Supervised Learning Techniques for Intrusion Detection System based on Multi-layer Classification Approach

Machine Learning and Intrusion Detection System

Mansoor Farooq
Department of Management Studies
University of Kashmir
Srinagar, India

***Abstract*—The goal of this study is to discover a solution to two problems: first, the signature-based intrusion detection system SNORT can identify a new attack signature without human intervention; and second, signature-based IDS cannot detect multi-stage attacks. The interesting aspect of this study is the growing ways to address the aforementioned issues. We introduced a multi-layer classification strategy in this study, in which we employ two layers, the first of which is based on a decision tree, and the second of which includes machine learning technique fuzzy logic and neural networks. If the first layer fails to identify fresh attacks, the second layer takes over and detects new signature assaults, updating the SNORT signature automatically.**

Keywords*—IDS; SNORT; fuzzy logic; neural networks; decision tree; Naïve Bayes

I. Introduction

According to information technology, a network intrusion is a sequence of attacks against network-based security measures [13]. Network traffic is monitored by the Intrusion Detection System (IDS), which alerts information security personnel when harmful activity is discovered [10].

Because of their effectiveness in blocking assaults on network resources, IDSs are not able to adapt to scenarios where new attacks are being carried out, requiring human intervention [13]. On the other hand, if the IDS is used on an overloaded network, it might constitute a bottleneck. For the IDS to be launched to production, it needs time to analyse network data [13].

Using an existing IDS, SNORT would be used to compare packet signatures to the criteria set out by SNORT. Packets that are thought to be malicious will be run through an intelligent model that has been trained to look for harmful content [14][9]. Using an intelligent model, SNORT might be used as the initial step of a strainer to limit traffic for unnecessary exploration, to put it another way: SNORT's workload is reduced, which in turn reduces human mediation since the intelligently trained model is responsible for determining whether or not a certain group of packets is harmful. SNORT will establish an automatic signature if a malicious group of packets is detected.

For the first time, a training model is being combined with a reasoning model to detect abuse of network data packets [9]. IDS on a production level device may then utilize the rule generated by the justification model to identify and block malicious data packets of the kind just described.

To address some of SNORT's inadequacies, this study proposes a new technique to intrusion detection that works in combination with it. In order to address these issues, a variety of data mining approaches are being presented in the answer. The following goals must be accomplished in order to attain the goal:

- Make sure the data set for training and assessment is appropriate since certain machine learning techniques are involved in the solution.
- For new threats, the first line of defence will be a classifier module, built using machine learning algorithms.
- The second layer of classification is needed for traffic that cannot be accurately classified by the first layer, is based on a reasoning module.

II. Methodology and Results

The goal of the comparison research for algorithm classification is to develop a training model for detecting abuse. The results of this comparative study are offered in the form of a perplexity matrix and metrics such as true-positives, false-positives, true-negatives, and false-negatives. It also provided links between expected and predicted classes of KDD'99 intrusion detection data, with an arbitrary split of 66% for training model development and 34% for training model testing for abuse detection

A. Data Set: KDD'99

KDD'99's intrusion detection data collecting is employed. Researchers have tested several intrusion detection methods using this data collection, which is based on a DARPA programme from 1998. Using raw TCP/IP dumps, Sniffer was able to capture all network traffic.

Each instance in the dataset has been assigned to one of 22 assault classes or 1 normal class based on the data set's 41

distinct and continuous properties [6]. This includes the DoS attack, which is also known as a "user of root attack," as well as the "remote to user attack" (Probe).

Feature Selection (CFS) was used to identify the most important data points in the network. The value of each feature is determined by the search algorithm and the classifier function, and a subset of features is provided by CFS (Hall 1999).

B. Classifier Module

Both Nave Bayes and Decision Tree may be used to build a training model that can be used to identify abuse.

