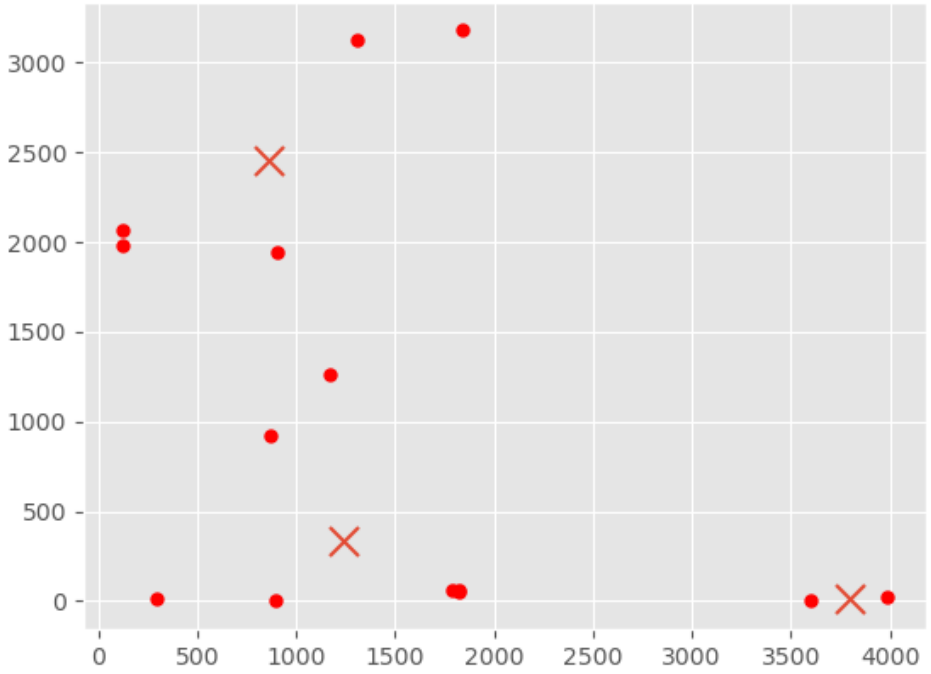

Figure 30: Scatter of regular flows, flows appeared during the time window deleted. X = Interval, Y = Jitter. Data points: 14.

In figure 30, an empty cap can be seen between 2000 seconds and 3500 seconds. In this case, the command and control channel was one of the two points on the right side of the scatter. Like in test number 4, the rest of the data could be enriched with other sources to determine if a source or destination IP-address malicious or not and filter them away from the results.

9.4 Discussion about the tests

Performed stress tests showed one weak link in the system which could be improved or fixed in the next development iterations. The high number of traffic flows from different sources causes a denial of service for the capture process, but practically this kind of activity can be detected in other metrics such events per seconds.

However, this kind of behaviour would cause a blind spot for the things that happen in the network. The one solution for the problem is to start sampling the flows, but in normal situations, it causes a reduction of data accuracy which leads

to possible blind spots. The better solution to this is to make a more optimal seek function for larger circular buffers that do not take long to execute. Some slowness in the seek function was noticed during the tests that were performed in chapters 9.1.2 and 9.1.3. The traffic capture process runs in a single thread allowing ~225k packets per second to be captured if the capture process does not contain bottlenecks. Increasing the number of threads to 4 and optimize the seek function of the circular buffer, capture rates over 1 million packets could be achieved with used hardware.

The decision tree-based network traffic anomaly detector seems to be good for networks where the type or content of flows does not often change. These kinds of networks are industrial OT-networks (Operational Technology) but also network management networks. The developed model did not operate well enough in networks that contain workstations with different users, because of the higher variance of the traffic behaviour, which causes a lot of false-positive results. The traffic baseline model would work fine in the case where the network includes users with workstations if the source or destination IP-address could be replaced with a username and generate a user-based traffic baseline model from the data by using user behaviour analytic methods.

