UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR IT-SICHERHEIT

Anomaly Detection in Login Data

Anomalieerkennung in Logindaten

Bachelorarbeit

im Rahmen des Studiengangs
IT-Sicherheit
der Universität zu Lübeck

vorgelegt von
Johannes Liebenow

ausgegeben und betreut von
Prof. Dr. Thomas Eisenbarth

mit Unterstützung von
Ida Bruhns

Lübeck, den 27. Oktober 2019

Abstract

Nowadays, most of the companies have to handle critical data on the basis of customer information and internal secrecy. It often leads to the necessity to establish some kind of employee surveillance which monitors diverse applications and activities as a type of logging system. These resulting log protocols can be analyzed by an intrusion detection system which supports multiple detection mechanisms.
During this thesis, log protocols provided by an external company will be analyzed and in the first place an anonymization tool will be developed to assure the necessary level of privacy in regard of the personal data within the protocols. The currently used intrusion detection system which is based on a rule-based detection shall be reimplemented with a better adaption to the provided protocols which is followed by the search for new methods to find unknown anomalies.
The developed prototype will consist of two parts: A signature detection part which is based on the current system of the external company and an anomaly detection part which includes a statistical analysis and the appliance of algorithms using artificial intelligence. The result is a hybrid system which is able to find known anomalies with a rule-based detection system and unknown anomalies on the basis of deviations in employees' workflows and handmade anomalies.

Abstract

In der heutigen Zeit müssen die meisten Unternehmen kritische Daten handhaben, die sich aus Kundendaten und allgemeinen Firmengeheimnissen zusammensetzen. Dies führt oft zu der Notwendigkeit, eine Überwachung der Mitarbeiter einzuführen, die dann als System zur Zusammenführung und Auswertung von Logdaten fungiert und dazu dient, die Aktivitäten verschiedenster Anwendungen und Prozesse aufzuzeichnen. Die so entstehenden Logdaten in Form von Protokollen können von einem Intrusion Detection System anhand verschiedenster Methoden ausgewertet werden.
Während dieser Arbeit werden Log-Protokolle von einem externen Unternehmen zur Verfügung gestellt und ausgewertet, jedoch wird vorher ein Anonymisierungstool entwickelt, um den nötigen Grad an Privatsspähre der enthaltenen Informationen zu gewährleisten. Das Intrusion Detection System, welches von dem externen Unternehmen genutzt wird, um die Protokolle auszuwerten, wird neu implementiert, damit eine bessere Anpassung an die gegebenen Protokolle erreicht werden kann. Danach folgt die Suche nach neuen Methoden, um das Auffinden von bisher unbekannten Anomalien zu ermöglichen. Ein Prototyp wird entwickelt, der im Kern aus zwei Komponenten besteht: Eine Signaturerkennungs-Komponente basierend auf dem jetzigen Analysesystem des externen Unternehmens und eine Anomalieerkennungs-Komponente, die eine statistische Auswertung beinhaltet und versucht, verschiedenste Methoden der Künstlichen Intelligenz zur Anwendung zu bringen. Das Endergebnis ist ein hybrides System mit der Möglichkeit, bekannte Anomalien durch ein regelbasiertes System zu entdecken und anhand von Abweichungen im normalen Nutzerverhalten und durch handgemachte Anomalien bisher unbekannte Anomalien zu detektieren.

Erklärung

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

Lübeck, 27. Oktober 2019

Contents

1 Introduction

1.1 Motivation

In the present time, most of the work in a company is done by computers or employees sitting in front of a computer. To guarantee a fast and reliable data exchange or in general some sort of interconnected work, there is often the need for a company-intern network which could have a connection to the internet. If there is no outward connection then there are still some threats which have to be considered. On the one hand it could be the stereotypical hacker who tries to fulfill a malicious goal and on the other hand there are some threats which come from the inside like an internal who might know the system and the company-specific circumstances much better. These threats can or will lead to an attack which often entails a lot of damage to the company in the sense of high costs and required time to undo or at least minimize the consequences. Besides a real attack there is also the interest in detecting technical malfunctions.
One type of a defense against attacks is an intrusion detection system in combination with log data analysis. Every activity gets monitored which is called logging and then the intrusion detection system analyzes these logs and tries to find unusual actions which can be part of an attack or malfunction. Of course, this system cannot prevent attacks but rather be an early-warning system so that attacks can be stopped and their source found before the whole damage is done.

1.2 Task and Goals

This thesis aims to achieve an application for automated protocol evaluation. The focus lays on anomaly detection which includes technical malfunctions, potential intrusions and malicious behavior. This is realized by a prototype containing a hybrid system which consists of a signature detection and an anomaly detection part. Especially latter creates some sort of user profile to improve its results, but that is why the protocols are only used after they got anonymized and therefore, an anonymization tool gets developed to provide the necessary level of confidentiality.
The signature detection tries to find atypical or malicious behavior by analyzing protocols under predefined rules. That means every time a user satisfies a rule, he arouses an anomaly. A few rules are already given which were created with a practical orientation so

that most of the critical cases, a brute force attack for instance, are covered and the false alarm rate is not too high.
The goal of this thesis is also about finding and classifying unknown anomalies by analyzing the given protocols. During the process two different methods are used. First, a statistical analysis which helps to find anomalies by comparing protocols among each other and then different machine learning algorithms get applied, since they can handle and analyze the protocols completely different than any human could do. When new anomalies are discovered, new rules can be created to intercept these anomalies.
The application has to handle protocols of several ten thousands lines in total. It does not have to work in real time but rather is a tool which can be used on demand. In practice, it analyzes weekly submitted protocols from various clients. If an anomaly occurs, the specific employee can be stated, because the output shows the event and the corresponding lines from the uploaded protocols. The resulting prototype should be at least as efficient as the currently applied system, which is a necessary criterion for the development process.

1.3 Circumstances

1.3.1 Statutory Basis

The protocols which are used during this thesis consist of log data which was recorded from each client by itself, but the underlying log program is consistent in the whole company. The log data is basically a recording of every single user or employee activity at their individual workplace. This means that the protocols contain profoundly personal data, because e.g. even the working time can be derived. The question which appears at first is whether or in which cases this kind of employee surveillance is allowed. The answer is a basic issue in the Work Constitution Act, because it says as follows:

> "The works council shall have a right of co-determination in the following matters in so far as they are not prescribed by legislation or collective agreement: [...] the introduction and use of technical devices designed to monitor the behavior or performance of the employees."
> - §87 S6 Works Constitution Act

In other words, the works council or the employee representative has to agree to the activity logging process. This decision is based on a consideration between the protection of every single employees' privacy and the benefits of analyzing recorded activities. So when a company has to handle personal data of their clients, works in critical sectors or has some other reasons to ensure higher security standards, some kind of employee surveillance is often necessary.

Since these protocols get analyzed during the thesis, they have to be anonymized so that there is no leak of personal information and they can be used without restrictions, thus an anonymization tool gets developed. This tool can pseudo-anonymize the log data to ensure the necessary degree of privacy. Especially the user ID and the username have to be anonymized, because they can be used to surly trace back an individual activity to a specific user.

1.3.2 Protocols from Log Data

The protocols itself have a precisely specified format which gets explained in the following section. All protocols are in the CSV-format which stands for "comma-separated values" and provides a simple tabular data format separating columns with a special character like a semicolon (delimiter) and rows with a line break. The first four lines of a protocol are always filled with a special header containing meta-data which has to be skipped, because it is not an important part of the later analysis. The fifth line specifies the attributes whereby every column stands for a single attribute:

Datum	The date when the specific process happened.
Zeit	The exact time when the specific process happened.
User	The unique username of the user who started the process.
Timestamp	This alphanumeric string stands for the explicit number of the used software.
ID	The unique ID of the user who performed the activity.
Prozess	This attribute shows the name of the process which was executed by a user.
FKL-Prozess	This is a short number which indicates if the process has succeeded or failed.
RC-Prozess	This number gives a more exact hint if the process failed.
Backend Laufzeit	The time the specific process needed on the backend system.
Laufzeit Gesamt	The total time from the process execution to its end.

In general, every line in a protocol represents one event or activity of a specific user. Multiple processes are possible: A login, a logout and a password change. In the process, a user always performs an action in conjunction with a single application. There are also different types of users (e.g. technical users, employees, administrators etc.). The individual user type is represented by the first character in the attribute "User". In addition, every line also contains an indication whether the process succeeded or failed.

Table 1.1: An exemplary extract from one of the provided protocols.

Datum	Zeit	User	Timestamp	ID
12.09.2018	05:00:05	S_DEMO1	2012-13-20-19.12.45.836103	93271
12.09.2018	05:36:47	S_DEMO1	2012-13-20-19.12.45.836103	93271
12.09.2018	08:01:00	S_DEMO2	2003-03-31-21.84.32.861023	89381
		...		

Prozess	FKL-Prozess	RC-Prozess	Laufzeit Backend	Laufzeit Gesamt
LOGIN	2	0124	0.005	0.010
LOGOFF	2	0124	0.005	0.010
PASSWORT_AENDERN	2	7312	0.010	0.030
		...		

1.3.3 Status Quo

The prototype getting developed during this thesis is based on the current system which shall be replaced. It is an online tool which needs every single protocol to be uploaded in advance. Then follows a rule-based analyzing process with the following rules:

- Identification of the misuse of technical users if the amount of fail logins > 0
- Identification of the misuse of user accounts if...
  ... there are fail logins outside the normal working time
  ... the amount of fail logins exceeds a specific threshold once a month
  ... there is no correct login within 5 minutes after a fail login
- Visual check of the distribution of different user behaviors
- Recording of admin activities due to super user rights
- Identification of password changes of special user groups
- General check for a rapid drop of user activities per hour

The general problem which appears is that the current detection system is some kind of superficial, because of two reasons. First, the tool itself is not designed for these protocols in the proper sense and second, the analysis is equal for all clients which means there is no consideration for the clients' properties in detail. Additionally, the analysis itself is very time-consuming which accompanies with a complex handling and graphical user interface (GUI). Therefore, the development of the prototype tries to achieve a few improvements which shall ensure a perfect adjustment to the provided protocols.

1.3.4 Attack Vectors

The recorded activities of the employees or users take place in an internal network which is isolated from the outer world of the internet. That is why the chance of a hacker attack from outside is almost zero. The mass of possible attacks can only be performed by insiders. An employee could do some malicious transactions for instance and such cases happen but they are very rare and cannot be retraced in the log-in data. In view of this, the prototype is focused on the detection of attacks and malfunctions which can be derived from the log data.
The fact that there was no detection of any attack since the beginning of the log process complicates the anomaly detection development, because knowledge about malicious attacks makes it much easier to build up an efficient detection mechanism. But of course, there is also no negligible chance that anomalies appeared in the past without any recognition.
Nevertheless, the biggest problem is called ambiguity. If an anomaly appears it has to be analyzed and then it has to be decided whether it is an attack, a malfunction or just a false alarm. A fail log in is a perfect example. After various log in fails users get blocked and cannot log in even if they type in their correct log in credentials. As a result, some users have a lot of log in fails which are not necessarily an anomaly. But the intrusion detection system would raise the alarm and in the worst case, this behavior gets labeled as a bruteforce attack.
Even though there are attacks which are possible and have to be detected:

Bruteforce Attack This sort of attack is a very basic one but still can be very efficient. If an employee wants to use the account of his colleague he can guess his password e.g. by testing every possible combination of words and numbers. A bruteforce attack would lead to a very high rate of fail logins on the victim's account and this is also a criterion for the detection.

