# Feature Set Reduction for Automatic Network Intrusion Detection with Machine Learning Algorithms

R.C. Staudemeyer
Max-Born-Institute for Nonlinear Optics and Short Pulse Spectroscopy
Max-Born-Str. 2 A, 12489 Berlin, GERMANY
Tel: +49-30-6392-1542, Fax: +49-30-6392-1549
e-mail: rcs@mbi-berlin.de

Prof. C.W. Omlin
Middle East Technical University Northern Cyprus, Campus Kalkanli
Guzelyurt, KKTC, Mersin 10, TURKEY
email: omlin@metu.edu.tr

***Abstract*—Selecting a minimum set of core features for automatic network intrusion detection with a variety of machine learning algorithms is a challenging problem. In this paper we propose a minimum feature set which can be easily extracted from network traffic. We compare decision trees, neural networks, naive Bayes and Bayesian networks classifiers performing on the KDDCup99 datasets. We show that by feature selection and preprocessing a comparable classification performance is achievable for the benefit of a significant reduction of training time.**

***Index Terms*—network intrusion detection, feature selection, feature reduction, machine learning, decision trees, neural networks, naive bayes, bayesian networks**

## I. Introduction

Huge amounts of data are being transmitted via computer networks nowadays; any attempts to perform real-time traffic analysis on continuous streams of data necessitates a wise selection of information to be extracted. Traffic analysis for the purpose of network intrusion detection is no exception. Communication sessions between hosts can be characterized by so-called connection records. Every connection record contains a number of features uniquely identifying the connection. Some features such as the duration of the connection, bytes transferred in each direction or the TCP/UDP ports used for communication can be easily extracted. Other more complex features which include information from application layers which require packet inspection are more difficult to obtain.

There has been an increasing diversity and sophistication of threats to modern IT systems; these threats call for novel security mechanisms capable of coping with sophisticated attacks. Merely patching vulnerabilities in a system is no longer a viable solution. Intrusion detection systems aiming at identifying various kinds of malicious activities are thus becoming vital in order to safeguard networks against attacks.

Classical signature-based approaches do not provide suitable solutions for the detection of novel attacks. Machine learning methods offer alternative approaches which are able to learn from monitored network data how to differentiate between normal and anomalous traffic. In particular, they hold the promise of providing a solution that can detect possible attacks in real-time so that countermeasures may be taken in a timely manner.

The purpose of this work was to investigate the application of standard machine learning methods to network intrusion detection; in particular, we were interested in the effects of a reduced feature set on the network intrusion detection performance. We applied decision trees, naive Bayesian learning, Bayesian networks and the neural network backpropagation algorithm to the publicly available DARPA/KDDCup99 dataset. It consists of connection records with 41 features whose relevance for intrusion detection are not clear. We report experiments with different subsets of these features; in particular, we present a subset of 11 features whose performance with the standard machine learning algorithms is comparable to the performance with the full feature set. The majority of selected features are basic features which are easily extracted from a network stream.

## II. Related Work

An excellent introduction into various areas of Machine Learning is provided by Mitchell (1997) [1]. Basic concepts of Machine Learning and their application to the field of network intrusion detection are summarized in Maloof (2006) [2] and Vemuri (2006) [3].

Machine Learning techniques have been applied to network intrusion detection for some time. Usually, the aim is the automatic generation of rules in order to classify network connections. Sinclair et al. (1999) [4] proposed the use of genetic algorithms and decision trees for the automatic generation of such rules. Peddabachigari et al. (2005) [5] investigated and evaluated decision trees and support vector machines. The results showed that decision trees perform slightly better if the dataset is small.

Kruegel et al. (2003) [6] proposed an event classification scheme that is based on Bayesian networks. The scheme significantly reduces the number of false alarms in comparison to threshold-based systems.

