

# User Authentication via Mouse Biometrics and the usage of Graphic User Interfaces: An Application Approach

**J. Octavio Gutiérrez-García**  
CINVESTAV-IPN

Unidad Guadalajara  
Av. Científica 1145, Col. El Bajío,  
45010 Zapopan, Jal., México  
jgutierrez@gdl.cinvestav.mx

**Félix F. Ramos-Corchado**  
CINVESTAV-IPN

Unidad Guadalajara  
Av. Científica 1145, Col. El Bajío,  
45010 Zapopan, Jal., México  
framos@gdl.cinvestav.mx

**Herwig Unger**

FernUniversität in Hagen  
58084 Hagen, Germany  
herwig.unger@FernUni-Hagen.de

**Abstract** - *In this paper we propose the construction of the user behaviour based on a relatively new mouse-based behavioural biometric and the usage of graphic user interfaces (GUIs) utilizing the application approach, pretending to use this behaviour to transparently re-authenticate the users with a free interaction in the computer system using an anomaly detection approach, avoiding the user masquerading that could derive other kind of attacks. We also present several improvements to related works such as the reduction of the number of features to construct the user behaviour and the reduction of the required data in the enrolment phase of the users, resulting in a lower computational cost; in addition we implemented a feedback loop which captures the variations of the user behaviour through the time, and thus extends the validity of the user behaviour.*

**Keywords:** User authentication, mouse biometrics.

## 1 Introduction

The masquerade attack by itself is a serious form of computer abuse, moreover it can be used as an initial point to perpetrate others attacks, therefore methods for user authentication should be reinforced, in order to do this, we propose a method to authenticate users once they have logged in, our approach is based on user's behavioural characteristics extracted from the computer mouse, since are more suitable for establishing a re-authentication process through the interaction with the computer systems every determined period of time. By building the user behaviour with mouse information which also provides relevant information of the usage of GUIs, we can acquire the data transparently and process it with a low computational cost. Now, given that our approach is based on behavioural characteristics we considered the variations on the user behaviour depending of the environment that surround them, therefore, it's supposed the existence of changes in their behaviour with different applications, consequently we profile users in each application, helping thus our objective of authenticating users transparently in an un-intrusive manner. In addition a feedback loop was

implemented, with the aim of extending the validity of the user behaviour.

The paper is structured as follows: section 2 discusses related works; section 3 presents the feature extraction of the mouse data for the definition of the user behaviour; section 4 explains the detector architecture and the method used to construct the user behaviour; section 5 presents an empirical evaluation of our approach, and finally in the section 6 there are some concluding remarks, a comparison with related work, and the future work.

## 2 Related work

There are several works applying behavioural biometrics to user re-authentication, however most of them are based on the model of *active authentication* that consists in one time authentication usually in the beginning of the session where the user must follow various predefined steps. Examples of such approach can be found in [2 & 6]. In contrast other works are based on the model of *passive authentication* where users interact freely with the system and every determined period of time are re-authenticated without any predefined steps, hiding the re-authentication process.

The passive authentication can be implemented using numerous data sources such as: application/command usage [5], content in the title bar [3], keystroke dynamics [4], among others; nonetheless there have been just a few works which try to authenticate users based on data coming from the computer mouse, one of these works is described in [7], where the mouse movements are analyzed, extracting raw features such as: distance, angle and speed; and extracted features such as: mean, standard deviation and third moment of all the raw features; it was also considered the mouse wheels, no-client moves, and clicks; afterwards a supervised learning method (decision tree classifier) is applied to discriminate among  $k$  users, using a smoothing filter. Another work is presented in [1], here the mouse actions are classified in four sections: mouse movement, drag and drop, point and click, and silence; then for each category are extracted the mean and the standard deviation; next a statistical method is used to profile the users, combining the mouse dynamics with the keystroke dynamics.

