SNIDS: An Intelligent Multiclass Support Vector Machines Based NIDS

Srinivasa K G AdarshPatil, Harsha K C, Akshay V Joshi and Pramod N

Machine Learning Applications Laboratory, Department of Computer Science and Engineering,
M S Ramaiah Institute of Technology, Bangalore,
{*kgsrinivasa, adarshpatil123, akshayvjoshi25, harshakc01, npramod05*}*@gmail.com*

Increase in the number of network based transactions for both personal and professional use has made network security gain a significant and indispensable status. The possible attacks that an Intrusion Detection System (IDS) has to tackle can be of an existing type or novel. The challenge for researchers is to develop intelligent IDS, which can detect new attacks as efficiently as they detect known ones. Intrusion Detection Systems are rendered intelligent by employing machine learning techniques. In this paper we present a statistical machine learning approach to the IDS using the Support Vector Machine (SVM). The network traffic is modeled into connections based on protocols at various network layers. These connection statistics are given as input to SVM which in turn plots each input vector. The new attacks are identified by plotting them with respect to the trained system. The experimental results demonstrate the lower execution time of the proposed system with high detection rate and low false positive number. The 1999 DARPA IDS dataset is used as the evaluation dataset for both training and testing. The proposed system, *SVM NIDS* is bench marked with *SNORT* [1], an open source IDS.

Keywords: Intrusion detection system; Support vector machine; SNORT.

1. Introduction

Intrusion detection has been an active field of research for about two decades. Over these years the world has seen intrusions of varying intensities ranging from small and less intense attacks such as *port sweep* or *port scanning* to attacks which compromise the whole network. In response the researchers have developed systems to counterfeit these intrusions. Every attack is characterized by a signature which is derived from the network traffic. The efficiency of the network intrusion detection system (NIDS) is related to the variety of the attacks it can counter.

Intrusion detectors typically base their decisions either on signature or anomaly characterization. A wide range of Artificial Intelligence (AI) techniques have been adopted in IDSs. Initially, Rule Based Systems (RBSs) were the first to be employed successfully, and are still at the core of many IDSs. This allows for IDSs that automatically filter network traffic and/or analyze user data to identify patterns of known intrusions. This method is apt to detect previously known attacks. In case of unseen attacks, only an appropriate abstraction of the pattern can be deployed to predict intrusions. They are inherently unable to detect novel attacks.

Anomaly based IDS pick on the abnormalities in the characteristics to classify the suspect as an attack. These characterization approaches range from statistical, inference methods to techniques which are inspired from the human immune system or bio inspired in general. The primary strength of this approach is its ability to recognize novel attacks. The drawback here is that, to attain required accuracy, intensive training would be indispensable. The efficiency depends largely on the diversity of the training data set and the aptness of the parameters used in the training. This paper deals with an implementation of intrusion detection on the lines of anomaly based system by adopting a statistical machine learning approach.

Based on the parameters used for training, SVM [2] creates a hyper plane which can be viewed as the demarcation between the regions. The number of regions being involved is decided by the choice of classification method within SVM. The accuracy of the classification relies largely on the optimality of the parameters used for classification. In intrusion detection the standard set of attacks and their corresponding parameters are used

to draw this hyper plane. The hyper plane divides the region into classes of attacks and normal part. The parameters of the suspect are also plotted with respect to the hyper plane. If this plot lies in the normal zone then it is credited to be safe else it is signaled as an attack. The system formulation takes place in two stages, training and testing. The system learns from the statistics during the training and draws the hyper plane. Testing scores the efficiency of the trained system to detect attacks.