Debar et al. (1992) [7] and Cannady (1998) [8] suggested the use of neural networks as components of intrusion detection systems. Mukkamala et al. (2003) studied [9] the application of artificial neural networks and support vector machines in network intrusion detection; their results showed an ensemble of artificial neural networks and support vector machines to show superior detection performance compared to single neural networks. Zhang et al. (2001) [10] applied and compared perceptron, backpropagation, perceptron-backpropagation-hybrid, fuzzy artmap, and radial-based function neural networks for statistical anomaly detection to four different scenario data sets. Their results showed that backpropagation and perceptron-backpropagation-hybrid nets outperformed the other methods. Bivens et al. (2002) [11] further illustrated that neural networks can efficiently be used in network intrusion detection. The authors used classifying, self-organizing maps for data clustering and multi-layer perceptron neural networks for classification. They trained their system to detect denial of service attacks, distributed denial of service attacks, and portscans. Shah et al. (2004) [12] used artificial neural networks and fuzzy inference systems to design an intrusion detection system. Their hybrid system combining fuzzy logic with neural networks outperformed neural networks.

The required infrastructure to capture, prepare and analyze large quantities of network data is defined by Brodie et al. (2005) [13]. Sabhnani et al. (2003) [14] evaluated the performance of a comprehensive set of pattern recognition and machine learning algorithms on a selection of attacks in the KDDCup99 intrusion detection dataset. Sung et al. (2003) extracted a reduced dataset with comparable performance from the 1998 DARPA/KDDCup99 datasets by deleting one feature at a time. They applied neural networks and support vector machines.

In [15] Kayacik et al. (2005) investigated the relevance of all features provided in the KDDCup99 intrusion detection dataset to substantiate the performance of machine learning based detectors trained on KDDCup99 training data.

Chebrolu et al. [16] identified important input features to build computationally efficient and effective intrusion detection systems. They investigated the performance of Bayesian networks and classification and regression trees and suggested a hybrid model. They concluded that the reduction to relevant dataset features can improve performance of machine learning algorithms. Chen et al. (2005) [17] presented a flexible neural tree model for intrusion detection systems with a focus on improving the intrusion detection performance by reducing the input features. Lee et al. (2006) [18] presented a novel feature selection method based on genetic optimization. The performance of the proposed approach was contrasted against the performance of the naive Bayesian classifier. The proposed approach was especially effective in detecting unknown attacks.

## III. Machine Learning Background

### A. Decision Trees

Decision Tree learning is one of the most common machine learning methods. Learned functions are usually represented in the form of a tree-like structure representing a set of decisions, which can be translated into if-then rules. Depending on the algorithm used, the representation may be binary or multibranched.

Nodes in a decision represent some attribute of an instance and branches descending from a node correspond to possible attribute values. Leaves represent possible values of the target variable given the path starting from the root node and ending at the observed leave. To classify an item, the decision is followed from the root to a leaf. At every node, an attribute is tested and based on the outcome the corresponding branch is followed. This procedure continues until a leaf is reached. [1]

### B. Neural Networks

Artificial Neural Networks are inspired by biological learning systems and loosely model their basic functions. They consist of a densely interconnected group of simple neuron-like threshold switching units. Each unit takes a number of real-valued inputs and produces a single real-valued output. Based on the connectivity between the threshold units and element parameters these networks can model a complex global behavior.

In feed-forward neural networks, sets of neurons are organized in layers where each neuron computes a weighted sum of its inputs. Input neurons take signals from the environment and output neurons present signals to the environment. Neurons which are not directly connected to the environment but are connected to other neurons are called hidden neurons.

The most common neural network learning technique is the error backpropagation algorithm. It uses gradient descent to learn the weights in multilayer networks. It works in small iterative steps starting backwards from the output layer towards the input layer. A requirement is that the activation function of the neuron be differentiable. [1]

### C. Bayesian Learning

The naive Bayes is a simple probabilistic classifier. It assumes that the effect of a variable value on a given class is independent of the values of other variables. This assumption is called class conditional independence.

The naive Bayesian classifier is based on Bayes' theorem which provides a way to calculate the posterior probability from the prior probability.