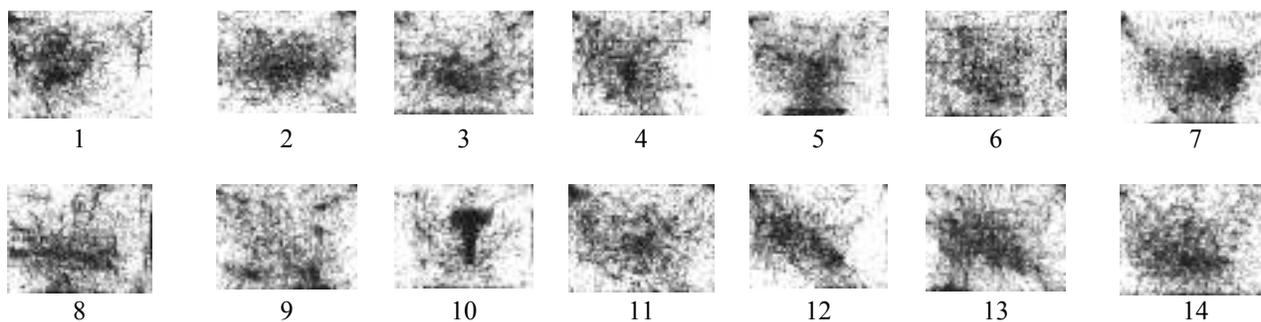


Fig. 1. Mouse data point distribution of each user

In this paper we are introducing a combination of the mouse biometrics and the usage of GUIs, applying it to intrusion detection systems using the application approach that provides detailed user behaviours.

### 3 Data analysis

#### 3.1 Mouse data collection

We collected data from 14 users, providing them of the same computer mouse and even the same mouse pad, with the aim of eliminating any kind of noise that could alter their behaviour; we also requested to the users to utilize Microsoft Windows and Internet Explorer for the time that takes to record 20,000 mouse points every 50 milliseconds with all the mouse events generated by the mouse wheels and mouse clicks, recording mouse points only when the users moved the mouse. We only collected data coming from one application just to make more evident the advantages of applying the application approach.

#### 3.2 Feature extraction

The mouse sensor extracts data such as: cursor positions, hit test codes, mouse wheels, click events and the time; when a click occurs it also stores the button pressed, whether is single or double and its duration.

We took the rough data obtained from the mouse sensor and then we applied a statistical analysis to acquire features that could identify users, after that we obtained the next categories:

*a) Mouse position:* as is shown in the figure 1 the distribution of the mouse data points is extremely different among all the users. In addition we could identify the permanency of these distributions through the user sessions. We extracted 48 features from this category, dividing the mouse data point distribution in a 6x8 grid as is shown in the fig. 2, afterwards is computed the percentage of the mouse data points per cell; from this we can indirectly measure the usage of GUIs, for example, where the user positions the windows on the screen or whether the user prefers to work with maximized windows.

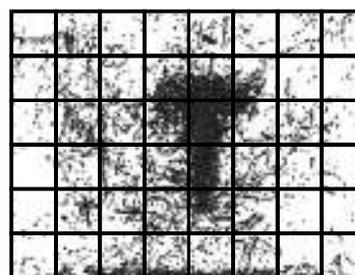


Fig. 2. Mouse point distribution divided by a 6x8 grid

*b) Hit test code percentages:* also called hit point codes, these features provide important information of the usage of GUIs. By considering the hit test codes we are directly measuring the usage of scroll bars, system menu, minimize, maximize, and restore actions.

*c) Mouse clicks:* we realized that click duration is different for each button (left and right), and it's also maintained for each user through different sessions; Now given the variety of the clicks duration we classified them in 5 groups of click duration for each button from 0 to 100, 100 to 200, 200 to 300, 300 to 400, and bigger than 400 ms. We also noticed that the duration of the mouse clicks varies depending on the activity performed by the user that is linked to the application.

*d) Movement distance:* we identified 4 groups of movement distances that are representative for each user, from 0 to 5, 5 to 20, 20 to 100, and bigger than 100 pixels; we also computed the movement distance mean. As the other features the movement distance is strongly linked to the application since the design of the GUIs could be very different.

