Technische Universität Berlin

Outlier Intrusion Detection System

Intelligente Sicherheit in Netzwerk (InSiN) Project Report
Fachgebiet Agententechnologien in betrieblichen Anwendungen und der Telekommunikation (AOT)
Prof. Dr. Dr. h. c. Sahin Albayrak
Fakultät IV Elektrotechnik und Informatik
Technische Universität Berlin

submitted by
Alwin Alwin, Dmytro Gerasymchuk,
Evgeny Bardin, Nesrine Doghri

Group-03

Supervisor: Dr.-Ing. Karsten Bsufka

Alwin Alwin, Dmytro Gerasymchuk,
Evgeny Bardin, Nesrine Doghri

Contents

List of Figures

Chapter 1

Introduction

Due to the rapidly expanding IT-infrastructure and ,thus, its complexity, IT-security has become a concern that poses increasingly difficult challenges for IT-security experts in protecting the system from malicious attacks. One major contributing factor is the growing complexity of cyber-attacks which, assisted by automation development, lowers the required knowledge to perform such attacks even further [11]. In order to deal with this phenomena, various cyber-defence mechanisms and tools have been developed over the years, one of which is *intrusion detection system*.

An intrusion detection system (IDS), proposed by Denning in 1987 [4], is a system that aims to detect intrusion (security-breach) into a system and fires an alert in order for security-personals to decide on necessary course of actions. While there has been significant amount of research in the field of IDS, it remains a young research-field and is still plagued by several weaknesses, such as high false positive rates.

This project aims to tackle the task of developing an IDS system in order to be familiarised with the challenges of IDS-development as well as to be able to recognise both the strength along with the weaknesses of different IDS-strategies/-mechanisms. The goal of this project is to develop an IDS system that has the capability to identify malicious (anomaly) behaviour in the IT-infrastructure and provides an alert accordingly.

This project report is structured as follows. Chapter 2 explains some necessary background that is required to understand the work of this project, Chapter 3 sets the objectives which are relevant in order to achieve the goal of this project; Chapter 4 describes the approach of the authors to reach the objectives; The actual implementation details are explained in Chapter 5 whereas the performance of the implementation is described in Chapter 6. Finally, Chapter 7 sums up this project and provides some insight to future works related to this project.

Chapter 2

Background

In computer security, intrusion detection is the fact of detecting malicious actions that attempt to compromise the confidentiality, integrity or availability of a resource. Intrusion detection can be performed manually or automatically. In the process of manual intrusion detection, a human examines log files to find any suspicious sign that can indicate an intrusion. A system that performs the automated intrusion detection is called intrusion detection system (IDS).

In this chapter, we will take a deeper look on the Intrusion detection systems. We will explain the possible categories of those systems in the first section, present the available datasets to evaluate them in the second section and give a brief overview of the algorithms used in our project in the remaining sections.

2.1 Intrusion detection systems (IDS)

An intrusion detection system is a mechanism intended to analyse the security status of an information system in order to detect anomalies or suspicious activities.

2.1.1 Typical Components

As described in [2], the software components of an Intrusion Detection System can be be grouped, according to their functions, into four types:

- Sensors: They are responsible for collecting events from host or network-based data sources and transmitting them for further analysis and storage.

- Evaluation stations: They analyze the events and identify the attacks. The evaluated informations will be also transmitted as events, which allows repeated analysis.

- **Management stations:** They regulate the communication between the individual components of an IDS, determine the necessary parameters, monitor the individual event streams and send warning messages.

- **Database Components:** They are used to efficiently manage and store the collected data of all components.

2.1.2 IDS Taxonomy

As presented in figure 2.1, IDS systems could be classified into four categories.

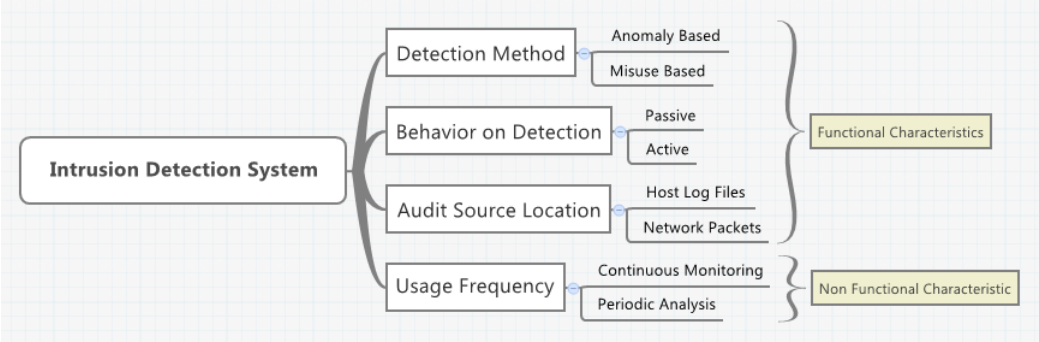

Figure 2.1: Taxonomy of intrusion detection systems

The network security tool uses either of two main approaches to detect the anomalies. The first one, anomaly based, explores issues in intrusion detection associated with deviations from normal behaviour of the monitored system. The second approach uses informations about attacks to discriminate between anomaly or attack patterns. After detecting an attack, an active IDS tool can automatically block suspected attacks in progress (known as an intrusion response system), whereas a passive IDS can only alert an operator to potential vulnerabilities and attacks and log network packets.

Intrusion detection systems can be classified according to the source of data. An IDS tool can either use informations derived from a single host: host based IDS (HIDS) and or exploit informations obtained from a whole segment of a local network (network based IDS, i.e. NIDS).

An IDS tool can be characterized by its usage frequency, we distinguish between a dynamic IDS that runs on a continuous feed of information and performs real-time analysis and a static IDS that periodically takes a snapshot of the environment and analyzes it.