The algorithm stores the prior class probabilities and the posterior probability of each attribute assigned to that class. During the learning phase, it estimates these probabilities from examples by simply counting frequencies of occurrence. The prior probability is the portion of examples from each class. The posterior probability is the frequency that attribute values occur in the given class.

During an observation, the algorithm operates under the assumption that attributes are conditionally independent. The algorithm uses Bayes' theorem to calculate the posterior probability of each class. It returns the class label with the highest probability as the decision.

Despite its simplicity and the assumptions made, this algorithm can often outperform more sophisticated classification methods. The performance and applicability is comparable to decision trees and neural networks. [1]

### D. Bayesian Networks

Bayesian networks are another statistical classifier. They are drawn as a directed acyclic graph where every node represents an attribute and the edges describe the relations between them.

Every node contains a conditional probability table which defines the probability distribution. It is used to predict the class probabilities for every given instance. The probability of each feature value depends on the values of the attributes of the parent nodes. Nodes without parents have an unconditional probability distribution.

Learning of Bayesian networks is basically a search through the space of all possible networks.

The main advantage of Bayesian networks in comparison to naive Bayes is that it is less constraining. They are easy to interpret for humans. The provided estimates can be ranked, which allows the cost to be minimized. [1] [19]

## IV. The Data

The choice of training data available for machine learning in the field of network intrusion detection systems is very limited. One of the few but at the same time most comprehensive, widely used datasets are the DARPA datasets. They are freely available from the Information Systems Technology Group (IST) of the MIT Lincoln Laboratory[1].

The tcpdump data provided by 1998 DARPA Intrusion Detection Evaluation network was further processed and used for the 1999 KDDCup contest at the fifth International Conference on Knowledge Discovery and Data Mining. The learning task of this competition was to classify the preprocessed connection records to either normal traffic or one out of the four given attack categories ('dos', 'probe', 'r2l', 'u2r').

The seven weeks of network traffic collected in four gigabytes of compressed raw tcpdump files from the DARPA training data were preprocessed into five million labeled and categorized connection records with approximately 100 bytes each; and the two weeks of training data were processed into two million unlabeled connections records. Preprocessing of the DARPA data for the 1999 KDDCup contest was done with the MADAMID framework and is described in Lee (1999) [20], Lee (2000) [21]. The KDDCup99 datasets are available from the UCI KDD Archive as the 1999 KDDCup Dataset [22].

A connection record summarizes the packets of a communication session between a connection initiator with a specified source IP address and a destination IP address over a pair of TCP/UDP ports. The labeled connection records in the training set are categorized normal or indicate one of 22 types of attacks. As far as we know, the KDDCup99 dataset is the only publicly available dataset with fully labeled connection records. Training and test sets have different probability distributions.

Each connection record contains 41 input features grouped into basic features and higher-level features. The basic features are directly extracted or derived from the header information of IP packets and TCP/UDP segments in the tcpdump files of each session (basic features 1-9 in table I). This was done by using a modified version of the freely available 'Bro Intrusion Detection System'[2]. Each connection record was produced when either the connection was terminated or Bro was closed. The 'listfiles' for tcpdump from the DARPA training data where used to label the connection records.

The so-called 'content-based' higher-level features use domain knowledge to look specifically for attacks in the actual data of the segments recorded in the tcpdump files. These address 'r2l' and 'u2r' attacks which sometimes only require a single connection or which are without any prominent sequential patterns. Typical features include the number of failed login attempts or whether root access was obtained during the session (features 10-22 in table I).

Furthermore, there are 'time-based' and 'connection-based' derived features to address 'dos' and 'probe' attacks. 'time-based' features examine connections within a time window of two seconds and provide statistics about these. To provide statistical information about attacks extending a two seconds time-window such as slow probing attacks 'connection-based' features use a connection-window of 100 connections. Both are further split into 'same host' features which provide statistics about connections with the same destination host and 'same service' features that examine only connections with the same service (features 23-41 in table I).

