Enhancing Login Security Through the Use of Keystroke Input Dynamics

Kenneth Revett[1], Sérgio Tenreiro de Magalhães[2], and Henrique M.D. Santos[2]

[1] University of Westminster,
Harrow School of Computer Science, London, UK HA1 3TP
revettk@westminster.ac.uk
[2] Universidade do Minho,
Department of Information Systems, Campus de Azurem,
4800-058 Guimaraes, Portugal
{psmagalhaes, hsantos}@dsi.uminho.pt

Abstract. Security is a critical component of most computer systems – especially those used in E-commerce activities over the Internet. Global access to information makes security a critical design issue in these systems. Deployment of sophisticated hardware based authentication systems is prohibitive in all but the most sensitive installations. What is required is a reliable, hardware independent and efficient security system. In this paper, we propose an extension to a keystroke dynamics based security system. We provide evidence that completely software based systems based on keystroke input dynamics can be as effective as expensive and cumbersome hardware based systems. Our system is behavioral based that captures the typing patterns of a user and uses that information, in addition to standard login/password security to provide a system that is user-friendly and very effective at detecting imposters.

1 Introduction

With the increasing number of E-commerce based organizations adopting a stronger consumer-orientated philosophy, web-based services (E-commerce) must become more user-centric. As billions of dollars worth of business transactions occur on a daily basis, E-commerce based enterprises must ensure that users of their systems are satisfied with the security features in place. As a starting point, users must have confidence that their personal details are secure. Access to the user's personal details is usually restricted through the use of a login ID/password protection scheme. If this scheme is breached, then a user's details are generally open for inspection and possible misuse. Hardware (physiological) based systems are not yet feasible over the Internet because of cost factors and in addition, the question as to their ability to reduce intruder detection has not yet been answered equivocally. Our system is based on what has now become known as "keystroke dynamics" with the addition of keyboard partitioning [1,2]. We also consider in this study the affect of typing speed and the use of a rhythm when a user enters their login details. Keystroke dynamics

D. Zhang and A.K. Jain (Eds.): ICB 2006, LNCS 3832, pp. 661 – 667, 2005.

was first introduced in the early 1980s as a method for identifying the individuality of a given sequence of characters entered through a traditional computer keyboard. Researchers focused on the keystroke pattern, in terms of keyboard duration and keyboard latency [2,10]. Evidence from preliminary studies indicated that when two individuals entered the same login details, their typing patterns would be sufficiently unique as to provide a characteristic signature that could be used to differentiate one from the another. If one of the signatures could be definitively associated with a proper user, then any differences in typing patterns associated with that particular login ID/password must be the result of a fraudulent attempt to use those details. Thus, the notion of a software based biometric security enhancement system was born. Indeed, there are commercial systems such as BioPassword that have made use of this basic premise.

A critical issue with respect to enhancement of login based security systems is the criteria for success. There are two basic errors associated with biometric applications with respect to verification: false rejection (FRR -type I error) and false acceptance (FAR - type II error). One wishes to develop a system that minimises type II errors without increasing type I errors. In this paper, we employ the Crossover Error Rate (CER) as our measure of the balance between false acceptance ratio (FAR) and the false rejection ratio (FRR), as depicted in Figure 1. Striking the balance between sensitivity and specificity is a difficult balancing act. Traditional approaches have employed either machine-learning or deterministic algorithms. Among the solutions based on machine learning, the work presented by Chen [3] achieved a CER less than 1% and a 0% FAR. Ord and Furnell [4] also tested this technology, with a 14 people group, to study the viability of applying this technology on PINs (Personal Identification Numbers) typed on a numeric-pad. Although the results were initially promising, it was found that the results did not scale up well and the authors indicated that this technology was not feasible for community based applicability. Deterministic algorithms have been applied to keystroke dynamics since the late 70's. In 1980 Gaines [5] presented the results of a study of the typing patterns of seven professional typists. The typists were asked to enter a specified text (3 paragraphs) repeatedly over a period of several months. The authors collected data in the form of keystroke latencies from which they constructed digraphs were constructed and analysed statistically. Unfortunately, no real conclusion could be drawn from this study regarding the uniqueness of each typist's style – most likely resulting from the small sample size and/or inadequate data sample. The method used to establish a keystroke pattern was a breakthrough, which introduced the concept of a digraph, the time spent to type the same two letters (digraph), when together in the text. Since then, many algorithms based on Algebra and on probability and statistics have been presented. Joyce & Gupta presented in 1990 [6] an algorithm to calculate a metric that represents the distance between acquired keystroke latency times over time, thus introducing a dynamic approach. In 1997 Monrose and Rubin employed an Euclidean Distance and probabilistic method based on the assumption that the latency times for one-digraph exhibits a Normal Distribution [7]. Later, in 2000, the same authors presented an algorithm for identification, based on the similarity models of Bayes [8], and in 2001 they presented an algorithm that employed polynomials and vector spaces to generate complex passwords from a simple one, using keystroke patterns [9].

