



Continuous Authentication Using Behavioral Biometrics

Ingo Deutschmann, Peder Nordström, and Linus Nilsson, *BehavioSec, Sweden*

A continuous behaviometric authentication system is tested on 99 users over 10 weeks, focusing on keystroke dynamics, mouse movements, application usage, and the system footprint. Keystroke dynamics proved most appropriate for continuous behaviometric authentication.

Behaviometrics refers to measurable behavior used to recognize or verify a person's identity. Unlike biometrics, it focuses on behavioral patterns rather than physical attributes such as fingerprints and voice patterns. For example, during World War II, soldiers used behaviometrics to identify Morse-code radio operators—in particular, they used a methodology called “the fist of the sender.” Skilled listeners could identify an operator's transmission style, thus recognizing enemy troops.¹

Behaviometrics offers a way to enhance system security after a user is verified with traditional security techniques, such as a login password. However, aside from work done by Kevin Killourhy and Roy Masion,² only small sets of data are available to test behaviometric algorithms. Furthermore, although Charles Tappert and his colleagues have captured keystroke dynamics on long input texts,³ and John

Monaco and his colleagues developed a system for continuous-authentication of the same,⁴ no researchers have yet continuously monitored a user's behaviometrics during an entire working session.

Such a continuous-authentication system would need a set of behavioral traits to calculate a similarity ratio (score) between the user's current and expected behavior. If this score were to drop below a certain threshold, the system could send an alert about the user being a possible intruder. Here, we review a research project we conducted to investigate the possibility of authenticating users continuously on desktop computers.

Gathering Data

The behaviometric inputs we used were keystroke dynamics, mouse movements (together with display resolution), CPU and RAM used, and processes and applications (that is, the programs used most

often). We gathered these behaviometrics from 99 people for 20 hours per week for 10 weeks. The workstations ran Microsoft Windows operating systems with a software monitor installed to gather data and upload results daily. The behavior was anonymized, and application names were hashed to address user privacy concerns, so the user and used applications couldn't be inferred.

Our first task was to organize the gathered data. We defined *active time* as the amount of time the user was constantly using the computer. After 10 seconds of inactivity, we no longer considered the user to be active. The collected data contained 2,302 hours of active time, which corresponds to 13.52 weeks of data, with 2.8 million *interactions*—that is, an event such as a mouse move, key press, or key flight. For example, the number of interactions for writing the word “test” is seven, which is four presses and three flights.

We used the collected data to simulate attacks (through cross comparisons) equivalent to 92,577 hours of active time, which corresponds to 10.57 years with 3,906 comparisons totaling 120 million interactions.

Developing a New Metric

The basis for introducing a new metric for continuous authentication was influenced by previous research from Patrick Bours and his colleagues.⁵ A new metric was needed, because one of the main differences between one-time and continuous authentication is the extra time dimension. User recognition must occur in a moving time window, while the computer is continuously used, so traditional methods (such as false acceptance rate, false rejection rate, and equal error rate⁶) aren't suitable for authentication.

So, we first calculated a similarity score to determine how well the measured behavior matched the expected behavior, and we mapped it against a trust model using a threshold. If the score is above the threshold, the trust level increases; if the score is below the threshold, the trust level decreases. Staying above the threshold improves the trust level; the higher the score, the quicker the trust reaches its maximum level.

The BehavioSec Trust Model

In the BehavioSec model we developed, trust is represented as a value between 0 and 100, where the higher numbers indicate a higher level of trust. Our starting condition is 100, which means we trust the user by default. We then update the trust level

using the following formula, with input from the different continuous tests (keyboard, mouse, and so on) and take action depending on the trust level:

$$C := \begin{cases} 100, & \text{at startup} \\ \text{Max } C - \frac{T - P}{100 * \frac{T}{Z}}, & 0, P < T \\ \text{Min } C + \frac{P - T}{100 * \frac{1 - T}{Z}}, & 100, P \geq T \end{cases}$$

In this formula, C is the trust of the user; T is the threshold between trusting and not trusting the user, based on one single test; P is the score (that is, the probability from the last test of the user's input against the template); and Z is the constant, which controls how much to increase or decrease the trust on each test.

We calculate the increase in C , which is proportional to the difference between $|P - T|$ by a variable factor Z , using $(P - T)/100 * ((1 - T)/Z)$. 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 and decrease functions. The value of Z sets the slope of the curve for how much a specific probability alters the trust C .

The model increases the trust C if the probability P is equal or greater than the threshold T . The trust level can never increase beyond 100.

