A Case for Continuous Authentication

Written by

Prof. Suman Banerjee, CTO, StratusWorX & Professor of Computer Sciences, Univ. of Wisconsin-Madison

Alok Sharma, PhD, CEO and President, StratusWorX

Kamran Ziaee, Group CIO, CenturyLink

Jamie Lin, Director, Cloud Infrastructure and AI, CenturyLink

Abstract

Authentication of users, devices, organizations, and entities in the digital world is a key enabler of any security architecture. Today's common security solutions adopt an entrypoint based approach for authentication where an accessing entity is asked to complete a challenge (often to enter a password and a one-time shared secret) at initial access request. Such approaches have limitations in situations where the secrets have been leaked (such as, easily guessable passwords, or side channel attacks that reveal them). This document describes a powerful complement to such basic security mechanisms, called Continuous Authentication, whereby the identity of the entity requesting access can be continuously tracked and verified against known profiles of the entity. A Continuous Authentication system can significantly enhance the overall security of systems today.

1. Introduction

Authenticating an entity as the intended user of any software or a device has always been one of the most critical tasks to be undertaken by the administrators of computer systems. The importance of authentication comes from the fact that any breach inside the system can result in theft of important business secrets, monetary loss, and can pose significant risk to the life and well-being of human population. For example, in cases of some critical infrastructure, like power stations, traffic control system, or large command and control systems that manage a nation's defense infrastructure, breaches can be deadly. Security of all of these systems, therefore assume a paramount responsibility of the operators and administrators.

1.1 The present-day landscape of authentication

Present-day authentication mechanisms, predominantly adopt an **entry-point** based approach, wherein any entity which wants to access the system/services is made to successfully authenticate itself by successfully completing some challenge at an authentication gateway before being let in. Inability to authenticate oneself will lead to denial of access to the service, while successful authentication will generally allow the entity to be given all the privileges that the authenticated user (which we authenticate the entity to be) is entitled to. Figure 1 depicts the steps for authentication described above.

As an illustrative example, consider the authentication systems used by banks for access to their on-line banking facilities. Any entity desirous of conducting some bank transaction for a specific account, will first need to login by submitting accurate information demanded by the authentication page.

Entry point based mechanisms, allow the entity, who has authenticated as a specific eligible user, to perform all activities (contingent on the user having the privilege to perform the action in the first place) till the authenticated session is over (entity logs out or the session times out).

Over the period, entry-point-based authentication techniques have evolved significantly, and today, it is common to find authentication systems use a combination of techniques for authentication.