The regular traffic detector module seems to be a good tool to detect traffic flows caused by a command and control channels of a specific malware. However, the method causes a relatively high amount of false-positive results caused by legitimate services which are currently required to be filtered manually from the results. Most of the false-positive results could be filtered away easily by the used protocol or application, but some of them would require a large whitelist of legitimate IP-addresses. As some services are moving to the cloud, where multiple service instances could share a single IP-address, the possible intruder also may have the possibility to move the listener of the beacon into the same cloud service provider. This would make the determination of malicious flow more difficult.

When using the regular traffic detector module, it requires the definition of the questions about what kind of traffic is being searched:’

- Are we looking at regular flow with a small interval or a large one?
- Does the interval of the regular flow jitter much or not?
- Which ports does it utilize?

Reflecting on the test results, it seems that the regular flow with a small interval is harder to detect than the regular flow with a higher interval. The jitter in the regular flow interval makes detection harder in real-time, but the effect of the jitter seems to turn against the malware when traffic flow information can be obtained with a large time window. Unfortunately, the jitter function did not operate fine in the malware beacon, so there was not much jitter seen in the results, even it was set to 99%.

Higher intervals may also expose the malware easily because most of the regular traffic intervals are under 60 seconds, which are caused by regular API calls of the cloud applications, website trackers or Ajax requests. Intervals higher than 60 seconds contain mostly traffic flows containing NTP and rare calls from the applications such as checking software updates from the server. This behaviour should hypothetically apply to most of the enterprise or home networks.

10 DISCUSSION

The topic and viewpoint of this thesis were quite wide, but the prototype of the system was developed successfully, answers were found for the defined research questions and main objectives were fulfilled. Some fields, such as data collection could have been extended easily to another thesis project.
The amount of set research and development objectives was set high which caused project limitations in project objectives during the project. artificial intelligence and machine learning processes could have been researched more but due the time taken in the data collection processes and solving issues reduced the allocated time for these.

The original plan included an initial design of a user interface of the system. Due to time was taken in other objectives and the wideness of the topic about creating the user interface, it was moved to further development iterations. However, an initial dashboard user interface was created as a side project during the thesis.

10.1 Answers for the defined research questions

In this subchapter, the answers to the defined research questions are discussed. The first question: Can the system setup and detection process be automated efficiently to get solutions like this quickly into production? The answer is theoretically yes. The data collection processes can be set up relatively quickly if all required hardware is available. The time goes for waiting until enough data is collected for machine learning processes. Machine learning processes reduces time to define baselines for the data flows significantly.

For the second question: How much computational resources would be needed? This depends on significantly from the environment where this kind of centralized data collection system is installed and what data is being collected. Data collection itself does require much computational power, but processing it may take. Generating ML models from data can be a heavy task.

For the third question: Is it possible to detect malicious network traffic from network flow records? Yes, but it requires a lot of processing and enriching the data to make decisions is some result malicious or not.

For the fourth question: What is the false positive rate when flow-based traffic statistic gathering is used as a source for anomaly detection? Depending on the case, the false positive rate can be high, but those can be reduced significantly by filtering out results that are known to be legitimate.

10.2 Future of the project and project proposals

The developed system requires a lot of development in the anomaly detection stage. The system itself provides quite a lot of data for telemetry use about the

users and the devices of the network. This data could be used for further research and development of detection modules such as deeper device or software behaviour analysis and user behaviour analysis. Some sources and project proposals are outside of the scope of the thesis project but could be integrated into the project.

Adding new data sources opens new possibilities for data analysis to create telemetry about network users and devices. Some possible upcoming data sources are:

- 802.11 Monitor
- Internal honeypots
- External honeypots
- Firewall log
- Electronic lock and access control system

IEEE 802.11 monitoring provides information about the wireless local area network devices around the monitored area. It could be implemented with a dense network of 802.11 adapters set up in the monitor mode as a sensor and send the received data to the centralized data collector. A wireless adapter in monitor mode provides hardware-level information about the radio interface such as MAC-address and power levels. From the data, the location of the wireless device can be calculated and the IP-address and user of the device can be determined by enriching the MAC address with ARP tables and authentication logs.

Internal honeypots could be made to detect lateral movement in the internal network. The data provided by this kind of honeypot can be used for determining the source and the content of the anomalous lateral movement. However, the data provided by network traffic capture exposes lateral movement without external devices.

