BLEKINGE TEKNISKA HÖGSKOLA

BTH

DEGREE PROJECT FOR MASTER OF SCIENCE IN ENGINEERING
COMPUTER SECURITY

User and Entity Behavior Anomaly Detection using Network Traffic

Oskar Carlsson | Daniel Nabhani

Blekinge Institute of Technology, Karlskrona, Sweden, 2017

Supervisor: Dr. Abbas Cheddad, Department of Computer Science and Engineering, BTH

Abstract

An Advanced Persistent Threat (APT) represent a crucial threat to modern organizations, where the attackers are motivated and highly professional. In this thesis, a solution to detect APTs by analyzing network flows is proposed. Our approach focuses on three actions an APT performs, namely, callback to server, lateral movement and exfiltration of data. Features have been selected that are based on the amount of traffic sent, the timing of sending packets, direction of the traffic and ports used. These features have been used to train six different machine learning algorithms. In the results section, the performance evaluation of each of these algorithms have been presented.

The experiment has been performed on a dataset consisting of 1732 hosts in a real network environment, with APT data being simulated and injected. The results show that the algorithms K Nearest Neighbor (KNN) and Random Forest (RF) achieved the highest accuracy. Another conclusion is that while the algorithms are still able to detect the APT behavior on different stages, specifying one targeted scenario will significantly increase their accuracy.

Keywords: Advanced Persistent Threat, Intrusion Detection, Machine Learning

Sammanfattning

Ett avancerat långvarigt hot representerar ett allvarligt hot mot moderna organisationer, där angriparna är motiverade och professionella. I denna avhandling, föreslås en lösning för att detektera avancerade hot genom att analysera nätverksflöden. Vårt tillvägagångssätt fokuserar på tre handlingar som ett avancerat långvarigt hot utför, nämligen, återkoppling till server, lateral rörelse och exfiltration av data. Egenskaper har valts baserat på mängden trafik skickat, timingen av ett paket skickat, riktningen av trafiken samt portarna som användes. Dessa egenskaper har använts för att träna sex olika algoritmer inom maskininlärning. I resultat kapitlet presenteras en utvärdering av prestanda för varje algoritm.

Experimentet har utförts på ett dataset som består av 1732 enheter i en verklig nätverksmiljö, med trafik från avancerade långvariga hot som har simulerats och inkluderats. Resultaten visar att K Nearest Neighbor (KNN) och Random Forest (RF) uppnådde högsta noggrannhet. En annan slutsats är att medans algoritmerna fortfarande detekterat beteendet av avancerade hot på olika stadier, genom att specificera ett scenario kommer noggrannheten avsevärt ökas.

Nyckelord: Avancerade långvariga hot, Intrångsdetektering, Maskininlärning

Preface

We are two students in Blekinge Institute of Technology studying Masters in Science in Engineering, Computer Security. The education combines several branches of learning, such as mathematics, technology, science and IT-Security.

This thesis covers which machine learning algorithms, and which features are the most appropriate to be able to detect Advanced Persistent Threats.

Deep appreciation goes to SecureLink Sweden, Malmö office, for their support and aid of providing the data, for us to be able to explore the area of machine learning and security.

We deeply appreciate the help from our supervisor Dr. Abbas Cheddad as he has guided us through the area of machine learning, and by taking the time to help us overcome difficulties.

We would like to thank our examiner Dr. Dragos Ilie, reviewer Dr. Patrik Arlos and our opponents Edward Fenn and Erik Olsson Fornling, for their comments and advice on how to improve this thesis.

Nomenclature

Acronyms

AI Artificial Intelligence.

ANN Artificial Neural Network.

APT Advanced Persistent Threat.

CI Computational Intelligence.

CIDS Collaborative Intrusion Detection System.

DNN Deep Neural Network.

FNR False Negative Rate.

FPR False Positive Rate.

IDS Intrusion Detection System.

KNN K Nearest Neighbor.

NB Naïve Bayes.

RF Random Forest.

SVM Support Vector Machine.

UEBA User and Entity Behavior Anomaly Detection.

Table of Contents

1 INTRODUCTION

Advanced Persistent Threat (APT) represent the most threatening attack to modern organizations. APTs are attacks where motivated and knowledgeable individuals target an organization and spend a significant amount of time customizing the attack to be successful. The goal of an APT attack is to steal information from an organization while remaining undetected, thus allowing continuous exfiltration of data over a long period of time. Current Intrusion Detection System (IDS) can often be bypassed by performing reconnaissance and targeting users, rather than the systems the IDS focus on protecting. APTs utilize spear-phishing as the most common way of system infiltration. Detecting spear-phishing often requires host-based IDS which tracks system calls on each host computer individually, this requires a significant amount of resources which is not desirable in a large network environment. After the system is infiltrated, the attacker tries to avoid detection by communicating over normal protocols and using encryption to make messages unreadable.

In this paper the goal is to create a method of detecting APTs. The detection will be based on analyzing the behavior of a specific host using logs of network flows, then using machine learning algorithms to find anomalies that may indicate intrusion. This approach has the advantage of using logs which are simple to collect and store. The data is gathered from a real network, combined with specific scenarios containing known malicious traffic, and properties such as the size, timing and protocols are utilized.

1.1 Background

As the manual labor of finding malicious traffic in log-files is cumbersome, and there are companies that claim to be using User and Entity Behavior Anomaly Detection (UEBA) with machine learning, but none of these share any details regarding the method and algorithms used. The questions is: Are those companies using "real" machine learning, or simply adapted algorithms. SecureLink, the security company that is supporting this study, aims to explore UEBA with machine learning in order to know if it is an effective approach to detect APTs.

1.2 Objectives

The objective of the study is to create a module to facilitate analysis of network traffic to provide UEBA, using machine learning algorithms to detect APTs. By combining characteristics and creating multiple profiles, APTs can be identified at different stages of their life cycle.

Information flowing on the network is highly valuable for a security analyst but as the amount of data is usually enormous, manual work is cumbersome and hard and scalable solutions are always sought after.

Using the produced tool, experiments will be conducted which test different features of the network traffic. The results of these experiments will show which elements of the network traffic produce reliable results for finding anomalies indicating an APT. Experiments will also be done to show which machine learning algorithm is best performing, out of six different algorithms.

1.3 Delimitations

The experiment in this paper used a dataset containing 1732 hosts during a time of approximately four hours. Ideally a longer time period could have been used to establish more certainty that the profiles match the hosts, but this was the network traffic that was provided to us. Another interesting topic is to analyze how each host acts depending on the time of the day. Due to the restrictions of time in the logs used, this topic has not been explored.

The profiles that are created are intended to be requiring a low amount of resources as the data being analyzed can be enormous. The data used is taken from network flows, if the data used contained information gathered from specific hosts, detection can occur on additional phases of the APTs life cycle.

1.4 Thesis Question

The thesis attempts to answer the following questions:

- Which supervised algorithm in machine learning gives the best accuracy when analyzing the behavior for entities and users?
- Which characteristics of network traffic is most suitable for behavioral analysis?
- Does analysis of the behavior of devices and users provide an effective way to detect an advanced persistent threat?
- How reliable is an automated solution for analyzing the behavior of devices and users?

2 THEORETICAL FRAMEWORK

In this chapter we will explain the methodologies for intrusion detection, advanced persistent threat and its life cycle. This chapter will also cover machine learning and the algorithms used in this thesis, and the relationship between an IDS and machine learning.

2.1 Methodologies for Intrusion Detection

IDS is a software component designed to detect misuse and intrusion of a network. A recent review of IDS is found in [1], in which three categories of intrusion detection methodologies are classified, Signature-based Detection, Anomaly-based Detection and Stateful Protocol Analysis. The work presents both advantages and issues with each method, as it often is a question of which attacks the IDS will be effective at preventing and what level of resource usage is acceptable.

Signature-based detection uses patterns or strings which relate to a previously known attack or threat. Signature-based detection is an effective method when dealing with previously known attacks, it requires time and effort being spent in maintaining and updating the patterns' definition.

Anomaly-based detection utilizes profiles with expected behavior of hosts or users over a period of time, then searches for anomalies, it is effective at finding new or unknown vulnerabilities. Anomaly-based detection is reliant on accurate profiles and changes to the network will decrease its accuracy. The profiles in anomaly-based detection are often created using statistical or machine learning methods. In this thesis machine learning algorithms will be used to detect one recent method of attack, APT, which will be described later.

2.2 Technologies for Intrusion Detection

The classes of intrusion detection are defined by where they are deployed and which specific events they detect. Host based IDS gathers information on a per-host basis, where different characteristics are analyzed to separate legitimate users from intruders. Network-based IDS captures network traffic and analyses the protocols used and general usage to detect suspicious activity. Wireless-based IDS also analyses network traffic, but the gathering process is being done using wireless technologies. Network behavior analysis studies network traffic and detects unexpected behavior, and is the chosen method in this thesis.

In large networks relying on one specific method or technology for intrusion detection will result in more flaws than is considered acceptable. Collaborative Intrusion Detection System (CIDS) is a system that utilizes information from several detection methods- and technologies. Our method focuses on finding one specific threat, and should be considered one part of a CIDS.

2.3 Advanced Persistent Threat

Advanced Persistent Threat (APT) represents the most advanced and coordinated attacks to modern networks or computer systems. APTs are executed by a person or group of people with knowledge, time and motivation. An APT is customized to infiltrate a specific target, utilizing spear-phishing, zero day exploits or any other vulnerability found. Beyond infiltration, maintaining access and periodically extracting data is another goal of the attacker. Previously existing IDS often fail to effectively detect APTs. In [2] the shortcomings of traditional security

against APTs are explained, with the basic idea that unknown zero-day attacks or targeting personnel with access is a vulnerability difficult to mitigate.