2. Background

Early development started with understanding the attacks and their identities. System to trace occurrences of these identities was the order of the day. This gave rise to the classic signature based intrusion detection systems [3]. Though robust these systems lack the potential to handle novel attacks [4]. Data mining techniques are often used to make signature matching more efficient. With increasing complexity of the networks to state the least the increasing protocol layers, the attackers have modeled the attacks to exploit the loop holes in this complex system. Hence there is a need to generalize the Intrusion Detection process. Newer models to detect intrusion have become prevalent replacing mechanisms which fish for specific character set to find attacks. This lead to the rise of another class of IDSs which uses anomaly in the traffic characteristics for intrusion detection. Of the several models available one model describes the use of protocols, time based analysis for building the model. This is coupled with algorithm for learning from the conditional rules [5]. Another model involves the combination of signature based detection for known attacks and anomaly based detection for new attacks forming a hybrid intrusion detection system. They illustrate the use of fuzzy data mining techniques for anomaly based detection [6]. A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification studies the various well known algorithms used for classification of data in intrusion detection systems. A number of innovations root their inspiration from the system in living being, and the intrusion detection counterpart found in humans is their immune system and genetics. Modeling this artificial immune system is based on the way the immune system in the body acts against the pathogens [7]. Another approach is the application of Genetic algorithm, a machine learning tool, in intrusion detection systems. The use of Genetic Algorithm and Decision Trees to automatically generate rules for classifying the network connections [8]. A team from Columbia University has proposed an algorithm for unsupervised anomaly based intrusion detection system for unlabeled attacks. The use of sequence learning methods like instance based learning and Hidden Markov Model in anomaly based intrusion detection [9], but as the model looses out on the specificity so does it on the accuracy. Increasing number of false alarms was another trouble emerging. Hence there has been work concentrated towards reducing the false alarms. The use of data mining and machine learning specifically to reduce the number of false alarms. They propose to achieve this objective by two methods 1) Improving the quality of alerts 2) Alert correlation. The data mining techniques becomes an indispensable tool when it comes to handling the enormous network data for examination [10].

The need for a generalized independent model steered the research toward intelligent intrusion detection systems. By intelligent we mean a system which can learn from the data trends and improve its efficiency over time. Such systems cancel out the need for human intervention and make way of autonomous detection. The intelligence is induced by the use of techniques from fields like Machine Learning, Genetic Algorithm, Data Mining. Hence an ideal intrusion detection system would be one which 1) detects novel attacks 2) produces minimal (preferably 0) false alarms and 3) learns over time and hence voids the need of human assistance for up gradation.

3. Motivation and Contribution

So far there has been immense development in the field of intrusion detection. But most of it has been concentrated towards signature based intrusion detection systems. These signature based systems can handle only the known attacks. When it comes to new attacks these systems fail. With the rapidly changing face of the attacks we need systems which learn from these changes and hence efficiently detect new attacks. The alternative to a signature based system, anomaly based system is the answer to this problem. Anomaly based systems can

be efficiently used to detect novel attacks. In this paper we illustrate the implementation of an anomaly based system. In this implementation, the network traffic is mined for statistical data and is then fed to a statistical machine learning algorithm, Support Vector Machines.

4. Dataset for Evaluation, Training and Testing

Application of machine learning techniques involves use of a good dataset and extraction of relevant features from the dataset. Many of the times these datasets are generated in a controlled environment. This ensures the controlled injection of intrusions (artificially and intentionally induced) which can later be used for detection. Many of such research based dataset often come in handy but are never up to the mark due various reasons ranging from the out datedness of the dataset (hence the attacks which are present in it) to the inadequate spread of the incident attacks in the dataset. Keeping these in mind researchers tend to generate specific sets involving highly specific attacks and thus use them in evaluation of IDSs. In the forthcoming method used for Intrusion Detection, we have used the standard DARPA Intrusion detection and evaluation dataset (98–99).

In the DARPA IDS evaluation dataset, all the network traffic including the entire payload of each packet was recorded in *tcpdump* format and provided for evaluation. In these evaluations, the data was in the form of sniffed network traffic, Solaris BSM audit data, Windows NT audit data (in the case of DARPA 1999) and filesystem snapshots and tried to identify the intrusions that had been carried out against a test network during the data-collection period. The test network consisted of a mix of real and simulated machines; background traffic was artificially generated by the real and simulated machines while the attacks were carried out against the real machines. Classification of the attacks into four main classes namely, Denial of Service (DoS), Remote to Local(R2L), User to Remote(U2R) and the Data attacks/surveillance and probing [11].

