ISSN:0975-9646

Network Intrusion Detection Using Improved Decision Tree Algorithm

K.V.R. Swamy, K.S. Vijaya Lakshmi
Department Of Computer Science and Engineering
V.R.Siddhartha Engineering College, Vijayawada, Andhra Pradesh, India

***Abstract* – Intrusion detection involves a lot of tools that are used to identify different types of attacks against computer systems and networks. With the development of network technologies and applications network attacks are greatly increasing both in number and severe. Open source and commercial network intrusion detection tools are not able to predict new type of attacks based on the previous attacks. So, data mining is one of the methods used in IDS (Intrusion Detection System). In recent years data mining based network intrusion detection system has been giving high accuracy and good detection on different types of attacks. In this paper, the performance of the data mining algorithms like C4.5 and improved C4.5 are being used in order to detect the different types of attacks with high accuracy and less error prone.**

***Keywords*- C4.5 Decision Tree; Improved C4.5 Decision Tree; Intrusion detection system.**

I. INTRODUCTION

Nowadays, many organizations and companies use Internet services as their communication and marketplace to do business such as at EBay and Amazon.com website. Together with the growth of computer network activities, the growing rate of network attacks has been advancing, impacting to the availability, confidentiality, and integrity of critical information data. Therefore a network system must use one or more security tools such as firewall, antivirus, IDS and Honey Pot to prevent important data from criminal enterprises.

A network system using a firewall only is not enough to prevent networks from all attack types. The firewall cannot defense the network against intrusion attempts during the opening port. Hence a Real-Time Intrusion Detection System (RT-IDS), shown in Fig 1, is a prevention tool that gives an alarm signal to the computer user or network administrator for antagonistic activity on the opening session, by inspecting hazardous network activities [1].

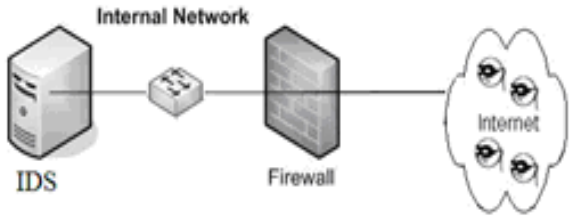

Fig 1: Intrusion detection system environment

IDSs have gained acceptance as a necessary addition to every organization's security infrastructure despite the documented contributions intrusion detection technologies make to system security, in many organizations one must still justify the acquisition of IDSs. We may use IDSs to prevent problem behaviors by increasing the perceived risk of discovery of those who would attack or abuse the system.

There are two general categories of attacks which intrusion detection technologies attempt to identify - anomaly detection and misuse detection. Anomaly detection identifies activities that vary from established patterns for users, or groups of users. Anomaly detection typically involves the creation of knowledge bases that contain the profiles of the monitored activities. The second general approach to intrusion detection is misuse detection. This technique involves the comparison of a user's activities with the known behaviors of attackers attempting to penetrate a system. While anomaly detection typically utilizes threshold monitoring to indicate when a certain established metric has been reached, misuse detection techniques frequently utilize a rule-based approach. When applied to misuse detection, the rules become scenarios for network attacks. The intrusion detection mechanism identifies a potential attack if a user's activities are found to be consistent with the established rules. The use of comprehensive rules is critical in the application of expert systems for intrusion detection [2].

