One Versus All classification in Network Intrusion detection using Decision Tree

Mr. Vilas S. Gaikwad, Dr. Prakash J. Kulkarni

Department of Computer Science and Engineering, Walchand College of Engineering, Sangli, India

Abstract- One-versus-all (OVA) classification is one of the multiclass classification problems as well as it is a binary classifier. On the basis of this, we propose a network intrusion detecting system for the security of computers and networks. In this paper, we present a new learning algorithm for detection of a network intrusion using one versus all decision tree algorithm, that differentiates attacks from normal behaviors and identifies different types of intrusions. Experimental results on the KDD99 dataset of network intrusion. The proposed learning algorithm achieved very good result in form of detection rate (DR) in comparison with other existing methods.

Index Terms- ID3, OVA decision tree, intrusion detection

I. INTRODUCTION

OVA is the k-class classification problem, OVA learns one binary classifier for each class to distinguish instances of this class from instances of the remaining (k-1) classes. In computer network technology expand for communications and commerce in recent times, the rate of intrusions increase rapidly every year. Intrusion detection is the process of identifying actions that attempt to compromise the confidentiality, integrity or availability of computers or networks. The use of data mining algorithms for detecting intrusions is now considered to build efficient and adaptive intrusion detection systems that detect unauthorized activities of a computer system or network. The system was first introduced by James P. Anderson in 1980 [1], and later in 1986, Dr. Dorothy Denning proposed several models for IDS based on statistics, Markov chains, time-series, etc[2] Anomaly based intrusion detection using data mining algorithms such as decision tree (DT), naïve Bayesian. Classifier (NB), neural network (NN), support vector machine (SVM), k-nearest neighbors (KNN), fuzzy logic model, and genetic algorithm have been widely used by researchers to improve the performance of IDS [3]. However, today's commercially available IDS are signature based. Signature based IDS performs pattern matching techniques to match an attack pattern corresponding to known attack patterns in the database and produces very low false positives (FP), but it requires regular updates of rules or signatures and not capable of detecting unknown attacks. On the other hand, anomaly based IDS builds models of normal behavior and automatically detects anomalous behaviors. Anomaly detection techniques identify new types of intrusions as deviations from normal usage [4], but the drawback of the techniques is the rate of false positives (FP). The use of data mining algorithms for anomaly based IDS are to include an intelligent agent in the system that can detect the known and unknown attacks or intrusions. Intrusion detection systems (IDS) gather and analyze information from a variety of systems and network sources for signs of intrusions. IDS can be host-based or network based systems. Host-based IDS located in servers to examine the internal interfaces and network-based IDS monitor the network traffics for detecting intrusions. Network-based IDS performs packet logging, real-time traffic analysis of IP Network, and tries to discover if an intruder is attempting to break into the network. The major functions performed by IDS are: (1) monitoring users and systems activity, (2) auditing system configuration, (3) assessing the data files, (4) recognizing known attacks (5) identifying abnormal activities. A variety of IDS have been employed for protecting computers and networks in last decades, but still there some issues that should be consider in the current IDS like low detection accuracy, unbalanced detection rates for different types of attacks, and high false positives. In this paper, we proposed a new decision tree based learning algorithm for classifying different types of network attacks, which improves the detection rates (DR) and reduces false positives (FP) on the basis of one versus all criteria using KDD99 benchmark network intrusion detection dataset in comparison with other existing methods.

II. ANOMALY BASED INTRUSION DETECTION

The concept of IDS began with Anderson's seminal paper [1]; he introduced a threat classification model that develops a security monitoring surveillance system based on detecting anomalies in user behavior. In Anderson's model threats are classified as external penetrations, internal penetrations, and misfeasance. External perceptions are intrusions in computer system by outside intruders, who do not have any authorized access to the system that they attack. Internal perceptions are intrusions in computer system by inside intruders. Inside intruders are users in the network and have some authority, but seek to gain additional ability to take action without appropriate authorization. Misfeasance is defined as the misuse of authorized access of both to the system and to its data based on statistics. In Denning model, user's behavior that deviates sufficiently from the normal behavior is considered anomalous. In the early some Intrusion Detection Expert System that could operate in real time for continuous monitoring of user activity or could run in a batch mode for periodic analysis of the audit data, an audit data is a record of activities generated by the operating system that are logged to a file in chronologically sorted order. And some intrusion detection system enables the system to compare the

