NETWORK SECURITY USING DATA MINING

MR..MAYANK MANGAL[1], MR.AKSHAY PATIL[2], MR.UMAIR MOMIN[3], MR.HEMANT RAWAL[4]
[1]Assistant Professor, Department of Information Technology, ARMIET, Thane, India,
[2]UG Scholar, Department of Computer Engineering, ARMIET, Thane, India,
[3]UG Scholar, Department of Computer Engineering, ARMIET, Thane, India,
[4]UG Scholar, Department of Computer Engineering, ARMIET, Thane, India,

***Abstract*— Maximum Processing computation and more time consuming task has always been a limit in processing huge network intrusion data. This problem can be minimized through feature selection to condense the size of the network data involved. In this paper, we first preprocess dataset KDD 99 cup. Then we study and analysis of two decision tree algorithms (ID3 and C4.5) of data mining for the task of detecting intrusions and compare their relative performances. Based on this study, it can be concluded that C4.5 decision tree is the most suitable with high true positive rate (TPR) and low false positive rate (FTR) and low computation time with high accuracy.**

***Index Terms*—Intrusion Detection, KDD 99 Cup Dataset, ID3, C4.5**

I. INTRODUCTION

Intrusion detection has become a vital component of network administration due to the vast number of attacks constantly threaten our computers. Conventional intrusion detection systems are limited and do not provide a complete solution for the problem and are constructed by manual programming (encoding) of expert knowledge, changes to them are expensive and time consuming. They search for potential malicious activities on network traffics; they sometimes do well to find true security attacks and anomalies. However, in many cases, they fail to detect malicious behaviors (false negative) or they fire alarms when nothing wrong in the network (false positive).[4] In addition, they require comprehensive manual processing and human expert intervention. Applying Data Mining (DM) techniques on network traffic data is a capable solution that helps develop better intrusion detection systems. Moreover, with data integrity, confidentiality and availability, the system must be reliable, easy to manage and with low maintenance cost. Various modifications are being applied to IDS regularly to detect new attacks and handle them. In the proposed system, we are focusing on applying data mining algorithms for an Intrusion Detection System by comparing effectiveness and efficiency of C4.5 and ID3 algorithm. We are using KDD 99 cup dataset for training above algorithms. First step is preprocessing for KDD 99 cup dataset, using this dataset training our algorithms (ID3 and C4.5) and observing computational efficiency and time required to build decision tree. Our paper will give results related to decision tree by both algorithms. To declare efficiency in detecting intrusion, we need to use testing dataset to test our decision tree and hence we will be able to find accuracy to detect attack. [3]

II. LITERATURE SURVEY

Let's take analysis of different proposed methodologies for efficient intrusion detection system and our proposed method for intrusion detection. Different data mining approaches are applicable for efficient intrusion detection system. Various popular methods are:- k-means algorithm is a simple iterative method to partition a given dataset into a user specified number of clusters, k. In today's machine learning applications, support vector machines (SVM) are considered a must try—it offers one of the most robust and accurate methods among all well-known algorithms. One of the most popular data mining approaches is to find frequent item sets from a transaction dataset and derive association rules. Apriori is a seminal algorithm for finding frequent item sets using candidate generation. Ensemble learning deals with methods which employ multiple learners to solve a problem. The generalization ability of an ensemble is usually significantly better than that of a single learner, so ensemble methods are very attractive. The AdaBoost algorithm proposed by Yoav Freund and Robert Schapire is one of the most important ensemble methods. Given a set of objects, each of which belongs to a known class, and each of which has a known vector of variables, our aim is to construct a rule which will allow us to assign future objects to a class, given only the vectors of variables describing the future objects. Problems of this kind, called problems of supervised classification, are ubiquitous, and many methods for constructing such rules have been developed. One very important one is the naive Bayes method—also called idiot's Bayes, simple Bayes, and independence Bayes. This method is important for several reasons. It is very easy to construct, not needing any complicated iterative parameter estimation schemes. This means it may be readily applied to huge data sets. It is easy to interpret, so users unskilled in classifier technology can understand why it is making the classification it makes. And finally, it often does surprisingly well: it may not be the best possible classifier in any particular application, but it can usually be relied on to be robust and to do quite well. In our system we will use C4.5, a descendant of CLS and ID3. Like CLS and ID3, C4.5 generates classifiers expressed as decision trees, but it can also construct classifiers in more comprehensible rule set form.[1][2][4]