There are many methods applied into intrusion detection, such as methods based on statistics, methods based on data mining, methods based on machine learning and so on. In recent years, data mining technology is developing rapidly and increasingly mature. Now it is gradually applied to the intrusion detection field, and has made a number of important achievements at home and abroad. The basic principles of intrusion detection based on data mining are as follows: Firstly intelligently analyze and deal with security audit data from different data sources(such as host-based, network-based, alarm-based), this can help system generate intrusion rules and establish anomaly detection model by extracting regularity of data; Then use these knowledge to discriminate new network behaviors. The main methods are: classification analysis, clustering analysis, genetic algorithm, neural networks, association rule mining, sequential pattern mining, and outlier detection and so on. Decision tree technology is an intuitionist and straightforward classification method. It has great advantage in extracting features and rules. Therefore applying decision tree technology into intrusion

detection is of great significance [3]. Locations of Intrusion Detection Systems in Networks:Usually an intrusion detection system captures data from the network and applies its rules to that data or detects anomalies in it. Depending upon the network topology, the type of intrusion activity (i.e. internal, external or both), and our security policy (what we want to protect from hackers), IDSs can be positioned at one or more places in the network . For example, if we want to detect only external intrusion activities, and we have only one router connecting to the Internet, the best place for an intrusion detection system may be just inside the router or a firewall. On the other hand, if we have multiple paths to the internet, and we want to detect internal threats as well, we should place one IDS box in every network segment. Fig. shows typical locations where you can place an intrusion detection system.

II. RELATED WORK

In his paper, Except for the information gain measure and its improved versions, Lopez de Mantaras[4] presented a distance-based attribute selection measure. His experimental study proves that the distance based measure is not biased toward attributes with large numbers of values, and avoids the practical issues towards the gain ratio measure. Mingers[5] provides an experimental study of the relative accuracy of different attribute selection measures in the decision tree in order to overcome the bias in the tuples. Nageswara Rao, Dr. D. Rajya Lakshmi, Prof T. Venkateswara Rao et at[6] proposed robust statistical preprocessor in order to improve the accuracy. But the limitation in that paper is existing c45 does not handle when the dataset is large. An expert system consists of a set of rules that encode the knowledge of a human "expert". These rules are used by the system to make conclusions about the security-related data from the intrusion detection system. Expert systems permit the incorporation of an extensive amount of human experience into a computer application that then utilizes that knowledge to identify activities that match the defined characteristics of misuse and attack. Unfortunately, expert systems require frequent updates to remain current. While expert systems offer an enhanced ability to review audit data, the required updates may be ignored or performed infrequently by the administrator. At a minimum, this leads to an expert system with reduced capabilities. At worst, this lack of maintenance will degrade the security of the entire system by causing the system's users to be misled into believing that the system is secure, even as one of the key components becomes increasingly ineffective over time. Rule-based systems suffer from an inability to detect attacks scenarios that may occur over an extended period of time. While the individual instances of suspicious activity may be detected by the system, they may not be reported if they appear to occur in isolation. Intrusion scenarios in which multiple attackers operate in concert are also difficult for these methods to detect because they do not focus on the state transitions in an attack, but instead concentrate on the occurrence of individual elements. Any division of an attack either over time or among several seemingly unrelated attackers is difficult for these methods to detect. Rule-based systems also lack flexibility in the rule-to-audit record representation. Slight variations in an attack sequence can affect the activity-rule comparison to a degree that the intrusion is not detected by the intrusion detection mechanism. While increasing the level of abstraction of the rule-base does provide a partial solution to this weakness, it also reduces the granularity of the intrusion detection device[11].

III. PROPOSED ARCHITECTURE

Following framework gives the overall description about the proposed approach. In this framework,KDD dataset[7] is used as training data for classification purpose.

Proposed framework has following algorithms.

1) Min Max Normalization
2) Decision Tree Algorithms.