The KDDCup99 competition provides the training and testing datasets in a full and a so-called '10%' subset version. The '10%' subset was created due to the huge amount of connection records present in the full set; some 'dos' attacks have millions of records. For this reason, not all of these connection records were selected. Furthermore, only connections within a time-window of five minutes before and after the entire duration of an attack were added into the 10% datasets. To achieve approximately the same distribution of intrusions and normal traffic as the original DARPA dataset, a selected set of sequences with 'normal' connections were as well left in the 10% dataset.

The full training dataset contains 4.898.431 records and the '10%' subset contains 494.021 records. Both contain 22 different attack types which are in the order they were used during the 1998 DARPA experiments.

The full testset with 2.984.154 records is only available unlabeled; but a 311.029 record subset is provided both as unlabeled and labeled test data. It is specified as the '10% corrected' subset with a different distribution and additional attacks not part of the training set.

For the KDDCup99 competition the '10%' subset was intended for training. The '10% corrected' subset containing 37 different attacks can be used for performance testing.

Out of 24 submitted entries the first three places of the original KDDCup99 challenge were using variants of decision trees. Elkan (2000) [23] summarizes the winning entries results of the KDDCup99 challenge.

## V. Feature Set Reduction

From the perspective of data mining, feature set reduction aims to find the set of core features which best classifies the presented data. Some features may contain redundant information, while others may contain information suggesting false correlations; both can hinder correct classification. Additionally, unnecessary features add to computation time.

From the perspective of network intrusion detection systems, there are strong reasons to reduce the number of collected features and choose features which can easily be extracted out of a high-speed data stream. Especially when aiming connections in today's local area networks forward packets with tens of gitabit per second and millions of frames per second.

[1] DARPA Intrusion Detection Evaluation. Information Systems Technology Group (IST). MIT Lincoln Laboratory. web site. http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/index.html

[2] http://bro-ids.org/

TABLE I
KDDCUP99FEATURES

| Nr | features | | | | 12 | 17 | 11 |
|---|---|---|---|---|---|---|---|
| | name | category | attr. type | description | Che04 | | NEW |
| 1 | duration | basic feature | numeric | duration of the connection in seconds | | X | X |
| 2 | protocol_type | basic | nominal | connection protocol (tcp, udp, icmp) | | X | X |
| 3 | service | basic | nominal | destination port mapped to service | X | X | X |
| 4 | flag | basic | nominal | normal or error status flag of the connection | | | |
| 5 | src_bytes | basic | numeric | number of data bytes from source to destination | X | X | X |
| 6 | dst_bytes | basic | numeric | bytes from destination to source | X | | X |
| 7 | land | basic* | nominal | 1 if connection is from/to the same host/port; | | X | |
| 8 | wrong_fragment | basic* | numeric | number of wrong fragments (values 0,1,3) | | X | X |
| 9 | urgent | basic* | numeric | number of urgent packets | | | |
| 10 | hot | content feature* | numeric | number of hot indicators | | | |
| 11 | num_failed_logins | content | numeric | number of failed login attempts | | X | |
| 12 | logged_in | content | nominal | 1 if successfully logged in; 0 otherwise | X | X | |
| 13 | num_compromised | content | numeric | number of compromised conditions | | | |
| 14 | root_shell | content | nominal | 1 if root shell is obtained; 0 otherwise | | X | |
| 15 | su_attempted | content | numeric | 1 if su root command attempted; 0 otherwise | | | |
| 16 | num_root | content | numeric | number of root accesses | | | |
| 17 | num_file_creations | content | numeric | number of file creation operations | | X | |
| 18 | num_shells | content | numeric | number of shell prompts | | | |
| 19 | num_access_files | content | numeric | number of operations on access control files | | | |
| 20 | num_outbound_cmds | content | numeric | number of outbound commands in an ftp session | | | |
| 21 | is_hot_login | content | nominal | 1 if the login belongs to the hot list | | | |
| 22 | is_guest_login | content | nominal | 1 if the login is a guest login | | X | |
| 23 | count | time-based | numeric | number of connections to the same host as the current connection in the past two seconds | X | X | |
| 24 | srv_count | time-based | numeric | number of connections to the same service as the current connection in the past two seconds | X | X | |
| 25 | serror_rate | time-based | numeric | % of connections that have SYN errors | X | X | X |
| 26 | srv_serror_rate | time-based | numeric | % of connections that have SYN errors | | X | |
| 27 | rerror_rate | time-based | numeric | % of connections that have REJ errors | | | |
| 28 | srv_rerror_rate | time-based | numeric | % of connections that have REJ errors | X | | |
| 29 | same_srv_rate | time-based | numeric | % of connections to the same service | | | |
| 30 | diff_srv_rate | time-based | numeric | % of connections to different services | | X | |
| 31 | srv_diff_host_rate | time-based | numeric | % of connections to different hosts | X | | |
| 32 | dst_host_count | host-based | numeric | count of connections having the same destination host | X | X | |
| 33 | dst_host_srv_count | host-based | numeric | count of connections having the same destination host and using the same service | X | | X |
| 34 | dst_host_same_srv_rate | host-based | numeric | % of connections having the same destination port and using the same service | | | |
| 35 | dst_host_diff_srv_rate | host-based | numeric | % of different services on the current host | X | | X |
| 36 | dst_host_same_src_port_rate | host-based | numeric | % of connections to the current host having the same source port | | | X |
| 37 | dst_host_srv_diff_host_rate | host-based | numeric | % of connections to the same service coming from different hosts | | | |
| 38 | dst_host_serror_rate | host-based | numeric | % of connections to the current host that have an S0 error | | | |
| 39 | dst_host_srv_serror_rate | host-based | numeric | % of connections to the current host and specified service that have an S0 error | | | |
| 40 | dst_host_rerror_rate | host-based | numeric | % of connections to the current host that have an RST error | | | X |
| 41 | dst_host_srv_rerror_rate | host-based | numeric | % of connections to the current host and specified service that have an RST error | | | |
| 42 | connection_type | | nominal | | X | X | X |

