Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 2, February 2014, pg.558 – 567

RESEARCH ARTICLE

DETECTION OF MASQUERADERS USING TREE STRUCTURED SVM

P.Anishprabu[1], M.Sangeetha[2], M.Nagulanand[3]

[1]M.E. Computer Science and Engineering, SriGuru Institute of Technology, Coimbatore
[2]M.E. Computer Science and Engineering, SriGuru Institute of Technology, Coimbatore
[3]M.E. Computer Science and Engineering, SriGuru Institute of Technology, Coimbatore

[1] anishbtechit@gmail.com; [2] msangeetha2612@gmail.com; [3] msnagul07cs48@gmail.com

Abstract— Creating and recognizing automatically the behavior profile of a user from the commands in a command line interface. Computer user behavior is represented as a sequence of UNIX commands. This sequence is transformed into a distribution of relevant subsequences in order to find out a profile that defines its behavior .The existing system novel evolving user behavior classifier is based on Evolving Fuzzy Systems and it takes into account the fact that the behavior of any user is not fixed, but is rather changing. Timely detection of computer system with intrusion is a problem that is receiving increasing attention. Previous approach cannot prevent legitimate user from abusing their rights in a computer system. Proposed system analysis also be used to supervise, analyze, and detect abnormalities based on a time-varying behavior of same is not considered; we proposed our work to monitor and detects the Masquerader from user behavior profile. Tree-structured architecture is adopted in the partition to avoid the problem of predetermining the number of partitioned data in the region. Then, in the second stage, multiple SVMs, also called SVM experts, that best fit partitioned regions are constructed by finding the most appropriate kernel function and the optimal free parameters of SVMs. Different UNIX command data, proposed system show that a system based on our approach can efficiently recognize a UNIX user and detects masquerader from data. SVMs experts achieve significant improvement in the generalization performance in comparison with the single SVMs models in the existing system.

Index Terms—Evolving fuzzy systems, fuzzy-rule-based (FRB) classifiers, user modelling
Keywords – Tree structured SVM, user modelling

1 INTRODUCTION

Data mining is the process of discovering actionable information from large sets of data. Data mining uses mathematical analysis to derive patterns and trends that exist in data. Data mining is one of the most important research fields that are due to the expansion of both computer hardware and software technologies, which has forced organizations to depend heavily on these technologies. Data is considered as the number one asset of any organization, it is clear that this benefit should be used to predict

future decisions sequent, and since organizations are continuously rising, their relative databases will produce as well; as a result their current data mining techniques will fail to cope up with large databases which are dynamic by nature.

Recognizing the behavior of others in real-time is significant in different tasks, such as to predict their future behavior, to manage with them or to help them. In order to act efficiently, humans usually try to recognize the behavior of others. New theories claim that a high percentage of the human brain capacity is used for predicting the future, including the behavior of other humans. Specifically, computer user modelling is the process of learning about ordinary computer users by observing the way they use the computer. This process needs the creation of a user profile that contains information that characterizes the usage behavior of a computer user. Knowledge has shown that users themselves do not know how to articulate what they do, particularly if they are very well-known with the tasks they perform. Computer users, like all of us, leave out activities that they do not even notice they are doing.

Construction of effective computer user profiles is a difficult problem because of the following aspects: human behavior is usually irregular, and sometimes humans behave differently because of a change in their goals. Profiling and recognizing general user behavior profiles is proposed. This approach is called ABCD (Agent Behavior Classification based on Distributions of relevant subsequences of commands) and can be applied for creating and recognizing any behavior represented by a sequence of commands (or events). ABCD creates a user profile as a distribution of relevant subsequences and then statistical methods are applied for recognizing a given sequence of commands. However, for evaluating ABCD, the UNIX operating system environment is used.

The creation of the UNIX user profiles from a sequence of UNIX commands should consider the sequentiality of the commands typed by the user and the temporal dependencies. In a human-computer communication by commands, the sequentiality of these commands is essential for the result of the interaction. This aspect motivates the idea of automated sequence learning for computer user behavior classification; if we do not know the features that influence the behavior of a user, we can consider a series of past actions to incorporate some of the historical context of the user. ABCD can be applied for creating and recognizing any behavior profile represented by a sequence of commands; this research is focused on creating computer user profiles from a command-line interface. Specifically, ABCD is detailed using the UNIX commands environment. Fuzzy-rule-base system which satisfies all of the criteria of the incremental classifiers. Though, the approach has also important advantages which make it very useful in real environments

- It can manage with massive amounts and data.
- Its evolving structure can capture sudden and abrupt changes in the stream of data.
- Its structure meaning is very understandable, as we propose a rule-based classifier.
- It is non-iterative and single pass; so, it is computationally very efficient and fast.
- Its classifier structure is easy and interpretable.