Unauthorized Password Change Whenever a user exceeds the limit of fail logins he gets blocked and can not log in successfully until the blockade is repealed which can be done by calling the technical support. And this is exactly the point which can be exploited: If a user somehow manages to masquerade himself as his colleague, then he could change the accounts password. Of course, this could be detected by the colleague itself, but the intrusion detection system can only detect that kind of attack if the malicious actions diverge significantly from the normal workflow.

Privilege Escalation Privilege Escalation describes the scenario when an attacker e.g. an employee exploits his rights or makes use of rights which were not entitled to him in the first place. This usually allows more powerful or critical actions and can only be detected if the specific user diverges from his normal work patterns. To achieve the possibility of performing an attack like privilege escalation, another attack has to be performed as a first step in the majority of cases.

2 Related Work

2.1 Anonymization

The usage of sensitive or private data requires anonymization so that the privacy of users is assured [Sin18]. However, it is important that the internal structure remains, because there are attributes, for instance a username, which have to be distinguishable after the anonymization process. Especially when the data sets are used further on, the identification of single items is necessary. Besides it has to be clear whether an attribute is a direct identifier so it can be used to identify a single line or an attribute is an indirect identifier which means that one needs more than one attribute to identify a single line [Eis18]. In addition to that there can be the case that an attribute is not important at all which should be noticed as well, because less attributes could lower the computing time [Sin18]. In order to guarantee a good anonymization there are the following possibilities:

Character Masking Sometimes a part of an attribute is not necessary and the attribute itself can be identified without this special part but without violating the criterion of recognizability [Rep17]. This can be done by cutting out or replacing a part of the attribute (e.g. "TEST-123" $\rightarrow$ "TEST-XXX"). However, it is not easy to determine the perfect length and place of the mentioned part, because it can happen very quickly that the item cannot be clearly identified afterwards [Sin18].

Pseudoanonymization In order to guarantee a good anonymization the data sets can be anonymized with a method called pseudoanonymization. Thereby, the whole attribute or a part of it can be replaced with a string which gets explicitly generated for that attribute [Rep17]. Then it is possible to identify the item as before, but one is not able to figure out the original string. One method is based on hash functions. Every attribute gets hashed and obtains its personal string. A general problem occurring is the so-called collision resistance, which means that there is a chance that two attributes get mapped to the same string [Eis18]. This problem can be minimized by using an accepted and not deprecated hash function. Another way for pseudoanonymization is encryption. Every attribute gets encrypted and hopefully, it has its own unique string. In doing so one has to be very careful with the used key and information about the encryption, because an intruder should not be able to recover the original data sets [Sin18].

Data Perturbation Anonymization can also be obtained by data perturbation. If numbers and continuous variables are used, they can be interfered in some way [Sin18]. This can be achieved by bringing some noise into the data which means that the attributes get modified. For instance, a static or random value can be added to the attributes. It is important that the noise does not change the data too much, because an item should not be identified as another one.

Considering the structure of the given data sets, which can be seen in Section 1.3.2, some methods can be directly excluded. This includes the usage of K-Anonymity which is a way for data anonymization via suppression and generalization [Eis18]. Data perturbation is also a rather unsuitable method, because the data sets contain almost exclusively discrete attributes where noise is an unusual approach [Sin18].
When indistinguishability is required, the most promising approach consists of pseudoanonymization, because every username can uniquely be retraced. In the process it has to be considered that pseudoanonymized data still counts as personal data, since it is theoretically possible to re-identify the data. Additionally, suppression can be used to eliminate unnecessary attributes which would just slow down the further processing [Rep17].

2.2 Artificial Intelligence

With the rapid increase of the size of datasets which have to be analyzed and classified, artificial intelligence (AI) underwent a new peak phase, not least because of the big data discussion [Bea17]. However, the whole complex of AI is still at a very basic level (Narrow Artificial Intelligence) and is not even close to everyone's futuristic thoughts of an independent and humanoid artificial intelligence (General Artificial Intelligence). A narrow artificial intelligence which is also called the weak form of AI can handle one specific function or task. One of the best examples is Google's AlphaGo [Dee], because it beat the European Go champion in 2016, but it is still a weak AI. AlphaGo has all characteristics of an AI, since it is very powerful in playing Go and can compete with the worlds' best players, but the big difference between AlphaGo and a strong AI is that it can only play Go and nothing else. It sounds a bit pejorative, but a strong artificial intelligence would have the ability to learn Go and could express itself to others in an appropriate language for instance. There are many more application fields for AI which are for example Google's search engine, image recognition software and Siri inside Apple's iPhone.
In general, there exist three basic terms in the context of AI: artificial intelligence itself, machine learning where algorithms and methods are often referred to and deep learning including the very popular neural networks.

The terminology gets mixed up oftentimes and can be classified as follows:

> “Artificial intelligence is a set of algorithms and intelligence to try to mimic human intelligence. Machine learning is one of them, and deep learning is one of those machine learning techniques.”
> - Frank Chan [Che17]

Machine learning techniques are very powerful and also get used during this thesis, because they have a variety of usages which include among other things classification and prediction techniques as well as regression techniques. The big advantage lays in the very basics of machine learning, because once it gets the prepared datasets it learns from them and there is no need for hundreds of thousands lines of code which would be needed to solve the problem otherwise. But whenever one wants to apply machine learning techniques on data, it has to be considered that noisy or large-scaled datasets can pose some kind of challenging task. The so-called “Curse of Dimensionality” describes the difficulties for AI methods to handle high dimensional datasets [Spr14]. It is easy to imagine that in a space with a massive amount of dimensions, whereby each dimension represents a feature in the given data, every data point can be treated as an anomaly, because of the incredible size of the space. In such high dimensional spaces it is very hard to compute some kind of baseline, because the amount of training data has to be very high. In addition, the distance measurement also becomes a problem due to the fact that in high dimensions most of the data points reside in corners which makes the learning process much harder. Every dataset can be assigned to one of two classes. The first one is called labeled data or in terms of algorithms supervised learning [Nie15]. This class includes all algorithms which can derive knowledge out of data consisting of explicitly labeled data points, e.g. demonstrating whether a data point is an anomaly or not. The other class handles unlabeled data, thus it is called unsupervised learning. These datasets do not contain any information about the single class affiliation and that is why the algorithms try to find some sort of classes or separations by themselves [bui19]. The next subsections give a slight overview about the most promising approaches considering the protocols which are used during this thesis.

2.2.1 Neural Networks

The most popular deep learning approach of artificial intelligence are probably neural networks. At the same time, they seem to be a very complex and scientific construct for most of the people. But quite the contrary, the theory behind neural networks is very basic. As the name suggests, the biology of the brain was used as an inspiration. A neural network as well as the brain are based on neurons. A neuron itself represents a binary

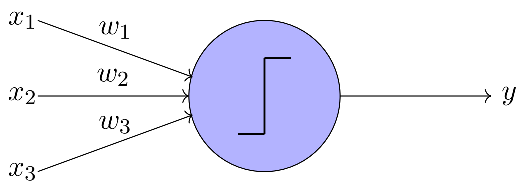


Figure 2.1: The structure of a neuron with the input vector $x = \{x_1, x_2, x_3\}$, their respective weights w_1, w_2, w_3, the decision function inside the neuron and the binary output y.

classifier which has an input and an output [WL90]. Inside the neuron there is some kind of decision function which can be trained by various algorithms. These training algorithms adjust the special coefficients called weights which are included in the decision process, but the output always consists of a binary decision.

A neural network has multiple layers: an input layer, an output layer and different hidden layers which contain the whole learning progress [Nie15]. Every layer consists of neurons which receive their input from the previous layer and send their output to the next layer. It can be seen as some sort of abstract representation learning, because the neural network receives raw data and adjusts it to the input of the abstract function which has to be learned. The deeper the layer the more abstract features are considered and that is why very complex function can be learned such as the automated recognition of handwritten numbers [Che17]. Before a function works almost correctly there has to be a training phase, whereby one round of training and error correction is called an epoch. The error correction is a very complex issue, because a neural network contains a lot of neurons and weights which have to be adjusted in every epoch. The function which describes the error is called loss-function and in the best case it is a convex function, since this kind of function has one minimum which is the global minimum at the same time [Ped14]. The global minimum symbolizes the point of the lowest error which of course every constructor of a neural network wants to reach.

2.2.2 Clustering

Anomalies can be seen as outliers which mean they can differ from the normal group of data. Clustering is a technique which is used to find these outliers, because it helps to find patterns in data with many dimensions [PP07]. Clustering uses every attribute as a dimension and spans a room where every single item with all its attributes represents a single point in the cluster of data sets. Interpreted in terms of an n dimensional Cartesian coordinate system, the cluster of data sets can be seen as a accumulation of points and an anomaly is a point which is for instance far more away from the center of this specific

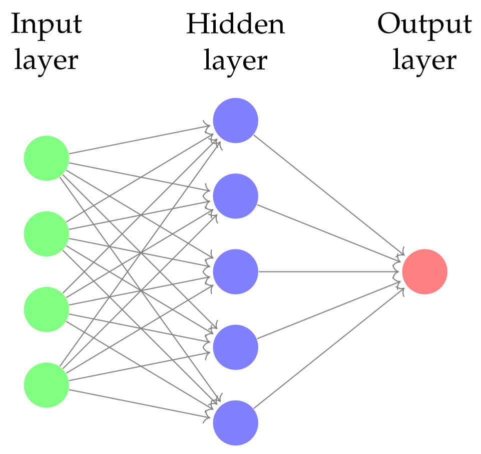


Figure 2.2: A neural network with three layers.

accumulation as points usually are [CBK09]. But outlier detection can also detect abnormalities relating to the local neighborhood. For example, a point has too few neighbors or it is located nearbyF other outliers. In general, there are methods to detect anomalies in the k^{th} nearest neighborhood of a point whereby k is a sensitive parameter which has to be chosen very carefully, because it can lead to high false alarm rates [EAP$^+$02]. On the one hand the definition of an outlier has a high adaptability, so one can choose various rules or thresholds to determine an anomaly and on the other hand it is even possible to make changes to the spanned room itself. This means it is possible to choose the used metrics. The probably most common used metric is the euclidean distance metric. Translated in the terms of data sets, the euclidean distance uses the same weight for every attribute within the calculation. But there are a lot more distance metrics like the Manhattan or Mahalanobis distance. For example, the Mahalanobis distance calculates the distance from the center of the given cluster, but as a general measure it includes the amount of standard derivation the specific point is far off the center. It is possible to build up a matrix during the training phase determining how strong the influence of every attribute in the distance calculation is [KM09].