2.3.1 Life cycle

Several definitions exists for APTs. In [3, Ch.3] the process to compromise an organization is defined in five steps. Another definition is found in [4] where the initial steps are highly similar. In the latter definition of APTs the final phase is "Data exfiltration", which is an important action to take note of, as it is one of the actions which can be detected to prevent some of the data loss during the final steps in an attack.

1. Reconnaissance.
2. Scanning.
3. Exploitation.
4. Create backdoors.
5. Cover their tracks / Exfiltrate data.

The reconnaissance phase is when the attacker identifies possible vulnerabilities and entry points. This phase generally considers information that is simple to receive, for example public information. Scanning is the action that follows. Here more specific information regarding the company, the network or the employees is found. Scanning for IP addresses and open ports is a usual process, and can be utilized during an APT if the attacker is not concerned about being detected at this stage. A stealthier version of scanning would be to use social media platforms, sending emails or directly calling the help-desk.

Exploitation is based on the information gained in the previous steps. The quickest way of gaining access is using spear-phishing, but other exploits using known vulnerabilities or zero-day attacks are also a possibility. By using personal information found through reconnaissance, spear-phishing can be done with high success rate. After the exploitation, a backdoor is created or a remote administration tool is installed. By creating an encrypted tunnel, information can be sent from the internal network while disguising it as a normal traffic.

The final step is to cover the tracks. The end goal is to maintain access and extract information during an extensive time period. The first and final steps are what separate APT from basic attacks. APTs has a focus of remaining undetected and maintaining access to the compromised system. Other definitions also put emphasis on the exfiltration of any data stolen.

This thesis will focus on detecting three phases of an APT, the stage when the intruder is communicating with a control and command server, when lateral movement is made to establish additional backdoors and during data exfiltration.

2.3.2 Detection of Advanced Persistent Threat

It may seem that the threat as defined is nearly impossible to prevent, but the attacker still has to abide by some rules, which are described in [5].

The attacker needs to run code inside the target organization, which leaves opportunities of prevention by traditional methods and with host-specific monitoring.

The paper [6] presents a method of detecting APTs by analyzing system events, detecting their dependencies and relationships within a network. By analyzing data from a semi-synthetic dataset, which then has malicious traffic injected. Detection is done on an anomaly-based method. This is an interesting approach which is similar to our method, but with a significant difference in the input data required. One major issue with trying to detect an infiltration using information gathered on each host system is the amount of resources required to monitor system calls on all hosts.

After an attacker gains access they need to establish a connection outside of the internal network. At first this connection is often used for command and control, later a connection will be used for data exfiltration. This communication can be detected but that is often not a simple task. Attackers try to imitate regular network traffic, communicating over common protocols and using encryption to hide messages. Another paper which, similarly to our method, uses information from network flows is [7]. In this paper the data is collected from a real network containing approximately 10,000 hosts, with the attack being simulated. This paper uses three features, focusing on the number of external connections and the amount of data sent from an internal host, the detection is being made against the exfiltration stage. The most interesting topic considered in this paper is the presentation of results, as it provides a ranking-list of suspected hosts. Our method will attempt at detecting additional phases of an APT, as well as evaluate additional features and consider the change in the profile of any specific host.

In order to bypass the difficulties in detecting APTs the expected behavior during specific life cycles will be analyzed. Information such as amount of data sent, flow direction and timestamps can not be hidden using encryption. If an attacker wishes to extract data or make external connections on a host, where this is an unusual action, it will leave traces that can be detected.

2.4 Machine Learning

Machine learning exists everywhere in our daily lives, everything from Google search and Netflix suggestions to automated self driving cars, and Computer-Aided Medical Diagnosis. The term Machine learning suggests the use of a machine or a computer to learn in analogy similar to how the brain learns and predicts [8], [9]. Essentially, machines make sense of data in similar manners as humans. The idea behind machine learning is that the computer or machine learns to perform a task by learning from a set of examples, and then performs the same task with new data.

Machine learning are performed in three phases:

1. Phase 1 - Training Phase
2. Phase 2 - Validation and Test Phase
3. Phase 3 - Application Phase

The first phase is to train the algorithm with the expected output. The second phase is to measure how good the model have been trained and check it properties, such as recall, precision, and evaluate the output based on the validation dataset, and the last phase is to subject the model to the real-world data [10].

There exists a couple of studies implementing machine learning for IDS, the paper [11] mentions that there are two machine learning-based approaches to IDS. The two types are Artificial

Intelligence (AI) or Computational Intelligence (CI), where the AI approaches is towards statistical modeling while CI is a nature-inspired methods.

Algorithms like Support Vector Machine (SVM) and K Nearest Neighbor (KNN) are considered Artificial intelligence, examples of Computational intelligence algorithms are Artificial Neural Network (ANN) and Genetic Algorithms.

Under the AI approach a comparison between supervised and unsupervised methods are evaluated based on known and unknown attacks. The outcome of the tests is that supervised learning is in general better at classifying than unsupervised, however with unknown data, the results of the classification is similar.

The Computational Intelligence is considered more adaptive and fault tolerant, it conforms the requirements for IDS even though the paper [12] received a higher score by using the Artificial intelligence approach of SVM. It proves there is potential for SVM in IDS, and SVM may replace neural networks due to its scalability, the drawback however is that it is limited to only use binary classification.

In the paper [13] the author evaluates the applicability of machine learning on IDS and compares unsupervised and supervised learning methods, supervised methods outperform unsupervised in known attacks, but deteriorates with unknown attacks, however not below unsupervised. These studies prove the benefits of using supervised machine learning algorithms over unsupervised.

2.4.1 Naïve Bayes

The Naïve Bayes classifier is based on the Bayes theorem and means knowing the value of one attribute does influence the value of any other [14]. Bayes theorem can be defined by the following steps:

$$p(A \text{ and } B) = p(B \text{ and } A) \tag{2.1}$$

Where the probability of A and B is the probability of A and the probability of B.
A:

$$\begin{aligned} p(A \text{ and } B) &= p(A)p(B|A) \\ p(B \text{ and } A) &= p(B)p(A|B) \\ p(A)p(B|A) &= p(B)p(A|B) \\ p(A|B) &= \frac{p(A)p(B|A)}{p(B)} \end{aligned} \tag{2.2}$$

To predict the outcome, Bayes theorem must be run for every outcome, and the calculated probability with the highest score will be chosen as the predicted class. The algorithm assumes a Gaussian distribution for each feature and is also called Gaussian Naïve Bayes Classifier [14]. According to the book [10, Ch.9] "Naïve Bayes is particularly good when there is missing data".

In the paper [15] the authors compares the accuracy performance of Naïve Bayes, Bayesian Networks, Lazy Learning of Bayesian Rules and Instance-Based Learner. They come to the conclusion that Naïve Bayes is one of the best performing algorithms, due to the results based on this study, the algorithm can therefore be considered feasible.

2.4.2 Artificial Neural Network

Artificial Neural Network (ANN) is based on the computations of the brain, the brain works with highly interconnected network of neurons that communicate by sending electric pulses through the wiring [16]. ANN receives input from a number of other units and weighs each of the features in the input and adds them up, and if the sum is above the threshold the output is one otherwise it is zero. The simplest form of neurons are linear neurons and represented in Figure 2.1.

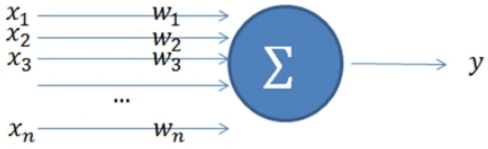

Figure 2.1: Linear neuron

Where the output y is a sum of the product of the input x_i and the weight w_i and b is the bias:

$$y = b + \sum_{i=1}^{n} (w_i x_i) \tag{2.3}$$

The most common ANN is based on a feed-forward network and contains either one or two hidden layers besides the input and output layer. A fully connected multilayer feed-forward network or multilayer perceptrons has multiple layers where one layer is the input to a subsequent layer [10].

Krogh explains the risk of over-fitting ANN as "Over-fitting occurs when the network has too many parameters to be learned from" [16], and a network that over-fits on the training data is unlikely to generalize to input that are outside the scope of the training. One of suggestions to avoid over-fitting is to make a small network, and to use methods from the Bayesian statistics. Neural networks have been widely applied to many interesting problems in science, as ANN methods performs well for classifying latent variables that are difficult to measure.

2.4.2.1 Deep Neural Network

Deep learning focuses on unifying machine learning with artificial intelligence [10]. Deep learning makes it possible to build computational models that are composed of multiple processing layers, the Deep learning methods have improved speech recognition and visual object recognition [17]. In the paper [18], the authors managed to beat the champion in Go with their program AlphaGo by building policy networks and value networks. Both the policy networks and value networks are considered as a Deep Neural Network (DNN). DNN is a version of ANN with two or more hidden layers [19].

2.4.3 K Nearest Neighbor

K Nearest Neighbor (KNN) is based on that each data point being assigned to the label that has the highest confidence among the K data points nearest the new data point. The key components of KNN are the numbers of k(nearest neighbors) and the distance [10], [14]. The distance is measured in Euclidean distance:

$$D\left(x, x'\right) = \sqrt{\sum \left|x_d - x'_d\right|} \tag{2.4}$$

The benefit of KNN is that its classification decision is based on a neighborhood of similar objects and is well suited for multimodal classes. As the authors Sohail and Bhattacharya mentions "even if the target class is multimodal, it can still lead to a good classification accuracy" [20].

2.4.4 Random Forest

Random Forest (RF) is based on decision trees and consists of many decision trees, where the output is decided by the votes given by all the individual trees. The decision trees is built by randomly selected data and combined with a random selection of a subset of attributes. The collection of tree-structured classifiers can be defined as $\{h(x, \Theta_k), k = 1, ..\}$ from Breiman definition where the $\{\Theta_k\}$ is independent identically distributed random vectors.