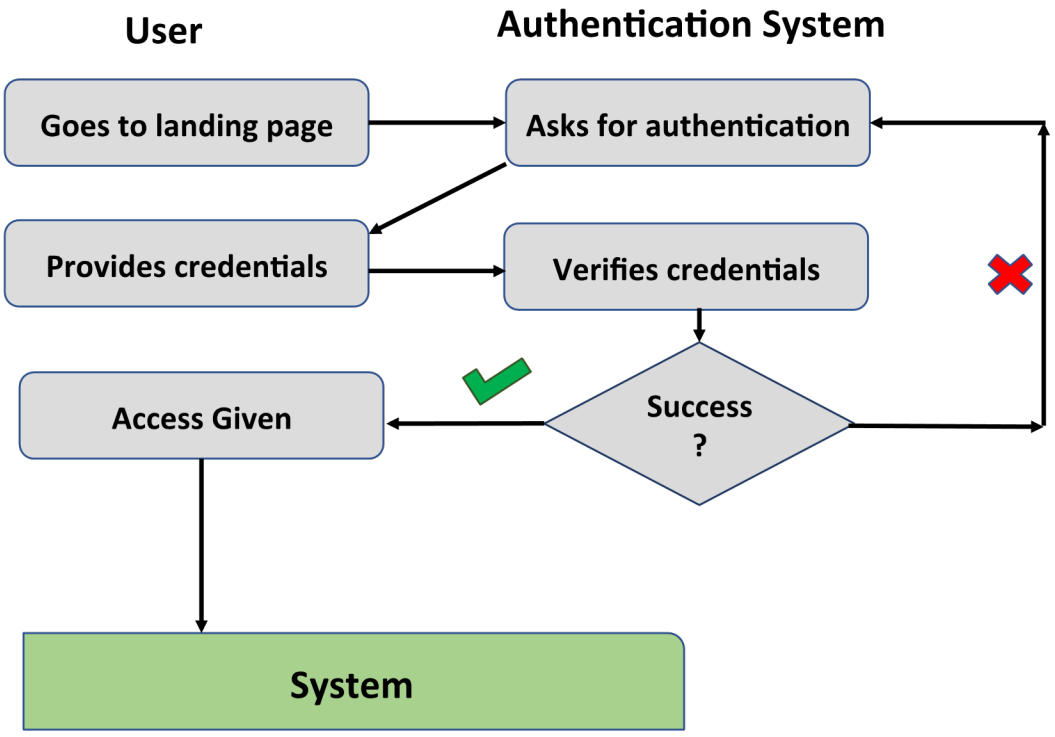

Figure 1: Example interactions for an entry-point based authentication system

The most common technique used for entry-point authentication is **password-based authentication**. In this method, the entity, is expected to provide the "correct" password (for a specific user's login id) at the authentication gateway for successful authentication (Figure 1). The assumption is that the password is known only to the user and the authentication system) and hence, the entity who furnished it correctly to their authentication gateway has to be the user.

Another popular technique for entry point-based authentication involves using **security tokens**. The security (authentication) token method depends on existence of a unique communication channel (small hardware, smart card, mobile phone, email etc.) between the user and the authentication system. The communication channel is used to generate a secret key (or communicate), which can then be authenticated by the authentication gateway. The assumption in this case is that the user is the only one in possession of the independent communication channel and hence will be the only entity who can provide the secret.

Apart from the above, biometrics-based techniques (such as fingerprint, iris, voice, facial matching etc.) also used in certain cases where appropriate hardware support is available. Similarly, PKI (public key base cryptography) and digital certificate-based authentication systems are also popular traditional authentication mechanisms.

Figure 2: Example of interactions in a two-factor authentication system

Authentication systems nowadays combine multiple authentication mechanisms in a tiered manner with to increase the stringency of authentication. create a tiered authentication mechanism.

Two factor authentication is the most prevalent example of such tiered scheme. As the name suggests, two factor authentication test the entity on at least two dimensions. For example, Figure 2 depicts a typical two-factor authentication mechanism. As a first step, it requires entities to present their user credentials to access a system or personal account (could be their password or challenge response using PKI etc.). Once the entity succeeds in first challenge, a second round of authentication using a different technique is done. the second step might be security token based authentication, in which case, the entity might be required to enter another set of credentials that only the user is expected to have – a unique code provided via SMS message sent to the user's mobile phone. Alternatively, the second authentication challenge might be a biometric based challenge such as fingerprint matching.

Note that two-factor based authentication makes it significantly harder for the malicious entities from gaining access to a user's account.

1.2 Issues with traditional authentication mechanisms

While entry-point based methods are the de-facto choice for authenticating users for system access, they suffer from significant shortcomings in terms of usability and security. Over time it has been observed that passwords (and digital certificates and PKI keys) get stolen.

Specifically, password stealing is made easy by the fact that significant number of users set predictable passwords (easy to remember) which can be easily guessed by hackers. Furthermore, users tend to reuse password across different applications, in which case stealing

of one system's password allows the malicious entities to gain access to user's access credentials for multiple systems.

Note that the problem of users setting an easy (and)or common password is even more acute in mobile systems, where the power-save features frequently log out the user, thus necessitating frequent re-authentication by the user. Hence, mobile phone users choose simple (weak secrets), make the screen lock timeouts long, or completely disable the locks. This further exacerbates the risk of password theft compared to other compute devices.

Apart from that, malicious entities such as hackers and cyber-criminals (attackers), motivated by the potential financial rewards (or other reasons), go to great lengths to gain access to user's login credentials. To this end, attackers use a host of techniques to steal the credentials from a user. Techniques such as social engineering, credential theft and Man-in-the-Middle/Browser are prevalent. Such techniques are shown to be successful even against tiered authentication systems such as two-factor authentication. For example, social engineering techniques, involve tricking users into divulging confidential information like login names, passwords, social security number, credit card numbers and other personal information. As an example, a technique known as '**Phishing**' involves sending fake emails or SMS messages which appear to originate from a trusted bank or other organization with whom the user has an account. Figure 3 depicts an example of a 'Phishing' email. The message asks the user to share their credentials to access the account and rectify certain malfunction. A large number of variants of Phishing exist, such as **Spear-Phishing** which is a very directed version of Phishing.