Data Analysis

We analyzed the data using the following five steps.

Collect data from the testbed. We collected the data over a 10 week period, during which time a set of users worked at their workstation in a normal fashion. The only constraint was that they couldn't let anyone else use their workstation. The data was regularly uploaded to a central server. The collector software was built to be as nonintrusive as possible.

Analyze and find patterns. We explored the data, visualizing the results and trying to find repeatable usage patterns on an individual level. We narrowed down the data, focusing on approaches that were the least erratic and seemed to more consistently identify the user.

Table 1. Mouse and keyboard results. The detection times don't include inactivity periods of more than 10 seconds.

	User (simulation hours)	Detection time (False reject/true positive)	Interactions* (Before detection/ rejection)
Mouse	Correct (1,276)	823.74 minutes (13.73 hours)	17,470
	Incorrect (4,772)	4.08 minutes	88
Keyboard	Correct (450h)	Never falsely rejected (627.49 minutes)**	Infinite (16,930)**
	Incorrect (2,716)	5.25 minutes (1.16 minutes)**	174 (38)**
Combined	Correct (1,726)	828.72 minutes (18.37h)	25,028
	Incorrect (7,489)	4.44 minutes	114

* Interactions for keyboard are up/down/flight (typing "test" counts for seven interactions)

** Calculation networks specifically tweaked for keyboard rather than general

Identify differences. We compared the patterns we found among the set of users to find ways to distinguish between the users. This further narrowed the list of usable approaches.

Train the system. Our approach consists of type 2 fuzzy sets building a fuzzy net to build the users profiles. Then we used a Bayesian network to compare the profiles with the data. To train the system, we used the first 5,000 interactions for each user and feature (keyboard, mouse, and so on).

Calculate accuracy. We matched the trained system profile against the user's subsequent behavioral data. The user profiles were built dynamically during the analysis and used to match against the correct user. Because the first 5,000 interactions were used as the training phase, we didn't include them in the score for the correct user. We used data from users different from the current profile to simulate attacks.

Results

According to our results, keystroke dynamics, mouse usage, and system footprint can be reliable for continuous authentication of computer users.

Keystroke Dynamics

The results in Table 1 show that keystroke dynamics can achieve reliable continuous authentication of users for small groups. Correct users were never falsely rejected, and incorrect users were recognized after 38 interactions (between 20 and 25 keystrokes). Further studies are needed to show if this is valid for larger groups.

Figure 1a shows the score and the corresponding trust of a correct user over the full period of 10 weeks, with the training phase indicated.


Even when the score is fluctuating heavily for the correct user, the trust is stable after the training period, only twice dropping below 0.5. Figure 1b shows the score and trust for an incorrect user. This user's score is, on average, lower than the score for the correct user. The trust of the incorrect user is fluctuating heavily over the observed period, so it's easy to detect the incorrect user.

Mouse Usage

We analyzed the mouse movements together with the monitor resolution. The mouse position for different applications were generated and used as system input. Users took a long time to be falsely rejected (17,470 interactions), while incorrect users were quickly recognized (88 interactions).

Training times

The required training times for users shows that a profile can be trained with only (median) 103 keyboard and 6.606 mouse interactions.

Our work has shown that using our metrics, it's possible to successfully authenticate users in a continuous setting. Further work will test these results with bigger groups and study additional parameters, such as failure to enroll or failure to acquire. 

Acknowledgments

This work was supported by the Active Authentication Program of the DARPA under Grant DARPA-BAA-12-06.

