The Role of Environmental Factors in Keystroke Dynamics

Kevin S. Killourhy
ksk@cs.cmu.edu

Dependable Systems Laboratory
Computer Science Department
Carnegie Mellon University
5000 Forbes Ave,
Pittsburgh, PA 15213

Abstract

Keystroke dynamics is a promising biometric with several potential applications, and yet it is a sensitive instrument with unknown limitations. In other biometric domains, environmental factors (e.g., the scanner technology used for fingerprints or the headset used in speaker recognition) can have deleterious effects on the accuracy of algorithms. The goal of this work is to establish whether keystroke-dynamics algorithms are also vulnerable to the effects of environmental factors (e.g., the keyboard, operating system, and processor load).

As a first step in exploring these effects, we develop a method for establishing the error introduced in recorded keystroke timestamps. We conduct an experiment that reveals one environmental factor (I/O load) that can greatly increase the error in calculating keydown-keydown latencies. We acknowledge that an adversary might exploit this error to evade detection, and we consider options that would mitigate the risk. Finally, we judge this first step in establishing the role of environmental factors to be a success, and we consider further analysis and experimentation.

1. Introduction

Keystroke dynamics—the analysis of typing rhythms to discriminate among users—is a promising biometric with many potential applications [6]. For instance, a program might differentiate between (a) a genuine attempt to authenticate using a password, and (b) an impostor with a compromised password. Alternatively, keystroke dynamics might provide a continuous reaffirmation of a genuine user's identity during regular typing, without requiring an explicit authentication step.

However, keystroke dynamics is a sensitive instrument in a noisy domain. Even if two users could be discriminated by their typing under ideal conditions, environmental factors such as the keyboard, operating system, and programs running on the system may obscure these differences. If algorithms are deployed without accounting for environmental effects, unforeseen changes in the environment might cause the algorithms to fail without warning (e.g., an algorithm works in the laboratory but fails in the field). More alarming is the possibility that an adversary might adjust environmental conditions (e.g., adding load on the system) to avoid identification. The goal of this work is to establish the role that environmental factors play on keystroke-dynamics algorithms.

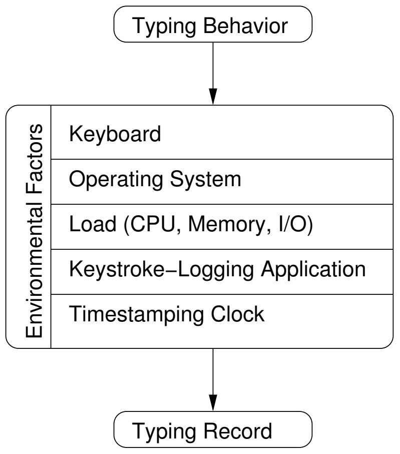

Figure 1. This illustration shows how environmental factors stand between a typist's actual keystrokes and the timestamps that are recorded. The various factors add delay, which manifests as error in keystroke timing calculations. This error reduces the accuracy of keystroke-dynamics algorithms.

Figure 1 shows five environmental factors, and illustrates that—coming between the typist's actual behavior and the digital record of that behavior—they might introduce delays in the recorded timestamp. The keyboard's electronics buffer keystrokes; the operating system enqueues keystroke events; other programs compete for finite resources; the keystroke-logging application may be inefficient in assigning a timestamp; and, the clock, the source of the timestamps, may be skewed or lack sufficient precision. Any of these factors may introduce errors in the timestamps. These errors will compound when the *keydown-keydown latency* (the difference between successive timestamps) is calculated. Latencies are the input to most keystroke-dynamics algorithms, and errors may reduce the ability of these algorithms to discriminate between typists.

2. Related work

More established biometrics have also had to contend with issues regarding environmental noise. In fingerprint identification, researchers determined the effect of different fingerprint scanning technologies on the fidelity of the digital image of a print [1]. In speaker recognition, a speaker who changes handsets was discovered to be more difficult to recognize than one consistently using the same handset; methods were developed to accommodate different handsets [2]. The effects of environmental factors on keystroke dynamics must be explored and mitigated just as in these other biometric disciplines.