- **Denial of service** (DoS) attacks were designed to disrupt a host or network service. Some DoS attacks (e.g. smurf) excessively load a legitimate network service, others (e.g. teardrop, Ping of Death) create malformed packets which are incorrectly handled by the victim machine, and others (e.g. apache2, back, syslogd) take advantage of software bugs in network daemon programs.
- **Remote to Local** (R2L) attacks, an attacker who does not have an account on a victim machine, sends packets to that machine and gains local access. Some R2L attacks exploit buffer overflows in network server software (e.g. imap, named, sendmail), others exploit weak or misconfigured security policies (e.g. dictionary, ftp-write, guest) and one (xsnoop) is a trojan password capture program. The snmp-get R2L attack against the router is a password guessing attack where the community password of the router is guessed and an attacker then uses SNMP to monitor the router.
- **Users to root** (U2R) attacks occur when a local user on a machine is able to obtain privileges normally reserved for the UNIX root or super user. Some U2R attacks exploit poorly written system programs that run at root level which are susceptible to buffer overflows (e.g. eject, ffbconfig, fdformat), others exploit weaknesses in path name verification (e.g. loadmodule), bugs in some versions of suidperl (e.g. perl), and other software weaknesses.
- **Probe or scan** attacks include many programs that can automatically scan a network of computers to gather information or find known vulnerabilities. Such probes are often precursors to more dangerous attacks because they provide a map of machines and services and pinpoint weak points in a network. Some of these scanning tools (e.g. satan, saint, and mscan) enable even an unskilled attacker to check hundreds of machines on a network for known vulnerabilities.

5. Support Vector Machine: An Introduction

Basic SVM algorithm is an efficient binary classifier. The idea behind SVM approach to intrusion detection is that we map our data to a feature space. This feature space is the basis for the SVM algorithm which determines a linear decision surface (hyperplane) using the set of labeled data within it. This surface is then used to classify future instances of data. Data is classified based upon which side of the decision surface it falls. SVM is applicable to both linearly separable and non-linearly separable patterns. Patterns not linearly separable are transformed using kernel functions-a mapping function, into linearly separable ones.

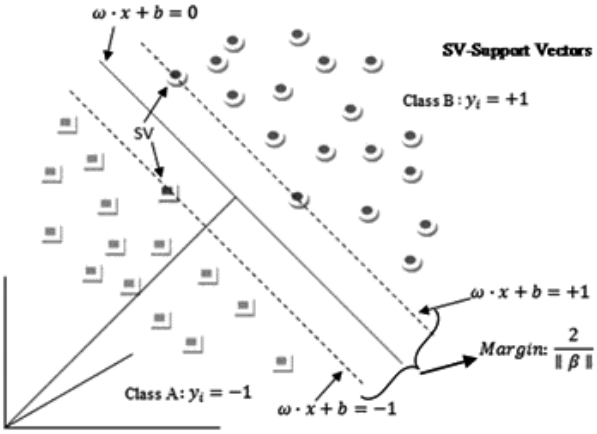

Figure 1. Classification in linearly separable two class data

It can be formulated as follows, the optimal hyperplane separating the two classes can be represented as:

$$\boldsymbol{\omega} \cdot \boldsymbol{X} + \boldsymbol{\beta} = \mathbf{0} \tag{1}$$

where, X – sample input vectors defined as

$$\{(x_1, y_1), (x_2, y_2) \ldots (x_k, y_k)\} x_k \in R^n y_i \in \{1, -1\}$$

ω, β – non zero constants ω indicating the weight component and β indicating the bias component.

The ordered pair x, y is the representation of each input used to form hyperplane which are N dimensional vectors labelled with corresponding y.

$$\omega \cdot X + \beta \geq 1 \text{ if } y_i = 1$$
$$\omega \cdot X + \beta \leq -1 \text{ if } y_i = -1$$

These can be combined into one set of inequalities:

$$yi(\boldsymbol{x}_i \cdot \boldsymbol{\omega} + \boldsymbol{\beta}) \geq 1 \forall i$$

The above inequalities hold for all input samples (linearly separable and suffice the optimal hyperplane equation). The optimal hyperplane is the unique one which separates the training data with a maximal margin. The figure 1 depicts the above mathematical representation [12].

Similarly the mathematical representation of Multi Class SVM [13]. Figure 2 sketches the idea of hyperplanes and classification in case of multi class data. One of the highlighting difference between the binary and multi class SVM is the set $y = \{1, 2, 3, \ldots, k\}$ and operations which are dependent on this set. In our experiments and the proposed system, libSVM [14] is used as the library providing the SVM algorithm.