1) Naïve Bayes: Using probabilistic inference, Bayesian reasoning may be used in decision-making in situations where previous occurrences are utilised to predict future events [2]. Using the Bayes Theorem, we can calculate the posterior likelihood using the formulas P(q|c), P(c), and P(s|y). According to the Naive Bayesian Classifier, one predictor's influence on a given class (c) is independent of the effect of another predictor (y) [12]. Conditional freedom is granted in this way.

The Bayes algorithm explains the following:

$$P(s|y) = \frac{P(q|c)P(c)}{P(y)} \quad (1)$$

$$P(s|Y) = P(q1\ |c)*P(q2\ |c)*...P(qn\ |c)*P(c) \quad (2)$$

2) Decision tree: In a decision tree, the current node's choice promotes the next node's decision in a sequence of decisions [4] Open-source version of the C4.5 decision tree method – J48 [4]– is accessible through Weka [7]. J48 accepts a wide range of data kinds as input, including nominal, textual, and numeric, but it is also quite inefficient.

The algorithm constructs a decision tree starting from a training set T S, which is a set of cases, or tuples in the database terminology. Each case specifies a value for a collection of attributes and for a class [5]. Each attribute may have either discrete or continuous values. Moreover, the special value unknown is allowed, to denote unspecified values. The class may have only discrete values."

The algorithm works as

- The algorithm operates over a collection of training instances, T.
- If all occurrences of T is in class K.
  - Then create a T and an end node.
  - Select a characteristic S. Create a division node as well.
- Instant T's value for attribute S is divided into a subset (U1..n).
- Recursively apply the method to each of the T subgroups.

3) Experiment: Data from the KDD'99 intrusion detection training set was utilised in our investigation, and a complete KDD dataset was supplied. 34% of the data gathering, approximated at 150,000 of the famed classified insistences, was utilised for the persistence of these prototypes' effectiveness testing.

Using a two-model development technique, we created the training mode

- All classes in the IDS have been considered as a training model in this approach.
- Malicious and natural classifications are created for the data set of training models in this method.

a) All-Classes Based Model Creation Strategy: [4] Bhargava claim that Decision Tree findings outperform Naive Bayes [2]. Table I shows a Naive Bayes and Decision Trees training model.

The Decision Tree classifiers and Naive Bayes respectively, provide different projected and expected classes, as seen in the Fig. 1 and Fig. 2.

Table II displays the cumulative relative impacts for each classifier using TP and FP measures. The FP findings skew Naive Bayes' TP. However, the Decision Tree regularly produces low FP and high TP.

TABLE I. INCLUDES ALL CLASS MODEL BASED ON THE RESULT OF NAIVE BAYES AND DECISION TREE

Instances Classified	Naïve Bayes	Decision Tree
Correctly	92.45 % (145321)	99.95% (146756)
Incorrectly	7.17% (12543)	0.04% (66)

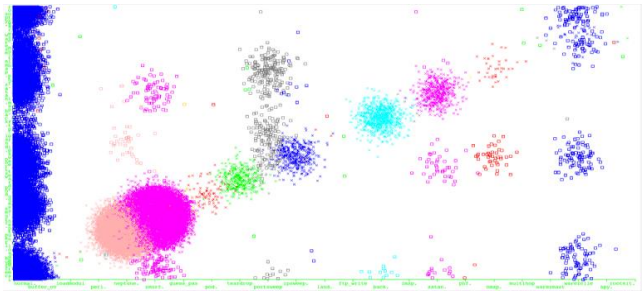

Fig. 1. Shows Naïve Bayes all Class Model Strategy, Predicted vs. Expected Class, Variances.

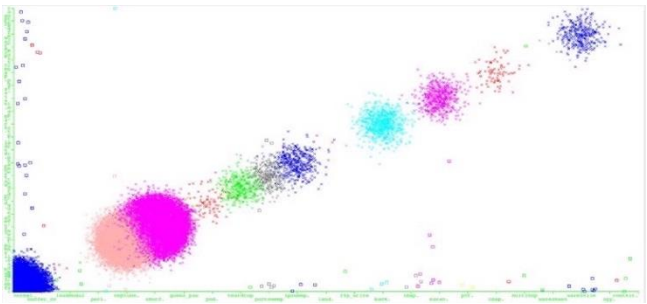

Fig. 2. Shows Decision Tree all Class Model Strategy, Predicted vs. Expected Class, Variances.