* = feature provided by the 'Bro Intrusion Detection System'

For feature reduction we used a custom training set with 10.422 instances. This new dataset was sampled and randomised from up-to the 1.000 first examples out of the 23 traffic types contained in the full dataset. Afterwards we classified the traffic to one out of five types ('normal', 'dos', 'probe', 'r2l', 'u2r'). Feature selection was done by examination of the J48 decision tree after every run reducing and/or adding individual and groups of features. Features close to the root of the tree were considered more important than features close to leaves. Features extracted easily from network data were preferred to features requiring domain knowledge or detailed traffic data analysis. Classification and runtime performance of naive Bayes, Bayesian nets and backpropagation neural networks were also observed in every run. The resulting 11 selected features set consists out of seven basic features and four higher level features. They are described in table I.

## VI. Experiments

We used the weka data mining suite which provides a large number of different machine learning algorithms[3]. For feature reduction we applied the C4.5 decision tree algorithm (in weka specified as J48), standard backpropagation with a multilayer feed-forward network (in weka MLP), naive Bayes and Bayesian networks to the DARPA/KDDCup99 training data using 10-fold cross-validation.

We did an extended series of experiments with the aim to extract a reduced features set with only few, if any, content features. This resulted in our 11 features set described in section V.

In our next step we investigated into optimizing preprocessing the selected features to further increase performance.