2.2.3 Support Vector Machine

A support vector machine (SVM) is a technique used in machine learning primarily for classification task. An SVM can handle both, unlabeled and labeled data, but the focus is on supervised learning which requires labeled datasets. As in most machine learning algorithms, there has to be an initial training phase [LBH15]. An SVM training phase is memory and time intensive but it is often worth the effort, because the training process itself resists outliers up to a certain degree [ERKL16]. In general, a support vector machine

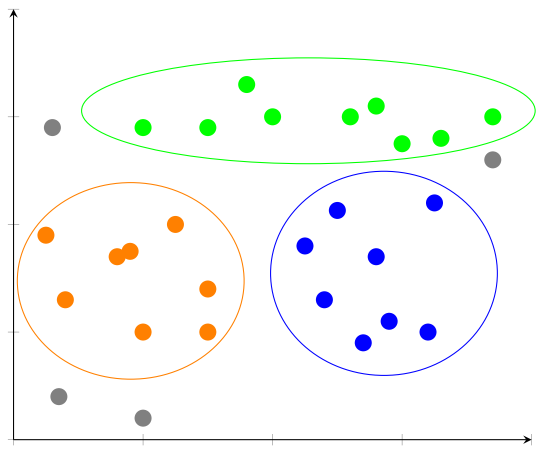


Figure 2.3: A demonstration of a clustering approach on a random set of points.

tries to find a separation with so-called support vectors which can be imagined as straight lines. As long as the given dataset is linear separable the SVM can easily calculate the necessary vectors and the classification is nearly perfect due to the in-build maximum margin algorithm. It tries to maximize the distance between the nearest data points of the given classes [Ray17].

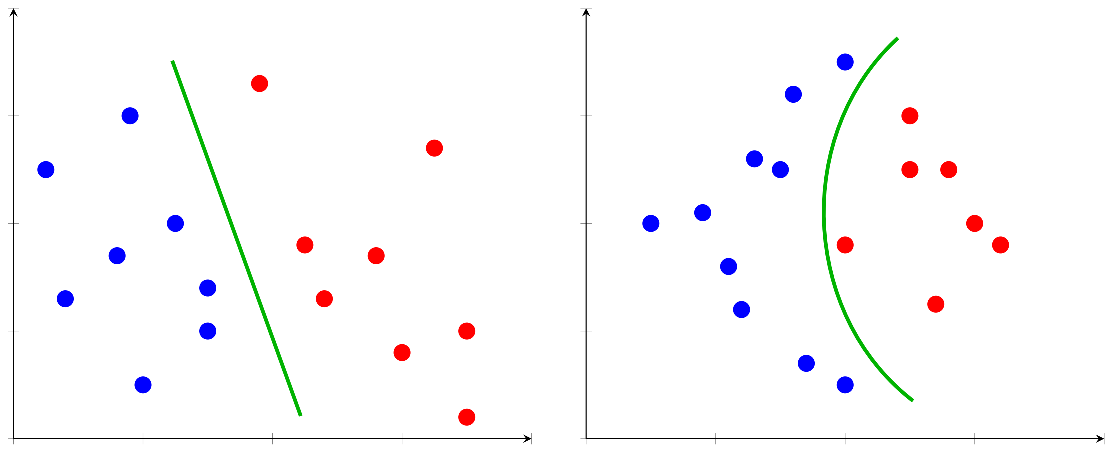


Figure 2.4: The coordinate system on the left shows a set of points which is linear separable and the right one shows data which can not be linear separated.

But there can be the case that the given data is not linear separable which means a linear separation is not possible. The support vector machine can even handle this kind of data by using the kernel trick. The kernel trick is a mathematical algorithm transforming the given low-dimensional data into a space with much more dimensions. Then the SVM tries to find a linear separation in this space which is not a straight line in the original space but rather some kind of circle or wave which is called hyperplane [Ray17]. The kernel trick can be based on different kernel functions which are for example a polynomial kernel, a gaussian RBF kernel or a sigmoid kernel. Every function leads to a mapping which could in turn lead to linear separable classes. A support vector machine has different parameters which have to be chosen very carefully. The parameters determine the degree of sensitivity which can result in a problem called overfitting. It describes the problem that if an SVM adopts itself too strong to the given dataset then the classification would be nearly perfect, but when it comes to datasets of the same kind which slightly differ from the training set, the classification would be everything but good [LGT97]. Nevertheless, a support vector machine is very efficient when it comes to high dimensional spaces and if the number of samples is higher than the amount of dimensions itself. In addition, the SVM uses only a subset of the original training set so that the decision process is memory efficient.

2.3 Intrusion Detection

An intrusion detection system (IDS) is defined as a dynamic monitoring entity which tries to identify anomalies within a system. Those anomalies can be possible security breaches which could be triggered by an individual abusing its privileges or performs specific actions without any permission [MHL94]. This includes technical malfunctions as well as software bugs and errors. In general, an intrusion detection system wants to detect everything which is able to compromise the security goals confidentiality, integrity and availability [PP07].

An IDS is the second layer of defense after the typical firewall. Most of the computer systems already have an IDS which is called virus scanner and can be split up into two main tasks. On the one hand, the typical virus scanner works on real time and checks incoming data as well as system files. It allows a normal workflow without any interruptions but with an active security layer [MHL94]. On the other hand, there is always the possibility to check specific files for malicious code, trojans and hidden data. This function needs the current workflow to be interrupted, but then the file itself can be checked without any restrictions.

As well as there are two tasks of an intrusion detection system, there are two approaches to detect anomalies [TJ03]. The first one is called signature detection which compares the observed behavior with predefined rules and patterns. The second approach is known as anomaly detection. It differs from signature detection insofar as it does not use any rules but rather compares the observed behavior with a self-learned baseline [LS98]. In principle, both are based on the belief that the attackers' behavior diverges significantly from

the behavior of normal users or employees [MHL94].
Due to the classification process which can be a bit ambiguous sometimes, there is always the risk of wrong decisions. In terms of an intrusion detection system, they can be classified as false positives or false negatives [EAP$^+$02]. A false alarm can produce high costs and effort and on the contrary, a false negative means an uncaught failure or even an uncaught attack. This is why signature and anomaly detection need a steady training with continuous updates.

2.3.1 Signature Detection

There exists an incredible variety of threats and attacks for a computer system, but a lot of them are known and well defined. Therefore, the technique called signature detection is very powerful, because the impact on a system of such an attack is easily measurable [EC07]. That is why signature detection is one of the most used techniques e.g. in virus scanners. The goal is to detect the attack during the execution or at least a short time period after its completion. In general, it tries to achieve the status of an expert system which contains as much signatures as possible, though such a system will never exist, because of the fact that there are so many attacks which are still unknown or will be discovered in the future [TJ03]. Additionally, a signature detection system also contains so-called rules. These rules are predefined patterns which forbid a specific action or state in general. A fulfillment of such a rule would lead to the recording of an anomaly or at least a warning. So one could say that signatures and rules want to guard known weak spots which cannot be protected in the first place [LS98]. New developments introduced improvements for the plain rules and signatures by creating so-called case-based patterns which do not use a single rule but rather are a set of rules which try to cover a specific scenario [TJ03]. This can be extended with the inclusion of the system circumstances and both can lead to a more advanced and precise signature detection. More specifically, the rules and patterns consist of statistical data and empirical values which get compared to the observed behavior of the system. For example, it can be a predefined threshold or some sort of rate limiting. If there is a match then the chance of an attack is very high and the system can notify the administrator or raise the alarm.
In most of the cases, there will only be an anomaly or some sort of alarm if an action satisfies a rule and this leads to the advantage of a low false alarm rate in comparison to other detection architectures. In addition to that, once a rule or pattern is created in a correct way, there is a very low chance that an attack which led to the creation of this specific pattern evades the detection system [PP07]. But it also suffers from a few drawbacks. The main difficulty is about the definition of the attack patterns, because an attack has to be discovered and understood in the first place before it can be detected. That is

why a signature detection system is unable to detect unknown or new attacks which differ from the other rules [TJ03]. The only way to handle this drawback are updates in constant intervals needing the system to be expendable. Furthermore, a general problem refers to wide-spread attacks, because a lot of rules have to be defined which would also lead to a high system monitoring and scan effort.

2.3.2 Anomaly Detection

Anomaly detection is a completely different approach to locate malicious actions. In general, anomalies are strong deviations of the observed behavior from the expected or typical behavior and the core of anomaly detection is that the system itself tries to define what normal behavior in this context means [20008]. Before an anomaly detection system can be started there has to be an initial training phase. The training phase is used for preparing the system by feeding it with loads of data so it can create a baseline profile. This training does not have to be a single appearance but rather can be a constant process which would mean some kind of online learning. In the process, the anomaly detection system itself would update its baseline profiles every time it is necessary. Whenever something diverges from this baseline, the system treats it as an anomaly and thereby uncovers a potential attack or malfunction [CBK09].
Anomaly detection in general can be based on one of the following concepts. The first one uses a lot of math and is called statistics [TJ03]. Thereby, the baseline or profiles get calculated during the training phase by using special functions like the standard deviation and the mathematical mean. When the anomaly detection process gets started, it has to transform the current system state in some kind of suitable values and then it can compare them to the profiles in its database. The big advantage lays in the mathematics, because during and after the process every step is some kind of calculation or comparison and can easily be understood. The other underlying concept of anomaly detection is a more modern approach which is called Artificial Intelligence (AI). It is able to find hidden patterns or deviations which are hardly detectable for humans trying to find some of these with acceptable effort themselves, because it uses special methods noticing the data in a complete different way which can be seen in Section 2.2. The only problem which occurs is that AI methods need at least a tiny set of anomalies so they can learn their nature. But there can be the case, as it is in this thesis, that there are no concrete anomalies which means they have to be constructed which often entails a very high effort and complexity [LS98]. In addition to that, it is not even trivial which AI method is the most suitable, because every method has its own advantages and drawbacks.
Nevertheless, an anomaly detection system has some great benefits. For example, signature detection is very fast but it can not protect itself or the system against insider attacks

which does not apply for anomaly detection [PP07]. An insider attack is performed by an employee or former employee who knows the system and thus may be aware of the stored signatures. Anomaly detection does not require patterns and that is why it can detect insider attacks as well as attacks which are still unknown (zero day attacks), because these attacks are mostly not part of a typical workflow [KV03]. Furthermore, the exact strategy and the anomaly criteria are unpredictable, because when the training phase is completed then nobody really knows where exactly the border for an anomaly is. This fact makes it a lot more difficult for intruders or even insiders, since they do not know which actions have the potential to unmask them [PP07].
But there are also a few drawbacks. First, anomaly detection systems have the problem of a high false alarm rate, because the system raises the alarm every time when there is an activity that diverges from the normal profiles and this can happen quite often in the normal workflow. Therefore, an anomaly detection system needs precise thresholds which are tailored to the system [KM09]. The second drawback consists of the problem that it is not that easy to create the mentioned baseline profiles of the normal workflow, because the training phase requires a lot of data without any malicious actions and these profiles have to be adjusted in constant intervals, for example if work steps have changed, they should obviously not be tagged as anomalies. In addition to that, the profiles representing normal work behavior are vulnerable to little changes as the system tries to integrate them into the profiles, which means they can be trained so that they accept some kind of malicious action after some time [CBK09]. The whole anomaly detection process is built on the assumption that the created user profiles or the baseline make sense or in other words allows an efficient analysis process. It has to be possible that something like an average behavior can be derived. But on the contrary, the creation process can be arbitrary difficult if the goal is to achieve the best possible results. To finalize the detection process, there has to be some kind of notification at the very end, because one wants to know which actions caused the anomaly.