Detect intrusion by recognizing well defined attacks patterns in user's data. In anomaly detection, usually a user profile is first built using the user's normal activity data. Any significant deviation of current user data pattern from the profile indicates possible intrusion. Since masquerader behaviors are highly unpredictable, the second path is often used in masquerader detection. EVABCD can also be used to supervise, analyze, and detect abnormalities based on a time-varying behavior of same is not considered in existing system ,we proposed our work to monitor and detects the Masquerader from user behavior profile.

- The user profile behavior sequence created by the operating system is transformed into relevant subsequences in order to find out a profile that defines its behavior
- Usage of Tree structured Support Vector Machines to detect Masqueraders.
- Motivates the idea of automated sequence learning for computer user behavior classification; Detect intrusion by recognizing well defined attacks patterns in user's data. In anomaly detection, usually a user profile is differing from normal activity data. Proposed motivate to found the detects the Masquerader from user behavior profile.

II RELATED WORK

Different techniques have been used to find out relevant information related to the human behavior in many different areas. The literature in this field is vast; Macedo et al. [5] propose a system (WebMemex) that provides recommended information based on the captured history of navigation from a list of known users. Pepyne et al. [6]describe a method using queuing theory and logistic regression modelling methods for profiling computer users based on simple temporal aspects of their behavior. In this case,

the goal is to create profiles for very specialized groups of users, who would be expected to use their computers in a very similar way. Gody and Amandi [7] present a technique to generate readable user profiles that accurately capture interests by observing their behavior on the web. There is a lot of work focusing on user profiling in a specific environment, but it is not clear that they can be transferred to other environments. However, the approach we propose in this paper can be used in any domain in which a user behavior can be represented as a sequence of actions or events. Because sequences play a crucial role in human skill learning and reasoning [8], the problem of user profile classification is examined as a problem of sequence classification. According to this aspect, Horman and Kaminka [9] present a learner with unlabeled sequential data that discover meaningful patterns of sequential behavior from example streams. Popular approaches to such learning include statistical analysis and frequency based methods. Lane and Brodley [10] present an approach based on the basis of instance-based learning (IBL) techniques, and several techniques for reducing data storage requirements of the user profile. Although the proposed approach can be applied to any behavior represented by a sequence of events, we focus in this research in a command-line interface environment. Related to this environment, Schonlau et al. [3] investigate a number of statistical approaches for detecting masqueraders. Coull et al. [11] propose an effective algorithm that uses pairwise sequence alignment to characterize similarity between sequences of commands. Recently, Angelov and Zhou propose in [12] to use evolving fuzzy classifiers for this detection task. In [13], Panda and Patra compared the performance of different classification algorithms—Naive Bayesian (NB), C4.5 and Iterative Dichotomizer 3 (ID3)—for network intrusion detection. According to the authors, ID3 andC4.5 are robust in detecting new intrusions, but NB performs better to overall classification accuracy. Cufoglu et al. [14] evaluated the classification accuracy of NB, IB1, SimpleCART, NBTree, ID3, J48, and Sequential Minimal Optimization (SMO) algorithms with large user profile data. According to the simulation results, NBTree classifier performs the best classification on user-related information. It should be emphasized that all of the above approaches ignore the fact that user behaviors can change and evolve. However, this aspect needs to be taken into account in the proposed approach. In addition, owing to the characteristics of the proposed environment, we need to extract some sort of knowledge from a continuous stream of data. Thus, it is necessary that the approach deals with the problem of classification of streaming data. Incremental algorithms build and refine the model at different points in time, in contrast to the traditional algorithms which perform the model in a batch manner. It is more efficient to revise an existing hypothesis than it is to generate hypothesis each time a new instance is observed. Therefore, one of the solutions to the proposed scenario is the incremental classifiers. An incremental learning algorithm can be defined as one that meets the following criteria [15]:

- It should be able to learn additional information from new data.
- It should not require access to the original data, used to train the existing classifier.
- It should preserve previously acquired knowledge.
- It should be able to accommodate new classes that may be introduced with new data. Several incremental classifiers have been implemented using different frameworks.

