ISSN: 2229 – 6956(ONLINE)
DOI: 10.21917/ijsc.2011.0021

ENSEMBLE DESIGN OF MASQUERADER DETECTION SYSTEMS FOR INFORMATION SECURITY

T. Subbulakshmi[1], S. Mercy Shalinie[2] and A. Ramamoorthi[3]
Department of Computer Science and Engineering, Thiagarajar College of Engineering, Tamil Nadu, India
E-mail: [1]subbulakshmitce@yahoo.com, [2]shalinie_m@yahoo.com and [3]armoorthi@gmail.com

Abstract
Masqueraders are a category of intruders who impersonate other people on a computer system and use this entry point to use the information stored in the systems or throw other attacks into the network. This paper focuses on Ensemble Design of a Masquerader Detection System using Decision trees and Support Vector Machines for classification with two kernel functions linear and linear BSpline. The key idea is to find out specific patterns of command sequence that tells about user behaviour on a system, and use them to build classifiers that can perfectly recognize anomalous and normal behaviour. Real time truncated command line data set collected from a debian Linux server is used for performance comparison of the developed classifiers with the standard truncated command line data set of Schonlau[4]. The results show that Ensemble Design of Masquerader Detection Systems is much faster than individual Decision trees or Support Vector Machines.

Keywords:
Masquerader Detection, Support Vector Machines, Decision Trees, Truncated Command Sequences

1. INTRODUCTION

Intrusion detection: the most effective way to detect intrusions is by using the audit data generated by the operating system. Since almost all activities are logged on a system, it is possible that a manual inspection of these logs would allow intrusions to be detected. It important to analyze the audit data even after an attack has occurred to determine the extent of damage sustained this analysis also helps in tracking down the attacks and in recording the attacks patterns for future detection. A good Intrusion Detection that can be used to analyze audit data makes a valuable tool for information systems.

The idea behind anomaly detection is to establish each user's normal activity profile as possible intrusion attempts. A main issue concerning misuse detection is how to develop signatures that include all possible attacks to avoid false negatives, and how to develop signatures that do not match non-intrusive activities to avoid false positive. Through false negatives are frequently considered more serious, the selection of threshold levels is important so that neither of the above problems is unreasonably magnified.

A masquerade attack in which one user impersonates another is one of types of computer abuse, largely because such attacks are often mounted by insiders and can be very difficult to detect. Automatic discovery of masqueraders is sometimes undertaken by detecting significant departures from normal user behavior, as represented by user profiles based on users command histories. Masquerading is the act of substituting oneself for another Masquerading will be to disguise by assuming the appearance of someone or not. Sometimes it will be furnishing with a false appearance or an assumed identity or obscure the existence or true state or character of something. The computer masquerade problem can be explained in the following scenario. A legitimate user takes a coffee break, leaving his/her terminal open and logged in during the users absence, an interloper assumes control of the keyboard, and enters commands, taking advantage of the legitimate user's privileges and access to programs and data.

Specifically, our motivation is from three facts of Intrusion Detection community. First systems that embedded in hosts of a network should react quickly to drifting normal behaviors e.g. user behavior reveals particular regularity during a period, but it might change to another pattern because of tasks at hand.Under such case, systems should undergo the changing and thus update the normal profile it has established. Second Intrusion Detection is a real time-critical mission with the ability to identify attack at the moment it happens, and thus systems should be capable of rapid response, both learning time and running time should be as short as possible while keeping high detection accuracy. Third because systems are embedded in the hosts, they run as a background task with little computational overhead, rather than a foreground application needs considerable resource cost, less processing memory and simpler operating environment characterization.

2. LITERATURE REVIEW

Intrusion can be defined as any unauthorized access to the legal network using any of the backdoors or using a legitimate user's identity Intrusions are divided in [1] into 8 basic categories: 1. Eves dropping and packet sniffing: sniffing is done by turning the Network Interface card of the machine into promiscuous mode and thereby capturing all the information passing through the network even though the packets are not intended for that machine. So if an attacker gets access to a particular machine he can get all the packets passing through the network. 2. Snooping and downloading: Downloading the content which is restricted in that domain by using some other proxy servers. 3. Tampering or data diddling: The data passing from machine a to machine b is interrupted in between and truncated to some other machine in the network. 4. Spoofing: Artificially spoofing the IP address of the machine in the network and getting all the information intended for that machine. 5. Jamming, Flooding: sending continuous requests to the server sothat it can't respond to authorized user requests. 6. Masquerading: Impersonating other users in the network and using all the resources intended for the users. 7. Exploiting Vulnerabilities: finding out and using vulnerabilities like honey nets in the network. 8. Password Cracking and keys: cracking the passwords of the users and using them for personal use.