2.1.3 Detection Methodologies

In this section we discuss the primary classes of detection methodologies: Misuse-based based and Anomaly based.

2.1.3.1 Misuse-based intrusion detection

The Misuse-based intrusion detection system possesses patterns describing known attacks called signatures. It is the simplest detection method as it identifies possible attacks by just comparing those signatures against observed events.

While these systems are very effective at uncovering known attacks with a few number of false alarms, they are not able to detect novel attacks for which the signatures are not yet available, therefore the signatures must be constantly updated. Two approaches are commonly used to implement this technique:

- **Pattern matching approach:** In this approach attack patterns are modelled, matched and identified based on the packet header and/or packet content.

- **Rule-based approach:** This approach makes use of expert systems that contain a set of rules describing the attacks, which are matched against audit or network traffic data. Any deviation in the rule matching process is reported as an intrusion.

2.1.3.2 Anomaly detection

Anomaly detection constructs normal activity profiles for the system during a training phase and then considers any current behavior that does not match the stored profile as a suspicious action. These systems are able to detect novel attacks, but they generally have a substantial false alarm rate and the training of a very dynamic system requires a constant update of the normal behaviour profile database. According to [14] anomaly detection techniques can be grouped into three main approaches:

- **Statistical-based techniques:** A user or a system profile reflecting its behavior is constructed by measuring a number of variables sampled over time.

- **Knowledge based techniques:** They require the availability of prior knowledge. The expert system approach is one of the most widely used Knowledge based schemes.

- **Machine Learning techniques :** They are widely used in the anomaly based intrusion detection since they enable to automatically construct a model that reflects the subject's normal behavior. They fall into two categories: supervised and unsupervised learning.

- **Supervised learning:** It proceeds in two steps, first in the training phase a classification model is learned based on datasets labeled with predefined classes . Then, in the testing phase, ths classification model is used to predict the classes of new instances. SVM, neural networks, Bayesian network are some of the widely used supervised learning techniques. Section 2.4 will be dedicated to better explain the support vector machines.
- **Unsupervised learning:** They try to detect intrusions without using any prior knowledge of attacks or normal instances. Clustering and outlier detection based techniques are among the most popular approaches suggested. Among the clustering strategies we find the K-means approach which will be further discussed in section 2.5.

2.1.4 Data source locations

Intrusion detection systems can be categorised into two groups based on the location of the data source: network-based IDS or host-based IDS.

Host based IDS

In this case, informations on the activities of the users of a given machine are gathered based only on the Host audit sources, for example: system commands like: *vmstat*, *pstat*; Syslog audit service.

Network based IDS

A network-based IDS collects informations from the network traffic streams as data is transmitted between particular network segments, suspicious activities are identified inspecting the contents and header informations of all the packets transferred across those segments. In the following we'll give a brief overview of the TCP/IP protocol suite and then we will cover the major components of network-based IDSs.

1. **Transmission Control Protocol**

   To provide network communication the internet protocol stack is composed of 4 layers: the data link, network, transport, and application layers. When a user wants to transfer data across networks, the data is passed through all the layers starting from the application layer, with each layer encapsulating more header informations. The data link layer sends the packet through the physical network; when received by a host the data gets decapsulated as it is passed up through the layers.

The TCP protocol is a connection oriented transport protocol, that is, the sender and the receiver processes must handshake with each other by exchanging control messages initialzing many state variables contained in the TCP header before establishing the connection.

Figure 2.2 depicts the TCP headers as defined in RFC 793.

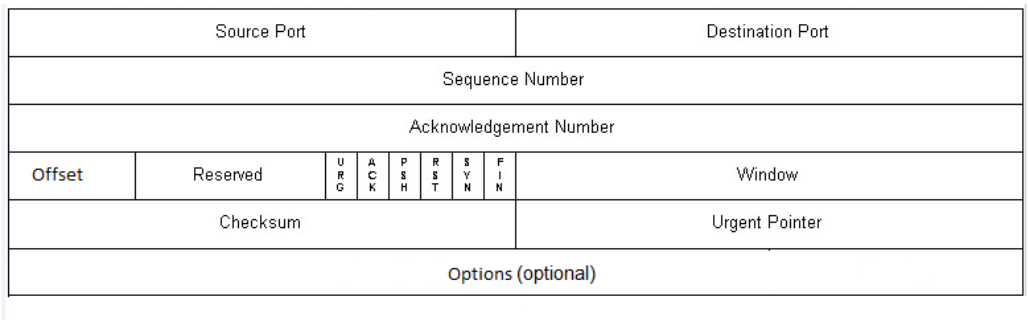

Figure 2.2: TCP Header

The source and destination ports contain respectively the source and the destination port number of the packet. For each TCP connection, numbers are negotiated between the communication partners. The sequence number field contains the number of first byte in segment's data. The acknowledgment number is the sequence number of the next byte expected from the other side, thus the sender can determine whether the data arrived at the receiver.

2. **Architecture overview:**

The network IDS usually has two logical components: the sensors and the management station [1].

- **The sensors:** They are on the network segments, they are dedicated to monitor them for suspicious traffic. Their network interface is set in the promiscuous mode, which means they receive all network traffic. If they detect a suspicious packet, they send it to the management station.
- **The management station:** It receives alarms from the sensor(s), stores them in the Simple Network Management Protocol (SNMP) Management Information Base (MIB). It can perform some additional analysis and display them to the network administrator.

The placement of the sensors and the management station is shown in Figure 2.3.