The problem of processing streaming data in online has motivated the development of many algorithms which were designed to learn decision trees incrementally [16], [17]. Some examples of the algorithms which construct incremental decision trees are: ID4 [18], ID5 [19], and ID5R [20]. IGLESIAS ET AL.: CREATING EVOLVING USER BEHAVIOR PROFILES AUTOMATICALLY 855. Artificial neural networks (ANN). Adaptive Resonance Theory (ART) networks [21] are unsupervised ANNs proposed by Carpenter that dynamically determine the number of clusters based on a vigilance parameter [22]. In addition, Kasabov proposed another incremental learning neural network architecture, called Evolving Fuzzy Neural Network (EFuNN) [23]. This architecture does not require access to previously seen data and can accommodate new classes. A new approach to incremental learning using evolving neural networks is proposed by Seipone and Bullinaria [24]. This approach uses an evolutionary algorithm to evolve some MLP parameters. This process aims at evolving the parameters to produce networks with better incremental abilities. Prototype-based supervised algorithms. Learning Vector Quantization (LVQ) is one of the well-known nearest prototype learning algorithms [25].

LVQ can be considered to be a supervised clustering algorithm, in which each weight vector can be interpreted as a cluster center. Using this algorithm, the number of reference vectors has to be set by the user. For this reason, Poirier and Ferrieux proposed a method to generate new prototypes dynamically. This incremental LVQ is known as Dynamic Vector Quantization (DVQ) [26]. However, this method lacks the generalizing capability, resulting in the generation of many prototype neurons for applications with noisy data. Bayesian. Bayesian classifier is an effective methodology for solving classification problems when all features are considered simultaneously. However, when the features are added one by one in Bayesian classifier in batch mode in forward selection method, huge computation is involved. For this reason, Agrawal and Bala [27] proposed an incremental Bayesian classifier for multivariate normal distribution data sets. Several incremental versions of Bayesian classifiers are proposed in [28].

Support Vector Machine (SVM). A Support Vector Machine performs classification by constructing an N-dimensional hyperplane that optimally separates the data into two categories. Training an SVM "incrementally" on new data by discarding all previous data except their support vectors, gives only approximate results. Cauwenberghs et al. consider incremental learning as an exact online method to construct the solution recursively, one point at a time. In addition, Xiao et al. [29] propose an incremental algorithm which utilizes the properties of SV set, and accumulates the distribution knowledge of the sample space through the adjustable parameters. However, as this research focus in a command-line interface environment, it is necessary an approach able to process streaming data in real time and also cope with huge amounts of data. Several incremental classifiers do not consider this last aspect. In addition, the structure of the incremental classifiers is assumed to be fixed, and they cannot address the problem of so-called concept drift and shift [30].

By drift, they refer to a modification of the concept overtime, and shift usually refers to sudden and abrupt changes in the streaming data. To capture these changes, it is necessary not only tuning parameters of the classifiers, but also a change in its structure. A simple incremental algorithm does not evolve the structure of the classifier. The interpretation of the results is also an important characteristic in a classifier, and several incremental classifiers (such as ANN or SVM) are not good in terms of interpretation of the results. Finally, the computational efficiency of the proposed classifier is very important, and some incremental algorithms (such SVM) have to solve quadratic optimization problems many times. Taking all these aspects into account, we propose in this paper an evolving fuzzy-rule-base system which satisfies all of the criteria of the incremental classifiers. However, the approach has also important advantages which make it very useful in real environments.

- It can cope with huge amounts and data.
- Its evolving structure can capture sudden and abrupt changes in the stream of data.
- Its structure meaning is very clear, as we propose a rule-based classifier.
- It is non-iterative and single pass; therefore, it is computationally very efficient and fast.
- Its classifier structure is simple and interpretable.

Thus, an approach for creating and recognizing automatically the behavior profile of a computer user with the time evolving in a very effective way. To the best of our knowledge, this is the first publication where user behaviour is considered, treated, and modelled as a dynamic and evolving phenomenon. This is the most important contribution of this paper.

