Behavioral biometrics for DARPA's active authentication program

Ingo Deutschmann, Johan Lindholm

Research & Development
Behaviometrics AB
Aurorum Science Park 8
SE-977 75 Lulea
ideutschmann@behaviosec.com, jlindholm@behaviosec.com

Abstract: The aim of the US Defense Advance Research Project Agency`s (DARPA) Active Authentication program is the continuous authentication of users by using behavioral biometrics authentication systems, which does not depend on specific hardware or sensors. This paper presents how such a continuous authentication system would perform in an office like environment. The analysis is performed on a data set captured from 99 users over a 10 week period. Our continuous authentication system builds a behavior biometric profile of the user by observing mouse movement, keystrokes and application usage. The user is then actively matched against his profile. The goal was, as DARPA is mentioning in their Active Authentication program, "This means the system would, potentially have to falsely reject the user more than five times in a row during continuous usage over a 40 hour period to fail to meet this target. The technologies developed under this solicitation should be able to work invisibly to the user unless five false positives are reached". The results of our study indicate that the correct user can work through a regular workday without being falsely rejected, while the incorrect user would be detected within 18 seconds using keyboard or 2.4 minutes using mouse. Application process usage results show that the incorrect user would be detected in just over 1.5 minutes.

1 Introduction

Behaviometrics, or behavioral biometrics, is the identification of humans by their measurable behavior. It focuses on behavioral patterns rather than physical attributes. Behavioral biometrics utilizes the characteristics of the user's keystoke, mouse input and application usage to create virtual fingerprints of their behavior. It can efficiently prevent intrusions on laptops or workstations by continuously verifying that it is the authorized user that is accessing the computer. The user is monitored during the whole working session to create an ongoing authentication process. When using behavioral biometrics rather than static biometrics such as fingerprint, the normal biometric approach is not ideal. One of the core strengths of behavioral biometrics is that it can continuously improve and adapt to the user. The reason that behavioral authentication is so beneficial is, that it is very non-intrusive and provides continuous protection independent of what application is being used. In this study we are using passive attacks to calculate the accuracy. This mean that we are using different users normal behavior and cross compare them to determine the performance of the system.

3 General Continuous Authentication System

An authentication system generally consists of four major parts:

- A monitor, which collects relevant behavioral data from the underlying system. In our research we gathered mouse, keyboard data and application usage.

- A classifier that sorting and filtering the data into different categories.

- A user profile which is storing condensed timings and is continuously updated when the input is assumed to be from the correct user. The system has to perform a initial training when the behavioral profile is empty. During this phase our system will assume that it is the correct user that is interacting with the system.

- An evaluator that continuously verifying the filtered data. The evaluator uses the stored profile to run its tests on the data from the classifiers to calculate a similarity score, which is used to verify the integrity of the user.

4 Calculating FAR/FRR/EER

To calculate the FAR/FRR/EER of the behavioral system, a test group of 99 users were selected. They worked in DoD-like environment and their behavior was captured for three months by a monitor installed on every machine. We monitored applications as well as keyboard and mouse interactions.

4.1 Interactions

In order to assess the likelihood that the originally authenticated user is still in control of the keyboard, the collection of the various independent interactions from typing (press, flight, sequence), mouse movement (movement, click, drag and drop) and application usage are combined into a single similarity score.

4.1.1 Keyboard interactions

Keystroke patterns are collected by the way users type at the keyboard. We count one keystroke as one interaction.

4.1.2 Mouse interactions

Mouse dynamics can be divided into different categories of actions, such as: main action types: Mouse move, click, drag and drop and action characteristics: Angle, velocity, direct distance.

4.1.3 Application interactions

Keyboard or mouse events are tied to the application that is active when they are performed. The system uses application specific profiles to enhance performance and accuracy. We analyzed information such as time open, apps open in parallel, how they are opened and closed, number of open views, memory and CPU consumption. We tested our data without taking into account holidays and daily fluctuations of use.

4.2 Training

A profile starts empty and the system has to train it to learn the behavior of the user. At the beginning an early stage it is difficult to differentiate between different users, so our system initially assumes that it is the correct user handling the computer. As users evolve their behavior over time the profile is continuously updated when the system is sure, that the right user is using it.