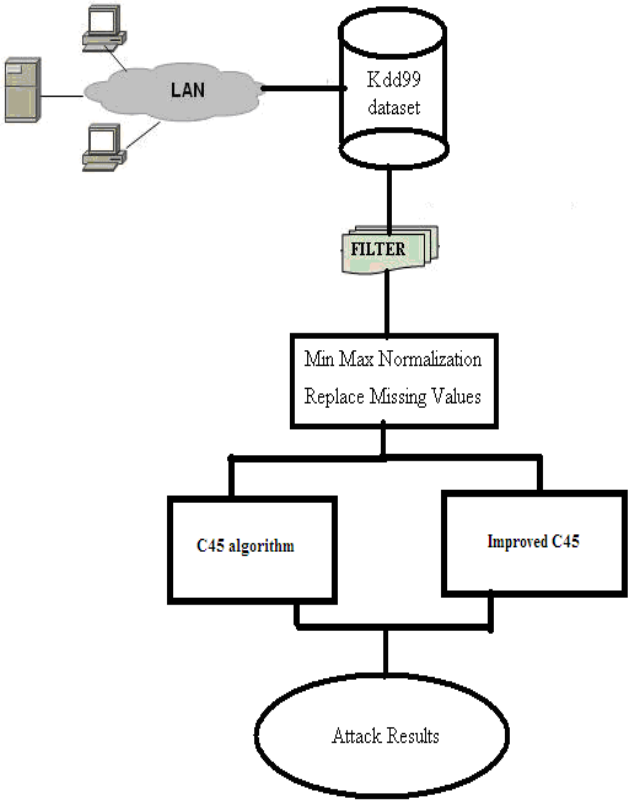

Fig 2: Proposed Framework

A. KDD Dataset

The KDD Cup 1999 dataset was derived from the 1998 DARPA Intrusion detection evaluation program prepared and managed by MIT Lincoln Laboratory. The dataset was a collection of simulated raw TCP dump data over a period of nine weeks. There are 4,898,430 labeled and 311,029 unlabeled connection records in the dataset [8]. The labeled connection records consist of 41 attributes: 7 symbolic and 34 numeric. The complete listing of the set of features in the dataset is given in Table 1.

TABLE I: List of attributes in KDD dataset

No	Name of the attribute	No	Name of the attribute
1	duration	22	is_guest_login
2	protocol_type	23	count
3	service	24	srv_count
4	flag	25	serror_rate
5	src_bytes	26	srv_serror_rate
6	dst_bytes	27	rerror_rate
7	land	28	srv_serror_rate
8	wrong_fragment	29	same_srv_rate
9	urgent	30	diff_srv_rate
10	hot	31	srv_diff_host_rate
11	num_failed_logins	32	dst_host_count
12	logged_in	33	dst_host_srv_count
13	num_compromised	34	dst_host_same_srv_rate
14	root_shell	35	dst_host_diff_srv-rate
15	su_attempted	36	dst_host_same_srv_port_rat e
16	num_root	37	dst_host_srv_diff_host_rate
17	num_file_creations	38	dst_host_serror_rate
18	num_shells	39	dst_host_srv_serror_rate
19	num_access_files	40	dst_host_rerror_rate
20	num_outbound_cmd	41	dst_host_srv_rerror_rate
21	is_host_login		

B. Data Transformation

In data transformation, the data are transformed or consolidated into forms appropriate for mining. Min-max normalization performs a linear transformation on the original data. Suppose that minA and maxA are the minimum and maximum values of an attribute A. Min-max normalization maps a value, v, of A to v0 in the range [new_minA, new_maxA] by computing

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A.$$

Min-max normalization preserves the relationships among the original data values. It will encounter an "out-of-bounds" error if a future input case for normalization falls outside of the original data range for A.

C. Data Preprocessing

Incomplete, noisy, and inconsistent data are commonplace properties of large real world databases and data warehouses. Incomplete data can occur for a number of reasons. Attributes of interest may not always be available. Other data may not be included simply because it was not considered important at the time of entry. Relevant data may not be recorded due to a misunderstanding, or because of equipment malfunctions. Data that were inconsistent with other recorded data may have been deleted. Furthermore, the recording of the history or modifications to the data may have been overlooked. Missing data, particularly for tuples with missing values for some attributes, may need to be inferred. There are many possible reasons for noisy data (having incorrect attribute values). Data cleaning (or data cleansing) routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data. Handling Missing Values: The attribute mean or stddev to fill in the missing value.