In earlier work with Maxion, we showed that using standard computer clocks with a resolution of 15ms (milliseconds), instead of a high-precision timer, can increase the equal-error rate of keystroke-dynamics algorithms by 4.2% [3]. For some deployment scenarios, near-perfect accuracy is required, and even a small increase in the error rate will be problematic. The finding that clock resolution makes a difference in algorithm performance led to the current, broader investigation into the effects of other environmental factors.

3. Objective and approach

The broad goal of this research project (current and future) is to measure and characterize the effects of environmental factors on keystroke-dynamics algorithms. The long-term strategy involves two phases. In phase one, we plan to measure how much error each of the five environmental factors introduces in keydown-keydown latencies. In phase two, we intend to measure what effect the error in keydown-keydown latencies has on the accuracy of keystroke-dynamics algorithms.

In the current work, we aim to develop a methodology that will make phase one possible. Specifically, our objective is to design an experimental procedure to measure the effect that a particular environmental factor—the load induced by other programs competing for CPU, memory, and I/O resources—has on latency errors.

We follow a four-step approach.

1. Develop a device for automatically generating keystrokes at a fixed rate, thereby establishing the true keydown-keydown latency between successive keystrokes.
2. Design and implement a keystroke-logging application for the Windows operating system. The application records timestamps using two different timestamping clocks: a standard Windows-event clock and a high-precision timer.
3. Develop a load-inducing program that spawns processes which contend for fixed amounts of CPU time, memory space, and I/O bus usage.
4. Run a series of calibrated experiments. The keystroke generator taps out keystrokes at a fixed rate; the logging application records timestamps; and the resource usage of the load-inducing program is systematically varied, thereby allowing us to quantify the effect on latency errors.

In the process of establishing the role that load plays, we will also be testing out the general methodology for measuring the effect of additional environmental factors.

4. Methods

We describe the four steps of the approach in more detail.

4.1. Keystroke generator

To measure the error introduced by environmental factors on keydown-keydown latencies, we must have an accurate ground-truth account of those latencies. Ground truth is obtained by using a device to generate keystrokes at a known rate.

The device was constructed by removing the keyboard encoder in a Dell USB keyboard from its housing. It was attached to a solid-state relay so that when the relay switch is closed, the keyboard encoder registers that a key is pressed, and when the switch is open, the key is released. A function generator was used to generate a square wave that drives the relay to open and close the switch. When the function generator is set to 1Hz, the device generates a keydown at 1-second intervals. By changing the frequency of the wave, we change the true keydown-keydown latency.

In the future, we should be able to explore the effect of the keyboard on keydown-keydown latencies (e.g., whether USB and PS/2 keyboards have different effects), by swapping the present keyboard encoder with others.

4.2. Keystroke-logging application

A Windows XP application was developed to record keystrokes and the times at which they occurred. The application displays a text-entry box, and when keys are pressed or released, it stores the details of the key event (e.g., whether it was a keydown event or a keyup event and what key was pressed) along with two timestamps, which we call the *Windows-event* timestamp and the *QPC* timestamp.

When a key is pressed, the application receives an event message from the Windows operating system. The event message contains a timestamp field that is set by the Windows XP operating system [4]. This timestamp is recorded as the *Windows-event* timestamp. When the event is received by the application, it queries a counter available in Windows XP for high-precision timestamping [5]. Since the function to generate the timestamp is called `QueryPerformanceCounter`, the timestamp is recorded as the *QPC* timestamp.

Two types of timestamp are recorded because it was a simple addition to the program, and it allows us to examine whether the effect of load differs for different timestamping mechanisms.

4.3. Load-inducing program

A load-inducing program was developed to generate workloads that compete for CPU, memory, and I/O resources. The program generates workloads similar to the Unix tool, `stress` [7], but for a Windows environment. Specifically, when invoked, the program spawns a configurable number of subprocesses, each of which performs a simple action designed to use system resources. There are three types of subprocess: CPU, memory, and I/O.

A CPU subprocess simply generates a random number and calculates its square root in an endless loop. A memory subprocess allocates a block of memory, writes to it, and holds it for a period of time. After the wait time has expired, the subprocess frees the block, allocates a new block, and repeats the cycle. The size of the block and the wait time are configurable. In this experiment, we used blocks of 14MB and a wait time of 60 seconds. An I/O subprocess writes a file to disk and deletes it in an endless loop. The size of the file and the size of each write-block is configurable. In this experiment, we used a file size of 10MB and a write-block size of 1MB.