Flag	Meaning	Function
URG	Urgent	Set if data is urgent
ACK	Acknowledgment	Acknowledges the received data
PSH	Push	Informs the receiving host that the data should be immediately pushed up to the receiving application
RST	Reset	Reset the connection in response to an invalid segment
SYN	Synchronise	Initiates a connection SYN = 1, ACK = 0 for connection request; SYN = 1, ACK = 1 for acknowledgment of receipt
FIN	Finish	Closes a connection FIN = 1, ACK = 0 Disconnection request; FIN = 1, ACK = 1 Confirmation

Table 2.1: TCP Flags

2.2 Available Datasets

Datasets are utilized to evaluate the performance of Intrusion Detection Systems. There are two publicly available datasets: The DARPA datasets developed by the MIT Lincoln Labs and the KDD Cup dataset derived from them, they both consist of US air force local network data with a variety of simulated attacks.

2.2.1 DARPA Datasets

The DARPA datasets were built by MIT Lincoln Laboratory in 1998 and 1999 in the context of a project supported by DARPA aiming to provide a source of data that enables to quantitatively evaluate Intrusion detection systems. They performed a simulation of an Air Force base connected to the Internet.

The 1999 Off-line Simulation Network architecture is depicted in figure 2.4. As shown in the figure, the simulation network has two segments where the left side represents the networks inside of the simulated Eyrie Air Force Base, it includes victim

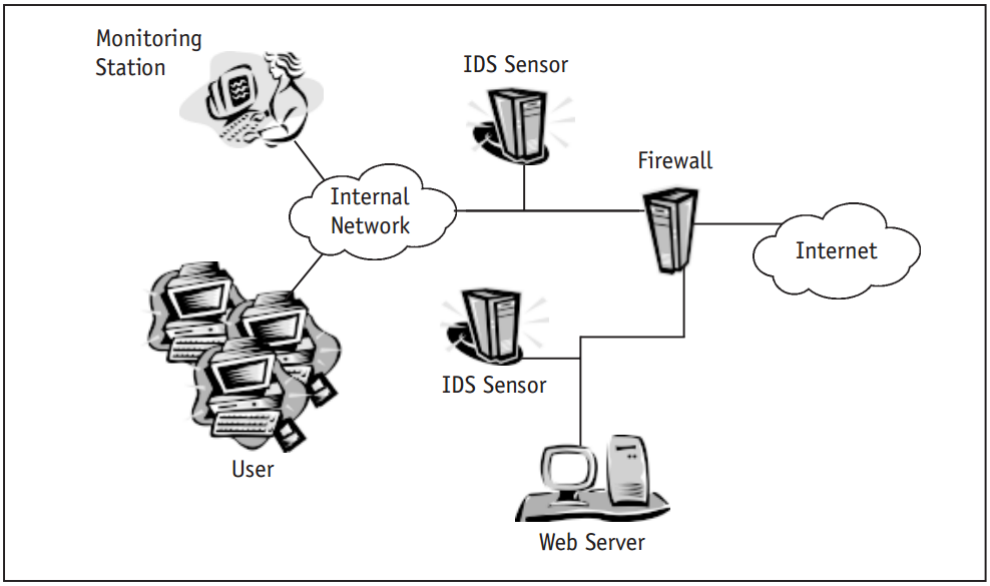

Figure 2.3: Network based IDS architecture [1]

machines with different operating systems and a gateway to other inside workstations and the right side represents the Internet outside the base.

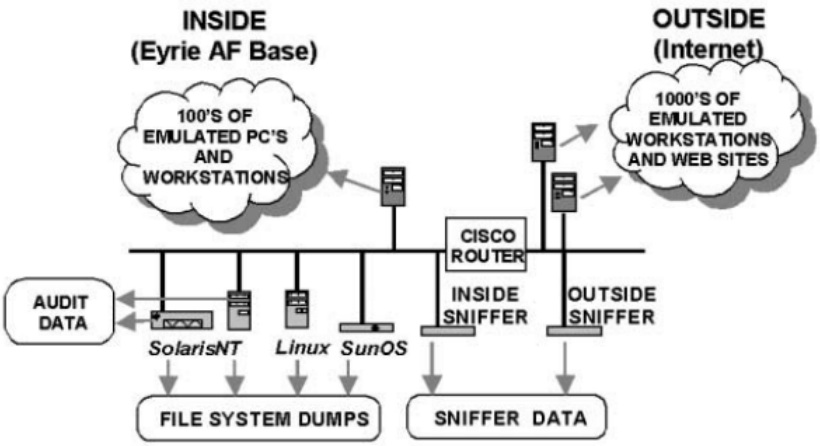

Figure 2.4: DARPA'99 simulation network architecture
[15]

The test bed network generated three weeks of training data with background traffic and labeled attack, the first and third weeks of the training data can be used in the training of anomaly detection systems as they do not contain any attacks, and two weeks of

test data with background traffic and unlabelled attacks. The attacks used in the simulations can be divided into four categories [5], namely:

- Probing: Attacks that attempts to gather information about network of computers or find known vulnerabilities.
- Denial of Service (DoS): Attacks that deny legitimate requests from legal users on the system by making full usages of computing or memory resources.
- User to Root (U2R): Attacks that exploit the vulnerability of the system to obtain root access to it from a normal user privilege.
- Remote to Local (R2L): Attacks that exploits the vulnerability of the system to gain a local user account launching remote exploits.

The procedures used in building the dataset and in performing the evaluation were criticized by MacHugh [6].

2.2.2 KDD Dataset

The KDD dataset is a labeled version of 1998 MIT DARPA dataset where data packets forming a complete session were gathered in a single feature vector. KDD records have 41 features, defined for each connection samples which can further be classified into three groups namely: basic features, traffic features, content features [8]. The goal of this dataset was to demonstrate the applicability of different knowledge discovery and machine learning techniques on intrusion detection systems.