External honey bots could be implemented to detect malicious IP addresses and events. This information could be useful to be used alongside the IP reputation

database provided by Open Threat Exchange to detect connections which source or destination IP is known as malicious.

Firewall log provides a good sight of the events that happen at the edge of the internal networks. All actions that have been denied, could be indexed as malicious activity. The information might provide valuable information about the reconnaissance phase, which is the first phase in the cybersecurity kill chain. (Hutchins et al., n.d.) Further network activity from the internal network to the destinations that are indexed can be used as one factor to detect a possible security breach.

Electronic lock and access control systems could be used as a source to determine a person's persistence in the area. The information could be used for user behaviour analysis alongside other data.

10.3 Telemetry and user behaviour-based multi-factor authentication for critical network assets

This chapter introduces a research and project proposal for how telemetry and user behaviour data could be used to protect critical network assets. One idea for possible practical implementation is presented.

Encryption in the network protocols and hashed passwords in databases have made traditional account takeover attacks less common making social engineering to be the highest attack vector in the account takeover process. Traditional multi-factor authentication provides an extra layer of security by sending verification code via short message service and the user must input the code in the login process. It makes the user account take over difficult, but it does not prevent social engineering from obtaining the verification code. (Cullen, 2016)

Employees of the organization might complain that they must remember multiple complex passwords and two-factor authentication takes a lot of time. Adaptive multi-factor authentication might be a way to solve these kinds of problem. Andy Zindel at Centrify defines that user behaviour-based adaptive authentication

should include a device profile, location awareness and user behaviour. (Zindel, 2017)

If including information from the network traffic with IP-address identity, 802.11 monitoring and electric lock and access control appliance, user behaviour analysis could also include the following features for an example:

- Which time users usually log on to the workstation and come into the premises?
- Which is the person's path to its office? How much time does it usually take from the main door to the office door? Does the time differ from the normal?
- Does a person have their mobile phone with him?
- What person does in the network? Does it differ from normal activity?
- Was a person on the premises when the logon happened?

Answers to these questions could be used to determine if the user is the user that it claims to be. Figure 31 presents an idea level implementation.

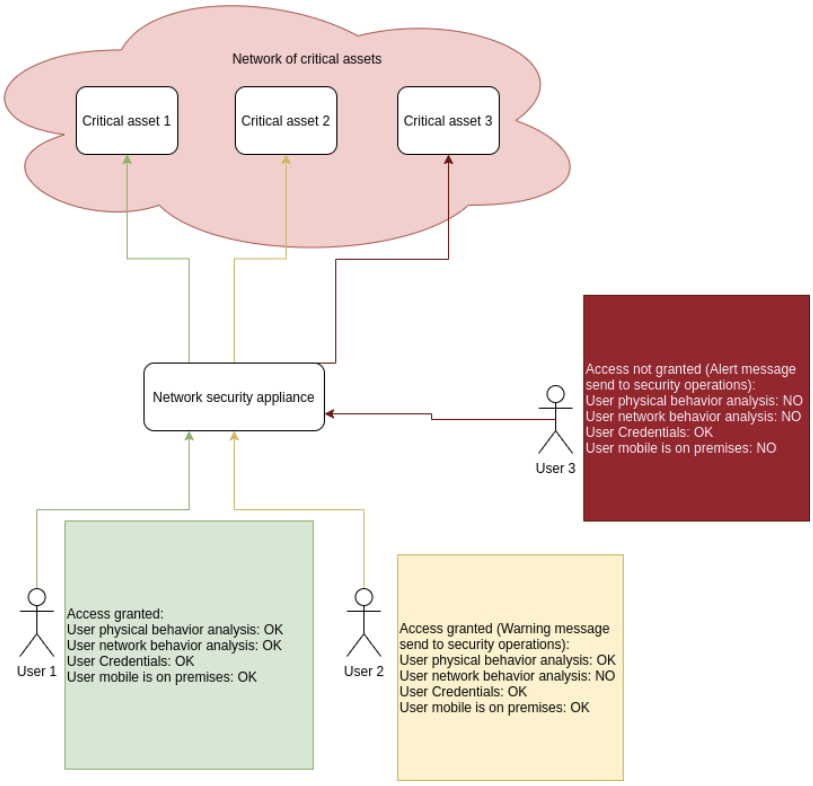

Figure 31: Idea level solution for telemetry and user behaviour based multi-factor authentication

The user which it claims to be. The information could be given into a network security appliance that permits or denies the traffic with user-based access lists.