The audit data is generated [10] using UNIX acct auditing mechanism. The data generated will have the fields command name, user, terminal, start time, end time, real, CPU, memory usage (K) among these the first two fields are used in this

research. The first 15,000 commands of 70 users are collected. The time span of command collection differs for each user. The data is decomposed into 150 blocks of 100 commands each. The blocks 51 through 150 contain contaminated blocks. It is assumed that either a block is contaminated completely or not at all. There are no mixed blocks Six methods are used.

Unix command data was collected in [4] from 50 users and this is used as benchmark in evaluating IDS using command sequences. Results of the six methods uniqueness, Bayes one-step markov, Hybrid Multi-step Markov, Compression, Sequence-Match, incremental probabilistic action modeling shows that the false negatives is very low i.e. within the interval (1-7). The missing alarms fall in the range of (30-60%).

Number of six different statistical approaches was [4] presented for detecting intrusions with commands of 50 users including two from the computer science community. The experiments shows that a total of 40 masquerader incidents for the 50 users which accounts for 474% of data. There were at most three incidents for any users and 21 users were no masqueraders at all. Half the masqueraders were present for four or fewer blocks while three masqueraders were present for three blocks. The results are presented along with performance of selected individuals, correlation methods, ROC curves with survival analysis of intrusions and ROC curves without updating. Finally each method is discussed with it's pros and cons. The complexity and scale of the problem is discussed.

This approach described in [11] shows some results concerning the generalization and robustness capabilities of Machine learning algorithms in creating user profiles for anomaly detection based on classification of artificially created Unix command line arguments. The hybrid approach begins with applying Expert Fuzzy rules to reduce the dimensionality of data, followed by initial clustering of the data and refinement by LVQ. Since LVQ is an nearest neighbor classifier, a new record which lies outside a specified distance is classified as masquerader. So anomalous records need not be included in the training data. A four step approach is followed.

Common test parameter settings are given as: Commands in each user record: 100, training sample size: 1875 records (75% of available records), legitimate user test sample size: 625 records (25%), Anomaly records: 200, LVQ: 002 with 50000 iterations the results are shown for three tests which show they seem lackluster and this approach merits future consideration.

A highly effective approach to masquerader detection [12] using Hidden Markov Models is presented, also a formulation is presented to calculate the effectiveness of masquerade detection with Schonlau's dataset Detection rate is the ratio of dirty blocks to total number of dirty blocks in the dataset. False positive rate is the ratio of number of clean blocks classified as dirty blocks to total number of clean blocks. According to Maxicon R The cost of masquerade detection is defined as cost = 6 X FPR + (100-DR). In this paper a formula to calculate the effectiveness is given Effectiveness = (1-α) * DR + α * FPR. The value of α = 02 is fair. This formula adds weight to DR while enhancing the FPR.

A novel anomaly detection is proposed in [13] for targeting masquerades based upon unpopular and uniquely used commands. An SGI server running on IRIX 62 is used for data collection of 10,000 commands of 53 users. They are further divided into training data sets of 5,000 commands and test datasets of 1,000 commands. The 1,000 test data sets are further divided into 2 replications of 500 commands or 10 replications of 100 commands. A test statistic is defined based on unpopular and unique commands with their weight and uniqueness index. The uniqueness method used in this paper is based on the assumption that commands not used by the common community of user are indicative of the masquerader. This conceptually simple approach reports false alarm rate of 5% corresponds to 10% missing alarms based on 100 commands about 2% based on 500 commands and about 1% based on 1000 commands i.e., The user can be easily distinguished as the test data sequence is larger.