III EXISTING SYSTEM

Evolving Agent behavior Classification based on Distributions of relevant events (EVABCD) and it is based on representing the observed behavior of an agent (computer user) as an adaptive distribution of her/his relevant atomic behaviors (events). Once the model has been created, EVABCD presents an developing method for updating and evolving the user profiles and classifying an observed user. The approach we present is generalizable to all kinds of user behaviors represented by a sequence of events. The creation of a user profile from a sequence of UNIX commands should consider the consecutive order of the commands typed by the user and the influence of his/her past experiences. This feature motivates the idea of automatic sequence learning for computer user behavior classification

Existing approach includes at each step the following two main actions:

1. Creating and evolving the classifier. This achievement involves in itself two subactions

a. Creating the user behavior profiles. This *subaction* analyzes the sequences of commands typed by different UNIX users online (data stream), and creates the matching profiles.

b. Evolving the classifier. This subaction includes online learning and update of the classifier, including the possible of each behavior to be an example, stored in the EPLib.

2. User classification. The user profiles created in the previous action are associated with one of the prototypes from the EPLib, and they are classified into one of the classes formed by the prototypes.

Disadvantages

- Masquerader detection is not performed.
- Time varying user behavior is not considered.

IV PROPOSED APPROACH

Masqueraders are people who impersonate other people on a computer system and they pose threat to the system security. We proposed a masquerader detection using UNIX command sequences. First we set the threshold that describes how widely a user's normal behavior varies over time. One threshold was set for each user by getting the average probability density distances between each pair of adjacent command sequences. We detect masquerader by comparing current user variation with this threshold and trigger the alarm when a sequence of commands shows a big deviation from the user normal behaviour.

Tree structure SVM can partitioned regions have different characteristics, by taking this architecture the SVMs experts that best fit particular regions that is best classifying an observed user and masquerader by choosing the most appropriate kernel function and the optimal learning parameters of SVMs will be used for the final prediction.

Advantages

- Detect Masqueraders Using UNIX Command Sequences.
- Improves the performance of the creating User Behavior Profiles
- Improves the performance of the creating User Behavior Profiles with best classification using tree structure SVM.

Architecture diagram

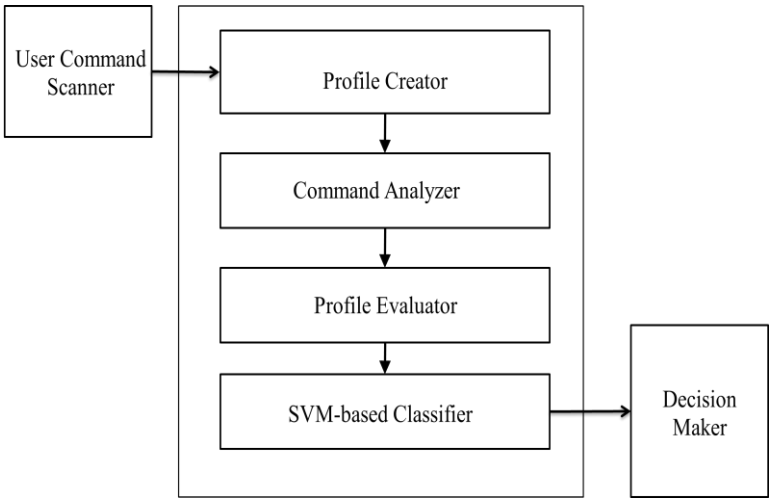

Figure 4.1 Architecture diagram of Detecting Masqueraders

A. Data Preprocessing

Command-line data collected by Greenberg using UNIX csh command interpreter. This data are identified in four target groups which represent a total of 168 male and female users with a wide cross section of computer experience and needs. The four target groups are Novice programmers, experienced programmers, computer scientist, and non-programmers. The non-voice programmers' users of this group had little or no previous exposure to programming, operating systems, or UNIX-like command-based interfaces. Experienced programmers are group members were senior Computer Science under graduates, expected to have a fair knowledge of the UNIX environment. Computer scientist is group, graduates and researchers from the Department of Computer Science, had varying experience with UNIX, although all were experts with computers.

B. Naive Bayes Classifier (Nb)

In this module Naive Bayes classifier (NB) is a simple probabilistic classifier based on applying Bayes theorem with strong (naive) independence assumptions. Naive Bayes classifier assumes that the presence of a particular feature of a class is unrelated to the presence (or absence) of any other feature. In this case, the numeric estimator precision values are chosen based on analysis of the training data. For this reason, the classifier is no incremental. In this Naive Bayes classifier (NB) accurately predict the class of the test instance and training results includes the test result.