10.4 Research about detecting advanced command and control channels and malicious activity

Could the collected data be useful for the detection of more advanced command and control channels of the different kinds of malware that use top-level applications as communication channels to achieve maximum stealthiness? The scenario mentioned in this chapter would be theoretically possible, but the practical implementation of the malware command and control channel would be complex.

For example, malware named backdoor.makadocs uses Google Docs for proxying command and control channel communications. It would be difficult to detect communication of this kind of malware with network-level security solutions such as next-generation firewalls. (Constantin, 2012)

Could application-level behaviour analytics expose this kind of activity by comparing activity to record one? Google Docs is used as an example Figure 32 shows differences in network packet count in different cases of Google Docs events with a resolution of one second. In Figure 32, the blue bar presents a single keypress in the Google Docs application. A yellow one presents a slow writing event and the red one fast writing. There is a significant difference in the shape that is generated from the values of packets per second and time.

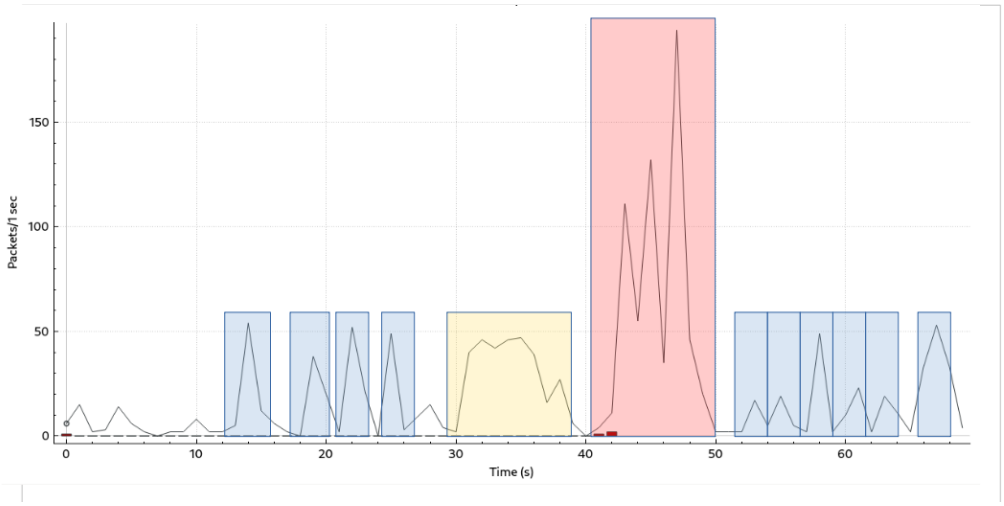

Figure 32: Three user-made keyboard-based events in Google Docs application.

Malware could also use a cloud-based document as the stealth command and control channel causing similar activities to happen. An example situation is seen in figure 33, where malware communicates through cloud office suites.