3 Process

The developed prototype for finding anomalies within the given protocols is based on two strategies. The first one is called signature detection and is basically a rule-based detection system which is a rework of the former detection system and it consists of all rules which were used before. In addition, it was developed in regard of the given protocols to create a better adaption and easier rule design which was achieved by a generic rule structure. The second strategy the prototype is based on is certainly known as anomaly detection, but in this context it means the search for unknown anomalies by using statistical algorithms and methods from artificial intelligence.
Signature detection as well as anomaly detection are developed and implemented with the assistance of a parallel and steady evaluation phase. It helps to find the right structure of the learned profiles in the statistical part, but also during the search of the right kernel mode for the support vector machine. The general hybrid structure of the prototype can be seen in the following figure.

- Prototype
  - Signature Detection
    - Rule-based Analysis
  - Anomaly Detection
    - Statistical Analysis
    - Support Vector Machine

Figure 3.1: The functional structure of the implemented prototype.

3.1 Rule-based Analysis

The implemented rule-based detection system is primarily developed to replace the current system. The rules itself which were derived from practice should be maintained, but the focus is on a better adaption to the given protocols. The following subsections exactly show the implemented structure and functionality of all rule types with practical orientated examples. The generic principle behind the structure of the rules is very basic, but it allows a fast and clear rule creation: Every keyword or parameter which is not specified or in other words was left out will be classified as irrelevant.

3.1.1 Rule Types

Plain Rule

The plain rule is the most basic type of the implemented rules, because it has a specified set of conditions which can be true or false. Those conditions can also be seen as a boolean function which is a conjunction of the given conditions. It gets applied to every line in the given protocols and if the line or in other words the action of a specific user satisfies the boolean function then the activity will be treated as an anomaly.
Basically, a plain rule has a generic structure which can be split into keywords and parameters. Every feature in the protocols stands for one keyword and every possible value of a feature can be assigned to the keyword to create a boolean condition. The username is the only keyword which differs in its functionality, because it accepts every kind of regular expression due to the set of assignable values. The parameters are predefined patterns which affect the behavior of the specific rule:

Events The keyword "Events" allows the determination of an upper or lower border for the amount of rule satisfactions until it arouses an anomaly. In detail, it can be specified as follows: "Events $> X$" or "Events $< X$", whereby the operator can be "$>$" or "$<$" and X indicates an arbitrary amount of satisfactions.

Interval The "interval" keyword can be used to define a time interval in which the rule has to be satisfied, because otherwise it just gets ignored. The exact line has to be in the following format: "20:30:00 - 07:30:00" which means this rule can only be satisfied between half past eight p.m. and half past seven a.m..

Reset Time A plain rule also allows the keyword "RESET $= X, Y$", whereby Y can be "MINUTE", "STUNDE", "TAG" or "WOCHE" to specify the time unit and X is an arbitrary time specification. The reset time resets the rule after the specified time period which means that a user has to satisfy the rule within the time period to arouse an anomaly.

Details A plain rule can record all lines which were included in an anomaly by using the keyword "MIT_DETAILS".

Name There exists one parameter which does not affect the behavior of the rule. The string "NAME=X" can be used to assign the name X to the specific rule which can only be seen in the output.

The sole goal of a plain rule can be expressed as a simple threshold detection. In detail, one should be able to specify an exact threshold which applies for all users or a subset of them and checks if one of them satisfies the condition. Including all keywords and parameters, a plain rule has two dictionaries. The first one records every single rule satisfaction of every user and the second one counts the "real" anomalies, because everything that lands in here can be seen in the results of the specific rule. The results or the output by itself consist of a list of all users and their individual amounts of rule satisfactions during the analysis process.

Listing 3.1: The following example shows a plain rule with the name "Rule1". The condition of this rule is true whenever the features of a protocol line correspond to those specified in this rule. The feature "PROZESS" has to be a login and the expression "^[JS].*" only allows usernames starting with "J" or "S". The reset keyword leads to a reset of rule satisfactions every day and the specified interval ensures that this rule can only be satisfied out of the normal working time.

```
          NAME=Rule1;PROZESS=LOGIN;FKL-PROZESS=8,16;USER=^[JS].*
          ;RESET=1,TAG;INTERVAL=20:30:00-07:00:00;
```

Relation Rule

One of the advanced rule types which is based on two plain rules is called relation rule. It can output various relations by calculating the quotient of the sums of the rules' results, but in contrast to the other rules the final results will not be user-separated but rather a list of relations per time interval. The relations itself gets calculated periodically after a time interval Δt or at the end of the analysis process if Δt is not specified. The following equation summarizes the mentioned process:

$$\{r_1, r_2, ..., r_n\} = \left\{ \frac{|\text{Rule1}|}{|\text{Rule2}|}_{[0;t]}, \frac{|\text{Rule1}|}{|\text{Rule2}|}_{[t;2t]}, ..., \frac{|\text{Rule1}|}{|\text{Rule2}|}_{[(n-1)t;nt]} \right\}$$

This rule has a generic structure "ERSTENS(X) ZWEITENS(Y)", because it consists of two plain rules, X and Y, which can be created independently from each other. A relation rule supports a subset of keywords which are: "MIT_DETAILS", "NAME" and "RESET TIME". The first two keywords have exactly the same functionality as in the plain rule, but the reset time represents the time interval Δt.

Listing 3.2: The relation rule "Relation1" calculates the fail login ratio every four hours. The amount of fail logins gets divided by the amount of total logins due to the first plain rule which captures all fail logins and the second plain rule which captures every login. The special feature of a relation rule is that details about the upper or lower event border in both plain rules get cut out and used as a general border which will be applied on the total sum of the events per rule.

```
NAME=Relation1;ERSTENS(NAME=FailLogin;PROZESS=LOGIN;FKL-PROZESS=8,16;)
ZWEITENS(NAME=Login;PROZESS=LOGIN;)RESET=4,STUNDE;
```

Time Rule

The time rule is the second type of rule which is based on plain rules. Its structure is very similar to a basic if-statement "IF ... THEN ... ELSE ...", but the condition is twofold. The generic structure consists of three parts: "WENN(X)", "KEIN(Y)" or "DANN(Y)" and "IN(Z)". The keyword "WENN" takes a plain rule X symbolizing a condition which starts the observation process of the specific user who fulfilled this plain rule. "KEIN" is the keyword which gets a plain rule Y which has to be fulfilled in a specific interval Δt, otherwise the time rule would arouse an anomaly. A time rule also allows the opposite of "KEIN" which is specified as "WENN", in this case an anomaly is the fulfillment of the plain rule Y within the time interval. The interval itself gets specified by the keyword "IN", whereby Z is not a plain rule but it has the same functionality as the keyword "RESET TIME". In general, whenever a user satisfies the "WENN"-rule, he has to satisfy the "KEIN"-rule within the "IN"-interval Δt.

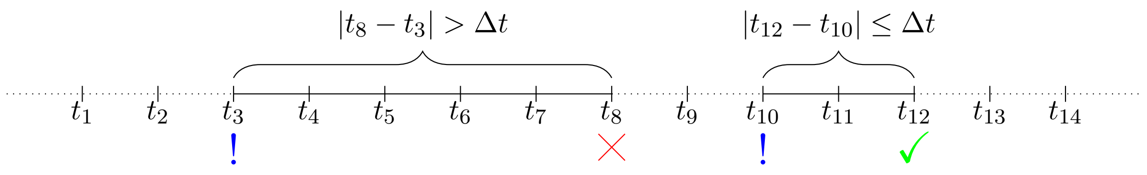

$|t_8 - t_3| > \Delta t$
$|t_{12} - t_{10}| \leq \Delta t$
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14}
!
×
!
✓

Figure 3.2: This figure shows the generalized concept of a time rule with a "KEIN"-condition instead of a "DANN"-condition. The "!" stands for the moment when a user fulfills the first rule, the red cross symbolizes the case where a user was not able to fulfill the second rule within the interval Δt and the check mark is the exact opposite of the red cross which means a user did not arouse an anomaly.

The output of a time rule is very similar to the output of a plain rule, because it consists of a user-separated list which shows all violations of every user which can be extended with the involved protocol lines by using the keyword "MIT_DETAILS". Additionally, a time rule can have a name and the keyword “EVENTS” activates a lower/upper border to eliminate too many or to little rule satisfactions.

Listing 3.3: This is a typical example for a time rule derived from practice. Whenever a user's login attempt is not successful, whereby the username has to start with "Q", "S" or "J", he has to login successfully in the next five minutes, otherwise this behavior will be treated as an anomaly, because it is not part of a typical workflow.

```
        NAME=Time1;WENN(PROZESS=LOGIN;FKL-PROZESS=8,16;USER=^[QSJ].*)
        KEIN(PROZESS=LOGIN;FKL-PROZESS=0,2,4;USER=^[QSJ].*)IN(5;MINUTE)
```

3.1.2 Application in Practice

This rule-based detection system is based on the current system 1.3.3, but due to its strong adaptation to the given protocols it is the more advanced and superior system. The generic rules are the main advantage, because they can illustrate almost every possible rule and even most edge cases. It is also possible to use a plain rule for a monitoring task so the system records every occurrence of a special activity. An advantage also lays in the extensibility of the system itself. The keyword “MIT_DETAILS” just outputs the involved protocol lines for now, but in combination with the object-orientated development it is possible to rework the output in any desired direction.
During the development and testing process, a very striking fact appeared: The current applied rules do not control the appearance of password changes. That is why a new rule was created to detect successful password changes in a short time interval by using a time rule.

Listing 3.4: A time rule which tries to detect multiple successful password changes within two hours.

```
    NAME=PWChangeDetection;WENN(PROZESS=PASSWORT_AENDERN;FKL-PROZESS=0,2,4;)
    DANN(PROZESS=PASSWORT_AENDERN;FKL-PROZESS=0,2,4;)IN(2;STUNDE)
```

3.2 Statistical Analysis

Although the rule-based detection part is able to detect every kind of threshold exceeding and interval violation, there is still nothing which can deal with unknown anomalies. To achieve this, the next step in the search for appropriate anomaly detection methods is the implementation and evaluation of a component which is specialized in a statistical analysis. As a special case or subfield of anomaly detection, it brings along most of the

benefits of AI without the typical resulting complexity.
The process of a statistical analysis often gets confused with AI in general, but is has to be distinguished that AI algorithms can adapt themselves to the given problem by evaluating different mathematical functions or by changing the structure of the data and the statistical analysis does not have any of these abilities. It still provides a new approach for the detection of new anomalies and in this respect, the statistical analysis represents a good addition to the rule-based component.