The author Biau and Scornet also describes in mathematical terms that the jth tree estimate takes the form:

$$m_n(x; \Theta_j, D_n) = \sum_{i \in D_n^*(\Theta_j)} \frac{\mathbb{1} X_i \in A_n(x; \Theta_j, D_n)^{Y_i}}{N_n(x; \Theta_j, D_n)} \tag{2.5}$$

And for a finite forest estimate the mathematical term is:

$$m_{M,n}(x; \Theta_1, ..., \Theta_M, D_n) = \frac{1}{M} \sum_{j=1}^{M} m_n(X; \Theta_j, D_n) \tag{2.6}$$

The predicted value x is denoted by $m_n(x; \Theta_j, D_n)$, where $\Theta_1, ...\Theta_m$ are independent random variables and independent of D_n, where $D_n^*(\Theta_j)$ is the set of data points selected for the tree construction [22].

As the random forest consist of many decision trees, each of these trees have a decision to label the testing data. The output classification result is based on the most set label among trees. Random Forest (RF) is a learning algorithm that scales with the volume of information while maintaining statistical efficiency [10], [14]. In the paper [21] the author goes in depth into the mathematics behind RF and for a deeper explanation about the formula and the mathematics a neat discussion can be found in [22].

2.4.5 Support Vector Machine

Support Vector Machine (SVM) is an algorithm that learns by example and is used for solving classification problems. There are a few basic concepts needed to understand SVM. The first is a separating hyperplane, and can be defined as a line in 2-Dimensions and as a plane in 3-Dimensions. The hyperplane is separating two clusters, and predicts the label of an unknown value by following the simple rule of which side of the line it falls into. The hyperplane is defined by the maximum margin. The maximum-margin hyperplane helps to define which gradient of all the possible separating lines is the best classifier, SVM selects the maximum-margin separating the hyperplane, and maximizes the ability to classify any unseen values. The new hyperplane can be represented by a linear equation [10]:

$$f(x) = ax + b \tag{2.7}$$

The distance between the hyperplane and the point can be calculated by:

$$M1 = |f(x)|/||a|| = 1/||a|| \tag{2.8}$$

To maximize $||a||$ is a non-linear optimization task and can be done by the Karush-Kuhn-Tucker condition, using the Lagrange multiplier λ_i:

$$\begin{aligned} a &= \sum_{i=0}^{N} \lambda_i y_i \overrightarrow{x_i} \\ \sum_{i=0}^{N} \lambda_i y_i &= 0 \end{aligned} \tag{2.9}$$

To handle outliers in the data, the SVM algorithm introduced the soft margin to allow a few outliers to fall on the wrong side without affecting the final result [23]. In cases of non-separable datasets, where the values can not be separated by a single point and a soft margin would not help, the kernel function is the solution. The kernel function adds a dimension to the data, as mention by Noble "the kernel function is a mathematical trick that allows the SVM to perform a 'two-dimensional' classification of a set of originally one-dimensional data" [23].

3 METHOD

This chapter will cover information about the dataset and how the data were collected, and the features utilized. The chapter will also cover short information about the machine learning algorithms and their parameters.

3.1 Dataset

Using data based from a real network scenario has both advantages and disadvantages compared to a virtually manufactured one, the differences are studied in [24]. Using a synthetic model for creating data allows full control of the amount of data gathered, and how the network is set up. Synthetic models create a model with the desired properties, no regular noise and no unknown properties. The lack of noise can be considered an advantage when the goal is to create a model which allows simple reproducibility.

Previous studies that focus on APTs use either synthetic, semi-synthetic [6] or real network data [7]. Existing solutions which focus on host-based detection and studying system-calls use synthetic data to a greater extent. The studies which focuses on the behavior of network traffic are willing to accept the disadvantages of real network data to gain the most realistic scenario possible. Previously existing datasets for IDS are often not considered as they contain outdated attacks that are not similar to an APT in behavior.

The goal of our study is to study real behavior, having data with noise, unexpected or unpredictable behavior is crucial if the results are going to be representative to a real network scenario.

3.2 Data Collection

The data used in the experiments can be separated in two parts. The data indicating an APT is created in a controlled environment. A real network is used to generate the data which is considered regular, it is containing 1732 hosts and covers the total time of approximately four hours. The data indicating the attack is generated by simulating various attacks either between two computers in the same local area network or towards a website, to simulate outgoing network traffic.

The beacon traffic is generated with a tool from open security research[1], the tool simulates a beacon communication attempt towards a single URL over HTTP. The simulation ran for four hours and sent a beacon every hour.

The lateral movement attack is simulated using bash and netcat, with a predefined range of IP-addresses and ports. The IP-range is taken from the regular data, and the following ports are used: 443, 23, 389, 137, 53, they are all used frequently in the dataset.

The Port scan scenario is from the internal network, utilizing Nmap in a controlled virtual network containing five virtual machines, all running Linux Mint. Nmap was run with the flag "-F" for the 100 most common ports, over the whole subnet.

[1] T. Lee, "Testing Your Defenses - Beaconing", 2012, [Online].
Available:
http://blog.opensecurityresearch.com/2012/12/testing-your-defenses-beaconing.html. [Accessed: 2017-May-06].

Exfiltration simulates data gathering over a session with a short time between sending the information, the sessions is open between each transmission of data, the amount of data sent is 100MB, 1GB, 3GB, 5GB, 10GB and 30GB. This is done with a bash script sending to an open netcat socket set in listening mode, and every transmission sends the content of a file of 50MB in size.

All of the generated traffic is captured with tcpdump a network capture tool, into pcap-files and turned into flow records with tshark, a Wireshark network analyzer tool.

The regular data is traffic-logs from Alto Palo Networks Firewalls, with information about each session between a source IP and a destination IP, each of the sessions contains information about the start time, duration, ports used, the total bytes and amount of packets, the amount of packets received and the bytes received, and a few other parameters that are not relevant for this study.

3.3 Feature Extraction and Normalization

The main aspect that affects the results when using machine learning to solve a task is the features selected, and how they are preprocessed. In the experiments seven features are used, and certain experiments are performed using only a part of them, to allow comparison and effectiveness of specific features.

1. Destination port
2. Flow direction
3. Bytes sent
4. Average packet size
5. Average received size
6. Traffic flow ratio
7. Interval of packets sent

The destination port is used as a feature to target intruders during the stage of gaining additional lateral access in a network. The flow direction determines if the session is established from the internal or external network. The direction of traffic is of interest as different stages of an APT will behave in somewhat predictable manners. During the early stages of an APT an infiltrated host is expected to communicate to a command and control server, detecting this communication will rely partially on the flow direction. If the intent is to detect lateral movement of an intruder the flow direction can be used to limit the traffic to internal hosts and reduce noise. During the data exfiltration stage flow direction can also be used as a first step of reducing noise due to the predictable flow direction.

The amount of bytes sent is another action which can be expected to be predictable. When an intruder is inside of the network the amount of traffic will often be low to avoid detection, this will change when the attacker is ready to exfiltrate data. During exfiltration stage this feature is predictable as the data in modern systems will reach significant numbers and will show a clear anomaly.

The data is stored based on sessions, the average packet size indicates the average size of a packet in one specific session. The average received packet size considers the same feature, except when the traffic is flowing to the host being analyzed, and not from the selected host. In general the

packet size in any session will be low, when file transmissions occur this value will increase, in Figure 3.1 the value of this feature within the dataset is presented.

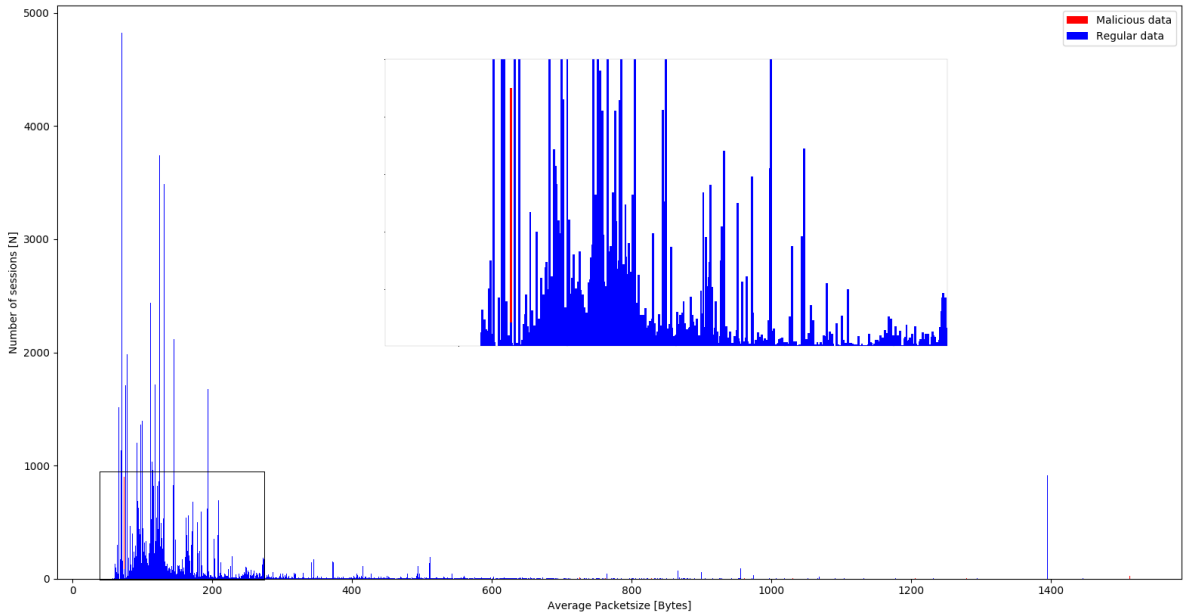

Figure 3.1: Average packet size and occurrences for each session. The full size region rectangle is inserted to show the malicious data

The traffic flow ratio compares the total traffic sent to the total traffic received during one session.