D. C45 ALGORITHM

Algorithm**:** Geneate_decision_tree
Input**:** Data partition, D, which is a set of training tuples and their associated class labels. Attribute_list, the set of candidate attributes. Attribute_selection_method, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a splitting_attribute and, possibly, either a split point or splitting subset.
Output**:** a decision tree
Method:
(1) create a node N;
(2) if tuples in D are all of the same class, C then
(3) return N as a leaf nod labeled with the class C;
(4) If attribute_list is empty then
(5) Return N as a leaf node labeled with the majority class in D; //majority voting
(6) Apply attribute_seletion_method (D, arrtibute_list) to find the "best" splitting_criterion;
(7)Label node N with splitting_criterion;
(8)If splitting_attribute is discrete-valued and
Multiway splits allowed then // not restricted to binary trees
(9) attribute_list→attribute_list - splitting_attribute; //remove splitting_attribute
(10) for each outcome j of splitting_criterion // partition the tuples and grow sub-tees for each partition
(11) Let Dj be the set of a data tuples in D satisfying outcome j; // a partition
(12) If Dj is empty then
(13) Attach a leaf labeled with the majority class in D to node N;
(15) Else attach the node returned by Geneate_decision_tree (Dj, attribute list) to node N;
(16) Return N;

E. IMPROVED C45

(1) create a node N;
(2)if tuples in D are all of the same class, C then
(3) return N as a leaf node labeled with the class C;
(4) if attribute list is empty then
(5) return N as a leaf node labeled with the majority class in D; // majority voting
(6) apply Attribute selection to each attribute(L, attribute list) to find the "best" splitting criterion;

Gain measures how well a given attribute separates training examples into targeted classes. The one with the highest information is selected. Given a collection S of c outcomes The expected information needed to classify a tuple in D is given by

Modified Information or entropy is given as

$$\text{ModInfo(D)} = -S_i \sum_{i=1}^{m} l \log \sqrt{S_i} \quad \text{,m different classes}$$

$$\text{ModInfo(D)} = -S_i \sum_{i=1}^{2} l \log \sqrt{S_i}$$

$$= -S_1 \log \sqrt{S_1} - S_2 \log \sqrt{S_2}$$

Where S_1 indicates set of samples which belongs to target class 'anamoly', S_2 indicates set of samples which belongs to target class 'normal'.

Information or Entropy to each attribute is calculated using

$$Info_A(D) = \sum_{i=1}^{v} |D_i| / |D| \times ModInfo(D_i)$$

The term Di /D acts as the weight of the jth partition. ModInfo(D) is the expected information required to classify a tuple from D based on the partitioning by A. Information gain is defined as the difference between the original information requirement) and the new requirement .That is,

$$Gain(A) = Mod\inf o(D) - \inf o_A(D)$$

(7)Label node N with splitting_criterion;
(8)If splitting_attribute is discrete-valued and
Multiway splits allowed then // not restricted to binary trees
(9) attribute_list→attribute_list - splitting_attribute; //remove splitting_attribute
(10) for each outcome j of splitting_criterion // partition the tuples and grow sub-tees for each partition
(11) Let Dj be the set of a data tuples in D satisfying outcome j; // a partition
(12) If Dj is empty then
(13) Attach a leaf labeled with the majority class in D to node N;
(15) Else attach the node returned by Geneate_decision_tree (Dj, attribute list) to node N;
(16) Return N;

IV. EXPERIMENTAL RESULTS

All experiments were performed in a one-year-old computer with the configurations Intel(R) Core(TM)2 CPU 2.13GHz, 2 GB RAM, and the operation system platform is Microsoft Windows XP Professional (SP2). The dataset to be used in our experiments in KDD99 labeled dataset. The main reason we use this dataset is that we need relevant data that can easily be shared with other researchers, allowing all kinds of techniques to be easily compared in the same baseline. The common practice in intrusion detection to claim good performance with "live data" makes it difficult to verify and improve pervious research results, as the traffic is never quantified or released for privacy concerns. As our test dataset, the KDD99 dataset contains one type of normal data and 24 different types of attacks. For implementation Netbeans is used.