6. Feature Space

The statistical learning algorithm works on the idea that all the points are numerically represented in the given feature space. Intuitively it is more relevant if this data is statistical in nature that is a measure of some statistics of the input data. Considering the above mentioned criterion, researchers developed another dataset which was derived from the DARPA's IDS evaluation dataset called the KDD 99 cup dataset. 41 features were extracted each of certain significance considering the attacks which were inherent in the data. The common intersection of these features representing attack instances and the statistical nature of the feature are more effective in detecting attacks especially when the machine learning algorithm under consideration is a statistical one. Among the 41

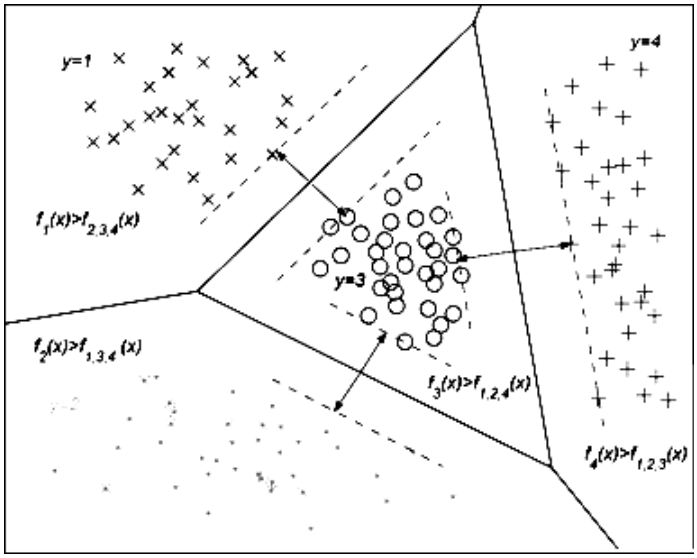

Figure 2. Classification in multi class data

Table 1. List of features and their description.

Feature name	Feature description
duration	length (number of seconds) of the connection
protocol type	type of protocol, e.g. tcp, udp, etc.
service	network service on the destination, e.g., http, telnet, etc.
src_bytes	number of data bytes from source to destination in a connection
dst_bytes	number of data bytes from destination to source in a connection
count_src	Number of connections made by the same source as the current record in the last k seconds
count_dst	Number of connections made to the same destination as the current record in the last k seconds
count_srv_src	Number of different services from the same source as the current record in the last k seconds
count_srv_dst	Number of different services to the same destination as the current record in the last k seconds
dst_host_count	count of connections having the same destination host
dst_host_srv_count	count of connections having the same destination host and using the same service

features, the most relevant features which represents some statistical behavior along with few other count based features were considered for our system. These features were constant time based counts. The following table lists the feature and the description of the feature.

The proposed system architecture is as shown in figure 3. It include 3 main components namely the preprocessor, the learning engine and the detection engine. The proposed architecture is a connection based NIDS where each TCP/IP connection is analysed over the duration for which the connection is persistent.

6.1 *Preprocessor*

Preprocessor interacts with network interface and is responsible for collecting data at various network layers. It comprises of extraction engine and the encoding.