*e) Wavering:* when we move the mouse we usually make curves or waves in our movements, because we normally don't move the mouse in straight line, from this we could extract the altitude of the curve (wave).

The above feature categories give as a result a set of 81 features which were distinctive for each user and permanent

through the user sessions. We also could notice that some users maintain certain features more than others.

In the later sections we will demonstrate the feasibility of the 81 features, making an empirical evaluation with data of 14 users.

### 3.3 Definition of the user behaviour

According to the past section 3.2, a dimensional vector can be extracted:

$$\Psi = \{\tau_1, \tau_2, \dots, \tau_{81}\} \quad (1)$$

Where:

- $\Psi$  is the feature vector.
- $\tau_i$  represents the feature  $i$  for  $1 \leq i \leq 81$  and  $\tau_i \in \mathcal{R} \mid 0 \leq \tau_i \leq 1$

However the majority of the features vary depending on the application as was shown above; consequently the user behaviour is defined as follows:

$$\Gamma = \{\psi_1, \psi_2, \dots, \psi_n\} \quad (2)$$

Where:

- $\Gamma$  is the user behaviour.
- $n$  represents the number of the existent applications in the operative system.
- $\psi_i$  is the user behaviour for a determined  $i$  application.

In the next sections we'll explain the construction and identification of this behaviour with the One-Class Statistic Classifier.

## 4 Detector architecture

The basic architecture of the detector is depicted in the Figure 3. Next will be described the functions of each component and their interactions such as: the feedback loop conformed by the behaviour update unit and the behaviour construction unit with the intervention of the database module.

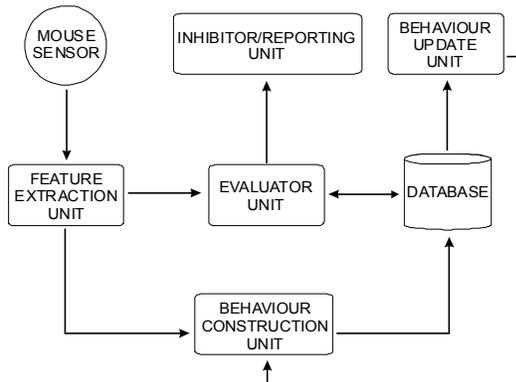


Fig. 3. Detector architecture

### 4.1 Mouse sensor

The mouse sensor obtains the following information from the user interaction: cursor positions, hit test codes, click events (double click, left, medium, and right click), mouse wheel events, the active application and the time. The mouse sensor participates in the *enrolment phase* as well as the *verification phase*. The main objective of the *enrolment phase* consists in the registration of the user in the system, and the *verification phase* consists in a continuous monitoring of the user's interaction via the computer mouse with the GUIs, in order to evaluate the collected data and emit a judgement of authorized or non-authorized user. The mouse sensor in both phases collects the data either to construct the user behaviour or to evaluate the collected data.

### 4.2 Feature extraction unit

As well as the mouse sensor, the feature extraction unit participates in both phases, the function of this unit is to extract from the raw data the 81 features previously identified in the data analysis. Basically it makes use of basic statistics such as the mean, standard deviation and percentages.

### 4.3 Behaviour construction unit

The behaviour of each user is unique and complex, and some of them have certain characteristics that make them quite identifiable; these characteristics need different number of mouse points in order to be identifiable; therefore is necessary to make a search taking into account different windows sizes (number of data points) to create the features vectors, with the intention of identifying users in the shortest time; once the different feature vectors were extracted from the data windows, a determined number of vectors is randomly selected for each application, with the objective of observing the users at different times of their interaction with the system, capturing variations in the behaviour. Once we have these vectors, a classifier can be trained in order to identify the user behaviour on every different application of the system. The selection of the window size is determined by evaluating the remaining feature vector of each application calculating its False Positive Rate (FPR) which is represented by the total number of false positives (valid feature vectors classified as invalids) divided by the total number of valid sessions; this rate give us the level of solidity of the determined behaviour. In this work the one-class machine learning approach is used, since it is only necessary to validate one user and everything outside of that user will be rejected as intruder. The selection of one-class machine learning corresponds to its advantages such as: less data collection, independent training, fast training and testing, and its decentralized management.