3.2.1 Data-specific Characteristics

A statistical analysis wants to calculate tons of values by using different mathematical functions. The occurring problem is that these functions only take usual numbers as input, but the given protocols almost contain discrete features like “PROZESS” which is a string consisting of “LOGIN”, “LOGOUT” or “PASSWORT_AENDERN”. As a result, general mathematical methods e.g. the average can not be applied but appearances and frequencies are derivable. In other words, it is necessary to perform some kind of feature extraction which means that ratios and amounts have to be derived by some kind of counting process. So the analysis itself does not get performed on the original protocols but rather on data which gets created out of the protocols. That means that measuring, counting, value extraction and the calculation of ratios, amounts and lists form the first step towards a statistical analysis.
It has to be considered that every protocol line stands for one activity or action which was produced by a specific user. Wherever humans are involved, fluctuations and deviations in the everyday workflow can be observed due to mistakes, differing behavioral patterns and random incidents. It is a very difficult problem which can not be solved easily, moreover it has to be accepted. One phenomenon which arouses because of employees can only be observed on Mondays. The applied statistics clearly show an increase of failed login attempts every Monday which has to be noted, as otherwise the detection process would produces a lot of false alarms. In addition, not only human behavior can lead to fluctuations, but also general rules, regulations and habits can cause troubles when it comes to a statistical evaluation. The typical decrease of employee activity on Friday has to be considered as well as other days implying less events. It is even possible that a so-called test user is activated for a specific time interval who intentionally tries to arouse a lot of successful or failed events which can lead to an anomaly. The largest potion which can cause an anomaly are failed events, because they are the main indicator for attacks. The problem is that they are a small amount in comparison to all events in general 3.3, thus they can easily get lost in the pure mass of events and are very vulnerable to fluctuations.

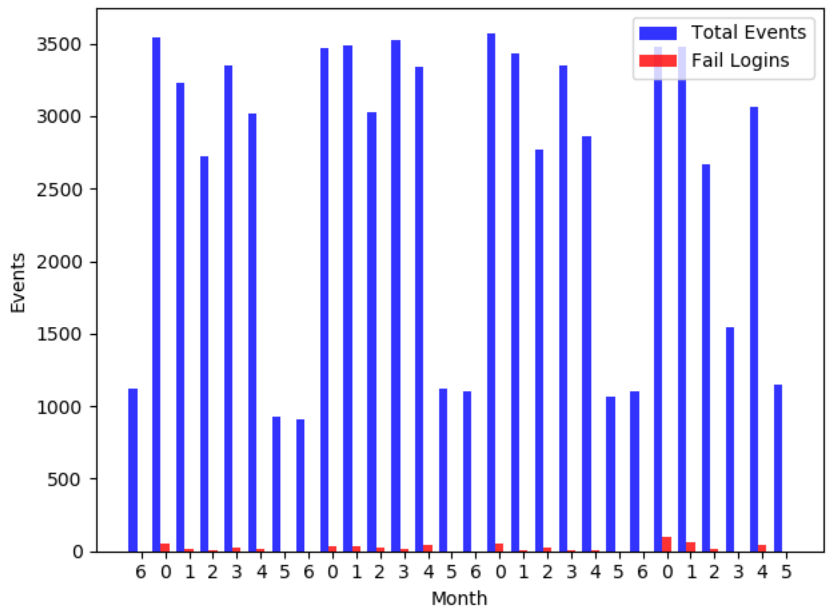


Figure 3.3: A plot which shows the total amounts of events per day within one month in contrast to the amounts of fail logins.

3.2.2 Training Phase

Before a statistical analysis can detect any anomalies there has to be a training phase first. It analyzes the given data with statistical methods to create some kind of profile for every user which tries to describe the standard behavior or in other words the baseline. The created profile is saved in an extra file so that this file can serve the detection phase as input. The evaluation is a simultaneous process with the implementation itself, thus it is possible to adopt results and new discoveries to enhance the training phase immediately. A strict separation by weekday is one of these improvements. The evaluation clearly shows that the behavior of employees differ significantly from day to day which calls for a weekday separation, otherwise the learned profile will be too general and will lead to a lot of false positives. Due to various types of users, only human users are integrated into the analysis process, since the other user types (technical/dummy users etc.) either have a very monotonous workflow or suffer from incredibly high fluctuations.
The listing 3.5 represents a cutout from a training file in JSON-format. It shows the user profile of a single user “S_TEST1”, whereby the weekday separation can clearly be seen.

Average

Considering the set of all events in a protocol $X = \{x_1, x_2, ..., x_n\}$ with $n = |X|$ as the total amount of events. Then the average $\overline{X}$ can be calculated as follows:

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

The paragraph "AVG" contains all values describing an average. "WORKING_TIMES" consists of the average begin and end of the working time of a user as exact time unit. The average usages of timestamps symbolizing different programs which were called by a user are stored in "TIMESTAMPS" and the topic "EVENTS" shows the average amount of events in total on a specific day.

Standard Derivation

The standard derivation σ can be calculated by using the amount of events n, the set of events X in general and the average $\overline{X}$:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{X})^2}$$

The standard derivation "STDEV" helps to describe to what extent the specific user deviates from its average. It presents a good measurement whether a user behaves in his limits or does something unusual. Therefore, this paragraph only contains values corresponding to those in "AVG".

Ratios

A ratio can be calculated with the following equation by assuming that X is the set of all events and x is a subset of X:

$$r = \frac{|x \subseteq X|}{|X|}$$

To complete a user profile, special thresholds can be derived which can be the ratio of fail login attempts for instance. In addition, they can easily be integrated into the detection phase and they are a good instrument to represent user behaviors.

Listing 3.5: An exemplary extraction of a learned user profile which was derived from the provided protocols.

```
"S_TEST1": {
  "MONDAY": {
   "AVG": {
    "WORK_TIMES": {
     "BEGIN": "08:54:59",
     "END": "18:27:29"
    },
    "TIMESTAMPS": {
     "2008-01-09-11.44.32.067892": 15.285714285714286,
     "2003-09-22-14.38.43.161955": 6.5,
     "2014-12-17-11.07.41.225512": 4.0,
     "2008-01-09-11.44.32.067891": 3.8461538461538463,
     "2008-01-09-11.44.32.067890": 3.5714285714285716,
     "2005-08-02-11.55.04.042100": 2.0
    },
    "EVENTS": 30.2143
   },
   "STDEV": {
    "WORK_TIMES": {
     "BEGIN": "01:25:57",
     "END": "00:11:40"
    },
    "EVENTS": 8.11625691890211
   },
   "RATIOS": {
    "FAIL_LOGIN": "7.05882%"
   }
  },
  "TUESDAY": {
   ...
  },
  "WEDNESDAY": {
   ...
  },
  "THURSDAY": {
   ...
  },
  "FRIDAY": {
   ...
  }
 }
```

3.2.3 Detection Phase

The detection phase is the last part of a statistical analysis which handles the comparison and the decision making. It takes the previously created training data as input and iterates through all users to process every user statistic separately. The comparison process first computes all necessary statistics out of the uploaded protocols $AVG = \{\text{Work}_{\text{Begin}}, \text{Work}_{\text{End}}, \text{Timestamps}, \text{Total Events}\}$, $RATIO = \{\text{Fail\_Login}\}$ and then compares them with the training data by using the standard derivation $STDEV(x)$ and thresholds $T(x)$ which were derived from practice. The decision making or in other words the real detection part observes the comparison and if a statistical value exceeds its limit an anomaly gets aroused, otherwise the value can be classified as normal behavior. So one could say that during the statistical analysis an anomaly is treated as a deviation from the standard.

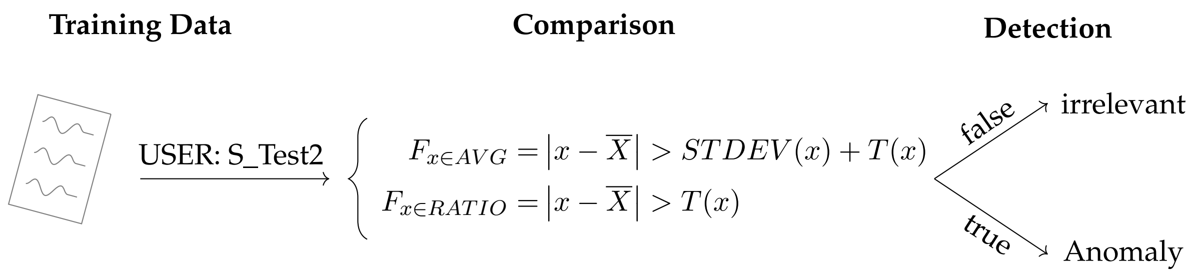

Training Data
Comparison
Detection
USER: S_Test2
$F_{x \in AVG} = |x - \overline{X}| > STDEV(x) + T(x)$
$F_{x \in RATIO} = |x - \overline{X}| > T(x)$
false
irrelevant
true
Anomaly

Figure 3.4: Illustrates the anomaly detection process of the statistical analysis.

The output will be saved in a separate file which can be seen in 3.4. Every user who showed malicious behavior obtains its own section within the output. In addition, the anomalies are strictly separated by date which guarantees full transparency and traceability so that the underlying reason can be examined. If a user exceeds its limit of the fail login ratio, the output will consist of a line in the following format "FAIL_LOGIN: A: INCREASE OF X% FROM Y% TO Z%". A describes the total amount of fail logins, Y symbolizes the baseline value, Z the current ratio in the uploaded protocols and X stands for the absolute difference between Y and Z. An anomaly in the timestamps appears as "T: FROM Y TO Z BY X", whereby T stands for the current timestamp and X, Y and Z exactly have the same functionality as in "FAIL_LOGIN". Whenever a user exceeds its limit for the amount of total events, the anomaly gets displayed just like a fail login anomaly without an information about the total amount. A deviation in terms of the start/end of the working time is represented by the following string: "START/END $T1$: AT $T2 \rightarrow T3$". $T1$ is the time stored in the user profile, $T2$ is the current recorded time and $T3$ shows the absolute difference between $T2$ and $T1$.