TABLE II. ALL-CLASSES MODEL CREATION STRATEGY FOR NAÏVE BAYES AND DECISION TREE

	True-Positive		False-Positive	
	Naïve Bayes	Decision Tree	Naïve Bayes	Decision Tree
Normal	0.617	0.999	0	0
Buffer Overflow	0.462	0.615	0.001	0
Load Module	0.4	0.2	0.001	0
Perl	0	0	0	0
Neptune	0.999	1	0.001	0
Smurf	0.998	1	0	0
Guess Password	0.952	1	0.025	0
Pod	0.987	1	0	0
Teardrop	0.988	0.997	0	0
Portsweep	0.111	0.979	0.01	0
IPsweep	0.97	0.993	0.007	0
FTP Write	0	0.5	0.002	0
Back	0.984	0.996	0	0
IMAP	1	0.4	0	0
Satan	0.894	0.986	0.002	0
PHFF	1	0	0.011	0
Rootkit	0.667	0	0.012	0
Spy	0	0	0	0
Land	0.75	1	0	0

b) Two-Classes based Model Creation Strategy: For a two-class model approach, the results of the Naive Bayes and Decision Tree algorithms are shown in Table III. The usage of Decision Tree-generated training models has been shown to be superior than Naive Bayes.

A comparison of Nave Bayes and Decision Tree Classifiers utilizing a two-class modelling technique shows the difference between predicted and anticipated classes.

Fig. 3 shows the band in the top-left and bottom-right quadrants of the graph shows that the number of incorrectly categorised cases has decreased, resulting in a more reliable model for instance projection.

TABLE III. RESULTS OF TWO-CLASSES MODEL FORMATION STRATEGY USING NAIVE BAYES AND DECISION TREE

Instances Classified	Naïve Bayes	Decision Tree
Correctly	97.98% (148788)	99.97% (149888)
Incorrectly	1.5 % (2430)	0.03% 70

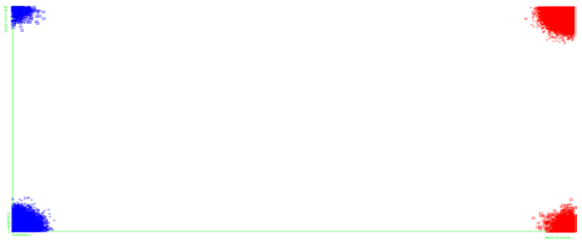

Fig. 3. Naïve Bayes Two Class Model Strategy, Predicted vs. Expected Class, Variances.

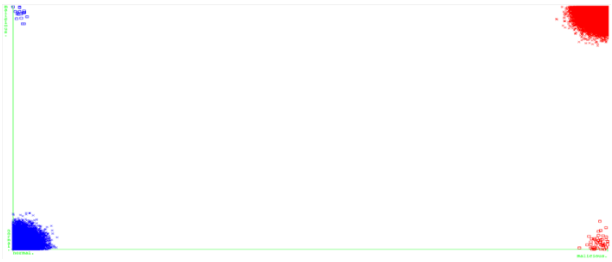

Fig. 4. Shows Decision Tree Two Class Model Strategy, Predicted vs. Expected Class, Variances.

Although the model construction technique changed, the Decision Tree was always attained. In Fig. 4, the number of erroneously identified occurrences decreases in the upper left and lower right quadrants.

