Proximity User Identification Using Correlogram

Shervin Shahidi[1], Parisa, Mazrooei[1], Navid Nasr Esfahani[2], Mohammad Saraee[3]

[1]*Intelligent Databases, Data Mining and Bioinformatics Research Laboratory*
Department of Electrical and Computer Engineering
Isfahan University of Technology, Isfahan, Iran 84156-83111

[2]*Isfahan Mathematics House, Isfahan, Iran 81645-356*

[3]*School of Computing, Science and Engineering, University of Salford, Greater Manchester, UK*

sh.shahidi@ec.iut.ac.ir, p.mazrooei@ec.iut.ac.ir, navid@ec.iut.ac.ir, m.saraee@salford.ac.uk

ABSTRACT. *This paper represents a technique, applying user action patterns in order to distinguish between users and identify them. In this method, users' actions sequences are mapped to numerical sequences and each user's profile is generated using autocorrelation values. Next, cross-correlation is used to compare user profiles with a test data. To evaluate our proposed method, a dataset known as Greenberg's dataset is used. The presented approach is succeeded to detect the correct user with as high as 82.3% accuracy over a set of 52 users. In comparison to the existing methods based on Hidden Markov Model or Neural Networks, our method needs less computation time and space. In addition, it has the ability of getting updated iteratively which is a main factor to facilitate transferability.*

KEYWORDS: *User Identification- Correlation- Classification- Pattern Recognition*

1. Introduction

The task of distinguishing the current system's user from other users, i.e. User Identification, is one of the issues which have been studied in many fields such as Human Computer Interaction (HCI) and Computer Security. As an instance, User Identification has tight resemblance with intrusion detection, which is the key to solve two important problems in computer security: Detecting the presence of an intruder masquerading as a valid user (*anomaly detection*) and detecting the perpetration of abusive actions on the part of an otherwise innocuous user (*misuse detection*). (Lane *et. al.*, 1997)

On one point of view, different approaches to User Identification problem are classified as below:

- Detection based on supervised learning; in which labelled data is needed to train the agent. These approaches build a model over the normal data and then check to see how well new data fits into that model. (Warrender *et. al.*,1999) As an instance, in network security field, a technique developed at SRI in the EMERALD system (Javitz *et. al.*,1993) compares the distribution of test data to a previously made distribution over a known user data to indicate an intrusion. The problem that causes supervised learning not to be always applicable is that preparing the labelled training data is sometimes difficult and expensive.(Eskin *et. al.*, 2002)
- Detection based on unsupervised learning; which uses unlabeled data for anomaly detection. The overall idea of these approaches is based on two assumptions. First, the number of normal users enormously outnumbers the number of intruders in a large data set. Second, the set of user actions are different from each other, in a way that each user's profile can be classified as a dense region in a proper feature space. The feature space is an n-dimensional metric space, to which user actions are mapped. In intrusion detection systems, this approach is used as identifying sparse regions in a feature space and using proper metric, assuming that normal actions will tend to appear as dense regions in the same feature space, as in (Knorr *et. al.*, 1998) , (Knorr *et. al.*, 1999), (Breunig *et. al.*, 2000) and (Portnoy *et. al.*, 2001). Although this assumption seems to be sensible, this is not always the case. For example, for a network being attacked too many times in the same way (e.g. a network under DOS attack), the anomaly patterns are so dense that cannot be distinguished as sparse regions. The same situation holds for user identification; the assumption of neatly classifiable regions for different users is not always correct.

The problem of user identification can be solved using different Machine Learning techniques; some learners try to generate a model consisting of a group of rules based on different attributes of user action history. Any new set of actions will be compared to the models to find out deviations from each user's profile. For instance, a rule learning program, named RIPPER is used to achieve such model in (Lee et. al.,1998). Some probabilistic methods have also been used to make a

predictive model of user's next actions, and then, computing how well new data fits into the model (Eskin *et. al.*, 2001). Markov Chains and Hidden Markov Models are also two widely used methods in this field. (Nong Ye, 2000)(Galassi *et. al.*, 2005)

In this paper, we have considered the set of user actions as samples of signals and have used some signal processing techniques to identify the user. This approach is considered as a supervised learning approach, since we build user profiles over labelled training data and then, compare the profile patterns to test data patterns for getting results.

The training data is the stream of commands typed by each user in Unix *csh.*, saved as offline log files. We place our concentration on the importance of action sequences rather than the attributes calculated by the set of actions (e.g. user activity rate, mistake rate or word usage distribution). Based on the hypothesis that humans' set of actions are causal, autocorrelation and cross-correlation are used for a moving window on the sequence of actions to quantify the amount of serially performed actions. Finally, we compare the test sequence with previously built profiles using Mean Square Error (MSE) criterion to find the best profile matching with the test data.