The naive Bayes classifier assigns an instance s_k with attribute values (A_1=v_1, A_2=v_2, …, A_m=v_m) to class C_i with maximum Prob (C_i|(v_1, v_2, …, v_m)) for all *i*.

The naive Bayes classifier exploits the Bayes's rule and assumes independence of attributes.

Likelihood of s_k belonging to C_i

$$= \text{Prob}\left(C_i \mid (\text{v}_1, \text{v}_2, ..., \text{v}_m)\right) = \frac{P\left((\text{v}_1, \text{v}_2, ..., \text{v}_m) \mid C_i\right) P(C_i)}{P\left((\text{v}_1, \text{v}_2, ..., \text{v}_m)\right)}$$

Likelihood of s_k belonging to C_j

$$= \text{Prob}\left(C_j \mid (\text{v}_1, \text{v}_2, ..., \text{v}_m)\right) = \frac{P\left((\text{v}_1, \text{v}_2, ..., \text{v}_m) \mid C_j\right) P(C_j)}{P\left((\text{v}_1, \text{v}_2, ..., \text{v}_m)\right)}$$

Therefore, when comparing Prob(C_i| (v_1, v_2, …, v_m)) and P(C_j |(v_1, v_2, …, v_m)), we only need to compute P((v_1, v_2, …, v_m)|C_i) P(C_i) and P((v_1, v_2, …, v_m)|C_j)P(C_j)

- Under the assumption of independent attributes Furthermore, $P(C_j)$ can be computed by

$$P\left((\text{v}_1, \text{v}_2, ..., \text{v}_m) \mid C_j\right)$$
$$= P(A_1 = \text{v}_1 \mid C_j) \cdot P(A_2 = \text{v}_2 \mid C_j) \cdot \ldots \cdot P(A_m = \text{v}_m \mid C_j)$$
$$= \prod_{h=1}^{m} P(A_h = \text{v}_h \mid C_j)$$
$$\frac{\text{number of training samples belonging to } C_j}{\text{total number of training samples}}$$

C. K Nearest Neighbor Classifier

In this module K Nearest Neighbor classifier (kNN) is an instance-based learning technique for classifying objects based on closest training examples in the feature space. The parameter k affects the performance of the kNN classifier significantly. kNN classifier adopts an instance-based approach whose conceptual simplicity makes its adaptation to an incremental classifier straightforward. There is an algorithm called incremental kNN as the number of nearest neighbor's k needs not be known in advance and it is calculated incrementally. However, in this case, the difference between Incremental kNN and nonincremental kNN is the way all the data are processed.

Input: D the set of k training objects and test objects $Z = x', y'$

Process: Compute $d(x', x)$ the distance between Z and every point ,(x, y) ∈ D

Select $D_z \subseteq D$ the set of k closest training objects to Z.

Output: $y' = arg \max_v \sum_{(x_i,y_i)\in D_Z} I(v = y_i)$

Description

Given a training set D and a test object x = (x', y'), the algorithm computes the distance (or similarity) between z and all the training objects (x, y) ∈ D to determine its nearest-neighbor list, Dz.

- X is the data of a training object, while y is its class.
- x' is the data of the test object and y' is its class.

Once the nearest-neighbor list is obtained, the test object is classified based on the majority class of its nearest neighbors:

$$y' = arg \max_v \sum_{(x_i,y_i)\in D_Z} I(v = y_i)$$

v is a class label, y_i is the class label for the ith nearest neighbors, I (·) is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

D. Decision Tree Using C4.5 Algorithm

A decision tree induction algorithm C4.5 which is C4.5 algorithm utilizes the information theoretic approach in classifying a network traffic pattern. The decision tree is initially created from the pre-classified dataset. Each instance is defined by values of the attributes. A decision tree consists of nodes, edges and leaves. A node of a decision tree identifies an attribute by which the instance is to be partitioned. Every node has a number of edges, which are labeled according to a potential value of edges and a probable value of the attribute in the parent node. An edge links either two of the nodes in a tree or a node and a leaf. Leaves are labeled with class labels for classification of the instance. Information gain is calculated for each of the attribute. The best attribute to divide the subset at each stage is selected using the information gain of the attributes. If the value of attributes is nominal then a branch for each value of the attribute is formed, but if it is numeric a threshold value is determined and two branches are created. This procedure is recursively applied to each partitioned subset of the instances. The procedure ceases when all the instances in the current subset belong to the same class.