III. STAGES IN OUR PROPOSED SYSTEM

i. Resource Dataset:-[2]
   Training Dataset (KDD 99 Cup Dataset)
ii. Preprocessing Dataset
iii. Using cleaned dataset for training ID3 and C4.5
iv. Obtaining Decision tress of above algorithms
v. Using testing dataset to find accuracy of decision tree[3]
vi. Comparing error rate , time taken, efficiency using different types of instances of above dataset

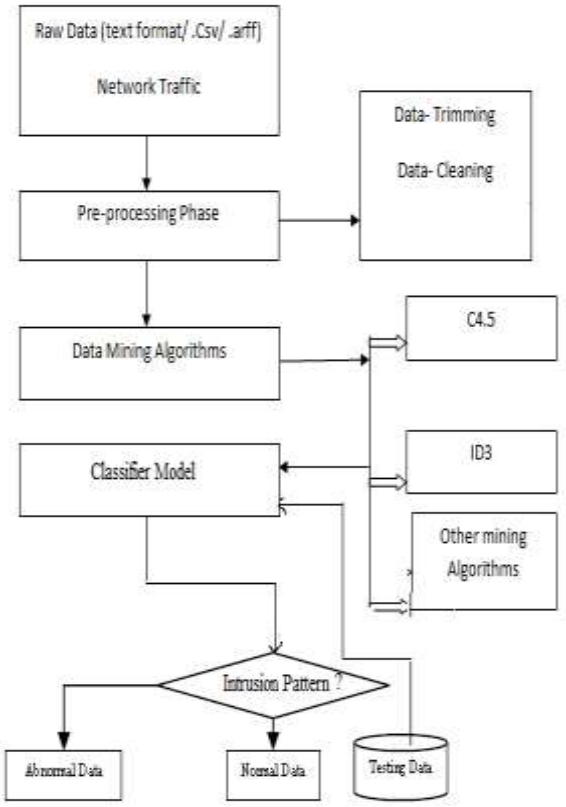

Fig: 1 Proposed System for Intrusion Detection

IV. EFFICIENT DATASET

A standard set of data which includes a wide variety of intrusions simulated in a military network environment, The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records. A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection record consists of about 100 bytes. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. We require all attributes since every attribute is valuable and hence in my proposed system i have not included reduction in terms of attributes and those attributes are:-

V. PREPROCESSING DATASET [2]

Following changes are Implemented in original KDD 99 cup dataset:

i. No redundant records in the train set, because classifiers may be biased towards more frequent records.

ii No duplicate records in the proposed test sets; therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records.

iii The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates

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_rate
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		

Fig:2 Attributes for Kdd99 cup dataset[3]

of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.

iv The number of records in the train and test sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research works will be consistent and comparable.

A) Data Trimming

Original KDD 99 Dataset having 4,940,000 instances can be trimmed to 4, 94,000. We can use SAS Enterprise Miner can be used (SAS EM) to experiment with our various Models and obtain results. I have preferred to randomized data and hence now data can be trimmed easily because dataset is having no particular pattern. Given the size of the data set it was trimmed down to only 10% of its original 744 MB and 4,940,000 records. The 10% subset was trimmed from the total data set instead of being sampled because the position of each record in the log file is valuable and should not be discarded. For example, DoS attacks are characterized by thousands of consecutive and identical records. It would make little sense to break up that valuable time-domain information by random sampling of the data. This is why the 10% of the overall data set correspond to a randomly chosen but consecutive cluster of data. For the same reason, we did not randomly sample the data during our experiments. Instead, we made sure to preserve the order in which records were initially created, since that order carries some valuable information that should not be discarded. [2]

B) Data cleansing or Data Cleaning:

We had to pre-process the data set in order to adapt it to our experiments' requirements. The first task was to verify the log file for unacceptable data. A comma-separated list of all attributes plus an extra trailing target field define each record. The features are either in binary, continuous, or symbolic format. Java code were written and used to check that each record contained the appropriate number of features. Out of the 494,000 records, only one was discarded because it contained 42 instead of 41 features. In my system I will use space separation for detecting attributes and accordingly will continue my system. Now our dataset is ready for training our proposed algorithms (ID3 and C4.5) [2]

VI. Decision Tree Algorithms

ID3 and C4.5 are algorithms introduced by Quinlan for inducing *Classification Models*, also called *Decision Trees*, from data.We are given a set of records. Each record has the same structure, consisting of a number of attribute/value pairs. One of these attributes represents the *category* of the record. The problem is to determine a decision tree that on the basis of answers to questions about the non-category attributes predicts correctly the value of the category attribute. Usually the category attribute takes only the values {*true*, *false*}, or {*success*, *failure*}, or something equivalent. In any case, one of its values will mean failure.

The basic ideas behind ID3 are that:

- In the decision tree each node corresponds to a non-categorical attribute and each arc to a possible value of that attribute. A leaf of the tree specifies the expected value of the categorical attribute for the records described by the path from the root to that leaf. [This defines what a Decision Tree is.]

- In the decision tree at each node should be associated the non-categorical attribute which is *most informative* among the attributes not yet considered in the path from the root. [This establishes what a "Good" decision tree is.]

- *Entropy* is used to measure how informative is a node. [This defines what we mean by "Good". By the way, this notion was introduced by Claude Shannon in Information Theory.]

A) ID3 Algorithm

The ID3 algorithm is used to build a decision tree, given a set of non-categorical attributes C1, C2, .., Cn, the categorical attribute C, and a training set T of records.

function ID3 (R: a set of non-categorical attributes,
C: the categorical attribute,
S: a training set) returns a decision tree;
begin
If S is empty, return a single node with value Failure;
If S consists of records all with the same value for the categorical attribute, return a single node with that value
If R is empty, then return a single node with as value the most frequent of the values of the categorical
attribute
that are found in records of S; [note that then there will be errors, that is, records that will be improperly classified];
Let D be the attribute with largest Gain(D,S) among attributes in R;
Let {dj| j=1,2, .., m} be the values of attribute D;
Let {Sj| j=1,2, .., m} be the subsets of S consisting respectively of records with value dj for attribute D;
Return a tree with root labeled D and arcs labeled d1, d2, .., dm going respectively to the trees

ID3(R-{D}, C, S1), ID3(R-{D}, C, S2), .., ID3(R-{D}, C, Sm);
end ID3;[4]

B) C4.5

An example which shows C4.5 can reduce attributes in decision tress based on entropy and information gain. Below is the sample KDD 99 cup dataset used to represent an example:

Network no	Logged_in	Is_host_login	protocol_type	Root_shell	attacks
1	1	1	tcp	1	Normal
2	0	1	udp	0	smurf
3	1	1	tcp	1	normal
4	0	0	tcp	1	Smurf5
5	1	1	tcp	1	Neptune
6	1	0	tcp	1	Normal
7	0	0	tcp	0	Neptune
8	1	1	udp	1	Normal
9	0	0	udp	1	smurf
10	1	1	tcp	1	Normal

Fig: 3 Sample Dataset (KDD 99 Cup)

Entropy(S) = - (5/10) * log (5/10) – (3/10) * log (3/10) – (2/10) * log (2/10) = 1

Gain (is_host_login, S) = Entropy(S) – 6/10 [-(4/6) * log (4/6) – (2/6) * log (2/6)] – 4/10 [-(1/4) * log(1/4) - (3/4) * log(3/4)] = 1 - 6/10(0.3899+0.5283) - 4/10(0.5+0.3113) = 1-0.5509-0.3245 = 0.2146

Gain(protocol_type,S)= Entropy(S) – 7/10 [-(4/7) * log (4/7) – (3/7) * log (3/7)] – 3/10 [-(1/3) * log(1/3) - (2/3) * log(2/3)] = 1 - 7/10(0.46134+0.5283) - 3/10(0.5283+0.2006) = 1-0.6895 -0.09831 = 0.21219

Gain(root_shell,S)= Entropy(S) – 8/10 [-(5/8) * log (5/8) – (3/8) * log (3/8)] – 2/10[0] = 1 - 8/10(0.423+0.5306) = 1-0.76288 = 0.23712

Gain(logged_in,S)= Entropy(S) – 6/10 [-(5/6) * log (5/6) – (1/6) * log (1/6)] – 4/10 [0] = 1 - 6/10(0.21919+0.43082) = 1-0.390006 = 0.609994

Thus, logged_in is the root node.