SVM based Masquerader detection [14] was done with truncated and enriched command data sets. The concept of common commands was introduced to reflect the diverse command patterns exhibited by different users. Truncated command lines detects the Masquerades 801% and 948% (pervious study results: 693% and 628%) of the time and Enriched command lines detects the masquerades 873 % (pervious study results: 821%). The architecture proposed in this paper includes creation of Audit DB, data preprocessing and feature selection module, classification with voting engine, SVM kernel and parameter selection and the actual experiments with truncated and enriched command lines with the common command assigned with threshold values. The experiments convincingly demonstrate that SVM[15] is an effective approach for masquerader detection. The false alarm rates can be further reduced by considering additional factors like CPU usage, memory usage and file access.

Shonlau's work is extended in [2] and the sequences of truncated command lines is classified into two categories self and non self using an algorithm inspired by Naïve Bayes. The truncated data of 15,000 commands from 50 users is configured in two ways. The first one is the SEA configuration which is used by Shonlau where there are randomly injected commands from users outside the 50 user community. The second one which is used in this paper is lv49 configuration where each user is crossed with every other user to compare the effects of every user acting as a masquerader against all other users So the paper focuses on determining the performance improvement in Detection rate of a new classifier and to provide a detailed error analysis. The performance is assessed by applying a cost function of the form.

$$\text{Cost} = \alpha\,(\text{Misses}) + \beta\,(\text{False alarms})$$

where $\alpha = \beta = 1$ to give equal weight to misses and False alarms So the cost function becomes

$$\text{Cost} = \text{Misses} + \text{False alarms}$$

Maxicon and Townsend's work is extended [3] by using enriched command lines and increases the detection rate by 82% with corresponding 30% reduction in overall cost of errors and a small increase in false alarms. Enriched command lines are the commands with all arguments , options, flags or elements of shell grammar such as pipes and semicolons Examples of enriched and truncated command lines are clearly given. Then the two experimental methods were discussed along with selection of subjects for data collection, selection of training and test data and the detection algorithm.

3. AUDIT SOURCE

3.1. SHONLAU'S STANDARD TRUNCATED COMMAND SEQUENCES

The Audit source for our experiments is derived from the Truncated command sequences of Schonlau[4]. Data set is collected with seeded masquerading users to compare various Masquerader detection methods. The data set consist of 50 records representing to one user each record contains 15,000 commands. The audit data is generated with unix_acct. The first 5000 commands for each user do not contain any masqueraders and are intended as training data. The next 10,000 commands can be thought of as 100 blocks of 100 commands each. So a single user has 100 blocks of commands. Every block is represented using a score of '0' or '1'. '0' means that the corresponding 100 commands are not contaminated by a masquerader '1' means they are contaminated. This 100 values form the first row of a matrix and the other users are also represented in the same way. Finally the complete matrix will have the size of 50x100 which will represent the scores of the blocks. The first 50 blocks are used for building the model using Training and the next 100 blocks are used for testing the developed model.

3.2 REAL TIME TRUNCATED COMMAND LINE DATA

A debian Linux server connected with 125 nodes using 100 users is used for real time truncated command line data collection. The data collection is done for two months period All the 100 users were observed initially. The 100 users are from various groups 60 Students, 10 Research scholars, 7 scientists, 14 academic staff of that organization and 9 novice users are observed. After 15 days the less active 50 users were eliminated and only the active 50 users were taken into account. A total of 15000 commands were collected for each and every user first the raw commands are filtered for their arguments, pipes and options. Record for each and every user is created with only truncated command lines. The commands are divided into blocks of command size 100. Now all the 50 users are having a record with 150 blocks of 100 commands each. The windows ASCII file is created to represent whether the block is normal or anomalous. The first 50 blocks are used for building the model using Training and the next 100 blocks are used for testing the developed model.

4. DECISION TREES FOR CLASSIFICATION

Decision tree is made of decision nodes and leaf nodes Each decision node corresponds to a test X over a single attribute of the input data and a number of branches, each of which handles an outcome of the test X Each leaf node represents a class that is the result of decision for a case. The idea of constructing the decision tree is 'divide and conquer technique'. A set T of training data consists of k classes (c1, c2…ck). If T only consists of cases of one single class, T will be a leaf. If T contains no case, T is a leaf and associated class with this leaf is the default class of parent node. If T contains cases of mixed classes, then T will be divided into n subsets based on the test of some attribute of the training data. The process is continues until every subset belongs to a single class. The major issue here is to choose the attribute which acts as the best classifier. This is done by calculating the information gain of each attribute using the entropy of each attribute. The attribute with highest information gain is chosen as the classifier. Once the decision tree is built it can be used to classify the test data which has the same features as the training data.