The interval of packets sent is a value which is calculated to find any traffic that resembles continuous beacon traffic. This feature is calculated as follows: The time difference of each packet which one internal host sends to one external host is calculated. The standard deviation of these time differences is then calculated, after this the result is divided by the number of differences found. This will result in a number where suspected beacon traffic will be close to zero, and most other traffic will be higher.

3.4 Experiment setup

In this section the setup for the experiment is explained in more detail. The specific optimization options and other parameters for each algorithm are explained.

3.4.1 Preparation

The dataset is split into two groups, train and test, where the train is 70% of the set and the test 30%, the dataset is grouped into two categories, "normal" and "malicious".

3.4.2 Scikit-learn

Scikit learn is a python library for simple and efficient tools for data mining and data analysis, and have been used for the machine learning algorithms classifiers, and calculating the accuracy score based on the amount correct estimations from the test data [25]. Scikit-learn is used due to

it being one of the most prominent machine learning libraries for python, with numerous machine learning algorithms implemented.

The accuracy calculation is done by utilizing the library function $accuracy\_score$ that is based on every classifiers prediction functions, where the classifier tries to estimate the label and the correct label. With the return value of the prediction function it is able to take out both classification report and confusion matrix. The classification report shows the main classification metrics and shows the precision, recall, f1-score and support for the predictions. The confusion matrix evaluate the accuracy of the classifiers [26].

3.4.2.1 Support Vector Machine

The classifier for SVM is the $sklearn.svm.SVC$. The parameter C is set to two, C is the penalty parameter, and instructs the algorithm on how much miss-classification to avoid. $class\_weight$ is set to 0.9 for the malicious data and 0.3 for the valid data to emphasis the focus on the malicious as the dataset is uneven. The class weight is determined by testing and evaluating results. The default kernel is used, radial basis function kernel, and $verbose$ is set to 10 to enable a verbose output for more information.

3.4.2.2 Artificial Neural Network

ANN algorithm is from $sklearn.neural\_network.MLPClassifier$, with the hidden layers based on [27], where the optimal size for the first layers is:

$$L_1 = \sqrt{(m+2)N} + 2\sqrt{\frac{N}{m+2}} \tag{3.1}$$

and the second layer:

$$L_2 = m\sqrt{\frac{N}{m+2}} \tag{3.2}$$

where N is the distinct samples and m is the number of outputs, in our case two. The $activation$ for the layers is the hyperbolic tan function $f(x) = tanh(x)$. The solver stochastic gradient descent, that performs a parameter update for each training example [28]:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{3.3}$$

and the value $verbose$ is set similar to the SVM settings.

3.4.2.3 Naïve Bayes

The Gaussian Naïve Bayes algorithm is from $sklearn.naive\_bayes.GaussianNB$ and was used with the default values.

3.4.2.4 K Nearest Neighbor

Nearest Neighbor algorithm KNN have the neighbor size of five and is using the algorithm KDTree, it is built on the class $sklearn.neighbors.KNeighborsClassifier$. n_jobs is set to -1 to increase the numbers of cores used from the processor.

3.4.2.5 Random Forest

RF is based on the scikit-learn class $sklearn.ensemble.RandomForestClassifier$. The $class\_weight$ parameter is set as for SVM, $n\_estimators$ is set to 50 and the parameter $min\_samples\_leaf$ is set to two. RF also have the same value on n_jobs as KNN.

3.4.3 Tensorflow

Tensorflow is a new machine learning library for python from Google that works more towards deep learning. The classifier DNN is based on $tensorflow.contrib.learn.DNN$ to simplify the construction of a DNN. The hidden inputs are based on the same mathematical formula used for ANN, except the use of multiple layers, where the amount of layers is based on the amount of features used, and every layer from two and forth is using the formula for L_2. The number of classes is set to two and the $activation\_fn$ for DNN is using the same as ANN, tanh.

The $optimizer$ is set to the default Adagrad. Adagrad is a gradient-based optimization algorithm, that adapts it's learning and parameters. Adagrad was published in 2011 and can be found in paper [29].

The training of the classifier is done in $10,000$ steps with the batch size of 500.

4 RESULTS

In section 4.1 a brief explanation about the metrics used to present results is provided. Section 4.2 will present the results from each of the scenarios.

4.1 Metrics

The false positive rate is the amount of positive test outcomes of the negatives and the false negative rate is the reverse, the amount of negatives test outcomes from the positives. The calculation of False Positive Rate (FPR) and False Negative Rate (FNR) is based on the confusion matrix data from the machine learning algorithms. Structure of confusion matrix:

TP	FN
FP	TN

The calculation of FPR is done with False Positive and True Negative:

$$\frac{FP}{FP + TN} \tag{4.1}$$

And the calculation of FNR is done with True Positive and False Negative:

$$\frac{FN}{FN + TP} \tag{4.2}$$

The accuracy score is calculated by:

$$\frac{TP + TN}{N} \tag{4.3}$$

Where N is the total amount of values.

4.2 Experiment

Three categories of target hosts exist in these experiments, "Low", "Medium" and "High", these reflect the amount of activity the targeted host has in the network. A host under the "Low" category has connections with 1-5 different hosts, "Medium" has connections with 100-300 hosts and "High" has connections with over 500 hosts.

The feature weight figures is based on information from the algorithm RF, as the library only provides this function for RF.

4.2.1 Lateral movement

Table 4.1, 4.2 and 4.3 contains results from the experiment with real data and injected data from lateral movement. In the scenario of lateral movement, the training data used two internal hosts sending packets to 60 and 80 respectively different internal hosts using the ports 443, 23, 389, 137 and 53. The test data used an internal host sending to 40 different internal hosts.

The results show that in the scenario of lateral movement the algorithms RF and KNN provide the highest accuracy combined with low FPR and FNR. Naïve Bayes (NB) also provided accurate results but with more flaws especially in the test with high usage hosts.

In Figure 4.1, 4.2 and 4.3 the feature weight for the algorithm RF is shown, the most common feature between the lateral movement scenarios is the *packet size*.

Algorithm	Accuracy	FPR	FNR
ANN	0.9979	0.2140	2×10^{-4}
KNN	1	0	0
NB	0.9995	0	5×10^{-4}
RF	1	0	0
DNN	0.9978	0.2177	3×10^{-4}
SVM	0.9998	0.0184	0

Table 4.1: Lateral movement Low usage hosts

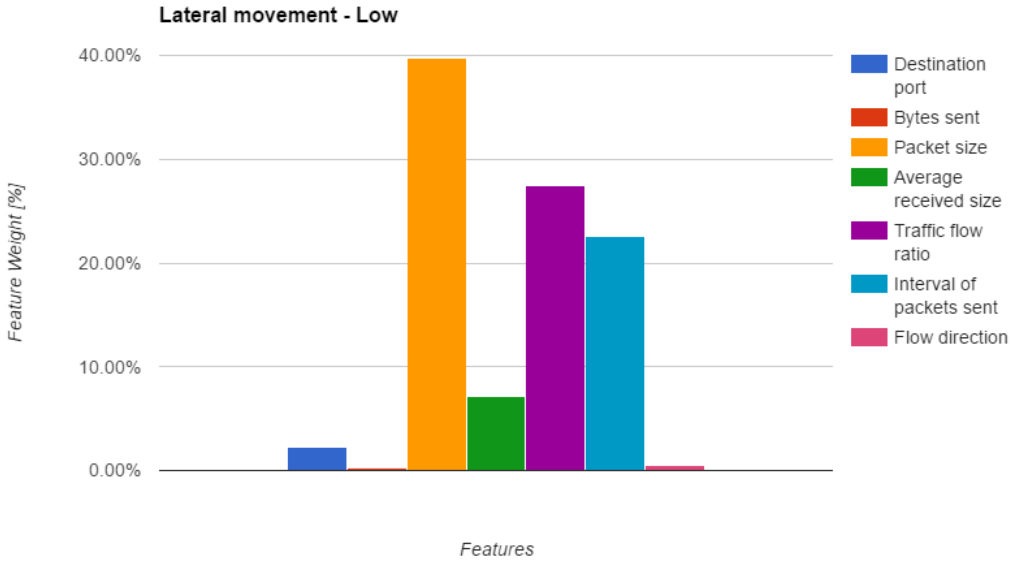

Figure 4.1: Feature weights for the scenario lateral movement with low usage hosts

Algorithm	Accuracy	FPR	FNR
ANN	0.9951	0.5181	2×10^{-4}
KNN	0.9988	0.1232	1×10^{-4}
NB	0.9822	0	1.79×10^{-2}
RF	0.9991	0.0870	1×10^{-4}
DNN	0.9933	0.6812	4×10^{-4}
SVM	0.9984	0.1630	6.6×10^{-5}