As was shown in the section 3.2 some features are more constant than others; according to this level of

constancy is assigned the degree of the significance of each feature; however the 81 features are not equally significant for all users, for example some users maintain a constant speed of the mouse movements and some users don't, then we need to determine what features are significant for each user, but taking into consideration the remaining features although with less value. The preceding idea suggested to design a one class classifier called One-Class Statistic Classifier (OCSC) which is capable of selecting the most significant features of the user, and to determine a confidence range for that features, assigning a penalty value depending on their level of significance with the objective of evaluating new cases and generating a penalty amount, that will be compared with a predefined threshold to determine whether it belongs to the user or not.

The OCSC has three phases: the training phase where the user behaviour is modelled; the evaluation phase where new cases are classified as valid users or intruders, and the third phase that consists in the threshold adjustment for the evaluation phase. In this section will be explained the first and third phase, which are the phases involved in the behaviour construction.

### 4.3.1 OCSC training phase

This phase receives as input a set of training vectors which will be used to model the user behaviour in one application, that will be represented by a set of ranges, one range per feature of the user behaviour; next it will be described in detail the training phase:

Input:

- The set of training vectors  $F = \{u_1, u_2, u_3, \dots, u_N\}$

Where:

- $u_i = \{u_{i1}, u_{i2}, u_{i3}, \dots, u_{iDim}\}$  where  $u_{ij} \in \mathcal{R} \mid 0 \leq u_{ij} \leq 1$
- $N$ : number of training vectors
- $Dim$ : number of features of the user behaviour

The objective of this phase is to compute the set of ranges  $R$  denoted as follows:

$$R = \{\text{range}_1, \text{range}_2, \dots, \text{range}_{Dim}\} \quad (3)$$

Where:

- $\text{range}$  is a 7-tuple  $(id, m, s, p, sf, inf, sup)$
- $id$ : Feature identifier
- $m$ : Mean of the feature
- $s$ : Standard deviation of the feature
- $p$ : Penalty (value assigned to qualify inputs)
- $sf$ : Spread factor (how spread are the values)
- $inf$ : Inferior limit of the confidence range
- $sup$ : Superior limit of the confidence range

Now to compute each range 4 steps have to be done for every  $k = 1, 2, \dots, Dim$ , one per each feature:

1. Computation of the mean and the standard deviation:

$$\text{range}.m_k = \text{Mean}\left(\sum_{i=1}^N u_{ik}\right) \quad \& \quad (4)$$

$$\text{range}.s_k = \text{StdDev}\left(\sum_{i=1}^N u_{ik}\right) \quad (5)$$

2. Computation of the spread factor: the spread factor represents how much the data is spread in each feature; this is computed in the following way

$$\text{if } (m_k=0) \text{ range}.sf_k=1000 \text{ else } \text{range}.sf_k = s_k/m_k \quad (6)$$

When  $m_k$  is equal to 0, the training vectors don't provide enough information to consider the feature; therefore it is assigned a high value of spread factor to give it the lowest penalty value.

3. Assignment of the penalties: the ranges are sorted by spread factor in ascendant order, in order to assign the penalty values in an easier way; subsequently the value of the penalty is assigned in this way:

$$\text{range}.p_k = \text{Dim} - k \quad (7)$$

By doing this, it is assured that features which are more constant get higher penalty values and features which are inconstant get smaller penalty values.

4. Computation of the confidence ranges:

$$\text{range}.inf_k = \text{range}.m_k - \text{range}.s_k \quad \& \quad (8)$$

$$\text{range}.sup_k = \text{range}.m_k + \text{range}.s_k \quad (9)$$

These limits provide a confidence range so as to facilitate the evaluation of new cases. As final step we sort the  $R$  set by feature identifier in ascendant order.