5. DECISION TREES FOR MASQUERADER CLASSIFICATION

Decision trees are capable of classifying large datasets. The classification process starts from the top of the decision tree. The process of finding the root node is done by calculating the information gain of the attributes block score, command frequency and block type.

The steps in classification process are as follows

5.1. Assigning scores for blocks

5.2. Calculating the command frequency

5.3. Design of Decision tree classifier

5.1 ASSIGNING SCORES FOR BLOCKS

The block score is calculated from the random values assigned for each command and it belongs to any of the three values low, medium and high.

5.2 CALCULATING THE COMMAND FREQUENCY

The command frequency of the block is also calculated based on the no of occurrence of that command in the block which belongs to upper or lower threshold levels.

5.3 DESIGN OF DECISION TREE CLASSIFIER

The structure of the decision node is closely related to the performance of the developed classifier. So it is very important to determine the structure of the tree according to the training samples and their attribute values. To have better classification accuracy more separable classes based on their information gain should be classified as upper nodes of the decision tree.

The entropy is the maximum number of bits needed to encode class (I or II) of randomly drawn member of S So Entropy S can be defined as

$$\text{Entropy}(S) \equiv -p_I \log_2 p_I - p_{II} \log_2 p_{II}$$

The information gain of a particular attribute can be calculated as the expected reduction in entropy due to sorting on attribute A.

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

The process of training is the process of forming the decision tree The Gain (S, A) is calculated for the attributes block score, command frequency and block type It is found that block score has the highest of all values. So the decision tree is formed by choosing the block score as the root node. The block type is taken from the windows ASCII file which is of type masquerader of non-masquerader. A sample decision tree is

illustrated in Fig.1. For training the first 5000 commands of Shonlau's standard command and Real time data is taken. When learning of the decision tree classifier is completed using the training data, the next 10,000 commands of the standard and real time data were given as test data to test its performance. The results are shown in Table.1 for a representative set of users.

Table.1. Classification Results of Decision Tree Classifiers (SD – Standard Data, RTD – Real Time Data)

Sl.No	User	Classification Accuracy	
		SD	RTD
1	User 3	98.13	99.15
2	User 14	51.31	49.37
3	User 21	62.37	72.19
4	User 24	100.00	85.41
5	User 38	99.58	79.42
6	User 41	100.00	98.48

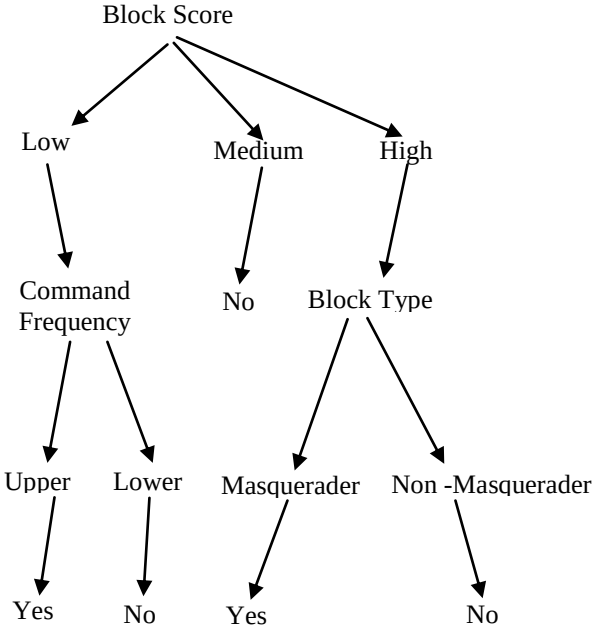

Fig.1. Decision Tree Classifier

6. SVM FOR CLASSIFICATION

Statistical theory was first started by Vapnik and chervonenkis in the sixties. The data model generated by an unknown stochastic regularity is observed. Learning is done by extracting regularity from the data the analysis of the learning problem leads to notation of the capacity of the function classes that a learning machine can implement, the supporter vector machines use a particular type of function class classifiers with large "margins" in a feature space induced by a kernal.