Table 4.2: Lateral movement Medium Usage Hosts

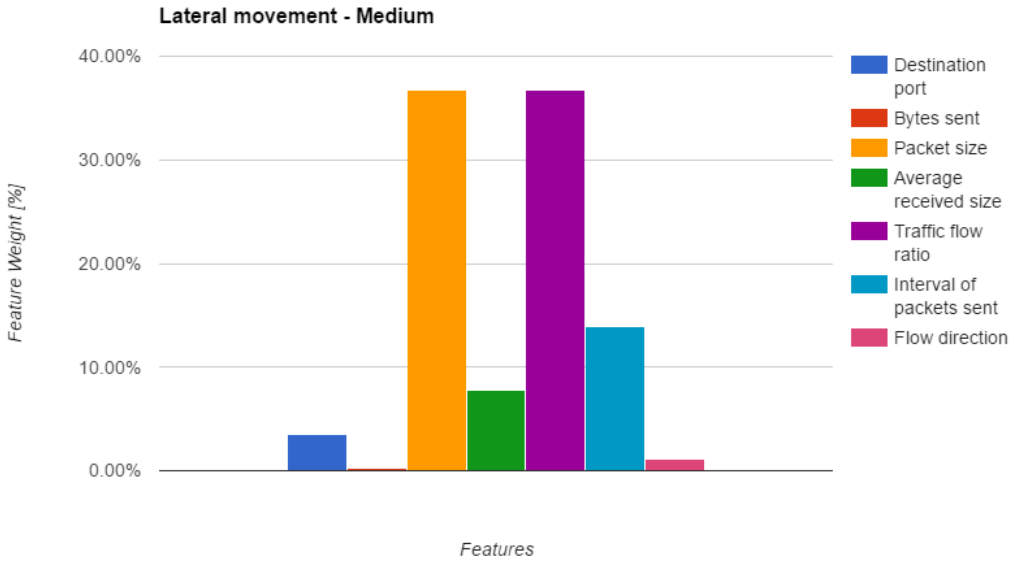

Figure 4.2: Feature weights for the scenario lateral movement with medium usage

Algorithm	Accuracy	FPR	FNR
ANN	0.9979	0.2148	2×10^{-4}
KNN	0.9998	0.0185	0
NB	0.9894	0.0185	1.05×10^{-2}
RF	0.9998	0.0185	0
DNN	0.9962	0.4111	6.6×10^{-5}
SVM	0.9998	0.0185	0

Table 4.3: Lateral movement High Usage Hosts

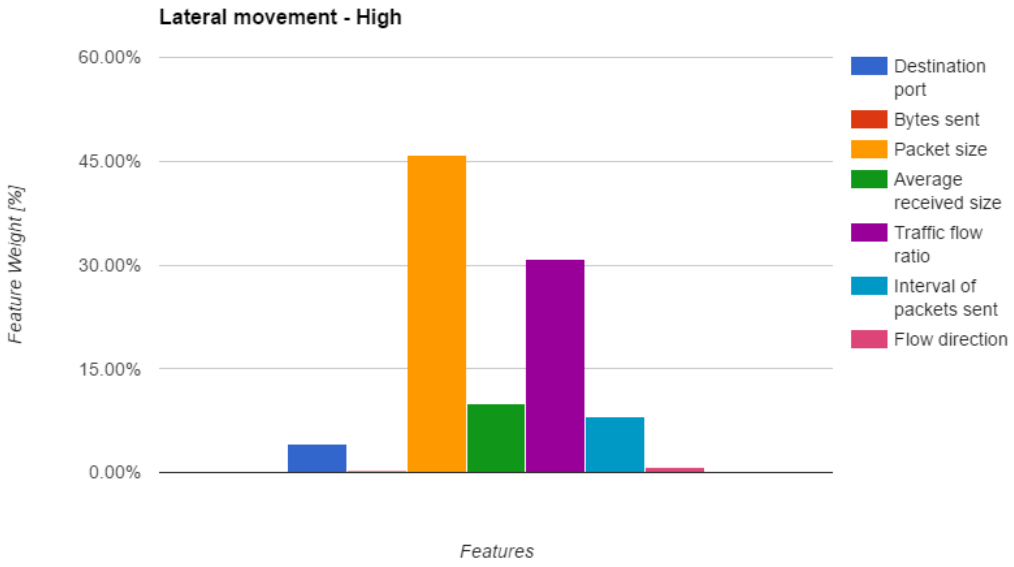

Figure 4.3: Feature weights for the scenario lateral movement with high usage hosts

4.2.2 Exfiltration

Table 4.4, 4.5 and 4.6 reflect the experiment when an internal host transmits data to a host outside of the network. The scenario contains exfiltration data of 100 MB, 1 GB, 3GB, 5 GB and 10 GB data, this data is injected into the real data.

In this experiment KNN, RF and SVM all succeeded in detecting the exfiltration sessions. ANN and NB have a FPR of 1.0 and 0.83 respectively, they were both unable to detect the exfiltration scenario regardless of host usage.

The Figures 4.4, 4.5 and 4.6 shows the feature weights for the algorithm RF, for the exfiltration scenarios. Where the *bytes sent* during the sessions is the most valuable feature.

Algorithm	Accuracy	FPR	FNR
ANN	0.9998	1	0
KNN	1	0	0
NB	0.9998	0.8333	0
RF	1	0	0
DNN	1	0.1667	0
SVM	1	0	0

Table 4.4: Exfiltration Low Usage Hosts

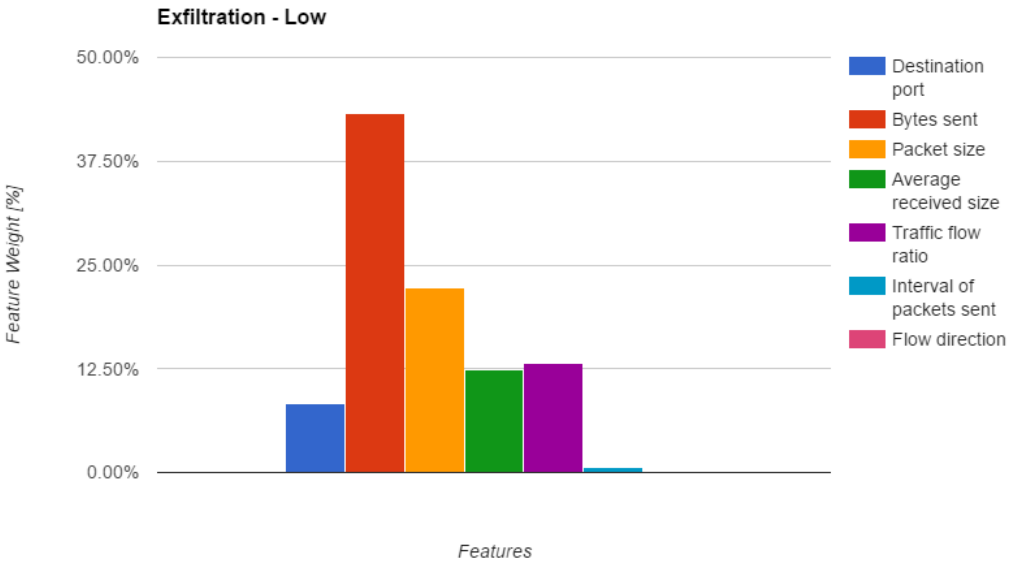

Figure 4.4: Feature weights for the exfiltration scenario with low usage hosts

Algorithm	Accuracy	FPR	FNR
ANN	0.9998	1	0
KNN	1	0.1667	0
NB	0.9998	0.8333	6.6×10^{-5}
RF	1	0	0
DNN	0.9999	0.3333	0
SVM	1	0	0

Table 4.5: Exfiltration Medium Usage Hosts

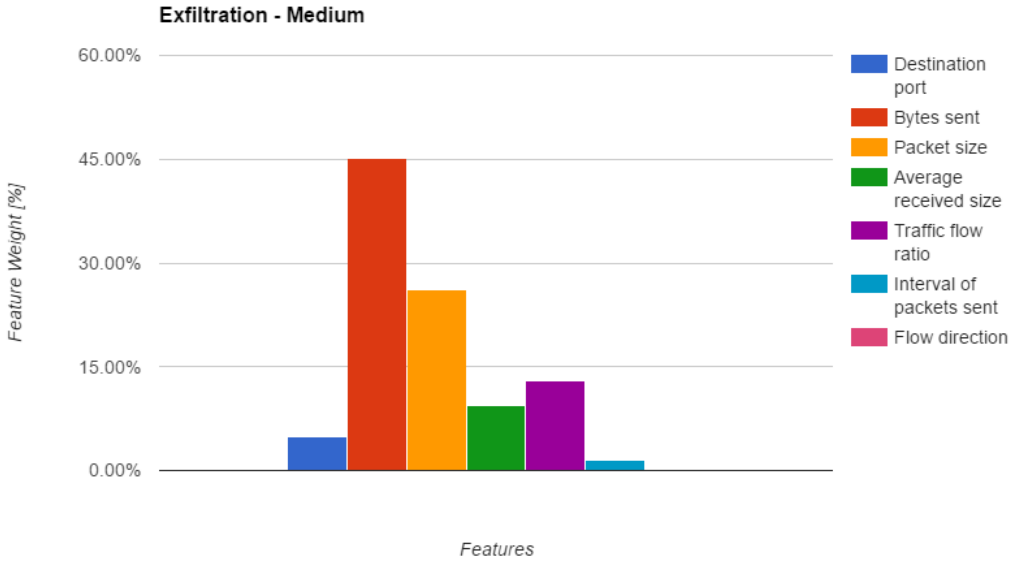

Figure 4.5: Feature weights for the exfiltration scenario with medium usage hosts

Algorithm	Accuracy	FPR	FNR
ANN	0.9998	1	0
KNN	1	0	0
NB	0.9997	0.8333	9.9×10^{-5}
RF	1	0	0
DNN	1	0.1667	0
SVM	1	0	0