This load-inducing program is simple and intended only to stress the system in well-understood ways. More realistic and more diverse workloads might be used in the future.

4.4. Experimental procedure

The following five-step procedure was used in a series of experiments to measure and summarize the keydown-keydown latency error.

1. Set the keystroke generator to produce keystrokes once a second, and record 1001 keystrokes using the keystroke-logging application.
2. Repeat step 1 a total of seven times, generating 1001 keystrokes at intervals of 1sec, 500ms, 333ms, 200ms, 143ms, 91ms, and 77ms (chosen to reflect typical keydown-keydown latencies over the range from slow to fast).
3. For each of the 7 intervals, calculate 1000 keydown-keydown latencies using the Windows-event timestamps, and 1000 latencies using the QPC timestamps, so we have 7000 latencies for each timestamping mechanism.
4. For each latency, calculate its absolute error (i.e., the absolute difference between the calculated latency and the true latency, given the settings of the keystroke generator).
5. To summarize these 7000 latency errors into a single number, sort them and find their 95th percentile. In other words, calculate the number x such that 95% of the observed errors are within $\pm x$ of zero error.

For our preliminary results, we use the 95th percentile since it summarizes the amount of error that will typically be seen (i.e., 95% of the time). A more comprehensive analysis of the distribution of the latency errors is planned.

First, we use this procedure to establish a baseline error with no induced load. The latency error was measured on a standard Windows XP machine with 1GB of memory and approximately 40GB of disk space (23GB free). Then, we added CPU load (1, 32, and 64 subprocesses), measuring the latency error at each step. Next, we added memory load (1, 32, 56, and 64 subprocesses) and measured the latency error. Finally, we added I/O load (1, 32, 56, and 64 subprocesses) and measured the latency error. Any more than 64 subprocesses caused system errors (e.g., the process table was full). We did not combine loads (e.g., memory and I/O), but that is a possible avenue for further test.

5. Preliminary results and analysis

Figure 2 compares the 95th percentile of the Windows-event latency error and the QPC latency error as a function of CPU, memory, and I/O load. At the baseline (i.e., zero load of any kind), the Windows-event latency error is about 16ms, which is consistent with the 15.625ms resolution of the clock [3]. However, the QPC latency error is about 8ms, much higher than the resolution of the clock (280 nanoseconds). It may be that other environmental factors (e.g., the keyboard and operating system) are introducing delays which increase the latency error despite the high-precision of this timestamping mechanism.

The top graph in Figure 2 demonstrates that increasing the CPU load has little effect on the latency error of either timestamping mechanism. The middle graph demonstrates that increased memory load has no effect at all on the Windows-event latency error, and no effect on the QPC latency error until a threshold (64 subprocesses, when available memory is exhausted) at which point the latency error increases rapidly. The bottom graph shows that increased I/O load has a severe effect on QPC latency error, but no effect on Windows-event latency error.

6. Discussion and future work

We found that at least one environmental factor—massive, ongoing I/O operations—increases the error in QPC timestamps but not in Windows-event timestamps. The error in the QPC timestamps is on the order of hundreds of milliseconds, surely affecting the accuracy of keystroke-dynamics algorithms.

The additional error in the QPC latencies could be a consequence of I/O load causing delays in context switching. A keystroke triggers an interrupt within the operating system, and the operating system assigns the Windows-event timestamp. Then, a context switch passes control to the keystroke-logging application, and the application assigns the QPC timestamp. Delays in context switching would manifest as QPC latency errors.