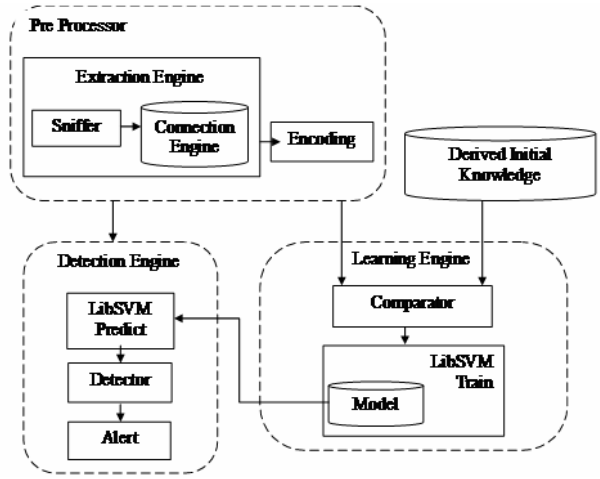

Figure 3. Architecture of NIDS based on multi class libSVM

- **Extraction Engine**: The main task of an extraction engine is to interpret the network data into structures of meaningful representations according to the respective formats in various layers of the network architecture i.e. into frames at Ethernet layer and packets at TCP and IP layer. These tasks can be further divided into Sniffing and connection maintenance.
- **Sniffer**: Sniffer is a pcap based packet capture routine. It extracts the header information such as the Ethernet header, IP header, TCP header. It also maintains these headers in a structure which can be used by the other part of the system for further processing. The output of a sniffer is the packet information contained in the above mentioned structure. Collection of packets form connection given that the protocol under discussion is a connection oriented protocol (eg:tcp). The connection engine uses this packet information to form the connection information which is used for the statistical analysis.
- **Connection Engine**: Connection engine maintains the connection information of the packets at the transport layer protocol level. This connection maintenance is dependent on various connection indicating flags in the TCP header (SYN, ACK etc). It also takes care of the multiple connections spanning a pair of IP, port.no ordered pair. The connection engine is responsible for maintaining various derived features which are incident in the packet but not directly available. Eg: srv_count, dst_host_count etc.
- **Encoding**: The features extracted from the extraction engine(direct and derived) must be encoded to a format which is compatible with the libSVM library. Encoder does this building of input vector from the features extracted. The encoded features are written on to a file and this is used as on of the input for the learning engine.

6.2 *Derived initial knowledge (DIK)*:

The DARPA IDS evaluation dataset [15] consists of a file describing connection based attack information. This file is the original marker indicating the actual attacks in the system in a summarized manner. Each line indicates partial connection information which is used to pin point which is the connection in the tcpdump (packet data) and the label (attack type or normal) is used to mark the input vector for a class type which is useful for the SVM algorithm.

6.3 *Learning engine*

Learning engine is a vital part of the system and it builds the knowledge base. This knowledge base enables the system to make intelligent decisions. The knowledge base is very similar to the equation of the hyperplane and is the output of the libSVM training.

The following are the two components of the learning engine:

- **Comparator**: This unit matches derived initial knowledge against the data provided by the extraction engine. This match formulates a database of attacks and non attacks in the libSVM encoded form. The main task of this unit is to match the input feature vector from the extraction engine with the derived initial knowledge in order to mark the vector with the type of class to which it belongs as indicated in the DIK provided as a part of the IDS evaluation Dataset(Supervised Learning Ideology).
- **libSVM-train**: libSVM is an integrated software for support vector classification, regression and distribution estimation. LIBSVM uses the training data set and formulates a model. This model acts as the knowledge base for the IDS consisting of the classification parameters (hyperplane details). The model file thus generated is used for prediction during the detection phase.

6.4 *Detection engine*

The detection engine uses the model constructed by the learning engine to classify real time traffic. The inputs to this stage is the encoded feature vector from the preprocessor and the training model from learning engine.

The following is the component of the detection engine:

- **libSVM-Predict**: The predictor uses the SVM model and analyzes the given encoded input vector. The output of this stage gives the class label after the classification stage and the confidence with which the classifier was able to classify it into a particular class.
- *Detector*: The detector maintains a state variable which is used to flag the input connection as the attack connection. This state variable in combination with the input from the libSVM-predict is used to flag the incident connection. This depends on the the confidence value of classification, the connection under consideration etc. Once the connection is flagged the vector and the connection summary is written onto a file with the attack label. This file corresponds to the **Alert** part in the architecture.

6.5 *Input sampling and evaluation*

The output generated after the preprocessing stage which includes feature extraction from the network data and the encoding of this feature is as shown below.

1 1:0 2:1 3:1 4:239 5:486 6:8 7:8 8:2 9:2 10:7 11:6
1 1:2 2:1 3:3 4: 5:486 6:8 7:8 8:2 9:2 10:7 11:6
1 1:0 2:1 3:5 4:1367 5:335 6:3 7:3 8:2 9:2 10:21 11:72
6 1:182 2:1 3:6 4:1511 5:2957 6:1 7:1 8:2 9:2 10:1 11:3

6.6 *Performance analysis*

An IDS can be tuned in such a way that it scores particularly well on a particular data set. Some of the attributes specifically are: Remote Client Address, TTL, TCP options and TCP window size – have a small range in the DARPA simulation, but have a large and growing range in real traffic. IDS which take into account the above-mentioned attributes are likely to score much better on the DARPA set than in real life. Since our system does not consider these attributes, we can legitimately expect that the system in real life performs as well as it does on the DARPA benchmark. The proposed system is trained using internal network traffic of week 1 and week 3 [15]. Then the testing weeks 4 and 5 [15] data are used to benchmark the network intrusion detection process. This data contains several attack instances as well as legal traffic, directed against different hosts of the internal network: the attack source can be situated both inside and outside the network. The results of the experiment conducted on the system using the DARPA IDS Evaluation dataset are tabulated in the following graphs and tables.