5 Trust Model

The trust model is intended to enhance or replace such simple detection mechanisms as score/threshold models, where a score is matched against a threshold for verification. One issue with the score/threshold model is that small deviations in behavior may cause false rejects; i.e. the user's similarity score dips below the current threshold and is considered to be incorrect. The idea behind a trust model is, that the correct user will generate better scores over time than the incorrect user, thus smoothing out small mistakes by the correct user, while still detecting the incorrect user. The trust of a user is a value calculated from a set of evaluated scores, which could be the similarity of the current user's behavior against the stored profile, as well as the system current state. The system triggers detection if the trust reaches a certain threshold.

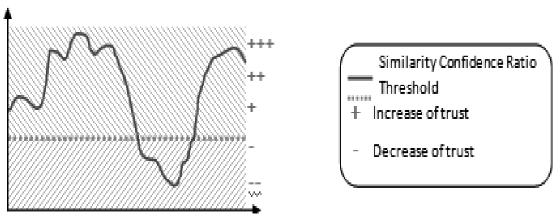

Figure 5.1: Simplified Score-Trust mapping example

The similarity score is mapped against the trust model by using a trust-threshold. If the user is above the threshold the trust will increase, and if the user is below the threshold, the trust will decrease. Staying above the threshold improves the trust level to its maximum; the higher the user is above the threshold, the quicker the trust reaches its maximum.

The detection speed between users may vary because of:

1 How much the behavior differs between the active user and the profile. Larger difference gives faster detection time.

2 How much trust the previous user has achieved. If the previous user has worked him up to be fully trusted, it will take longer time for the incorrect user to reach the not trusted level. Note that the trust will decay over time; so that a user that leaves his computer will have his trust set back to the start value (see 5.1).

Abnormal actions can trigger a decrease in trust, such as using key combinations that have never been used before or repeatedly providing immeasurable samples (compare to pushing your forehead against a fingerprint scanner).

5.1 BehavioSec model

The trust in the BehavioSec model is represented as value between 0 and 100, where higher value means higher trust. The initial trust value is configurable and sets how aggressive the system should be from start. Trust is updated using equation 5.1, with input from the different continuous tests (keyboard, mouse, application usage etc.). A detection is made when the trust decreases below the trust threshold.

$$C := \begin{cases} \text{startup value} \\ Max\left(\left(C - \frac{\frac{T-P}{T}}{Z\prime}\right), 0\right), P < T \\ Min\left(\left(C + \left(\frac{\frac{P-T}{1-T}}{Z\prime}\right)\right), 100\right), P \geq T \end{cases} \qquad (5.1)$$

C = Trust of the user

T = Similarity score threshold between trusting and not trusting the user based on one single test

P = Similarity Score from the last test of the users input against the template

Z = Constant which controls how much to increase or decrease the trust on each test

Z'= Z/100

$$\frac{\frac{P-T}{1-T}}{Z'} \quad (5.2)$$

Equation (5.2) calculates the increase in C which is proportional to the difference between |P-T| by a variable factor Z. The decrease function works in the same way as the increase function, when P is strictly less than T. The Z variable can be set independently for the increase (5.2) and decrease functions. The value of Z sets the slope of the curve for how much a similarity score alters the trust C. The model increases the trust C if the score P is equal or greater than the threshold T. The trust level can never increase beyond 100. Another type of trust model, by Patrick Bours, can be found in [Ha01].

5.2 A practical example of trust

The trust model can be compared against a score-threshold model to display the differences between the two. We have used the mentioned dataset of behavior on two different models of detection, score-threshold and trust, to create a comparison between the two. Note that the trust models parameters in these tests are not tweaked to be the best possible, one could for example allow more false rejects to detect the incorrect user faster. We have selected users that are quite similar in score on purpose, to demonstrate how the trust model can take small variances in score into account. The data from figure 5.2 and figure 5.3 are taken from real user input, where the upper diagram is the similarity score and the lower is the trust, calculated from the score. The correct user, overtime, generates a much better trust value than the incorrect user. Note that the trust value is reset after each detection (value goes to zero), that is why the incorrect user trust value seems to oscillate. One can see why a score-threshold has the drawback of more false rejects. Connecting the score to our trust model, would produce only one false rejection for the right user. While it takes a bit longer to detect an intruder, we believe it is within reasonable time. Also note that these values and graphs do not reflect the accuracy of the solutions, they are merely here to show the correlation between score and trust, and how a score-threshold system is different from a trust model system.