2.3 Principal Component Analysis

Principal Components Analysis is a technique introduced by Pearson and Hotelling that aims to reduce the dimensionality of multivariate data while preserving as much as possible of the present variation.

It is an unsupervised learning approach that only relies on the input data. PCA transforms a large number of variables to a new reduced set of variables, the principal components, that are uncorrelated and linear functions of the original variables, such that, the greatest variance by any projection of the data comes to lie on the first coordinate, the second greatest variance on the second coordinate, and so on.

In practice, this is achieved by computing the covariance matrix for the full data set. Next, the eigenvectors and eigenvalues of the covariance matrix are computed, and sorted according to decreasing eigenvalue. This approach can be inappropriate for some applications as features with low variance might actually have high predictive relevance [17].

2.4 Support Vector Machines

SVMs have been introduced in 1992 at the COLT conference by Boser, Guyon, Vapnik as a supervised learning approach used to solve classification and regression problems.

The simplest kind of support vector machines called Linear SVM aims to find a hyperplane that optimally separates the training vectors into two classes by achieving the maximum separation, such that it will be equidistant to both datasets. Support vectors are defined to be a subset of the training data having the minimum distance to the hyperplane. An example of a simple linear SVM is illustrated in Figure 2.5.

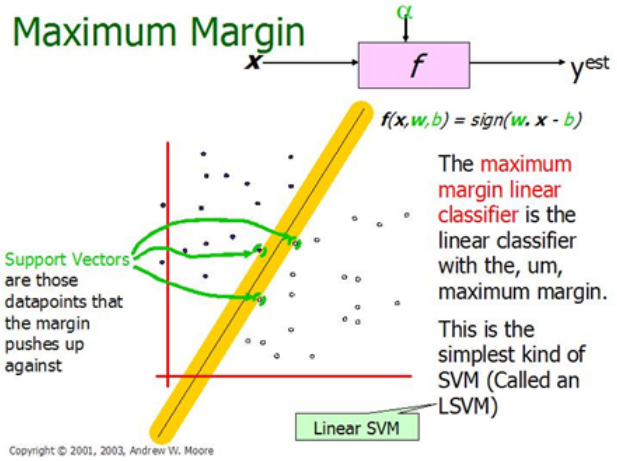

Figure 2.5: Linear Support Vector Machines [13]

The goals of SVM are separating the data with a hyper plane and extend this to non-linear boundaries when the data is far from linear and the datasets are inseparable by using the kernel methods that allow SVMs to form nonlinear boundaries, by non-linearly mapping the input data to a high-dimensional space. The new mapping is then linearly separable [3] [12].

The kernel functions enable operations to be performed in the input space rather than the potentially high dimensional feature space. Various kernel functions can be used [3], such as: *Polynomial*, *Gaussian Radial Basis Function*, as well as *Exponential Radial Basis Function*.

2.4.1 One Class SVM

One Class SVM is an extension of the SVM methodology that handles training using only positive information, it identifies ”outliers” amongst the positive examples and use them as negative examples [10]. The OC-SVM algorithm maps input data into a high dimensional feature space using a kernel function and iteratively finds the maximal

margin hyperplane which best separates the training data from the origin. It may be viewed as a regular two-class SVM where all the training data lies in the first class, and the origin is taken as the only member of the second class. Thus, the hyperplane (or linear decision boundary) corresponds to the classification rule. The decision function is given by: [7]:

$$f(x) =< w, x > +b \tag{2.1}$$

where *w* is a vector perpendicular to the hyperplane and *b* is a bias term. The OC-SVM solves an optimisation problem to find the decision function f with maximal geometric margin. This function can be used to label a test example x as an anomaly if $[f(x) < 0]$ and as normal otherwise. Solving the OC-SVM optimisation problem is equivalent to solving the dual quadratic programming problem:

$$\min_{\alpha} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \tag{2.2}$$

subject to the constraints

$$0 \leq \alpha_i \leq \frac{1}{v_l} \tag{2.3}$$

and

$$\sum_{i} \alpha_i = 1 \tag{2.4}$$

where α_i is a Lagrange multiplier (or "weight" on example *i* such that vectors associated with non-zero weights are called "support vectors" and solely determine the optimal hyperplane), *v* is a parameter that controls the trade-off between maximising the distance of the hyperplane from the origin and the number of data points contained by the hyperplane, *l* is the number of points in the training dataset, and $K(x_i, x_j)$ is the kernel function. Given a feature map:

$$\phi : X \longrightarrow \Re^N \tag{2.5}$$

where ϕ maps training vectors from input space *X* to a high dimensional feature space, the kernel function can be defined as:

$$K(x, y) =< \phi(x), \phi(y) > \tag{2.6}$$

After solving for the dual problem, a set of model weights α_i can be obtained and the decision function for any test vector x can be given by :

$$f(x) = sgn(\sum_{i} \alpha_j K(x_i, x_j) - \rho) \tag{2.7}$$

where ρ is the distance to the origin.

2.5 KMeans

Kmeans is an unsupervised learning algorithm that aims to organise data from a given training set into few clusters. The Kmeans algorithm proceeds in two steps: First K data items from the set of data points are randomly chosen as initial centroids, then, in the second step, each data point is assigned to the cluster which has the closest centroid and each cluster centroid is moved to the mean of the points assigned to it. This second step is repeated until all data points are assigned to one of the K clusters.

The Kmeans algorithm proceeds in two steps: First K data items from the set of data points are randomly chosen as initial centroids, then, in the second step, each data point is assigned to the cluster which has the closest centroid and each cluster centroid is moved to the mean of the points assigned to it. This second step is repeated until all data points are assigned to one of the K clusters.