current activities of the user/system/network with the audited intrusion detection variables stored in the profile and then raise an alarm if the current activity is sufficiently far from the stored audited activity. A statistical anomaly-based IDS was proposed [5], which used both user and group based anomaly detection strategies. In this system, a range of values were considered normal for each attribute and during a session if an attribute fell outside the normal range then an alarm raised. It was designed to detect six types of intrusions: attempted break-ins by unauthorized users, masquerade attacks, penetration of the security control system, leakage, denial of service, and malicious use Statistical Packet. Forrest et al. proposed an analogy between the human immune system and intrusion detection that involved analyzing a program's system call sequences to build a normal profile [6], which analyzed several UNIX, based programs like send mail, ipr, etc. If the sequences deviated from the normal sequence profile then it considered as an attack. The system they developed was only used off-line using previously collected data and used a quite simple table-look up algorithm to learn the profiles of programs then developed an anomaly based intrusion detection system that employed naïve Bayesian network to perform intrusion detecting on traffic bursts[7]. Lee et al. [8] proposed classification based anomaly detection using inductive rules to characterize sequences occurring in normal data. The Fuzzy Intrusion Recognition Engine (FIRE) using fuzzy logic that process the network input data and generate fuzzy sets for every observed feature and then the fuzzy sets are used to define fuzzy rules to detect individual attacks [9]. FIRE creates and applies fuzzy rules to the audit data to classify it as normal or anomalous. In another paper the anomalous network traffic detection with self organizing maps using DNS and HTTP services for network based IDS that the neurons are trained with normal network traffic then real time network data is fed to the trained neurons [10], if the distance of the incoming network traffic is more than a preset threshold then it rises an alarm.

III. PROPOSED OVA LEARNING ALGORITHM

A. *Decision Tree Learning*

The decision tree (DT) is very powerful and popular data mining algorithm for decision-making and classification problems. It has been using in many real life applications like medical diagnosis, radar signal classification, weather prediction, credit approval, and fraud detection etc. DT can be constructed from large volume of dataset with many attributes, because the tree size is independent of the dataset size. A decision tree has three main components: nodes, leaves, and edges. Each node is labeled with a one attribute versus all attributes by which the data is to be partitioned. Each node has a number of edges, which are labeled according to possible values of the attribute. An edge connects either two nodes or a node and a leaf. Leaves are labeled with a decision value for categorization of the data. To make a decision using a decision Tree, start at the root node and follow the tree down the branches until a leaf node representing the class is reached. Each decision tree represents a rule set, which categorizes data according to the attributes of dataset. The DT building algorithms may initially build the tree and then prune it for more effective classification. With pruning technique, portions of the tree may be removed or combined to reduce the overall size of the tree. The time and space complexity of constructing a decision tree depends on the size of the data set, the number of attributes in the data set, and the shape of the resulting tree. Decision trees are used to classify data with common attributes. The ID3 algorithm builds decision tree using information theory, which choose splitting attributes from a data set with the highest information gain [11]. The amount of information associated with an attribute value is related to the probability of occurrence. The concept used to quantify information is called entropy, which is used to measure the amount of randomness from a data set. When all data in a set belong to a single class, compare to the other there is no uncertainty, and then the entropy is zero. The objective of decision tree classification is to iteratively partition the given data set into subsets where all elements in each final subset belong to the same class. The entropy calculation is shown in equation 1. Given probabilities $p_1, p_2, \cdots, p_s$ for different classes in the data set

Entropy $(p_1, p_2, \ldots p_s) = \sum_{i=1}^{s} p_i \log(1/p_i)$ ……(1)

Given a data set, *D, H (D)* finds the amount of entropy in class based subsets of the data set. When that subset is split into s new subsets *S = {D1, D2… Ds}* using some attribute, we can again look at the entropy of those subsets. A subset of data set is completely ordered and does not need any further split if all examples in it belong to the same class. The ID3algorithm calculates the information gain of a split by using following equation and chooses that split which provides maximum information gain.