References

1. D. Kahn, *The Codebreakers: The Story of Secret Writing*, Macmillan, 1974.

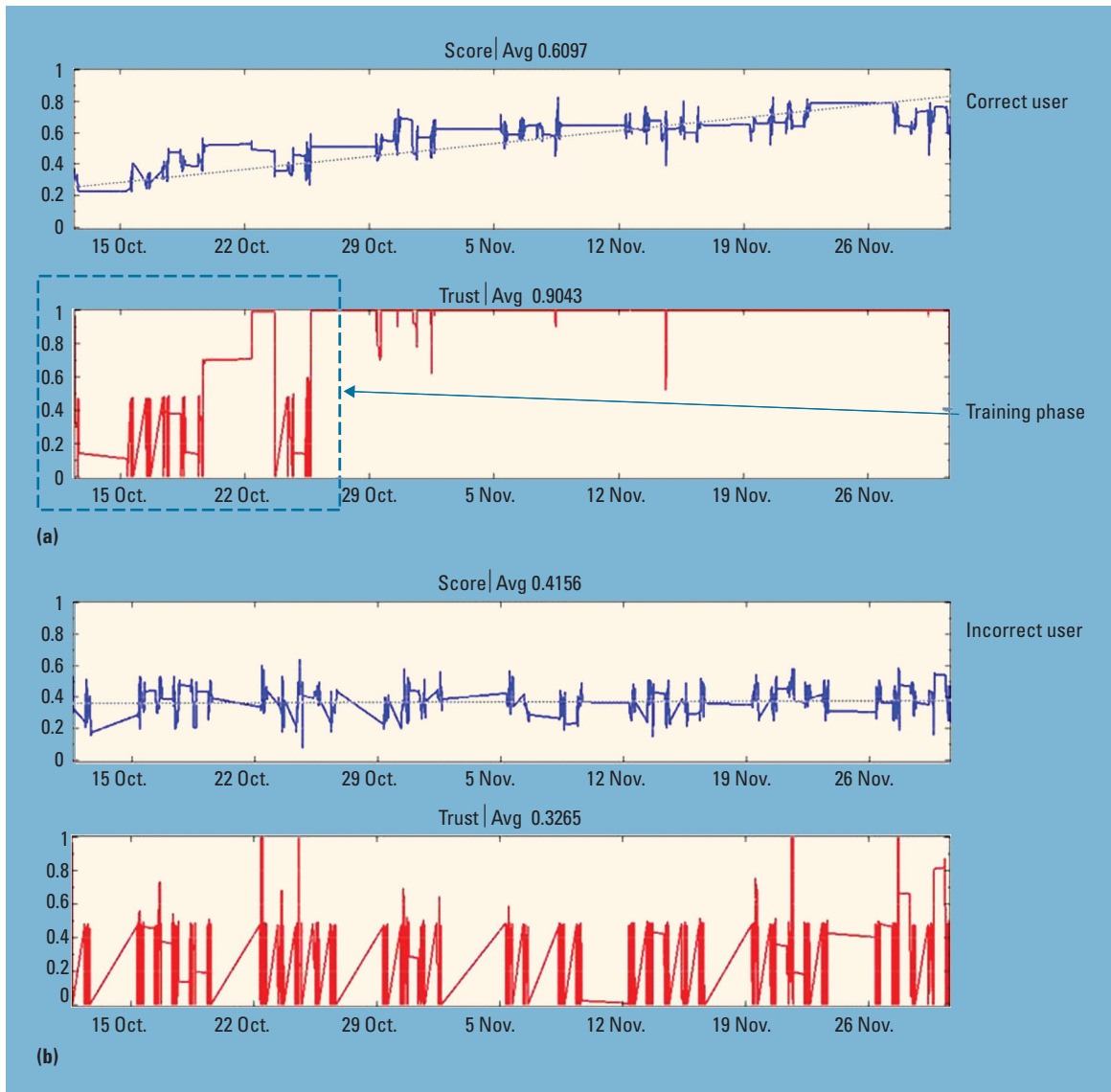


Figure 1. Comparison of readouts for correct and incorrect users: (a) a user in the training and authentication phase and (b) an incorrect user in an authentication phase.

2. K.S. Killourhy and R.A. Maxion, "Comparing Anomaly Detectors for Keystroke Dynamics," *Proc. 39th Ann. Int'l Conf. Dependable Systems and Networks (DSN 09)*, IEEE CS, 2009, pp. 125–134.
3. C.C. Tappert, M. Villani, and S.-H. Cha, "Keystroke Biometric Identification and Authentication on Long-Text Input," *Behavioral Biometrics for Human Identification: Intelligent Applications*, Medical Information Science Reference, 2009, p. 342–367.
4. J.V. Monaco et al., "Developing a Keystroke Biometric System for Continual Authentication of Computer Users," *Proc. Intelligence and Security Informatics Conference (EISIC 12)*, 2012, pp. 210–216.
5. P. Bours, "Continuous Keystroke Dynamics: A Different Perspective Towards Biometric Evaluation," *Information*

Security Technical Report, vol. 17, nos. 1-2, 2012, pp. 36–43.

6. M. Bromba, "Evaluation Criteria and Evaluations for Biometric Authentication Systems," 2009; www.bromba.com/knowhow/bioeval.htm.

Ingo Deutschmann is the business development director at BehavioSec. Contact him at ideutschmann@behaviosec.com.

Peder Nordström is the cofounder and vice president of research and development of BehavioSec. Contact him at pnordstroem@behaviosec.com.

Linus Nilsson is a sales engineer at BehavioSec. Contact him at lnilsson@behaviosec.com.