- Silhouette Value:

A silhouette is a graphical display for partition techniques which shows which objects lie well within their cluster, and which not. The average silhouette width provides an evaluation of clustering validity and might be used to select an 'appropriate' number of clusters [16].
For each data point i, the silhouette $s(i)$ is given by the equation 2.8:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{2.8}$$

where $a(i)$ is the average dissimilarity of i with all other data within the same cluster and $b(i)$ is the lowest average dissimilarity of i to any other cluster, of which i is not a member.

Chapter 3

Objectives

Based on the project goal which is to developed an IDS system to detect anomaly, potentially malicious, behaviours in an IT-system, various objectives were defined, which includes:

- Extract relevant information from given data-sets
- Training of the IDS system using machine learning methods on the extracted information
- Testing of the IDS system on DARPA'99 data-sets

These objectives merely serve as a rough guideline during the IDS-development process. A detailed approach taken by the authors in order to achieve the project goal is described and explained in the next chapter.

Chapter 4

Approach

Having determined the objectives, the first step of the development process is designing the IDS system's architecture. The design involved in this project report is based on the work of Manandhar [9], which is illustrated in Figure 4.1.

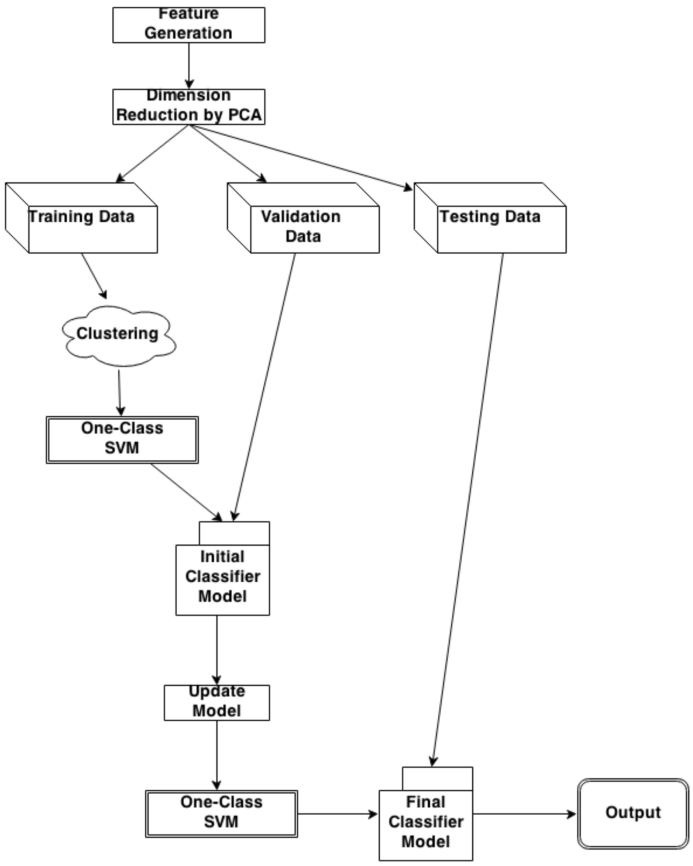

Figure 4.1: Architectural Design of IDS system [9]

The IDS system of this architectural design is set to work on real-time since it performs analysis only based on TCP/IP packet headers. The architecture consists of three main components, namely the data-sets preprocessing component, the training component and the testing (classification) component. This chapter describes the design of each component and its functionality.

4.1 Data-Set Preprocessing Component

The data-set preprocessing component is designed to perform extraction of relevant information from given data-sets. This component aims to increase the performance of the training component by providing only necessary information that is required to train the IDS system. Since the IDS system analyses only TCP/IP packet headers, all packet header information (e.g. payload size, flags, etc.) is extracted from the network packets which is contained in the data-sets. The complete workflow of data-set preprocessing component can be viewed on Figure 4.2.

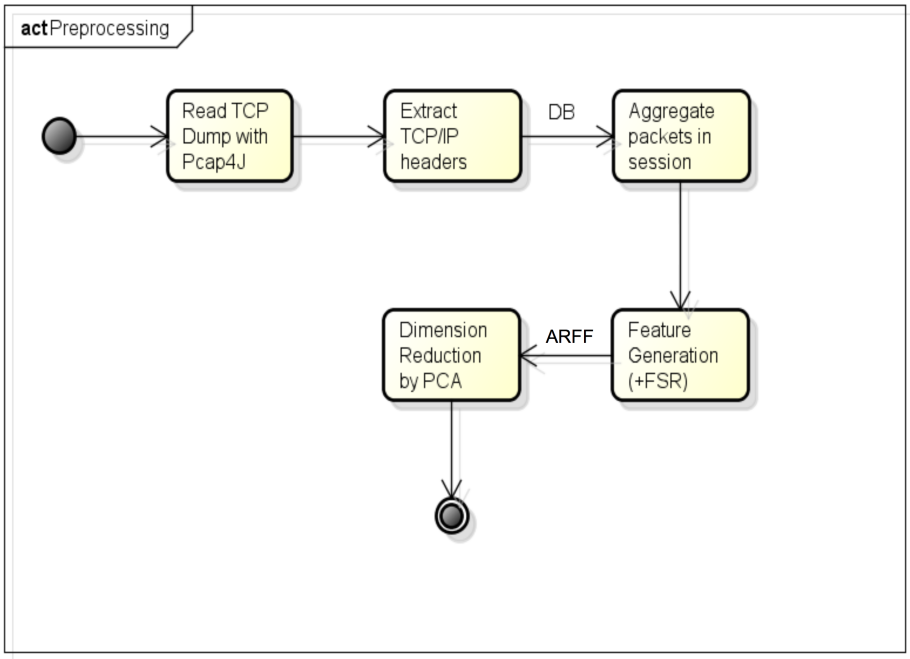

Figure 4.2: Activity diagram describing the workflow of data-set preprocessing component

Preprocessing data-set starts by reading the data-set and converting each network packet into a $Java\ Object$, which simplifies the header information extraction process. Due to the fact that a single abnormal network packet does not provide enough evidence regarding the nature of the network connection, the extracted header information

from single packets has to be aggregated to represent one complete network-connection (*TCP-session*). The aggregation process is done by building an average of each header information (e.g. average payload size, average count of SYN-flags, etc.) over four different characteristic of the network packet, namely *source IP-address*, *source port-number*, *destination IP-address* and *destination port-number*. This means that all network packets with the same characteristics mentioned above are aggregated into one TCP-session.