Now, for branch having value 1:
Gain(logged_in,is_host_login)=
Gain(logged_in,protocol_type)=
Gain(logged_in,root_shell)=
Similar computations will be carried out and node having greater value will be selected.

Now, for branch having value 0:
Gain(logged_in,is_host_login)=
Gain(logged_in,protocol_type)=
Gain(logged_in,root_shell)=
Similar computations will be carried out and node having greater value will be selected.

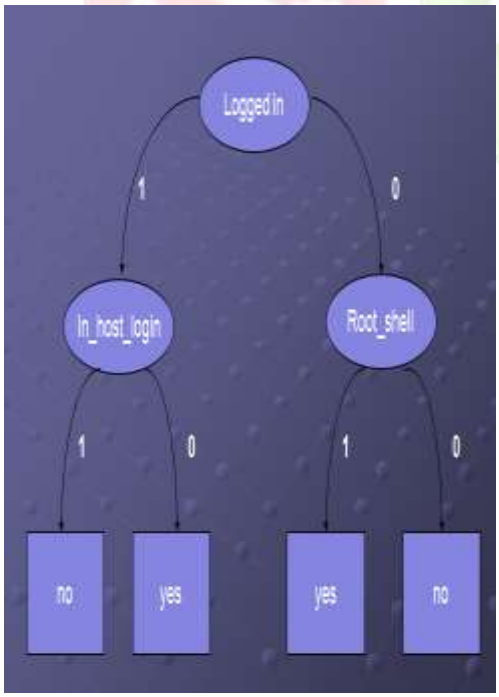

Fig: 4 Decision Tree

C) ID3 vs. C4.5

C4.5
A possibility to use continuous data.
Using unknown (missing) values which have been marked by "?".
Possibility to use attributes with different weights.
Pruning the tree after being created.

VII. Our Experimental Results:

Above Concepts shows how ID3 and C4.5 works, more pruned data is available in C4.5. We have trained these algorithms using our cleaned dataset and decision tree are obtained as follows:
Note: for verifying efficiency of algorithm for multiple instances we divide our dataset in two types

i) Traffic1 (more instances)
ii) Traffic 1 (less instances)

Below is the decision tree (in java) using our dataset with a trimmed dataset (traffic1) for ID3[Implemented in Java]

```
C:\Users\sachin\Desktop\desktop\ME PROJECT IDS>java ID3 traffic1.txt
if( service == "http") {
                output = "normal";
} else if( service == "ecr_i") {
                output = "smurf";
} else if( service == "private") {
                output = "neptune";
} else if( service == "Z39_50") {
                output = "neptune";
} else if( service == "smtp") {
                output = "normal";
} else if( service == "finger") {
                output = "normal";
} else if( service == "domain_u") {
                output = "normal";
} else if( service == "service") {
                output = "output";
}
4 Seconds

C:\Users\sachin\Desktop\desktop\ME PROJECT IDS>
```

Fig: 5 Output of ID3 using Traffic1

Below is the decision tree using traffic1 as dataset for C4.5

```
C:\Users\sachin\Desktop\desktop\ME PROJECT IDS>java C45 traffic1.txt
if( flag == "0") {
        if( protocol_type == "icmp") {
                        output = "smurf";
        } else {
                if( rerror_rate == "1.00") {
                                output = "neptune";
                } else {
                                output = "normal";
                }
        }
} else {
        if( duuration == "duuration") {
                        output = "output";
        } else {
                        output = "normal";
        }
}
763 Seconds

C:\Users\sachin\Desktop\desktop\ME PROJECT IDS>_
```