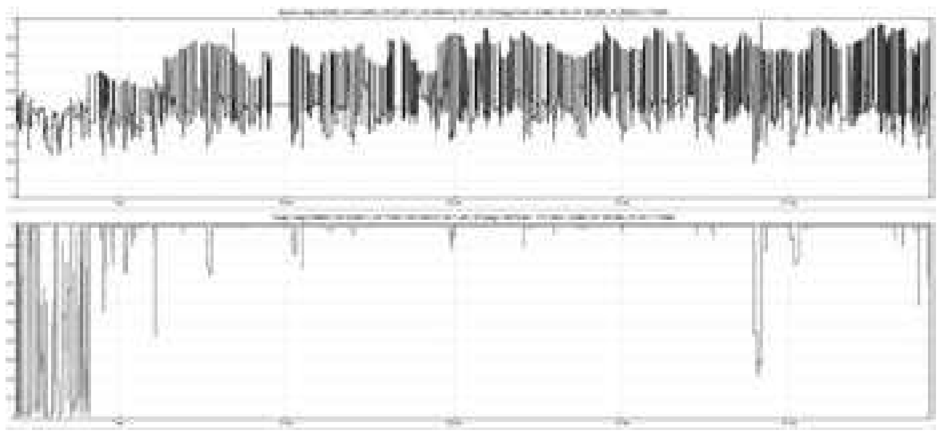

Figure 5.2: Correct user score and trust for three months

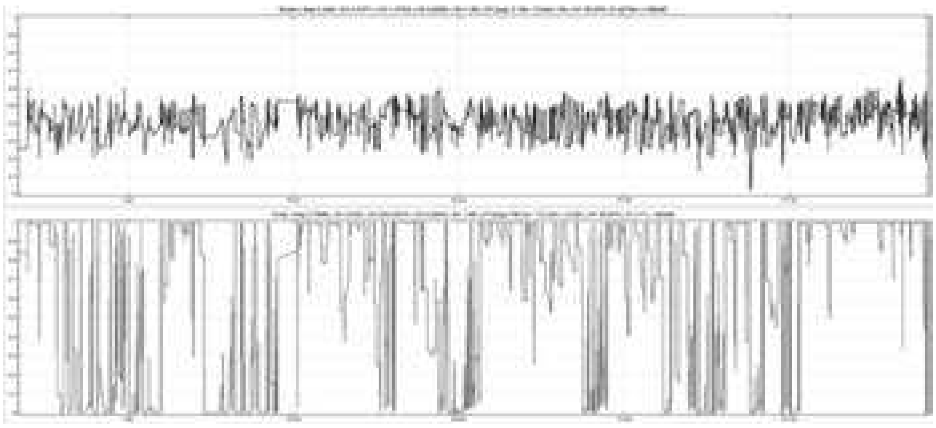
Figure 5.3: Incorrect user score and trust for three months

6 Methodology

6.1 The test bed

The test bed was consisting of 99 users, which have been using the computer for regular work for at least 20 hours per week, for a total of 10 weeks on standard Windows 7 desktops with office suite, web browser, antivirus etc. The data was collected by software, which tracked various events that could be interesting regarding their behavior. The aim was that the data collector should be as invisible as possible for the test subjects, even though the users know it is collecting the data, they should be carefree so that they can keep working as usual without any attention. The data had been collected bi-weekly. The users where asked not to share their working space with other people to avoid a “contamination” of the data. The data collection could be stopped by the users at any time, if they shared their working space or wanted to obtain privacy.