In support vector machines there are two sets, input set and output set. Input set is represented by X and Output set is represented by Y. The training set $(x_1, y_1)\ldots(x_m, y_m) \varepsilon X \times Y$. Generalization is to find a suitable $y \varepsilon Y$, given a previously unseen $x \varepsilon X$, where (x, y) should be similar to $(x_1, y_1)\ldots (x_m, y_m)$. The objective is to Classify the points $X = \Phi(x)$ in feature space according to which of the two class means is closure.

$$c_+ = \frac{1}{m_1} \sum_{y_i=1} \phi(x_i) \tag{1}$$

$$c_- = \frac{1}{m_2} \sum_{y_i=-1} \phi(x_i) \tag{2}$$

7. SVM FOR MASQUERADER CLASSIFICATION

A new method is proposed for classifying the masquerader using Support Vector Machines The Standard dataset is taken first and classified using the method proposed in this paper and the real time dataset is taken and fed to the classifier The classification thus obtained gives the actual measure of classification of the classifier The steps in the classification process are

- Data Pre-processing
- Selection of Kernal Function and parameters
- Design of SVM Classifier

7.1. DATA PRE-PROCESSING

The training data for our experiment is taken from schonlau dataset is a collection of 50 files representing to one user each file contains 15,000 commands. The unique commands are derived from the 50 files representing the 15000 commands of each user Random values are assigned for each unique command. Each user file is individually taken and assigned with the previously determined random values for the commands. The last 10000 commands are taken and divided into blocks of 50 commands each. Two blocks are combined to form the training input X of Size 100x2. The training target Y of size 100x1 is formed from the windows ASCII file of the dataset. Each column of the ASCII file corresponds to one of the 50 users. Values in each row correspond to a pair of 50 commands. The values of Y are +1, -1; +1 is for masquerading block and -1 is for non- masquerading block. This process is repeated for all the 50 users to have their own training and target inputs. The real time test data is also processed in the same way for testing the performance of the classifier.

7.2. SELECTION OF KERNAL FUNCTION AND PARAMETERS

They are six no of different kernels available which are are linear , polynomial, Gaussian RBF, linear spline, linear bspline, and exponential RBF if the kernel function is selected differently then svm will produce different outputs.

Values for 'ker' can be selected from any of the following kernel functions

- Linear

$$k = u \times v` \tag{3}$$

- 'bspline'(p1 is degree of bspline)

$$k = \exp\left(-sqrt\left((u-v)\times(u-v)`\right)/(2\times p_1 \wedge 2)\right) \tag{4}$$

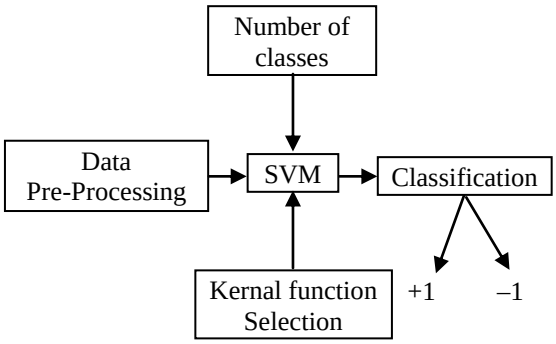

Fig.2. SVM Classifier

7.3. DESIGN OF SVM CLASSIFIER

The SVM Classifier is designed as in Fig.2 for the classification process. The data is preprocessed and it is converted into a feature vector X of size 100x2 and the target vector Y is created with the size 100x1. Feature vector and target vector is given to the SVM classifier. The kernel function is selected from the available list of six kernel functions. The number of classes is fed to the classifier. The No of classes are two +1 for masquerading block and -1 is for non-masquerading block. The input is first given to a single user and the remaining users are also classified in the same manner. The Results are shown in Table.2 for a representative set of users.

Table.2. Classification Results of Support Vector Machine (Linear Kernal) Classifiers (SD – Standard Data, RTD – Real Time Data)

Sl. No.	User	Classification Accuracy	
		SD	RTD
1	User 3	88.13	98.13
2	User 14	52.58	45.19
3	User 21	65.59	74.59
4	User 24	99.89	100.00
5	User 38	100.00	78.32
6	User 41	98.56	97.52

Table.3. shows the classification result of standard data and real time data using SVM-L classifier for different users. Real time data instances are classified better than the standard data.