$Gain\ (D,S) = H(D) - \sum_{i=1}^{s} p(Di) H(Di)$ ……(2)

The C4.5 algorithm [12], which is the upgraded version ofID3 algorithm uses highest *Gain* (Classification and Regression Trees) is a process of generating a binary tree for decision making [13]. CAR Handles missing data and contains a pruning strategy. The SPRINT (Scalable Parallelizable Induction of Decision Trees) algorithm uses an impurity function called *gini* index to find the best split [14].

$gini\ (D) = 1 - \sum p_j^2$

Where, *pj* is the probability of class *Cj* in data set *D*. The goodness of a split of *D* into subsets *D1* and *D2*is defined by

$gini_{split}(D) = n1/n(gini(D1)) + n2/n(gini(D2))$ ……(3)

B. *Proposed Learning Algorithm*

In a given dataset, first the algorithm initializes the weights for each example of dataset; *Wi* equal to 1/*n*, where *n* is the number of total examples in dataset. Then the algorithm estimates the prior probability *P(Cj)* for each class by summing the weights that how often each class occurs in the dataset. Also for each attribute, *Ai*, the number of occurrences of each attribute value *Aij* can be counted by summing the weights to determine *P(Aij)*. Similarly, the conditional probabilities *P(Aij | Cj)* are estimated for all values of attributes by summing the weights how often each attribute value occurs in the class *Cj*. After that the

algorithm uses these probabilities to update the weights for each example in the dataset. It's performed by multiplying the probabilities of the different attribute values from the examples. Suppose the example *ei* has independent attribute values {*Ai1, Ai2,…,Aip*}. We already
Know *P(Aik | Cj)*, for each class *Cj* and attribute *Aik*. We then estimate *P(ei | Cj)*by

$$P(ei \mid Cj) = P(Cj)\ \Pi k=1 \rightarrow p\ P(Aij \mid Cj)$$

To update the weight, we can estimate the likelihood of *ei* in each class *Cj*. The probability that *ei* is in a class is the product of the conditional probabilities for each attribute value. The next probability *P(Cj | ei)* is then found for each class. Now the weight of the example is updated with the highest next probability for that example. Finally, the algorithm calculates the information gain by using updated weights and builds a tree for decision making. Following describes the main procedure of algorithms:
Algorithm: Tree Construction
Input: dataset *D*
Output: decision tree *T*
Procedure:
1. Initialize all the weights in *D*, Wi=1/*n*, where *n* is the total number of the examples.
2. Calculate the prior probabilities *P(Cj)* for each class *Cj* In *D*

$$.P(Cj) = \frac{\sum_{ci} w_i}{\sum_{i=1}^{n} w_i}$$

3. Calculate the conditional probabilities *P (Aij | Cj)* for each attribute values in *D*.

$$P(Aij \mid Cj) = \left(\frac{P(Aij)}{\sum_{ci} w_i} \right)$$

4. Calculate the next probabilities for each example in *D*.
$P(ei \mid Cj) = P(Cj)\ \Pi\ P(Aij \mid Cj)$
5. Update the weights of examples in *D* with Maximum Possibility (MP)of next probability *P(Cj|ei)*;
$Wi = PML\ (Cj|ei)$
6. Find the splitting attribute with highest information gain using the updated weights, *Wi* in *D*.
7. *T* = Create the root node and label with splitting attribute.
8. For each branch of the *T*, *D* = database created by applying splitting predicate to *D*, and continue steps
1 to 7 until each final subset belongs to the same class or leaf node created.
9. When the decision tree construction is completed the algorithm terminates.

