KeystrokeAuth

Forrest Pieper
Carlo Biedenharn
Kenneth Seibert
Ameesh Goyal

December 10th, 2013

Abstract

1 Introduction

KeystrokeAuth is an example website implementation that uses keystroke timing to provide stronger user authentication. Measuring keystroke timing is a method for passive biometric authentication. Traditional biometric authentication such as fingerprint or retinal scanners require external hardware and is not suited for web service authentication where users may login from a variety of machines. Keystroke timing can be gathered using javascript embedded in the login and registration pages. Thus this method requires no additional hardware. The user must enter her password several times during registration instead of just once or twice, but other than that this method is completely unobstrusive. Authenticating passwords with keystroke timing makes it more difficult for an attacker who possesses a user's plaintext password to compromise the account. Additionally, it discourages account sharing which may be useful for highly secure systems and premium acconts. In this paper we describe past work on the topic, introduce our example implementation called KeystrokeAuth, analyze the added security of our system, and examine a small set of test data.

2 Background and Related Work

TODO: forrest / carlo / kenny / ameesh

describe various features

1. flight:
2. dwell:
3. down-down:
4. up-up:

describe various detectors

3 KeystrokeAuth Implementation

KeystrokeAuth uses javascript to capture the timestamps on each keydown and keyup event while typing the password. During registration, the user enters her password 10 times. The data is sent to the server and KeystrokeAuth computes a model specific to that user and password. When logging in, the user enters the password once and KeystrokeAuth compares the new timing data to the registered model. If the timing data differs by too much, the user will not be logged in.

3.1 Gathering Timing Data

TODO: forrest

Gather code, uptime, downtime –¿ compute various features

3.2 Generating User Timing Model

In order to have success with verifying users using keystroke dynamics, we first have to collect a set of training data. The keystroke dynamics that we are interested in include down time, down-down time, flight time, and dwell time which are defined below.

Down time: The time that each key is pressed down at, where the first key pressed down equates to a time of zero

Down-Down time: The time in-between two consecutive key down events

Flight time: The time in-between a key up and the following key down event. This value can be negative as a key may not be released before the next is pressed down.

Dwell time: The time that a given key is held down

We are able to calculate all of these values from the initial 10 password entries that are passed in during registration. The calculated data is then encrypted with the users password and then stored for the given user. These metrics will be used when a login attempt is received to determine whether or not to authenticate a given user. TODO: Ameesh add stuff about how things are actually encypted

3.3 Login Timing Authentication

In order to determine whether or not the user attempting to log in should be authenticated, we collect the down time, the down-down time, the flight time, and the dwell time to compare against the training data. We make use of the mahalanobis distance to compute the similarity between two vectors.

$$D_m(\vec{x}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})}$$

We make use of the mahalanobis distance in both of authentication functions. The first function we use takes the mean of the training data vectors and computes the distance between the login attempt and the computed mean vector. If the distance from the mean vector is below a certain threshold, then the login attempt is accepted. The 2nd function that we employ is we take the login attempt and find its distance from each of the training vectors. If the k closes distances all fall below a threshold, then the login attempt is accepted.

The threshold is computed using the training data. We find the mahalanobis distance between each vector in the training data. We then compute the average distance. The threshold is set to be one deviation away from the mean in the direction of smaller distances. In other words we only accept login attempts where the distance of the attempt is at least one deviation than our computed mean distance.

4 Security Analysis

TODO: forrest

proof that security is not worse

strategy: make it no less convinient/difficult for users, and at least slightly more secure

5 Data Collection and Analysis

TODO: carlo

5.1 Data Overview

5.2 Feature Comparison

5.3 Error Rates

Goal: Find ideal thresholds and weights for each feature that give 1% false negative rate ideal would be .001% false positive, but anything less than 100% is an improvement over state of the art Compromise: increase false positive rate to accomodate 1% false negative

6 Conclusion

TODO: forrest / carlo / kenny / ameesh

References

[1] Killourhy, Kevin S., and Roy A. Maxion. "Comparing anomaly-detection algorithms for keystroke dynamics." *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on.* IEEE, 2009.

[2] Cho, Sungzoon, et al. "Web-based keystroke dynamics identity verification using neural network." *Journal of organizational computing and electronic commerce* 10.4 (2000): 295-307.