Table.3. Classification Results of Support Vector Machine (Linear BSplineKernal) Classifiers (SD – Standard Data, RTD – Real Time Data)

Sl. No.	User	Classification Accuracy	
		SD	RTD
1	User 3	97.19	95.15
2	User 14	49.52	68.19
3	User 21	64.82	100.00
4	User 24	99.88	88.42
5	User 38	95.52	100.00
6	User 41	98.59	94.58

8. ENSEMBLE DESIGN OF MASQUERADER DETECTION SYSTEMS

In the Ensemble design each classifier is designed individually for classifying the users as either Masquerader or non Masquerader. Ensemble design of masquerader detection system is shown in Fig.3.

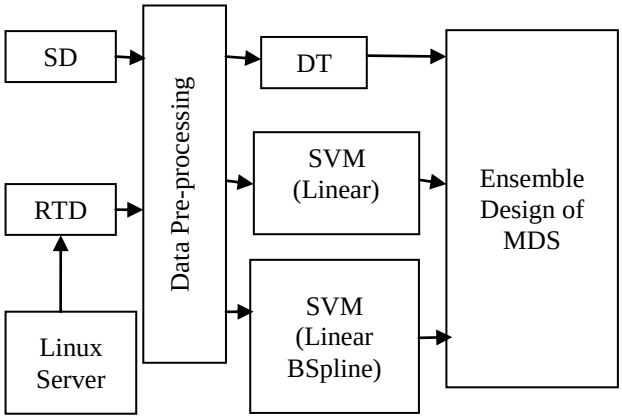

Fig.3. Ensemble Design of MDS

Decision trees are used for classifying the user as either normal or abnormal based on the information gain value. Based on the calculated information gain the user will be classified as normal as abnormal. Support Vector Machines with six kernel functions is used for classification. Among the available six kernel functions the linear and linear bspline are the functions which is applicable for all the available users. Also Support Vector Machines with this kernel functions classifies the users in a very shorter duration of time. So these three classifiers are designed separately and their outputs are observed. A threshold value between 2 to 5 is assigned for each classifier. The Ensemble Design of Masquerader Detection Systems receives input from each classifiers output and based on the threshold value the user will be finally classified as normal or abnormal. If a user is classified as normal with high threshold value by one or more than one classifier then the user is actually classified as normal user by Ensemble classifier. Otherwise the user will be classified as abnormal. The Table .4 Shows the Classification Accuracy of Standard and Real Time data for the selected set of users. The Ensemble Design classifiers are found to be better for many users than individual classifiers. Since all the classifiers' outputs are taken along with the threshold value, the Ensemble Design classifier will give better results.

Table.4. Classification Results of Ensemble Design Classifiers (SD – Standard Data, RTD – Real Time Data)

Sl. No.	User	Classification Accuracy	
		SD	RTD
1	User 3	99.25	99.45
2	User 14	88.19	81.29
3	User 21	70.15	75.97
4	User 24	99.98	100.00
5	User 38	99.79	89.52
6	User 41	99.56	93.19

9. OUTCOMES OF THE RESEARCH & COMPARISON

Table.5 and Table.6 shows the performance results of User3 & User 24 for all the classifiers for Standard and Real time data set. The Table.5 shows the classification results of all the classifiers for user3. From the Table.5 it is observed that Ensemble Design classifiers perform better than other classifiers. The classification rate is found to be better for the Ensemble Design classifiers. For each and every user and for real time data or standard data the classification rate will be different. Table.6 is shown for user24. User24 is best classified by the Decision Tree and Ensemble Design Classifiers for Real and Standard data.

Table.5. Performance Results of User 3

Sl. No.	Data set	DT	SVM (L)	SVM-LBSP	ED
1	SD	98.13	88.13	97.19	99.25
	RTD	99.15	98.13	95.15	99.45

Table.6. Performance Results of User 24

Sl. No.	Data set	DT	SVM (L)	SVM-LBSP	ED
1	SD	100.00	99.89	99.88	99.9
	RTD	85.41	100.00	88.42	100

Table.7. Selection of Best Classifier for the Representative set of users (BC –Best Classifier, CR- Classification Rate)