IV. EXPERIMENTAL ANALYSIS

A. Intrusion Detection Dataset

The KDD99 dataset was used in the 3rd International Knowledge Discovery and Data Mining Tools Competition for building a network intrusion detector, a predictive model capable of distinguishing between intrusions and normal network connections [16]. In 1998, DARPA intrusion detection evaluation program, a simulated environment was set up to acquire raw TCP/IP dumps data for a local-area network (LAN) by the MIT Lincoln Lab to compare the performance of various intrusion detection methods. It was operated like a real environment, but being blasted with multiple intrusion attacks and received much attention in the research community of adaptive intrusion detection. TheKDD99 dataset contest uses a version of DARPA98 dataset .In KDD99 dataset, each example represents attribute values of a class in the network data flow, and each class is labeled either normal or attack. The classes in KDD99 dataset categorized into five main classes (one normal class and four main intrusion classes: probe, DOS, U2R, and R2L).
1) Normal connections are generated by simulated daily user behavior such as downloading files, visiting web pages.
2) Denial of Service (DoS) attack causes the computing power or memory of a victim machine too busy or too full to
Handle legitimate requests. DoS attacks are classified based on the services that an attacker renders unavailable to legitimate users like apache2, land, mail bomb, back, etc.
3) Remote to User (R2L) is an attack that a remote user gains access of a local user/account by sending packets to a Machine over a network communication, which include send mail, and X lock.
4) User to Root (U2R) is an attack that an intruder begins with the access of a normal user account and then becomes a root-user by exploiting various vulnerabilities of the system .Most common exploits of U2R attacks are regular buffer-overflows, load-module, Fd-format, and Ffb -config.
5) Probing (Probe) is an attack that scans network together information or finds known vulnerabilities. An intruder with a map of machines and services that are available on a network can use the information to look for exploits. In KDD99 dataset these four attack classes (DoS, U2R, R2L, and probe) are divided into 22 different attack classes that tabulated in Table I.

TABLE I. DIFFERENT TYPES OF ATTACKS IN KDD99 DATASET

Main Attack Classes	Attack Classes
Denial of Service (DoS)	back, land, Neptune, pod, smurt, teardrop
Remote to User (R2L)	ftp_write, guess_passwd, imap, multihop, phf,spy, warezclient, warezmaster
User to Root (U2R)	buffer_overflow, perl, load module, rootkit
Probing	ipsweep, nmap, ports weep, Satan

There are 41 input attributes in KDD99 dataset for each network connection that have either discrete or continuous values and divided into three groups. The first group of attributes is the basic features of network connection, which include the duration, prototype, service, number of bytes from source IP addresses or from destination IP addresses and some flags in TCP connections. The second group of attributes in KDD99 is composed of the content features of network connections and the third group is composed of the statistical features that are computed either by a time window or a window of certain kind

of connections. Table II shows the number of examples of 10% training examples and 10%testing examples in KDD99 dataset. There are some new attack examples in testing data, which is no present in the training data

TABLE II: NUMBER OF EXAMPLES IN TRAINING AND TESTING KDD99 DATA

Attack Types	Training Examples	Testing Examples
Normal	97277	97277
Denial of Service	391458	391458
Remote to User	1126	1126
User to Root	52	52
Probing	4107	4107
Total Examples	494020	494020

B. Experimental Analysis

In order to evaluate the performance of proposed algorithm for network intrusion detection, we performed 5-classclassification using KDD99 dataset. All experiments were performed using an Intel Core i5 Processor with 2 GB of RAM. The results of the comparison of proposed algorithm with ID3

TABLE III: COMPARISON OF THE RESULTS USING 41 ATTRIBUTES

Method	Normal	Prob e	DOS	U2R	R2L
Proposed Algorithm (DR %)	98.76	98.21	98.55	98.11	97.16
ID3 (DR %)	98.53	97.85	97.51	49.21	94.65

TABLE IV: COMPARISON OF THE RESULTS USING 15 ATTRIBUTES

Method	Normal	Prob e	DOS	U2R	R2L
Proposed Algorithm (DR %)	99.76	98.91	99.55	98.81	98.16
ID3 (DR %)	99.03	97.92	98.21	59.21	95.65

V. CONCLUSION

This paper presents learning algorithm for anomaly based network intrusion detection using decision tree, on the basis of one versus all criteria which adjusts the weights of dataset based on probabilities and split the dataset into sub-dataset until all the sub-dataset belongs to the same class. In this paper, we developed the performance of IDS using decision tree. In conventional decision tree algorithm weights of every example is set to equal value which contradicts general intuition, but in our approach weights of every example change based on probability. The experimental results on KDD99 benchmark dataset proposed algorithm achieved high detection rate on different types of network attacks. The future part of this work will focus on parallelism of one versus all problems.