6.2 Analysis methodology

The collected behaviors (behavioral data files) had been fed into the system, which built behavioral profiles that were used to compare against themselves and others (posing as passive attackers). The scores for the correct user where generated as the behavioral profile built up, meaning that the results reflects how the system would work in a continuous setting. As the attacking simulations took a long time, we had to restrict the amount of attackers. To calculate how long an attacker could use a user’s workstation, five different users where randomly selected to act as passive attackers (comparing the attacking users’ normal behavior). The statistical data, which can be viewed as a list of raw scores for each comparison, was then analyzed in our FAR/FRR tool that calculated thresholds, active time, training times etc. If a parameter adjustment allowed the correct user to use the computer for longer time, without significantly increasing the detection time for the incorrect user, that setting was be picked instead.

7 Performance Metrics

We have proposed a model for determining the performance of a continuous authentication system by using the metric of the total number of detections after a given amount of interactions. Detections are places in the interaction stream, where the system decides that the latest subset of interactions is not a behavioral match with the profile. It is both hard and unpractical, to use time as metric for determining the performance of a continuous authentication system. We instead propose the use of number of interactions as the metrical time factor.

7.1 Proposed usage of model

The model we propose can be used on an offline version of the system, where one build profiles of users from stored logs, simulating the actual continuous authentication system, to compare users against themselves and others. We have U users with a log-file L associated with each user, where the log file contains all the users' interactions over a period of time. The simulation builds a profile P for each user; using a subset of L (we took the fist 40% of the log). A research of the needed training times will be discussed in a later paper. The rest of the log file was used to test the user against himself, as well as updating the profile. The system notes how many detections (if any) are made. The next step was to test each profile against the other logs, also noting how many detections that are made. To test how good a system performes, we measured the amount of interactions for each input that is required to catch an intruder on average, while not rejecting the right user. Based on the results, a global optimal threshold was calculated.

8 Results

The detection times that are presented in table 8.1 shows the median for each modality. We differentiated between 'active' time and wall clock time. As a user is not pushing buttons on the keyboard every second of the day, nor moving the mouse, we counted only the active time. We chose a timeout period of 10 seconds. So after this timeout period the active time counter stopped counting. The table 8.1 shows the number of interactions the correct user can perform before rejected, and an incorrect user can perform before getting detected and how many interactions the users where performing during a typical work day for each modality. Please note, that the interactions are counted only in the 'active' time window.

Modality	Typical day *	Incorrect user	Detection times ***	Correct User	False rejection times ***
		Interactions	Minutes	Interactions	Minutes
Keyboard**	400-1300	15-50	0.3	10000-32000	138
Mouse	743	59	2.4	1596	66
Application	7690	67	1.5	17181	384

Table 8.1: Performance Results for continuous usage

* The amount of interactions for the modality that the users on our test bed performed during a regular work day.

** for keyboard one interaction is one keystroke (typing "test" counts for 4 interactions)

*** The time is calculated from continuous active usage (no pauses included).

9 Discussion & Conclusions

In this paper we have explored the performance of a continuous authentication system, as a second factor for a DoD personnel. The correct user can work through at least a regular workday without being falsely rejected, the incorrect user would be detected within 18 seconds using keyboard (15-50 keystrokes) or 2.4 minutes using mouse (66 interactions). Application process usage results show that the incorrect user would be detected in 1.5 minutes. What is missing in our study is the test with real attackers. We think that an attacker would switch much quicker between applications as well as starting a lot of applications after another, so would be more easy to recognize. A further study will target a continuous authentication system on mobile devices under DARPA-BAA 13-16.

References

[HA01] Hafez Barghouthi Patrick Bours. Continuous authentication using biometric keystroke dynamics. Number 9788251924924/ in Vitenskaplig publisering, pages 1–12, Nardoveien 12, 7005 Trondheim, November 2009. Norwegian Information Security Laboratories (NISlab), Tapir Akademisk Forlagbraham

[PH01] P.J. Phillips, A. Martin, C.L. Wilson, and M. Przybocki. An introduction evaluating biometric systems. Computer, 33(2):56 –63, February 2000

[CS01] Crowd Sourced. Biometrics

[VA01] V.S Valencia. Biometric testings "its not easy as you think", biometrics consortium conference. Arlington VA USA, September 2003