Table 2 represents the true positive rate of each class of attack in the DARPA Dataset. The true positive rate is indicative of the fact that the modelling of features is efficient for detecting the following classes of attack. The

Table 2. True Positive Rate for each class.

Attack class	Normal	Probe	DOS	U2R	R2L
TPR	0.9936	0.9613	0.9711	0.8725	0.882

Table 3. Confusion matrix of DARPA test set (one day) on SVMNIDS.

	Normal	Probe	DoS	U2R	R2L
Normal	74425	163	180	57	79
Probe	48	3039	63	8	5
DoS	5486	1821	292712	856	547
U2R	69	18	28	919	20
R2L	155	14	60	21	1873

Table 4. Total number of instances of each class of data in test set used.

	Number of instances
Normal	74904
Probe	3163
DoS	301422
U2R	1054
R2L	2123

test set consisted of new forms of attacks which belonged to various classes. The system was able to classify the novel (but ones belonging to the existing class) attacks with the tabulated accuracy.

The low detection rate of R2L and U2R is due to the nature and the way these attacks are launched. These attacks are prominently payload based which include access of a remote system by the use of commands for example the use of ftp commands such as *su,ls,user* etc. Some of these commands are not of attack type and the rest are. Modellingthese attacks for a statistical learning algorithm is relatively not accurate as it lacks the statistical features. Although it can be modelled as a binary 0/1 value in the vector, depending on the payload analysis, increasing the dimension of the feature space. This was partly incorporated in the KDD cup data set [16] and the results of our SVMNIDS on DARPA dataset is as shown in table 3 Further to this, the table can be compared with the KDDcup winner result [16].

Table 3 suffices the true positive rate of the system and gives the anatomical view of the classification via the confusion matrix. It also indicates that the incident attack on the test dataset (of a particular day) consists more of DOS class(including novel DOS attacks). The account to number of instances in the test set is as shown in Table 4.

The above analysis indicate that the system performs better on DARPA dataset (and inturn on realtime traffic) and matches closely with the desired results obtained on a more statistical KDDcup dataset [16]. The following is a benchmark comparison with the Open Source IDS SNORT [1]. Snort is a rule based Intrusion Detection System which effectively compares the rule set and makes decision on intrusion behavior. It analyses various parts of the packet including headers of different layer, preprocessing on payload etc. Snort though widely used and deployed in real time systems, it is known to produce undesirable results with attacks and datasets which are found in DARPA IDS evaluation dataset. The large number of false positives has been a raising concern for network administrators. Figure 4 gives an insight on the behaviour of SVM NIDS and the SNORT IDS. The ROC curve was plotted for the same testing dataset which was run on both the system. SVM NIDS at best was able to detect attacks with a True Positive Rate (TPR) of 0.9672 and a False Positive Rate (FPR) of 0.122 and snort on the other hand did the same with a True Positive Rate (TPR) of 0.853 and False Positive Rate

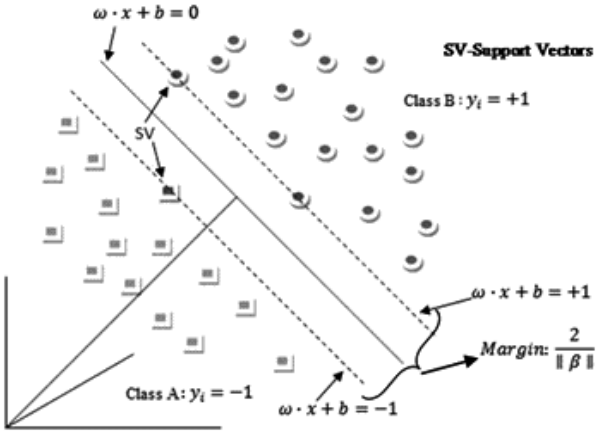

Figure 4. TPR vs FPR (ROC curve) plot comparing SVM NIDS and snort IDS

(FPR) of 0.12. On an average SVM NIDS has a higher TPR and a corresponding low FPR. An analysis of the performance of SNORT on the DARPA dataset has allowed us to perform the comparison [17].

7. Conclusion and Future Work

It is often difficult to know which items from an audit trail will provide the most useful information for detecting intrusions. The process of determining which items are most useful is called feature selection in the machine learning literature. The selection process for our system yielded the feature set shown in Table 1. This feature set resulted in more than satisfactory overall results favouring few classes. This can be overcome to detect every class of attack with the similar accuracy by selecting more effective statistical features by increasing the feature space and also taking care of redundant features i.e features which do not help in detecting many of attacks. This will help in improving the performance in terms of detection rate as well as the time involved in computation during training.

