Ryan McGowan
June 6, 2012

Cadence

Pattern Recognition on User Input
for Authentication

Introduction

Typing style has been acknowledged as a legitimate way to identify users for decades. DARPA and others are currently using this phenomena to develop continuous authentication systems (CAS). The purpose of a CAS is to thwart attackers who could some how hijack a session of a user post initial authentication.

Consider some fictional user Bob. Bob logs in to his computer and decides to take a trip to the kitchen. An attacker, Eve, could swoop in and use the computer with all of the permissions Bob has because once authenticated the computer assumes the current user is still Bob thus allowing Eve to run malicious code as Bob or to do some other nasty business. Some may consider this behavior a feature since it easily allows others to user Bob's computer if he logs in for them. Despite this convenience session hijacking is a serious security concern. Thus we have the existence of so called “lock screens” which forces the user to reauthenticate with his or her password once the lock screen is engaged.

CAS protects against this sort of attack by, like the name suggests, continuously authenticating the current user by monitoring various input devices (mainly keyboard and mouse).

However, even with CAS the first step the user takes to use a system is to enter his or her username and password. When one considers the original purpose of CAS the persistent requirement of username and password is not surprising, but is it necessary?

Perhaps not, but confidence in this “new” type of authentication is low and then there is the issue of training. A user who has never used the system before would not have enough associated typing style data to be accurately identified. Despite CAS being a helpful tool for security it is not common. This is due to the issues described above and a lack of a common implementation.

CAS for the Web

The current state of authentication on the internet is almost unanimously password based. There are new protocols like OAuth and OpenID that allow secure sign in using various keys

and tokens, but even they require the initial step of signing in with a username and password.

To be clear, password authentication is a secure method of authentication if the password is carefully chosen. Unfortunately, it seems that a carefully chosen password is the exception, not the rule. A list of 10,000 most common passwords matched 98.8% of passwords, the top 1,000 matched about 91%. Creating a memorable, and secure password is often a struggle for the user.

There is no true implementation of CAS for the web despite there very much being a desire for one. Take most eCommerce sites as an example. They might not know what CAS is, but they try to emulate it by often requiring their users to authenticate a second time before completing their purchases. This secondary authentication indicates a desire for some kind of CAS. Unfortunately, secondary authentication does little to increase security.

It is common for a user to remain logged in to an internet based account for an extended amount of time through the use of cookies or perhaps the user has passwords saved on his or her computer to make signing in less of a chore. These tools do make life easier for the user but there is a cost. Eve could then simply drop in at a computer with stored passwords and repeatedly authenticate as Bob at her pleasure thus negating any security benefit of using secondary authentication.

DARPA style CAS, which continually monitors typing style cannot be so easily applied to the web. Firstly, a browser is often a black box allowing no direct access to the user's system by the website. This is a necessary ssecurity feature of most modern browsers, but in the case of implementing CAS for a website it is a major limitation. Furthermore, privacy is a serious concern for users. A major difference between DARPA CAS on a local system and on the browser is that the user has some say in how the local CAS is configured and has the choice to use it or not. Conversely, on the internet the user does not control the application stack of every website they visit. In general, storing sensitive information about a user from a website (like typing style information) is a huge liability and security concern. Thus, continuously taking user typing data is both a confusing mess and possibly a breach of privacy.

In summary, password based security is at the mercy of the user and secondary authentication adds little real security for the problem it is trying to solve. DARPA CAS seems like a valid alternative, but is near impossible to implement effectively in the browser and also could be viewed as a breach of user privacy.

The Advantages of Cadence

Cadence authentication is a marrying of typical secondary authentication and DARPA style CAS. In an effort to avoid privacy concerns Cadence only monitors typing style in a single input field during authentication and training. It has been designed in such a way that it could easily be fitted to be a drop in replacement for password based secondary authentication. However, unlike its password based counterpart, Cadence requires first hand user interaction and can not be forgettably bypassed by some stored passwords setup.

Like DARPA style CAS, Cadence analyzes the user's typing style. During the authentication process a users *cadence*, a simple object containing a timeline of typing style information, is generated as the user types in a given phrase. This *cadence* is then passed to the application and classified by a classifier constructed from previous entries of the same phrase. Thus the security of this system depends on the quality of the classifier and the uniqueness of the user's typing style. Relative to standard secondary authentication Cadence could be considered more secure. Consider that a voluntary eye-blink takes 275 milliseconds, and the average length of a key press is 100 milliseconds. For an attacker to successfully mimic another user's typing style they would have to quite accurately account for the subtle differences in keystroke timings to be successfully. These minute several millisecond differences can not be easily replicated by a human.

The Cadence Demo

To illustrate the effectiveness of this type of authentication a demo has been designed and constructed. Access links are provided on the final page of this report.