Afterwards, the header information is utilised as feature vectors for the training component of the IDS system. However, in order to further improve the performance of the training component, the dimensions of these feature vectors are reduced with the help of principal component analysis (PCA). The resulting dimensional-reduced feature vectors are then passed onto the training component to be utilised as input for the IDS system's training process.

4.2 Training Component

Clustering will be performed on the given dimensional-reduced feature vectors using $k\text{-}means$ method in order to determine the cluster centroids which serves as input for the one-class support vector machine. The resulting hyper-plane from one-class SVM is the IDS model, based on which the classification component functions. A corresponding activity diagram of this workflow is illustrated in Figure 4.3.

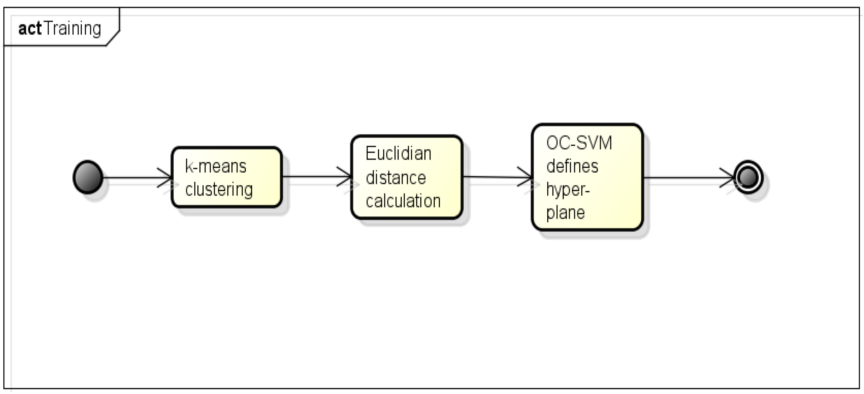

Figure 4.3: Activity diagram describing the training process of training component

Additionally, Manandhar decided on the trade-off between false-positives and false-negatives in the sense that in order to minimise false-negatives, a higher false-positives rate is expected with the design of this IDS system [9]. Therefore, the training component is not only responsible for the training process but also for the validation process, whose activity diagram is described in Figure 4.4.

The validation process aims to heightens the sensibility of the IDS detection capability. This is achieved by running the classification process on a validation data-set and only 'normal' data-points which were correctly classified by the IDS model (*true positive*) is collected to construct the final IDS model whereas other data-points which were either anomaly or wrongly classified (*true negatives*, *false negatives*, and *false positives*) are ignored and left out.

Data-points collected during the validation process are analysed by calculating the distance of each data-point to all cluster centroids and the distance measure is used as an input for the one-class SVM to construct the final IDS model.

4.3 Classification Component

Based on the constructed final IDS model, the one-class SVM is able to classify new data-points. Beside the classification result, performance statistics is also collected in order to evaluate the IDS system afterwards.

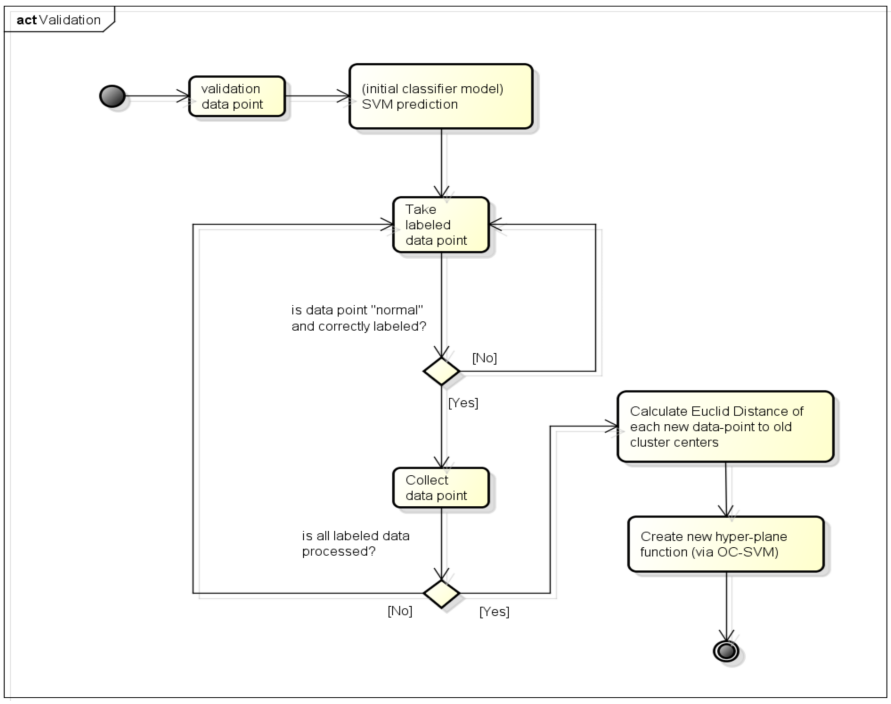

Figure 4.4: Activity diagram describing the validation process of training component

Chapter 5

Implementation

Based on the approach of the authors explained in previous chapter, the implementation of the IDS system also consists of three parts, namely the implementation of data-set preprocessing component, training component, as well as classification component. The subsequent sections explain how each of these components are implemented and describe as well as explain any necessary changes which are made to the initial approach.