Output:  $R$  – Set of ranges

The above output contains the user behaviour as well as the confidence ranges for each feature giving the basis to evaluate new cases, as it will be explained in the evaluation unit.

### 4.3.2 OCSC threshold adjustment

The procedure to set the threshold can be separated in 3 parts:

1. The training cases are evaluated and the mean of their penalizations is computed obtaining the *TrainingMean*.
2. Then the testing cases that belong to the target user are evaluated and the mean of their penalizations is computed obtaining the *TestingMean*.
3. The *TrainingMean* don't provide enough error margin, since the ranges were computed using those training cases,

therefore we add the *TestingMean* to the *TrainingMean* to enlarge the error margin, giving as a result the *Threshold*.

The complexity of the OCSC in the training phase is  $Dim * N$  where  $Dim$  is the dimension of the feature vector and  $N$  is the number of training cases, and the complexity of the threshold adjustment phase is linear.

#### 4.4 Inhibitor/Reporting unit

This unit can be used to inhibit an intruder closing the session for several minutes or it also can be used to send a message to the security administrator.

#### 4.5 Evaluator unit

The evaluation phase has as main objective to classify new cases as belonging to the target user or as intruder, besides giving the resulting penalty amount of the evaluation, in order to facilitate the threshold adjustment in the third phase of the OCSC. In this unit the second phase of the OCSC is located.

The input of this phase is the set of ranges computed in the training phase, the new case to be classified, besides the threshold, which are denoted as follows:

Input:

- Set  $R = \{range_1, range_2, \dots, range_{Dim}\}$
- Testing vector  $v = \{v_1, v_2, v_3, \dots, v_{Dim}\}$   
where  $v_i \in \mathfrak{R} \mid 0 \leq v_i \leq 1$
- *Threshold*

Next it is verified whether the value of each feature is between the ranges; if it is, there is not any penalization, but if it isn't, there is a penalization based on its penalty value assigned and in how much it differs from the range, this is computed as follows:

*Penalty* =

$$\text{if } (v_k < \text{range.inf}_k) \sum_{k=1}^{Dim} \frac{(\text{range.inf}_k - v_k) * \text{range.p}_k}{(\text{range.sup}_k - \text{range.inf}_k)} \quad (10)$$

or

$$\text{if } (v_k > \text{range.sup}_k) \sum_{k=1}^{Dim} \frac{(v_k - \text{range.sup}_k) * \text{range.p}_k}{(\text{range.sup}_k - \text{range.inf}_k)} \quad (11)$$

$$\text{Now if } (Penalty > Threshold) \text{ then} \quad (12)$$

$$\text{Result} = \text{Intruder} \text{ else } \text{Result} = \text{Valid user}$$

Output: *Result* & *Penalty*

The *Penalty* basically represents how much the testing vector  $v$  differs from the behaviour of the target user, and the *Result* gives the verdict of the evaluation according to the *Threshold*. The complexity of the evaluation phase is linear.

#### 4.6 Behaviour update unit

Features obtained from a behavioural biometric of a person changes through the time; then it is essential to establish a feedback loop that adjusts the user behaviour.

The feedback loop is implemented using the record of penalties received of the feature vectors of the legitimate user; the proposed procedure is detailed as follows:

1. Store the training vectors utilized to generate the feature vector  $\psi$  for each application  $i$ .
2. Store a list of feature vectors extracted from the user interaction in the verification phase, let's call them "interaction vectors" and also store their corresponding penalty.
3. Every certain number of authentications, the mean of the penalizations of the interaction vectors should be computed.
4. Once computed the mean of the penalizations, the interaction vector that has the nearest penalty value to the mean is selected. Doing this the user behaviour is updated with a vector that represents a normal session of the user and it also presents a small deviation of the behaviour.
5. Then the oldest element of the training vectors should be deleted, afterwards the selected interaction vector is added and finally we repeat the construction of the user behaviour.

This process should be repeated for each application  $i$ .