The organization of the rest of this paper is as follows. In section 2 we introduce our general scheme along with some basic aspects of our model. Section 3 represents empirical results. Analysis of the results is provided in section 4. Finally, Conclusion and future works are presented in section 5.

2. General Scheme

2.1. *Problem Model*

The human-computer interaction is essentially a causal process. (Lane et. al., 1997) Therefore, some specific patterns can be found is a user's action sequence. Relying on this fact and assuming that these series of patterns vary between different users, we extract the repeated patterns in an action sequence and use these patterns for user identification.

For this purpose, after mapping the string of actions to a string of numerical codes, we define a moving window which takes a subsequence of a user action history each time, to be used as a unit for modelling and processing user behaviour. The window size determines the maximum distance of two related commands. Since user action patterns which follow a specific goal are not usually very long, it is reasonable not to consider the relevance of all of the commands in the sequence. To consider the dependence of sequential commands at the end of one window and the beginning of its successor, windows can overlap.

As a measure of self similarity, we calculate autocorrelation for each window. In other words, autocorrelation of a window and its cross-correlation with another window will be a scale to compare it with the other window. In extreme case, if the two windows contain the same sequence of commands, the autocorrelation of one of them is the same as their cross-correlation. Plotting the autocorrelation Coefficients against lag ($\boldsymbol{\tau}$) gives a Correlogram indicating different autocorrelation values for each lag value $\boldsymbol{\tau}$. Technically, Autocorrelation Coefficients represent the correlation of a sequence's elements with certain distance apart.

Correlation coefficient for lag value $\boldsymbol{\tau}$ and time *t* can be computed as:

$$R_{XY}(\tau) = \frac{E[(X_t - \mu_X)(Y_{t+\tau} - \mu_Y)]}{\sigma_X \sigma_Y} \qquad [1]$$

Where $\boldsymbol{E}$ indicates the Expected value operator, μ is the mean value and $\boldsymbol{\sigma}$ is the standard deviation for $\boldsymbol{X}$ and $\boldsymbol{Y}$ random variables. The value $\boldsymbol{R_{XX}}$or simply $\boldsymbol{R_X}$ is called auto-correlation coefficient. In signal processing, the above definition is often used without the normalization, that is, without subtracting the mean and dividing by the variance. In this work, we did the same, since comparisons are made between different Correlograms, and normalization decreases the distance between different user profiles which will lead to less classification accuracy. Therefore, the correlation for two arbitrary windows is calculated via the following formula:

$$R_{XY}(\tau) = E[(X_t)(Y_{t+\tau})] = \begin{cases} \sum_{t=0}^{N-\tau-1} X_t Y_{t+\tau} & \tau \geq 0 \\ R(-\tau) & \tau < 0 \end{cases} \qquad [2]$$

Note that since the data is inherently non-numerical, only similarity of two elements should be taken into account. Hence, the multiplication between two elements of X and Y is defined as:

$$X_i Y_j = \begin{cases} 1 & if\ X_i = Y_j \\ 0 & if\ X_i \neq Y_j \end{cases} \qquad [3]$$

Calculating the correlograms for each window in a user action sequence, we can store each user's profile, which consists of user action history and its correlograms. To be able to compare a user's profile with a test data -which is in the form of a user action sequence broken into overlapping windows-, we calculated the cross-correlation between each window in the test data sequence and every window in a user profile. These correlation values will be used for classification.

2.2. Classification

To compare cross-correlations obtained from a test data with users' profiles autocorrelations, Mean Squared Error (MSE) is used as a measure of distortion:

$$D = E[\| R_X - R_{XY} \|^2] = \frac{1}{2N-1} \sum_{\tau=-N}^{\tau=N} \left(R_X(\tau) - R_{XY}(\tau)\right)^2 \quad [4]$$

where N is the length of the windows.

The window in profile which has minimum distortion *D* with the window in test data is chosen as the matching window and the distortion is saved as a penalizing factor. Average distortion of the test data from user profile is calculated over all of the matching windows' distortions. This job is done for all of the user profiles and the average distortion is considered as a discriminant function for classification; That is, the profile with the minimum average distortion is considered as the resulting profile.

Instead of taking average over all distortions, another method was used for classification by means of confusion matrix. Confusion matrix is a well-known visualization method in supervised learning, in which each row of the matrix represents the correct class and the columns reveal the predicted classes according to the classifier. Note that confusion matrix is used when there are several test data's available, but here we have modified it to be used for classification over one test data. In this method, each window of the test data is classified independently similar to the above method and the resulting class for each window will be rewarded one point. Finally, the class with maximum number of points will be considered as the resulting class. Figure 1. reveals a confusion matrix for some test data.