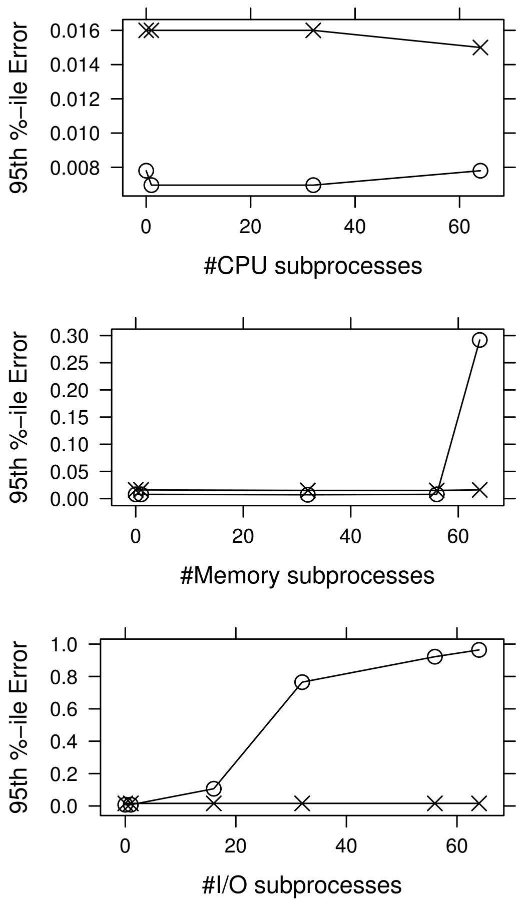

Figure 2. The 95th percentile of the latency error is shown as a function of increasing CPU, memory, and I/O load for each of two timestamping mechanisms. The Windows-event latency error ($\times$) is not affected by the loads. The QPC latency error ($\circ$) increases under high memory load and moderate I/O load.

An alarming implication of this result is that keystroke-dynamics using QPC timestamps either will generate spurious false alarms when I/O intensive programs are run (e.g., automatic backups), or may miss an adversary who uses I/O intensive programs to evade detection. Despite having a higher baseline latency error, Windows-event timestamps may be preferred over QPC timestamps due to their robustness under load.

These preliminary results demonstrate that environmental factors do have an effect, and they uncover some interactions between factors to be explored. Consequently, we intend to continue with phase one of our research project, measuring how much error each environmental factor introduces in keydown-keydown latencies.

For phase two, some planning and preparation is still required. In order to measure the effect that latency error has on the accuracy of keystroke-dynamics algorithms, we need to evaluate those algorithms using realistic keystroke data (i.e., from real typists, not our keystroke generator). To do so, we must either conduct extensive user studies (to obtain data with the effects of each environmental factor), or determine a valid means of *adding* the latency-error effects of an environmental factor to pristine user data.

7. Summary

In other biometric domains, the effect of environmental factors on the accuracy of algorithms has been investigated, and deleterious effects have been identified and mitigated. The goal of this work is to establish whether keystroke-dynamics algorithms are also vulnerable to the effect of environmental factors. We present a method for establishing the error introduced in keydown-keydown latencies by environmental factors. Our preliminary experiments demonstrate that one environmental factor (I/O load) can greatly affect the latency error, with implications for the design of robust keystroke-logging applications. We consider this first step to have been a success, and are designing followup experiments and analysis.

Acknowledgments

This work was supported by National Science Foundation grant numbers CNS-0430474 and CNS-0716677. Many thanks to Harvey Vrsalovic for electronics help, and to Roy Maxion, my advisor, for guidance on experimental design.

References

[1] R. M. Bolle, J. H. Connel, S. Pankanti, N. K. Ratha, and A. W. Senior. *Guide to Biometrics*. Springer-Verlag, New York, 2004.

[2] G. R. Doddington, M. A. Przybocki, A. F. Martin, and D. A. Reynolds. The NIST speaker recognition evaluation – overview, methodology, systems, results, perspective. *Speech Communication*, 31(2–3):225–254, 2000.

[3] K. Killourhy and R. Maxion. The effect of clock resolution on keystroke dynamics. In R. Lippmann, E. Kirda, and A. Trachtenberg, editors, *International Symposium on Recent Advances in Intrusion Detection*, volume 5230, pages 331–350, September 15–17, 2008, Boston, MA, 2008. Lecture Notes in Computer Science (LNCS), Springer-Verlag, Berlin.

[4] Microsoft Developer Network. EVENTMSG structure, 2008. `http://msdn2.microsoft.com/en-us/library/ms644966(VS.85).aspx`.

[5] Microsoft Developer Network. QueryPerformanceCounter function, 2008. `http://msdn2.microsoft.com/en-us/library/ms644904(VS.85).aspx`.

[6] A. Peacock, X. Ke, and M. Wilkerson. Typing patterns: A key to user identification. *IEEE Security and Privacy*, 2(5):40–47, 2004.

[7] A. Waterland. Stress project page, 2008. `http://weather.ou.edu/~apw/projects/stress`.