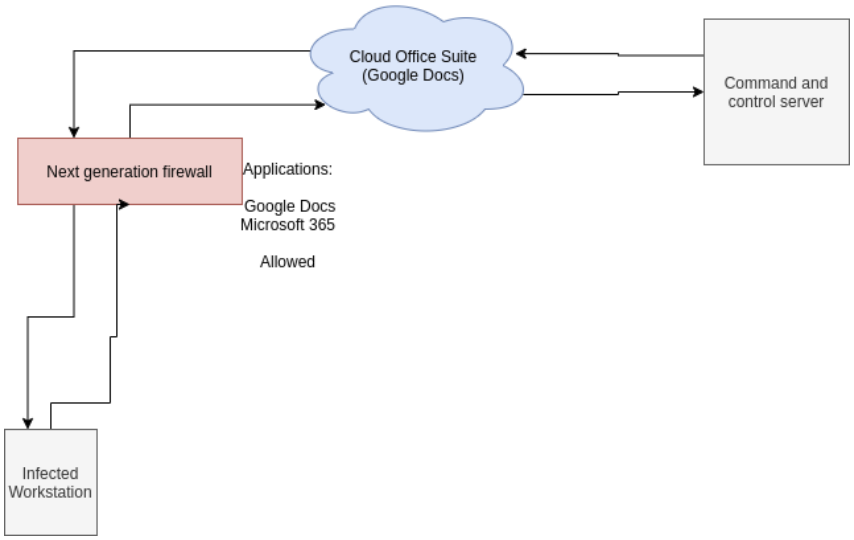

Figure 33: Example situation where command and control channel is implemented to communicate trough could office suite.

In figure 33, the situation where the infected workstation is located behind the next-generation firewall is presented. In this example case, the enterprise allows Office365 and Google Docs to allow employees to do their work. Because more people are using cloud applications such as office suites like Microsoft Office 365 and Google Docs, a straight block of these services does not cure the problems. Practically this is not limited only to cloud office suites, also different kinds of

forums or boards would be possible. Communication might be easier to detect because it is hypothetically easier to find small unique traffic flow records than flow records that are caused by activity that is normal in the network.

However, the connection speed is not high, especially in the case the malware emulates human behaviour to be as stealthy as possible. An average human can write around 40 words per minute, which translates to between 190 and 200 characters per minute. (“LiveChat,” n.d.) ASCII character encoding provides 95 printable characters. To encode one byte at least 2 ASCII characters are required to perform values in the range of 0-255.

With these values, it is possible to maximum calculate the approximate average connection speed of 90-100 Bytes/minute. However, no human can write for a long time without pauses. In this case malware command and control, a channel could be detected if there is a long period of fast writing activity. To avoid detection, malware communication must mimic human behaviour, which reduces the speed of the connection significantly.

The connection speed is way too low to deliver significant information in the channel of Google Docs in a reasonable amount of time. However, the speed of the connection would be high enough to trigger some further actions, such as faster command and control channel through an action such as phone or video call with Microsoft Teams for example. The audio or even video stream allows the full-duplex data transmission. With help of steganography, the command and control channel is theoretically possible to hide into the legitimate data.

Detection of this kind of activity would be difficult to detect from the collected data. However, using telemetry data mentioned in the previous subchapter and the method mentioned in regular traffic detection could help to expose the malicious activity through the mentioned techniques.

For example, it could be analyzed from the telemetry data if the legitimate user of the infected workstation is present in the area where the workstation is located.

Or does a single workstation perform a writing activity when it is not theoretically possible: the person is at lunch break or somewhere else. The third example would be the following: does fast writing activity happen regularly? In case it happens regularly, is there some legitimate reason for that?

10.5 Discussion about the project

In this thesis, the prototype of a modular framework for network security monitoring was successfully implemented. The implementation requires a lot of improvements in areas because some features were also implemented partially just to the needs of the initial research. Also, the codebase would require more standardization to make this for wider use.

The data collection methods would require more security improvements. However, some of them are insecure by default and it might not be possible to improve the security of them without using third party methods to provide a secure communication channel between the collector and the device.

The thesis project improved the visibility of the commissioner network significantly and also started research projects and thesis projects around the system in the fields of data collection and machine learning. A one side project, a security operations center for research and educational purposes, was built meanwhile. The environment with this system was piloted in a demo exercise in the autumn of the year 2020.

REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., 2017. Agile Software Development Methods: Review and Analysis. arXiv:1709.08439 [cs].

Akbas, E., 2019. SureLog SIEM Data Enrichment [WWW Document]. Medium. URL https://drertugrulakbas.medium.com/surelog-siem-data-enrichment-7125a5ed27b1 (accessed 11.12.20).

Anderson, B., Paul, S., McGrew, D., 2016. Deciphering Malware's use of TLS (without Decryption). arXiv:1607.01639 [cs].