ACKNOWLEDGMENT

Support for this work received from Department of Computer Science and engineering Walchand College of Engineering Sangli,Maharastra, India

REFERENCES

[1] James P. Anderson, “Computer security threat monitoring andsurveillance,” Technical Report 98-17, James P. Anderson Co., FortWashington, Pennsylvania, USA, April 1980.

[2] Dorothy E. Denning, “An intrusion detection model,” IEEE Transactionon Software Engineering, SE-13(2), 1987, pp. 222-232.

[3] Barbara, Daniel, Couto, Julia, Jajodia, Sushil, Popyack, Leonard, Wu,andNingning, “ADAM: Detecting intrusion by data mining,” IEEEWorkshop on Information Assurance and Security, West Point, New York, June 5-6, 2001.

[4] Lazarevic, A., Ertoz, L., Kumar, V., Ozgur,.A., Srivastava, and J., “Acomparative study of anomaly detection schemes in network intrusion detection,” In Proc. of the SIAM Conference on Data Mining, 2003.

[5] S.E. Smaha, and Haystack, “An intrusion detection system,” in Proc.Ofthe IEEE Fourth Aerospace Computer Security ApplicationsConference, Orlando, FL, 1988, pp. 37-44.

[6] S. Forrest, S.A. Hofmeyr, A. Somayaji, T.A. Longstaff, “A sense of selffor Unix processes,” in Proc. of the IEEE Symposium on Research inSecurity and Privacy, Oakland, CA, USA, 1996, pp. 120-128.

[7] A. Valdes, K. Skinner, “Adaptive model-based monitoring for cyberattack detection,” in Recent Advances in Intrusion Detection Toulouse,France, 2000, pp. 80-92.

[8] W. Lee, S.J. Stolfo, “Data mining approaches for intrusion detection,” In Proc. of the 7th USENIX Security Symposium (SECURITY-98),Berkeley, CA, USA, 1998, pp. 79-94.

[9] J.E. Dickerson, J.A. Dickerson, “Fuzzy network profiling for intrusiondetection,” In Proc. of the 19th International Conference of the NorthAmerican Fuzzy Information Processing Society (NAFIPS), Atlanta,GA, 2000, pp. 301-306

[10] M. Ramadas, S.O.B. Tjaden, “Detecting anomalous network traffic with self-organizing maps,” In Proc. of the 6th International Symposium onRecent Advances in Intrusion Detection, Pittsburgh, PA, USA, 2003, pp.36-54.

[11] J. R. Quinlan, “Induction of Decision Tree,” Machine Learning Vol. 1,pp. 81-106, 1986

[12] J. R. Quinlan, “C4.5: Programs for Machine Learning,” MorganKaufmann Publishers, San Mateo, CA, 1993.

[13] L. Breiman, J. H. Friedman, R. A. Olshen and C.J. Stone, “Classification and Regression Trees,” Statistics probability series, Wadsworth, Belmont, 1984.

[14] John Shafer, RakeshAgarwal, and Manish Mehta, “SPRINT: A Scalable Parallel Classifier for Data Maining,” in Proceedings of the VLDB conference, Bombay, India, September 1996.

[15] Hashemi, S.; Ying Yang; Mirzamomen, Z.; Kangavari, M.; "Adapted One-versus-All Decision Trees for Data Stream Classification," *Knowledge and Data Engineering, IEEE Transactions on* ,vol.21,no.5,pp.624-637,May 2009 doi: 10.1109/TKDE.2008.181

[16] The KDD Archive KDD99 cup dataset, 1999 http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

AUTHORS

First Author – Mr. Vilas S. Gaikwad, Department of Computer Science and Engineering, Walchand College of Engineering, Sangli, India, Email-id: vilasgaikwad11@gmail.com

Second Author – Prof. Dr. Prakash J. Kulkarni, Department of Computer Science and Engineering, Walchand College of Engineering, Sangli, India, Email-id: pjk_walchand@rediffmail.com