C4.5 algorithm handling the attributes with high information gain (IG) is consider as the root node to construct the decision tree with the rules generated with each class values.C4.5 algorithm is best method for Dealing with Missing Values .In C4.5 algorithm training samples are found using set of class $\{C_1, C_2,\ldots, C_k\}$. There are three possibilities for the content of the set of training samples T in the given node of decision tree.

- T contains one or more samples, all belonging to a single class C_j. The decision tree for T is a leaf identifying class C_j
- T contains no samples.

The decision tree is again a leaf, but the class to be associated with the leaf must be determined from information other than T, such as the overall majority class in T. C4.5 algorithm uses as a criterion the most frequent class at the parent of the given node.

3. T contains data samples that belong to a mixture of classes.

In this situation, the idea is to refine T into subsets of samples that are heading towards single- class collections of samples. An appropriate test is chosen, based on single attribute, that has one or more mutually exclusive outcomes $\{O_1,O_2, \ldots,O_n\}$

T is partitioned into subsets $T_1, T_2, \ldots, T_n$ where T_i contains all the samples in T that have outcome O_i of the chosen test. The decision tree for T consists of a decision node identifying the test and one branch for each possible outcome.

Test – entropy:

If *S* is any set of samples in healthcare data, let *freq* (C_i, *S*) stand for the number of samples in S that belong to class C_i , and $|S|$ denotes the number of samples in the set *S*. Then the entropy of the set *S*, Info(S) = - $\sum$ ((freq(C_i, S)/ $|S|$) · $\log_2$ (freq(C_i, S)/ $|S|$)) After T set of the samples is partitioned according with the n outcomes of attribute set X

$Info_x(T) = \sum ((|Ti| / |T|) \cdot Info(Ti))$
$Gain(X) = Info(T) - Info_x(T)$
Select an attribute with the highest Gain value.

E. Tree Structured Svm

A tree-structured architecture is adopted in the partition to avoid the problem of predetermining the number of partitioned regions. Then, in the second stage, multiple SVMs, also called SVM experts, that best fit partitioned regions are constructed by finding the most appropriate kernel function and the optimal free parameters of SVMs. Training data points which have similar characteristics in the input space will be classified into the same region. As the partitioned regions have more uniform distributions than that of the whole input space, it will become easier for a SVMs expert to capture such a more stationary input–output relationship between the original user and output user results of the profiles and evolving user profiles. By taking this architecture the SVMs experts that best find particular regions by choosing the most appropriate kernel function and the optimal learning parameters of SVMs will be used for the final prediction. This is very different from a single SVMs model that learns the whole input space globally and thus cannot guarantee that each local input region is the best learned.

Input: Number of the training samples with dataset w as input data point for SVM classification

Output: Classification result and prediction of the masquerader detection result

Procedure SVM (w) // input training data results from the SVM classification

Begin

Initialize C=0 //initially the class labels should be zero

Get input fileintrusion dataset w for training //the input dataset result as the example for training the user data and prediction results of the masquerader result

Read the number of input training dataset W from original dataset

$x_i.w + b = 0$ ////Input input training dataset W is represented as matrix and denoted by x_i and w is the weight value matrix whose product is summed with bias value to give the class value.

$x_i.w + b = 1$ // This above equation marks a central classifier margin. This can be bounded by soft margin at one side using the following equation.

Decision function $f(W) = x_i.w - b$ //decision function f(w) decides the class labels for the SVM classification training examples ,

If $f(W) \geq 1$ for x_i is the first class // if the F(w) is greater than or equal to the 1 is labeled as first class

Else

$f(W) \leq -1$ for x_i is the second class // if the f(w) is less than or equal to the value of -1 is labeled as second class

The prediction result for (i=1,…n) number of training samples //after the classification result are performed then check the classification result by testing phase it is check the below function

$y_i(x_i.w - b) \geq 1$ //if the function is greater than one the results as predicted result of intruder/normal user

Display the result //finally we display the classification result .

F. Masqueraders Detection Approach

Detection approach is: a user uses a relatively fixed set of commands. Given a command sequence accumulated during a specific time period, the proportion each of those commands occurred is also user specific. We can use a vector of attributes representing the proportional occurrences of each command in a sequences and we expect these proportional occurrences vector can describe a user's behavior during a specific time. A user's behavior is not fixed but keeps changing by the user's adaptive behavior. Measure this user behavior variation by looking at the distance between two proportional occurrence vectors. Break a user's command sequence into blocks in some natural way .Build proportional occurrence vectors for those blocks and learn the user's behavior adaptation rate by looking at how fast.