The input is KDD data set. It is about 10% of KDD dataset.

	A	B	C	D	E	F	G	H	I	J	K	L
1	duration	protocol_t	service	flag	src_bytes	dst_bytes	land	wrong_fra	urgent	hot	num_faile	logged_in
2	0	tcp	ftp_data	SF	491	0	0	0	0	0	0	0
3	0	udp	other	SF	146	0	0	0	0	0	0	0
4	0	tcp	private	S0	0	0	0	0	0	0	0	0
5	0	tcp	http	SF	232	8153	0	0	0	0	0	1
6	0	tcp	http	SF	199	420	0	0	0	0	0	1
7	0	tcp	private	REJ	0	0	0	0	0	0	0	0
8	0	tcp	private	S0	0	0	0	0	0	0	0	0
9	0	tcp	private	S0	0	0	0	0	0	0	0	0
10	0	tcp	remote_j	S0	0	0	0	0	0	0	0	0
11	0	tcp	private	S0	0	0	0	0	0	0	0	0
12	0	tcp	private	REJ	0	0	0	0	0	0	0	0
13	0	tcp	private	S0	0	0	0	0	0	0	0	0
14	0	tcp	http	SF	287	2251	0	0	0	0	0	1
15	0	tcp	ftp_data	SF	334	0	0	0	0	0	0	1
16	0	tcp	name	S0	0	0	0	0	0	0	0	0
17	0	tcp	netbios_n	S0	0	0	0	0	0	0	0	0
18	0	tcp	http	SF	300	13788	0	0	0	0	0	1
19	0	icmp	eco_i	SF	18	0	0	0	0	0	0	0
20	0	tcp	http	SF	233	616	0	0	0	0	0	1
21	0	tcp	http	SF	343	1178	0	0	0	0	0	1
22	0	tcp	mtp	S0	0	0	0	0	0	0	0	0
23	0	tcp	private	S0	0	0	0	0	0	0	0	0

Fig 3: KDD Dataset

The existing C4.5 decision tree gives the 95.7 percent of accuracy for detecting attacks.

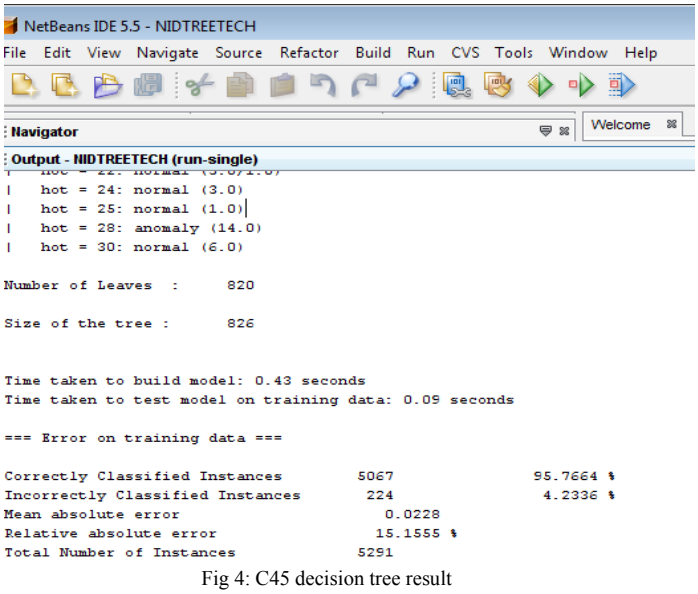

Fig 4: C45 decision tree result

The proposed C4.5 decision tree gives the 96.7 percent of accuracy for detecting attacks with with less false positive and true negative rates.