In our research, we examine various typing characteristics that might provide subtle but consistent signatures that we can use for keystroke verification purposes. Our initial study was designed to provide a baseline case for the CER from a group of informed users that were asked to participate in this study. Once we established a baseline CER< we then wished to determine if there were factors related to typing styles that could alter the CER. We selected two basic factors: length of the passphrase and typing speed. In the next section we describe in detail the algorithms deployed in this study, followed by a Results section, and lastly a brief discussion of this work.

2 Implementation

Our primary goal is to produce a software-based system that is capable of performing automated user ID/password verification. We employ the following steps when a new user is added to the system (or is required to change their login details):

1. The login ID/password or simply the new password is entered a certain number of times (enrollment).
2. A profile summarising the keystroke dynamics of the input is generated and stored for access to the verification component.
3. A verification procedure is invoked which compares stored biometric attributes to those associated with a given login ID/password entry after the enrollment process.

The enrollment process, made by the user once on the first use of the service, consists on typing the users usual password, or passphrase, twelve times. If the user mistyped the passphrase, they were prompted to continue entering until all twelve entries were entered. During the enrollment procedure, statistics were calculated and stored for the verification process. Specifically, our algorithm calculates and stores the average, median, standard deviation, and the coefficient of variation for the latency times for each digraph (13 in all) and the total time spent entering each passphrase. Our enrollment phase was based on a series of 14 character passphrases entered into our system by a group of 8 volunteers, all of whom were fully aware of the purpose of this study and all reasonably computer literate. Each volunteer was requested to input a passphrase a minimum of twelve times in order to generate the statistics required for the verification phase. In addition, each volunteer served as their own control for FRR rates by entering their respective passphrases for an additional period of four weeks after the start of the study (yielding an average of 10,000 entries for FRR determination). The stored data table for the enrollment statistics was updated over time, with the oldest entry replaced by the most recent enrollment episode.

For our verification stage, we recruited a group of 43 volunteers (34 through the internet version of this software) and 9 users via a laptop running our software. All participants in the verification phase of this study (including the volunteer group) were required to enter at least 16 entries per user. For the volunteers (enrollment and verification), this provided use with the means to calculate the FRR (the first 12 entries were for enrollment and the rest for verification) and also for FAR on

passphrases entered by other volunteers. All verification participants (43) only participated in determination of the FAR of the system. In total, we had over 187,000 login attempts in the baseline determination phase, with less than 0.01% successful attacks.

To allow a comparison of our FAR/FRR values with existing published results [6], we used a threshold of 60% for the time latencies for a positive match between a verification request and stored data for that passphrase. When a verification entry was input into our system, we used the following measure to determine if the digraph latency time was appropriate for a given passphrase. For each pair of keystrokes (digraphs) the algorithm will measure the time latency, defined as *TLP*, and compare it with the one stored.

$$Lowest(Average; median)*\left(0{,}95-\frac{SDesviation}{Average}\right)\le TLP$$

$$TLP\le Higher(Average; median)*\left(1{,}05+\frac{SDesviation}{Average}\right)$$

Equation 1. Crtieria for acceptance of a given input for the digraph latency

The comparison result will be a hit if and only if this criterion has been met. A total of 13 digraphs exist for each 14-character passphrase, and the results for each digraph are stored in a temporary Boolean array. A '1' is placed in the table if the LTP is within the specified boundary conditions and is the first occurrence of a '1' in the passphrase (always true for the first correctly entered character). Subsequent correctly input keystrokes would result in a '1' being replaced by a '1.5' for that digraph entry in the array. If the keystrokes did not result in a hit, then a '0' is entered for that digraph position in the array. Then the elements in the array for a particular passphrase are added together. If the sum is greater than a given threshold, then the entry is considered valid, otherwise it is invalid. For instance, if the threshold is set on 70%, users will only be authenticated to the system if the value A obtained from a given attempt is over 70% of the highest possible value, which is given by: $(number\_of\_characters-1)*1.5+1$.