There isn't any study of the durability of the behaviour of a user, however if the proposed procedure is utilized every determined number of authentications, given its characteristics, it assures to select the correct feature vector that will correctly update the user behaviour, no matter how often the behaviour is updated.

#### 4.7 Database

In the database is stored the user behaviour, the window sizes, the training vectors which conform the user behaviour, and a record of the user authentications with their corresponding penalties, for each application.

The mouse sensor was calibrated with the intention of capturing the necessary amount of data to compute detailed user behaviours in order to low the computational cost in the feature extraction unit, as we have already seen it only make use of basic statistics. The behaviour construction unit with the use of the OCSC selects the most significant features of the user behaviour, allowing a meticulous evaluation of the user behaviour in the evaluator unit that uses a penalty based evaluation. The behaviour update unit probably makes the most important function in order to reduce the anomaly detection rates and to extend the validity of the user behaviour, capturing the variation of the

Table 1. Users parameters for one-class statistic classifier experiment

User	1	2	3	4	5	6	7	8	9	10	11	12	13	14	AVG
Window size	2000	1000	2000	2000	1250	1250	1000	800	1250	1250	1250	2000	2000	2000	1503.57
Threshold	917	697	723	903	639	594	876	609	528	522	558	985	574	848	751.71
Time(min)	3.6	6.0	7.8	7.4	5.9	4.8	5.1	1.4	4.5	3.2	3.0	5.3	6.7	6.5	5.08

behaviour according to the adaptation or light changes in the environment.

## 5 Empirical Evaluation

### 5.1 Experiment design

With the purpose of identifying how much time or how many mouse points we need to re-authenticate users, we made several experiments with distinct window sizes (number of points), taking the 20,000 mouse points collected for the data analysis, dividing them in sets of 400, 500, 800, 1000, 1250 and 2000 data points to build the feature vectors; afterwards, we separated them in two groups: the training sets and the testing sets; each group is conformed from feature vectors of different times (beginning, middle and ending) of the user interaction with the purpose of building an accurate user behaviour, because when the users were in the collecting session we could notice certain fatigue in the users having as a result modifications in their behaviours. The intruder testing cases are conformed by data from the remaining users. As we are using a one class classifier, the training sets only consist of feature vectors of the target user; we set the number of training vectors to 10 avoiding the overtraining of the OCSC. The experiment design was based on the presented in [7].

### 5.2 Results

One of the objectives of the experiment was to determine what would be the smallest number of mouse points to re-authenticate each user. The table 1 shows the necessary points to re-authenticate the users with the minimum error and also shows the time taken by the users to generate the mouse points, this time was measured with the assumption that when users start using the pc make more mouse movements, as an intruder would do, therefore we took the beginning time of the data collection session of each user, even though this time could vary in different applications or situations. Since we are following a statistical method a larger window size will result in a smaller error of classification, but it would be useless for intrusion detection.

The figure 4 shows the error and the anomaly detection rates obtained per user, here we can notice the existence of two exceptional cases, the first case is the user #11 that has the largest false negative rate and the second case is the user #13 with a perfect classification of all the testing cases; we consider the existence of these exceptional cases is

completely normal, since there are users which have several distinguish characteristics in their behaviour, and others which don't; all the remaining users have similar values in their false acceptance/rejection rates.

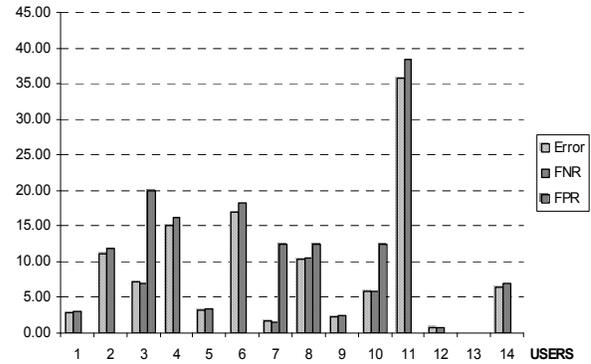


Fig. 4. Classification errors and anomaly detection rates obtained in the OCSC experiment

The global anomaly detection rates obtained were FPR 2.86% and FNR 10.13%; we have to mention that we adjusted the threshold automatically; therefore the obtained behaviour of the users is more accurate than the anomaly detection rates reflect. With this empirical evaluation we confirm that it is possible to authenticate users based on their mouse interaction and the usage of GUIs.