5.1 Data-Set Preprocessing Component

The data-set preprocessing component is implemented to not only read the *pcap*[1] files from DARPA'99 data-sets but also to extract necessary relevant packet-headers from each network-packet in order to build feature vectors. Reading pcap files is assisted by the $Java$ library *Pcap4J*[2], which is also able to extract the packet-header information. The extracted packet-header information is saved to the corresponding MySQL database, using the $Java$ standard SQL library.

As soon as all packet-header information is saved to the database, an aggregation of network-packets into TCP sessions is performed. This aggregation process is done by the means of SQL-aggregation query. Following the aggregation process is the dimension reduction of the feature vectors with the aids of PCA. However, due to lack of both know-how and experience as well as time limitation of this project, the authors decided to skip the dimension reduction process and proceeded with clustering process. Another reason to skip the PCA part is that the initial implementation of the dimension reduction process yields results that the authors are unable to understand and explain. Moreover, after several iterations of tests, the PCA part is determined to produce non-deterministic

[1] Pcap (packet capture) files are files that contains captured network traffic in form of network-packets.

[2] Pcap4J is a $Java$ library that is able to work with different aspects of network-packet, including capturing, crafting and sending the network-packets. Homepage of Pcap4J: `http://www.pcap4j.org`

output which further encourages the authors to remove the dimension reduction process from the IDS system. By skipping the PCA part, the next step in the workflow is the training process which is handled by the training component.

5.2 Training Component

Building the initial as well as the final IDS model is the function of the training component. This component performs two main processes, namely the clustering process and the one-class SVM model building process, both of which are the core of the whole training as well as validation process.

Clustering process is performed using k-means method with the help of *Weka*[3] library. Based on the work of Manandhar [9], the optimal cluster size for the DARPA'99 data-sets is three clusters, which is also the settings the authors agreed upon. Afterwards, the data-set preprocessing component calculate the Euclidian distance of each data-point in the training data-set to all three cluster centroids. As a result, each data-point contains three different distance measures, which are then utilised to build an initial IDS model.

The one-class SVM takes the three distance measures of each data-point as input and constructs an IDS model which can be used as the classifier. Similar to k-means, the configuration of one-class SVM ($gamma$, nu, and $degree$) is also taken from the work of Manandhar [9].

Based on the initial IDS model, the validation process is started by extracting building feature vectors from the validation data-set, followed by Euclidian-distance calculation of each validation data-point to the cluster centroids that are already built in the training process. Subsequently, the distance measures are used as input for the one-class SVM to classify the validation data-point. Every validation data-point that is classified as 'normal' and are actually normal (true negative), is collected and further utilised by the one-class SVM as the final IDS model.

5.3 Classification Component

As the name already suggests, the classification component is responsible for classifying a new data-point. This is done by utilising the final IDS Model supplied by the training component. Since the final IDS model contains only information about cluster distances, the IDS system has to trace back the data-point to the original aggregated TCP-session which contains valuable information necessary for the final output (alerts). Fortunately, the implemented IDS system works by preserving order of the data-points. Thus, the index of the final IDS model can be utilised to trace back to the original TCP-session

[3]Weka is a $Java$ library that contains a collection of machine learning algorithms such as PCA, K-means, as well as SVM. Homepage of Weka: `http://www.cs.waikato.ac.nz/~ml/weka/`

contained in the database. Furthermore, the index makes it also possible to retrieve (available) labelling of the data-point so that performance statistics can be collected for the evaluation purposes.

Chapter 6

Results

Following the development process of the IDS system, an experiment was conducted in order to be able to objectively evaluate the performance of the IDS system. The evaluation process begins by choosing the (testing) data-set as input for the one-class SVM who is performing the classification based on the final IDS model. During the classification, several performance statistics such as *true-positives*[1], *true-negatives*[2], *false-positives*[3], *false-negatives*[4], as well as combinations of these measures, i.e. *precision*[5], and *recall*[6]. This chapter explains the data-set that was chosen for the experiment and the evaluation result of the experiment.

6.1 Dataset Used

The publicly available DARPA'99 data-set is chosen for the experiment since this data-set is a widely popular benchmark for IDS and, thus, allows for comparison to other IDS-research works [9]. As mentioned before in Chapter 2, the DARPA'99 data-set consists of five weeks of data, where each week contains five days of collected network traffic. The conducted experiment works with three weeks of data, namely the first week, the fourth week and the fifth week as the training, validation and testing (classification) data-sets respectively. Since the first week data-set contains no attacks, it is most suitable to be used for the IDS training process, whereas the fourth as well as the fifth week data-sets contain attacks and are therefore used for validation and testing (classification) process.

[1] True-positives (TP) are data-points which are classified as anomalies and are actually anomalies.
[2] True-negatives (TN) are data-points which are classified as normal and are actually normal.
[3] False-positives (FP) are data-points which are classified as anomalies but are actually normal.
[4] False-negatives (FN) are data-points which are classified as normal but are actually anomalies.
[5] $Precision = TP/(TP + FP)$
[6] $Recall = TP/(TP + FN)$

Labelling Attacks

In order for the developed IDS system to function properly, every attacks in the fourth and fifth week data-sets have to be labeled accordingly. The labelling of attacks in the data-sets serves the purpose of being able to evaluate the detection rate of IDS systems since otherwise the true-positives, true-negatives, false-positives, as well as false-negatives can not be determined. Unfortunately the search for such (machine-readable) labelling does not yield any promising results. This means that the validation process of the IDS system may produce non-ideal (potentially weak) final IDS model due to the fact that, without proper attacks labelling, the validation process might include false negatives (instead of only true-negatives) into the final IDS model. Additionally and most importantly, the evaluation process of the IDS system is also compromised.

