Labelling clusters in an anomaly based IDS by means of clustering quality indexes

Roger Storløkken

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2007

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Abstract

Intrusion detection systems have in the recent years become an important part of most defence-in-depth network perimeter security setups. These intrusion detection systems either apply misuse or anomaly based techniques to detect malicious activities. To detect malicious activity, misuse based systems search for signatures of previously known malicious activities. Anomaly based systems on the other side, flag behaviour that deviates from what is expected to be benign activities. Most intrusion detection systems are misuse based, but more and more systems now use a combination of both techniques, where anomaly based techniques are often used to assist the misuse based detection engines.

One method for classifying observed activities in an anomaly based intrusion detection system is clustering. A major difficulty regarding clustering based systems is to interpret the nature of the obtained clusters and label them as normal or malicious. One labelling strategy is to apply clustering quality evaluation techniques to determine whether or not a massive attack is present in the observed activities. Then, a combination of these evaluation techniques and cluster parameters can be used to interpret the nature of the clusters.

This thesis investigates the application of Dunn's index and C-index in this labelling strategy. These clustering quality indexes are combined with clustering parameters to develop algorithms for proper labelling of observed activities. The original definition of the C-index requires similar cluster sizes for accurate clustering quality evaluations. As similar cluster sizes are rare in clustering based intrusion detection systems, we propose two modified C-indexes for use in situations with unequal cluster sizes.

A prototype, simulating a clustering based intrusion detection system, has been developed to test and measure the performance of clustering quality indexes in this labelling strategy. The results from these simulations show that the application of both the Dunn's index and the modified C-index in this labelling strategy produce accurate labelling of activity clusters. These results also demonstrate that this labelling strategy outperforms classical cardinality based labelling strategies in heavily attacked environments where massive attacks are frequent.

Keywords: Information security, Intrusion detection, Anomaly detection, Clustering, Clustering quality evaluation, Dunn's index, C-index, Labelling strategy.

Sammendrag

Innbruddsdeteksjonssystemer har i de siste årene blitt en viktig komponent i løsninger for nettverkssikkerhet. Disse systemene bruker enten signaturbaserte eller anormalitetsbaserte teknikker for å detektere angrep. Signaturbaserte systemer bruker signaturer av tidligere kjente angrep for å detektere ondsinnet aktivitet. Anormalitetsbaserte systemer derimot, etablerer en grunnlinje med forventet normal aktivitet og genererer en alarm hvis observert aktivitet avviker fra denne grunnlinjen. De fleste systemer er i dag signaturbaserte, men trenden er at anormalitetsbaserte systemer ofte blir brukt i kombinasjon med signaturbaserte systemer.

En metode for å klassifisere aktivitet i et anormalitetsbasert system er clustering, dvs. dannelse av såkalte klynger. Et problem som må løses for å bruke clustering i innbruddsdeteksjonssystemer er å tolke hvorvidt en klynge består av normal aktivitet eller angrep, såkalt merking av klynger. En strategi for å løse dette problemet, er å bruke teknikker som undersøker kvaliteten på klyngen for å kontrollere hvorvidt det finnes et massivt angrep i de observerte aktivitetene. Det er da mulig å kombinere disse teknikkene med andre fysiske egenskaper til klyngene for å lage algoritmer som tolker innholdet i klyngene.

I denne masteroppgaven har vi undersøkt bruken av Dunn's index og C-index i en slik strategi for merking av klynger. Disse teknikkene, som evaluerer kvaliteten til klyngene, er brukt i kombinasjon med andre fysiske egenskaper for å utvikle algoritmer for merking av observert aktivitet. C-index krever i sin originale definisjon klynger av lik størrelse for å gi riktig evaluering av kvaliteten på klyngen. Ettersom lik størrelse på klyngene er svært uvanlig i clusteringbaserte innbruddsdeteksjonssystemer, presenterer vi en modifisert C-index som håndterer situasjoner med ulike klyngestørrelser.

En prototype, som simulerer et clustering basert innbruddsdeteksjonssystem, har blitt utviklet for å teste og evaluere ytelsen til de to evalueringsteknikkene i den nevnte strategien for merking av aktivitet. Resultatene fra disse simuleringene viser at bruken av både Dunn's index og C-index gir nøyaktig merking i denne strategien. Resultatene viser også at strategien, sammenliknet med tradisjonelle strategier for merking, er bedre egnet for merking av aktivitet i situasjoner hvor massive angrep er vanlig.

Acknowledgements

I would like to thank my supervisor, Professor Slobodan Petrović, for great guidance throughout the work on this master's thesis. His help and comments, and the interesting discussions concerning the research topic, have been invaluable for the outcome of this thesis. My classmates at the master's lab (A220) at Gjøvik University College also deserve thanks for interesting discussions, and for making the work on this thesis an enjoyable and memorable time. I would also like to thank my fellow student Katrine Aam Svendsen for valuable discussions and feedback regarding this report.

– Roger Storløkken, 20th June 2007

Contents

Abstract . . . . . **iii**
Sammendrag . . . . . **v**
Acknowledgements . . . . . **vii**
Contents . . . . . **ix**
List of Figures . . . . . **xi**
List of Algorithms . . . . . **xiii**
List of Tables . . . . . **xv**
1 Introduction **1**
1.1 Topic 1
1.2 Research problem 1
1.3 Justification and motivation 2
1.4 Research questions 2
1.5 Summary of claimed contributions 2
1.6 Research methodology 3
1.7 Terms and definitions 4
1.8 Outline 5
2 Related work **7**
2.1 Cluster analysis 7
2.2 Clustering quality indexes 8
2.3 Unsupervised anomaly intrusion detection 10
2.4 Testing intrusion detection systems 15
2.4.1 The KDD Cup 99 data set 15
3 Application of Dunn's index and C-index for labelling clusters **19**
3.1 Dunn's index 22
3.1.1 Time complexity 23
3.2 C-index 23
3.2.1 Modified C-index 24
3.2.2 Outliers in the clustering 24
3.2.3 Time complexity 27
3.3 Labelling algorithms 27
4 Experimental work **31**
4.1 The prototype 31
4.1.1 Computation of Dunn's index 33
4.1.2 Computation of the modified C-index 34
4.2 Experimental setup 34
4.3 Expectations 36
4.4 Experiments 37
4.4.1 Exp. 1: Labelling clusters by means of cluster cardinality 38
4.4.2 Exp. 2: Labelling clusters by means of Dunn's index 38
4.4.3 Exp.3: Labelling clusters by means of modified C-index, C-mean . . 38

4.4.4 Exp. 4 and 5: Labelling clusters by means of modified C-index, C-small . . . 38
4.5 Results . . . 39
4.5.1 Labelling clusters by means of cluster cardinality . . . 39
4.5.2 Labelling clusters by means of Dunn's index . . . 40
4.5.3 Labelling clusters by means of modified C-index index, C-mean . . . 42
4.5.4 Labelling clusters by means of modified C-index index, C-small . . . 44
4.5.5 Clustering errors . . . 47
4.6 Discussion . . . 48
5 Conclusions . . . **53**
6 Further work . . . **55**
Bibliography . . . **57**
A Samples of the prototype source code . . . **61**
A.1 Implementation of Dunn's index . . . 62
A.2 Implementation of the modified C-index . . . 63
A.3 Relabelling algorithm for Dunn's index . . . 65
A.4 Relabelling algorithm for the modified C-index . . . 66
B Optimal system accuracy . . . **67**

List of Figures

1 Illustration of the clustering of records 452000-453000 26
2 Illustration of the clustering of records 41000-42000 26
3 The design of the multiple classifier prototype 32
4 ROC curve describing the system accuracy (Cardinality based labelling) . . 39
5 ROC curve describing the system accuracy (Dunn's index) 40
6 ROC curve describing the system accuracy (modified C-index, C-mean) . . 42
7 ROC curve describing the system accuracy (modified C-index, C-small 1) . 44
8 ROC curve describing the system accuracy (modified C-index, C-small 2) . 46
9 Summary of the accuracy of the labelling algorithms 48

List of Algorithms

1 Labelling algorithm with Dunn's index 28
2 Labelling algorithm with the modified C-index 28

List of Tables

1 Difference between normal and attack vector 20
2 Difference between normal vectors 20
3 Difference between two attack vectors within a massive attack 21
4 Difference between two standalone attack vectors 22
5 Partial indexes and clustering parameters from two clusterings 25
6 Clustering errors due to the use of “2-means” 37
7 Labelling with Dunn’s index 41
8 Labelling with the modified C-index; C-mean 43
9 Labelling with the modified C-index; C-small 45
10 KDD CUP ’99 statistics 67

1 Introduction

1.1 Topic

This thesis considers the application of clustering quality indexes in a strategy to handle problems associated with activity labelling in clustering based intrusion detection systems (IDS). IDSs have in the recent years rightfully become an important part of most defence-in-depth network perimeter security setups. There are, in general, two different approaches to interpret and classify activity in an IDS; misuse and anomaly detection. Misuse based systems detect malicious activity by searching for distinct patterns, i.e. signatures, of known malicious activities. Anomaly based systems, on the other hand, flag activities that deviate from what is considered normal in the observed system.

Traditional approaches for anomaly based intrusion detection often use statistical methods based on the models proposed by Denning [10]. The problem with these approaches is that they assume that the observed activities follow a Gaussian distribution. This is not always the case when monitoring computer networks or systems, and causes high error rates. Another problem is that these systems must learn from cleansed data sets to establish a baseline of normal benign activity. This process of supervised learning is very expensive and prone for errors.

To handle these problems, clustering has been proposed as an approach to classify observed data in anomaly based IDSs. Clustering is the art of grouping data together into clusters. The aim is to find clusters where the instances within the same cluster are very similar, while different clusters are very distant from each other. The underlying assumption in intrusion detection is that malicious activities are significantly different from benign activities. It is therefore possible to group benign and malicious activity into separate clusters by means of clustering techniques, which is the aim of an anomaly based intrusion detection engine.

Clustering based techniques for intrusion detection are often referred to as *unsupervised intrusion detection*, because the differences in characteristics of the observed activities are used directly to classify the activities without any initial learning process. These systems do not require any prior knowledge about attack signatures or labelled training data to classify activity. Clustering based systems are therefore highly capable of correctly classifying previously unknown malicious activities, and less vulnerable for errors caused by small changes in user behaviour over time.

1.2 Research problem

A major problem in anomaly detection systems based on clustering, is to determine the nature of the obtained clusters, so-called labelling. The clustering algorithms merely classify observed activities into clusters and do not perform any interpretation of the content of these clusters. We therefore need a strategy to interpret and label the obtained clusters.

A classical labelling strategy is to measure the cardinality of the clusters, and label some percentage of the smallest clusters as malicious. This approach does, however, have some limitations, and does not detect massive attacks properly, e.g. Denial-of-Service

attacks.

Another approach for labelling clusters, which solves the limited capability of detecting massive attacks, is proposed by Petrovic et al. [50, 51, 52]. In their approach, different cluster properties and parameters are measured in order to interpret the nature of the obtained clusters. The underlying assumption is that clustering quality evaluations may indicate the presence of a massive attack in the observed data. Then, by combining these techniques with cluster parameters, e.g. the cluster diameters, it becomes possible to interpret the nature of the clusters. In this thesis, we extend the research from [50, 51, 52], by investigating the application of two well known clustering quality evaluation indexes, Dunn's index and C-index, in such a labelling strategy.

1.3 Justification and motivation

Accuracy and efficiency are two very important performance measures for an IDS. High accuracy is necessary to provide valuable information to the human IDS analyst monitoring the computer system or network. Too high false positive rates will leave the human IDS analyst frustrated, and may cause that important alarms are ignored. In addition to this, it is important for an IDS to work in real-time, or as close to real-time as possible. Real-time operation is e.g. necessary for the human IDS analyst to be able to take countermeasures against attacks in progress, before they can do much harm to the protected systems.

The Silhouette index and the Davies-Bouldin index have already been applied to provide labelling of the clusters produced in a clustering based IDS implemented at GUC[1] [50, 51, 52]. It has been shown that clustering quality indexes can be used to detect malicious activity, but the two indexes performed quite differently [50]. The Silhouette index proved to be slightly more accurate than the Davies-Bouldin index. However, the Silhouette index showed some performance penalties compared to the Davies-Bouldin index, which is much less computationally complex. It is therefore interesting to investigate the performance of other clustering quality indexes applied in this labelling strategy.

1.4 Research questions

The research questions, which we will try to answer in the course of this work, are the following:

1. Can Dunn's index and C-index be applied in a labelling strategy for clustering based intrusion detection systems?
2. Which combinations of the clustering quality indexes and clustering properties yield the best accuracy of the labelling strategy?
3. What clustering quality index is best suited for labelling activity clusters, regarding accuracy and efficiency?

1.5 Summary of claimed contributions

Our aim for this thesis is to apply Dunn's index and C-index in a labelling strategy for clustering based IDSs. In this labelling strategy, we identify clustering properties and parameters that can be used to interpret the observed activities. This involves the identification of what combination of clustering quality indexes and clustering properties yields

[1] Gjøvik University College (Høgskolen i Gjøvik), http://www.hig.no

the best labelling performance. The performance of the labelling strategy is compared to the performance of the application of other clustering quality indexes and classical cardinality based labelling strategies.

The C-index requires, in its original form, clusters of similar size for proper evaluation of the clustering quality. As similar cluster sizes are a rare occurrence in clustering based IDSs, we propose a modified C-index appropriate for clustering quality evaluations regardless of the cluster sizes.

1.6 Research methodology

The methods, which have been used to solve the research questions, are of a quantitative nature. Our choice of methods involves literature study of related work, theoretical research concerning the use of cluster quality indexes for labelling, and experimental work to confirm the output from the theoretical research and measure the system performance.

According to Leedy et al. [38], who give a thorough survey of the research process, it is common to perform a literature study at an early stage in quantitative research projects. A review of previous work related to the research topic is necessary to gain the required background knowledge to solve our research questions. In the course of this review, it was important to identify issues such as problems, theories, results, answered questions, etc. presented by other researchers on related topics.

Based on the knowledge gained from the literature study, we engaged on the theoretical research. An important issue during this phase was to study the behaviour of the cluster quality indexes and the clustering properties in the presence of different kinds of observed activities. The prototype was an important tool during this phase of the project, as it was used to gather information about the behaviour of the clustering quality indexes and the clustering properties. Our findings from this theoretical research were then applied in the same prototype for testing and verification.

Most of the development of the prototype was performed in parallel with the literature study, because it was necessary to use the prototype in the theoretical research. This prototype is based on the base system developed by Petrovic et al. [50, 51, 52], which performs the actual clustering of the data. The clustering quality indexes, their respective labelling algorithms, and functions used in the theoretical research were implemented on top of this base system. There was no need for a formal methodology for this development phase. It was, however, an informal cycle between design, implementation and testing. Samples of the prototype source code are given in Appendix A.

In our experimental work, comprehensive simulations with the prototype were performed to evaluate and analyse the application of the two clustering quality indexes. As emphasized by Leedy et al. [38], it is important that the experimental work is performed under as controlled conditions as possible, e.g. to avoid biased results from third variables etc. To achieve this, automated scripts were used to perform the experiments; where only a single parameter of the experimental setup was adjusted at the time. All experiments were followed by thorough evaluations and analysis, some of which lead to new experiments and investigations.

1.7 Terms and definitions

- **Intrusion / attack -** an action that aims to violate the security policy for a given computer network or system. Heady et al. [24] define intrusion as "any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource".
- **Intrusion detection -** defined by Bace et al. [2] as "the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions".
- **Intrusion detection system (IDS) -** the automation of the intrusion detection process. Bace [3] lists three components an IDS must consist of in its simplest form:
  - An audit preprocessor responsible of providing the system with a stream of observed activities, e.g. network activities or system audit logs.
  - A detection engine responsible of interpreting and classifying the observed activities. In general, these detection engines are based on two types of detection models: misuse and anomaly detection.
  - A decision component responsible of responding to the outcome of the detection engine, e.g. generating alarms or reports to the human IDS analyst.
- **Misuse detection -** the detection engine uses signatures of known malicious activities to detect malicious activity. This is defined by Bishop [6] as determining whether "a sequence of instructions being executed is known to violate the site security policy".
- **Anomaly detection -** the detection engine flags behaviour that deviates from an expected norm. According to Bishop [6], "Anomaly detection analyzes a set of characteristics of the system and compares their behavior with a set of expected values"
- **True positive (TP) -** alert raised on malicious activity
- **False positive (FP) -** alert raised on normal activity
- **True negative (TN) -** no alert raised on normal activity
- **False negative (FN) -** no alert raised on malicious activity
- **Base rate -** the probability of attacks in the observed data.

$$P(I) = \frac{TP + FN}{N}$$

  where I - intrusion, N - observed data
- **Detection rate (True Positive Rate, TPR) -** the probability of an alert being raised when malicious activity is observed.

$$TPR = \frac{TP}{TP + FN} = P(A \mid I)$$

  where $(TP + FN)$ is the total number of intrusions, A - alert and I - intrusion

- **False alarm rate (False Positive Rate, FPR) -** the probability of an alert being raised when normal activity is observed.

$$FPR = \frac{FP}{FP + TN} = P(A \mid \neg I)$$

  where $(FP + TN)$ is the total number of normal activities, A - alert and I - intrusion

- **Distance metric -** a measure of the distance between two measurable objects. Let Ω be a data set. A function *d(x,y)* must satisfy the following conditions to be defined as a distance metric for all $x, y, z \in \Omega$ [33]:
  - $d(x, y) \geq 0$, the output distances must be positive.
  - $d(x, y) = d(y, x)$, independent of the order of operands.
  - $d(x, z) < d(x, y) + d(y, z)$, the triangle inequality holds.

- **A clustering -** the output of the clustering algorithm, which consists of two or more clusters, where a cluster is a group of instances from the observed data.

- **Cluster cardinality -** the number of elements a cluster consists of. Another term, which is often used to describe the cluster cardinality in literature, is "the cluster size".

- **Inter-cluster distance -** a measure of the distance between different clusters

- **Intra-cluster distance -** a measure of the distances between the instances within the same cluster. This measure therefore represents how scattered the instances within the cluster are in the cluster space.

- **Cluster centroid -** the computed average instance of all instances within a cluster. The cluster centroid represents the center of a cluster.

- **Massive attack -** occurs when the number of malicious connections within a range or time frame outnumbers the number of normal benign connections. This is the case during e.g. a SYN-flood Denial-of-Service attack against a resource.

- **Standalone attack -** malicious activities that are only related to a very few or none of the other activities in the observed range or time frame. An example is a buffer overflow exploit observed together with only normal benign activities.

1.8 Outline

The outline of this document closely resembles the different phases of our chosen research methods. Chapter 2 presents a review of topics related to this thesis. This involves topics such as the art of cluster analysis, clustering quality evaluation techniques, approaches to unsupervised intrusion detection and issues concerning performance testing of IDSs. The theoretical research is presented in Chapter 3. Here we present the assumptions behind our labelling strategy, the modified C-index and the labelling algorithms. Chapter 4 presents our experimental work and we discuss the results obtained from our experiments. The conclusions from our work are given in Chapter 5 and we discuss some issues that may be interesting for further research in Chapter 6.

2 Related work

It is important to study the state of the art of several related topics before we can begin answering the research questions. Important topics we need to investigate and gain in-depth knowledge about, are the concepts behind finding clusters in data, how to evaluate the quality of the obtained clustering and how clustering has been used for intrusion detection. There are also issues concerning the evaluation of IDS performance that must be studied in order to answer our research questions.

The following provides a short summary of our review of the state of the art. Efficiency is important for IDSs. For this reason, partitional clustering algorithms are often chosen for clustering based IDSs, despite the decreased clustering accuracy. Traditional labelling strategies in clustering based intrusion detection use different techniques and methods to measure the cardinality of the clusters to predict the nature of the observed activities. As this strategy has some limitations, other strategies are based on using cluster characteristics for labelling. One alternative strategy is to use clustering quality indexes to predict the presence of massive attacks, and then interpret the nature of the clusters with the use of clustering parameters, e.g. the cluster diameters. Besides the efficiency, is the labelling accuracy an important issue that must be taken into consideration when choosing which clustering quality index is best suited for use in an IDS. The detection rate and the false alarm rate are two of the most important measurements to measure this labelling accuracy. ROC curves are well suited to measure the trade-off between these two performance measurements.

2.1 Cluster analysis

Cluster analysis is defined by Kaufman and Rousseeuw as “the art of finding groups in data” [30]. The goal of clustering is to find groups in data, where the instances within the same group are very similar, while different groups are very distant from each other. A good comprehensive survey on the art of cluster analysis is given by Jain et al. [27]. At the top level, clustering algorithms can be divided into two categories; hierarchical and partitional.

To find clusters, hierarchical algorithms initially create many small clusters of similar data. The most similar clusters are then merged together, based on inter-cluster distances, until the desired cluster size is obtained. Variants of the single- and complete-linkage algorithms are the most popular hierarchical algorithms. The difference between single-linkage and complete-linkage algorithms is the way the inter-cluster distances are computed. Single-linkage algorithms use the shortest distance between any pair of instances in different clusters, while the maximum distance is used by complete-linkage algorithms. Hierarchical clustering algorithms produce very accurate clusterings, but the drawback is that they are often very computationally complex [8].

Partitional clustering algorithms obtain “a single partition of the data instead of a clustering structure” [27]. The advantage is that finding this partition is much less computationally complex than constructing the cluster structures. Partitional algorithms are therefore better suited for handling large data sets [5, 27, 63], which is the situation in

network based IDSs. A simple and very popular partitional clustering algorithm is the K-means algorithm [41]. A simple overview of the steps in the K-means algorithm is given by Guan et al. [20]:

1. **Initialization:** Randomly choose k instances from the data set and make them initial cluster centers of the clustering space.
2. **Assignment:** Assign each instance to its closest center.
3. **Updating:** Replace each center with the mean of its members.
4. **Iteration:** Repeat Steps 2 and 3 until there is no more updating.

Step 4 can also be stopped after a given number of passes. A drawback with the K-means algorithm is that the number of clusters must be known in advance. As a consequence, an additional algorithm should be used in most cases to find the optimal number of clusters. This is a problem associated with most partitional algorithms. Another problem that must be handled is that the result of the partitioning depends on the choice of initial cluster centers. It is therefore more difficult to achieve good clustering accuracy with partitional algorithms, than with hierarchical algorithms.

A distance measure must be defined before any clustering algorithm is applied, and the clustering accuracy depends on this measure. If the vectors to be clustered are of equal length and their coordinates take continuous values, the Minkowski distance metric is well suited. The Minkowski distance is defined by the following:

for two vectors $(x_1, x_1 ... x_n)$ and $(y_1, y_1 ... y_n)$

$$d(x, y) = \left(\sum_{i=1}^{n} \mid x_i - y_i \mid^p \right)^{\frac{1}{p}} \tag{2.1}$$

where n is the number of coordinates. For $p = 1$ the Minkowski metric is the Manhattan (city block) distance, and for $p = 2$ we get the well known Euclidean distance.

2.2 Clustering quality indexes

Clustering quality indexes have been used so far to tell us how well the data has been grouped into clusters, e.g. in algorithms used to find the optimal number of clusters for partitional clustering algorithms. There are several quality indexes available, for example the Davies-Bouldin index [7, 9], the Silhouette index [7, 54], Dunn's index [7, 11] and the C index [22, 26]. In this section we give a brief summary of how these indexes are computed, and which cluster parameters are used to evaluate the clustering quality.

The **Davies-Bouldin index** is defined by the following formula [7]:

$$DB = \frac{1}{M} \sum_{i=1}^{m} \max_{j=1,\dots,M; j \neq i} (D_{ij}) \, , where \, D_{ij} = \frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \tag{2.2}$$

Parameters used in the Davies-Bouldin index, to evaluate the quality of the clustering, are the total of the average intra-cluster distances and the average inter-cluster distances. In the formula, M is the number of clusters, σ is the average distance between the entities within the cluster and the cluster center c, and $d(..)$ is the distance between the clusters. The output from the Davies-Bouldin formula is a value between 0 and ∞. We have good clustering when a cluster is compact and the different clusters are distant from each other. In such cases the value of the Davies-Bouldin index is low.

The **Silhouette index**, uses the Silhouette width of each entity in a cluster to evaluate the clustering quality. This width is the confidence indicator of the entities' membership of a cluster. To compute this width, the minimum average distance to entities in other clusters is used, as well as the average distance to all other entities in the same cluster. To normalize this result, the maximum of the two distances is used. Computation of the Silhouette width yields a value between -1 and 1. A value near 1 indicates that the entity is within the correct cluster, a value near 0 means that the entity could also be a part of another cluster, while a value near -1 indicates that the entity has been placed in a wrong cluster. The Silhouette width of a cluster is the average sum of the silhouette widths of the entities within the cluster, and the Silhouette index of the entire clustering is the average sum of all cluster Silhouette widths.

The following formulas are used to compute the Silhouette index [52]:

$$s_i^j = \frac{b_i^j - a_i^j}{max\{a_i^j, b_i^j\}} \tag{2.3}$$

where a_i^j is the average distance between entity *i* and the other entities in the cluster, and b_i^j is the minimum average distance to entities in other clusters. We can then find the silhouette width for a cluster by:

$$S_j = \frac{1}{m_j} \sum_{i=1}^{m_j} s_i^j \tag{2.4}$$

where m is the number of entities. The global silhouette index of the whole clustering is then:

$$S = \frac{1}{c} \sum_{j=1}^{c} S_j \tag{2.5}$$

where c is the number of clusters.

Dunn's index only measures two parameters and the index is defined by the following formula [7]:

$$D = \min_{1 \le i \le c} \left(\min_{\substack{1 \le j \le c \\ j \ne i}} \left(\frac{d(c_i, c_j)}{\max\limits_{1 \le k \le c} \sigma_k} \right) \right) \tag{2.6}$$

The parameters used to evaluate the clustering with Dunn's index are the minimum inter-cluster distance and the maximum intra-cluster distance. In this formula, c is the number of clusters, σ is the average distance between the entities within the cluster and the cluster center c, and d(..) is the distance between the clusters. Because the index only measures two parameters, it may not yield stable results in some situations, e.g. when there are so-called outliers in the clustered data set. On the other hand, it can be computed quite fast, which is important for the efficiency of our labelling strategy. The two parameters used to compute this index are the minimum inter-cluster distance between clusters and the maximum intra-cluster distance. This is better illustrated by the simplified formula for Dunn's index, given by Gunter et al. [22]:

$$D = \frac{d_{min}}{d_{max}} \tag{2.7}$$

In this formula, d_{min} is the minimum inter-cluster distance and d_{max} in the maximum intra-cluster distance. Good clustering means that the inter-cluster distance is high and the intra-cluster distance is low. Higher values of the Dunn's index therefore indicate good clustering quality.

The **C-index** uses the distance between all pairs of entities within a single cluster, and the smallest and largest distances between pairs in the whole clustering to compute the clustering quality. This is defined by the following formula for the computation of the C-index [22]:

$$C = \frac{S - S_{min}}{S_{max} - S_{min}} \tag{2.8}$$

We must first choose a reference cluster to compute S in the formula from. S is then the total distance between all pairs within the reference cluster. Let l be the number of pairs within that cluster. Then, S_{min} is the sum of the l smallest distances between pairs in the whole clustering, while S_{max} is the sum of the l largest distances between pairs in the whole clustering. Good clustering quality requires small intra-cluster distances. The nominator is small in such situations, and low values of the C-index are therefore an indication of good clustering quality. The denominator is used to normalize the formula, and the computation of the C index will produce an index value between 0 and 1.

The C-index is, in its original form, only appropriate when the clusters have similar cardinality. When the clusters have similar cardinality it is arbitrary what cluster we choose as reference cluster to compute S from. This is, however, not the case when we evaluate clusterings with different cluster cardinalities. Determining what cluster to compute S from is therefore an issue that must be handled when C-index is used in situations with unequal cluster cardinalities, e.g. in our labelling strategy. Another issue that must be answered in this thesis, is that the high time complexity might make it unsuitable for intrusion detection in real time.

2.3 Unsupervised anomaly intrusion detection

To detect intrusion attempts, anomaly based detection systems define a baseline of normal activity, and observed activity deviating from this baseline is assumed malicious. There are in general two approaches to create this baseline, supervised learning and unsupervised learning. The traditional approach has been supervised learning with the use of cleansed training data, where known attacks have been removed or labelled [53, 63]. Obtaining this training data is very difficult and costly, and there are several problems associated with obtaining a good training data set. One problem is that it is not possible to be certain that all the attacks in the training data set have been identified and/or removed, which may lead to inaccurate measurement of detection success. Other problems involve that "(...) it may be impossible to unambiguously assign a label to a data instance" [36], and it is very difficult to create a training data set that fully reflects the activity of the monitored system.

Unsupervised intrusion detection techniques are used to overcome the problems with supervised learning. Because systems using unsupervised techniques do not rely on knowledge learnt from training data, they are better suited than supervised techniques for detecting new unknown attacks. According to Zhong et al. [63], "(...) the purpose of unsupervised intrusion detection is to discover new attacks in a new dataset". Unsupervised techniques are also, for the same reason, better suited for tracking changes in user

behaviour and activity over time.

Laskov et al. analyse both approaches in [36], and conclude that supervised techniques outperform unsupervised techniques if the observed activity only contains known attacks. However, this is not the case in real world environments. When unknown attacks are present in the observed data set, Laskov et al. [36] show that the performance of supervised techniques deteriorates while the performance of unsupervised techniques remains unaffected.

Clustering is unsupervised classification of data [27], and many research projects have focused on clustering techniques for intrusion detection in the recent years. In the remainder of this section, we describe important research on intrusion detection by means of clustering, with special attention on the labelling strategies.

In [53], Portnoy et al. present a clustering based approach for intrusion detection to solve the problems with supervised learning. Based on their proposed approach, they introduce the term *unsupervised anomaly detection*; an IDS that operates on unlabeled data. An earlier approach to unsupervised anomaly detection, by Eskin et al. [14], used probabilistic models to indicate abnormal activity. In this new approach, Portnoy et al. [53] state that "(...) we drop the requirement of a probabilistic model and instead use inter-point distances to motivate our algorithm". This approach, which we describe in detail below, has been used as the basis for many clustering based IDSs.

The system proposed by Portnoy et al. [53] extracts and normalizes features from the activity data, and uses these features to cluster the activities. These features must be normalized because the extracted features have different scales and types. Some features may then dominate the output from the distance metric, which leads to incorrect clustering. To normalize the data, the value of each feature in the activity data is set to the number of standard deviations it differs from the average value of that feature.

The main idea behind their approach is to use distances between activity features to classify the activities. A solid distance metric is therefore necessary to measure the distances between the entities to be clustered. Portnoy et al. [53] tested several distance metrics, and different metrics were tried for different features. Based on these tests they chose to use the well known Euclidean distance metric. Weighting the different features only showed slightly increased performance, so equally weighted features were used.

A simple variant of the single linkage algorithm [55] is used for clustering, despite that hierarchical clustering techniques are generally not very efficient. The clustering is performed by a single pass over the data set, beginning with an empty set of clusters. The distances between an instance of observed activity and the centers of existing clusters are measured. Then this instance is placed into the cluster to which it has the smallest distance, if this distance is within a predefined cluster width. If the distance is larger than the cluster width, a new cluster with the observed activity as its center is created.

Portnoy et al [53] base their approach on two assumptions; the number of normal activities vastly outnumbers the number of malicious activities, and benign and malicious activities are qualitatively different. These two assumptions are noticeable when we look at how the system labels the nature of the obtained clusters. In [53], it is assumed that if the cluster width is properly set, then different kinds of activity are placed in different clusters. Then, from the assumption that normal activities vastly outnumber malicious activity, they use the cardinality of the clusters for labelling. To achieve this they first sort the clusters by their cardinality, and then label N percentage of the clusters with largest

cardinality as normal.

To detect malicious activity, the system first learns from an unlabelled training set and creates clusters based on this training set. These clusters are then sorted by their cardinality and labelled. When new observed activity is observed, the activity is measured against the centers of the clusters obtained from the training set and given the label associated with the cluster to which it has the smallest distance.

Portnoy et al. use in [53] a hierarchical single linkage algorithm for clustering, which has some performance drawbacks regarding its efficiency. Partitioning based clustering techniques, like the k-means algorithm, may be better suited for intrusion detection because of its efficiency. Guan et al. [20, 21] propose Y-means, a new clustering algorithm for use in a clustering based IDS. This algorithm is an improvement of the K-means algorithm, and overcomes two of K-means' shortcomings; number of cluster dependency and degeneracy. The authors also aimed to develop an algorithm where there is no need to manually set a fixed cluster width, which must be done in the system proposed in [53].

The Y-means algorithm begins by clustering the data into k clusters. The next step is to search for empty clusters. If there are empty clusters, new clusters are created to replace the empty clusters and the instances are reassigned to the new clusters. This process is repeated until there are no empty clusters. It then uses outliers in the clustering to create new clusters. When no new clusters can be found, the next step is to merge overlapping clusters together until the optimal number of clusters is obtained. The last step of the Y-means algorithm is to label the obtained clusters. Like in [53], the assumption behind the labelling strategy is that normal activity vastly outnumbers abnormal activity and clusters with cardinality below a given threshold is labelled as abnormal.

A geometric framework for mapping network traffic and system call traces into a feature space, is presented by Eskin et al. [15]. Based on this framework the authors present three algorithms for detecting outliers in the obtained feature space, as the authors expect that malicious activities will appear as outliers. The algorithms presented are; an algorithm very similar to the algorithm presented in [53], a K-nearest neighbours algorithm and a Support Vector Machine. The assumption that normal behaviour vastly outnumbers malicious activity is used to label the smallest clusters as malicious, since these clusters will consist of the outliers.

Oldmeadow et al. [47] focus on the importance of coping with changing traffic conditions in their clustering based approach to intrusion detection. Their aim for the proposed approach, which is based on the work presented in [15], is to develop a time varying clustering algorithm where the clusters are updated during real-time operation. This makes the approach capable of handling changes in traffic conditions over time, and independent of the activity seen in the training set. To prevent an update from causing too large impact on a small cluster and small impact on a large cluster, an influence factor is used to regulate the effect an update should have on the affected cluster.

Another topic covered by the authors of [47] is feature weighting, and they demonstrate a performance enhancement in accuracy with the use of feature weighing. This contradicts the authors of [53], who did not find a significant advantage of using feature weighting in their algorithm. The authors conclude that "Although our feature weighting was based on a manual analysis (...), our results provide a clear motivation for further research into automated approaches for feature weighting" [47].

The approaches described above use training data sets to obtain clusters, and then use

these clusters to classify new observed activity. Zhong et al. [63] propose an approach that does not use any historical training data, which only use the observed activities to obtain the clusters. The authors investigate four clustering techniques for use in intrusion detection. Their results show that clustering is suitable for intrusion detection, and that the performance of clustering based IDSs is comparable with traditional approaches. They also show that combining traditional techniques with clustering techniques can increase the performance of an IDS. The results also confirm the authors' hypothesis that clustering is better suited for detecting previously unknown attacks than traditional approaches.

For labelling, Zhong et al. [63] propose a heuristic labelling algorithm, which is based on an assumption that resembles the assumption used by Portnoy et al. [53]. However, they do not assume "(...) the strict hypothetical requirement that the percentage of attacks has to be less than a certain threshold" [63]. Instead they use inter- and intra-cluster distances for labelling. The largest cluster is assumed to be normal, and then other clusters are sorted by the distance from that clusters' centroid to the centroid of the largest cluster. The entities within each cluster are sorted in the same way, by their distance to the centroid of the largest cluster. A given or estimated number of those entities, with the shortest distance to the centroid, are then labelled normal and the remaining entities are labelled as attacks.

Zhang et al. present an agent based distributed IDS based on clustering in [62]. They propose a method that performs clustering twice to detect malicious activity. Agent IDSs are deployed on the monitored system or network. These agents use a clustering technique similar to the one presented in [53], but differs slightly in how to measure distances and how to choose anomalies. The agents send potential malicious activity to a central IDS where the reported activities are scrutinized closer. The central IDS uses a variant of the single linkage algorithm to create clusters from the activity reported by the agents. This algorithm is very accurate and it is efficient enough for this purpose. Efficiency is not the most important issue in this setting, because the number of reported activities is assumed to be rather low in the central IDS. From the obtained clusters, the algorithm creates an attack cluster based on the cardinality of these clusters and the inter-cluster distances between them. The activities in the attack cluster are considered as true attacks and reported to the human IDS analyst.

The systems described so far base their labelling strategy on the assumption that normal activity vastly outnumbers abnormal activity. This assumption holds for most situations, but there are limitations with basing a labelling strategy solely on this assumption. One limitation is that benign traffic rarely seen on the network will produce small clusters that would be labelled as malicious. Another limitation is that the assumption is wrong in the presence of a massive Denial-of-Service attack [15, 53]. In [39], where Leung et al. present a grid and density based clustering algorithm for intrusion detection, the authors point out that this assumption may cause the performance of the detection algorithms to deteriorate in the presence of a massive Denial-of-Service attack. This limitation is also emphasised by Zhong et al. [63] who state that "(...) we have seen in our experiments many relatively large attack clusters as well as small normal clusters".

As a solution to the limitations described above, Petrovic et al. [50, 52] present a new strategy for labelling obtained clusters. Instead of using the cardinality of the clusters, other characteristics of the clusters are used to predict the nature of the cluster contents.

The assumption behind the new labelling strategy is that the attack clusters will be very compact in the presence of a massive attack, because the attack vectors in such attacks are very similar, if not identical. In most cases, where a massive attack is not present, the assumption is that the attack clusters will be less compact than normal clusters. The reason for this is that different standalone attacks are very different from each other.

Based on these assumptions, the use of clustering evaluation techniques is proposed for proper labelling of the clusters. Good clustering may be used as an indication of the presence of a massive attack and clustering evaluation techniques, like clustering quality indexes, are well suited for controlling the quality of the clustering. Clustering quality indexes are therefore used to investigate the clustering for the existence of a massive attack. The result from the clustering quality index can then be used in combination with cluster diameters to predict the nature of the obtained clusters.

In [50, 52] it is demonstrated that the Davies-Bouldin index and the Silhouette index may be used for proper labelling of obtained activity clusters. A comparison of the two systems is given in [51]. The results show that the Silhouette index yields more accurate labelling of the clusters than the Davies-Bouldin index. However, the time complexity of the Silhouette index is much higher than the complexity of the Davies-Bouldin index. This makes the Davies-Bouldin index better suited for labelling clusters in a clustering based IDS.

Another approach for labelling is presented by Gomez et al. in [17, 18, 19], where a set of simple fuzzy rules (fuzzy classifiers) is used to detect abnormal activity. These fuzzy rules are based on fuzzy logic, and are according to the authors well suited for detecting abnormalities. The idea behind this approach is; “As the difference between the normal and the abnormal activities are (...) rather fuzzy, fuzzy logic can reduce the false signal rate” [17]. When using fuzzy classifiers, observed activity does not have to belong entirely to a group, but can partially belong to several groups. This concept is well suited for intrusion detection, since it is very difficult to define boundaries between normal and abnormal activity.

Most proposed clustering based approaches use the Euclidean distance metric for measuring similarity when activity are placed into clusters. Wang et al. [59] propose a new clustering algorithm, FCC, for intrusion detection. This clustering algorithm uses fuzzy connectedness for measuring similarity. The fuzzy connectedness metric uses both the Euclidean distance and statistical properties of the clusters, and the results show that this metric is very robust. Since the algorithm can be used to find clusters regardless of the shape of the clusters, it handles a limitation of the K-means algorithm, which only finds clusters with spherical shape. However, a drawback of the proposed clustering algorithm is that it needs some prior knowledge, from e.g. labelled data, to initiate the clustering process.

Clustering based approaches to intrusion detection are not limited to monitor network traffic. Last et al. [37] present clustering as a method for content based anomaly detection on the web, where clustering is used to find abnormal content in web pages. Oh et al. [46] present a clustering method for detection of system abnormality. In this approach, clustering is not used in its traditional way to group similar data into clusters. According to the authors, clustering “(...) can be employed to extract the common knowledge i.e. properties of similar data objects commonly appearing in a set of transactions.” [46]. They therefore propose a clustering method for modelling profiles of normal user

activity. From these profiles, internal and external differences, i.e. intra- and inter-cluster distances, are used to detect abnormal behaviour in user transactions.

In [45], this method is improved by Oh et al. to continuously model a data stream. This makes the system capable of tracking changes in user behaviour over time, without storing large amounts of historical data. To detect abnormal behaviour, two abnormality levels, green and red, are defined to determine the nature of the observed activity. Each feature of a profile is represented by a cluster center and the standard deviation. Features with too long distance to the profile are considered suspicious. If the number of suspicious features from the observed activity is above a certain threshold, the activity is considered abnormal and labelled as red.

Clustering based intrusion detection techniques are generally applied for anomaly detection. However, an example where clustering is used for signature recognition is proposed by Ye et al. [61].

2.4 Testing intrusion detection systems

Mell et al. [44] provide a comprehensive study of issues concerning the testing of IDSs. Intrusion detection systems have become a very important part of any defence-in-depth perimeter security setup, but "(...) no comprehensive and scientifically rigorous methodology to test the effectiveness of these systems" [44] are currently available.

Several measurements regarding the performance of an IDSs are discussed in [44]. Examples of these measurements are the coverage, the probability of false alarms, the probability of detection, the ability to detect unknown attacks and capacity verification of network IDSs. Two very important measurements that a metric measuring the performance of an IDS must take into consideration, are the probability of false alarms, i.e. the false positive rate (FPR), and the probability of detection, i.e. the true positive rate (TPR).

A ROC (Receiver Operating Characteristic) curve [12] is a performance measure that considers both the FPR and the TPR. ROC curves have been widely used for analysing cost/benefit relationships, e.g. in biometric systems. By using ROC curves to analyse IDS performance, the relationship between FPR and TPR is measured and plotted in a graph. According to Mell et al. [44], these performance measures are two of the most important IDS characteristics, and ROC curves have therefore become an important measurement in the IDS testing community. ROC curves for measuring IDS performance have been met with some criticism, since a measure of the relationship between FPR and TPR would result in a single point rather than a curve. However, by varying a parameter in the IDS, e.g. a threshold, we can obtain a curve. This curve may then be used to find the IDS configuration that yields the optimal performance. Another problem concerning the ROC curves is that they are senseless for small values of the probability of false alarms.

2.4.1 The KDD Cup 99 data set

To test the performance of the prototype we need a test data set. A problem with obtaining such data sets is that very few are publicly available. One publicly available labelled test data set, widely used in academic communities, is the KDD Cup '99 data set [13, 40, 58]. The KDD Cup '99 has seen some criticism in the literature, see e.g. [43]. This criticism is mainly based on the fact that the KDD Cup '99 is an artificial data set that originates from simulated traffic from a military environment. The data set is also

quite old, and contains some obsolete attacks. However, it is still widely used for IDS benchmarking and all network based systems described in Section 2.3 use the KDD Cup '99 in their performance evaluations. And most important, the KDD Cup '99 test data set provides us with sequences of massive Denial-of-Service attacks, which our proposed labelling strategy should be capable of detecting.

KDD Cup '99 is a result of 9 weeks of raw tcpdump data from a simulation of a military air-base data network. The data set consists of nearly 5 million data instances, i.e. vectors containing extracted features from connection records. These features, 41 in total, were extracted to describe the properties of the connections in the data set. Kayacik et al. [31] provide a comprehensive study on these features and what features are usable for intrusion detection. These features can be grouped into three categories:

- Basic features from individual TCP connections, sorted by destination host, e.g. protocol and flags set.
- Features extracted from the content of a connection. These features look for suspicious content, e.g. failed login attempts.
- Traffic based features computed over a two second time window or the last 100 connections, e.g. the number of connections to the "same host" or "same service".

KDD Cup '99 is divided into 2 parts, one intended for the training phase and one intended for performance testing. The training data set consists of 24 attacks, while 12 additional attacks are included in the test data set. These attacks can be placed in four categories:

- DoS: denial of service, e.g. SYN flood.
- R2L: unauthorized access from a remote machine, e.g. password guessing.
- U2R: unauthorized access to superuser or root functions, e.g. various "buffer overflow" attacks.
- Probing: surveillance and other probing for vulnerabilities, e.g. port scanning.

A comprehensive overview of the attacks included in the data set is given by Kendall [32]. The data set we have chosen to use for our performance tests is a reduced version (10 %) of the KDD Cup '99 test data set. This data set consists of 490000 connection records and typical attack records in this data set are:

- Neptune attack
  The Neptune attack is a SYN-flood attack [4], which is a Denial-of-Service attack that exploits a weakness of the TCP protocol. The first step of the three-way handshake used to set up a TCP connection, is to send a packet with the SYN flag set [57]. During a Neptune attack, massive amounts of such connection requests are sent to the targeted machine. Each of these requests creates a half-open TCP connection on the targeted machine, and information about this half-open connection is stored

in memory until a connection timeout occurs. The attacker's aim is to exhaust the memory available to store this information. If the attacker succeeds, the result is that the system may crash or otherwise become unavailable for legitimate users.

- Smurf attack
  The Smurf attack utilizes the ICMP protocol and the internet infrastructure to cause a Denial-of-Service attack [56]. ICMP echo request packets are sent to the broadcast address of different subnets, with the spoofed source address of the targeted machine/network. By sending ICMP echo requests to the broadcast address, the requests are amplified with the number of active host on the subnets. Each host on these subnets will then issue an ICMP echo reply to the targeted machine. In worst case, this means that a single ICMP echo request will cause that 255 ICMP echo replies are sent to the targeted machine/network. If the attacker sends a stream of ICMP echo request to various subnets, the amount of replies may exhaust the resources of the targeted machine/network and render it unavailable for legitimate users.
- Teardrop
  The teardrop attack targets a flaw in old implementations of the TCP/IP stack. This flaw resides in the IP fragmentation re-assembly code, which does not handle overlapping IP fragments properly and causes the targeted host to crash or reboot [32]. Operating systems utilizing older implementations of the TCP/IP stack, often GNU/Linux distributions, are therefore vulnerable for this Denial-of-Service attack.
- Buffer overflow exloit
  A common security problem is services or commands with poor boundary and syntax checking of the input buffer. Buffer overflow attacks utilize this problem to achieve elevated user privileges [16]. By overflowing the input buffer with specially crafted data, the attacker tries to overwrite the memory pointers on the system stack. This technique may then be used to execute malicious code, i.e. run commands with elevated privileges that open a remote shell connection.
- Root kit
  A Root kit is a collection of programs installed on a compromised system by the attacker, in order to maintain access to the system and hide the presence of malicious activities. Root kits typically consist of sniffers and other advanced tools for attacking computer systems [35]. Altered versions of common system commands and services are used to hide malicious activities and set up backdoors to provide system access to the attacker. Once a Root kit has been installed on the system, the attacker has complete control over the system without the knowledge of the system owner.
- Ipsweep
  Ipsweep is a surveillance probe that sweeps through host addresses and ports on the targeted network. The aim is to identify open ports on hosts that can be used to attack the network [29].

3 Application of Dunn's index and C-index for labelling clusters

In this chapter, we present our approach for labelling activity clusters in an anomaly based IDS that use clustering for classification. This labelling strategy uses a combination of clustering quality evaluation techniques and clustering properties to handle some limitations associated with classical cardinality based labelling strategies. Here we describe the theory and assumptions behind our labelling strategy. We also present the application of two well known clustering quality indexes, the Dunn's index and the C-index, in this strategy for proper labelling of activity clusters.

We limit ourselves to only study the two-cluster case in our work. In this case, the clustering of the activity data only consists of two clusters, where one cluster corresponds to normal activities and one cluster corresponds to the malicious activities. The reason is that the aim of a labelling strategy is, regardless of the number of clusters used to classify the observed activities, to find two "superclusters", where one cluster consist of normal activities and the other cluster consists of malicious activities.

Classical labelling strategies based on cluster cardinality have some limitations, as emphasized in Section 2.3. A major limitation is the limited capability of detecting massive attacks, e.g. Denial-of-Service attacks. The main idea behind our labelling approach is that the physical properties of the clustering are better suited to determine the nature of the clusters than the cluster cardinalities. These properties depend on the nature of the observed activity and may therefore be used to interpret the nature of the clusters. The aim of this thesis is therefore to handle the limitation of detecting massive attacks with the use of a labelling strategy that takes these clustering properties into consideration.

It is important to identify the clustering properties that can be used to interpret the nature of the clusters and how these properties are affected by different types of activity. The following examples, from the KDD Cup '99 data set, describe how different types of activity can be distinguished from each other. These examples also demonstrate the effect different activities have on the physical properties of the clustering and how this information can be utilized for intrusion detection.

The goal of clustering is to group data into clusters, where the instances within the cluster are very similar and different clusters are distant from each other. An underlying assumption behind the use of clustering to classify activity, is that normal and malicious activities are placed into separate clusters as a result of their dissimilarities. This dissimilarity is illustrated in Table 1, which shows the features of a benign HTTP connection and a buffer overflow exploit over the telnet service. Many of these features have very different values, and a distance metric will therefore yield a large distance between these two records. The consequence is that we can expect that normal and malicious activities are clustered into separate clusters.

We can also expect that similar activities are placed into the same cluster, because their features are quite similar to each other and distant from other activities. This is illustrated in Table 2, which describes the features of three normal connections. If we compare the records in Table 1 with the records in Table 2, we see that the normal

Record number / KDD Cup feature	1	82523
Activity type	Normal HTTP	Buffer overflow
duration	0	158
service	80	23
flag	10	10
src_bytes	239	1567
dst_bytes	486	3095
hot	0	3
logged_in	1	1
num_compromised	0	4
root_shell	0	1
num_file_creations	0	1
num_shells	0	2
count	8	1
srv_count	8	1
same_srv_rate	100	100
dst_host_count	19	2
dst_host_srv_count	19	2
dst_host_same_srv_rate	100	100

Table 1: The difference between a normal vector and an attack vector when no massive attack is present, from the 10 % reduced KDD Cup '99 data set. The rest of the 41 features are equal to 0

activities are rather similar to each other, compared to the malicious activity in Table 1. We see that the Euclidean distance, which we consider appropriate for use in this setting, between the features of the normal connections is rather small. The distance is, on the other hand, large between normal and malicious records. Table 2 also show that there are some differences between the normal activities. Clusters that consist of normal activities will therefore not be very compact. This is also expected, as there is a very wide range of different normal activities.

Record number / KDD Cup feature	1	205	206
Activity type	Normal HTTP	Normal HTTP	Normal SMTP
service	80	80	25
flag	10	10	10
src_bytes	239	195	3366
dst_bytes	486	24572	329
logged_in	1	1	1
count	8	2	1
src_count	8	2	1
same_srv_rate	100	100	100
dst_host_count	19	255	8
dst_host_srv_count	19	255	7
dst_host_same_srv_rate	100	100	100

Table 2: The difference between three normal vectors, from the 10 % reduced KDD Cup '99 data set. The rest of the 41 features are equal to 0

Malicious activities that are part of a massive attack and standalone malicious activities have very different effect on the physical properties of a clustering. Table 3 describes

two attack vectors that are part of a massive Neptune attack, and shows that the two vectors are very similar. Another example of a massive attack is the Smurf attack, where the difference between the attack vectors may be as little as the source IP-addresses. Clusters containing activity vectors from a massive attack will consequently be very compact, because of the very short distance between the activity vectors.

Because the malicious activities are very similar, if not identical, we may end up with situations where all the observed activities are clustered into the same cluster, when a massive attack is present. In these situations, we get one very compact cluster that consists of the massive attack, while the normal cluster is empty. We may also in other situations observe clusterings where the malicious cluster is empty, because we only observe normal activities. This normal cluster will, on the other hand, not be very compact.

Record number / KDD Cup feature	71143	71144
Activity type	Neptune attack	Neptune attack
service	49152	49152
flag	6	6
count	188	198
srv_count	1	18
serror_rate	100	100
srv_serror_rate	100	100
dst_host_count	255	255
dst_host_srv_count	1	18
dst_host_serror_rate	100	100
dst_host_srv_serror_rate	100	100

Table 3: The difference between two attack vectors within a massive attack, from the 10 % reduced KDD Cup '99 data set. The rest of the 41 features are equal to 0

Standalone attack vectors are on the other hand very different from each other. Table 4 consists of the features from two standalone malicious activities, a Root kit and a buffer overflow exploit over the telnet service. These features show that there are many differences between these two activities, which will lead to very scattered clusters if these activities are clustered together. We expect that standalone malicious activities and normal activities are clustered into separate clusters. The reason is that these standalone attacks are very distant from the many normal activities, as can be seen by comparing the records in Table 2 and 4. In the two-cluster case, this means that the standalone malicious activities will be clustered together into a scattered cluster, while the other cluster consists of the normal activities.

We can now draw some generalizations about the effect different types of activity have on the clustering properties, which are the cornerstones of our labelling strategy. Malicious activities that correspond to a massive attack form very compact clusters, much more compact than clusters that consist of normal activity. The situation is the opposite when no massive attack is present in the observed activity. Then, the differences between the standalone attack(s) will result in clusters that are more scattered than clusters that consist of normal activity. The compactness of the clusters can therefore be used to determine the nature of the clusters, assumed that we can determine whether or not a massive attack is present.

A hallmark of good clustering is small intra-cluster distances and large inter-cluster

Record number / KDD Cup feature	141511	82523
Activity type	Rootkit	Buffer overflow
duration	60	158
service	23	23
flag	10	10
src_bytes	90	1567
dst_bytes	233	3095
hot	0	3
logged_in	0	1
num_compromised	0	4
root_shell	0	1
num_file_creations	0	1
count	1	1
srv_count	1	1
same_srv_rate	100	100
dst_host_count	255	2
dst_host_srv_count	2	2
dst_host_same_srv_rate	0	100
dst_host_same_src_port_rate	100	0
dst_host_rerror_rate	100	0

Table 4: The difference between two standalone attack vectors when no massive attack is present, from the 10 % reduced KDD Cup '99 data set. The rest of the 41 features are equal to 0

distances. This is, as we have seen above, the situation when a massive attack is present in the observed activities. The similar activities within a massive attack will form very compact clusters, which are distant from clusters of normal activity. Clustering quality indexes take both intra-cluster and inter-cluster distances into consideration, and are well suited to measure the clustering quality. We can therefore measure the quality of the clustering to determine whether or not a massive attack is present. Proper labelling of the clusters can then be achieved by measuring the clustering quality and the compactness of the clusters.

3.1 Dunn's index

Dunn's index, described in Section 2.2, uses intra-cluster and inter-cluster distances to evaluate the quality of the clustering. Several intra-cluster and inter-cluster distances are appropriate for use with the Dunn's index, see e.g. Bolshakova et al. [7]. We have chosen to use the centroid diameter and the centroid linkage measures as intra- and inter-cluster distances for compatibility with the clustering algorithm. We use K-means to cluster the observed activity and these centroid based measures are compatible with the K-means algorithm, which computes the cluster centroids at each iteration.

Let $X_\tau = (X_1, \ldots, X_N)$ be the data set from which we obtain K clusters $C = (C_1, \ldots, C_K)$. The inter-cluster distance, i.e. the centroid linkage, between clusters is then computed with the following formula [7]:

$$\delta(C_i, C_j) = d(s_{c_i}, s_{c_j}) \tag{3.1}$$

where s_{c_i} and s_{c_j} denotes the cluster centroids and d() is the Euclidean distance between the cluster centroids. The cluster centroids are computed with the following

formulas:

$$s_{c_i} = \frac{1}{|C_i|} \sum_{X_k \in C_i} X_k \quad \text{and} \quad s_{c_j} = \frac{1}{|C_j|} \sum_{X_k \in C_j} X_k \tag{3.2}$$

The intra-cluster distance, i.e. the centroid diameter, of a cluster is computed with the following formula:

$$\Delta(C_i) = 2 \left(\frac{\sum_{X_k \in C_i} d(X_k, s_{C_i})}{|C_i|} \right) \tag{3.3}$$

where the average cluster radius is used to find the average centroid diameter. The centroid is computed with the same formula as for the inter-cluster distance:

$$s_{C_i} = \frac{1}{|C_i|} \sum_{X_k \in C_i} X_k \tag{3.4}$$

3.1.1 Time complexity

The computation of the Dunn's index involves the computation of the cluster centroids, and the computation of the inter- and intra-cluster distances. This means that the time complexity of the Dunn's index is linear in the number of observed activities, n. In the two-cluster case we study, the number of computations used to find the distances considered in the formula is $2n + 2$.

3.2 C-index

The C-index is quite different from Dunn's index in its approach to evaluate the clustering quality. While the latter is computed using inter- and intra-distances from e.g. the centroids of each cluster, the former is computed by measuring the distance between all elements in the clustering. We can expect very good evaluation accuracy when using the C-index, because it considers many distances, but this comes at the cost of high computation time. The consequence is that the time complexity of the C-index is much higher than e.g. the Dunn' index, which may be a major disadvantage when it is used for intrusion detection.

A problem that must be handled before the C-index can be used for intrusion detection is to determine how to evaluate a clustering that consists of clusters of different cardinalities. The C-index is only appropriate when the clusters have similar cardinality, which is expected to be rare when observing network activity. When clusters have similar cardinalities, the choice of which cluster to evaluate the clustering from does not have any major effect on the result from the C-index. However, when the clusters have unequal cardinalities, the result from the C-index evaluation depends on what cluster we choose to evaluate the clustering from. The C-index is defined by the formula (2.8) presented in Section 2.2. When we use the term "*evaluate the clustering from a cluster*", we refer to the cluster that is used to compute S in that formula.

S can be interpreted as the total sum of intra-cluster distances within the cluster, which indicates that the value of S depends on the cluster size. S_{max} and S_{min} also depend on the size of the cluster we choose to evaluate from, because the number of distances used to compute these variables, l, is directly related to the cluster size. We can therefore expect the result of the clustering quality evaluation to depend heavily on

the choice of cluster to evaluate from, when the clusters in the clustering have arbitrary sizes. Based on this, and because we study a two cluster case, we choose to compute the C-index from both clusters. We refer to these indexes as *partial indexes* in this modification of the C-index. These partial indexes can then be used separately or combined in our labelling algorithm. This may also provide additional information about the clustering, which may be taken into consideration by the labelling algorithm and improve the labelling accuracy.

3.2.1 Modified C-index

It is preferable to have a single value that describes the clustering quality when the C-index is used in our labelling algorithm. Then, the number of thresholds that needs configuration and maintenance is kept at a minimum, and the system remains manageable. We therefore need to either combine the partial indexes or determine what partial index to use. Two alternatives for this modified C-index are *C-mean* and *C-small*.

C-mean is, as the name indicates, the mean of the partial indexes. An advantage is that all distances are at least considered once and we can be sure that both intra-cluster and inter-cluster distances are considered in the evaluation. A possible drawback is that the average value of the two partial indexes may mask the existence of irregular values of the partial indexes, which can be useful information for the labelling algorithm.

C-small is the partial index when evaluating the clustering from the smallest cluster, i.e. the cluster with fewest elements. When the clustering quality is evaluated from the smallest cluster, we can expect that S_{max} mainly consists of inter-cluster distances, and that S_{min} mainly consists of intra-cluster distances. Then both the intra-cluster and inter-cluster distances are more equally considered in the evaluation formula and we should achieve good evaluation of the clustering quality. A drawback is that we may get inaccurate evaluations when the smallest cluster only consists of very few elements, since only a few distances are considered in those situations. Another drawback is that so-called outliers in the clustering may cause unstable results, which we describe closer in Section 3.2.2.

3.2.2 Outliers in the clustering

The presence of so-called outliers in the clustering may cause unstable C-small evaluations. Outliers are points very distant from the majority of the other points in the clustering, and often cause bad clustering quality. Because malicious activities are often very distant from normal activity, they may appear as outliers in the clustering when no massive attack is present. In these situations, it may be insufficient to only measure the quality of the clustering from a single cluster, because the outliers may have too much influence on the partial index. The instability of the partial indexes is a result of the use of the cluster cardinality to find the number of distances necessary to compute S_{min} and S_{max}. Both intra- and inter-cluster distances must be considered for a proper evaluation of the clustering quality, and the C-small index may perform incorrectly in situations where only a few distances are used to evaluate the clustering quality.

Outliers often form small clusters with large inter-cluster distances to the other cluster. The result is that only a few distances are used to compute S_{min} and S_{max}, when we use C-small. It is then a risk that S_{max} only consists of inter-cluster distances and that S_{min} only consists of intra-cluster distances from the cluster with normal activity. This may result in a very large S_{max} and a small S_{min}. If the observed outliers are quite similar,

e.g. two similar buffer overflow exploits, then the intra-cluster distances are rather small and we get a small S value. The result from the C-small index may then be incorrectly small and indicate good clustering quality.

We could use the partial index from the largest cluster when we have indications of outliers in the clustering. Then both intra-cluster and inter-cluster distances are included in S_{min} and S_{max}, and we would achieve improved accuracy of the clustering quality evaluation. However, it is not a good solution to evaluate the clustering quality from the largest cluster when we have several distant outliers in the clustering, e.g. a buffer overflow exploit and a Root kit exploit. These outliers are placed in the same cluster in a two cluster clustering, because of the large distances to the other elements in the clustering. This results in a clustering with large intra-cluster and inter-cluster distances. The consequence is that these distances may lead to a very high value of S_{max}, while S remains rather small since the largest cluster consists of normal activities. Evaluating the clustering quality from the largest cluster may then lead to an incorrectly small partial index, which indicates good clustering quality. In this case, we achieve the correct quality evaluation by evaluating the clustering from the small cluster, because the large intra-cluster distances equalize the high S_{max} value.

As described above, we cannot be certain of what cluster to evaluate the clustering quality from when outliers are present. However, the partial indexes are in opposition to each other when outliers are present. This difference between the partial indexes can be used to describe the clustering quality in these situations. We therefore assume that the clustering quality is bad when the difference between the partial indexes is above a defined threshold.

The following two examples from the 10 % reduced KDD Cup '99 data set illustrates how outliers cause C-small to indicate incorrect clustering quality. We use the centroid diameter and the centroid linkage in the examples to provide a single measure that illustrates the intra-cluster and inter-cluster distances in the clustering.

Records	A	Ci 1	Ci 2	CD 1	CD 2	CL	Card 1	Card 2
452001-453000	88	0.03	0.92	84718	23172	1519435	5	995
41001-42000	5	0.66	0.00001	2252490	7281	78179	4	996

Table 5: Partial indexes and clustering parameters from two clusterings from the KDD Cup '99 data set.

A: Number of attacks in the observed records
Ci 1 & 2: Partial index when evaluating the clustering quality from cluster 1 & 2
CD 1 & 2: Centroid diameter of cluster 1 & 2
CL: Centroid linkage
Card 1 & 2: Cardinality of cluster 1 & 2

The clustering of the connection records from 452001 to 453000 is illustrated in Figure 1, where red points represent malicious activities and green points represent normal activities. This clustering consists of 88 malicious connections, where 82 connections originate from Neptune attacks, four connections are Buffer overflow exploits and two connections are Root kit exploits. In this case the outliers also cause the clustering algo-

rithm to produce an incorrect clustering, a problem we discuss closer in Chapter 4. We see from Figure 1 that the Neptune attacks are clustered together with the normal activities, but the Buffer overflow and Root kit exploits in cluster 1 provide a good demonstration of the outlier problem.

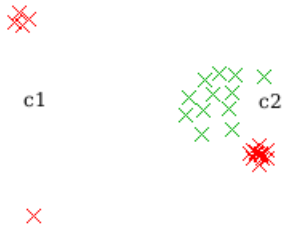

Figure 1: Illustration of the clustering of records 452000-453000

Table 5 shows that cluster 1 has both large intra-cluster and inter-cluster distances, which results in a very large S_{max}. The interesting part is that cluster 1 consists of four very similar buffer overflow exploits. We see that the distances between these four connections are very small, which results in a small S value in the C-index formula. The consequence is that the partial index when evaluating from the smallest cluster incorrectly indicates good clustering quality. We see in Table 5 that the evaluation from the largest cluster correctly indicates poor clustering quality in this situation.

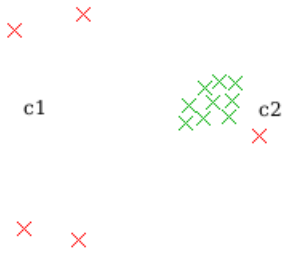

Figure 2: Illustration of the clustering of records 41000-42000

The clustering of the connection records from 41001 to 42000 consists of several distant outliers. From a total of five malicious connections we have two Ftp write exploits[1], two Satan probes[2] and one Buffer overflow exploit. This results in a cluster with very large intra-cluster distances, which can be seen in both Table 5 and Figure 2. In this situation the correct clustering quality is achieved by evaluating the clustering from the smallest cluster, while the evaluation from the largest cluster is incorrect. The reason

[1] privilege escalation through a poorly configured anonymous FTP server [32]
[2] early version of the SAINT probing utility [32]

is that the large intra-cluster distance yields a large S in the C-index formula, which neutralizes the large S_{max} value.

The difference between the two partial indexes is very large in both examples and exemplify that we can use the opposing partial indexes to determine the quality of the clustering.

3.2.3 Time complexity

It is computationally complex to compute the C-index. There are two main reasons for this; first computing distances between all pairs and then sorting these distances. Let n_1 and n_2 be the number of elements in the two clusters. The time complexity for computing the distance between all pairs in the clustering is then:

$$O\left(\binom{n_1}{2} + \binom{n_2}{2}\right)$$

The other contributor to the high complexity is the sorting algorithm. We need to sort an array that consists of $\frac{n(n-1)}{2}$distances, and this requires an efficient sorting algorithm. The time complexity of the C-index computation therefore depends on the complexity of the sorting algorithm. Martin et al. [42] provides a thorough discussion on sorting techniques, where several sorting algorithms are explained. The author also presents mathematical and experimental analysis of the algorithms.

We used the Merge-sort [34] technique in our application of C-index. Merge-sort is an efficient "divide and conquer" sorting algorithm that operates with $O(N \log N)$ time complexity. Among its advantages, compared to e.g. Quicksort [25], is that it is very stable both in complexity and implementation. The complexity only increases with $O(N)$ in the worst case scenarios [49], while Quicksort's time complexity is near $O(N^2)$ in such scenarios.

The total time complexity for the computation of the modified C-index is then near quadratic in the number of instances in the clustering:

$$O\left(\binom{n_1}{2} + \binom{n_2}{2}\right) + O\left((n_1 + n_2)\log(n_1 + n_2)\right)$$

3.3 Labelling algorithms

The study of the clustering properties gives us the following algorithms for labelling activity clusters, with the use of the two clustering quality indexes and the cluster diameters. Algorithm 1 describes the labelling algorithm with the application of Dunn's index and the application of the modified C-index is described in Algorithm 2.

The algorithms initially assume that C_1 consists of normal activities. C_1 is therefore initially labelled as normal, while C_2 is assumed to be the malicious cluster. If any of the conditions in the algorithms are met, indicating that this initial labelling is incorrect, the clusters are relabelled. By relabelling we mean that the label of the two clusters are interchanged.

Input:
• Clustering C of N activities into clusters C_1 and C_2, where C_1 initially is labelled as normal.
• The Dunn's index threshold, Δ_D.
• The centroid diameters difference parameters, Δ_{CD_1} and Δ_{CD_2}.
Output:
• Relabelled clustering if relabelling conditions are met.
begin
$D = DunnsIndex(C)$;
$cd_1 = CentroidDiameter(C_1)$;
$cd_2 = CentroidDiameter(C_2)$;

if *(*$D = 0$*)* ***and*** *(IsEmpty(*C_2*))* **then**
Relabel(C) ; /*Condition 1*/
else if *(*$D < \Delta_D$*)* ***and*** *(*$cd_1 > cd_2 + \Delta_{CD_1}$*)* **then**
Relabel(C) ; /*Condition 2*/
else if *(*$D > \Delta_D$*)* ***and*** *(*$cd_1 + \Delta_{CD_2} < cd_2$*)* **then**
Relabel(C) ; /*Condition 3*/
end

Algorithm 1: Labelling algorithm with Dunn's index

Input:
• Clustering C of N activities into clusters C_1 and C_2, where C_1 initially is labelled as normal.
• The modified C-index threshold, Δ_C.
• The centroid diameters difference parameters, Δ_{CD_1} and Δ_{CD_2}.
• The partial index difference threshold, Δ_{Diff}.
Output:
• Relabelled clustering if relabelling conditions are met.
begin
$Ci =$ *C-mean or C-small* ;
$C_{diff} = Abs(partial\ index_1 - partial\ index_2)$;
$cd_1 = CentroidDiameter(C_1)$;
$cd_2 = CentroidDiameter(C_2)$;

if $C_{diff} > \Delta_{Diff}$ **then**
$Ci = 1$ /*Condition 4*/

if *(*$Ci = 0$*)* ***and*** *(IsEmpty(*C_2*))* **then**
Relabel(C) ; /*Condition 1*/
else if *(*$Ci > \Delta_C$*)* ***and*** *(*$cd_1 > cd_2 + \Delta_{CD_1}$*)* **then**
Relabel(C) ; /*Condition 2*/
else if *(*$Ci < \Delta_C$*)* ***and*** *(*$cd_1 + \Delta_{CD_2} < cd_2$*)* **then**
Relabel(C) ; /*Condition 3*/
end

Algorithm 2: Labelling algorithm with the modified C-index

The formula for the centroid diameter (3.3), presented in Section 3.1, is used to compute the cluster diameter used by the labelling algorithms. The two diameter parameters, Δ_{CD_1} and Δ_{CD_2}, are used to optimize the precision of the algorithms, because the difference between the cluster diameters will vary with the data set or network we analyse. Proper adjustment of these parameters is important to achieve good accuracy, especially when there is a very small difference in the diameter size between normal and malicious clusters. The main reason for using two separate parameters is that we expect the optimal value of the difference parameter depends on whether or not a massive attack is present in the observed data.

The first three conditions are exactly the same for both Algorithm 1 and 2, while the fourth condition is only applied when we use C-small.

Condition 1

A condition that must be treated in a special way occurs when the output from the clustering algorithm yields one empty cluster. Dunn's index for a clustering with an empty cluster is zero, because there is no inter-cluster distance to measure. C-index is, for the same reason, equal to zero when one cluster is empty. S, S_{max} and S_{min} are then always equal, because all elements in the clustering are used to compute all three variables.

If the non-empty cluster is extremely compact, the natural conclusion is that this is the attack cluster, e.g. that all N observed activities are malicious. We therefore relabel the clusters if the clustering quality index is zero and cluster C_2 is empty.

Condition 2

Poor clustering quality indicates that there is no massive attack present in the observed activities. Small values of the Dunn's index indicate poor clustering quality, and we assume that no massive attack is present if Dunn's index is below the Δ_D threshold. High values of the modified C-index indicate poor clustering quality, and if the modified C-index is above the Δ_C threshold, we assume that no massive attack is present.

When no massive attack is present the large difference between malicious activities will cause the malicious cluster to be very scattered. We therefore relabel the clusters if the clustering quality index indicates poor clustering quality and C_1 is less compact than C_2, because this indicates that C_1 is the malicious cluster. The cluster diameter difference parameter (Δ_{CD_1}) is used to optimize the precision of the algorithm, e.g. that the diameter of cluster 1 must be Δ_{CD_1} larger than cluster 2 before we relabel the clusters.

Condition 3

Good clustering quality indicates that a massive attack is present in the observed activities. High values of the Dunn's index indicate good clustering quality and if the Dunn's index is above the Δ_D threshold, we assume that there is a massive attack present. Small values of the C-index indicate good clustering quality, and if the modified C-index is below the Δ_C threshold, we assume that a massive attack is present.

Malicious activities that are part of a massive attack form very compact clusters. We therefore relabel the clusters if the clustering quality indexes indicate good clustering quality and C_1 is more compact than C_2, because this indicates that C_1 is the malicious cluster. The difference parameter Δ_{CD_2} is used in the same way as Δ_{CD_1} above.

Condition 4

Condition 4 is only applied when we use the C-small alternative as the modified C-index. We have seen that a large difference between the two partial indexes indicates poor clustering quality. The C-means alternative always indicates poor clustering quality when there is a large difference between the partial indexes, because C-mean is the average of these two partial indexes. However, we need to adjust the quality index in these situations when we use the C-small alternative. We therefore set the modified C-index equal to 1 if this difference is larger than the Δ_{Diff} threshold. This condition is therefore not directly a relabelling condition, but an adjustment of the clustering quality evaluation.

4 Experimental work

The purpose of our experimental work is to investigate and measure the performance of our labelling strategy, which combines clustering quality evaluation techniques with cluster properties to achieve proper labelling of activity clusters. As presented in Chapter 3, the focus of our work is the application of Dunn's index and C-index in this labelling strategy. Our aim for the experimental work is therefore to gain knowledge about and evaluate the performance of the labelling strategy with the use of these two clustering quality indexes. Through the experimental work, the performance of the two clustering quality indexes can be compared with each other and with other clustering quality indexes previously applied for labelling activity clusters. It also enables us to compare the performance of this labelling strategy with classical cardinality based labelling strategies. This is necessary to answer our research questions and to determine whether or not our labelling strategy solves the limited capability of cardinality based strategies to detect massive attacks.

To measure the performance of our labelling strategy we must consider both the efficiency and the accuracy of the system. After the theoretical research, presented in Chapter 3, we assumed the following expectations associated with the application of Dunn's index and C-index for labelling activity clusters:

1. The modified C-indexes yield more accurate measurements of the clustering quality than Dunn's index, and should therefore yield better labelling accuracy.
2. The computation of the modified C-indexes is much slower than the computation of Dunn's index.

An important issue of the experimental work is therefore to evaluate whether or not these expectations hold. While the efficiency can be evaluated with time complexity analysis of the clustering quality indexes, we need to perform experiments on test data to measure the accuracy of the labelling strategy. For this purpose, a prototype that applies the labelling strategy on obtained activity clusters was developed. This prototype was also used to gather the data needed for the theoretical research, leading to the labelling algorithms presented in Chapter 3.

In the following, the prototype is presented in Section 4.1. The experimental setup is described in Section 4.2 and our expectations for the experiments are presented in Section 4.3. A short description of these experiments is given in Section 4.4 and Section 4.5 presents the results from these experiments. Then, a discussion and summary concerning these results are given in Section 4.6.

4.1 The prototype

This section presents the prototype that performs the clustering of activity data, and we describe how the two clustering quality indexes are applied in this prototype. The base system in the prototype was developed by Petrovic et al. [50, 51, 52]. This system has the design of a multiple classifier IDS, as illustrated in Figure 3, and consists of the following components:

1. Sensors that observe the same data set in parallel. These sensors merely classify the observed data by means of clustering, without performing any interpretation of the obtained clusters. The output from the sensors is K clusters, which are passed on to the assessors.
2. Assessors that are responsible to interpret and label the clusters obtained by the sensors. A criterion function, based on clustering quality indexes and cluster parameters, is used in the assessors to compute a value for every sensor on the data set. This value is then measured against a threshold to determine the nature of the obtained clusters.
3. The manager part of the system is where the parameters of the sensors and assessors can be adjusted to achieve optimized performance. This is also where the results from the assessors are presented to the human IDS analyst. The manager part of the prototype is outside the topic of this thesis, and is not described closer.

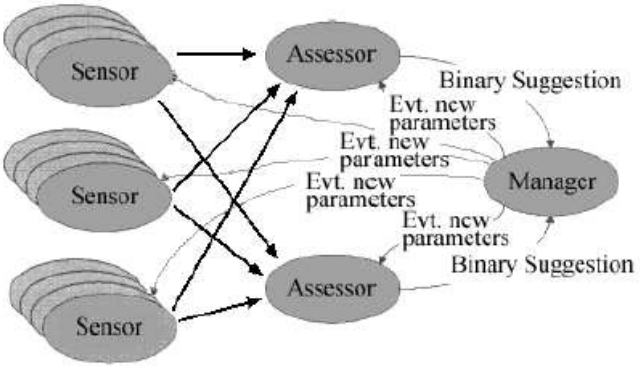

Figure 3: The design of the multiple classifier prototype. Derived from [50]

The prototype simulates the sensor-assessor structure of this multiple classifier IDS. In this prototype, a sensor is merely a clustering algorithm that clusters observed network activity, where the activity data is read from an offline database. The database of network activity is a reduced (10%) version of the KDD Cup '99 data set, which is described in Section 2.4.1. A short discussion on the choice of this data set is given in Section 4.2.

The prototype uses the well known K-means algorithm for clustering of the network activity, an algorithm we described in Section 2.1. A problem associated with the K-means algorithm, is choosing the optimal number of clusters. However, because the purpose of the prototype is to develop and analyse algorithms for labelling activity clusters, the prototype is limited to create only two clusters.

In this two-cluster setup, we assume that one cluster corresponds to normal activity and that the other cluster corresponds to malicious activity. The reason is that we expect that malicious and normal activities are clustered into separate clusters, as described in Chapter 3. It is then appropriate to use only two clusters, because the aim of a labelling strategy is to determine whether activities are malicious or benign. It also becomes easy to compare the performance of our labelling algorithms with other labelling algorithms.

We have seen in the literature that the K-means algorithm is used often in clustering based IDSs, see e.g. Zhong et al. [63]. These approaches indicate that it is very difficult to find the optimized number of clusters, and that this is very important for the accuracy of the clustering. We must therefore expect that we get a quite high error rate due to the clustering algorithm itself, when we choose to create only two clusters.

The prototype clusters 1000 elements of observed network connections from the KDD Cup '99 data set into two clusters at each iteration. To cluster these connections the K-means algorithm must compute the distance between observed activities and the cluster centroids. This distance is the Euclidean distance between the 41 features describing a connection in the KDD Cup '99 data set, and the 41 features defining a cluster centroid, which is the average of the 41 features of all instances within a cluster. We can then place the observed activity in the cluster to which it is closest to, by means of this distance. By using the Euclidean distance, the 41 features are reduced into one single measure, similar to what is done in e.g. the NIDES [28] system. The advantage with this feature reduction is that it is very simple to compare activities with the use of a single measure, but the disadvantage is that there is a risk of decreased accuracy. It is possible that, e.g. a low value of one feature may mask high values of other features etc.

When the clustering of observed activities has been performed by the sensors, the resulting clusters are sent to the assessors. There the quality of this clustering is computed with the chosen clustering quality index. Sections 4.1.1 and 4.1.2 describe the steps used to compute Dunn's index and C-index in these assessors. Together with the cluster diameters, this clustering quality evaluation is used to label the clusters with the algorithm of the respective clustering quality index, as presented in Section 3.3.

Several functions for the computation of cluster parameters were used to perform the theoretical research, which lead to the findings presented in Chapter 3. Most of these functions are primarily implemented in the prototype for the computation of the clustering quality indexes, e.g. the centroid diameter for the computation of Dunn's index. These functions provide good description of the physical properties of the clustering, and were therefore a good source of information for the theoretical research. In addition, functions that compute other cluster parameters, e.g. the maximum centroid distance, were implemented to investigate different cluster properties when different types of activity were observed. Another example of such a function is to find the maximum distance between two instances within the same cluster.

4.1.1 Computation of Dunn's index

The computation of Dunn's index is based on measuring the intra-cluster and inter-cluster distances from the cluster centroids. These operations are not very complex and the computation of the index is rather fast. The following steps are used to compute Dunn's index:

1. Compute the inter-cluster distances between clusters, by using the formula (3.1) to find the centroid linkage between the clusters.
2. Compute the intra-cluster distances of the clusters, by using the formula (3.3) to find the centroid diameters.
3. Compute Dunn's index according to the formula (2.7) presented in Section 2.2: Select the smallest intra-cluster distance in the clustering and divide this distance by the

largest inter-cluster distance. In the two-cluster case, there is only one single inter-cluster distance. This inter-cluster distance is therefore used directly in the formula.

4.1.2 Computation of the modified C-index

Dunn's index and the C-index take two distinct approaches to evaluate the quality of the clustering, which can be seen in the way the indexes are computed. While the former is computed using intra-cluster and inter-cluster distances from the centroids of each cluster, the latter is computed by measuring the distance between all elements in the clustering. It is easy to see that the computation of the C-index is much more computationally complex than Dunn's index. The following steps are used to compute the partial indexes, which are used to assemble the modified C-index (the two-cluster case):

1. If there are no empty clusters, we begin by computing the distance between all pairs in the clustering. The array of these distances is then sorted from small to large. If there is one empty cluster; set the C-index to zero and skip the remaining steps.
2. Choose a cluster that the partial index has not previously been evaluated from.
3. Compute the distance between pairs within the chosen cluster, S.
4. Find the number of pairs, l, necessary to compute S_{max} and S_{min}, with the formula $\frac{c(c-1)}{2}$, where c is the cardinality of the chosen cluster.
5. Compute S_{max} by adding up the l largest distances between all pairs in the clustering.
6. Compute S_{min} by adding up the l shortest distances between all pairs in the clustering.
7. Compute the formula (2.8), presented in Chapter 2.2, to compute the partial index.
8. Repeat Step 2-7 if we have not computed the partial index from both clusters.

When the partial indexes are computed, we can assemble the modified C-index, as proposed in Section 3.2.1. If we choose to use the C-mean alternative of this modified C-index, we compute the average of the two partial indexes. To find the other alternative, C-small, we measure the cluster cardinalities and choose the partial index from the smallest cluster to represent the modified C-index.

4.2 Experimental setup

Comprehensive simulations were carried out to study the behaviour of the labelling strategy when applied on activity data. Because we used the base system of the prototype developed by Petrovic et al. [50, 52], it was natural to base our experimental setup on their setup, with some modifications to satisfy our needs, assumptions and expectations. The following setup of the prototype was used to study the labelling strategy:

1. In the sensor we applied the K-means clustering algorithm, with $K = 2$, on 1000 instances of activity vectors at each iteration.
2. Our labelling strategy and a cardinality based labelling strategy were applied in four different assessors:

   2.1 The assessor applies the cluster cardinalities for labelling of activity clusters. This assessor is mainly used as a reference point to compare our labelling strategy with cardinality based strategies. A minimum difference threshold was used to

configure the assessor for optimal performance.

2.2 The assessor applies Dunn's index for labelling of activity clusters, according to Algorithm 1. This algorithm has the following parameters that need configuration to achieve optimal performance: the Dunn's index threshold Δ_D, and the cluster diameter parameters Δ_{CD_1} and Δ_{CD_2}.

2.3 The assessor applies the C-mean alternative of the modified C-index for labelling of activity clusters, according to Algorithm 2. This algorithm has the following parameters that need configuration to achieve optimal performance: the C-mean index threshold Δ_C, and the cluster diameter parameters Δ_{CD_1} and Δ_{CD_2}.

2.4 The assessor applies the C-small alternative of the modified C-index for labelling of activity clusters, according to Algorithm 2. This algorithm has the following parameters that need configuration to achieve optimal performance: the C-small index threshold Δ_C, and the cluster diameter parameters Δ_{CD_1} and Δ_{CD_2}. The assessor was tested with and without condition 4 in Algorithm 2. When condition 4 is applied, the parameter Δ_{Diff} also needs configuration.

We chose the KDD Cup '99 test data set as traffic source in our simulations of the labelling strategy. The selection of such test data sets is, according to Mell et al. [44], one of the challenges of testing IDSs. The main problem is regarding the use of background traffic in the test data. Four approaches are defined in [44] to handle this problem. Each of these approaches has advantages and disadvantages, and it is not clear what approach is the most efficient for testing IDSs:

1. Test data without background traffic:
   These data sets only consist of malicious data and can only be used to determine the detection rate of the system. Such data sets are therefore only appropriate for reference testing, where the aim is to prove the capability of detecting certain types of attacks. The assumption behind these data sets is that the detection rate is not affected by the background traffic surrounding the attacks. This assumption often fails at high levels of background traffic, i.e. when normal traffic vastly outnumbers malicious traffic, which is the situation in most networks. This is emphasized by Axelsson [1] who presents the base-rate fallacy, which shows that the performance of an IDS deteriorates at high traffic intensities.

2. Test data derived from real traffic logs:
   These data sets consist of injected attack vectors in a real data stream. The advantage of using real data is that it consists of all characteristics and anomalies of a real network. A drawback it that we cannot be sure that all attack vectors in the test data set have been identified. The consequence is that it is very difficult to measure the false positive rates correctly. Another drawback is that there are some legal issues concerned with the use of real data, because the data may consist of sensitive information that cannot be distributed.

3. Sanitized test data from real traffic logs:
   Sanitized data sets solve the legal issues concerned with the use of real data. However, it can never be guaranteed that 100% of the sensitive information have been identified and removed from a large data set. The problem with unidentified attacks

remains present.

4. Artificial simulated test data:
   Simulations of a network infrastructure are used to generate the test data sets. Both attack vectors and background traffic are generated and inserted into the simulated data sets. This ensures that we can achieve a correct measure of both the false positive rate and the true positive rate. The drawback is that it is difficult to create a simulation setup that yields high quality data sets that correctly reflect real networks.

KDD Cup '99 is an artificial data set, well known and often used in academic communities. This data set consists of several sequences of massive Denial-of-Service attacks, because such attacks are typical for the simulated environment it originates from. For our purpose this is a preferable characteristic of a data set; since the aim of our labelling strategy is to handle the limited capability of cardinality based labelling strategies to detect massive attacks. We therefore chose this test data set, in spite of the criticism the quality of the KDD Cup '99 data set has seen, as discussed in Section 2.4.1.

Receiver operating characteristic (ROC) curves are used to measure the accuracy of the system. These curves have become an important tool for measuring the performance in the IDS testing community, because a ROC curve "summarizes the relationship between two of the most important IDS characteristics: false positive and detection probability[1]" [44]. To achieve curves, and not just a single point, we tested different configurations of the parameters used in the labelling algorithms presented in section 3.3. Since more than one parameter is used in these algorithms, we created automated procedures/scripts to test all combinations of these parameters within reasonable ranges.

Testing all possible combinations of these parameters is too cumbersome and time consuming, as these parameters have large ranges of values. However, the two cluster diameter parameters, which are used in both labelling algorithms, primarily depend on the observed data used in the simulations. We can therefore find the optimal configuration of these two parameters first. When we have found this configuration, the ROC curve, describing the system accuracy, is obtained by adjusting the clustering quality index threshold. Both negative and positive values of the two cluster diameter parameters must be tested to obtain the optimal configuration. The reason for including negative values is that peculiar characteristics of the data set may cause the diameter of normal and malicious clusters to be very similar. It may then be necessary with small negative values of these parameters in order to separate the activities.

4.3 Expectations

Our "performance expectations" are reflected by the expectations presented in the beginning of this chapter. In addition to this, we expect that the similarity between the Dunn's index and the Davies-Bouldin index, proposed and used for labelling clusters by Petrovic et al. [51], will result in similar detection accuracy. We also expect that the assessor that applies cardinality based labelling will fail completely because of the high amount of massive Denial-of-Service attacks present in the data set.

An underlying expectation is that normal and malicious activities are placed in separate clusters because of their differences. However, since we apply a "2-means" variant of the K-means algorithm to perform the clustering in the prototype, this is not always the

[1] false and true positive rate

case. The quality of the obtained clustering from the K-means algorithm depends, among other things, on finding the optimal number of clusters, and the optimal number of clusters is rarely 2. We must therefore expect errors in the resulting clustering, where a few instances of malicious activities may be present in the cluster that corresponds to normal activity, and vice versa. The consequence is that we cannot achieve 100% detection rate and or 0% false alarm rate, even if the labelling algorithm is perfect.

We can measure this error rate by measuring the Hamming distance between the obtained clustering from the "2-means" algorithm and the correct clustering. This measure shows that the best possible true positive rate that can be achieved is 94%, while the best possible false positive rate is 7.1% when "2-means" is applied on the KDD Cup '99 data set[2]. Table 6 shows the number of attack vectors incorrectly classified as normal (false negatives) and the number of normal vectors incorrectly classified as malicious (false positives), from the clustering of the first 8000 connections in the data set. We see that a few normal vectors are misplaced in most iterations and that some malicious activities are incorrectly classified as normal, e.g. the two attack vectors in the clustering of connections 3001 to 4000. In total, the clustering errors produce four false negatives and 316 false positives from these 8000 connections.

Record no.	Number of attacks	FN	Total number of FN	FP	Total number of FP
1-1000	0	0	0	43	43
1001-2000	0	0	0	66	109
2001-3000	0	0	0	63	172
3001-4000	2	2	2	11	183
4001-5000	0	0	2	18	201
5001-6000	0	0	2	107	308
6001-7000	0	0	2	8	316
7001-8000	376	2	4	0	316

Table 6: Clustering errors due to the use of "2-means"

A consequence of the problem explained above is that these clustering errors may have an unfavourable effect on the cluster diameters. This may lead to decreased accuracy of the labelling algorithm, because these diameters are used directly to determine the nature of the clusters. One example is a normal vector misplaced into the malicious cluster in the presence of a massive attack. This misplacement may result in an incorrectly large cluster diameter of this cluster. In the worst case, this may cause the labelling strategy to fail, since this misplaced activity may have "tainted" the cluster diameters.

4.4 Experiments

A total of five experiments were performed to gather the information necessary to analyse the performance of the labelling strategy. In this section, we provide a description of the procedure used to carry out these experiments. Testing all combinations of all parameters and thresholds is a time consuming and cumbersome task, highly prone for errors. Initial experiments were therefore carried out to find the optimal configuration of the two cluster diameter parameters used in experiments 2 – 5. With this configuration found, the clustering quality threshold was adjusted separately to obtain the ROC curve. Both the diameter difference parameters, Δ_{CD_1} and Δ_{CD_2}, were tested in the range from

[2]The calculation of the best possible TPR and FPR is described in Appendix B

-1000 to 1000, initially with large intervals and later with very small intervals around the assumed optimal configuration based on the initial experiments. The optimal configuration was achieved with $\Delta_{CD_1} = 500$ and $\Delta_{CD_2} = -25$.

To control that the results were not biased by our procedures for finding the optimal configuration, we performed control tests after completing the experiments (2 – 5). In these control tests, we tested all possible combinations of all parameters within a smaller range around the optimal configuration of all parameters. The control tests showed that the optimal configuration of the labelling algorithm was obtained by this procedure in all cases, and confirm that our procedure for finding the optimal configuration is appropriate.

4.4.1 Exp. 1: Labelling clusters by means of cluster cardinality

We performed a simple experiment with the use of a cardinality based labelling strategy, where we adjusted a minimum difference threshold to obtain a ROC curve. This experiment was primarily performed in order to reproduce the results presented by Petrovic et al. [51, 52], which show that a cardinality based labelling strategy fails completely in the presence of massive attacks. It also enables us to compare the accuracy of the different strategies in such environments.

4.4.2 Exp. 2: Labelling clusters by means of Dunn's index

This experiment was performed to investigate the application of Algorithm 1 in the assessor, where Dunn's index is used in combination with cluster diameters to label the obtained clusters. The theoretical research indicated that the range between 1 and 4 is a reasonable range for adjusting the Dunn's index threshold (Δ_D). We therefore adjusted Δ_D with small intervals within this range, and with larger intervals outside it. The total range, for which Δ_D was tested, is from 0 to 10.

4.4.3 Exp.3: Labelling clusters by means of modified C-index, C-mean

This experiment was performed to investigate the application of Algorithm 2 in the assessor, where the C-mean alternative of the modified C-index is used in combination with cluster centroid diameters to label the obtained clusters. The theoretical research indicated that the range between 0 and 0.2 is a reasonable range for adjusting the C-mean index threshold (Δ_C). We therefore adjusted Δ_C with small intervals within this range, and with larger intervals outside this range. The total range, for which Δ_C was tested, is from 0 to 1.

4.4.4 Exp. 4 and 5: Labelling clusters by means of modified C-index, C-small

These experiments were performed to investigate the application of Algorithm 2 in the assessors. In these experiments, the modified C- indexes presented in Section 3.2.1 were applied in combination with cluster diameters to interpret the nature of the obtained clusters. We described in Section 3.2.2 that a large difference between the partial indexes used to assemble the C-small index could indicate poor clustering quality. Two separate experiments were therefore performed with this assessor. The first experiment (4), where this finding is ignored, was performed exactly like the experiments 2 and 3. In the other experiment (5), this condition was considered. It was then also necessary to find the optimal difference parameter (Δ_{Diff}) before we could adjust the C-small index threshold (Δ_C) to obtain the ROC curve. The optimal configuration was achieved with $\Delta_{Diff} = 0.5$.

The theoretical research indicated that the range between 0 and 0.2 is a reasonable

range for adjusting the Δ_C threshold. We therefore adjusted Δ_C with small intervals within this range, and with larger intervals outside it. The total range, for which Δ_C was tested, is from 0 to 1.

4.5 Results

The results from the experiments described in Section 4.4 are used to investigate and evaluate the accuracy of our labelling strategy. In this section we present the accuracy of the labelling strategy by means of ROC curves. This is followed by analyses of the labelling algorithms, when applied on the test data set with optimal configuration. A discussion and summary regarding these results are given in Section 4.6 and Figure 9.

To describe the general application of the labelling algorithms, we use the first 16 iterations performed by the prototype to label the KDD Cup '99 data set, where the first 16000 connections are clustered and labelled. We use these iterations because we observe all conditions that the labelling algorithms must take into consideration, when investigating the resulting clusterings of these connections; i.e. normal activities, a few standalone attacks and a massive Denial-of-Service (Smurf) attack.

4.5.1 Labelling clusters by means of cluster cardinality

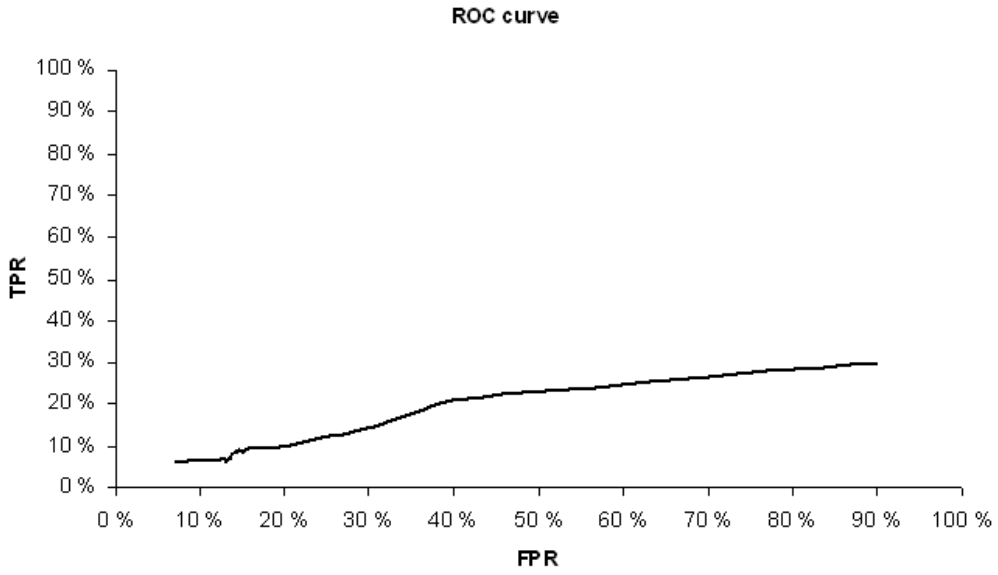

Figure 4: ROC curve describing the system accuracy, when a cardinality based labelling algorithm is applied in the assessor

Experiment 1 was performed to reproduce the results presented by Petrovic et al. [51, 52], which showed that cardinality based labelling fails completely in the presence of many massive attacks. The accuracy of the cardinality based labelling strategy is presented by the ROC curve in Figure 4 and confirms the results presented in [51, 52]. By studying this ROC curve, we observe that the true positive rate is very poor (below 20%)

for most configurations. This confirms our expectation that cardinality based labelling is not well suited to detect massive attacks. The false positive rate is rather good (below 10%) for the best configurations of the system and indicates that cardinality based labelling strategies are well suited for labelling when no massive attacks are present. This is in accordance with results presented in various approaches for clustering based intrusion detection, e.g. by Portnoy et al. in [53].

4.5.2 Labelling clusters by means of Dunn's index

Experiment 2 was performed to evaluate the performance of the system when Dunn's index is applied for labelling activity clusters. By following the procedure for adjusting the parameters of Algorithm 1 described in Section 4.4, the ROC curve illustrated in Figure 5 was obtained. The optimal configuration of Algorithm 1 is achieved with $\Delta_{\mathrm{D}} = 2.2$, $\Delta_{\mathrm{CD}_1} = 500$ and $\Delta_{\mathrm{CD}_2} = -25$. With this configuration, the true positive rate is 85.9% and the false positive rate is 15.3%.

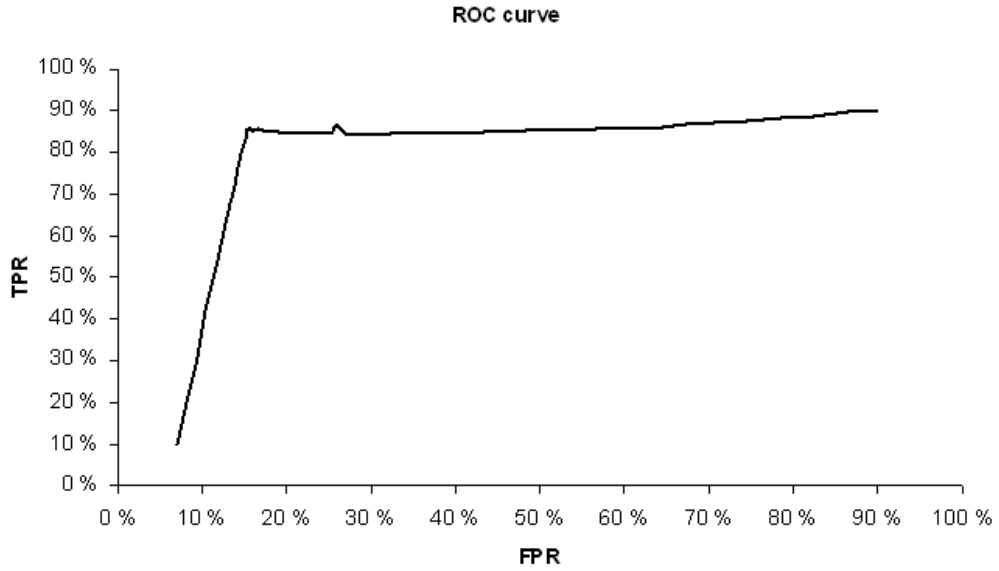

Figure 5: ROC curve describing the system accuracy, when Dunn's index is applied in the assessor to label activity clusters. This curve was obtained with the following configuration of Algorithm 1: varying Δ_{D} 0-10, $\Delta_{\mathrm{CD}_1} = 500$ and $\Delta_{\mathrm{CD}_2} = -25$.

The results presented in Figure 5 are, as expected, similar to the results achieved by Petrovic et al. in [51], where the Davies-Bouldin index is applied for labelling clusters. This similarity can be explained with the fact that the two indexes utilize the same clustering properties to compute the clustering quality, although they are used quite differently. Compared to the results from the cardinality based labelling presented in Section 4.5.1, we see that our labelling strategy yields much better detection rate. This supports our expectation that our labelling strategy is better suited for detection of massive attacks than cardinality based strategies.

Table 7 shows the application of Algorithm 1 on the first 16000 connections in the KDD Cup '99 data set. In addition to describing the number of attacks present in the observed connections, each column in this table shows the values of the parameters used by the labelling algorithm and what relabelling condition was applied on the clustering. If the relabelling condition is 0, this means that the initial labelling is correct. The application of Algorithm 1 does not produce any labelling errors on the clusterings described in Table 7.

Record no.	Attacks	Dunn's index	CD1	CD2	Relabel
1–1000	0	1.08	32759	7109	2
1001–2000	0	1.40	17273	5216	2
2001–3000	0	1.47	16234	4772	2
3001–4000	2	1.42	50254	6980	2
4001–5000	0	0.75	76358	7274	2
5001–6000	0	1.35	4345	14159	0
6001–7000	0	1.47	7488	85019	0
7001–8000	376	7.05	70	7096	3
8001–9000	1000	0	25	-	1
9001–10000	1000	0	11	-	1
10001–11000	1000	0	0	-	1
11001–12000	321	6.85	190	7303	3
12001–13000	0	0.99	33783	5844	2
13001–14000	0	1.21	5064	22257	0
14001–15000	0	1.06	243757	8973	2
15001–16000	21	1.12	18071	8786	2

Table 7: Labelling of the first 16000 records from the KDD Cup '99, with the use of Algorithm 1. This labelling is achieved with the following configuration: $\Delta_{\mathrm{D}} = 2.2, \Delta_{\mathrm{CD}_1} = 500$ and $\Delta_{\mathrm{CD}_2} = -25$. CD1 and CD2 are the centroid diameters of the two clusters.

The clustering of connections 7001 to 8000 is a good example of the application of Algorithm 1 on the test data set. In this clustering, cluster 1 consists of connections from a Smurf attack and cluster 2 consists of various normal activities. Cluster 1 is initially assumed to correspond to normal activities and it is therefore necessary to relabel the clusters. The clustering quality evaluation yields a value well above the threshold (Δ_{D}), which indicates that a massive attack is present. From our theoretical research, we recall that the cluster with the most compact cluster, i.e. the smallest cluster diameter, probably is the malicious cluster in the presence of a massive attack. In Table 7, we see that cluster 1 has a much smaller cluster diameter than cluster 2. This indicates that the initial labelling is incorrect, and we relabel the clusters with condition 3 of Algorithm 1.

Another example is the following clustering of connections 8001–9000, where we have one empty cluster. In this case Dunn's index is zero, because there is no inter-cluster distance to measure. The system applies relabelling condition 1 on this clustering, because cluster 2 is empty and the cluster diameter of the non-empty cluster is very small, which indicates that the cluster consists of malicious activities.

It is reported in literature, see e.g. [22, 23], that Dunn's index may be vulnerable for noise, e.g. outliers, in the data set and produces unstable results in such situations. We have only seen some few examples of this, e.g. in the clustering of connections 346001–347000. In this clustering we have a total of 181 malicious connections. These connections are mainly connections of Ipsweep and Teardrop attacks, but we also find one

Portsweep attack, one “Ping of Death” attack[3] and one Buffer overflow exploit. The three latter appears as outliers in the vector space, which causes the Dunn’s index to produce an incorrectly low value. This indicates that no massive attack is present and the algorithm therefore incorrectly labels the cluster with the smallest cluster diameter as normal. The result is that we get 181 false negatives and 819 false positives from these 1000 connections. By adjusting the clustering quality threshold or the cluster diameter parameters we can detect most of these attacks, but this will result in other errors.

4.5.3 Labelling clusters by means of modified C-index index, C-mean

Experiment 3 was performed to evaluate the performance of the system when the modified C-index, C-mean, is applied for labelling activity clusters. By following the procedure for adjusting the parameters of Algorithm 2 described in Section 4.4, the ROC curve illustrated in Figure 6 was obtained. The optimal configuration of Algorithm 1 is achieved with $\Delta_C = 0.07$, $\Delta_{CD_1} = 500$ and $\Delta_{CD_2} = -25$. With this configuration, the true positive rate is 84.7% and the false positive rate is 22.8%.

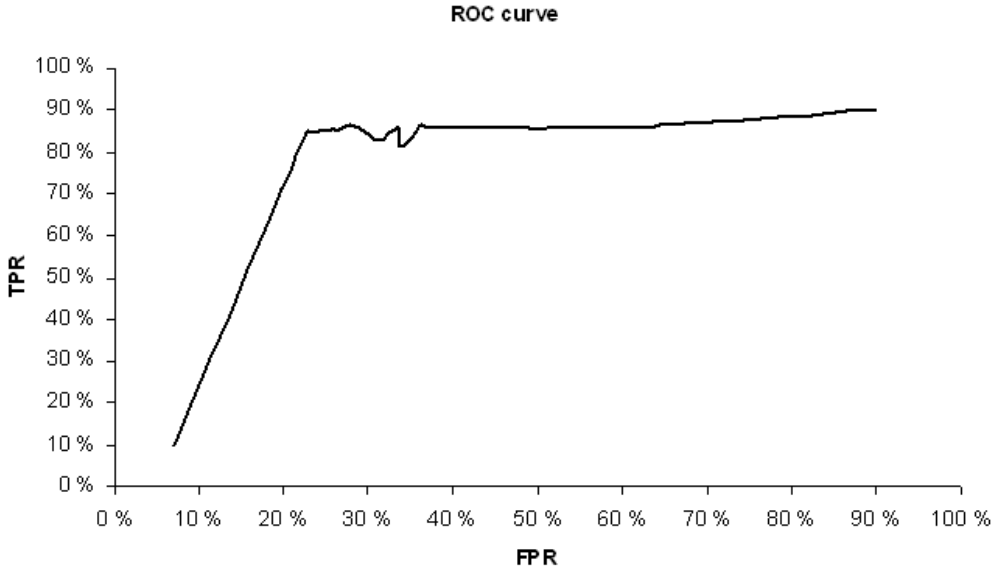

Figure 6: ROC curve describing the system accuracy, when the modified C-index (C-mean) is applied in the assessor to label activity clusters. This curve was obtained with the following configuration of Algorithm 2: varying Δ_C 0-1, $\Delta_{CD_1} = 500$ and $\Delta_{CD_2} = -25$.

We expected that the application of the modified C-index should yield better accuracy than Dunn’s index, and the accuracy of the labelling is therefore below the expectations. The slightly decreased true positive rate compared to Dunn’s index is acceptable, as it is more important to reduce the number of false positives than increasing the true positive rate. However, as illustrated in Figure 6, we see that the false positive rate is very poor (well above 20%) for configurations that yield acceptable true positive rates.

[3] oversized ping packet causing the targeted host to malfunction [32]

The application of Algorithm 2, with the use of the C-mean index, on the first 16000 connections in the data set is described in Table 8. Even though the total performance of the system proved to be rather poor, the application of C-mean did not produce any labelling errors on these clusterings. This suggests that the use of the C-mean index produces proper labelling when the clusters do not consist of any irregularities.

Record no.	Attacks	C-mean	CD1	CD2	Relabel
1–1000	0	0.20	32759	7109	2
1001–2000	0	0.15	17273	5216	2
2001–3000	0	0.10	16234	4772	2
3001–4000	2	0.17	50254	6980	2
4001–5000	0	0.23	76358	7274	2
5001–6000	0	0.11	4345	14159	0
6001–7000	0	0.13	7488	85019	0
7001–8000	376	0.04	70	7096	3
8001–9000	1000	0	25	-	1
9001–10000	1000	0	11	-	1
10001–11000	1000	0	0	-	1
11001–12000	321	0.04	190	7303	3
12001–13000	0	0.13	33783	5844	2
13001–14000	0	0.11	5064	22257	0
14001–15000	0	0.19	243757	8973	2
15001–16000	21	0.31	18071	8786	2

Table 8: Labelling of the first 16000 records from the KDD Cup '99, with the use of Algorithm 2 (without condition 4) and C-mean. This labelling is achieved with the following configuration: $\Delta_C = 0.07, \Delta_{CD_1} = 500$ and $\Delta_{CD_2} = -25$

The main cause for the poor accuracy originates from the problem described in Section 3.2. When we have clusters of very different cardinality, there is a risk that irregularities in the data set may cause incorrect clustering quality evaluations. The problem is that the inter- and intra-cluster distances are not equally considered in the partial indexes in these situations. Another problem is that a few distances may dominate the other distances in the evaluations. Most clusterings of the KDD Cup '99 data set consist of clusters with different cardinalities, and irregularities within these clusterings may then result in an incorrect partial index. An error in one of these partial indexes will therefore influence the output of C-mean, since the average of the partial indexes is used to compute the index. The result is many inaccurate clustering quality evaluations, which is the main contributor to the poor performance.

Outliers are a common cause of errors in the computation of the partial indexes, as discussed in section 3.2.2. The KDD Cup '99 data set consists of many standalone attacks, some of which will appear as outliers in the clustering space and lead to incorrect clustering quality evaluations. One example is the clustering of connections 76001–77000. The malicious connections in this clustering mainly consist of scanning attacks, but they also consist of a few Imap[4] and Land attacks[5]. These latter attacks appear as outliers in the clustering space, and causes C-mean to produce an incorrect evaluation of the clustering quality. As we discussed in section 3.2.2, the outliers may cause very different effect on the evaluations in different situations. The consequence is that C-means yields unstable

[4]Buffer overflow attack on the Imap service [32]

[5]Denial-of-Service attack with spoofed source IP address equal to the destination address [32]

results in these situations. It is also very difficult, if possible, to find configurations that yield proper labelling accuracy in such situations. By studying the ROC curves in Figure 6–8, we see this instability and the effect this has on the labelling accuracy; as the fluctuations in the ROC curve in Figure 6 are much larger than what is seen in the ROC curves in Figure 7 and 8. This indicates that using the average of the two partial indexes does not handle the outlier problem well.

4.5.4 Labelling clusters by means of modified C-index index, C-small

Experiment 4 was performed to evaluate the performance of the system when the modified C-index, C-small, is applied for labelling activity clusters. By following the procedure for adjusting the parameters of Algorithm 2 described in Section 4.4, the ROC curve illustrated in Figure 7 was obtained. The optimal configuration of Algorithm 2 is achieved with $\Delta_C = 0.095$, $\Delta_{CD_1} = 500$ and $\Delta_{CD_2} = -25$. With this configuration, the true positive rate is 84.5% and the false positive rate is 16.1%.

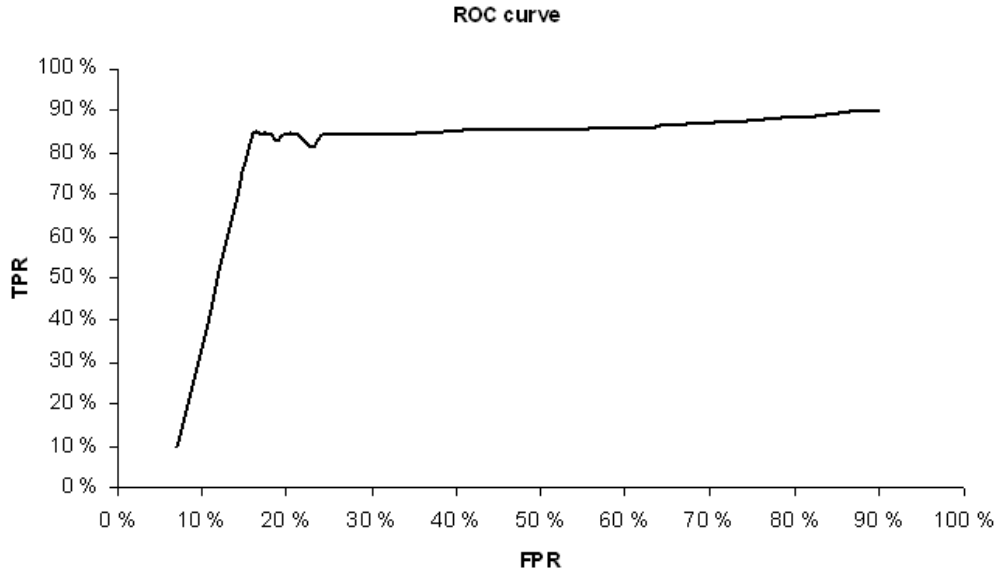

Figure 7: ROC curve describing the system accuracy, when the modified C-index (C-small) is applied in the assessor to label activity clusters. This curve was obtained without considering condition 4 in Algorithm 2 and with the following configuration: varying Δ_C 0-1, $\Delta_{CD_1} = 500$ and $\Delta_{CD_2} = -25$.

Compared to C-mean, C-small yields good improvement of the evaluation accuracy. Even so, the application of the modified C-index is still slightly less accurate than the application of Dunn's index, which is in opposition to our expectations regarding the labelling accuracy. These results do, however, show that the C-small alternative for the modified C-index provides proper evaluations of the clustering quality. They also confirm our assumption that both intra-cluster and inter-cluster distances are more equally considered with the C-small alternative. In addition, we expect that the labelling accuracy

will improve when condition 4 in Algorithm 2 is considered in experiment 5.

Before we present the results from experiment 5, where condition 4 is considered, we describe the application of Algorithm 2 on the first 16000 connections in the KDD Cup '99 data set. Since condition 4 does not apply to any of the clusterings in Table 9, this also describes the application of Algorithm 2 in experiment 5. By studying the values of C-mean and C-small in Table 8 and 9, we see that the evaluations with C-small yields values further away from the clustering quality threshold than C-mean. This indicates, what is also shown in Figure 6 and 7, that C-small yields improved evaluation accuracy. An example is the clustering of connections 7001-8000, where C-mean yields a value quite close to the optimal threshold, while C-small produces a value that clearly indicates that a massive attack is present.

Record no.	Attacks	C-small	CD1	CD2	Relabel
1–1000	0	0.17	32759	7109	2
1001–2000	0	0.17	17273	5216	2
2001–3000	0	0.19	16234	4772	2
3001–4000	2	0.30	50254	6980	2
4001–5000	0	0.23	76358	7274	2
5001–6000	0	0.19	4345	14159	0
6001–7000	0	0.26	7488	85019	0
7001–8000	376	0.00	70	7096	3
8001–9000	1000	0	25	-	1
9001–10000	1000	0	11	-	1
10001–11000	1000	0	0	-	1
11001–12000	321	0.00	190	7303	3
12001–13000	0	0.12	33783	5844	2
13001–14000	0	0.19	5064	22257	0
14001–15000	0	0.39	243757	8973	2
15001–16000	21	0.13	18071	8786	2

Table 9: Labelling of the first 16000 records from the KDD Cup '99, with the use of Algorithm 2 with C-small. This labelling is achieved with the following configuration: $\Delta_{CD_1} = 500$ and $\Delta_{CD_2} = -25$. For experiment 4 : $\Delta_C = 0.07$ and for experiment 5: $\Delta_C = 0.095$

The clustering of connections 15001–16000 in Table 9 is a good example of the application of Algorithm 2 with the use of C-small, and demonstrates the detection of attacks when there is no massive attack present in the observed activities. This clustering consists of 21 malicious connections, 20 Buffer overflow attacks on the Imap service and one password guessing attempt, which are initially clustered into cluster 1 and assumed to be normal activity. C-small correctly indicates that there is no massive attack present in this clustering, and the cluster with the smallest cluster diameter is probably the normal cluster. According to relabelling condition 2, this means that the initial labelling is incorrect and the labelling of the clusters is changed.

In sections 3.2.1 and 3.2.2 we claimed that the presence of outliers could cause C-small to produce inaccurate evaluations. This is also a common source of errors when C-small is applied for labelling through Algorithm 2. We reuse the first example given in Section 3.2.2 to demonstrate these errors. In the clustering of connections 452001–453000, four Buffer overflow attacks and one Root kit exploits are clustered together into a small cluster. As discussed in Section 3.2.2, the outliers in this clustering causes C-small to incorrectly indicate good clustering quality. The consequence of this error in the

quality evaluation is incorrect labelling by the labelling algorithm. Because of the incorrect evaluation, the relabelling algorithm incorrectly concludes that the initial labelling is correct. In Section 3.2.2, we proposed to use the opposing partial indexes in these situations to determine the clustering quality, which lead to condition 4 in Algorithm 2.

Condition 4 is considered in Experiment 5, where C-small is applied through Algorithm 2 for labelling the obtained clusters. By following the procedure for adjusting the parameters of Algorithm 2 described in Section 4.4, the ROC curve illustrated in Figure 8 was obtained. The optimal configuration of Algorithm 2 is achieved with $\Delta_{\mathrm{C}} = 0.095$, $\Delta_{\mathrm{CD}_1} = 500$, $\Delta_{\mathrm{CD}_2} = -25$ and $\Delta_{\mathrm{Diff}} = 0.5$. With this configuration, the true positive rate is 84.6% and the false positive rate is 12.9%. The ROC curve confirms our claim that the difference between the partial indexes, used to assemble C-small, can be used to determine the clustering quality. For most configurations the application of condition 4 decreases the false positive rate by 2.5–3.5%, while maintaining the same or in some situations also improving the true positive rate.

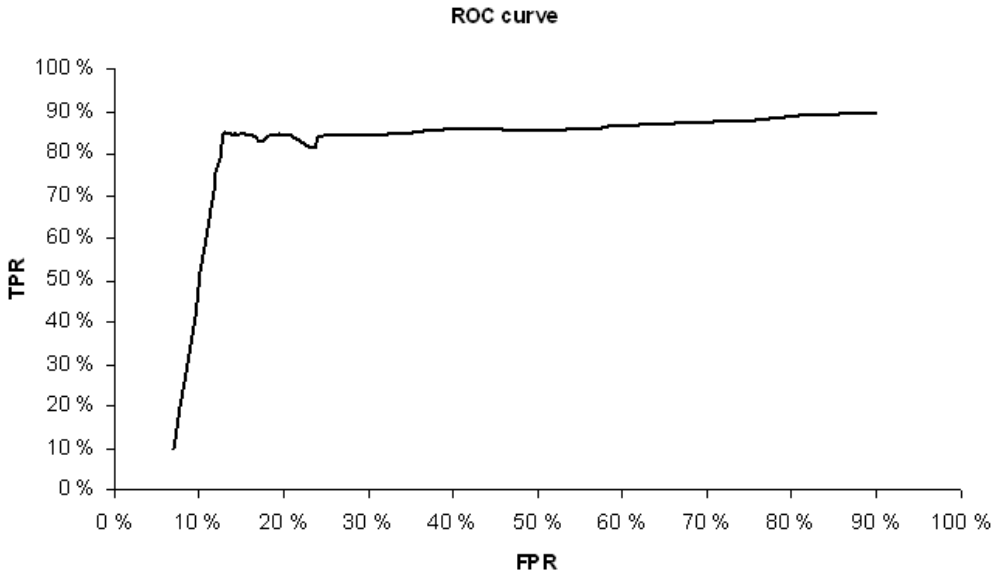

Figure 8: ROC curve describing the system accuracy, when the modified C-index (C-small) is applied in the assessor to label activity clusters. This curve was obtained by considering condition 4 in Algorithm 2 and using the following configuration: varying Δ_{C} 0-1, $\Delta_{\mathrm{CD}_1} = 500$, $\Delta_{\mathrm{CD}_2} = -25$ and $\Delta_{\mathrm{Diff}} = 0.5$.

We continue to use the clustering of connections 452001–453000 to describe the application of condition 4 and demonstrate the effect this condition has on the labelling accuracy. The partial indexes computed from this clustering is shown in Table 5, and show that the difference between the partial indexes is approximately 0.89. This is well above the difference threshold Δ_{Diff}. According to condition 4, the labelling algorithm should then adjust the clustering quality index to indicate poor clustering quality. Because the

C-small index now indicates poor clustering quality, the most compact cluster is probably the normal cluster. Algorithm 2 therefore detects that the initial labelling is incorrect, and correctly relabels the clusters when condition 4 is considered.

Our investigation of Algorithm 2 did not find any typical labelling situations that lead to errors when the C-small is applied and condition 4 is considered. The errors we did find are all related to the adjustment of the system parameters or the errors described in the following section.

4.5.5 Clustering errors

Many of the errors produced by the system are not caused by the labelling algorithm itself, but by the clustering algorithm used in the prototype and subsequent errors related to this algorithm. In this section, we describe typical errors produced by the system, independent of the applied labelling algorithms or clustering quality indexes.

Application of Algorithm 1 and 2 on the first 16000 connections does not produce any incorrect labelling of the clusters. We do, however, have some false positives and a few false negatives, even though the algorithms produce proper labelling. The reason is the clustering errors caused by the use of “2-means”, as discussed in Section 4.3. These clustering errors cause that we e.g. seldom obtain an empty malicious cluster in cases where there are only normal connections in the observed data. The first column of Table 7 – 9 is an example of this, which describes the clustering of the first 1000 connections in the data set. In this clustering, there are 43 benign connections that, according to “2-means”, are so different from the rest of the connections that they must be placed in a separate cluster. The labelling algorithm must therefore assume that these misplaced activities are of a different nature than the connections in the other cluster, which leads to false positives. To solve this, we could apply a more sophisticated and accurate clustering algorithm, e.g. one of those mentioned in Chapter 2. The problem with the use of such algorithms is high time complexity, which we discuss further in Section 4.6.

We observe several series of massive Neptune attacks in the KDD Cup ’99 data set, where all observed connections are attacks. About half of the clustering of these connections does not produce a clustering with one empty cluster, even though the observed Neptune attacks are very similar. In most of these situations we obtain clusterings where one cluster consists of approximately 950 connections, and the other of approximately 50 connections. The problem is that since both clusters have very small cluster diameters, it becomes impossible to determine which cluster to label as normal and malicious. This clustering error provides a good demonstration of the subsequent problems we have observed due to the clustering errors. Even though there are only approximately 50 misplaced connections in the clustering, the clustering error may become the direct cause of 950 incorrectly labelled connections. We have also observed a few instances where the same clustering error is produced when Smurf attacks are observed. In this situation the cluster sizes are approximately 650 and 350. Because the KDD Cup ’99 consists of many series of these two attack types, this error is the main contributor to the reduced true positive rate in the system.

Another typical situation that may cause the labelling algorithms to fail is the presence of two or more different massive attacks in the observed activities. This is the case in the clustering of e.g. connections 91001–92000, which consists of two different scanning attacks; Ipsweep and Satan. These two different scanning techniques are very different

from the normal traffic and are therefore placed together in a separate cluster. The problem is that they are also very different from each other, which leads to a malicious cluster that consists of two compact groups distant from each other. The cluster diameter of the malicious cluster is therefore larger than the normal cluster. When the clustering quality indexes correctly indicate that a massive attack is present, the labelling algorithm incorrectly labels the most compact cluster as malicious. The problem is related to the use of "2-means" for clustering. A solution to this specific problem, is to use K-means with e.g. $K = 3$, which will find two compact (malicious) clusters and one relatively wide (normal) cluster from these connections. However, even though the same principles behind our labelling strategy still hold, this will require a more complex labelling algorithm.

In one situation the K-means algorithm fails completely. Connections 42001–43000 consist of 127 Ipsweep connections and 873 connections of various benign activities. Even though the resulting cluster has an enormous cluster diameter, the algorithm places all connections into the same cluster. The reason behind this error has not been investigated further, but we register that this error produces parameters that may cause the algorithm to incorrectly label the clustering.

4.6 Discussion

In this section, we summarize and discuss the results and analyses presented in section 4.5. The ROC curves obtained in our experiments are all displayed in Figure 9, and illustrate the difference in accuracy between the applied indexes and strategies.

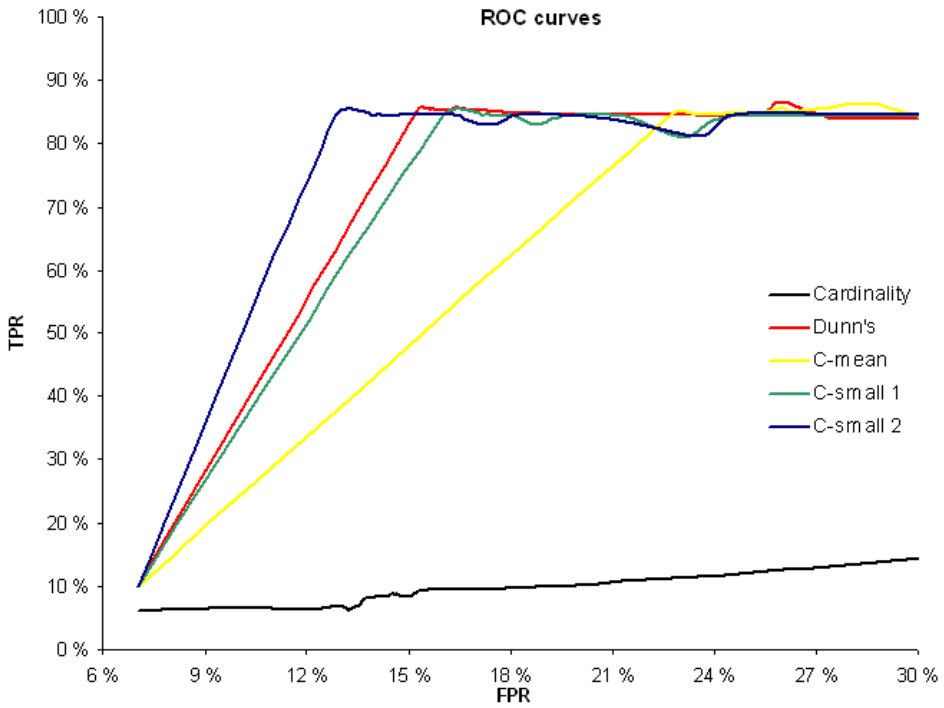

Figure 9: ROC curves from experiment 1-5, displaying the accuracy of the different algorithms and clustering quality indexes. C-small 1 corresponds to experiment 4 and C-small 2 to experiment 5.

The curves in Figure 9 show that the best true positive rate was achieved with the application of Dunn's index in Algorithm 1. We do, however, claim that the application of C-small yields the best overall system accuracy, when condition 4 in Algorithm 2 is considered (C-small 2 in Figure 9). The reason is that the false positive rate is considerably improved compared to the other indexes, while a good true positive rate is maintained. This is in accordance with our expectations regarding the system accuracy when the modified C-indexes are applied. However, the ROC curves also show that this expectation fails when the C-mean alternative for the modified C-index is applied.

The ROC curves also show that the application of all indexes in our labelling strategy outperforms the detection rate of the cardinality based labelling strategy on the KDD Cup '99 data set. This data set consists of a very high number of massive attacks. Time and resource constraints have limited us from testing the prototype on other data sets, which might provide more realistic environments than the KDD Cup '99. However, we have manually inspected the application of our labelling strategy in sections of the data set that do not consist of massive attacks. These inspections show that our labelling strategy produces almost exactly the same true and false positive rates as the cardinality based strategy in these sections. This is also reported by Petrovic et al. in [51, 52], where this labelling strategy, with the use of two other clustering quality indexes, is applied on data sets where the massive attacks are filtered out. These results therefore give a strong indication that our labelling strategy solves the problem of detecting massive attacks with cardinality based strategies.

We described our performance expectations for the system efficiency at the beginning of this chapter, and we assumed that the computation of the modified C-indexes is rather slow. The time complexity of the two indexes is analysed in Chapter 3. We do, however, mention the computation time of the two indexes in our experiments here to illustrate the difference in efficiency between the two indexes. The clustering and labelling of 490000 connections take approximately 20 minutes when the modified C-indexes are computed on a standard workstation PC. When Dunn's index is applied the computation time is, on the other hand, approximately 3.5 minutes on the same workstation. In both cases, approximately 2 minutes of the computation time is used to cluster the data. Even though the implementations of the clustering quality indexes are not by any means optimized, this shows that the modified C-indexes have much longer computation time than Dunn's index. It can be argued that a reduction of the number of observed connections at each iteration may improve the computation time, but probably at the cost of reduced detection accuracy. This is a trade-off the system administrator must take into consideration for the specific situation.

In Section 4.5.3, we observed that the application of C-mean produces more fluctuations in the ROC curve than the other indexes. The reason is the outliers that are present in the output from the clustering of the KDD Cup '99 data set. We also observe that the C-small alternative produces some fluctuations in the ROC curves. There are two reasons for this. The first reason, which also contributes to the fluctuations seen with C-mean, is that the C-index is very sensitive. This is because the modified C-indexes consider many distances between instances in the clustering, which is also the main reason for the improved accuracy our experiments have demonstrated. The other reason is that the smallest cluster only consists of a very few elements in some of the obtained clusterings. In these situations, the C-small index considers too few distances to achieve good

evaluations. This instability, which is only seen a few times, contributes to the problem of adjusting the system parameters, an issue we discuss closer below. However, we did not observe this problem to cause incorrect labelling of the KDD Cup '99 data set, when C-small was applied with the optimal configuration.

As mentioned in Section 4.5.2, it is reported that Dunn's index is vulnerable for noise, e.g. outliers, in the data set. We therefore expected that this instability would cause unstable evaluations, which would lead to unstable labelling and fluctuations in the ROC curves. According to Gunter et al. [22], the reason is that only two distances are considered, i.e. the minimum inter-cluster distance and the maximum intra-cluster distance. However, as we reported in Section 4.5.2, we have only seen a few examples of this. The reason is that the index does not produce unstable evaluations in the two-cluster case. In this situation there are only two intra-cluster distances and one inter-cluster distance to consider, and Dunn's index therefore provides stable clustering quality evaluations.

In our experimental work, we have observed that most of the errors produced by the system are caused by the clustering algorithm, regardless of the applied labelling algorithm. These errors can often be solved by the implementation of a more sophisticated and accurate clustering algorithm. The drawback is that a more sophisticated and accurate clustering algorithm often comes with the cost of high time complexity [27]. According to Xu et al. [60], there will always be a trade-off between the speed and accuracy of a clustering algorithm. It is therefore difficult to determine what clustering algorithm is best suited for the specific purpose.

For an IDS, the efficiency is as important as the accuracy, and this must be considered when the clustering algorithm is chosen. Real-time operation is necessary for an IDS analyst to act, i.e. perform countermeasures, upon observed attacks on the protected system. Poor IDS efficiency may lead to situations where the IDS must discard observed activities to achieve real-time operation. The danger is that these discarded activities are malicious. Because these activities are never investigated by the IDS, they must be considered as false negatives and this renders the IDS unreliable. We did see in our review of related work, see Section 2.3, that improvements of the K-means algorithm have been proposed for intrusion detection, which may provide the desired accuracy without too high efficiency penalty. These algorithms will, however, not always produce a two-cluster output, and more complex labelling algorithms must therefore be implemented to label these clusterings. On the other hand, the use of two clusters is only applied in our prototype to prove the concept and principles behind our labelling strategy, and this strategy is not limited to the two-cluster case.

We may expect that Dunn's index' vulnerability for noise increases by the number of clusters in the clustering, and that the application of Dunn's index will produce more unstable evaluations in a system that outputs an arbitrary number of clusters. This is, however, not the case, according to Halkidi et al. [23]. Another issue is that more sophisticated clustering algorithms handle noise much better than K-means with $K = 2$, and will, among other things, place outliers in separate clusters. This reduces the problem with noise in the clusterings and thereby the probability of unstable evaluations with Dunn's index. We therefore expect that the application of Dunn's index should yield proper evaluation of the clustering quality in these situations as well. The use of our proposed modified C-indexes for quality evaluations in such situations requires further research. Our expectations are, however, that the accuracy of C-small should remain

rather unaffected when the smallest clusters are of reasonable sizes, and that C-mean should achieve improved accuracy because of the improved noise handling.

Most of the few labelling errors caused directly by the labelling algorithms in our experiments can be corrected by adjusting one or more of the parameters in the algorithms. Finding the optimal configuration of these parameters is, however, very difficult. There is always a trade-off between the number of true and false positives, when these system parameters are configured. The reason is that by adjusting the parameters in order to detect one kind of attack, we may, and in most situations do, end up with a configuration that fails to detect other attacks. This is demonstrated with the ROC curves we have used to measure the system accuracy, which illustrates the relationship between the true and false positive rate at different configurations. In these curves, we see that by adjusting the system parameters we can achieve a higher detection rate, but at the expense of a higher false positive rate and vice versa.

The optimal configuration of an IDS is also very dependant on the observed data, and the environment surrounding the protected system therefore contributes to the problem of tuning the system parameters. The consequence is that a configuration that yields good accuracy in one environment may produce poor accuracy in another. Factors that must be taken into consideration when configuring the IDS involve issues such as: what assets must be protected, what kinds of attacks must be detected to protect these assets, which threats do these assets face, vulnerability analysis etc. In the end it is up to the system administrators to find the optimal IDS configuration for the given environment. The art of finding this configuration is then based on factors such as: system requirements, system knowledge and experience, and in many situations also the system administrator's gut feeling.

5 Conclusions

A major challenge in clustering based intrusion detection systems is to interpret and label the observed activities, especially the proper labelling of massive attacks. This thesis has investigated the application of Dunn's index and C-index in a labelling strategy for clustering based intrusion detection systems that handles this challenge. In this labelling strategy, Dunn's index and C-index are applied to control the observed activities for the presence of massive attacks. When the clustering quality indexes have established whether or not a massive attack is present, the labelling strategy uses the cluster diameters to interpret the nature of the observed activities.

In order to test and evaluate the application of these two cluster quality indexes, a prototype was developed to simulate a clustering based IDS on the KDD Cup '99 data set. This data set has seen some criticism regarding its quality, but it was chosen because it is a good source of massive attacks. Through the simulations, it has been demonstrated experimentally that this labelling strategy outperforms classical cardinality based labelling strategies in the presence of several massive attacks. The results presented in Chapter 4 are achieved with the parameters of the labelling algorithms fine tuned to fit the KDD Cup '99 data set. This may have lead to overestimated results, because of the characteristics of this data set. However, through manual inspection of the sections of the data set that do not consist of massive attacks, we have seen that the labelling accuracy of our labelling strategy is similar to cardinality based strategies in situations without massive attacks. The parameters in the labelling algorithm may also be used to tune the intrusion detection system to adapt to the specific characteristics of a given environment. We therefore conclude that our labelling strategy handles the limitation of the cardinality based labelling strategies regarding the detection of massive attacks.

In Section 1.4 we presented three research questions we aimed to answer in the course of this work. Here we summarize the answers to these questions, based on the analyses and discussions presented in Chapters 3 and 4:

1. *Can Dunn's index and C-index be applied in a labelling strategy for clustering based intrusion detection systems?*

   Our concern regarding the application of Dunn's index was that it has been reported in literature that the evaluation index is vulnerable to noise in the data set to be clustered. It has been shown through our experimental work that it is not a major problem in the two-cluster case. Dunn's index produces good and stable evaluations of the clustering quality in this situation, and is therefore suitable for our labelling strategy.

   The C-index cannot in its original form be applied in the labelling strategy, because it requires equal cluster sizes to produce stable evaluations. We have proposed two modifications of the C-index to handle this limitation: C-mean and C-small. Through our experimental work, it has been shown that the C-mean alternative only produces proper clustering quality evaluations at very low levels of noise in the clustered data. The application of the C-mean index is therefore not suitable, as noise, which is often

outliers caused by standalone attacks, must be expected in clustering based intrusion detection systems. C-small, on the other hand, handles noise much better and produced stable and accurate results in our simulations. The C-small alternative may therefore be successfully applied in our labelling strategy.

2. *Which combinations of the clustering quality indexes and clustering properties yield the best accuracy of the labelling strategy?*

   It is shown in Chapter 3 that the compactness of the obtained clusters depends on the nature of the observed activities and whether or not a massive attack is present in these activities. The cluster diameters can therefore be used to interpret the nature of the clusters. Our investigations show that the combination of a clustering quality index, which controls the clustering for the presence of a massive attack, and the cluster diameters yields the best labelling accuracy.

   In our theoretical research we discovered that the presence of so-called outliers in the clustering space could have a negative influence on the clustering quality evaluations when the C-small index is applied. As a solution to this problem, we have proposed to use the difference between the two partial indexes, used to assemble the modified C-index, to correctly evaluate the clustering in such cases. Our simulations show that the consideration of this clustering property increases the labelling accuracy when C-small is applied.

3. *What clustering quality index is best suited for labelling activity clusters, regarding accuracy and efficiency?*

   There is no unambiguous answer to this question. The application of C-small in the labelling strategy yields better accuracy than the applications of the other clustering quality indexes. However, this high accuracy comes at the cost of high time complexity. For this reason, the application of the C-small index is only best suited in situations where real-time operation is not necessary, e.g. forensic investigations of historical data. Real-time operation is important for most intrusion detection systems. The best overall performance, regarding both efficiency and accuracy, is then achieved with the application of Dunn's index.

   Compared to the Davies-Bouldin and Silhouette indexes, applied in the same labelling strategy by Petrovic et al. [50, 51, 52], the application of Dunn's index yields similar labelling accuracy as the Davies-Bouldin index. Both indexes have linear time complexity in the number of observed activities, which makes them evenly suited for intrusion detection. Both the modified C-index and the Silhouette index yields improved labelling accuracy, but at the cost of too high time complexity.

6 Further work

This thesis has studied the application of Dunn's index and C-index in a labelling strategy for clustering based IDSs. Time and resource constraints have limited our area of focus for this thesis. This section describes some interesting topics regarding the application of the clustering quality indexes that we feel deserve further research.

As mentioned on several occasions, the KDD Cup '99 data set consists of many massive attacks. Such a heavily attacked environment is not realistic for most networks other than military installations, from which the data set originates from. It would therefore be interesting to study the application of the labelling strategy on data sets that provide more realistic simulations. The problem is that very few labelled test data sets have been made publicly available. A solution is to filter out most of the massive attacks in the KDD Cup '99, but this filtration may again produce an unrealistic data set and this attack filtration must be performed with care.

In our experimental work, we emphasised that most labelling errors were caused by clustering errors from the K-means algorithm. It would therefore be very interesting to study the labelling algorithm when these errors are reduced to a minimum, e.g. by means of a more sophisticated and accurate clustering algorithm. The consequence is that the improved accuracy comes at the cost of increased time complexity. However, as the accuracy may be considerably increased, this may be a desirable trade-off.

A subsequent problem associated with the use of more sophisticated clustering algorithms is that these algorithms will output more than two clusters. Further research is therefore necessary to evolve the labelling strategy presented in this thesis to handle arbitrary numbers of clusters. Possible approaches involve merging of clusters into the two-cluster case or a few clusters, or more complex labelling algorithms for labelling an array of clusters. Another issue that requires further research is the application of our proposed modified C-indexes in other situations than the two-cluster case. As these indexes have been developed with the two-cluster situation in mind, they may show different behaviour in other situations.

There are also several topics regarding the application of the clustering quality indexes that may be interesting to study further. As discussed in Chapter 4, Dunn's index is vulnerable to noise in the clustering space. Pal et al. [48] have proposed three indexes based on Dunn's index that are more robust against noise. These indexes do not add considerable time complexity to the application of the index and it would therefore be interesting to investigate the application of these indexes in the labelling strategy. The major problem concerning the use of the proposed C-small index is the high time complexity. This may be solved by reducing the number of observed activities at the time, at the expense of decreased accuracy. It would, however, be more interesting to see whether it is possible to reduce the number of distances considered by the C-index, while remaining the good evaluation accuracy.

Bibliography

[1] S. Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *Information and System Security*, 3(3):186–205, 2000.

[2] R. Bace and P. Mell. Intrusion Detection Systems. *National Institute of Standards and Technology (NIST), Special Publication 800-31*, 2001.

[3] R. G. Bace. *Intrusion detection*. Macmillan Publishing Co., Inc., Indianapolis, IN, USA, 2000.

[4] E. L. Barse and E. Jonsson. Extracting attack manifestations to determine log data requirements for intrusion detection. In *ACSAC '04: Proceedings of the 20th Annual Computer Security Applications Conference*, pages 158–167, Washington, DC, USA, 2004. IEEE Computer Society.

[5] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[6] Matt Bishop. *Computer Security – Art and Science*. Addison Wesley, 2003.

[7] N. Bolshakova and F. Azuaje. Cluster validation techniques for genome expression data, 2003.

[8] T. Dalamagas, T. Cheng, K.-J. Winkel, and T. Sellis. A methodology for clustering xml documents by structure. *Inf. Syst.*, 31(3):187–228, 2006.

[9] D.L. Davies and D.W. Bouldin. A cluster separation measure. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.

[10] D. E. Denning. An Intrusion-Detection Model. *IEEE Trans. Software Eng.*, 13(2):222–232, 1987.

[11] J.C. Dunn. Well separated clusters and optimal fuzzy partitions. *J. Cybernetics*, 1974.

[12] J. P. Egan. *Signal Detection Theory and ROC Analysis*. Academic Press, San Diego, CA, 1975.

[13] C. Elkan. Results of the kdd'99 classifier learning. *SIGKDD Explor. Newsl.*, 1(2):63–64, 2000.

[14] E. Eskin. Anomaly detection over noisy data using learned probability distributions. In *Proc. 17th International Conf. on Machine Learning*, pages 255–262. Morgan Kaufmann, San Francisco, CA, 2000.

[15] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data, 2002.

[16] Dieter Gollmann. *Computer security*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[17] J. Gomez and D. Dasgupta. Evolving fuzzy rules for intrusion detection. In *Proceedings of the Third Annual IEEE Information Assurance Workshop 2002*, New Jersey.

[18] J. Gomez, F. Gonzalez, and D. Dagupta. An immuno-fuzzy approach to anomaly detection. In *Proceedings of The IEEE International Conference on Fuzzy Systems*, St. Louis, 2003.

[19] F. Gonzalez and D. Dasgupta. An imunogenetic technique to detect anomalies in network traffic. In *Gecco 2002: proceedings of the genetic and evolutionary computation conference*, pages 1081–1088, New York, 2002. Morgan Kaufmann Publishers.

[20] Y. Guan, A. Ghorbani, and N. Belacel. An unsupervised clustering algorithm for intrusion detection: Advances in artificial intelligence. In *16th Conference of the Canadian Society for Computational Studies of Intelligence*, pages 616–117, Ontario, Canada, 2003. Halifax, Springer-Verlag.

[21] Y. Guan, A. Ghorbani, and N. Belacel. *Y*-means: A clustering method for intrusion detection. In *Canadian Conference on Electrical and Computer Engineering*, Montreal, Quebec, Canada, 2003.

[22] S. Gunter and H. Bunke. Validation indices for graph clustering. *Pattern Recogn. Lett.*, 24(8):1107–1113, 2003.

[23] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

[24] R. Heady, G. Luger, A. Maccabe, and M. Servilla. The architecture of a Network Level Intrusion Detection System. Technical Report CS90–20, University of New Mexico, Agosto 1990.

[25] C. A. R. Hoare. Algorithm 64: Quicksort. *Commun. ACM*, 4(7):321, 1961.

[26] L. Hubert and J. Schultz. Quadratic assignment as a general data analysis strategy. *British Journal of Mathematical and Statistical Psychology*, 29:190–241, 1976.

[27] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[28] H. S. Javits and A. Valdes. The NIDES statistical component: Description and justification. Technical report, SRI International, March 1993.

[29] V. R. Vemuri K. Labib. Anomaly detection using s language framework: Clustering and visualization of intrusive attacks on computer systems. In *Fourth Conference on Security and Network Architectures, SAR'05*, Batz sur Mer, France, June 2005.

[30] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley, 1990.

[31] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood. Selecting features for intrusion detection: A feature relevance analysis on kdd 99. In *PST*, 2005.

[32] K. Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, MIT Department of Electrical Engineering and Computer Science, June 1999.

[33] N. Keshava. Distance metrics and band selection in hyperspectral processing with applications to material identification and spectral libraries, july 2004.

[34] D. Knuth. *The Art of Computer Programming*, chapter 5.2.4: Sorting by Merging. Addison-Wesley, 1998.

[35] W. E. Kuhnhauser. Root kits: an operating systems viewpoint. *SIGOPS Oper. Syst. Rev.*, 38(1):12–23, 2004.

[36] P. Laskov, P. Düssel, C. Schäfer, and K. Rieck. Learning intrusion detection: Supervised or unsupervised?. In *ICIAP*, pages 50–57, 2005.

[37] M. Last, B. Shapira, Y. Elovici, O. Zaafrany, and A. Kandel. Content-based methodology for anomaly detection on the web. In *AWIC*, pages 113–123, 2003.

[38] P. D. Leedy and J. E. Ormrod. *Practical Research: Planning and Design 8th edition*. Pearson, 2004.

[39] K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proc. 28th Australasian CS Conf.*, volume 38 of *CRPITV*, 2005.

[40] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. *Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation*. 2000.

[41] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[42] W. A. Martin. Sorting. *ACM Comput. Surv.*, 3(4):147–174, 1971.

[43] J. Mchugh. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. Inf. Syst. Secur.*, 3(4):262–294, 2000.

[44] P. Mell, V. Hu, R. Lippmann, J. Haines, and M. Zissman. An overview of issues in testing intrusion detection systems, 2003.

[45] S. H. Oh, J. S. Kang, Y. C. Byun, G. L. Park, and S. Y. Byun. Intrusion detection based on clustering a data stream. In *SERA '05: Proceedings of the Third ACIS Int'l Conference on Software Engineering Research, Management and Applications*, pages 220–227, Washington, DC, USA, 2005. IEEE Computer Society.

[46] S. H. Oh and W. S. Lee. Optimized clustering for anomaly intrusion detection. In *PAKDD*, pages 576–581, 2003.

[47] J. Oldmeadow, S. Ravinutala, and C. Leckie. Adaptive clustering for network intrusion detection. In *PAKDD*, pages 255–259, 2004.

[48] N. R. Pal and J. Biswas. Cluster validation using graph theoretic concepts. *Pattern Recognition*, 30(6):847–857, 1997.

[49] W. Panny and Helmut Prodinger. Bottom-up mergesort-a detailed analysis. *Algorithmica*, 14(4):340–354, 1995.

[50] S. Petrovic. A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters. In *Proceedings of the 11th Nordic Workshop of Secure IT Systems*, pages 53–64, 2006.

[51] S. Petrovic, G. Alvarez, A. Orfila, and J. Carbo. Labelling clusters in an intrusion detection system using a combination of clustering evaluation techniques. In *Proceedings of the 39th Hawaii International Conference on System Sciences, IEEE Computer Society Press*, page CDROM(8 pages). IEEE Computer Society Press, 2006.

[52] S. Petrovic, G. Alvarez, A. Orfila, and J. Carbo. Labelling ids clusters by means of the silhouette index. In *Proceedings of the IX Spanish Conference on Cryptography and Information Security, Barcelona, Spain*, pages pp. 760–772, 2006.

[53] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering, 2001.

[54] P. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987.

[55] P. Sneath and R. Sokal. Numerical taxonomy. *Freeman, London, UK*, 1973.

[56] S. Specht and R. Lee. Distributed denial of service: taxonomies of attacks, tools and countermeasures. In *Proceedings of the 2004 PDCS*, San Francisco, September 2004.

[57] W. R. Stevens. *TCP/IP Illustrated, Volume 1*. Addison-Wesley, 2000.

[58] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan. Kdd cup data set from the knowledge discovery and data mining tools competition. Available at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [Visited January 2007].

[59] Q. Wang and V. Megalooikonomou. A clustering algorithm for intrusion detection.

[60] R Xu and D. Wunsch II. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[61] N. Ye and X. Li. A scalable clustering technique for intrusion signature recognition. In *Preceedings of the 2001 IEEE Man Systems and Cybernetics Information Assurance Workshop*, United States Military Academy, West Point, NY, 2001.

[62] Y.-F. Zhang, Z.-Y. Xiong, and X.-Q. Wang. Distributed intrusion detection based on clustering. volume 4, pages 2379–2383 Vol. 4, 2005.

[63] S. Zhong, T. M. Khoshgoftaar, and N. Seliya. Clustering-based network intrusion detection. In *International Journal of Reliability, Quality, and Safety Engineering*, 2005.

A Samples of the prototype source code

Here we list samples of the source code used in the implementation of Dunn's index and C-index in the prototype. Appendix A.1 lists the implementation of Dunn's index, Appendix A.2 lists the implementation of the modified C-index and the relabelling algorithms are listed in Appendixes A.3 and A.4. The prototype is written in managed `C++` and we give a short description of the functions listed.

A.1 Implementation of Dunn's index

```
/*
DunnsIndex() computes and returns the Dunn's index of the clustering
*/
Double ClusteringEvaluator::DunnsIndex(){
  Double dunnsIndex, min ;
  Int32 numClust = clustering->GetNumClusters() ;
  Int32 i, j ;
     for(i=1;i<=numClust;i++){
       min = 9999999999999999999.0 ;
       for(j=1;j<=numClust;j++){
       if(i!=j){
                Int32 ni = clustering->GetCluster(i)->GetCardinality() ;
                Int32 nj = clustering->GetCluster(j)->GetCardinality() ;
                Double q,p,o ;
                if((ni!=0)&&(nj!=0)){
                        q = this->CentroidLinkage(i,j) ;
                        p = clustering->GetCluster(i)->CentroidDiameter() ;
                        o = clustering->GetCluster(j)->CentroidDiameter() ;
                        if(p > o){
                                q /= p ;
                        }else{
                                q /= o ;
                        }
                }else
                        q = 0.0 ;

                if(q<min)
                        min = q ;
        }
     }
  }
  dunnsIndex = min ;
  return dunnsIndex ;
}
```

The DunnsIndex() function in the ClusteringEvaluator class is called from the re-labelling function. It first checks whether there are empty clusters. If there are empty clusters, then there is no inter-cluster distance to measure in the two-cluster case and Dunn's index is set to zero. If there are two clusters with elements, then we divide the inter-cluster distance with the largest intra-cluster distance to compute the Dunn's index.

A.2 Implementation of the modified C-index

```
/*
ClusteringEvaluator::CIndex() computes the partial C-indexes
from the clusters in the clustering
*/
void ClusteringEvaluator::CIndex(){
Double S, Smin, Smax ;
Int32 numClust = clustering->GetNumClusters() ;
Int32 i,j ;
Int32 card[] = new Int32[numClust+1] ;
CIndexes = new Double[numClust+1] ;
S = 0.0 ;
Smin = 0.0 ;
Smax = 0.0 ;

    for(i=1;i<=numClust;i++){
        card[i] = clustering->GetCluster(i)->GetCardinality() ;
        CIndexes[i] = 0.0 ;
    }

    if(!EmptyCluster(1) && !EmptyCluster(2))
        clustering->GetClusterAll()->CreateArray(1000) ;

    for(i=1; i<=numClust; i++){
        if(!EmptyCluster(i))
                clustering->GetCluster(i)->CreateArray(card[i]) ;
                S = clustering->GetCluster(i)->GetSum() ;
                Smax = clustering->GetClusterAll()->GetMaxSum(card[i]) ;
                Smin = clustering->GetClusterAll()->GetMinSum(card[i]) ;

                CIndexes[i] = S - Smin ;
                Double temp = Smax - Smin ;
                CIndexes[i] /= temp ;

        }
    }
}
```

The CIndex() function in the ClusteringEvaluator class computes the partial indexes from both clusters in our two-cluster case. If there are empty clusters in the clustering the C-index is always zero, and we skip the computations. When both clusters consist of elements, the we first create a sorted array that consist of the $\frac{1000(1000-1)}{2}$ distances between all pairs in the whole clustering. This is done with the use of the GetClusterAll() function that returns a temporary cluster that consist of all elements in the whole clustering. We then use the CreateArray() function, which computes the distances between all elements in the cluster it is called from, to compute the distances between all pairs in the clustering. This array is later used to compute S_{max} and S_{min}. S is computed by calling the CreateArray() function from the cluster we compute the partial index from and adding up the distances between pairs within that cluster. The last step is to compute the C-index formula. We describe the CreateArray(), GetSum(), GetMaxSum() and GetMinSum() functions closer in the following:

```
/*
Cluster::CreateArray() Creates a sorted array of the distances
between all pairs in the cluster it is called from
*/
void Cluster::CreateArray(int c){
        int size = (c*(c-1))/2 ;
        a = new Double[size+1] ;
        int i,j,counter ;
        counter = 0 ;
        for(i=1; i<=c; i++){
                for(j=(i+1); j<=c; j++){
                        counter++ ;
                        a[counter] = vectors[i]->EuclidDistance(vectors[j]) ;
                }
        }
        mergeSort(0,counter-1) ;
}
```

a[] is the array stored in the cluster object.

```
/*
Cluster::GetSum() Returns the sum of distances between
all pairs in the cluster it is called from.
*/
Double Cluster::GetSum(){
        Double sum = 0.0 ;
        for(int i=1;i<=(card*(card-1)/2);i++)
                sum += a[i] ;
        return sum ;
}
```

```
/*
Cluster::GetMaxSum() returns the l (l = c*(c-1)/2) largest distances
between pairs in the cluster it is called from.
*/
Double Cluster::GetMaxSum(int c){
        Double max = 0.0 ;
        for(int i = ((card*(card-1)/2)-(c*(c-1)/2)); i <= (card*(card-1)/2); i++)
                max += a[i] ;
        return max ;
}
```

```
/*
Cluster::GetminSum() returns the l (l =c*(c-1)/2) smallest distances
between pairs in the cluster it is called from.
*/
Double Cluster::GetMinSum(int c){
        Double min = 0.0;
        for(int i = 1; i < (c*(c-1)/2); i++)
                min += a[i];
        return min;
}
```

A.3 Relabelling algorithm for Dunn's index

```
/*
LabelDunn() relabels the clustering according to Algorithm 1
*/
Int32 Clustering::LabelDunn(Double deltaD, Double deltaCD1, Double deltaCD2){
        Int32 relabel = 0 ;
        ClusteringEvaluator *ce = new ClusteringEvaluator(this) ;
        Double dunn = ce->DunnsIndex() ;
        Double cd1 = this->GetCluster(1)->CentroidDiameter() ;
        Double cd2 = this->GetCluster(2)->CentroidDiameter() ;

        if((dunn==0.0)&&(IsEmpty(2))){
                relabel = 1 ;
                this->ExchangeClusters(1,2) ;
        }
        else if((dunn<deltaD)&&(cd1>(cd2+deltaCD1))){
                relabel = 2 ;
                this->ExchangeClusters(1,2) ;
        }
        else if((dunn>deltaD)&&((cd1+deltaCD2)<cd2)){
                relabel = 3 ;
                this->ExchangeClusters(1,2) ;
        }

        return relabel ;
}
```

The LabelDunn() function in the Clustering class creates a new ClusteringEvaluator object, which is used to compute Dunn's as described in A.1. The relabelling procedure is explained in Section 3.3.

A.4 Relabelling algorithm for the modified C-index

```
/*
LabelCIndex() relabels the clustering according to Algorithm 2
*/
Int32 Clustering::LabelCIndex(Double deltaC, Double deltaCD1, Double deltaCD2,
 Double deltaDiff){
        int relabel = 0;
        Double CIndex, CIndex1, CIndex2;
        ClusteringEvaluator *ce = new ClusteringEvaluator(this);
        ce->CIndex();

        CIndex1 = ce->GetCIndex(1);
        CIndex2 = ce->GetCIndex(2);
        CIndex = CIndex1 + CIndex2;
        Double Cmean = CIndex/2;
        Double Cdiff = abs((CIndex1-CIndex2));
        Double cd1 = this->GetCluster(1)->CentroidDiameter() ;
        Double cd2 = this->GetCluster(2)->CentroidDiameter() ;

//      Double C = Cmean;       //If C-mean is applied

        Double C = CIndex1;     //If C-small is applied
        if(this->GetCluster(1)->GetCardinality() >
           this->GetCluster(2)->GetCardinality())
                C = CIndex2;

        if(Cdiff > deltaDiff)   //If C-small and condition 4 is used
                C = 1;

        if(C == 0.0 && IsEmpty(2)){
                relabel = 1;
                this->ExchangeClusters(1,2);
        }else if((C > deltaC) && (cd1>(cd2+deltaCD1))){
                relabel = 2;
                this->ExchangeClusters(1,2);
        }else if((C < deltaC) && ((cd1+deltaCD2)<cd2)){
                relabel = 3;
                this->ExchangeClusters(1,2);
        }
        return relabel;
}
```

The LabelCIndex() function in the Clustering class creates a new ClusteringEvaluator object, which is used to compute Dunn's as described in A.1. Then the two proposed alternatives for the modified C-index are computed. The rest of the relabelling procedure is explained in Section 3.3.

B Optimal system accuracy

The K-means algorithm produces a rather high error rate when $K = 2$. We cannot for that reason obtain perfect detection and false alarm rates. Since the KDD Cup '99 data set is labelled, it is rather easy to measure the optimal system accuracy with the use of this clustering algorithm.

To find the best possible false positive rate and true positive rate, we must check the output from the algorithm against the correct clustering. We can do this by finding the labelling that yields the minimum Hamming distance from the correct clustering, regardless of the labelling done by our algorithms. When we have found this optimal labelling, we count the number of false positives and false negatives, i.e. the number of normal activities misplaced in the malicious cluster and vice versa. Following this procedure for the entire data set we find 6704 false positives and 23608 false negatives, which give 395779 – 23608 = 372171 true positives. For the reduced (10 %) KDD Cup '99 database we have the following statistics:

Number of normal activities (FP+TN)	94221
Number of malicious activities (TP+FN)	395779
Number of activities	490000

Table 10: KDD Cup '99 (10% reduced) statistics

Using the definition of the false positive rate and true positive rate presented in Section 1.7, we get the best possible accuracy the system can obtain with the use of K-means:

$$FPR = \frac{FP}{FP + TN} = \frac{6704}{94221} = 7.12\%$$

$$TPR = \frac{TP}{TP + FN} = \frac{372171}{395779} = 94.04\%$$