TABLE IV. TWO-CLASSES MODEL CREATION STRATEGY ACCURACY / CLASS FOR NAÏVE BAYES AND DECISION TREE

Class	True-Positive		False-Positive	
	Naïve Bayes	Decision Tree	Naïve Bayes	Decision Tree
Normal	0.989	0.999	0.017	0
Malicious	0.983	1	0.011	0.001

Table IV displays the cumulative relative results per classifier for the TP and FP measurements. The Decision Tree has a high true-positive rate and a low false-positive rate.

C. Reasoning Module

In the event that the first stage of classification fails, this mechanism steps in to offer a backup classification stage. A hybrid model of neural network (MLP) and fuzzy logic is used in the reasoning process [8]. This module's output will be a signature, which will be included in the rule base as an addition.

The suggested reasoning tool in this study categories network traffic into two categories: normal (1) and attack (0). To put it another way, the hybrid model is built around two modules neural networks and a fuzzy logic module. It will categorize network traffic as normal if both modules classify it as such, but it will classify it as an attack if either module does so. The neural network has the benefit of being able to operate with both poor and correct data [3]. Fig. 5 shows the hybrid model. When employed in the IDS context, this capability may be used to identify attack patterns that have been provided throughout the training.

It is possible that certain assaults will not be detected by one of the modules, but they may be detected by the other one when utilizing a hybrid method. Furthermore, one module will compensate for weaknesses in other modules' anti-malware detection capabilities. As a result, the false-positive rate for malicious traffic might rise.

1) Neural network: As a computational model of the central nervous system, it can learn and recognise patterns. It has been described as a system that adapts to overt or covert information flows during learning [1].

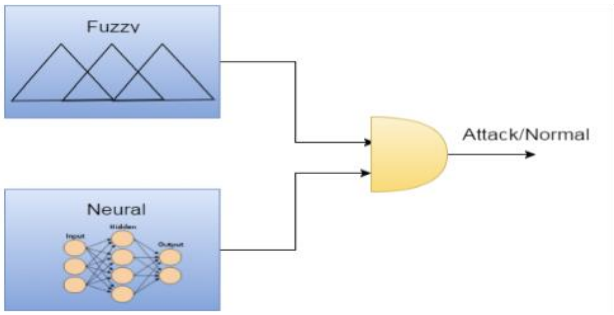

Fig. 5. Hybrid Model Overview.

This design has various tiers (one input layer, several hidden layers, and one output layer). Each layer has neurons, which are processing units. It connects to the mass of the next stratum. In the training phase, back-propagation is used. The input data is given to the neural network, and the output is compared to the intended output. This error is used to alter the weights. The error estimations and weight adjustments follow [1].

$$fj\ (n) = hj\ (n) – uj\ (n) \quad (3)$$

$$\varepsilon(n)= \frac{1}{2} \Sigma j\ f_j^2(n) \quad (4)$$

$$\Delta\ bji(n)= - \mu \frac{\partial \varepsilon(n)}{\partial\ vj\ (n)} ai(n) \quad (5)$$

2) Fuzzy logic: To be a computer model based on human language concepts. Rule-based systems are converted to their mathematical equivalents by fuzzy systems [11]. The fuzzifier, interference engine, rule basis, and fuzzified are all represented in Fig. 6. The following is how fuzzy systems work: [11].

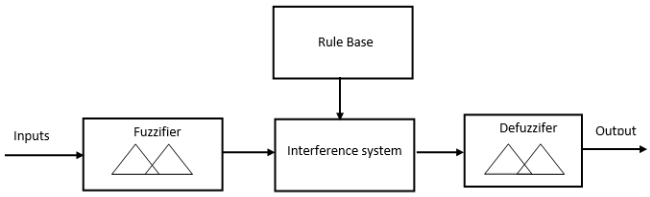

Fig. 6. Fuzzy Logic Components.

- Each input is transformed into a fuzzy input set using the appropriate membership methods.
- The interference engine creates a fuzzed performance based on the criteria supplied.
- The defuzzification membership functions are used to turn the fuzzy output into a crisp value.

Table V lists the inputs that the reasoning module gets from the ip info finder module.

The rule base includes the reasoning for generating the output. The interference engine will employ this set of (if.... then) rules to get a fuzzier result. Table VI demonstrates the reasoning module's criteria for predicting malicious traffic.

On the basis of information gathered, the reasoning module determines whether or not an IP address may be sending malicious traffic. This may be done using a data mining approach, such as clustering or regression. Many factors led to the selection of fuzzy logic for this module. The "if-then" rule form, which is supported by fuzzy logic, may be used to represent the analysis of the acquired data. Aside from that, determining whether or not an IP address is malicious might be tricky in certain cases.

The final output will be considered malicious if it is higher than 0.5, otherwise, it will be considered normal.

3) Experiment: A three-layer neural network module (MLP) is used in our experiment. Whereas the input layer has one neuron, the hidden layer has eight, and the output layer has 10. 10% of the whole KDD'99 IDS and the starting weights were used to train the neural network segment, and the module was trained by constraining the overall mean square to .01 and the maximum number of epochs to 3000.

The KDD'99 IDData collection was used to construct the fuzzy module system:

1) With the exception of 'support,' all of the specified features have been stabilized such that each property has the same range of values (between 0 and 1). This action contributes to the streamlining of the rule-generation process.

2) We have defined three values: U1, U2, U3, where: U1=0.45, U2=0.376, U3=0.76.

3) All features except service were transformed from numerical values into descriptions throughout the iteration through the training data.

$0 \leq$ attribute value $<$ U1 $\rightarrow$ Very Low (UL).

U1 $\leq$ attribute value $<$ U2 $\rightarrow$ Low (L).

U2 $\leq$ attribute value $<$ U3 $\rightarrow$ High (H).

U3 $\leq$ attribute value $\leq 1 \rightarrow$ Very High (UH).

TABLE V. THE REASONING MODULE INPUTS

Input Name	Description
IP Geographic Location	Specifies which country the IP is based at
Is IP in a block list	Specifies whether the IP is found in a block list or not
Is IP an anonymous proxy	Specifies whether the IP is an anonymous proxy or not
IP Rating	An array that shows the IP rating on different DNSBL

TABLE VI. IF THEN RULES USED IN THE REASONING MODULE

If Condition	**Statement**
(IP in a block list)	Possible malicious traffic
(IP country in a black list) AND (IP is an anonymous proxy)	Possible malicious traffic
(IP country in a black list) AND (IP is a TOR exit node)	Possible malicious traffic
(IP Rating is low)	Possible malicious traffic

The performance might be categorized as either normal or offensive. The rule was then written down as follows. The rule was then created in the following form:

if (feature1 is feature_desc AND feature2 is feature2_desc AND ……. feature10 is feature10_desc) then output is output_desc

4) If the previous phase's rule was added to the rule base, it will not be applied to the rule base again. There are a total of 1248 rules applied to the fundamental rule. As illustrated in Fig. 7 and 8, the last stage in the implementation of the fuzzy module was to pick relationship functions for both inputs and outputs.

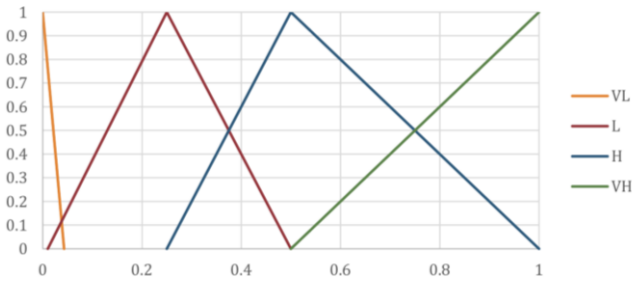

Fig. 7. Shows Relationship Function Excluding 'Service' Feature.

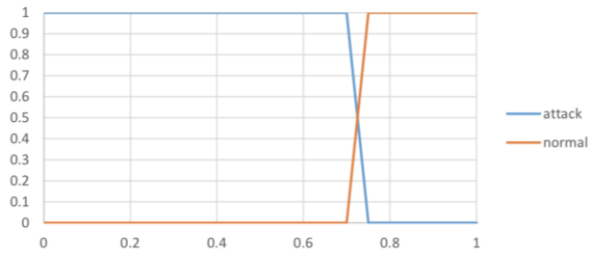

Fig. 8. Shows the Output's Relationship.

Table VII displays the hybrid model's assessment results after neural network training, rule development, and fuzzy module membership function selection.

TABLE VII. INCLUDES A HYBRID MODEL WITH NEURAL NETWORK AND FUZZY LOGIC RESULTS

Class	True-Positive			False-Positive		
	Neural Network	Fuzzy Logic	Hybrid Model	Neural Network	Fuzzy Logic	Hybrid Model
Normal	0.972	0.978	0.952	0.029	0.022	0.048
Malicious	0.966	0.9995	0.9997	0.034	0.0005	0.0003

III. CONCLUSION

Despite the fact that SNORT monitors and detects an attack, the reality is that it is not designed to identify new threats and, as a result, generates a large number of false alarms at a rapid pace. For the first time, data mining approaches have been employed to bring new stages into the solution of previously existing IDS. The suggested model's initial phase accurately detects the vast majority of data. According to Decision Tree, a comparison of two distinct training models using the Naive Bayes and the Decision Tree algorithms shows that the most effective outputs have a higher true-positive score and a greater degree of granularity.

The second stage of the proposed model (reasoning mechanism) was built using a hybrid approach. used a neural network and fuzzy logic to identify new attacks. The rate of intrusion detection rose after deployment.

REFERENCES

[1] Anthony, M. and Bartlett, P. "Neural Network Learning: Theoretical Foundations" 2009, Cambridge University.

[2] Altwaijry, H., Bayesian-based intrusion detection system, in IAENG Transactions on Engineering Technologies 2013, Springer. p. 29-44.

[3] Borah, S. and A. Chakraborty, Towards the Development of an Efficient Intrusion Detection System. International Journal of Computer Applications, 2014. 90.

[4] Bhargava, N., et al., Decision Tree Analysis on J48 Algorithm for Data Mining. International Journal, 2013. 3(6).

[5] Davis, J.J., and A.J. Clark, Data preprocessing for anomaly-based network intrusion detection: A review. Computers & Security, 2011. 30(6): p. 353-375.

[6] Hall, M. (1999) Correlation-based Feature Selection for Machine Learning. The University of Waikato.

[7] Hall, M., et al., The WEKA Data Mining Software: An Update. SIGKDD Explorations, 2009. 11(1).

[8] Kukiełka, P. and Kotulski, Z. (2010) "Adaptation of the neural network-based IDS to new attacks detection," Available from http://arxiv.org/abs/1009.2406 (Access Date: 17 Oct 2014).

[9] Kim, G., S. Lee, and S. Kim, A novel hybrid intrusion detection method, integrated anomaly detection with misuse detection. Expert Systems with Applications, 2014. 41(4): p. 1690-1700.

[10] Kang, D.-K., D. Fuller, and V. Honavar. "Learning misuse and anomaly detection classifiers using a bag of device calls representation." In Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC. 2005. IEEE.

[11] Rajasekaran, S. and Pai, G. "Neural Networks, Fuzzy |Logic, and Genetic Algorithm: Synthesis and Applications" 2003, PHI Learning Pvt. Ltd.

[12] Rawat, R. and A. Jain, Review: Boosting Classifiers for Intrusion Detection. International Journal of Scientific & Engineering Research, 2013. 4(7): p. 1-5.

[13] Roesch, M. SNORT: Lightweight Intrusion Detection for Networks. in LISA. 1999.

[14] Shanmugam, B. "Improved Intrusion Detection System Using Fuzzy Logic for Detecting Anamoly and Misuse Type of Attacks" in Proceedings of the Conference of Soft Computing Pattern Recognition. 2009, pp.212-217.