Table 4.6: Exfiltration High Usage Hosts

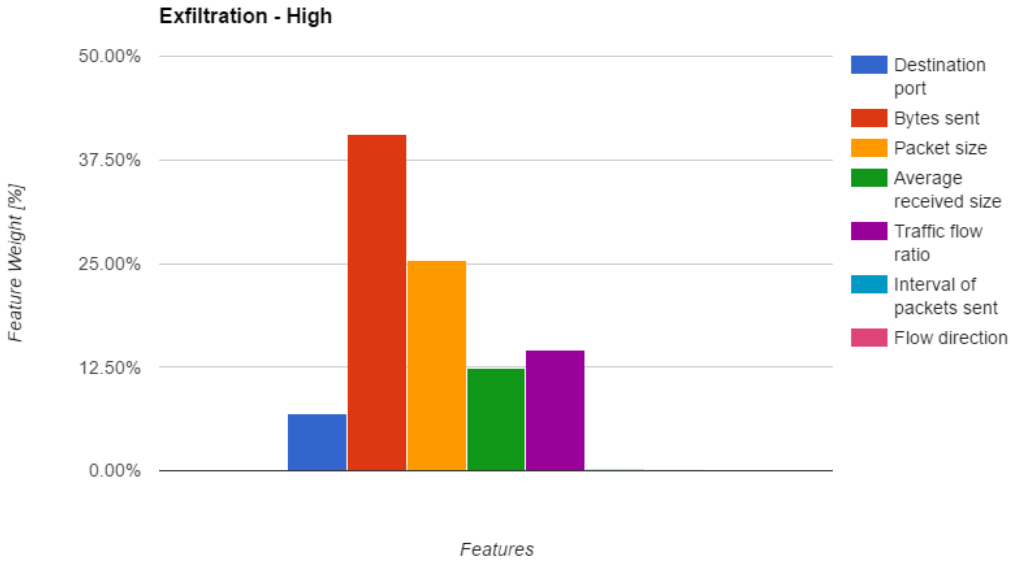

Figure 4.6: Feature weights for the exfiltration scenario with high usage hosts

4.2.3 Beacon

In Table 4.7, 4.8 and 4.9 results from the experiment with beacon data is found. In this scenario, an internal host sends data to an external host with a specific time between each packet. In this scenario results varied depending on the host usage. ANN, KNN, NB and RF were the most successful in the example with low host usage. RF was the only algorithm to provide a FPR below 0.05 when testing hosts with medium usage. SVM, RF and NB has high accuracy when testing hosts with high usage.

In Figure 4.7, 4.8 and 4.9, the feature weight is found. For this scenario, the feature *average received size* is significantly more valuable for the algorithm RF.

Algorithm	Accuracy	FPR	FNR
ANN	0.9993	1	0
KNN	1	0.0455	0
NB	0.9908	0	9.2×10^{-3}
RF	1	0	0
DNN	0.9993	1	0
SVM	0.9998	0.2273	0

Table 4.7: Beacon Low Usage Hosts

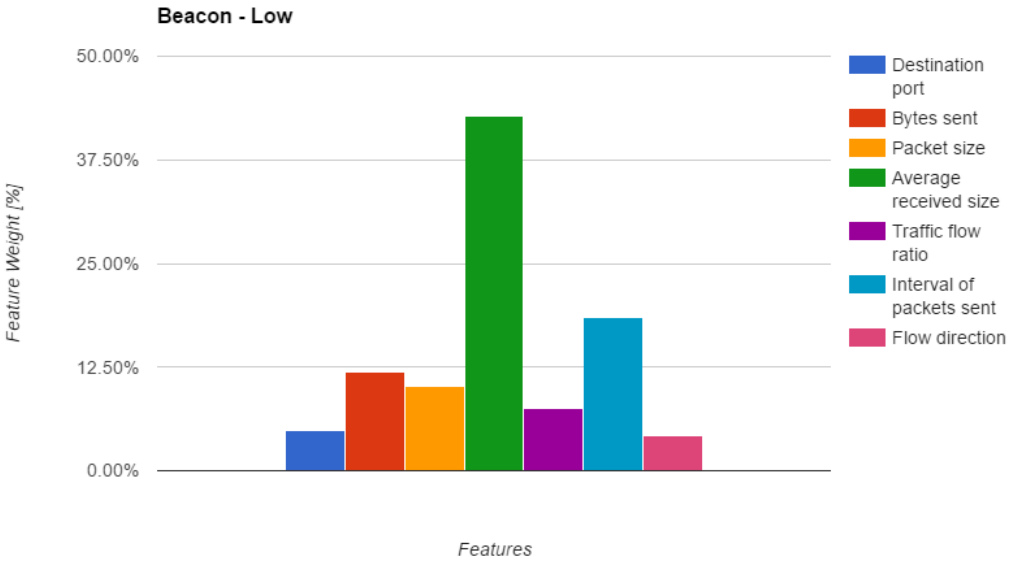

Figure 4.7: Feature weights for the beacon scenario with low usage hosts

Algorithm	Accuracy	FPR	FNR
ANN	0.9992	1	0
KNN	0.9994	0.7727	0
NB	0.9912	0.6818	8.4×10^{-3}
RF	1	0.0455	0
DNN	0.9993	1	0
SVM	0.9993	0.9091	0

Table 4.8: Beacon Medium Usage Hosts

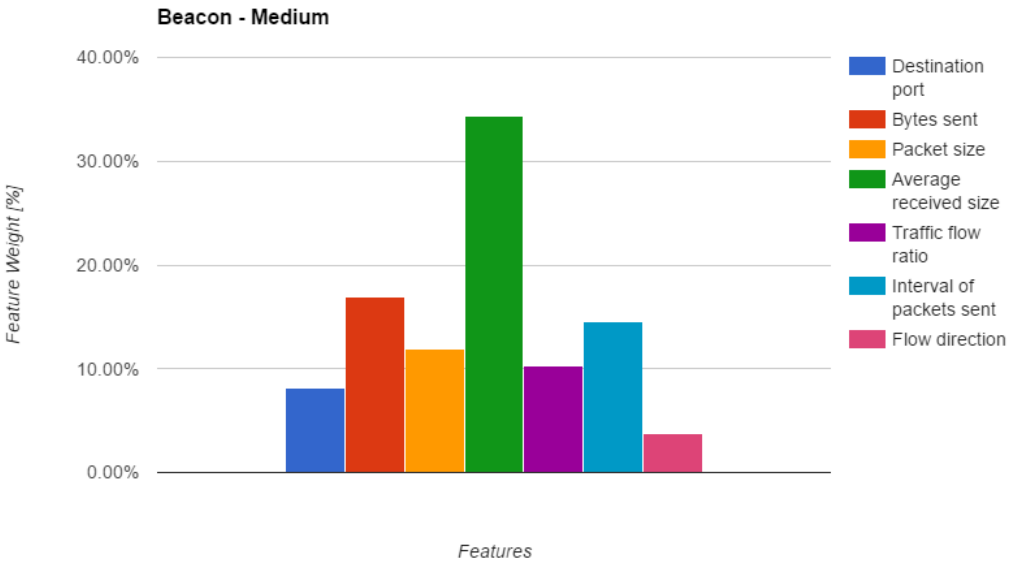

Figure 4.8: Feature weights for the beacon scenario with medium usage hosts

Algorithm	Accuracy	FPR	FNR
ANN	0.9993	1	0
KNN	0.9999	0.0909	3.3×10^{-5}
NB	0.9904	0	9.6×10^{-3}
RF	1	0	0
DNN	0.9993	1	0
SVM	1	0	3.3×10^{-5}

Table 4.9: Beacon High Usage Hosts

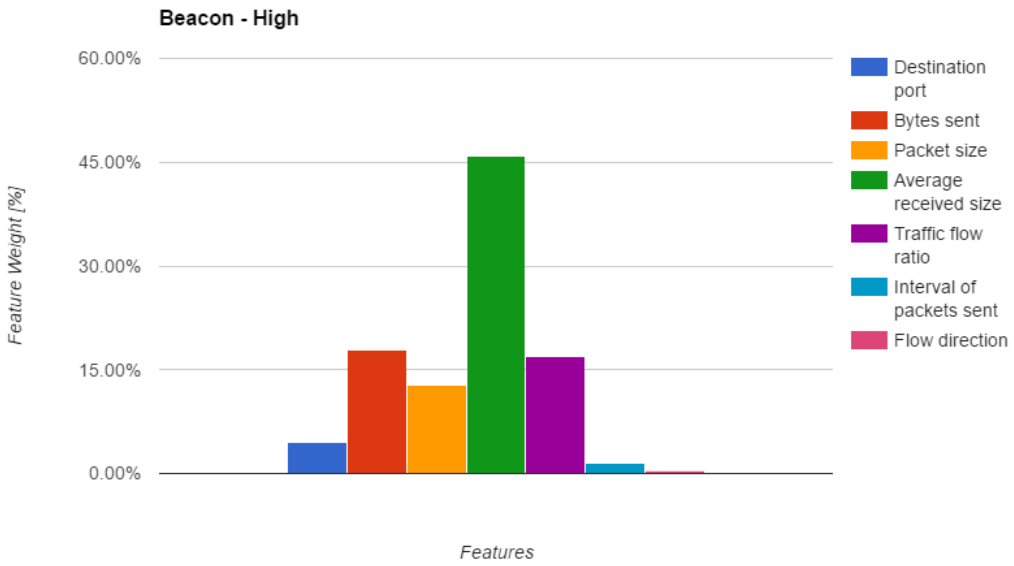

Figure 4.9: Feature weights for the beacon scenario with high usage hosts

4.2.4 Port scan

The Port scan scenario results can be found in table 4.10. The port scan is from one host and scanning was made for the 100 most common ports based on the nmap-services over the whole subnetwork, more details of the scan is described in section 3.2.

In this scenario KNN was the only algorithm to accurately detect the port scans, all other algorithms had a FPR of at least 0.15.

The feature weight for the Port scan scenario can be found in Figure 4.10. The feature *interval of packets sent* is considered the most valuable for RF.