V RESULTS

The proposed detection approach is evaluated in an environment in which each user behavior is represented as a sequence of UNIX commands and how the system detects the Masqueraders using UNIX sequence commands**.**

VI CONCLUSION

In this system the EVABCD, to model and classify user behaviors from a sequence of events. The underlying assumption in this approach is that the data collected from the corresponding environment can be transformed into a sequence of events. This sequence is segmented and stored in a trie and the relevant subsequences are evaluated by using a frequency-based method. Tree-structured SVM architecture is that by specifying a partition condition, it could automatically find a suitable network structure that is the normal user and masquerader detection in the user profile. Detection of anomalies from the time-varying behavior changes occurs at normal user, proposed our work to monitor and detects the Masquerader from user behavior profile. Proposed system detects masqueraders by looking at the classifiers misclassification behavior from normal user. Detect masquerader by comparing current user variation and improves the performance of user behavior profile.

ACKNOWLEDGEMENT

The authors would like to thank the staffs and students of SriGuru Institute of technology for their valuable support and guidance.

REFERENCES

[1] D. Godoy and A. Amandi, "User Profiling in Personal Information Agents: A Survey," Knowledge Eng. Rev., vol. 20, no. 4, pp. 329 361, 2005.
[2] J.A. Iglesias, A. Ledezma, and A. Sanchis, "Creating User Profiles from a Command Line Interface: A Statistical Approach," Proc. Int'l Conf. User Modeling, Adaptation, and Personalization (UMAP),pp. 90 101, 2009.
[3] M. Schonlau, W. Dumouchel, W.H. Ju, A.F. Karr, and Theus, "Computer Intrusion: Detecting Masquerades," Statistical Science, vol. 16, pp. 58 74, 2001. 12
[4] R.A. Maxion and T.N. Townsend, "Masquerade Detection Using Truncated Command Lines," Proc. Int'l Conf. Dependable Systems and Networks (DSN), pp. 219 228, 2002.
[5] A. Alaniz Macedo, K.N. Truong, J.A. Camacho Guerrero, and M. Graca Pimentel, "Automatically Sharing Web Experiences through a Hyper document Recommender System," Proc. ACM Conf. Hypertext and Hypermedia (HYPERTEXT '03), pp. 48 56, 2003.
[6] D.L. Pepyne, J. Hu, and W. Gong, "User Profiling for Computer Security," Proc. Am. Control Conf., pp. 982 987, 2004.
[7] D. Godoy and A. Amandi, "User Profiling for Web Page Filtering," IEEE Internet Computing, vol. 9, no. 4, pp. 56 64, July/ Aug. 2005.
[8] J. Anderson, Learning and Memory: An Integrated Approach. John Wiley and Sons, 1995.
[9] Y. Horman and G.A. Kaminka, "Removing Biases in Unsupervised Learning of Sequential Patterns," Intelligent Data Analysis, vol. 11, no. 5, pp. 457 480, 2007.
[10] T. Lane and C.E. Brodley, "Temporal Sequence Learning and Data Reduction for Anomaly Detection," Proc. ACM Conf. Computer and Comm. Security (CCS), pp. 150 158, 1998.

Authors Profile

P.Anishprabu was born in Namakkal on 14th May 1990. He received her B.Tech.(IT) degree from KSR College of Engineering, Tiruchengode, Tamil Nadu in 2011. He is currently pursuing M.E. (CSE) degree in SriGuru Institute of Technology, Coimbatore, Tamil Nadu. He has presented a papers in various conferences. He is interested in Data mining, Operating system.

M Sangeetha was born in Theni on 26th December 1990. She received her B.Tech.(IT) degree from Periyar Maniammai University, Thanjavur, Tamil Nadu in 2012. She is currently pursuing M.E. (CSE) degree in SriGuru Institute of Technology, Coimbatore, Tamil Nadu. She has Published articles in various international journals. She is interested in Secure Computing, Data mining, Audio mining.

M.Nagulanand was born in Salem on 10th June 1990. He received her B.E.(CSE) degree from Coimbatore Institute of Engineering and Technology, Coimbatore, Tamil Nadu in 2011. He is currently pursuing M.E. (CSE) degree in SriGuru Institute of Technology, Coimbatore, Tamil Nadu. He has presented papers in various conferences. He is interested in Data mining, Operating system.