Listing 3.6: An exemplary extraction from the output file generated at the end of the statistical analysis.

```
"S_Test2": {
 "2019-03-08": {
  "FAIL_LOGIN": "10: INCREASE OF 26.4477% FROM 4.80226% TO 31.25000%",
  "TIMESTAMPS": [
   "2008-01-09-11.44.32.067892: FROM 15.0244 TO 23 BY 7.9756",
   "2008-01-09-11.44.32.067890: FROM 3.0 TO 14 BY 11.0",
   "2003-09-22-14.38.43.161955: FROM 6.85 TO 13 BY 6.15"
  ]
 },
 "2019-03-27": {
  TOTAL EVENTS": "INCREASE OF 25.439 FROM 29.561 TO 55",
  "END 14:52:18": [
   "AT 17:49:25 -> 02:57:07"
  ]
 }
}
```

3.3 Use of AI Algorithms

At this stage, the analysis process checks the protocols by using a predefined set of rules and it can create a baseline of the work behavior of users to detect deviations. The next task is all about finding appropriate methods in the subset of anomaly detection algorithms from artificial intelligence. Therefore, various AI methods get applied on the given protocols or rather the attempt is made, because AI has some promising benefits.
The first algorithm which is tested is called clustering, because it is virtually predestined for classification. There exist various clustering algorithms, but in general they are all based on the same principle. During a clustering process, a few initial cluster centers get defined which is followed by a re-positioning of the centers and the expansions of the clusters itself. The problems which occurs and which is why clustering cannot be applied, can be described as the general state of the given data. It is exposed to high fluctuations leading to a lot of outliers. Moreover, an appropriate subset of features has to be chosen in which the data can be classified and where anomalies are characterized by a great distance to a cluster center. The generation of such a subset is not trivial and requires knowledge about anomalies in the given context. Figure 3.5 shows a 2D plot which clearly shows that a great distance from the main cluster does not indicate an anomaly.
Neural networks as one of the most famous AI methods can be used similarly to an SVM due to its classification ability. But the reason why a support vector machine was used

instead lays in the difficulty to apply such a network on the given data. There would have to be an enormous effort in the data preprocessing and even though neural networks are known as very powerful the usage is not trivial at all. In addition, one can not be totally sure whether the obtained results comply with the expected results. Of course, the time component of this thesis plays a big role as well which leads to the fact that more simple methods like the support vector machine were finally used. The goal in general is all about finding the perfect and most appropriate separation between a set of normal data points and anomalies. With an evaluation phase of different SVM kernels and their parameters such a separation could be achieved. The following sections present the integration of a support vector machine to create a new anomaly detection process.

3.3.1 Data Preprocessing

Even though a support vector machine seems to be the best method for the classification job in regards of the given protocols, the process of finding and preparing the data is still not trivial at all. An SVM takes labeled data as input and outputs a separation, which means one has to know whether data points present an anomaly or not. In mathematical words, one has to find an adequate subset of features to generate a feature subspace in which the data is arranged in a way making a classification possible.
In the current rule-bases system there is one main factor which indicates or helps to detect anomalies: the occurrence of multiple failed login attempts. In fact, fail logins can be measured in two ways. On the one hand, fail logins can be described by their total amount. This plain value suffers from two drawbacks. The difficulty lays in the fact that the total amount only allows an upper border high enough to withstand normal fluctuations, but still it has to reflect the normal behavior. In addition, one has to imagine an amount of fail logins of 130 which seems to be very unusual in the first place, but the additional information that the specific user performed 1500 login attemps in total relativizes the fail attempts to normal happening.

On the other hand, the ratio which is a quotient of the amount of fail logins and the amount of all login attempts in total is not exposed to very high fluctuations. However, if a user only performs 4 login attemps and two of them failed then the ratio will come to 50%. The sole ratio is conspicuous, but in regards of the few events the anomaly vanishes. To sum it up, both of these measurements for fail logins, taken in isolation, have a reduced information value, but somehow they seem to correlate, because a high ratio in combination with a high amount doubtlessly is an anomaly. The following section shows the training process and the detection process of a support vector machine by using the just considered representation forms of fail logins.

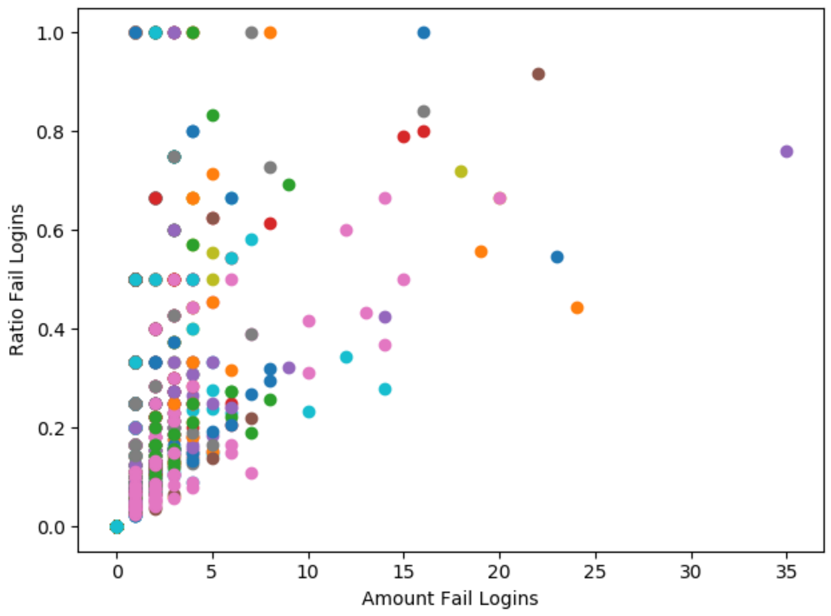


Figure 3.5: A plot of the total amount of fail logins in conjunction with the ratio of fail logins whereby every point stands for one user on a specific day.

3.3.2 SVM Anomaly Detection

Before a training can be started, the SVM has to be fed with two objects. The first one is a table consisting of all data points with their position in the feature room and the second object is a list containing the corresponding classification for each data point. During the training phase, a support vector machine uses very complex Mathematics to find the optimal separation due to the implemented maximum margin algorithm. However, this is accompanied by a long duration of the calculation process which of course increases significantly when more data is included. In addition, the state of the data also plays a role when it comes to an SVM which is the reason why a so-called scaler is used to prepare the data so that the training phase is as efficient as possible. The scaler manipulates every single value so that the average is 0 and the standard derivation is 1. This process allows it to even handle months consisting of very big protocols. The training set which is used to train the SVM in the feature set of the amount and ratio of failed login attempts can be seen in Figure 3.6 which already underwent a handmade anomaly classification.

The first attempt to achieve a separation is a linear separation due to the fast training process and less parameters. At first sight, the data seems to be linear separable 3.7, whereby every data point to the left is classified as normal and everyone to the right as anomaly. But after a short time, the section marked by a red question mark which symbolizes a low

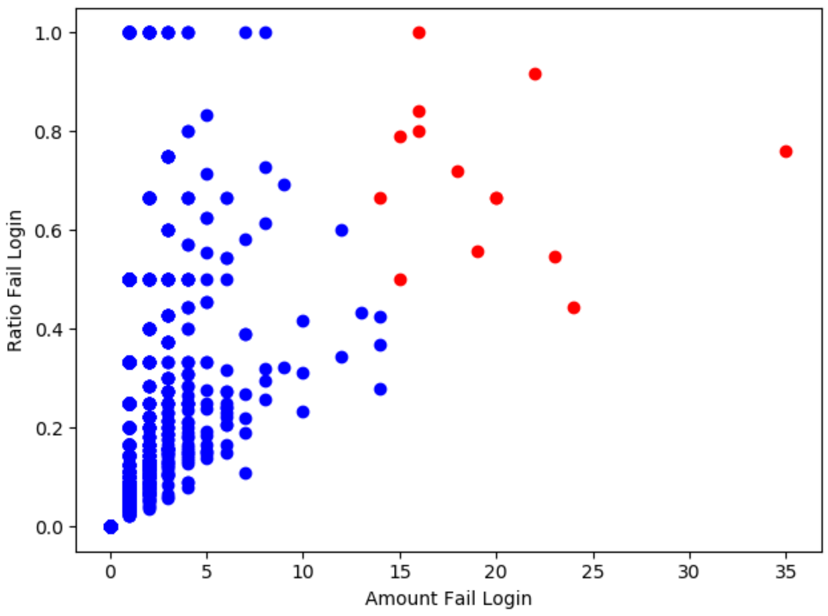


Figure 3.6: A plot of the total amount of fail logins in conjunction with the ratio of fail logins. The points already underwent a handmade anomaly separation.

ratio in combination with a high amount attracted attention, because every point in here is classified as an anomaly. But as already mentioned, a low ratio with a high amount does not represent an anomaly at all and will lead to a lot of false positives. This problem clearly demonstrates the fact that a linear separation is not suitable in this context.

In order to deal with nonlinear separable data, the next step is to find an appropriate kernel which lifts the data into a higher dimensional space and searches for a hyperplane leading to an optimal separation. A short evaluation phase has to be performed to find the optimal parameters for a polynomial kernel. In this process, multiple polynomials get tested and a few can be seen in Figure 3.8. Almost every polynomial is not bad at all, but most of them suffer from overfitting which means that they overadapt themselves to the given data set and are not useful when it comes to other data sets. The evaluation reveals that the sixth and the second order polynomials are the best ones when it comes to this classification task.

A support vector machine trained with these polynomial kernels can be applied on similar data sets to achieve a separation as shown in Figure 3.9. However, the main problem remains, the data is not labeled and so there has to be a very high effort in making handmade classifications to generate appropriate training data for an SVM. Once this is achieved, the detection process is very fast and reliable. But this reliability is highly premised on the condition that a separate set of handmade anomalies is used for every client due to the

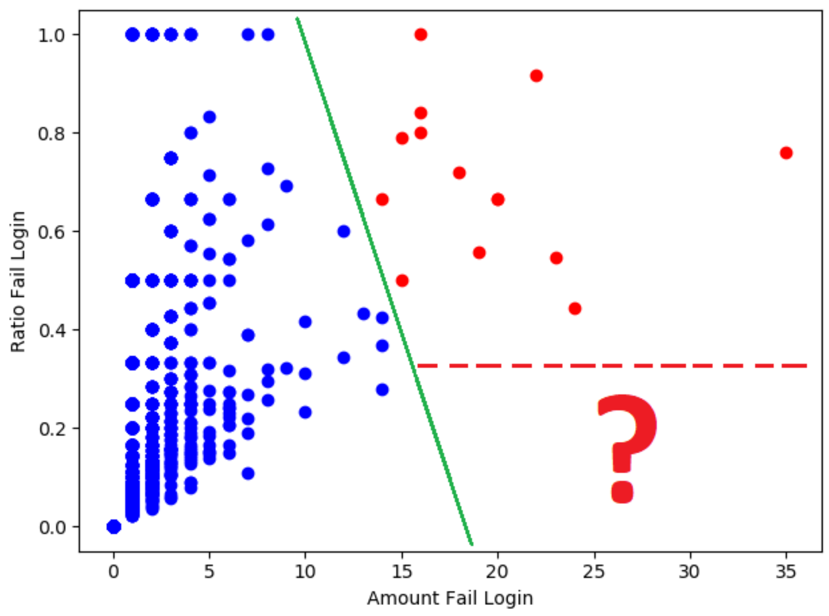


Figure 3.7: A plot of the total amount of fail logins in conjunction with the ratio of fail logins. An attempt was made to apply a linear separation on the data points which did not work as expected.