[3] http://www.cs.waikato.ac.nz/ml/weka/ (09/05/08)

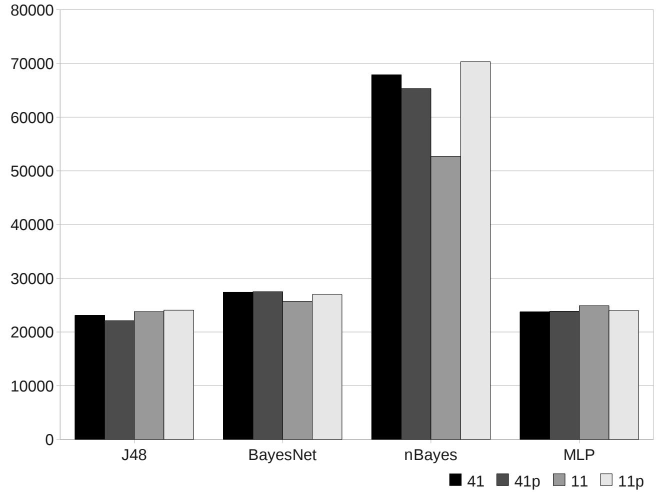

Fig. 1. Comparison of the total number of incorrect classified instances (false positives + false negatives) using the four dataset variants with the 41 features, 41 features preprocessed, 11 features and 11 features preprocessed.

Like most neural network implementations Weka's MLP implementation of the neural network backpropagation algorithm requires floating point numbers as input, preferred in the range $[-1, 1]$. The KDDCup99 dataset contains continues/numeric and nominal/discrete features preprocessed in very different ways.

We wrote our custom preprocessing scripts. Foremost we reset all outlier values to estimated maximum threshold values individually specified for every processed feature. Maximum threshold values were set based on expert knowledge. Next we applied the natural logarithm to selected continues features with strongly biased distributions. Then we scaled the values of continues features to the the range $[-1, 1]$ with a precision of $10^{-6}$.

We encoded binary values as $[-1, 1]$. For discrete features with three or fewer distinct values, we used effects coding. For features with a greater numbers of distinct values we sorted the values using least-first-ranking and scaled the score values to the range $[-1, 1]$. We removed features with non-changing values in the testset (e.g. num_outbound_cmds, is_host_login). Finally we mapped all attacks to one of the five attack types (normal, dos, probe, r2l, u2r).

For a performance comparison with the results of the KDDCup99 competition, we did run experiments using the original '10% training set' and the '10% test set'. The results are shown in table II. Figure 1 additionally shows the different number of misclassifications by the machine learning algorithms applied to all four variants of the '10% training set' (41/11 features, original/preprocessed).

In terms of total accuracy we find that decision trees, Bayesian networks and neural networks are able to hold their performance after preprocessing and reducing the features. The correct categorization of normal traffic remains stable for them as well. The naive Bayes is the only classifier which looses performance noticeably.

An investigation into the true positive rate and the false positive rate per attack traffic class reveal more interesting details:

For the J48 decision tree the detection of network probes decreases but the false alarm rate remains stable. The detection of u2r attacks increases.

Bayesian networks slightly decrease on the detection of probe attacks, but this comes with an improved false alarm rate. Here the detection of network probes, r2l and u2r attacks both decrease. The false alarm rate for u2r attacks is reduced.

The MLP neural network improves the false alarm rate on the detection of dos attacks. The detection of probe attacks improves with a decrease of the false alarm rate. Detection of r2l attacks is decreased and of u2r attacks is increased.

We note that due to the few examples of u2r attacks the very low false alarm rates are not very meaningful and difficult to compare.

Processing time improved for all four classifiers. Outstanding is the reduction of processing time for the neural network which was reduced in total by 97.5%.

Tables III shows the confusion matrix of the result with the highest accuracy of a classifier trained with the feature reduced and preprocessed 10% training set. The result is very close to the winning entry of the KDDCup99 competition.

TABLE III
CONFUSION MATRIX OF TRAINED NEURAL NETWORK

| **predicted →** | **0** | **1** | **2** | **3** | **4** | **%correct** |
|---|---|---|---|---|---|---|
| **actual ↓** | normal | probe | dos | u2r | r2l | |
| 0 normal | 60250 | 236 | 100 | 3 | 4 | 99.43% |
| 1 probe | 794 | 3136 | 235 | 1 | 0 | 75.28% |
| 2 dos | 6228 | 527 | 223098 | 0 | 0 | 97.06% |
| 3 u2r | 61 | 0 | 1 | 8 | 0 | 11.43% |
| 4 r2l | 15610 | 82 | 4 | 92 | 559 | 3.42% |

Experiments were performed on a Dual-Core AMD Opteron@2.22GHz with 32GB memory and a 64bit GNU/debian operating system.