Partially Labelled Attacks

Nevertheless, the work of Manandhar [9] also provides several SQL-scripts for attacks labelling. However, despite of identical database-structure, the SQL-scripts fails to label the attacks. Further analysis of the problem shows that the SQL-labelling-scripts contains time-offset. Consequently, the time-offset is readjusted[7] back to fit the original time-zone of the DARPA'99 data-sets.

Even after the time-offset was readjusted, some of the SQL-labelling-scripts still can not label the attacks properly, for example the fourth week-labelling-script fails to label any attacks. Therefore, the SQL-labelling-scripts only work partially on the fifth week data-set. More precisely, only 63% of 1000 SQL-labelling-commands in the fifth week data-set is properly executed, which amounts to 159000 successfully labelled attacks.

6.2 Evaluation

Testing the developed IDS system on the partially-labeled fifth week of DARPA'99 data-set yields a recall rate of 100% and 30% precision which is satisfying nevertheless since the high recall rate is the aim of the IDS system's architectural design [9]. Moreover, with proper and complete attacks-labelling, a much higher precision rate can be achieved.

It is also to be noted that the sensible IDS-system that achieves high detection rate always make the trade-off of having high false-positives as well, as in the developed IDS system within this project. In order to verify and/or estimate the false positive rate, the developed IDS system is given the first week of DARPA'99 data-set, which contains no

[7]The time-offset readjustment is performed by adding 6 hours of the *packet-time* to each packet's time-stamp.

attacks. As a result, 80% of the data-points are labelled as normal, where as the rest were mis-labelled as anomalies, which corresponds to approximately 20% false-positives.

Chapter 7

Conclusion

During the course of this project all of the objectives set in chapter 3 have been achieved and, therefore, the project goal was met. The authors have had first hand experience in the development of an IDS-system and how the focus-point of the IDS system can shape the architectural design which, in turn, may heavily determine the performance of the IDS system in regards of precision and recall rate. The missing attacks-labelling hinders the performance of the developed IDS system since the validation process requires labelled attacks in order to construct the final IDS model properly.

Nevertheless the evaluation is performed on partly labelled data-set which results in succesfull identification of all attacks in the testing data-set. However, as described in Chapter 6, such a high recall rate is accompanied with a low accuracy rate (high false-positives). Future works in this project might involve tinkering with the one-class SVM settings in order to achieve a higher accuracy rate with the minimal expense of lower recall rate. Additionally, introducing PCA into the IDS system properly may substantially decrease the required time to complete the training and validation process as currently the IDS system require approximately 10 hours to perform the training and validation process. Lastly, a Graphical User Interface would increase the usability of the developed IDS system.

Bibliography

[1] Intrusion detection systems buyer guide. Technical report, ICSA Labs.

[2] Einfuehrung von intrusiondetection-systemen, grundlagen, 2002.

[3] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. 0070428077. The Press Syndicate at the University of Cambridge, March 2000.

[4] Dorothy E. Denning. An intrusion-detection model. *IEEE Trans. Softw. Eng.*, 13(2):222–232, February 1987.

[5] Farnaz Gharibian and Ali A. Ghorbani. Comparative study of supervised machine learning techniques for intrusion detection. 2007.

[6] John Mc Hugh. Testing intrusiondetection systems: Acritique of the 1998 and 1999 darpa intrusion detection systemevaluations as performed by lincoln laboratory. *ACM Transactions Information System Security*, 3(4):262–294, Nov 2000.

[7] Svore Angelos D. Keromytis Salvatore Katherine A., Heller Krysta M. and J. Stolfo. One class support vector machines for detecting anomalous windows registry accesses. *ICDM Workshop on Data Mining for Computer Security, Melbourne, FL*, 2003.

[8] Wei Lu Mahbod Tavallaee, Ebrahim Bagheri and Ali A. Ghorbani. A detailed analysis of the kdd cup 99 data set. *IEEE Symposium on Computational Intelligence in Security and DEfence Applications*, 2009.

[9] Prajowal Manandhar and Zeyar Aung. Towards practical anomaly-based intrusion detection by outlier mining on tcp packets. In Hendrik Decker, Lenka Lhotská, Sebastian Link, Marcus Spies, and Roland R. Wagner, editors, *Database and Expert Systems Applications*, volume 8645 of *Lecture Notes in Computer Science*, pages 164–173. Springer International Publishing, 2014.

[10] Larry M. Manevitz and Malik Yousef. One-class svms for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

[11] John McHugh. Intrusion and intrusion detection. *International Journal of Information Security*, 1(1):14–35, 2001.

[12] Tom M. Mitchell. *Machine Learning*. 0070428077. McGraw-Hill Science / Engineering / Math, March 1997.

[13] Andrew W. Moore. Support vector machines, Nov 2001.

[14] G. Macia-Fernandez E.Vazquez P. Garcia-Teodoro, J.Diaz-Verdejo. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers and security*, 2009.

[15] David J. Fried Jonathan Korba Richard Lippmann, Joshua W. Haines and Kumar Das. Analysis and results of the 1999 darpa off-line intrusion detection evaluation. *Springer-Verlag Berlin Heidelberg*, 2000.

[16] Peter J. ROUSSEEUW. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1986.

[17] Martin Sewell. Principal component analysis. 2007.