Sl. No.	User	SD		RTD	
		BC	CR	BC	CR
1	U3	ED	99.25	ED	99.45
2	U14	ED	88.19	ED	81.29
3	U21	ED	62.37	SVM -LBSP	100
4	U24	DT	100	SVM (L)	100
5	U38	SVM (L)	100	ED	89.52
6	U41	SVM -LBSP	98.59	DT	98.41

Table.7 shows the best classifier for each and every set of user's standard and real time data. User 38 is best classified with Support Vector Machines Linear Classifier for standard data and using Ensemble Design Classifier for Real Time data. The classification rate is found to be 100.00% and 89.52 % respectively. This result is repeated for all the users and the users are classified using different algorithms using Ensemble Design. The best classifier for the representative set of users and their classification rate is mentioned in the Table.7 for the standard and real time data. ED classifier gives the best classification rate for user 3 for both standard and real time dataset. ED classifier results the best classification for standard dataset of user 21 and SVM-LBSP results the best classification rate for standard dataset of user 21. In Ensemble of classification best classifier is used for classification.

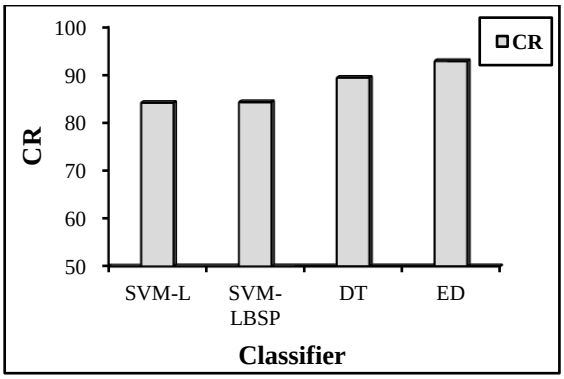

Fig.4. CR comparison for Standard data

Fig.4. show the performance comparison of classifiers. SVM-l classifier results the overall CR as 82.34 and SVM-LBSP results the 83 and DT results 89 and ED produce the high classification rate of 92.37 percentage.

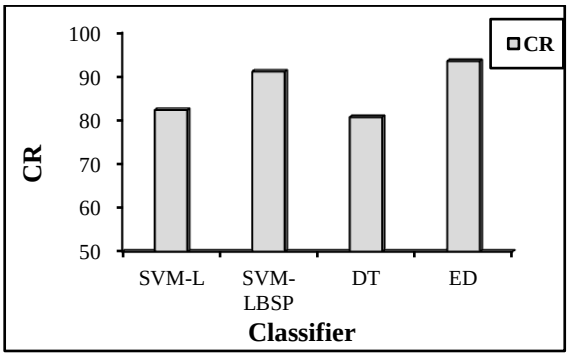

Fig.5. CR comparison for Real-time data

Fig.5. shows the performance comparison of classifiers for real time data. SVM-l classifier results the overall CR as 81.24 and SVM-LBSP results the 90.02 and DT results 79.34 and ED produce the high classification rate of 93.48 percentage.

10. KEY FINDINGS OF THE RESEARCH

This research outlines the following key findings

- The users of the standard and real time data differ in behaviour

The user 21 of standard data is best classified using Ensemble Design with the classification rate of 62.37%, whereas the User 21 of Real Time data is best classified using Support Vector Machines (LBSP) classifier with the classification rate of 100.00%. So for the blocks of user 21 either Ensemble Design or Decision Tree classifier will be employed.

- The decision tree classifiers perform best for minimum number of users
- The support vector machine with two kernel functions performs better than Decision Tree classifiers for the available users.
- Ensemble Design gives best classification for most of the users in Real Time and Standard Data.

From the table it is observed that the Decision Tree Classifiers gives best classification for 2 users and Support Vector Machines with Linear or Linear BSpline classifiers gives

best classification for 4 users and the Ensemble Design classifiers gives best classification for total of 6 users. Even for the other users the Ensemble Design classifiers perform well.

- The developed Ensemble Design offers best detection of the blocks of commands of the users in Standard and Real Time data and thereby we can better classify the users as normal or Masquerader user.

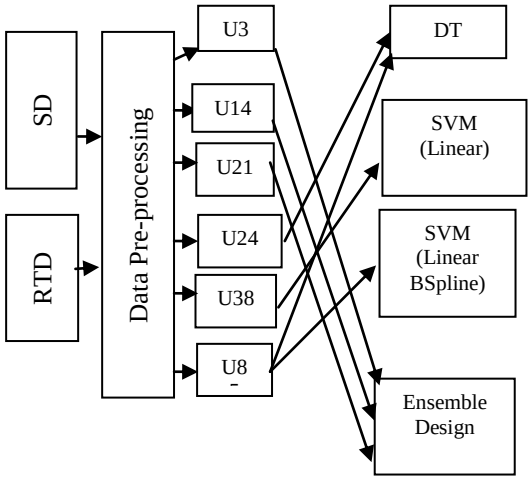

Fig.6. Ensemble Design of MDS

Fig.6. shows the MDS for different users. Standard and real time data are input to the preprocessing.

11. CONCLUSION AND FUTURE WORK

This paper proposes Ensemble Design to classify the normal user and Masquerader user Decision Trees and Support Vector Machines are used for classifying the normal users from the abnormal users Support Vector Machines shows better results than Decision trees. The Ensemble Design gives better classification accuracy than the other two classifiers. The detection rate of real time data set is found to be low since the users are from diversified community The detection rate can be still improved by using sophisticated data collection strategies and by including the hackers in the user community, So that we will have more number of masquerading blocks for the experiments The detection rate can also be improved by using the combination of machine learning algorithms Some of the suggested algorithms are Genetic algorithm and Fuzzy Logic Genetic algorithm can be used for optimizing the number of commands to a reasonable and reduced count and thus reducing the time taken for classification Fuzzy logic can be used to write more precise rules with Fuzzy Associative Memory enabling the accuracy of the classifiers This work can be extended to various SVMs and with different parameters.

REFERENCES

[1] Bhukya, Wilson Naik Kumar, G Suresh Negi, Atul, 2006, "A Study of Effectiveness in Masquerade Detection", In Proc of TENCON, IEEE Region 10 Conference, pp.1-4.

[2] Dorothy E Denning, 1987, "An intrusion - detection model", In Proc of the IEEE Transactions on Software Engineering, Vol. 13, No.2, pp.222 – 232.

[3] Han-Sung Kim, Sung-Deok Cha, 2005, "Empirical evaluation of SVM-based masquerade detection using UNIX commands", Elsevier proceedings of the Computers and Security, Vol.24, No.2, pp.160-168.

[4] Kwong H Yung, 2004, "Using Self- Consistent Naïve Bayes classifiers to detect masqueraders", PAKDD, Vol. 3056, pp.329-340.

[5] Matthias Schonlau and Martin Theus, 2000, "Detecting Masquerades in intrusion detection based on unpopular commands", Information processing letters, Vol.76, No.1-2, pp.33-38.

[6] Maxion, RA Townsend, T N, 2002, "Masquerade detection using truncated command lines", International Conference on Dependable Systems and Networks, pp. 219- 228.

[7] Robert Birkely, 2003, "A Neural Network based Intelligent Intrusion Detection System", M S Thesis, Griffith University, Gold coast campus.

[8] Roy Maxion, 2003, "Masquerade Detection Using Enriched Command Lines", In Proc of International Conference on Dependable Systems and Networks, pp.5-14.

[9] William DuMouchel, Wen Hua Ju, Alan F Karr, Matthius Schonlau, Martin Theus, and Yehuda Vardi, 2001, "Computer Intrusion: Detecting Masquerades", Journal of Statistical Science, Vol.16, No. 1, 58-74.

[10] Min Yang, Huang Zhang and H.J. Cai, 2007, "Masquerade Detection Using String Kernels", IEEE Wireless Communications, Networking and Mobile Computing, pp. 3681- 3684.

[11] Bao Cui-Mei and Zibo, 2009, "Intrusion Detection Based on One-class SVM and SNMP MIB Data" IEEE Information Assurance and Security, Vol.2, pp. 346-349.

[12] Zhang Hongmei, 2009, "SVM ensemble intrusion detection model based on Rough Set feature reduct", IEEE Conference on Control and Decision Conference, pp.5604-5609.

[13] Zhili Cai, Qiushi Ding, Mingting Wang, 2010, "Study on Automated Incident Detection Algorithms Based on PCA and SVM for Freeway", International Conference on Optoelectronics and Image Processing, Vol. 2, pp.420-424.