## VII. Conclusions

We have applied machine learning algorithms including decision trees, naive Bayes classifiers, Baysian nets and neural networks to the KDDCup99 dataset for network intrusion detection. Our results show that a large number of features are in fact redundant or at least unimportant for the majority of attacks. We were able to drastically reduce the number of features from initially 41 down to 11 core features. Furthermore, we could significantly decrease the classification time.

The naive Bayes classifier is not well suited for this learning task. It shows poor performance for all traffic types. Bayesian networks show strengths in the classification of network probes but suffer from high false alarm rates in general.

J48 decision trees and MLP neural networks show good performance for this type of datasets. Decision trees show strenghts in the detection of rare r2l and u2r attacks.

The slight decrease in detection of dos attacks and network probes does not hurt. Due to the large amout of connections initiated in series by these attacks a detection rate of 80% is still acceptable.

The first six of the selected core features (1,2,3,5,6,8) are base features which can be easily extracted from network traffic with very less overhead. The remaining features (25,33,35,36,40) are time-based and host-based traffic features. We were able to dismiss all 'content-based' features which are much more complex to extract.

Further research might reveal that some of these remaining traffic features are as well dismissable using machine learning algorithms which are able to extract time series information.

TABLE II

PERFORMANCE OF ORIGINAL 10% TRAINING SET WITH CORRECTED TESTSET USING TRAFFIC TYPE CLASSIFICATION

| | | | | normal | | dos | | probe | | r2l | | u2r | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| feat.set | classifier | time* | accuracy | TPR | FPR | TPR | FPR | TPR | FPR | TPR | FPR | TPR | FPR |
| 41 | J48 | 6m | 92.5759% | 0.995 | 0.089 | 0.973 | 0.003 | 0.747 | 0.002 | 0.058 | 0 | 0.086 | 0 |
| 11p | J48 | 1m | 92.2618% | 0.995 | 0.092 | 0.97 | 0.003 | 0.665 | 0.002 | 0.057 | 0 | 0.143 | 0 |
| 41 | naiveBayes | 1m | 78.1795% | 0.944 | 0.085 | 0.792 | 0.018 | 0.895 | 0.136 | 0.006 | 0.001 | 0.7 | 0.011 |
| 11p | naiveBayes | 7s | 77.3902% | 0.895 | 0.076 | 0.792 | 0.116 | 0.72 | 0.133 | 0.085 | 0.003 | 0.1 | 0.001 |
| 41 | BayesNet | 6m | 91.1892% | 0.99 | 0.084 | 0.95 | 0.002 | 0.836 | 0.014 | 0.101 | 0.001 | 0.629 | 0.005 |
| 11p | BayesNet | 24s | 91.3285% | 0.988 | 0.089 | 0.957 | 0.003 | 0.804 | 0.01 | 0.053 | 0.002 | 0.471 | 0.002 |
| 41 | MLP | 28h53m | 92.3657% | 0.984 | 0.09 | 0.973 | 0.011 | 0.725 | 0.001 | 0.056 | 0 | 0.086 | 0 |
| 11p | MLP | 43m | 92.2908% | 0.994 | 0.091 | 0.971 | 0.004 | 0.753 | 0.003 | 0.034 | 0 | 0.114 | 0 |

* = performed on Dual-Core AMD Opteron@2.22GHz with 32GB memory

## REFERENCES

[1] T. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[2] M. A. Maloof, "Some basic concepts of machine learning and data mining," in *Marcus A. Maloof (Ed.). Machine Learning and Data Mining for Computer Security*. Springer, 2006.

[3] V. R. Vemuri, *Relevance of Machine Learning*. Auerbach Publications, 2006.