Finally, if and only if the login attempt is accepted, the oldest values stored for the latencies are substituted by the corresponding values collected in this successful attempt. This last procedure will allow the data stored to evolve with the user. This allows the system to evolve over time, as the user's familiarity with their passphrase improves with time and practise, so will the statistics. The system administrator can change the sensitivity of the system at will. For instance, to maintain a 60% threshold, all users must generate a score given by (number_of_characters –1)*1.5 +1 that is over 60% of the maximum score. For a 14-character passphrase, this would yield a score of 12.3, which would be set to 12, since our threshold is a multiple of 1/2. Thus any score greater than 12 would be considered a legitimate entry into the system. By varying this threshold, we can extract an estimate of the FAR/FRR as a function of the sensitivity threshold. What we wish to produce is a system that yields a very low FAR without incurring a large FRR in the process. A reasonable criterion

is when the FAR/FRR intersect – the Crossover Error Rate What we wish to do is reduce the CER to the lowest possible value without placing an undue burden on the user community. We have explored two basic techniques in a previous work [11], focusing on keyboard partitioning and the typing speed. We present the results of an extended study on these factors, which we present in the following Results section.

3 Results

This algorithm presented a CER of 5,58% and it can achieve, at the lowest thresholds, a FRR of near zero that maximizes the comfort of the user. At the higher demanding thresholds the algorithm presents a near zero FAR, maximizing the security. The results of our baseline experiment, with a single 14-character passphrase are presented in Figure 1 below. The results presented in Figure 1 can be summarized by the CER – which was 5.58%. It is important to notice that the results obtained in this experiment are the worst case scenario, when a passphrase breech has occurred. If the passphrase was not disclosed, then we could extrapolate the FAR (considering a brute force attack) by:

$$FAR_{Brute\_force} = (1/(Number\_of\_possible\_passphrases)) * FAR_{Know\_passphrase} \quad (2)$$

Equation 2 states that if the passphrase were not known, then the FAR would be equivalent to the FAR when the passphrase was known, multiplied by the probability of guessing the passphrase. With a 14-character passphrase, the success rate of a brute force is near astronomical.

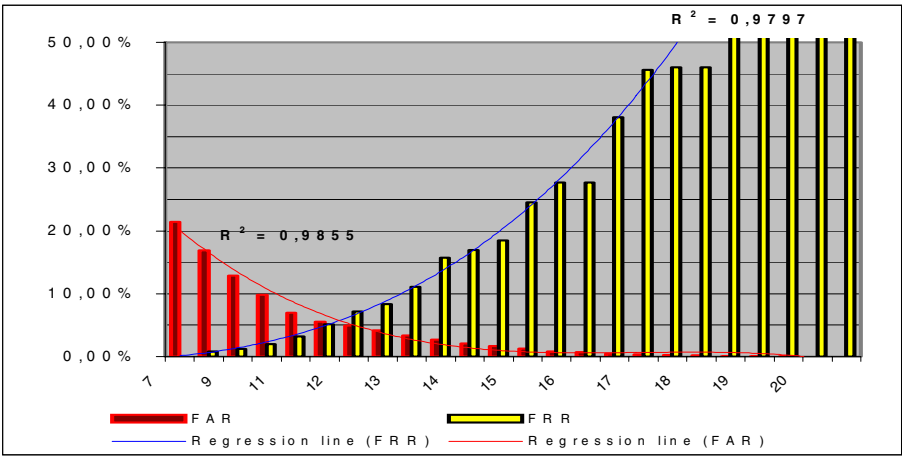

Fig. 1. False Acceptance Rates and False Rejection Rates for the range of possible thresholds for a 14 character passphrase. The x-axis is the threshold according to equation 1 and the y-axis is the resulting FAR/FRR. The data was generated from over 10,000 entries of the same passphrase.

3.1 Additional Experiments

We wanted to determine whether we could improve on our based CER of 5.58%. We investigated the length of the passphrase to see if it has an influence on the CER value. Our previous work [2] along with the work presented in this paper so far utilised long passphrases (14 characters). Generally, most IDs/passwords, PINs etc. are much shorter – on average between 4-8 characters in length. We therefore investigated a series of 7 character passphrases selected randomly by a computer programme. We enlisted a group of 10 volunteers to participate in this study. The results indicate that the FAR/FRR was reduced to approximately 2% (see Figure 2). We also incorporated keyboard gridding into our performance criteria by weighting characters that are in contiguous keyboard partition more heavily than those that were within the same partition or in non-contiguous partitions (by a factor of 2). The results indicate that the CER could be reduced to less then 0.01% using a combination of 14-character passphrase and keyboard partitioning.(data not shown).