## 6 Conclusions

### 6.1 Related work comparison

We have already mentioned the work presented in [7] where the user behaviour is constructed using a 111 dimensional feature vector, applying a supervised learning method with a smoothing filter, reaching the following anomaly detection rates FPR 27.5% and FNR 3.06%; in contrast we used a 81 dimensional feature vector, using a one-class classifier without smoothing filter, reducing the computational cost and offering a decentralized management with less data collection, in addition we reach better anomaly detection rates with a similar average time of the user re-authentication.

In the work described in [1] a mixture of the mouse dynamics and keystroke dynamics is proposed, obtaining the following results FPR 1.31% and FNR 0.65%; On the side of the mouse dynamics they used a statistical method similar to ours, nonetheless the required amount of data to enrol the users is very large since their approach is based on a key oriented neural network, increasing the required time in the enrolment phase and the computational cost in the

training phase, resulting in an impractical registration of new users in the system, comparing it to our method that only needs 10 feature vectors per application to construct the user behaviour, with a very low computational cost, taking into consideration only information coming from the computer mouse.

It is important to mention that none of the works related to behavioural biometrics have studied the addition of a feedback loop that updates the user behaviour, in spite of its important function in order to reduce the anomaly detection rates and to extend the validity of the user behaviour, capturing the variation of the behaviour according to the adaptation or light changes in the environment.

## 6.2 Final remarks and future work

We have shown in an empirical way, the capability of authenticating users utilizing a mouse based behavioural biometric and the usage of GUIs, however our approach don't cover all the masquerading scenarios such as when the user avoids the use of the computer mouse, therefore with our study we pretend to complement the keystroke dynamics approach, that is also insufficient given the increasing usage of graphical user interfaces where mouse devices are commonly used.

Regarding the application approach it has been explained the behaviour changes presented by the users according to the application used and the effort of the activity realized, being a fundamental part in the construction of the user behaviour.

The direction of our future work will be focused to complement our re-authentication method with the keystroke dynamics to provide a complete solution for user authentication, once delimiting the responsibility to the current user, we would like to combine other kind of intrusion detection systems based on systems calls to detect a wide range of internal attacks, and also the combination of network traffic monitoring is necessary to create a robust intrusion detection system.

## 7 References

[1] Ahmed, A. A. E., and Traore, I. "Detecting computer intrusions using behavioral biometrics". Paper presented at the Proceedings of the Third Annual Conference on Privacy, Security and Trust, 2005.

[2] Everitt, R. A. J. and McOwan, P. W. "Java-Based Internet Biometric Authentication System". In the IEEE Transactions on Pattern analysis and machine intelligence, Vol. 25, No. 9, 2003.

[3] Goldring, T. "User profiling for intrusion detection in windows NT". Paper presented at the Proceedings of the 35th Symposium on the Interface, Computing Science and Statistics Volume 35 Security and Infrastructure Protection, 2003.

[4] Hockett, S., Ramel, J. Y., and Cardot, H. "User authentication by a study of human computer interactions". Paper presented at the Proceedings of the 8th Annual Meeting on Health, Science and Technology, 2004.

[5] Lane, T., and Brodley, C. E. "An Application of Machine Learning to Anomaly Detection". Paper presented at the Proceedings of the 20<sup>th</sup> National Information Systems Security Conference, 366-380, 1997.

[6] Monrose, F., and Rubin, A. D. "Keystroke Dynamics as a Biometric for Authentication". Future Generation Computer System (Elsevier) 16(4) ISSN: 0167-739X, 351-359, 2000.

[7] Pusara, M., and Brodley, C. E. "User re-authentication via mouse movements". Paper presented at the Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, 2004.