[4] C. Sinclair, L. Pierce, and S. Matzner, "An application of machine learning to network intrusion detection," in *Proceedings of the 15th Annual Computer Security Applications Conference*, 1999.

[5] S. Peddabachigari, A. Abraham, and J. Thomas, "Intrusion detection systems using decision trees and support vector machines," *International Journal of Applied Science and Computations, USA*, vol. 11, no. 3, pp. 118–134, 2004.

[6] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Bayesian event classification for intrusion detection," in *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC 2003)*, 2003.

[7] M. Debar, D. Becke, and A. Siboni, "A neural network component for an intrusion detection system," in *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, 1992.

[8] J. Cannady, "Artifical neural networks for misuse detection," in *In Proceedings of the 1998 National Information Systems Security Conference (NISSC'98)*, 1998, pp. 443–456.

[9] S. Mukkamala, A. H. Sung, and A. Abraham, "Intrusion detection using ensemble of soft computing paradigms," in *Advances in Soft Computing*. Springer Verlag, 2003, pp. 239–248.

[10] Z. Zhang, J. Li, C. N. Manikopoulos, J. Jorgenson, and J. Ucles, "Neural networks in statistical anomaly intrusion detection," in *Advances in Neural Networks and Applications, N. Mastoraks (Ed.)*. World Scientific and Engineering Society Press, 2001, pp. 333–338.

[11] A. Bivens, M. Embrechts, C. Palagiri, R. Smith, and B. Szymanski, "Network-based intrusion detection using neural networks," in *Intelligent Engineering Systems through Artificial Neural Networks, Vol. 12*. Proc. ANNIE 2002 Conference, 2002.

[12] K. Shah, N. Dave, S. Chavan, S. Mukherjee, A. Abraham, and S. Sanyal, "Adaptive neuro-fuzzy intrusion detection system," in *IEEE International Conference on ITCC'04*, no. Vol. 1, 2004, pp. 70–74.

[13] M. Brodie, M. Mei, D. George, and S. Ma, "Using data mining for intrusion detection," in *Kantardzic and Zurada (Ed.). Next Generation of Data Mining Applications*. Wiley-Interscience, 2005.

[14] M. Sabhnani and G. Serpen, "Application of machine learning algorithms to kdd intrusion detection dataset within misuse setection context," in *In Proceedings of the International Conference on Machine Learning: Models, Technologies, and Applications*, 2003, pp. 209–215.

[15] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets," in *Proceedings of the PST2005*, 2005.

[16] S. Chebrolu, A. Abraham, and J. Thomas, "Feature deduction and ensemble design of intrusion detection systems," in *Computers and Security*. Elsevier Science, 2005.

[17] Y. Chen and A. Abraham, "Feature selection and intrusion detection using hybrid flexible neural tree," in *Second IEEE International Symposium on Neural Networks (ISNN 2005)*. Lecture Notes in Computer Science, Springer Verlag, Germany, 2005.

[18] C. H. Lee, S. W. Shin, and J. W. Chung, "Network intrusion detection through genetic feature selection," in *Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD06)*, 2006.

[19] I. H. Witten and E. Frank, *Data Mining - Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005, no. Second Edition.

[20] W. Lee, "A data mining framework for constructing features and models for intrusion detection systems," in *PhD thesis*, 1999.

[21] W. Lee and S. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM Transactions on Information and System Security*, no. 3, pp. 227–261, 2000.

[22] S. Hettich and S. Bay, "The uci kdd archive. web site, http//kdd.ics.uci.edu (last access: 08.05.2009)," in *Department of Information and Computer Sciences, University of California, Irvine*, 1999.

[23] C. Elkan, "Results of the kdd99 classifier learning. sigkdd explorations," in *Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining*, no. Volume 1, Issue 2, 2000, pp. 64–34.

**Ralf Staudemeyer** is employed as a network administrator at the Max-Born-Institute in Berlin, Germany. He is PhD candidate at the University of the Western Cape, South Africa. His interests are currently in the fields of machine learning and computer network intrusion detection.