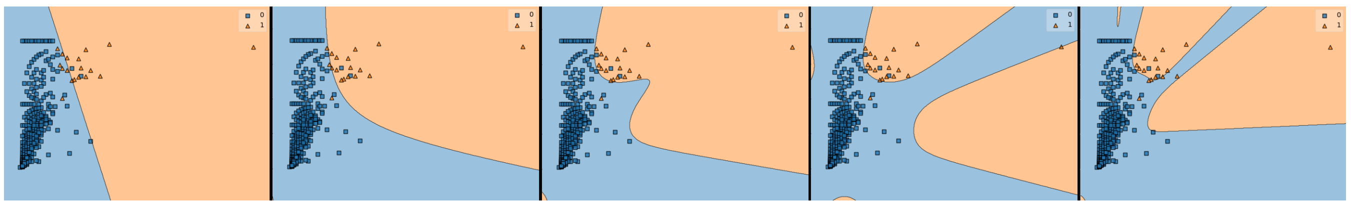


Figure 3.8: To find the optimal SVM kernel, multiple tests with different kernels were made.

strong behavioral differences. Nevertheless, the whole classification process depends on the created set of anomalies used for the training phase. The ability of these data points to describe the differences between anomalies and normal behavior finally determines the accuracy of the classification.

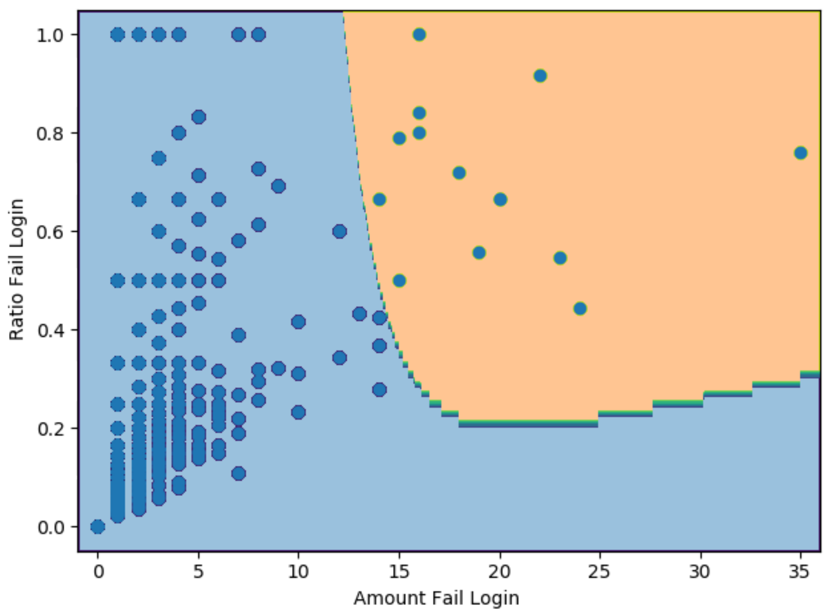


Figure 3.9: A plot of the total amount of fail logins in conjunction with the ratio of fail logins whereby the decision curve of a trained SVM can be seen.

4 Implementation

4.1 Anonymization Tool

4.1.1 GUI

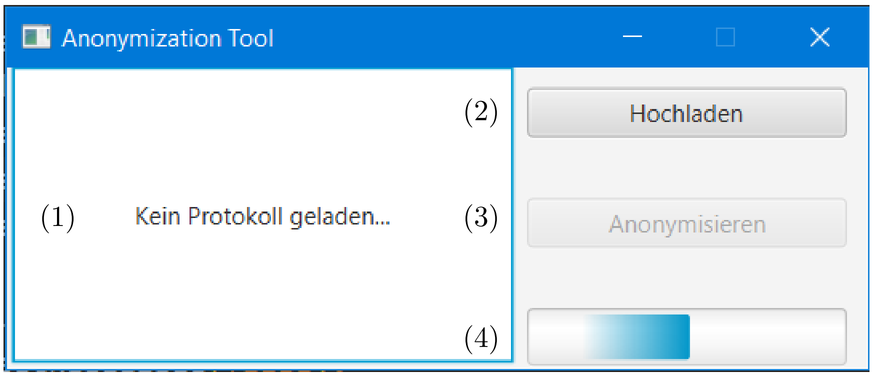


Figure 4.1: The graphical interface of the anonymization tool.

(1) A special list which contains all uploaded protocols. The user can select one of the selected protocols and by pressing the "Del" key the protocol will be removed not only from the list but also from the whole application.

(2) The button "Hochladen" uploads protocols to the application. A simple click opens a file chooser dialog which can be used to navigate through folder structures to select specific protocols.

(3) If one wants to start the main process of this application, the anonymization itself, the button "Anonymisieren" has to be pressed. All uploaded protocols get checked for the correct format 1.3.2 and then they will be anonymized.

(4) This loading bar shows the progress of the anonymization process.

4.1.2 Development/Functionality

The provided protocols fall under the Data Protection Act, because they contain the activity of many employees on several days. This yields to the problem that the provided protocols can only be used after an anonymization process. Therefore, the anonymization

tool has to be created to realize such an anonymization process.
The tool itself is as simple as possible to guarantee a transparent upload and anonymization. The protocols can easily be uploaded and the anonymization starts with just one click. An anonymization in general can be done in multiple ways, but in terms of a continued use of the anonymized protocols the selected method is called pseudo-anonymization. In detail, the process itself proceeds in an external thread which first loads the file "mapping.txt" consisting of a lot of street names. The reason why street names are used is that names in general can easily be distinguished and noticed in contrast to strings of random numbers and letters. At the end of the process, the user gets asked to save the currently created mapping in order to reuse it which allows an equivalent and repeated anonymization. It is therefore possible to select a custom mapping at the beginning. After the process has finished, the tool shows the percentage of warnings and errors by comparing the number of lines in the warning or error file with all lines in the uploaded protocols. In addition, three files with the following suffixes are created after the anonymization: "_warnings" contains all lines which are not corrupted but have a wrong format and still can be analyzed, "_errors" contains all lines which are corrupted and cannot be analyzed and " _anonymized" is the pre-processed file without any errors which can be used for further analyses. This tool is only used at the beginning of this thesis to make enough protocols available in order to use them for the development of the anomaly detection tool.

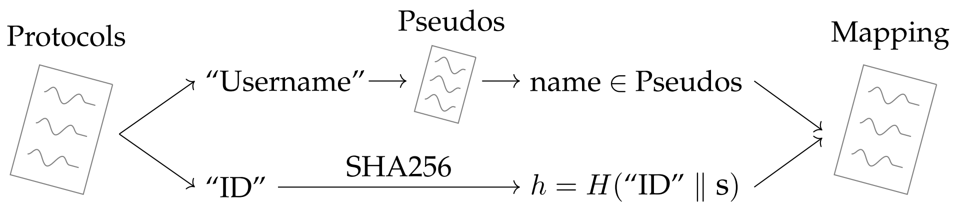
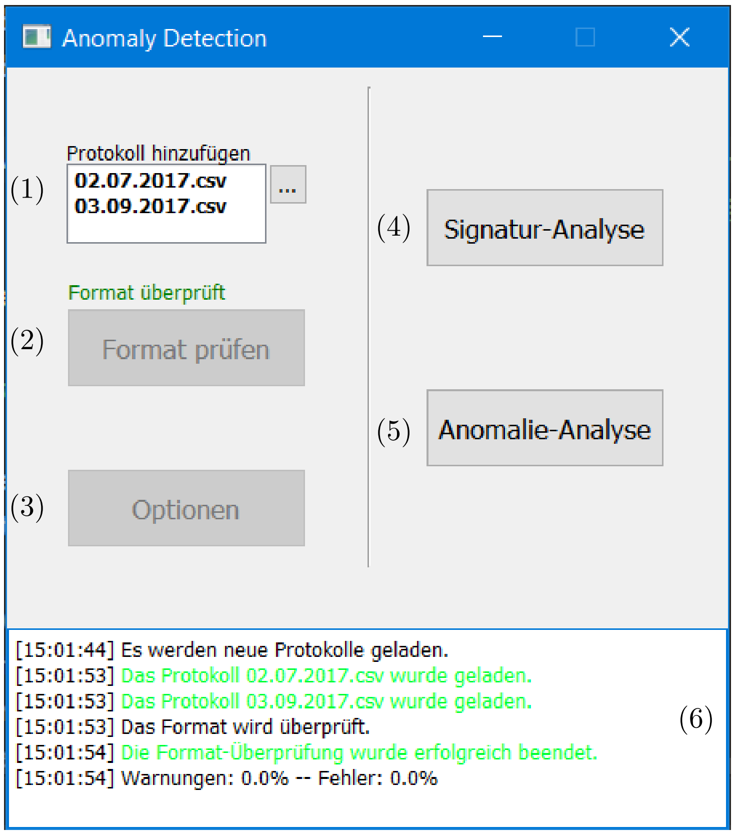


Figure 4.2: Illustrates the anonymization process. It includes the attribute "Username" which is mapped to an aforementioned name whereby the user type is added at the beginning with an underscore and the attribute "ID" is mapped to a hash of itself concatenated with a unique and random string (salt).

4.2 Anomaly Detection Tool

4.2.1 GUI

(1) Before any analysis can be started, the protocols have to be uploaded to the application. This can be done by pressing the button "..." which opens a file chooser dialog which makes the protocol selection much easier. Once a few protocols are selected,



```
[15:01:44] Es werden neue Protokolle geladen.
[15:01:53] Das Protokoll 02.07.2017.csv wurde geladen.
[15:01:53] Das Protokoll 03.09.2017.csv wurde geladen.
[15:01:53] Das Format wird überprüft.
[15:01:54] Die Format-Überprüfung wurde erfolgreich beendet.
[15:01:54] Warnungen: 0.0% -- Fehler: 0.0%
```

Figure 4.3: The graphical interface of the anomaly detection tool.

the list container displays every single protocol name. Another click on the upload button will lead to a message box which asks the user if he wants to continue. A confirmation entails a deletion of the uploaded protocols from the application so that new protocols can be selected.

(2) The button "Format prüfen" checks the format of the uploaded protocols, because every further process is based on the assumption that the protocols are in a correct format.

(3) This button is a placeholder for future configuration options.

(4) The signature detection process starts after the button "Signatur-Analyse" is pressed. The user gets asked to select the necessary rule file which is followed by the start of the rule-based analysis.

(5) A click on the button "Anomalie-Analyse" results in a message box which asks the user whether he wants to perform a training or detection process. Regardless of the

decision, another message box appears which asks if statistics or a support vector machine should be used.

(6) The log console displays three kinds of information: information (black), successes (green) and fails (red). This is complemented by the exact timestamp of the specific event.