Our system uses a standard one model system of libsvm where all of the training is reflected on to a single model file (KB). This can be improved by training the system with multiple models which when combined forms a multi class classifier. This is useful in detecting the minority classes and classes such as R2L and U2R. This will also help in determining the confidence level of the system by comparing the prediction against different model files.

The system does not provide filtering as in case of a normal firewall. This can be an improvement which can be included in our future work. Connection based approach is costlier because it has to maintain connection information for realtime traffic. This is especially costly when we consider the current multiple connection scenarios where much connection can be started from and to a source destination pair. Our system uses a linear linked list to maintain connection information which can be improved to use of tress or TRIEs (retrieval trees). Incremental statistical learning is another area where there can be lot of future development in Intrusion detection. With incremental learning the performance of the system can increase over time adjusting to the incoming traffic.

Finally, the results of the proposed system show promising development in the application of statistical learning for intrusion detection systems.The same is true for connection based IDS as the results from our system is better than those shown by rule based or packet based approaches.

8. Acknowledgements

This project is financially supported by DRDO sponsored project titled Machine Learning Techniques for Data Mining Based Intrusion Detection Systems (Ref. No.: ERIPR/ER/0705066/M/01/1256) to Dr. Srinivasa K G, Professor, Department of Computer Science and Engineering, M S Ramaiah Institute of Technology, Bangalore,

India. We acknowledge Dr. T V Suresh Kumar, Dr. K Rajanikanth, Dr. D E Geetha and Mrs. Mrunalini M for their kind support.

References

[1] M. Roesch. *Snort – Lightweight Intrusion Detection for Networks*. Proceedings of USENIX LISA'99, November 1999.
[2] Cristianini, Nello, ShaweTaylor, John; *An Introduction to Support Vector Machines and other kernel-based learningmethods*, Cambridge University Press, 2000.
[3] Nitin, Mattord, Verma. *Principles of Information Security*. Course Technology. pp. 290–301, 2008.
[4] Anderson, Ross. Security Engineering: A Guide to Building Dependable Distributed Systems. New York: John ley & Sons. pp. 387–388, 2001.
[5] M. Mahoney, *A Machine Learning Approach to Detecting Attacks by Identifying Anomalies in Network Traffic*, Ph.D Dissertation, Florida Institute of Technology, 2003.
[6] S. Bridges and R. Vaughn, *Fuzzy data mining and genetic algorithms applied to intrusion detection*, Proceedings twenty third National Information Security Conference, October 1–19, 2000.
[7] M. Glickman, Balthrop and S. Forrest. *A machine learning evaluation of an artificial immune system*. Evolutionary Computation, 13(2):179–212, 2005.
[8] Sara Matzner Chris Sinclair, Lyn Pierce, *An application of machine learning to network intrusion detection* in Proceedings of the 15th Annual Computer Security Applications Conference, pages 371–377, Phoenix, AZ, 1999.
[9] T. Lane and C. Brodley, *Temporal sequence learning and data reduction for anomaly detection*. ACM
[10] Transactions on Information and System Security, 2(3), August 1999.
[11] T. Pietraszek and A. Tanner *Data mining and machine learning towards reducing false positives in intrusion detection*. Inform Secur Tech Rep; 10(3):169–83, 2005.
[12] R. Lippmann, *et al. The DARPA Off-Line Intrusion Detection Evaluation*, Computer Networks 34(4) 579–595, 2000.
[13] V. Vapnic. *The Nature of Statistical Learning Theory*, Springer, New York, 1995.
[14] C. Cortes and V. Vapnik, *Support-vector network*, Machine Learning, vol. 20, pp. 273–297, 1995.
[15] C. C. Chang and C.-J. Lin, *LIBSVM: A Library for Support Vector Machines* 2001.
[16] DARPA intrusion detection evaluation, http://www.ll.mit.edu/IST/ideval/data/data index.html
[17] KDD Classifier Learning Contest http://cseweb.ucsdedu/∼elkan/clresultshtml, 1999.
[18] C. Thomas and N. Balakrishnan, *Usefulness of DARPA data set in Intrusion Detection System evaluation*,
[19] Proceedings of SPIE International Defense and Security Symposium, 2008.