Canner, B., 2019. The 5 Most Common Attack Vectors in Endpoint Security [WWW Document]. Best Endpoint Security Protection Software and Vendors. URL https://solutionsreview.com/endpoint-security/the-5-most-common-attack-vectors-in-endpoint-security/ (accessed 11.23.19).

Chakure, A., 2019. Decision Tree Classification [WWW Document]. Medium. URL https://medium.com/swlh/decision-tree-classification-de64fc4d5aac (accessed 11.12.20).

Cisco, n.d, Five Steps to Perimeter-Less Security

Conklin, K., 2018. What is Network Flow Monitoring? [WWW Document]. URL https://www.whatsupgold.com/blog/network-monitoring/why-you-need-network-flow-monitoring (accessed 11.27.19).

Constantin, L., 2012. Malware uses Google Docs as proxy to command and control server [WWW Document]. Computerworld. URL https://www.computerworld.com/article/2493242/malware-uses-google-docs-as-proxy-to-command-and-control-server.html (accessed 11.10.20).

Cullen, C., 2016. Password security fail? Add multifactor with user behavior analytics [WWW Document]. TechBeacon. URL https://techbeacon.com/security/password-security-fail-add-multifactor-authentication-behavior-analytics (accessed 11.10.20).

Davide, C., n.d. Figure 5.3: NetFlow collector architecture [WWW Document]. ResearchGate. URL https://www.researchgate.net/figure/NetFlow-collector-architecture_fig8_237812998 (accessed 11.12.20).

Davidoff, S., Ham, J., 2012. Network forensics: tracking hackers through cyberspace. Prentice Hall, Upper Saddle River, NJ.

Davin, J., Case, J.D., Fedor, M., Schoffstall, M.L., 1990. Simple Network Management Protocol (SNMP) [WWW Document]. URL https://tools.ietf.org/html/rfc1157 (accessed 11.25.19).

Douligeris, C., Serpanos, D.N., 2007. Network Security: Current Status and Future Directions.

Edn, 2009. Improvement of libpcap for lossless packet capturing in Linux using PF_RING kernel patch - EDN [WWW Document]. URL https://www.edn.com/improvement-of-libpcap-for-lossless-packet-capturing-in-linux-using-pf_ring-kernel-patch/ (accessed 11.12.20).

Endace, n.d. NetFlow Versus Full Packet Capture: understand the difference - Endace [WWW Document]. URL https://www.endace.com/articles/NetFlow-v-full-packet-capture (accessed 11.21.20).

Gardiner, J., Cova, M., Nagaraja, S., 2014. Command & Control - Understanding, Denying and Detecting.

Green, A., 2019, What is User Behavior Analytics URL https://www.varonis.com/blog/what-is-user-behavior-analytics/

Grimaldi, E., 2019. Decision Tree: an algorithm that works like the human brain [WWW Document]. Medium. URL https://towardsdatascience.com/decision-tree-an-algorithm-that-works-like-the-human-brain-8bc0652f1fc6 (accessed 11.30.19).

Guru99, 2020. MongoDB vs. MySQL: What's the difference? [WWW Document]. URL https://www.guru99.com/mongodb-vs-mysql.html (accessed 11.2.20).

Hetting, C., 2019. Smart home Wi-Fi devices to grow to 17 billion units by 2030. Wi-Fi NOW Events. URL https://wifinowevents.com/news-and-blog/research-smart-home-wi-fi-devices-to-grow-to-17-billion-units-by-2030/ (accessed 11.23.19).

Hintze, A., 2016. Understanding the Four Types of Artificial Intelligence [WWW Document]. URL https://www.govtech.com/computing/Understanding-the-Four-Types-of-Artificial-Intelligence.html (accessed 11.30.19).

Hutchins, E.M., Cloppert, M.J., Amin, R.M., n.d. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains 14.

IBM, 2019a. Artificial Intelligence for Smarter Cybersecurity [WWW Document]. URL https://www.ibm.com/security/artificial-intelligence (accessed 11.4.19).

IBM, 2019b. IBM QRadar SIEM - Overview - Finland [WWW Document]. URL https://www.ibm.com/fi-en/marketplace/ibm-qradar-siem (accessed 11.24.19).