Algorithm	Accuracy	FPR	FNR
ANN	0.9969	1	0
KNN	1	0	3.3×10^{-5}
NB	0.9582	0.1596	4.15×10^{-2}
RF	0.9994	0.1809	0
DNN	0.9957	0.8511	1.7×10^{-3}
SVM	0.9980	0.6383	0

Table 4.10: Portscan

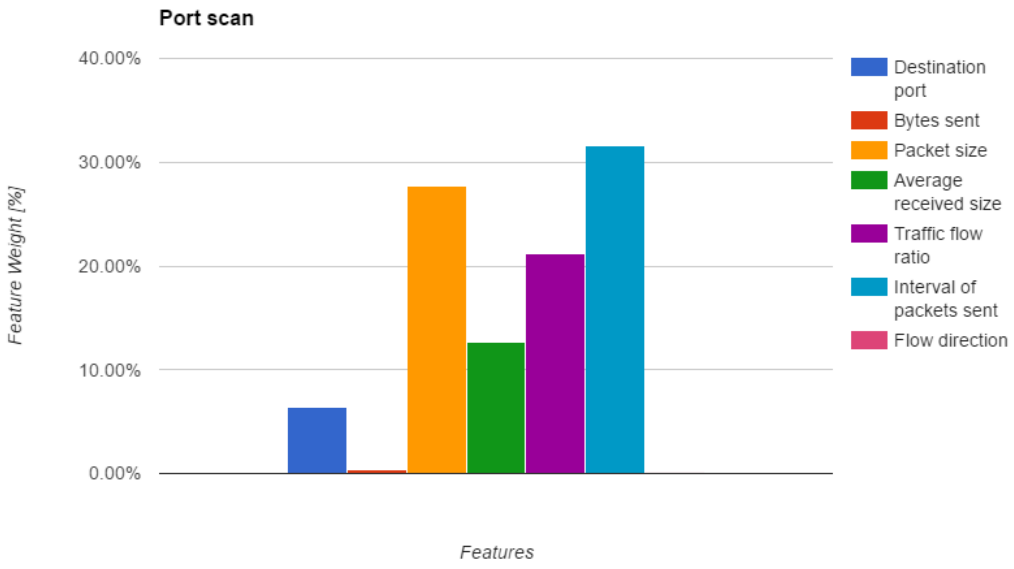

Figure 4.10: Feature weights for the port scan scenario with medium usage hosts

4.2.5 Multi-scenario

The Multi-scenario is a combination of lateral movement, beacon pings and exfiltration of a high volume of traffic, this test run will show how well the algorithms can detect more than one pattern. The result can be found in Table 4.11.

The results show that KNN, RF and SVM provided the most accurate results, with a FPR of 0.10-0.12 and an almost non-existent FNR.

The feature weight for the Multi-scenario can be found in Figure 4.11. The feature $flow\ direction$ have the highest weight for the RF.

Algorithm	Accuracy	FPR	FNR
ANN	0.9934	0.6469	1.7×10^{-4}
KNN	0.9988	0.1023	2×10^{-4}
NB	0.9900	0.9670	0
RF	0.9990	0.1023	0
DNN	0.9921	0.7921	3×10^{-5}
SVM	0.9988	0.1155	3×10^{-5}

Table 4.11: Combined test

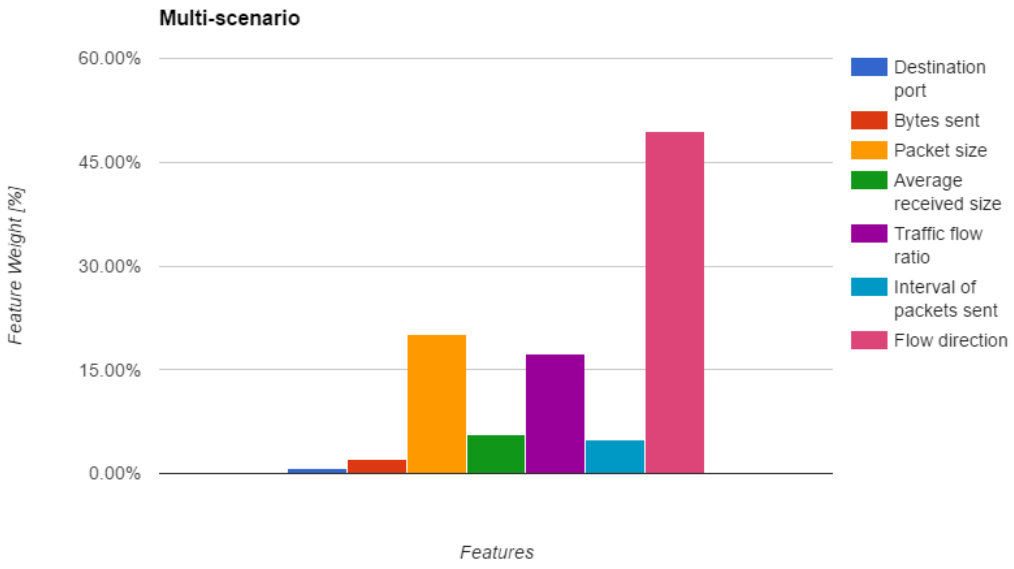

Figure 4.11: Feature weights for the multi-scenario

4.2.6 Algorithms Time Complexity Analysis

In Table 4.12 a comparison of the average time for each of the machine learning algorithms to train is shown. KNN, NB and RF could train in a matter of a second, whereas SVM and ANN used up to two minutes. The time of DNN was highly dependent on the optimization method. In this case, the focus was on gaining accurate results, therefore, over six minutes was spent on training.

The Table 4.13 shows a comparison of the average test time for each of the machine learning algorithms. KNN, NB and RF managed to predict the tests under one second, whereas both neural networks was just above one second. SVM took the longest time to predict the test data with twelve seconds in average.

Algorithm	Average Time/seconds
ANN	105.21
KNN	0.95
NB	0.03
RF	0.86
DNN	365.74
SVM	117.59

Table 4.12: Average Time for training

Algorithm	Average Time/seconds
ANN	1.23
KNN	0.82
NB	0.05
RF	0.22
DNN	1.12
SVM	12.21

Table 4.13: Average Time for testing

5 DISCUSSION

In this chapter, the method and results obtained from the experiment are analyzed and discussed. Some considerations regarding the challenges when studying APTs is mentioned. There is also some discussions of the related ethical and sustainable development issues that exist.

5.1 Feature Selection

In order to select appropriate features, several steps have been taken.

- A theoretical study of the features used in previous works.
- A statistical overview of the dataset.
- An evaluation of feature weights.

The emphasis in this thesis is on finding and using features that are simple to collect and difficult for the attacker to manipulate. The data is extracted from logs containing data on a per-session basis, as opposed to logs containing whole packets. If the full packets were analyzed, some interesting signature detection could be performed. This does however present two unavoidable problems. Collecting this data would be extremely resource demanding. The other issue is that the attacker can be expected to use encryption protocols to hide any content. In the dataset used in this thesis over 45% of the traffic is sent using various encryption methods.

In Figure 3.1, a statistical overview of the feature average packet size is presented, it is evident that there is a general trend of relatively small packets.

The feature weights are presented for the RF algorithm in each scenario. The feature weights indicate that depending on the scenario, different features are utilized to a higher extent.

5.2 Results of Experiment

The results of the experiment are presented using three values, accuracy, FPR and FNR. There is a significant majority of positive results in the test data, and yet detecting the negative cases is the most important in intrusion detection. This is an important observation as a high accuracy number may be accomplished without properly detecting the negative cases. This means that the FPR is the key factor to determining the success of the results and to give a complete picture of the performance.

The FNR is still an important value to consider, as any real situation of APT detection will contain a vast majority of positive results, and if they are presented as negative results the actual negative responses will be hidden among the false positives.

5.2.1 Lateral Movement

The scenario Lateral Movement shows a small difference between algorithms, in both FPR and FNR. The algorithms SVM, RF and KNN manage to outperform the rest of the algorithms in FPR, FNR and the total accuracy score.

There is a minor difference between the algorithms SVM, RF and KNN, but RF has the highest accuracy overall. If only FNR is considered, then SVM would be the best, but FPR is the most valuable metric for IDS and the Accuracy score has to be considered for overall success.

The feature weights for the algorithm RF shows that the *packet size*, *traffic flow ratio* and *interval of packetssent* were the most weighted features.

5.2.2 Exfiltration

Exfiltration results are even closer than the lateral movement, as most of the algorithms got both high results on FNR and the accuracy score. The main consideration when working with detection is aimed towards the FPR, where few algorithms misclassified most of the malicious traffic. As the amount of malicious traffic is minor, exfiltration is a tough scenario. The algorithm ANN misclassified all the malicious traffic, while NB misclassified 83%.

The only algorithms that manage to correctly classify the test data, are SVM and RF. As expected, the feature *bytes sent* had the highest feature weight. When hosts with high usage are considered, the *bytes sent* is less significant, and the *packet size* is instead of increased significance. This is to some extent expected as higher usage hosts will be more likely to send larger amounts of traffic.

5.2.3 Beacon

As the beacon can be hidden among the rest of the traffic, the results for RF is impressive as it manages to detect all malicious traffic and to classify the normal traffic, among the high usage hosts. Both ANN and DNN misclassified most of the beacon traffic. In this scenario the *average received packet size* was of the highest feature weight.

The expectation was that the *interval of packets sent* would score the highest, this feature was still considered as one of the highest in the low and medium host usage scenarios. In the experiment with high usage hosts, the *interval of packets sent* was barely considered. This is likely one of the reasons the algorithms failed to successfully classify those hosts.

5.2.4 Port Scan

The port scan scenario provides some interesting results as the algorithms had a difficult time achieving a high accuracy, except for KNN. The reason most algorithms had an issue with this scenario is likely the packet sizes being close to the normal values, this forces the algorithm to rely more on the time interval. In this scenario the *packet size* and *interval of packets sent* were the most important features.

5.2.5 Multi-scenario

The multi-scenario is the most difficult scenario, every algorithm has been tuned for best optimization as it requires classification of several threats. However due to the wide spread of malicious traffic it is understandable that none of the algorithms managed to classify all the malicious traffic correctly. The best algorithm is RF with the lowest FPR of 0.1023 and perfect FNR, the KNN algorithm is a close second with SVM right behind it.

The algorithms that managed to score the best of each of the previous scenarios also classified the multi-scenario best.

In this scenario $flow\ direction$ was of the highest weighted feature, followed by $packet\ size$ and $traffic\ flow\ ratio$. In this scenario considering flow direction to a high extent can be problematic, and it was likely a reason to the FPR of 10% for the best performing algorithms.

5.3 Real-world Application

The proposed solution can receive a high accuracy in the experiment setup, but in order to use it in a real world scenario some difficulties have to be considered. The first step is to train the algorithm on the network it will work on. In order to ensure that the network is not currently infected, analysis of the network has to be performed.

The algorithm also requires training on the malicious traffic, and while some behavior can be predicted it still needs to be scaled based on the network. Different networks will have different trends in regard to usage.

In a real-world application one advantage compared to the experiment model is that a host only requires to be detected once. The experiment achieves high accuracy by detecting each of the sessions containing malicious traffic. In an implemented model it is sufficient to find some malicious sessions of one host, which will highlight the host for further manual analysis.

5.4 Challenges

Several difficulties exists when studying APTs, in this section some of these are discussed.

5.4.1 Test Data

When analyzing APTs, one of the main difficulties is finding a dataset, partly due to the nature of the APT being very flexible. Another part of this difficulty is that organizations affected by APTs are normally reluctant to disclose these occurrences, let alone to provide data for scientific studies.

In this thesis the APT data is crafted then injected in real network traffic, the results are dependent on how similar the crafted APT data is to actual cases of an APT.

5.4.2 Base Rate Fallacy

When attempting to detect an APT, it can be expected to analyze millions of network flows without any attack of this type. This means that there is a risk of an overwhelming amount of false negatives, even if the rate of false negative is below 1%.

5.5 Ethics and Sustainable Development

APTs represent a modern method of attacking organizations, it is likely to become more frequently used in the future. Being able to detect APTs is extremely important, not only to prevent the currently ongoing data theft, but also to be able to study the progress attackers are making. This method of attack can be utilized to infiltrate infrastructure organizations and other government sections. Continuous study of this subject is required to be able to prevent the attacks in the future.

There can be an ethical argument that publicly presenting methods of detection can aid the intruders more than the protection systems. Bypassing known detection methods is certainly

a simpler task than when the detection method is unknown. However, any modern detection system should be constantly changing. Scientific work should be used as an additional resource of inspiration, but never solely be relied upon as a foolproof method.

5.6 Time Complexity

The time of each algorithm is found in Tables 4.12 and 4.13, it has not been of high priority in this thesis, and little to no optimization has been done to consider time. Table 4.12 contains the average time of training for each algorithm. There is one outlier in this table, DNN, which has a significant increase in time as compared to all other algorithms.

Table 4.13 contains the average time of test for each of the algorithms. In this table there is one outlier, SVM, as the only one over ten seconds. The main thing to take in consideration when looking at time complexity, is if the training or testing time is high enough to cause issues. As the results show most algorithms spend roughly one second to test, they can be used to analyze network traffic, even if it increases in size by a significant amount.

6 CONCLUSIONS

In this thesis detection of APTs has been performed using six different machine learning algorithms. Detection is based on three different scenarios, lateral movement, beacon traffic and data exfiltration.

When comparing the algorithms it is evident that RF, KNN and SVM achieved the highest accuracy. The results showed that by specifying one scenario the algorithms could perform the classification successfully, often with none or close to no flaws. When classifying the combined features the best FPR was 10.2%.

The most significant features were highly dependent on the scenario and to some lesser extent the regular usage of the host analyzed. In the lateral movement scenario the most significant features were $packet\ size$ and $traffic\ flow\ ratio$. In the exfiltration scenario $bytes\ sent$ was the most impactive feature, $average\ received\ size$ was the most significant feature for the beacon scenario. The features selected is highly dependent of which scenario has to be considered, this is the reason a combined scenario will have a lower accuracy.

Our conclusion is that it is possible to use machine learning algorithms, especially KNN, RF and SVM, to detect APTs. It is important to specify the scenario to detect, and potentially running multiple algorithms using various features for different scenarios to achieve the highest possible accuracy.

7 RECOMMENDATIONS AND FUTURE WORK

Detecting APTs using network flows and utilizing machine learning algorithms has been possible in our experiments, but there are still areas to explore. One of the topics we were unable to consider was how actions change during the time of day. With a dataset from a longer time period, this is an interesting topic to investigate in future works.

Another area to explore is how to utilize this method in a live network scenario, when testing is performed in real-time. A possibility is to use the more successful algorithms from this thesis and compare their results for a more thorough presentation of results.

As we decided to limit the training time for the deep learning algorithm, it is possible to improve DNN by allowing it to train for a longer period. This would involve the risk of over-fitting for the dataset.

REFERENCES

[1] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review", *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013. DOI: 10.1016/j.jnca.2012.09.004.

[2] J. Steer, "The gaping hole in our security defences", *Computer Fraud & Security*, vol. 2014, no. 1, pp. 17–20, 2014. DOI: 10.1016/s1361-3723(14)70009-0.

[3] E. Cole, *Advanced persistent threat*, 1st ed. Syngress, 2013.

[4] R. Brewer, "Advanced persistent threats: Minimising the damage", *Network Security*, vol. 2014, no. 4, pp. 5–9, 2014. DOI: 10.1016/s1353-4858(14)70040-6.

[5] M. Auty, "Anatomy of an advanced persistent threat", *Network Security*, vol. 2015, no. 4, pp. 13–16, 2015. DOI: 10.1016/s1353-4858(15)30028-3.

[6] I. Friedberg, F. Skopik, G. Settanni, and R. Fiedler, "Combating advanced persistent threats: From network event correlation to incident detection", *Computers & Security*, vol. 48, pp. 35–57, 2015. DOI: 10.1016/j.cose.2014.09.006.

[7] M. Marchetti, F. Pierazzi, M. Colajanni, and A. Guido, "Analysis of high volumes of network traffic for advanced persistent threat detection", *Computer Networks*, vol. 109, pp. 127–141, 2016. DOI: 10.1016/j.comnet.2016.05.018.

[8] J. Bell, *Machine learning*, 1st ed. John Wiley & Sons, Inc., 2015.

[9] S. Theodoridis, *Machine learning*, 1st ed. Academic Press is an imprint of Elsevier, 2015.

[10] s. Gollapudi and L. V, *Practical Machine Learning*, 1st ed.

[11] M. Zamani and M. Movahedi, *Machine learning techniques for intrusion detection*, 2017. [Online]. Available: https://arxiv.org/abs/1312.2177v2.

[12] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines", *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, DOI: 10.1109/ijcnn.2002.1007774.

[13] P. Laskov, P. Düssel, C. Schäfer, and K. Rieck, "Learning intrusion detection: Supervised or unsupervised?", *Image Analysis and Processing – ICIAP 2005*, pp. 50–57, 2005. DOI: 10.1007/11553595_6.

[14] S. Dua and X. Du, *Data mining and machine learning in cybersecurity*, 1st ed. CRC, 2011.

[15] A. Cufoglu, M. Lohi, and K. Madani, "Classification accuracy performance of naive bayesian (nb), bayesian networks (bn), lazy learning of bayesian rules (lbr) and instance-based learner (ib1) - comparative study", *2008 International Conference on Computer Engineering & Systems*, 2008. DOI: 10.1109/icces.2008.4772998.

[16] A. Krogh, "What are artificial neural networks?", *Nature Biotechnology*, vol. 26, no. 2, pp. 195–197, 2008. DOI: 10.1038/nbt1386.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. DOI: 10.1038/nature14539.

[18] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. e. a. Lanctot, "Mastering the game of go with deep neural networks and tree search", *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. DOI: 10.1038/nature16961.

[19] A. Gron, *Hands-on machine learning with scikit-learn and tensorflow*, 1st ed. O'Reilly Media, 2017.

[20] A. S. M. Sohail and P. Bhattacharya, "Classification of facial expressions using k-nearest neighbor classifier", *Computer Vision/Computer Graphics Collaboration Techniques*, pp. 555–566, DOI: 10.1007/978-3-540-71457-6_51.

[21] L. Breiman, "Random forests", *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. DOI: 10.1023/a:1010933404324.

[22] G. Biau and E. Scornet, "A random forest guided tour", *TEST*, vol. 25, no. 2, pp. 197–227, 2016. DOI: 10.1007/s11749-016-0481-7.

[23] W. S. Noble, "What is a support vector machine?", *Nature Biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006. DOI: 10.1038/nbt1206-1565.

[24] F. Skopik, G. Settanni, R. Fiedler, and I. Friedberg, "Semi-synthetic data set generation for security software evaluation", *2014 Twelfth Annual International Conference on Privacy, Security and Trust*, 2014. DOI: 10.1109/pst.2014.6890935.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[26] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: Experiences from the scikit-learn project", in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[27] G.-B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks", *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 274–281, 2003. DOI: 10.1109/tnn.2003.809401.

[28] S. Ruder, *An overview of gradient descent optimization algorithms*, 2017. [Online]. Available: https://arxiv.org/abs/1609.04747.

[29] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

Blekinge Institute of Technology, Campus Gräsvik, 371 79 Karlskrona, Sweden