Fig: 6 Output of C4.5 using Traffic1

Below is the decision tree for ID3 and C4.5 using dataset (traffic2) which is having fewer instances compare to above used dataset (traffic1)

```
C:\Users\sachin\Desktop\desktop\ME PROJECT IDS>java ID3 traffic2.txt
if( service == "http") {
                output = "normal";
} else if( service == "ecr_i") {
                output = "smurf";
} else if( service == "private") {
                output = "neptune";
} else if( service == "Z39_50") {
                output = "neptune";
} else if( service == "smtp") {
                output = "normal";
} else if( service == "finger") {
                output = "normal";
} else if( service == "domain_u") {
                output = "normal";
}
0 Seconds

C:\Users\sachin\Desktop\desktop\ME PROJECT IDS>javac C45.java
Note: C45.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\sachin\Desktop\desktop\ME PROJECT IDS>java C45 traffic2.txt
if( flag == "0") {
        if( protocol_type == "icmp") {
                        output = "smurf";
        } else {
                if( rerror_rate == "1.00") {
                                output = "neptune";
                } else {
                                output = "normal";
                }
        }
} else {
                output = "normal";
}
0 Seconds
```

Fig: 7 Output of ID3 and C4.5 using Traffic2

	Dataset for training	ID3	C4.5
Time	For more Instances (traffic1)	More Efficent W.r.t time	Time Consuming
	For less instances (traffic2)	0 sec (traffic2)	0 sec (traffic 2)
Efficency In Decision tree	For both datasets	Works only for one best attribute (depending on entropy)	Works for more attributes and hence is more efficient. More pruning of data is observed
Hence from our experimental results it shows that C4.5 is giving more efficient Decision tree			

Fig: 8 Conclusions from Above Decision Tress

Now next step is to use testing dataset on above decision tress (algorithm will decide whether normal or attack) and hence we can compare efficiency of algorithm with the known results (in output attribute). Our next paper will show accuracy of detecting attacks using testing dataset. Attacks fall into four main categories:

- DOS: denial-of-service, e.g. syn flood;
- R2L: unauthorized access from a remote machine, e.g. guessing password;
- U2R: unauthorized access to local super user (root) privileges, e.g., various "buffer overflow" attacks;
- Probing: surveillance and other probing, e.g., port scanning.

It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data. This makes the task more realistic.

In our next paper our aim is to decrease false positive rate and increase correct detection of attacks from below confusion matrix

Confusion Matrix		Predicted Class	
		Normal	Intrusion/Attack
Actual Class	Normal	True Negative	False Positive
	Intrusion / Attack	False Negative	Correctly Detected

Fig: 9 Confusion Matrix

Will soon present report on testing dataset and will prove the best algorithm efficient in various scenarios (size of dataset/ pruning etc) and having more rate of true positive and less rate of false negative.

ACKNOWLEDGMENT

I would like to thank Prof. Mayank Mangal for facilitating all the necessary inputs, study material and resources and guiding me with their rich experience. I would especially like to thank my parents, for their unconditional support.

REFERENCES

1.International Journal of Machine Learning and Computing, Vol. 2, No. 5, October 2012" A Comparison of Efficiency and Robustness of ID3 and C4.5 Algorithms Using Dynamic Test and Training Data Sets*"* Payam Emami Khoonsari and AhmadReza Motie

2. Data mining for intrusion detection by Bertrand Portier Froment-Curtil
The University of Texas at Austin, Spring 2000 Dr. Ghosh – EE380L Data Mining Term Paper

3. KDD 99 Cup Dataset Details from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

4. *IDDDT-10* 978-1-4244-9034-/10/$26.00©2010 IEEE 731 A New Data Mining Based Network Intrusion Detection Model Mrudula Gudadhe, Prakash Prasad,

5http://dmoz.org/Computers/Security/Intrusion_Detection_Systems/

6. http://www.cs.umn.edu/~aleks/intrusion_detection.html

7. http://www.cc.gatech.edu/~wenke/ids-readings.html

8.http://www.cerias.purdue.edu/coast/intrusiondetection/welcome.html