Joshi, N., 2019. 7 Types Of Artificial Intelligence [WWW Document]. Forbes. URL https://www.forbes.com/sites/cognitiveworld/2019/06/19/7-types-of-artificial-intelligence/ (accessed 11.30.19).

Kaplan, A., Haenlein, M., 2019. Siri, Siri, in my hand: Who’s the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence. Business Horizons 62, 15–25. https://doi.org/10.1016/j.bushor.2018.08.004

Kent, K., Souppaya, M.P., 2006. Guide to computer security log management (No. NIST SP 800-92). National Institute of Standards and Technology, Gaithersburg, MD. https://doi.org/10.6028/NIST.SP.800-92

Kentik, n.d. NetFlow Guide: Types of Network Flow Analysis | Kentik [WWW Document]. URL https://www.kentik.com/NetFlow-guide-types-of-network-flow-analysis/ (accessed 11.12.20).

Kenton, W., 2019. Why Research and Development (R&D) Matters [WWW Document]. Investopedia. URL https://www.investopedia.com/terms/r/randd.asp (accessed 11.26.19).

Laravel LCC, 2020. Eloquent: Relationships - Laravel - The PHP Framework For Web Artisans [WWW Document]. URL https://laravel.com/docs/8.x/eloquent-relationships (accessed 11.2.20).

Lehto, M., Neittaanmaki, P., Nyrhinen, R., Ojalainen, A., Polonen, I., Rautiainen, I., Ruohonen, T., Tuominen, H., Vahakainu, P., 2018. Teko¨alyn perusteita ja sovelluksia 167.

Listvan, R., 2018. Introducing flow formats and their differences [WWW Document]. URL https://www.flowmon.com/en/blog/introducing-flow-formats-and-their-differences (accessed 11.12.20).

LiveChat [WWW Document], 2020. . LiveChat. URL https://www.livechat.com/typing-speed-test/ (accessed 11.10.20).

Luis, R., Franklin, S., 2019. Detecting Malicious Traffic on Your Network [WWW Document]. URL https://www.brighttalk.com/webcast/12099/355279/detecting-malicious-traffic-on-your-network (accessed 11.2.20).

Mady), M.S.(, 2018. Chapter 4: Decision Trees Algorithms [WWW Document]. Medium. URL https://medium.com/deep-math-machine-learning-ai/chapter-4-decision-trees-algorithms-b93975f7a1f1 (accessed 11.2.20).

Missinglink ai, n.d. 7 Types of Activation Functions in Neural Networks: How to Choose? [WWW Document]. MissingLink.ai. URL https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/ (accessed 11.2.20).

MongoDB Inc, 2020. 7 Best Practices for MongoDB Security [WWW Document]. MongoDB. URL https://www.mongodb.com/security-best-practices (accessed 11.2.20).

Morgan, J., 2016. What is Centralized Log Management (CLM)? [WWW Document]. Mission. URL https://www.missioncloud.com/blog/what-is-centralized-log-management-clm/ (accessed 11.29.19).

Muniz, J., Santos, O., 2017. NetFlow Versions > NetFlow for Cybersecurity | Cisco Press [WWW Document]. URL https://www.ciscopress.com/articles/article.asp?p=2812391&seqNum=3 (accessed 11.12.20).

Myung-Sup Kim, Hun-Jeong Kong, Seong-Cheol Hong, Seung-Hwa Chung, Hong, J.W., 2004. A flow-based method for abnormal network traffic detection, in: 2004 IEEE/IFIP Network Operations and Management Symposium (IEEE Cat. No.04CH37507). Presented at the 2004 IEEE/IFIP Network Operations and Management Symposium, IEEE, Seoul, South Korea, pp. 599–612. https://doi.org/10.1109/NOMS.2004.1317747

nDPI project, 2020. ntop/nDPI. Ntop.

NJRD, 2019. The Research and Development Cycle – NJRD. URL https://njrd.org/the-research-and-development-cycle/ (accessed 11.26.19).

Oltsik, J., 2019. Examining and Addressing Threat Detection and Response Challenges [WWW Document]. URL https://www.esg-global.com/blog/examining-and-addressing-threat-detection-and-response-challenges (accessed 12.1.19).