4.2.2 Development/Functionality

The anomaly detection tool in general is a tool which combines all attempts to find anomalies in the provided protocols. In other words, it is a hybrid system which supports two main functions. On the one hand, there is the rule-based detection representing the signature detection part and on the other hand, there is the statistical analysis in combination with a support vector machine which can directly be assigned to the anomaly detection part.
In contrast to the anonymization tool which is implemented in Java, the anomaly detection tool is realized by using Python. Almost every created process needs some basic table editing functions in a more or less complex way and because Python provides excellent libraries for table editing, its library "Pandas" is used. The support vector machine is made available by the library "skit-learn" due to the mathematical complexity of an SVM and the massive effort of a new implementation. The implementation of the prototype is based on the "model view controller"-model which simultaneously achieves a very good transparency and separation in an object-orientated manner. In general, the model contains the whole program logic which is strictly separated from the view whose only task is handling the graphical interface. In contrast to that, the controller handles the user input and manipulates both components, the model and the view.
Every executed process is running in an external thread, but no multi-threading is intended which means that after the user starts a process, he has to wait until the process has finished. Afterwards, the results show up and can be processed. The correct handling is based on a little order of proceeding. Directly after the application start, the user can only upload protocols. Then the next possible process which can be started is the format check. A successful outcome leads to an activation of all the other functions which are in detail the signature- and anomaly detection.

4.2.3 Runtime Analysis

A runtime analysis gives a good overview about the average time and complexity of the different processes, but with the addition that only the detection processes are analyzed due to the fact that the anomaly detection tool will be used in practice and the training phases will optimally be done just once or at longer intervals. The standard usage of the prototype includes a whole month consisting of four protocols which have an average size of 5MB-90MB in total. So the protocols can variate in their size due to the amount of employees and the size of the specific client in general. The following table shows a short runtime analysis of the three detection methods:

Table 4.1: An overview of the average runtimes of the implemented analysis processes.

Protocols / Processes	2MB $\approx 18,000$ lines	20MB $\approx 180,000$ lines	100MB $\approx 900,000$ lines
Rule-based Analysis	$2.48s$	$1m37s$	$11m49s$
Statistical Analysis	$7.65s$	$56.17s$	$4m20s$
Support Vector Machine	$2.44s$	$14.49s$	$1m44s$

The rule-based analysis seems to be the slowest in regard of the other processes. But a detailed view into the process structure clearly shows the cause: Even though the process iterates through the uploaded protocols once, the whole rule set has to be applied to every line which leads to a significant increase of the runtime by a factor n determining the size of the rule set.
The benefit of the statistical analysis is that it first computes all necessary statistics out of the uploaded protocols and the following comparison process does not have to iterate through all lines, but has a consistent set of statistics which does not grow by the size of protocols but rather by the amount of users within the protocols.
The used support vector machine has by far the best runtime which can be put down to the massive mathematical optimization and the not sole dependence on the size of the uploaded protocols, but also to the dependence on the internal used support vectors. The attached scaler contributes to the runtime as well due to the adjustment of the data itself.

5 Conclusions

5.1 Summary

The goal of this thesis is two-fold. The current rule-based detection system has to be reimplemented to obtain an improved adaption to the provided protocols and then new practicable methods for anomaly detection can be researched. Before the protocols can be used a special tool is needed due to terms of privacy which is able to pseudo-anonymize the protocols. Since multiple protocols containing the same employees have to be anonymized, the functionality of loading and saving the applied mapping gets implemented so that every client has its consistent anonymity.
The first step after the anonymization is to create a simple but transparent graphical interface for the prototype which assures a straightforward usage. This is followed by the implementation of a rule-based detection system which is based on the current system and represents the signature detection component. Three different rule types are created: The plain rule which can detect the appearance of special feature and value combinations, the relation rule which can calculate customized ratios with the help of two plain rules and the time rule scanning the time axis to detect the fulfillment or the breach of a condition within a specific time interval. The generic rules achieve a perfect adaption to the given protocols due to their modular structure and simplicity. After the full reimplementation, the statistical analysis is the next method for finding anomalies. Therefore, a training is added which can be used to create a baseline profile of every user in the uploaded protocols. The profiles consist of averages in combination with standard derivations and some ratios e.g. the fail login ratio. To finally detect anomalies, the user can perform a statistical analysis on the uploaded protocols. First, the corresponding values are calculated and with the help of a separate threshold file a comparison can be done which outputs a log of all anomalies as a file. The last part is about anomaly detection in the sense of artificial intelligence. Several attempts are made to apply AI methods on the given protocols. Attempts on clustering and neural networks failed due to the absence of appropriate feature sets. But with some handmade anomalies in the context of failed login attempts, a support vector machine could successfully be implemented. After a short SVM kernel evaluation, the SVM is able to classify anomalies very fast and reliable on condition that the handmade anomalies fit to the data which should be classified.

In summary, the rule-based detection system can be used to intercept almost every possible anomaly within the provided protocols. This process is assisted by the statistical analysis which is able to find general deviations in the users' behavior and is thereby detached from the requirement of known anomalies or labeled data. A soon as an explicit anomaly context is known, the support vector machine can be used to generate an optimal classification.

5.2 Problems

During the thesis a few problems appeared which do not impede the analysis or the tool development, but they lead to a few difficulties during the application of new methods as analysis techniques.
The most profound discovery is about the nature of potential anomalies within the provided protocols. Every entry only exist because of a user who performed a per se legal action. The problem which occurs is that an anomaly can not finally be labeled as an anomaly, because in another situation it could be a normal work procedure. Those anomalies within such structures are called context anomalies. They can not be exposed by just looking at it, since the specific entry can be a legal or illegal one. For example, ten entries in a protocol represent ten failed login attempts. It could be a user who forgot his password but it could also be an attacker who tries to bruteforce a password of a colleague. This fact will compulsory lead to a higher rate of false alarms and has to be considered.
To support the analysis process protocols from the last two years are available (2017-2019). But as no anomalies were found in the past years, there is no guarantee that all protocols are completely free of anomalies. In other words there could have been an attacker who successfully attacked an employee and uses the account. In this case, the data would be falsified and this is undetectable without any training data as a clear reference. That is why some kind of time border has to be determined which defines the set of protocols without any anomalies per definition. It is a hard but necessary step to gain training data which will be used to analyze future protocols without any doubts. All protocols until 2019 are assumed as anomaly-free and the protocols from 2019 are only used to evaluate the analysis processes.
Most of the machine learning algorithms only work on labeled data which means that it is a clear matter whether a single entry is an anomaly or not. These algorithms use prior data to learn some kind of border or separation. But without these labels it is not possible or very hard. The given protocols do not have labels and only contain, if any, context anomalies. That means the standard machine learning algorithms cannot be applied. Even unsupervised learning methods need a tiny set of labeled data which symbolized

anomalies. At least when it comes to an evaluation of the algorithm results reference data would be useful which clearly shows if the detected anomalies are true positives.

5.3 Discussion

The developed prototype has the in-build functionality to generate a profile of all users within the uploaded protocols. As mentioned in Section 1.3.1, the general recording of user activities is allowed according to the law, but a conscientious usage should go without saying so that the prototype serves only the purpose of anomaly detection.
If there is more time left, several improvements would be possible. The runtime of the rule-based detection system could be decreased by a clever storage of already calculated data. In addition, protocols getting uploaded the first time can be pre-analyzed once and every time they will be used, the data which is needed for the calculation could be loaded instead of a whole calculation process. An improved deletion of unnecessary features is also thinkable, because depending on the current rule set different features are important. Furthermore, due to deviations in the amount of users and their behavior in general, a rule file for every client is an appropriate improvement to achieve an even better adaption to the provided protocols.
More time could also be used to enhance the process of making handmade anomalies by searching for other feature combinations than the amount and ratio of failed login attempts. In this scenario, neural networks are a promising candidate which could be used for classification as well besides support vector machines.

References

[20008] Introduction to anomaly detection. https://iwringer.wordpress.com/2015/11/17/anomaly-detection-concepts-and-techniques/, 2008. Accessed:2019/09/10.

[Bea17] Randy Bean. How big data is empowering ai and machine learning at scale. https://sloanreview.mit.edu/article/how-big-data-is-empowering-ai-and-machine-learning-at-scale/, 2017. Accessed:2019/09/22.

[bui19] builtin. What is artificial intelligence? https://builtin.com/artificial-intelligence, 2019. Accessed:2019/09/13.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[Che17] Frank Chen. Ai, deep learning, and machine learning: A primer. https://a16z.com/2016/06/10/ai-deep-learning-machines/, 2017. Accessed:2019/09/14.

[Dee] DeepMind. Alphago. https://deepmind.com/research/case-studies/alphago-the-story-so-far. Accessed:2019/09/20.

[EAP+02] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*, pages 77–101. Springer, 2002.

[EC07] Ozgun Erdogan and Pei Cao. Hash-av: Fast virus signature matching by cache-resident filters. *Int. J. Secur. Netw*, 2:50–59, 2007.

[Eis18] Thomas Eisenbarth. Cybersecurity lecture. Moodle Course, Universität zu Lübeck, 2018.

[ERKL16] Sarah M Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134, 2016.

[KM09] Kevin S Killourhy and Roy A Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, pages 125–134. IEEE, 2009.

[KV03] Christopher Kruegel and Giovanni Vigna. Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 251–261. ACM, 2003.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[LGT97] Steve Lawrence, C Lee Giles, and Ah Chung Tsoi. Lessons in neural network training: Overfitting may be harder than expected. In *AAAI/IAAI*, pages 540–545. Citeseer, 1997.

[LS98] Wenke Lee and Salvatore Stolfo. Data mining approaches for intrusion detection. 1998.

[MHL94] Biswanath Mukherjee, L Todd Heberlein, and Karl N Levitt. Network intrusion detection. *IEEE network*, 8(3):26–41, 1994.

[Nie15] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA:, 2015.

[Ped14] Fabian Pedregosa. Surrogate loss functions in machine learning. `http://fa.bianp.net/blog/2014/surrogate-loss-functions-in-machine-learning/`, 2014. Accessed:2019/09/11.

[PP07] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470, 2007.

[Ray17] Sunil Ray. Understanding support vector machine algorithm from examples. `https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/`, 2017. Accessed:2019/08/29.

[Rep17] GDPR Report. Data masking: anonymization or pseudoanonymization? `https://gdpr.report/news/2017/09/28/data-masking-anonymization-pseudonymization/`, 2017. Accessed:2019/08/27.

[Sin18] PDPC Singapore. Guide to basicdata anonymisation techniques. 2018.

[Spr14] Vincent Spruyt. About the curse of dimensionality. `https://www.datasciencecentral.com/profiles/blogs/about-the-curse-of-dimensionality`, 2014. Accessed:2019/09/15.

[TJ03] Marina Thottan and Chuanyi Ji. Anomaly detection in ip networks. *IEEE Transactions on signal processing*, 51(8):2191–2204, 2003.

[WL90] Bernard Widrow and Michael A Lehr. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990.