3. Experimental Results

Method represented in this paper has been tested by a data set of user actions collected by Greenberg (Greenberg S. , 1988). This data set includes actions of 168 people in four groups of Computer Scientists, Expert Programmers, Novice Programmers and Non Programmers, typing in UNIX csh. These logs contain quite complete information about user actions such as commands, start and end times login sessions, working directory, etc.. For our approach, only user commands have been used. In order to form each user action history, all commands used by each group were coded into integer values separately. Table 1. gives information about each group's number of users and number of different commands used. In order to consider occurrence of wrong commands, they are coded as another occurrence of

the first subsequent correct command; in this way, typing wrong commands will be shown as multiple occurrences of a command.

Table 1.*Data set information*

	Number of users	Different commands
Non-programmers	25	196
Computer scientist	52	851
Experienced programmers	36	588
Novice programmers	55	264
Overall	168	1307

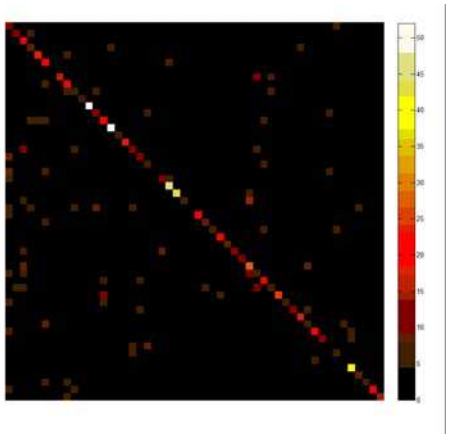

Figure 1. *Confusion matrix for Computer Scientists*

With regards to the fact that human actions are not stationary, using the whole user log would decrease the accuracy of identification. In other words, since users' behavior change through the time, their actions should be compared with more recent logs. To examine this ability in our method only last part of user logs have been considered. This last part has been defined to be of the size Maximum Log Length (MLL), so that the preceding actions in the log will be neglected. Using last *MLL* commands of the sequence in the algorithm has also the advantage of less computation and storage requirements.

In order to check the algorithm's efficiency we examined it for different values of variables, Window Size (*W*), Window Shift (W_s)-which indicates the amount of window overlaps- and Maximum Log Length (*MLL*). The abovementioned method has been run for all possible combinations of *W= 10, 20, 40,* W_s*=W/2, W/4* and *MML= 1000, 2000, 4000*. To avoid accidental results, 5-fold cross-validation is used, in which user logs are partitioned to five subsets and each subset is considered as test data, while the remaining is considered to be training data. The results of using average distortion and confusion matrix for classification are shown in Table 2. and Table 3. respectively.

Table 2. *Classification using Average distortion*

	W	W_s	MLL	Accuracy
Exp	10	3	2000	73.3%
Non	10	3	2000	59.2%
Novice	10	3	2000	45.8%
Scientist	10	3	1000	82.3%
Overall	10	3	2000	61.7%

Table 3. *Classification using confusion matrix*

	W	W_s	MLL	Accuracy
Exp	10	3	2000	71.6%
Non	10	3	2000	56%
Novice	20	5	2000	33.1%
Scientist	10	3	2000	80%
Overall	10	3	2000	54.3%

To test the method, we used each one of the four groups as a separate data set, so that the algorithm should recognize a member of each group in its own group. This gives us the opportunity to test it four times. Since these groups noticeably vary, we assumed that if the agent is able to find a computer scientist in a group of computer scientists (for instance), it would obviously be able to find it among all 168 users. To justify our assumption, we have also tried our method over all users to classify them to either correct user and correct group(Table 2., 3. last rows and table 4.). As can be seen, the method can classify the user in his right group with more than 90% average accuracy, in spite of no specific training for group classification and without any information about the groups of users.

Table 4. *Classification of users to their corresponding groups*

Predicted class / Target class	Exp%	Non%	Novice%	Scientist%
Exp	91.1	0.5	1.7	6.7
Non	4.8	86.4	1.6	7.2
Novice	1.8	0	97.8	0.4
Scientist	2.3	0.8	1.1	95.8

4. Analysis

To gain a deeper understanding of used methods, a brief discussion will be provided in this section.

4.1. *Analysis of the classification results:*

According to Table 2. and Table 3. some interesting results can be deduced. As can be seen, there is a huge gap between the classification accuracy of the groups "experienced programmers" and "computer scientists" with the groups "non programmers" and "novice programmers". We believe this difference is due to the fact that the two former mentioned groups use a wider variety of commands, use the system for a bigger set of goals and do more complicated tasks; hence, there exists a bigger implicit difference between the users' actions in these groups. On the other hand, the two latter mentioned groups use a smaller set of commands to achieve simpler and fewer tasks, which can be very common with their group-mates.