Panchen, S., Phaal, P., McKee, N., n.d. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks [WWW Document]. URL https://tools.ietf.org/html/rfc3176 (accessed 11.12.20).

Parmar, R., 2018. Common Loss functions in machine learning [WWW Document]. Medium. URL https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23 (accessed 11.2.20).

Petryschuk, S., 2019. NetFlow Basics: An Introduction to Monitoring Network Traffic [WWW Document]. Auvik Networks Inc. URL https://www.auvik.com/franklymsp/blog/NetFlow-basics/ (accessed 11.27.19).

Postel, J., 1981. Transmission Control Protocol [WWW Document]. URL https://tools.ietf.org/html/rfc793 (accessed 11.12.20).

Ring, M., Landes, D., Hotho, A., 2018. Detection of slow port scans in flow-based network traffic [WWW Document]. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6156027/ (accessed 11.12.20).

Sadek, R.A., Soliman, M.S., Elsayed, H.S., 2013. Effective Anomaly Intrusion Detection System based on Neural Network with Indicator Variable and Rough set Reduction 10, 7.

Salonen, K., 2013. NÄKÖKULMIA TUTKIMUKSELLISEEN JA TOIMINNALLISEEN OPINNÄYTETYÖHÖN 42.

Schroer, A., 2020. 30 companies merging AI and cybersecurity to keep us safe and sound [WWW Document]. Built In. URL https://builtin.com/artificial-intelligence/artificial-intelligence-cybersecurity (accessed 11.17.20)

Seif, G., 2019. The 5 Clustering Algorithms Data Scientists Need to Know [WWW Document]. Medium. URL https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68 (accessed 11.30.19).

Sharma, A., 2019. Data Enrichment: The Key Ingredient for SIEM Success [WWW Document]. Securonix. URL https://www.securonix.com/data-enrichment-the-key-ingredient-for-siem-success/ (accessed 11.30.19).

Sheridan, S., Keane, A., 2015. Detection of DNS-Based Covert Channel Beacon Signals. Journal of Information Warfare 14, 100–114.

Soniya, B., Wilscy, M., 2016. Detection of randomized bot command and control traffic on an end-point host. Alexandria Engineering Journal 55, 2771–2781. https://doi.org/10.1016/j.aej.2016.04.004

The Cylance Data Science Team, 2017. Introduction to Artificial Intelligence for Security Professionals. Cylance.

Todd, H., 2018. What Is Data Enrichment? [WWW Document]. RedPoint Global. URL https://www.redpointglobal.com/blog/what-is-data-enrichment/ (accessed 11.30.19).

Trammell, B., Claise, B., n.d. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information [WWW Document]. URL https://tools.ietf.org/html/rfc7011 (accessed 11.12.20).

ujjwalkarn, 2016. A Quick Introduction to Neural Networks. the data science blog. URL https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/ (accessed 11.2.20).

University of Jyväskylä, 2019. Määrällinen tutkimus — Jyväskylän yliopiston Koppa [WWW Document]. URL https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/menetelmapolku/tutkimusstrategiat/maarallinen-tutkimus (accessed 11.24.19).

Upguard, 2020. Splunk vs ELK: Which Works Best For You? | UpGuard [WWW Document]. URL https://www.upguard.com/blog/splunk-vs-elk (accessed 11.17.20).

Wilson, M., 2019. Port Mirroring - A Definition & How It Works,Tutorial Lab [WWW Document]. PC & Network Downloads - PCWDLD.com. URL https://www.pcwdld.com/port-mirroring-definition-and-tutorial (accessed 11.21.20).

Zindel, A., 2017. How Can User Behavior Analytics Kill the Password? [WWW Document]. Centrify. URL https://www.centrify.com/blog/user-behavior-analytics-password/ (accessed 11.10.20).

Zseby, T., Wagner, A., Mark, L., Boschi, E., Trammell, B.H., n.d. Specification of the IP Flow Information Export (IPFIX) File Format [WWW Document]. URL https://tools.ietf.org/html/rfc5655 (accessed 11.2.20).

LIST OF FIGURES

LIST OF TABLES: