Anomaly Detection Framework Using Rule Extraction for Efficient Intrusion Detection

Antti Juvonen*, Tuomo Sipola

Department of Mathematical Information Technology, University of Jyväskylä, Finland

Abstract

Huge datasets in cyber security, such as network traffic logs, can be analyzed using machine learning and data mining methods. However, the amount of collected data is increasing, which makes analysis more difficult. Many machine learning methods have not been designed for big datasets, and consequently are slow and difficult to understand. We address the issue of efficient network traffic classification by creating an intrusion detection framework that applies dimensionality reduction and conjunctive rule extraction. The system can perform unsupervised anomaly detection and use this information to create conjunctive rules that classify huge amounts of traffic in real time. We test the implemented system with the widely used KDD Cup 99 dataset and real-world network logs to confirm that the performance is satisfactory. This system is transparent and does not work like a black box, making it intuitive for domain experts, such as network administrators.

Keywords: intrusion detection, dimensionality reduction, cyber security, diffusion map, rule extraction

1. Introduction

Cyber security has become a very important topic in the past years as computer networks, services and systems face new threats from attackers, which has lead to increased interest towards these matters from companies and

*Corresponding author. Department of Mathematical Information Technology, P.O. Box 35 (Agora), FI-40014 University of Jyväskylä, Finland. Tel. +358 40 357 3875.

Email addresses: `antti.k.a.juvonen@jyu.fi` (Antti Juvonen), `tuomo.sipola@jyu.fi` (Tuomo Sipola) governments. Intrusion detection systems (IDS) are an important part of cyber security. They detect intrusions and abnormal behavior in networks or other systems producing big data. Practical environments are always different, and this affects the choice of the IDS and it's detection algorithms (Molina et al., 2012; Liao et al., 2013). These systems usually apply one of two detection principles: signature-based or anomaly-based (Scarfone and Mell, 2007). Signature-based approach means using manually created rules that detect intrusions, whereas anomaly-based systems try to profile normal behavior and detect abnormal action dynamically. It is also possible to combine these approaches to form a hybrid IDS, as we have done in this paper. This means creating signatures automatically and they can be periodically updated. In addition to the previous methodologies, stateful protocol analysis can be applied (Liao et al., 2013). This means finding unexpected sequences of commands. Moreover, anomaly detection systems could be used in combination with existing next-generation firewalls and other systems, so that the IDS benefits from the best properties of both approaches. Many different algorithms can be used for anomaly detection, e.g. self-organizing maps (Ramadas et al., 2003) and support vector machines (Tran et al., 2004). Another approach is to use genetic algorithms in intrusion detection context (Li et al., 2012; Goyal and Aggarwal, 2012).

Machine learning offers many benefits for intrusion detection (Chandola et al., 2009). However, it has limits that should be considered. Semantic gain, i.e. the practical meaning of the results, is usually more valuable in practical cases than marginal increases in performance accuracy (Sommer and Paxson, 2010). Anomaly detection should be used in combination with existing systems to bring added value. One problem with machine learning algorithms used for anomaly detection is the fact that many of them work like a black box from the end user perspective. It is not an easy task to know how the algorithm internally functions, and because of this, companies have difficulties deploying these systems. To overcome this problem, rule extraction algorithms have been proposed (Craven and Shavlik, 1994). These algorithms aim to create rules that replicate or approximate classification results. The main benefit of deploying rule-based systems is that they are fast and they might reveal comprehensible information to the users better than black box algorithms. However, it seems that many rule extraction algorithms depend on neural networks (Huysmans et al., 2006). Furthermore, rule extraction itself is usually a supervised learning task and needs previously created classification and labeling information in order to work.

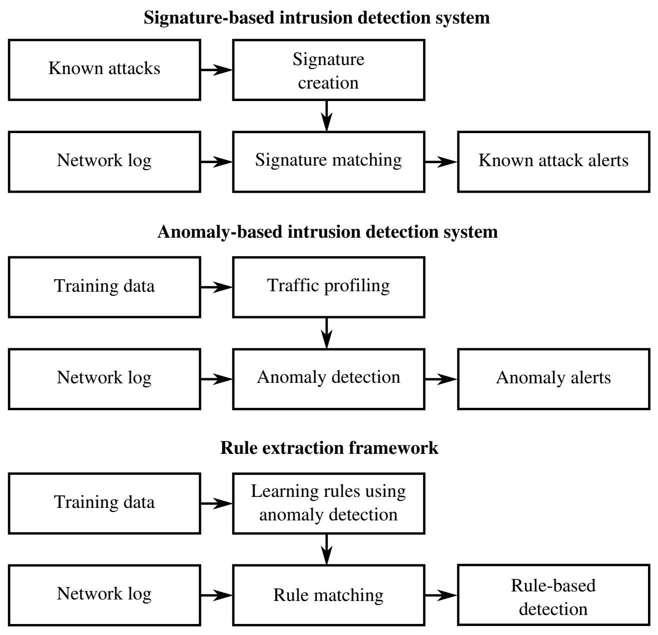

Figure 1: Different IDS principles.

We propose an intrusion detection framework that uses diffusion map methodology for dimensionality reduction and clustering to automatically label network traffic data. Figure 1 shows the different IDS principles, and how the rule extraction framework relates to them. Our system follows the three overall stages of network IDS: data parametrization, training stage and detection stage (García-Teodoro et al., 2009). The learning phase to label the traffic is unsupervised. Subsequently, the classification information is forwarded to a rule extraction algorithm that creates conjunctive rules. These automatically generated rules are used to classify new incoming traffic data. The rules can also be analyzed by a domain expert in order to acquire new information about the nature of the data. The presented framework combines knowledge-based expert system and machine learning-based clustering approaches (García-Teodoro et al., 2009). Consequently, the black box nature of some of the existing algorithms is avoided. The framework does not need preceding information about the intrusions that are present in the training data. In addition, the structure of the system is dynamic and individual algorithms can be changed if necessary. Low throughput and high cost, as well as lack of appropriate metrics and assessment methodologies have been identified as common problems in anomaly-based IDSes (García-Teodoro et al., 2009).

This paper contributes to methodology and practical analysis in the field of network security. Firstly, the framework addresses the above issues by generating rules that can classify incoming traffic with low computational cost. Secondly, this research uses a well-known public dataset and appropriate metrics to assess the performance. Finally, real-world network data is used to further investigate the effectiveness of the framework in a practical situation. The main contribution of this paper is finalizing and detailed description of the framework (Juvonen and Sipola, 2013) and tests with more varied data that show better detection rates than before.

The paper is structured as follows. First, we go through related reseach concerning dimensionality reduction methodologies and rule extraction. In the methodology section the overall system, all of the used algorithms and performance metrics are described. Then, we introduce our experimental cases and present the obtained results. Finally, we discuss the benefits and limitations of the system as well as future research directions.

2. Related work

This section briefly discusses research that is related to the methods used in this article. There are five areas that are covered: dimensionality reduction, anomaly detection, rule extraction, big data approaches and the earlier frameworks designed by the authors. These points have contributed to the design of the current framework.

Firstly, dimensionality reduction has been widely researched in the intrusion detection context. Perhaps the most well-known method is principal component analysis (PCA) (Jolliffe, 2005), which has been used in network anomaly detection (Ringberg et al., 2007; Callegari et al., 2011). However, it has some problems, such as the fact that it cannot handle non-linear data. In general, manifold learning approaches try to learn the structure of the data, retaining some meaningful qualities of the data mining problem. The point is to contain most of the information in the data using fewer dimensions. The goals in such a setting are understanding and classification of data and generalization for use with new data (Lee and Verleysen, 2007). Several manifold learning methods have been used for intrusion detection, including Isomap and locally linear embedding (LLE) (Zheng et al., 2009a,b; Yuancheng et al., 2010). This paper uses the diffusion map manifold learning method, which is a non-linear dimensionality reduction method (Coifman and Lafon, 2006). Diffusion map methodology has been used for network traffic classification and SQL intrusion detection (David, 2009; David et al., 2010; David and Averbuch, 2011). We have previously used it for anomaly detection from network logs (Sipola et al., 2011, 2012). This framework is enhanced by using clustering to detect multiple behaviors in the data (Juvonen and Sipola, 2012). In these experiments, the diffusion map algorithm acts like a black box, which is a drawback that makes the system hard to understand for people who are not familiar with data mining technologies.

Secondly, related research on anomaly detection in intrusion detection context is explored. Some of the common general approaches to intrusion detection are statistical methods, machine learning and data mining (Patcha and Park, 2007). Statistical methods do not require prior knowledge about attacks, but they need a certain statistical distribution, which is not always the case. Machine learning based methods (e.g., Bayesian networks and principal component analysis (PCA)) aim to learn from the behavior and improve the performance over time. Advantages and drawbacks depend on the used algorithm, e.g., PCA cannot handle non-linear data, which is why we use diffusion map methodology for dimensionality reduction. Finally, data mining methods such as genetic algorithms and artificial neural networks attempt to find patterns and deviations automatically from the data. It is also possible to use some kind of hybrid system. In addition, anomaly detection techniques can be divided into classification-based, nearest neighbor-based and clustering-based methods (Chandola et al., 2009). Also, artificial immune systems have been used extensively in intrusion detection (Kim et al., 2007). Ensemble systems have also been successful in the area Mukkamala et al. (2005). Using computational and artificial intelligence methods makes it possible to create adaptive, fault tolerant and fast systems (Wu and Banzhaf, 2010; Liao et al., 2013). Our system combines many of the previously mentioned approaches such as clustering-based anomaly detection and dimensionality reduction. The system, its differences to existing methodologies and our contributions are explained in more detail at the end of this section.

Thirdly, rule extraction is considered. In order to understand the black-box nature of non-linear classifiers, rule extraction methods can create sets of rules that describe the behavior of such systems in a more understandable manner (Martens et al., 2008). Our implementation of the presented framework is based on the conjunctive rule extraction algorithm (Craven and Shavlik, 1994). However, the overall framework does not depend on any specific clustering or classification algorithm. The choice of the used classification method influences the selection of other algorithms in the framework, because of varying performance and robustness. The approach presented in this research resembles spectral clustering frameworks and can be though as a special case of them (Bach and Jordan, 2006; von Luxburg, 2007). There are many algorithms that extract rules from trained neural networks, such as TREPAN (Craven and Shavlik, 1996) and Re-RX (Setiono et al., 2008). These algorithms depend on the neural network architecture and the associated weights. Another common approach is to use support vector machines as a basis for rule extraction (Núñez et al., 2002; Barakat and Diederich, 2004; Barakat and Bradley, 2010). One example for generating signatures for detecting polymorphic worms is Polygraph, which uses disjoint content substrings to identify them (Newsome et al., 2005). Regular expressions generated with a supervised domain expert dataset have also been used to screen unwanted traffic (Prasse et al., 2012). In addition, many swarm intelligence algorithms can be used for different intrusion detection applications. For example, ant colony optimization (ACO) can be utilized to detect intrusions, detect the origin of an attack and also induction of classification rules (Kolias et al., 2011). The field of swarm intelligence contains many other algorithms that are useful for IDS purposes, such as particle swarm optimization (PSO) and ant colony clustering (ACC).

Next, many of the above mentioned algorithms can be scaled up for big data applications using various parallel and distributed approaches (Bekkerman et al., 2012). For example, the map-reduce framework has proved itself as a feasible method for machine learning (Chu et al., 2006). The k-means algorithm used in our system can also be parallelized using map-reduce (Zhao et al., 2009). Moreover, the traditional intrusion detection methods can be sped up using graphics processors (Vasiliadis et al., 2008).

Finally, the authors have already proposed a system for extracting conjunctive rules from HTTP network log from real-life web servers (Juvonen and Sipola, 2013). The data mining approach to network security is similar to the methodology of this paper. We now extend and improve this framework for different kinds of data. Our framework works in an unsupervised manner and does not depend on the selection of the used classification algorithm.

The system proposed and tested in this paper combines many of the approaches mentioned previously in this section. The dimensionality reduction phase could have been done using principal component analysis, but it some-

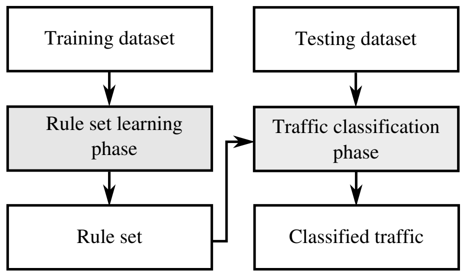

Figure 2: Block diagram of the overall approach.

times fails with non-linear data. We have learned through experiments that diffusion map sometimes gives more accurate results. In addition, clustering based anomaly detection has also been used previously in the literature. However, our approach is different because we do not cluster the original data, but the data points in low-dimensional space after diffusion map phase. The diffusion map already helps to separate the points, reducing the error that k-means clustering might introduce. Finally, because the diffusion map is not computationally very efficient, we use conjunctive rule extraction to approximate the clustering results and classify data. This way we aim to combine accurate results and feasible computational speed.

3. Methodology

Our system is divided into two phases: training phase and testing phase. Training consists of preprocessing and ruleset learning, and as an end result it produces a ruleset that can be used for traffic classification. Testing phase uses the created rules to classify testing data. Performance of the testing phase is also measured. The overall approach is described in Figure 2.

The training phase consists of the following steps:

- Training data selection
- Feature extraction
  - Discretization of continuous data (binning)
  - Binarization
- Unsupervised learning

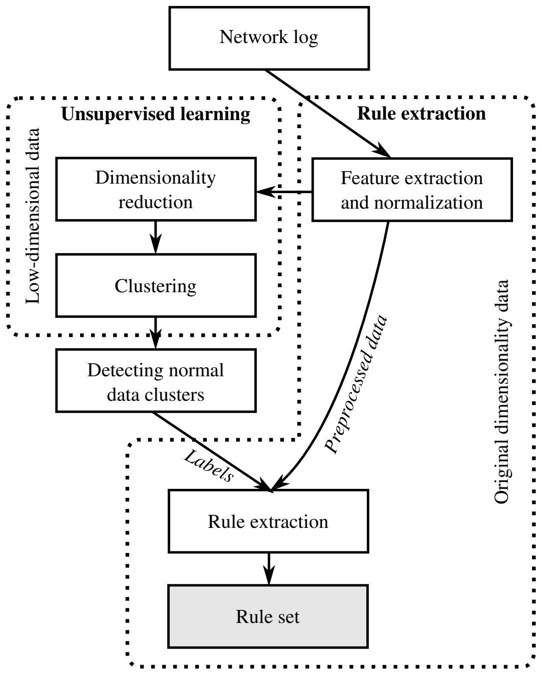

Figure 3: Block diagram of the ruleset learning process.

  - Dimensionality reduction
  - Clustering

- Detecting normal traffic
- Rule extraction

Figure 3 shows this ruleset learning phase in detail. Previously unknown data (network log) is used as an input, and this data is automatically labeled in an unsupervised manner. However, detecting the normal data cluster might need expert input. This classification information can then be used to create the ruleset, which tries to simplify the process of dimensionality reduction and clustering to human-readable rules.

Testing, or ruleset matching phase can be summarized as follows:

- Test data feature extraction
- Rule matching

This phase preprocesses new data and uses the rules created in the learning phase to classify the testing data as normal or intrusive. If needed, classification into more than two classes is also possible if this is taken into account in the training phase. The new traffic that the system identifies as normal is naturally not flagged. Traffic corresponding to intrusion rules are flagged as attacks. This flagging resembles misuse detection. In addition, traffic that does not match any rules, is flagged as unknown anomalies.

3.1. Dimensionality reduction using diffusion map

Dimensionality of the feature space is reduced using diffusion maps. This training produces a low-dimensional model of the data, which facilitates clustering and identification of internal structure of the data. Diffusion maps are useful in finding non-linear dependencies. At the same time the diffusion distances in the initial feature space correspond proportionally to the Euclidean distances in the low-dimensional space (Coifman et al., 2005; Coifman and Lafon, 2006; Nadler et al., 2006).

Each data point is represented by a feature vector $x_i \in \mathbb{R}^n$. The whole dataset $X = \{x_1,\ x_2,\ x_3,\ \dots\ x_N\}$ is a collection of these feature vectors. To perform the diffusion map, an affinity matrix

$$W(x_i, x_j) = \exp\left(\frac{-||x_i - x_j||^2}{\epsilon}\right)$$

describing the similarity between the data points is calculated. Here affinity is defined using the Gaussian kernel. The parameter ϵ, which is responsible for the neighborhood size, is selected from the optimal region in the weight matrix sum

$$L = \sum_{i=1}^{N} \sum_{j=1}^{N} W_{i,j},$$

which can be plotted on a logarithmic scale for the identification of the middle region (Coifman et al., 2008), while ϵ is changed.

The row sums of W are collected to the diagonal of matrix D. This matrix is used to normalize W in order to create transition matrix

$$P = D^{-1}W.$$

This transition matrix is symmetrized as

$$\tilde{P} = D^{\frac{1}{2}} P D^{-\frac{1}{2}} = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}.$$

The singular value decomposition (SVD) of $\tilde{P}$ finds the eigenvectors v_k and eigenvalues λ_k, which can be used to construct low-dimensional coordinates for the data points. Each data point in the original data gets corresponding diffusion map coordinates:

$$x_i \to [\lambda_1 v_1(x_i),\ \lambda_2 v_2(x_i) \dots\ \lambda_d v_d(x_i)].$$

The first eigenvectors retain most of the information in the data, which is why the later eigenvectors are left out. Some information is lost but the error is bounded and lower dimensionality facilitates clustering.

3.2. Clustering

The k-means method groups the data points into clusters. The algorithm is widely used in data mining and its implementation is simple. It finds the centroid point of clusters and then assigns each point to the nearest cluster centroid. This process is iterated until the clustering does not change after updating centroids. The algorithm and use cases are described in more detail in literature (Jain and Dubes, 1988; Tan et al., 2006; Jain, 2010). We chose to use k-means clustering together with diffusion map because its use is justified in the literature (Lafon and Lee, 2006).

In order to use k-means, the number of clusters needs to be determined. The clustering is repeated many times and the quality of the obtained clusters is measured. For ith data point, silhouette

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

expresses the quality of the clustering. Here $a(i)$ is the average dissimilarity of the point to all the other points in the same cluster, and $b(i)$ is the smallest of the average dissimilarities to all the other clusters. The larger the silhouette value, the better the clustering is for the data point. The average of the silhouette values for all data points is used to evaluate the quality of clustering. These qualities are used to compare different numbers of clusters (Rousseeuw, 1987).

3.3. Rule extraction

A rule defines a number of easy-to-understand conditions for the features in order to classify the data points according to those conditions. A ruleset consists of many rules and replicates a certain classification result. In an ideal situation a rule should be understandable by network experts without the effort of knowing the underlying classifiers. Rules correspond to data features, and are useful as such for finding the root cause of intrusions or other anomalous behavior.

It is not feasible to find all the possible rules because they span such a huge space. In an optimal setting all the rules should be checked, but in practice a sub-optimal but efficient method is used. Such rule extraction systems have been used to learn the information provided by, e.g., neural networks (Craven and Shavlik, 1994; Ryman-Tubb and d'Avila Garcez, 2010) or support vector machines (Núñez et al., 2002; Barakat and Diederich, 2004; Barakat and Bradley, 2010).

There are two main approaches to rule extraction. Firstly, *decompositional* rule extraction algorithms exploit the internal mathematical or algorithmic features of the underlying data mining method (d'Avila Garcez et al., 2001). Secondly, *pedagogical* algorithms do not depend on the method used (Craven and Shavlik, 1994). They only take into account the input data and classification result. We take the pedagogical approach, because this makes it possible to use different kinds of algorithms for data classification.

A conjunctive rule is a logical expression that combines logical symbols using the conjunction (i.e. AND) operation. Such a symbol states whether the value of a binary feature must be true or false. The absence of the symbol means that the feature can be anything. Note that in this research a binary feature means a truth value either belonging to a symbolic category, or a truth value that tells in which bin a continuous value belongs to.

Let us consider an example where we have five features a, b, c, d, e. This means that the feature matrix contains five columns corresponding to each binary feature. Each data point corresponds to a row in the matrix. Here is an example ruleset containing three rules classifying data into two separate classes c_1 and c_2:

$$\begin{aligned} r_1 =& \neg a && \text{for class } c_1, \\ r_2 =& a \wedge b \wedge c \wedge \neg d \wedge e && \text{for class } c_1, \\ r_3 =& a \wedge b \wedge \neg c && \text{for class } c_2. \end{aligned}$$

In this case, rule r_1 means that if the value of the feature a is false for a single data point, it belongs to class c_1. The values for the rest of features do not matter. Other rules work in a similar way: rule r_3 states that a and b have to be true while c must be false. If this rule r_3 holds true, the point is classified to class c_2.

In our actual implementation the truth values are converted to 1 and -1, and the values that do not matter are expressed as 0. Therefore, the rule r_1 can be expressed using vector $\begin{pmatrix}-1 & 0 & 0 & 0 & 0\end{pmatrix}$. This approach makes rule matching easy from the computational perspective.

The conjunctive rule algorithm (3.1) creates a ruleset for the training data. It creates new rules until the whole dataset is covered without misclassification (Craven and Shavlik, 1994). If the value of a symbol in a rule does not affect the classification result using the rules, the symbol can be omitted. The final rules contain only the symbols that are essential for the classification. The rules generated using the algorithm can be used for simple matching in the testing phase.

Algorithm 3.1: Conjunctive rule extraction (Craven and Shavlik, 1994).

```
Input: data points $E$, classes $C$
Output: rules $R_c$ that cover $E$ with classification $C$
  repeat
    $e$ := get new training observation from $E$
    $c$ := get the classification of $e$ from $C$
    if $e$ not covered by the rules $R_c$ then
      $r$ := use $e$ as basis for new rule $r$
      for all symbols $s_i$ in $r$ do
        $r' = r$ with symbol $s_i$ dropped
        if all instances covered by $r$ are of the same
        class as $e$ then
          $r := r'$
        end if
      end for
      add rule $r$ to the ruleset $R_c$
    end if
  until all training data analyzed
```

Table 1: Confusion matrix.

		predicted	
		normal	attack
actual	normal	true negative (tn)	false positive (fp)
	attack	false negative (fn)	true positive (tp)

3.4. Performance metrics

After classifying data points as normal or intrusive, and running the testing phase, the following performance metrics are calculated. Table 1 shows the confusion matrix and some abbreviations used in this research. Note that correct alarms of intrusions are regarded as true positive identifications.

Sensitivity or true positive rate (TPR) tells how well intrusive data points are identified:

$$sensitivity = \frac{tp}{tp + fn}.$$

False positive rate (FPR) is calculated in a similar way. False positives are data points that are classified as attacks but are actually normal. In intrusion detection context this is often called false alarm rate:

$$fpr = \frac{fp}{fp + tn}.$$

Specificity or true negative rate (TNR) explains how well normal data points are found:

$$specificity = \frac{tn}{fp + tn} = 1 - fpr.$$

Accuracy tells how many correct results there are compared to the whole data:

$$accuracy = \frac{tp + tn}{tp + fp + fn + tn}.$$

Precision explains the proportion of true positive classifications in all positive results, i.e. how many alarms were correct and not false alarms:

$$precision = \frac{tp}{tp + fp}.$$

The previous metrics are extensively used but cannot alone describe the results in a satisfactory way. For this reason, we also use Matthews correlation coefficient (Baldi et al., 2000; Brown and Davis, 2006; Fawcett, 2006). It can measure the quality of binary classification and is regarded as a good metric for measuring the overall classification performance:

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}}.$$

4. Iris data: an illustrative example

In this section a toy problem is presented to clarify certain concepts in the framework. The rule extraction algorithm described in subsection 3.3 is illustrated here using a practical example. In machine learning community, the IRIS dataset (Frank and Asuncion, 2010) is commonly used to illustrate the algorithms in a simple case. We use this well-known dataset to demonstrate the rule generation process and what kind of information is produced as a result. The data contains 150 samples from 3 different species of flowers. There are four different measurements made from each flower: sepal length and width as well as petal length and width. All these features are continuous. The task is to cluster the data and find corresponding rules using our framework.

Feature generation is performed by first dividing all 4 measurements into 3 bins. Larger number of bins is possible but in this case it was not necessary. The binning process works in the following way. For this dataset, the minimum value of petal width measurement is 0.1 cm and maximum value 2.5 cm. For this feature, the boundaries of the bins are as follows:

$$\begin{array}{ccccccccc} & | & Bin1 & | & Bin2 & | & Bin3 & | & \\ \hline & 0.1 & & 0.9 & & 1.7 & & 2.5 & \end{array}$$

Other features are binned using the same principle. After binning, the values are binarized from multi-category features to single-category binary features for rule extraction. Since 3 bins are used for each of the 4 features, we get 12 binary columns in the processed matrix. Consequently, the final feature matrix is of size 150×12. These 12 binary values also correspond to the rules, so that each rule has 12 elements.

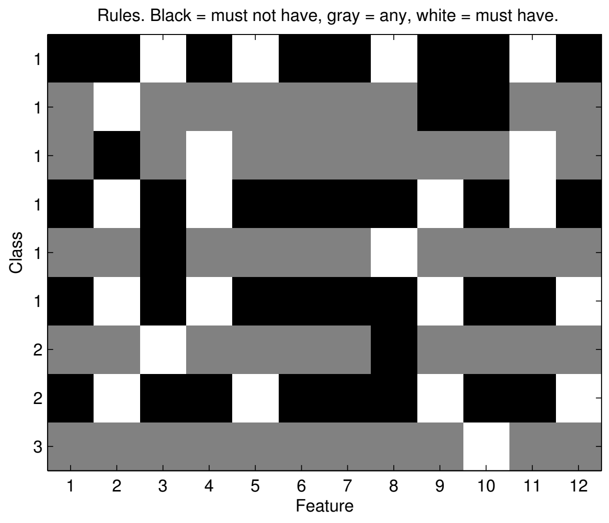

Figure 4: Rules generated from clustered IRIS data. Each line represents one rule. Note that classes 1, 2 and 3 correspond to the three species.

The data is clustered into 3 groups using the k-means clustering algorithm. There is no dimensionality reduction step beforehand because 12-dimensional feature space is still rather low-dimensional. The clustering is used to label flower samples using 3 labels. Subsequently, the rules are extracted.

The resulting ruleset can be seen in Figure 4. There are 9 rules in total. There is only one rule generated for class 3 (*setosa* species). The rule states that if the fourth measurement (petal width) belongs to bin 1, the flower's species is *setosa*. Bin 1 means that the petal width must be between 0.1 and 0.9 cm, as demonstrated above. The rules with mostly 'must not have' (i.e. black) symbols are usually created for single data points, which might be outliers. This is an example of how information can be gained from the rules. We can see that the classification process no longer works like a black box, but is based on human-readable information.

5. KDD Cup 99 data

For this experiment we used KDD Cup 99 10% data (Frank and Asuncion, 2010). It has some known limitations (Mahoney and Chan, 2003; Sabhnani and Serpen, 2004; Tavallaee et al., 2009) stemming from its source, the likewise problematic DARPA dataset (McHugh, 2000), but it is publicly available and labeled, which makes it easy for researchers to use the data and compare results (Davis and Clark, 2011). A serious concern is the unrealistic distribution of intrusions. Therefore, it is common to use the smaller 10% dataset or a custom version of it (Kolias et al., 2011). The dataset was created by gathering data from a LAN that was targeted with multiple attacks. It contains 494,021 traffic samples and each sample includes 41 features. The features include both continuous and symbolic values. The attacks fall into four general categories: denial-of-service, remote-to-local, user-to-root and probing. There are 24 attack categories in total in the training data. Data features include things such as duration of the connection, protocol type, network service, transferred bytes and failed login attempts. Even though the actual labels (normal or intrusive) are known, this information is not used in the rule creation phase. We use unsupervised learning for the framework, and only use the actual labels for getting the performance metrics and evaluating the results.

5.1. Feature extraction

The data that we use for testing and evaluation contains both continuous and discrete values. This must be taken into account in the feature extraction phase. In network traffic context, features such as the number of bytes transferred during a connection are continuous, and, e.g., used protocol type in contrast is a discrete valued feature.

During the preprocessing phase, continuous feature values must be transformed into discrete ones for the used rule extraction algorithm. Symbolic discrete values can be left alone at this stage. Discretization is performed using binning. We do this by acquiring the maximum and minimum values of each continuous variable, and dividing the measurements into n bins using equal intervals. The number of bins can be chosen according to the situation, e.g. $n = 3$ or $n = 10$. There is a practical example in section 4. Let us now present another example, where we assume that we have 3 bins for each variable and there are 3 different variables ($f1, f2, f3$) and 3 data points (rows). The number of the chosen bin for each measurement is placed in a feature matrix:

f1	**f2**	**f3**
1	2	1
3	2	3
2	1	1

This kind of matrix is not yet usable for rule extraction. In the final preprocessing phase, we binarize the matrix. This gives us a matrix that uses the following format:

f1=1	**f1=2**	**f1=3**	**f2=1**	**f2=2**	**f3=1**	**f3=3**
1	0	0	0	1	1	0
0	0	1	0	1	0	1
0	1	0	1	0	1	0

For example, the first column f1=1 means that if the chosen bin for the first variable is 1 on a certain row, the binary value in this binarized matrix will be 1, otherwise 0. In other words, a column tells whether a specific variable measurement belongs to a certain bin or not. This way we get nominal values for all the features. Columns that would be all zeros can be omitted. Now the matrix is in a format that can be directly used with the rule extraction algorithm. In addition, the rule extraction requires labeling information. The process of acquiring this is explained in the following subsections.

5.2. Random sampling from the 10% KDD set

For initial testing to showcase the feasibility of the approach, 5,000 data lines are randomly selected for testing and 5,000 for training, totaling 10,000 lines. The purpose of this initial experiment is to see if a rather limited training sample can represent the whole data. The feature matrix is binned and binarized as described previously. After this, dimensionality is reduced using a diffusion map.

In order to perform the dimensionality reduction in an optimal way, the value for parameter ϵ must be determined. Selecting ϵ for the diffusion map is performed by selecting the value from the region between the extremes of a log-log plot, as seen in Figure 5. The quality L is calculated by summing the weight matrix, as explained in subsection 3.1. For each value of ϵ, the sum is calculated using 200 randomly selected samples from the training data.

The eigenvalues λ_k from the diffusion map method are presented in Figure 6. The first eigenvalue is always 1 and is therefore ignored. Next three are used in the analysis, since they seem to cover most of the data. The eigengap is visible after them. The rest of the eigenvalues contain little information and are discarded. The underlying assumption of the analysis is that the first few eigenvalues and eigenvectors contain the most useful clustering information.

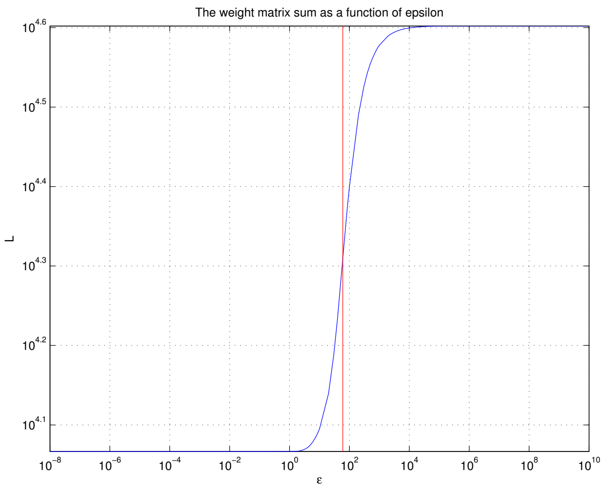

Figure 5: Finding the optimal region for epsilon selection for the diffusion map algorithm.

After dimensionality reduction the data is clustered using k-means algorithm. This step is believed to separate the normal data points from the attacks. To evaluate the best number of clusters, clustering process is repeated with different number of clusters, from 2 to 20. Figure 7 shows the quality of clustering with the average silhouette metric. Out of the silhouette values, the highest one should be selected. Based on this metric, the data is divided into 7 clusters.

Figure 8 shows the obtained clusters after dimensionality reduction and k-means clustering. The normal data is assumed to lie in cluster number 4, and all the others are regarded as attacks for labeling purposes. Normally we would assume that the largest cluster is the one containing normal data points, meaning that whole process could be completed automatically. However, due to the fact that KDD Cup 99 dataset unnaturally contains more intrusions than actual normal traffic, we had to choose the normal cluster using manual inspection.

The clustering is assumed to represent the classification of the training dataset. This information is used to create labeling, which is needed for rule creation phase. This is an example of how the rule extraction algorithm can be used in an unsupervised manner, even though it is in principle based on supervised learning. The conjunctive rules for this dataset are illustrated in Figure 9. There are 18 rules for the normal class and 15 rules for the attack class, 33 in total. This shows that the data can be represented and classified

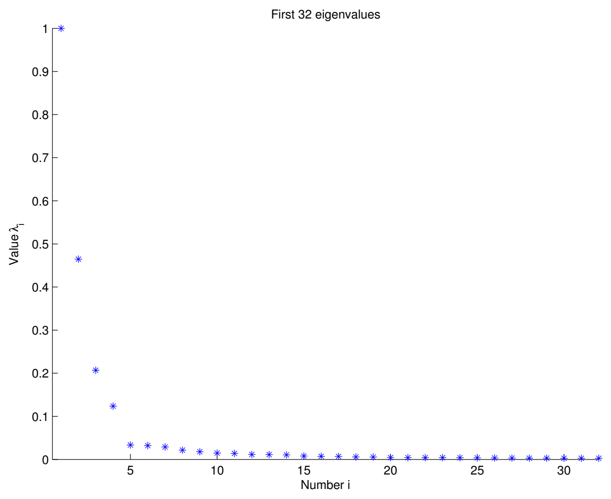

Figure 6: Eigenvalues of the diffusion map algorithm. The eigengap is visible between 4th and 5th eigenvalue.

Table 2: Performance metrics for the 5,000 testing data lines.

Metric	**Value %**
Sensitivity (TPR)	98.44
FPR	1.25
Specificity (TNR)	98.75
Accuracy	98.50
Precision	99.70
Matthews corr. coef.	95.31

by a relatively small and simple set of rules.

Table 2 shows the performance metrics for the 5,000 training lines setup. The confusion matrix is shown in Table 3. From these it can be seen that we get a good intrusion detection accuracy with a relatively low number of false alarms. It is important to note that Matthews correlation coefficient usually gives lower values than other commonly used metrics. Using the created rules to classify new incoming traffic is very fast, which is the main benefit of this method. Furthermore, the rules reveal comprehensible information about normal and intrusive data points.

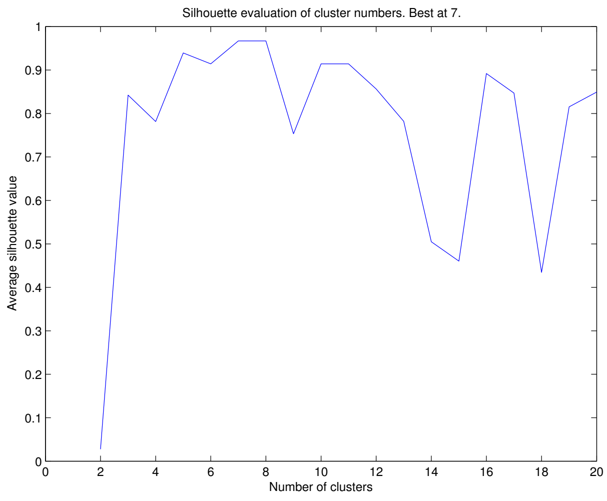

Figure 7: Cluster number quality measured using silhouette.

Table 3: Confusion matrix for the 5,000 testing data lines.

		predicted	
		normal	attack
actual	normal	947	12
	attack	63	3,978

5.3. The whole 10% KDD set

For a more comprehensive experiment we use the whole KDD Cup 99 10% dataset. A randomly selected subset of 5% (24,701 data points) is used for training phase, leaving 469,320 data points for testing. The goal is to use small training set size compared to the testing set to alleviate the scaling of the diffusion map method. Data preprocessing, binning and binarization as well as diffusion map dimensionality reduction and clustering are done in the same way as described in the previous case. As earlier, the normal clusters had to be identified using manual inspection. Rule extraction is again performed using the conjunctive rule algorithm.

Table 4 shows the performance measures for the testing data. From the

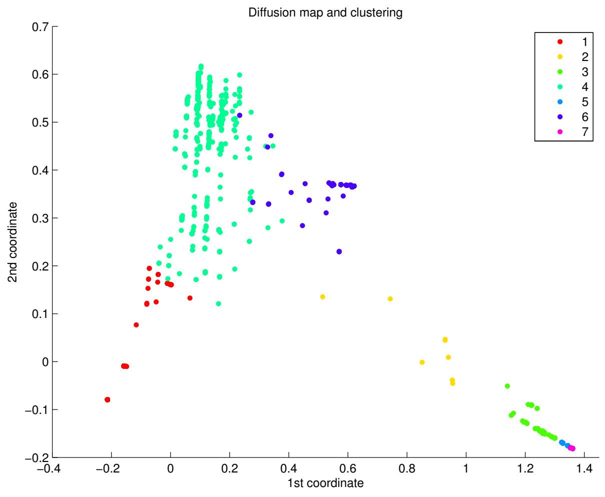

Figure 8: Found clusters mapped to the two first coordinates in the low-dimensional space.

Table 4: Performance metrics for the whole 10% dataset.

Metric	**Value %**
Sensitivity (TPR)	98.46
FPR	5.59
Specificity (TNR)	94.41
Accuracy	97.67
Precision	98.63
Matthews corr. coef.	92.64

table we can see that the overall performance is not as good as in the previous case. However, true positive rate is slightly better. The most important metric to look at is the Matthews correlation coefficient. The whole dataset has more variability in it and is not as compact as the smaller one. For these reasons the Matthew's correlation coefficient seems to be lower with this dataset. There are also more false alarms. Table 5 shows the confusion matrix, which supports the results in the performance metrics table. Note that attacks are regarded as positive classifications.

The obtained results confirm, using a bigger dataset, that a manifold learning framework can perform intrusion detection adequately. Earlier research supports the use of similar techniques (Zheng et al., 2009a,b; Yuancheng

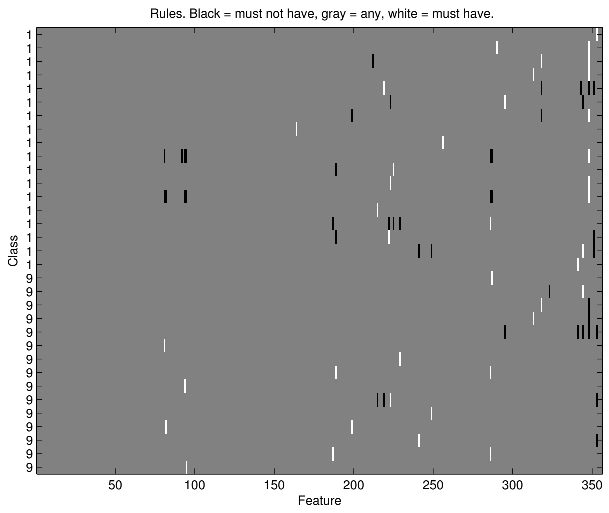

Figure 9: Conjunctive rules. Each line represents one rule. Note that class 1 means normal and class 9 intrusive.

Table 5: Confusion matrix for the whole 10% dataset, 469,320 testing data lines.

		predicted normal	**predicted** attack
actual	normal	87,228	5,162
	attack	5,795	371,135

et al., 2010). This also validates the assumption that a system based on conjunctive rules can achieve promising performance. Next, we will investigate real-life data.

5.4. Ruleset size

Training data size and characteristics affect the size of the created conjunctive ruleset. From Figure 10 we can see that the ruleset size does not increase as fast as the amount of training data points. Because the algorithm is not deterministic (random starting rule), it is also possible that the ruleset size decreases. If the data points are very similar, the ruleset might converge and

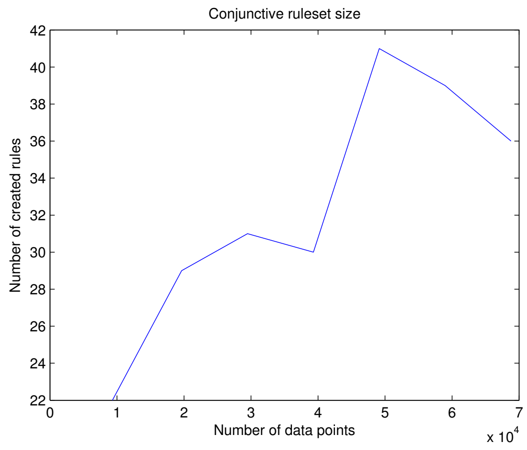

Figure 10: Training data amount effect on rule set size.

additional rules are not created. This means that the ruleset size created in training phase stays manageable even with bigger training data.

6. Real-world network data

Finally, we use real-world network server log data as a case study to describe the feasibility of the approach. This is the most realistic scenario presented in the article. This dataset does not contain labeling, and thus the diffusion map is used to cluster and classify the data for ruleset evaluation. These results have been explored earlier (Juvonen and Sipola, 2013). The dataset comes from a real web server, which serves the company's web services and pages. The logs are received from Apache servers running in a company network, and the lines are formatted as follows:

```
127.0.0.1 - -
[01/January/2012:00:00:01 +0300]
"GET /resource.php?
parameter1=value1&parameter2=value2
HTTP/1.1" 200 2680
"http://www.address.com/webpage.html"
"Mozilla/5.0 (SymbianOS/9.2;...)"
```

The log contains IP address, timestamp, HTTP request, server response code and browser information.

6.1. Feature extraction

In order to be able to use our methods to analyze the log data, textual log lines must be transformed into numerical vectors. For this purpose, we use n-grams (Damashek, 1995). In this context, an n-gram is a substring of characters obtained by sliding window with size n through the string. For example, let's assume that we have two strings, `anomaly` and `analysis`. If we use bigrams (2-grams), we get the following two feature vectors that can be placed into a matrix:

an	no	om	ma	al	ly	na	ys	si	is
1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1

For this study, bigrams are used. The frequencies of each individual bigram are stored in a feature matrix, where each row corresponds to one log line, as demonstrated in the example above. Longer n-grams could also be used, but due to added columns for individual n-grams the feature matrix becomes very large.

6.2. Training size effect

The first real-world dataset contains 4,292 log lines. We extract 490 unique bigrams from the data. Accordingly, after preprocessing the resulting matrix contains 4,292 rows and 490 columns. Because there is no existing labeling, the clustering analysis produces a label for each line, which identifies it as normal of anomalous. We select randomly 2,000 data points for ruleset creation. The labeling information of these lines is used for the rule extraction process. This leaves us with 2,292 log lines that are not present during ruleset learning phase, and these lines are used as a testing set which is classified using the ruleset. Labels for testing set are only used for performance evaluation, not the classification process.

The ruleset for the first real-world dataset contains 6 rules, 2 for classifying detecting normal traffic and 4 for anomalies. Then we classify the testing set using these rules. We discover that one anomalous sample is not covered by any rule, and therefore not classified. The rest are correctly classified. All the normal traffic falls under a single rule. In this case the system works with almost 100% accuracy, which suggests that the amount of data is limited.

The second dataset contains 10,935 log lines. We use the same procedure as with the smaller dataset. In this data, 414 unique bigrams are found,

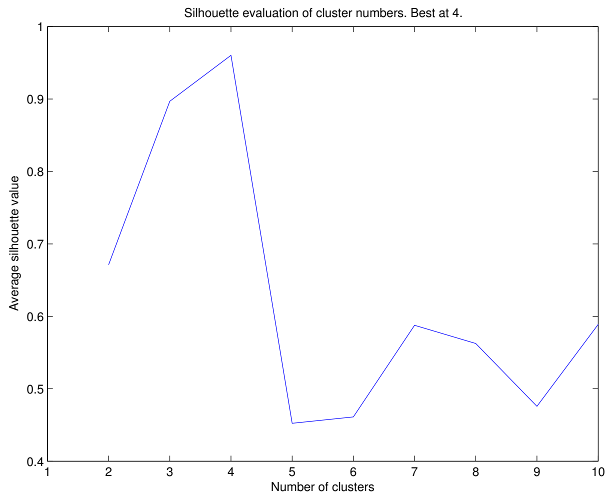

Figure 11: Optimal cluster number for k-means.

resulting in a matrix of size $10,935 \times 414$. For ruleset learning phase, a small number of data points are used. After dimensionality reduction, the number of clusters k is determined as described in section 3.2. Figure 11 shows that the best number of clusters is 4. Figure 12 shows all of the data points after dimensionality reduction and clustering used for unsupervised labeling step. As we can see from this visualization, cluster c_4 contains clearly more points than the others, which suggests that it represents the most common behavior in the data.

We randomly select training data for the rule extraction algorithm. The remaining lines are used for traffic classification testing. With 500 randomly selected training data points, the rule extraction algorithm finds a ruleset that covers the data perfectly. This might suggest overlearning. Table 6 shows these results. In order to generalize the ruleset, training with 100 randomly selected training data points was also performed. The confusion matrix in this case is shown in Table 7. The smaller training dataset is only about 1% of the whole data. This introduces some inaccuracies to the classification results. Because the real labeling is not known, no actual performance metrics can be calculated.

6.3. Discussion of data selection

When using real-world data for training, it must be selected carefully because it is possible to insert delusive information that will result in improper

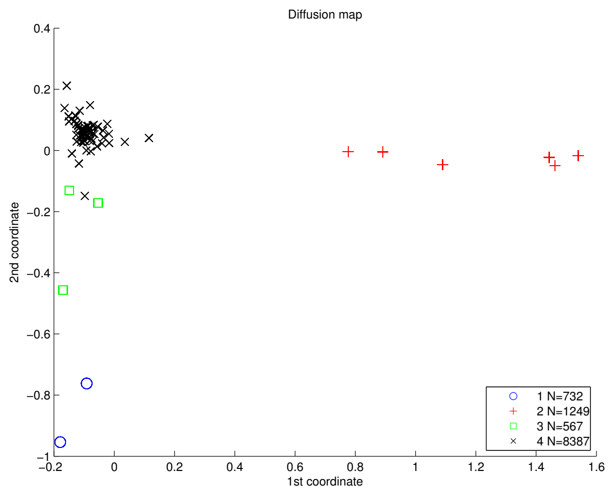

Figure 12: Two-dimensional visualization of the diffusion map of the whole dataset.

Table 6: Confusion matrix for log data with 500 training samples.

		predicted			
		class 1	class 2	class 3	class 4
actual	class 1	697	0	0	0
	class 2	0	1,191	0	0
	class 3	0	0	539	0
	class 4	0	0	0	8,008

Table 7: Confusion matrix for log data with 100 training samples.

		predicted			
		class 1	class 2	class 3	class 4
actual	class 1	725	0	0	0
	class 2	0	1,236	0	0
	class 3	43	0	519	0
	class 4	0	55	0	8,257

training. Therefore limits exist for automatic signature generation in this setting (Newsome et al., 2006; Chung and Mok, 2007; Venkataraman et al., 2008). With intelligent selection of data it is possible to make the training phase more robust against attacks. One way to overcome this is to deliberately insert malicious traffic by the security experts to the training data. Alternatively, the training data can be pre-screened before training to ensure its authenticity (Newsome et al., 2006). In this case the data was as given. We had no control on the data collection and it contains real attacks.

7. Conclusion

We present an anomaly detection system based on the concept of conjunctive rules, which could be used for automated big data monitoring. In particular, we use network log data to demonstrate the feasibility of the system. To achieve this, the structure of the data is learned in an unsupervised manner using the diffusion map dimensionality reduction methodology and labeling is acquired using clustering. Finally, conjunctive rules are extracted using the learned data classification.

Performance of the presented system was tested using KDD Cup 99 dataset, and the practical feasibility was evaluated with real-world data. The results are satisfactory even with a limited amount of training data. In the KDD Cup dataset, the small random subset has a higher Matthews correlation coefficient because the small sample does not include the whole variety of attacks, while the whole 10% dataset is more diverse. The KDD Cup dataset has its own issues, for example the high amount of attacks compared to normal traffic. This makes automatic anomaly detection very challenging. The rule learning process achieves good performance with the real-world dataset. However, because the actual attack labeling is not known, performance metrics cannot be calculated. Nonetheless, manual inspection suggests that normal and anomalous traffic are separated, and we confirm that actual intrusions are detected.

The system is aimed to be a tool for analyzing big datasets and detecting anomalies. It has the following main benefits:

- The resulting ruleset classifies traffic very efficiently. This enables big data classification in real-time. The traffic classification phase is much faster than the preceding rule creation phase.

- The unsupervised learning phase enables label creation without prior information about the data.
- The system aims to explain the classification result of a non-linear black box algorithm with rules that are easy to understand and practical for domain experts.
- The resulting rule-based classifier is simple to use and to implement, since only specific features need to be extracted and it is easy to compare the feature vectors to the conjunctive rules.

The main limitation of the system is the fact that training data has to represent the structure of the new incoming network traffic as accurately as possible. If the training data differs significantly from the current network data, created rules will not work in a satisfactory way. This can be prevented by using periodical training with new, more comprehensive data. Another concern is the performance of the learning phase. A bigger training set might give better results but training will take more time. However, rule creation is performed only periodically, while the fast traffic classification is done in real-time.

For future research, more testing with different kinds of data should be done in order to validate the feasibility of the rule extraction framework. So far we have used actual server HTTP log data and KDD Cup 99 dataset. The system could also be expanded to suit other cyber security and big data scenarios, not just network intrusion detection. This involves modifications to preprocessing but does not change the other components. In addition, some parts of the algorithms can be easily parallelized. This speeds up the processing and makes the system more scalable.

Acknowledgment

This research was partially supported by the Foundation of Nokia Corporation and the Finnish Foundation for Technology Promotion. Thanks are extended to Kilosoft Oy and Pardco Group Oy. The authors thank Professors Amir Averbuch, Timo Hämäläinen and Tapani Ristaniemi for support and guidance.

References

Bach FR, Jordan MI. Learning spectral clustering, with application to speech separation. The Journal of Machine Learning Research 2006;7:1963–2001.

Baldi P, Brunak S, Chauvin Y, Andersen CAF, Nielsen H. Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics 2000;16(5):412–24.

Barakat N, Bradley AP. Rule extraction from support vector machines: A review. Neurocomputing 2010;74(1–3):178–90.

Barakat N, Diederich J. Learning-based rule-extraction from support vector machines. In: The 14th International Conference on Computer Theory and applications ICCTA'2004. 2004. .

Bekkerman R, Bilenko M, Langford J. Scaling up machine learning: Parallel and distributed approaches. Cambridge University Press, 2012.

Brown C, Davis H. Receiver operating characteristics curves and related decision measures: A tutorial. Chemometrics and Intelligent Laboratory Systems 2006;80(1):24–38.

Callegari C, Gazzarrini L, Giordano S, Pagano M, Pepe T. A novel PCA-based network anomaly detection. In: Communications (ICC), 2011 IEEE International Conference on. IEEE; 2011. p. 1–5.

Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. ACM Computing Surveys (CSUR) 2009;41(3):15.

Chu CT, Kim SK, Lin YA, Yu Y, Bradski G, Ng AY, Olukotun K. Map-reduce for machine learning on multicore. In: NIPS. volume 6; 2006. p. 281–8.

Chung SP, Mok AK. Advanced allergy attacks: Does a corpus really help? In: Recent Advances in Intrusion Detection. Springer; 2007. p. 236–55.

Coifman R, Lafon S, Lee A, Maggioni M, Nadler B, Warner F, Zucker S. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. Proceedings of the National Academy of Sciences of the United States of America 2005;102(21):7426–31.

Coifman R, Shkolnisky Y, Sigworth F, Singer A. Graph laplacian tomography from unknown random projections. Image Processing, IEEE Transactions on 2008;17(10):1891–9.

Coifman RR, Lafon S. Diffusion maps. Applied and Computational Harmonic Analysis 2006;21(1):5–30.

Craven MW, Shavlik JW. Using sampling and queries to extract rules from trained neural networks. In: In Proceedings of the Eleventh International Conference on Machine Learning. Morgan Kaufmann; 1994. p. 37–45.

Craven MW, Shavlik JW. Extracting tree-structured representations of trained networks. Advances in neural information processing systems 1996;:24–30.

Damashek M. Gauging similarity with n-grams: Language-independent categorization of text. Science 1995;267(5199):843.

David G. Anomaly Detection and Classification via Diffusion Processes in Hyper-Networks. Ph.D. thesis; Tel-Aviv University; 2009.

David G, Averbuch A. Hierarchical data organization, clustering and denoising via localized diffusion folders. Applied and Computational Harmonic Analysis 2011;.

David G, Averbuch A, Coifman R. Hierarchical clustering via localized diffusion folders. In: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Fall Symposium Series. 2010. .

Davis JJ, Clark AJ. Data preprocessing for anomaly based network intrusion detection: A review. Computers & Security 2011;30(6–7):353–75.

Fawcett T. An introduction to ROC analysis. Pattern recognition letters 2006;27(8):861–74.

Frank A, Asuncion A. UCI machine learning repository. 2010. URL: `http://archive.ics.uci.edu/ml`.

d'Avila Garcez A, Broda K, Gabbay D. Symbolic knowledge extraction from trained neural networks: A sound approach. Artificial Intelligence 2001;125(1):155–207.

García-Teodoro P, Díaz-Verdejo J, Maciá-Fernández G, Vázquez E. Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security 2009;28(1–2):18–28.

Goyal M, Aggarwal A. Composing signatures for misuse intrusion detection system using genetic algorithm in an offline environment. In: Meghanathan N, Nagamalai D, Chaki N, editors. Advances in Computing and Information Technology. Springer Berlin Heidelberg; volume 176 of *Advances in Intelligent Systems and Computing*; 2012. p. 151–7.

Huysmans J, Baesens B, Vanthienen J. Using rule extraction to improve the comprehensibility of predictive models. Available at SSRN: `http://ssrn.com/abstract=961358`; 2006.

Jain AK. Data clustering: 50 years beyond k-means. Pattern Recognition Letters 2010;31(8):651–66.

Jain AK, Dubes RC. Algorithms for clustering data. Prentice Hall., 1988.

Jolliffe I. Principal component analysis. Wiley Online Library, 2005.

Juvonen A, Sipola T. Adaptive framework for network traffic classification using dimensionality reduction and clustering. In: Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012 4th International Congress on. St. Petersburg, Russia: IEEE; 2012. p. 274–9.

Juvonen A, Sipola T. Combining conjunctive rule extraction with diffusion maps for network intrusion detection. In: The Eighteenth IEEE Symposium on Computers and Communications (ISCC 2013). Split, Croatia: IEEE; 2013. .

Kim J, Bentley PJ, Aickelin U, Greensmith J, Tedesco G, Twycross J. Immune system approaches to intrusion detection–a review. Natural computing 2007;6(4):413–66.

Kolias C, Kambourakis G, Maragoudakis M. Swarm intelligence in intrusion detection: A survey. computers & security 2011;30(8):625–42.

Lafon S, Lee AB. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. Pattern Analysis and Machine Intelligence, IEEE Transactions on 2006;28(9):1393–403.

Lee J, Verleysen M. Nonlinear dimensionality reduction. Springer Verlag, 2007.

Li L, Zhang G, Nie J, Niu Y, Yao A. The application of genetic algorithm to intrusion detection in MP2P network. In: Tan Y, Shi Y, Ji Z, editors. Advances in Swarm Intelligence. Springer Berlin Heidelberg; volume 7331 of *Lecture Notes in Computer Science*; 2012. p. 390–7.

Liao HJ, Lin CHR, Lin YC, Tung KY. Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications 2013;36(1):16–24.

von Luxburg U. A tutorial on spectral clustering. Statistics and Computing 2007;17:395–416.

Mahoney MV, Chan PK. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In: Recent Advances in Intrusion Detection. Springer; 2003. p. 220–37.

Martens D, Huysmans J, Setiono R, Vanthienen J, Baesens B. Rule extraction from support vector machines: An overview of issues and application in credit scoring. In: Diederich J, editor. Rule Extraction from Support Vector Machines. Springer Berlin Heidelberg; volume 80 of *Studies in Computational Intelligence*; 2008. p. 33–63.

McHugh J. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln laboratory. ACM transactions on Information and system Security 2000;3(4):262–94.

Molina M, Paredes-Oliva I, Routly W, Barlet-Ros P. Operational experiences with anomaly detection in backbone networks. Computers & Security 2012;31(3):273–85.

Mukkamala S, Sung AH, Abraham A. Intrusion detection using an ensemble of intelligent paradigms. Journal of Network and Computer Applications 2005;28(2):167–82.

Nadler B, Lafon S, Coifman R, Kevrekidis I. Diffusion maps, spectral clustering and reaction coordinates of dynamical systems. Applied and Computational Harmonic Analysis 2006;21(1):113–27.

Newsome J, Karp B, Song D. Polygraph: Automatically generating signatures for polymorphic worms. In: Security and Privacy, 2005 IEEE Symposium on. IEEE; 2005. p. 226–41.

Newsome J, Karp B, Song D. Paragraph: Thwarting signature learning by training maliciously. In: Recent advances in intrusion detection. Springer; 2006. p. 81–105.

Núñez H, Angulo C, Català A. Rule extraction from support vector machines. In: In European Symposium on Artificial Neural Networks Proceedings. 2002. p. 107–12.

Patcha A, Park JM. An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer Networks 2007;51(12):3448–70.

Prasse P, Sawade C, Landwehr N, Scheffer T. Learning to identify regular expressions that describe email campaigns. arXiv preprint arXiv:12064637 2012;.

Ramadas M, Ostermann S, Tjaden B. Detecting anomalous network traffic with self-organizing maps. In: Vigna G, Jonsson E, Kruegel C, editors. Recent Advances in Intrusion Detection. Springer; 2003. p. 36–54.

Ringberg H, Soule A, Rexford J, Diot C. Sensitivity of PCA for traffic anomaly detection. ACM SIGMETRICS Performance Evaluation Review 2007;35(1):109–20.

Rousseeuw PJ. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 1987;20(0):53 – 65.

Ryman-Tubb N, d'Avila Garcez A. SOAR – Sparse oracle-based adaptive rule extraction: Knowledge extraction from large-scale datasets to detect credit card fraud. In: Neural Networks (IJCNN), The 2010 International Joint Conference on. IEEE; 2010. p. 1–9.

Sabhnani M, Serpen G. Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set. Intelligent Data Analysis 2004;8(4):403–15.

Scarfone K, Mell P. Guide to intrusion detection and prevention systems (IDPS). NIST Special Publication 2007;800(2007):94.

Setiono R, Baesens B, Mues C. Recursive neural network rule extraction for data with mixed attributes. Neural Networks, IEEE Transactions on 2008;19(2):299 –307.

Sipola T, Juvonen A, Lehtonen J. Anomaly detection from network logs using diffusion maps. In: Iliadis L, Jayne C, editors. Engineering Applications of Neural Networks. Springer Boston; volume 363 of *IFIP Advances in Information and Communication Technology*; 2011. p. 172–81.

Sipola T, Juvonen A, Lehtonen J. Dimensionality reduction framework for detecting anomalies from network logs. Engineering Intelligent Systems 2012;20:87–97.

Sommer R, Paxson V. Outside the closed world: On using machine learning for network intrusion detection. In: Security and Privacy (SP), 2010 IEEE Symposium on. IEEE; 2010. p. 305–16.

Tan P, Steinbach M, Kumar V. Cluster analysis: Basic concepts and algorithms. Introduction to data mining 2006;:487–568.

Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009. 2009. .

Tran Q, Duan H, Li X. One-class support vector machine for anomaly network traffic detection. China Education and Research Network (CERNET), Tsinghua University, Main Building 2004;310.

Vasiliadis G, Antonatos S, Polychronakis M, Markatos EP, Ioannidis S. Gnort: High performance network intrusion detection using graphics processors. In: Recent Advances in Intrusion Detection. Springer; 2008. p. 116–34.

Venkataraman S, Blum A, Song D. Limits of learning-based signature generation with adversaries. 2008.

Wu SX, Banzhaf W. The use of computational intelligence in intrusion detection systems: A review. Applied Soft Computing 2010;10(1):1–35.

Yuancheng L, Pan L, Runhai J. An intrusion detection method based on LLE and BVM. In: Information Networking and Automation (ICINA), 2010 International Conference on. volume 2; 2010. p. 264–7.

Zhao W, Ma H, He Q. Parallel k-means clustering based on mapreduce. In: Cloud Computing. Springer; 2009. p. 674–9.

Zheng Km, Qian X, Wang Pc. Dimension reduction in intrusion detection using manifold learning. In: Computational Intelligence and Security, 2009. CIS'09. International Conference on. IEEE; volume 2; 2009a. p. 464–8.

Zheng Km, Qian X, Zhou Y, Jia Lj. Intrusion detection using isomap and support vector machine. In: Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on. IEEE; volume 3; 2009b. p. 235–9.