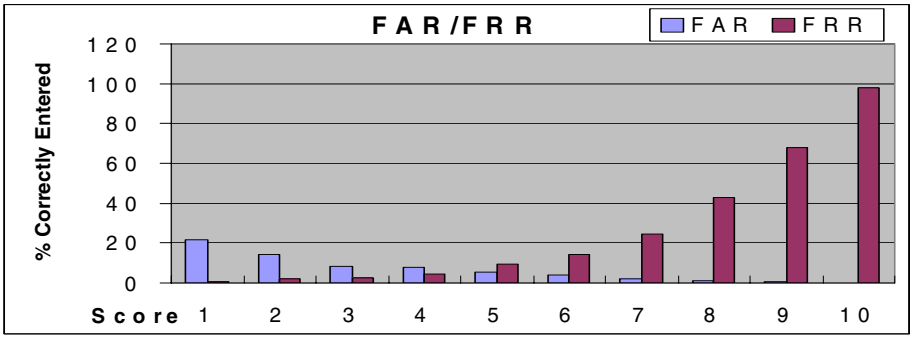

Fig. 2. FAR/FRR for the study using a 7 character passphrase. The CER (not on this display) was 4.1%. These results were obtained through 10 volunteers, entering a specific passphrase of 6 characters for a total of 1,000 trials (10 users).

4 Conclusions

This study provides supporting evidence to the role software based security systems can bring to the issue of enhanced computer security. Our system, based on keystroke dynamics, is not overly burdensome to the user, very cost-effective, and very efficient in terms of the overhead placed on an internet based server. We achieve a very low FAR/FRR (each less than 5%), compatible with those produced by very expensive hardware based systems. In addition, we have begun investigating additional strategies that can be combined with keystroke hardening, such as keyboard partitioning. Partitioning provides an added layer of security, but requires users to limit their selection of login IDs and passwords.

Our system incorporates the evolving typing styles of individuals. This is an important property of any software based biometric system. Users may experience through

personal development, variations in their typing styles and/or speed. For instance, when a user is forced to change their password, they will take time to adjust to it, which will certainly have an impact on their typing signature. Any system that fails to take this into account will yield an undue burden on the user if it is not capable of dynamically adjusting the required acceptance thresholds.

References

[1] Yan, J., Blackwell, A.F., Anderson, R. & Grant, A. , 2004, Password memorability and security: Empirical results, *IEEE Security and Privacy* **2**(5), 25-31.

[2] Magalhães, S. T. and Santos, H. D., 2005, An improved statistical keystroke dynamics algorithm, *Proceedings of the IADIS MCCSIS 2005*.

[3] Chen, Z., 2000. *Java Card Technology for Smart Cards.* Addison Wesley, U.S.A.

[4] Ord, T. and Furnell, S. M., 2000. User authentication for keypad-based devices using keystroke analysis. *Proceedings of the Second International Network Conference – INC 2000*. Plymouth, U.K.

[5] Gaines, R. et al, 1980. Authentication by keystroke timing: Some preliminary results. Rand Report R-256-NSF. Rand

[6] Joyce, R. and Gupta, G., 1990. Identity authorization based on keystroke latencies. *Communications of the ACM*. Vol. 33(2), pp 168-176.

[7] Monrose, F. et al, 2001. Password Hardening based on Keystroke Dynamics. *International Journal of Information Security*.

[8] Monrose, F. and Rubin, A. D., 1997. Authentication via Keystroke Dynamics. *Proceedings of the Fourth ACM Conference on Computer and Communication Security*. Zurich, Switzerland.

[9] Monrose, F. and Rubin, A. D., 2000. Keystroke Dynamics as a Biometric for Authentication. *Future Generation Computing Systems (FGCS) Journal: Security on the Web*.

[10] Alen Peacock, Xian Ke, Matthew Wilkerson. "Typing Patterns: A Key to User Identification, IEEE. *Security and Privacy*, vol. 02, no. 5, pp. 40-47, September-October, 2004

[11] Revett, K. and Khan, A., 2005, Enhancing login security using keystroke hardening and keyboard gridding, *Proceedings of the IADIS MCCSIS 2005*.