The demo was designed to be as simple and user friendly as possible while still informing the user. The main page of the site is purely informational and includes no significant functionality on its own. Before a user can test out cadence based authentication they must register and log in. This is a necessary step since individual *cadences* must be associated with whomever generated them. Once signed in the user may access the training page.

The classifier used in the Cadence demo uses a supervised learning algorithm to construct its classifiers. Thus, the classifier must be trained before it can be used. The training process for the user is indirectly related to the actual training of the final classifier used to authenticate them. For the user, training is simply repeatedly typing in a given phrase some arbitrary number of times. In the demo, the user

is required to repeat the given phrase six times. Once completed the user is "trained" for that phrase and he or she could then use the authentication page.

The authentication page is very similar to the training page. There are two fields, one containing the given phrase to type in and the other is available for the user to type in the phrase. The *cadence* object generated from the user's successfully typing in the entire given phrase is sent to the server for analysis.

The server either selects a previously constructed and stored classifier for the given phrase-user pair or it constructs a new one on the fly. The classifier used is a two class classifier constructed from existing *cadences* of the given phrase for both the currently authenticating user and a random selection of other users who have also completed training on that phrase. The current user's past data is labeled as *good* while the random users' data is labeled as *bad*. The time line of the *cadence* object is converted to a vector and these vectors along with their labels are used to build the classifier.

Once the classifier has been constructed or selected the *cadence* generated during authentication is classified. If the output label is *good* the user is successfully authenticated otherwise, if the output label is *bad* the user's attempt at authenticating is rejected.

Implementation Decisions

Taking advantage of established machine learning libraries, *clj-ml* was used to assist in the construction and use of classifiers. *Clj-ml* is really just a Clojure wrapper around a machine learning library written in Java called *Weka* that offers a simplified, Clojure friendly interface to *Weka* functions.

The classifier used by the Cadence demo is *Weka's* implementation of a support vector machine using sequential minimal optimization. The kernel chosen is the radial basis function. This choice was entirely motivated by the quality of its predictions. For instance, *Weka* offers several other algorithms, but based upon evaluation results via cross-validation SVM was the most effective. Furthermore, the value of the gamma parameter (σ^2) was doubled to due to over-fitting problems.

At the moment the demo has several limitations. Firstly, the given-phrase is chosen from a static set. They are not generated nor do they fit any criteria they were selected arbitrarily. Secondly, at this point the classier is rebuilt every time the user attempts to authenticate as a result of not being able to effectively store classifiers. Thus performance is somewhat lacking. Also, a lack of testing data means classifiers are built from relatively small sets and evaluation cannot be done with samples lest

the size of the training data be too small.

Further Extensions

As described above, there are some shortcomings with the current demo although it is a complete proof of concept it could be refined to be more convincing. Further research is required as many questions still need answering. For instance, does the phrase used during training need to match the one used for authentication? It is possible that a user could be uniquely identified based on the way to they do only a few operations. Perhaps the time user *A* takes to move from an *a* to a *k* is completely unique to them and identifies them on their own. More likely some small set of these sorts of movements could uniquely identify a user. The difficulty lies in determining what characteristics could be used and whether or not different metrics need to be considered for different users.

Another interesting question is how might phrases be generated dynamically. Are there any requirements or considerations necessary to construct a *cadence* ready phrase? The phrases chosen for the current demo are largely arbitrary besides a few simply criteria. The phrase had to make sense and it had to be at least 10 characters, but no more than 30. Could dynamic phrase generation improve security? Could the the design of the training and authentication process be altered such that they are combined (i.e. training is done with authentication). ReCaptcha does this quite effectively.

A final extension is portability. How could *cadence* style authentication be used on mobile devices? The keyboard is not the favored input device on a mobile phone, touch screens and gestures are. Unfortunately, this projects time constraints prevented further investigation in these areas, however the answers could be very valuable.

Conclusion

Despite various limitations, the Cadence demo is a working proof of concept that shows how typing style authentication can be applied to websites to help increase the security of secondary authentication. Furthermore, the possible extensions of this type of authentication could produce more robust forms of authentication as well.

Machine learning has influenced nearly every aspect of computing on the the internet besides authentication. As a result security on the internet is always in question. The use of cadence authentication is a step away from passwords into a new realm of truly user unique authentication.

Works Cited and Other Links

- Randall Stross, “Bypassing the Password.” The New York Times, 2012-3-17. http://www.nytimes.com/2012/03/18/business/seeking-ways-to-make-computer-passwords-unnecessary.html
- Cadence (the demo). https://cadence.herokuapp.com/
- Cadence.js. https://github.com/RyanMcG/Cadence-js
- clj-ml. https://github.com/leadtune/clj-ml