4.2. *Maximum Log Length of Command Sequences:*

As previously mentioned, using MLL limit on commands sequence in our method, provides a better basis for the algorithm to distinguish between users. As an evidence, the optimum size for MLL never exceeded 2000, while the method was also tested for size 4000. Another advantage of putting this limit is to decrease the required time and space for generating and storing profiles. Furthermore, this method allows us to update the profiles easily by calculating only the autocorrelations for the updated part of log files, while by removing the oldest

autocorrelations we keep the complexity of classification invariant. This is an advantage over a lot of proposed algorithms which need to generate the whole profile over, each time the log files are updated. Since calculating an autocorrelation is not a process of high time complexity, the aforementioned procedure can be done as an online process. Enhancing the system by this attribute would promote the system to an online supervised model, which can be used in real-time applications.

4.3. *Confusion Matrix:*

As mentioned briefly before, a confusion matrix consists of some rows indicating the target classes and the same number of columns (usually) for the predicted classes. Putting the corresponding rows and columns of target and predicted classes in the same order, each correct classification will appear on the diagonal of the matrix. Therefore in general, a perfect classification should result a diagonal matrix. Here, we used a modified version of confusion matrix, in which instead of classifying over each test data, we have done the classification on the consisting windows of the test data to form the confusion matrix. As a result of this modification, we have to take all of the classification results into account at the same time. Hence, for a perfect classifier for this modified matrix it is only needed to have the maximum value of each row on the diagonal. As it is shown in Figure 1. the confusion matrix for computer scientists has this property in most of the rows, although the proposed method did not work as good as the average distortion method for classification. (Tables 2. and 3.)

5. Conclusion

We introduced a new supervised learning algorithm based on using correlograms. Mapping users' history of commands to numerical values, autocorrelation is used to generate each user's profile. Afterwards, cross-correlation is used to compare a test data with user profiles. Based on Mean Squared Error, two measures of similarity were applied to find the matching profile.

We have presented experimental results supporting the applicability of our method on Greenberg data set. While this method preserves satisfactory accuracy (up to 82.3% accuracy within 52 different classes), it needs less computation time and the ability of being updated easily in comparison with other methods. In this paper, we focused only on taking advantage of the command patterns used by a user

for identification. Extracting other information out of user action history can be the subject of the future works.

6. Acknowledgement

The authors would like to thank Mr. Ali Bakhshali for his kind helps. Also to thanks Dr. S. Greenberg who shared his dataset.

7. References

E skin E., Lee W., Stolfo S., "Modeling system calls for intrusion detection with dynamic window sizes", *DARPA Information Survivability Conference and Exposition II (DISCEX II)*, Anaheim, vol. 1, 2001(page 165- 175).

Ye N., "A markov chain model of temporal behavior for anomaly detection", *the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, 2000, p. 171-174.

Lane T., Brodley C., "Sequence matching and learning in Anomaly Detection for computer security", *The Fourth National Conference on Artificial Intelligence*, 1997, p. 43-49.

Warrender C., Forrest S., Pearlmutter B., "Detecting intrusions using system calls: alternative data models", *1999 IEEE Symposium on Security and Privacy*, 1999, p. 133-145.

Javitz H., Valdes A., The NIDES statistical component: description and justification. Computer Science Laboratory, SRI International, Tech. Report, 1993.

Knorr E., Ng R., "Algorithms for mining distance-based outliers in large data sets", 24th Int. Conf. Very Large Data Bases, VLDB, Technique et Science Informatiques**,** 1998, p. 392-403.

Knorr E., Ng R., "Finding international knowledge of distance-based outliers", *The VLDB Journal*, 1999, p. 211-222.

Breunig V., Kriegel H., Ng R., Sander J., " LOF: identifying density-based local outliers ", *ACM SIGMOD Int. Conf. on Management of data*, 2000, p. 93-104.

Portnoy L., Eskin E., Stolfo S., "Intrusion detection with unlabeled data using clustering", *ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*,Philadelphia, PA,2001.

Galassi U., Giordana A., Saitta L., Botta M., "Learning Profiles Based on Hirarcical Hidden Markov Model", ISMIS 2005, LNAI 3488, pp. 47-55, 2005.

Link H., Lane T., Magliano J., “Models and Model Biases for Automatically Learning Task Switching Behavior”, In *Proceedings of the 2005 HCI International (HCII) Conference on Augmented Cognition*. HCI International (HCII) 2005.

Lee W., Stolfo S., “Data mining approaches for intrusion detection”, *1998 USENIX Security Symposium* ,1998.

Greenberg S., "Using Unix: collected traces of 168 users". *Research Report 88/333/45, includes tar-format cartridge tape. Department of Computer Science, University of Calgary, Alberta.1998.*