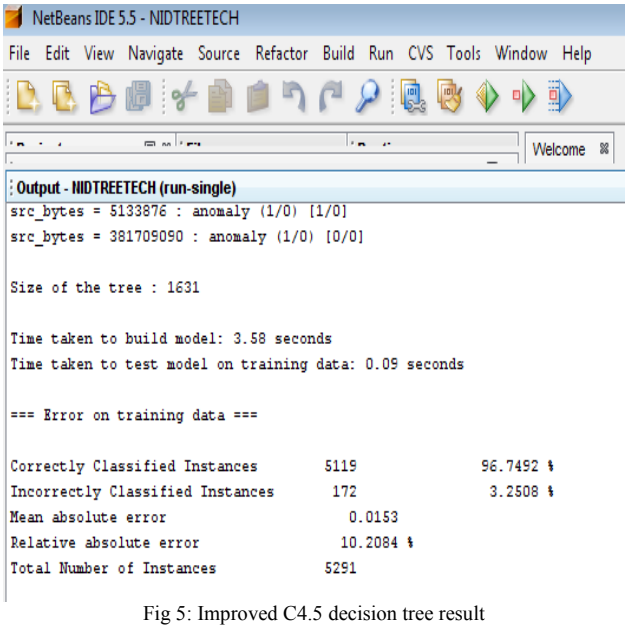

Fig 5: Improved C4.5 decision tree result

Following results gives the improved C45 performance on 10% KDD dataset with 5291 instances:

TABLE2: Improved C45 performance on 10% KDD dataset

PROPERTY	EXISTING C4.5	IMPROVED C4.5
Correctly Classified Instances	5067(95.76%)	5119(96.75%)
Incorrectly Classified Instances	224(4.23%)	172(3.25%)

CONCLUSION

Experimental results show the existing C4.5 decision tree gives 95.7 percent of accuracy for detecting attacks. But the proposed decision tree gives better attack classified results compare to existing C4.5 technique. Proposed Algorithm gives 96.9 percent of accuracy for detecting attacks with less false positive and true negative rates. Data mining algorithms require an offline training phase, but the testing phase requires much less time and future work could investigate how well it can be adapted to performing online.

REFERENCES

[1] Real-time Intrusion Detection and Classification by Phurivit Sangkatsanee1, Naruemon Wattanapongsakorn and Chalermpol Charnsripinyo.

[2] Intelligent Adaptive Intrusion Detection Systems Using Neural Networks (Comparitive study) by Aida O. Ali, Ahmed I. saleh and Tamer R. Badawy.

[3] An intrusion detection algorithm based on decision tree technology by Juan Wang, Qiren Yang and Dasen Ren.

[4] R. L. de Mantaras "A distance-based attribute selection measure for decision tree induction. Machine Learning, 6:81–92, 1991

[5] J. Mingers "An empirical comparison of selection measures for decision-tree induction. Machine Learning, 3:319–342, 1989.

[6] Nageswararao,Dr.D.RajyaLakshmi,Prof T.Venkateswara Rao, " Robust Statistical Outlier based Feature Selection Technique for Network Intrusion Detection" ,(IJSCE 2012).

[7] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani "A Detailed Analysis of the KDD CUP 99 Data Set", IEEE 2009.

[8] *www.cs.waikato.ac.nz/ml/**weka***

[9] J. R. Quinlan, "C4.5: programs for machine learning", Morgan Kaufmann Publishers, 1993.

[10] Hybrid Neural Network and C4.5 for Misuse Detection Zhi-Song Pan, Song-Can Chen, Gen-Bao Hu, Dao-Qiang Zhang, Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xi"an, 2-5 November 2003.

[11] C. Kruegel, D. Mutz, W. Robertson, F. Valeur, "Bayesian event classification for intrusion detection," in Proc. of the 19th Annual Computer Security Applications Conference, Las Vegas, NV, 2003.