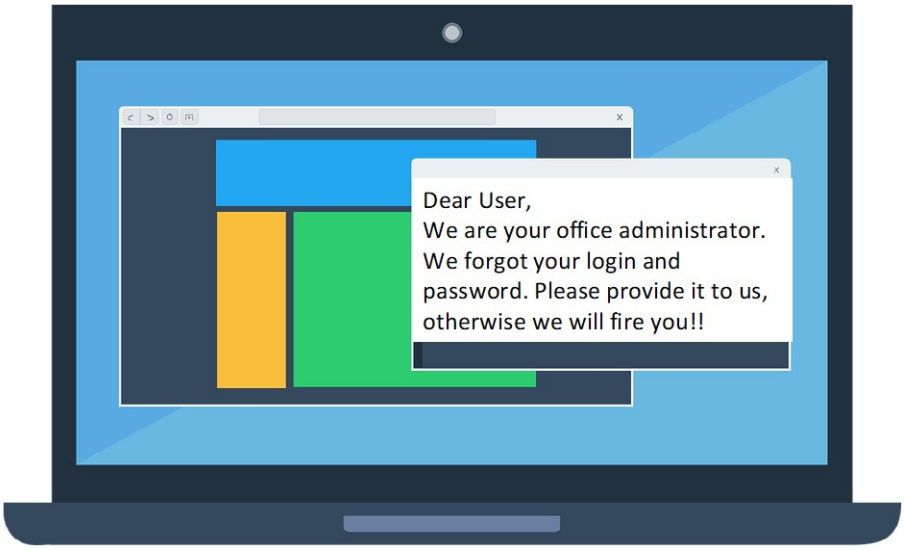

Figure 3: Example of phishing scams

Similarly, Interactive Voice Response (IVR) or Phone Phishing, entails setting up a legitimate IVR system, call up users imitating a specific organization (such as bank) with whom the user has an account and then try to fool target users to divulge their secrets over the phone.

Figure 4: Man in the middle attacks

Man-in-the-Middle/Browser attacks entail fooling the user by spoofing/infecting some component of access mechanism of the user with the goal of stealing authentication credentials. Specifically, Man-in-the-Middle (MitM) attack involve setting up false login pages to trick unsuspecting users to enter their credentials thinking that the page is authentic. The attacker can now user the same credentials to login to the service as the user. Man-in-the-Browser (MitB) attack on the other hand involves injecting the web browser of the victim with some virus which will steal the information that the user enters and send them secretly to the attacker. An even more sophisticated variant of such attack is known as RAT in the Browser (RitB) attack. This attack involves gaining remote access (using infected variants of RDP, VNC etc.) to a legitimate user's system (using trojan) and then accessing the service from a browser on the user's system on which an authenticated session is already in progress (or the credentials are also available with the attacker).

2. Continuous Authentication and its applicability

Given the danger that unauthorized access poses in an ever increasingly digitizing world, it is important to investigate ways to improve the efficiency of authentication mechanisms. This, on one hand, entails, educating users of better security practices. More importantly, it also necessitates, careful understanding of the lacunae of entry-point based-authentication mechanisms and selection of additional authentication techniques, which will compensate the short-comings of entry-point based authentication techniques.

The most critical flaw in entry-point based authentication mechanisms stems from the fact that once an entity has successfully authenticated as a specific user, the system does not re-authenticate till the user logs out (terminating the session) and wants to relogin or when the user has been dormant for a significantly long period of time. In other words, the authentication system completely TRUSTS the entity to be the user (post initial successful authentication) and hence, does not have any safeguards or checks in place to detect intrusion during the lifetime of the session.

Note that, depending on the perceived criticality of certain privileges, some systems might ask the users to re-authenticate before accessing the sub-component, i.e., make them go through another front-door. For example, a banking application might require the user to re-authenticate before allowing any changes to personal information. However, post successful re-authentication, the above-mentioned observation remains true in such cases as well. Hence, traditional authentication systems cannot detect intruders after the authentication step is performed successfully.

The above, flaw becomes a critical security vulnerability for systems using only entry-point-based mechanisms. Note that the vulnerability does not only exist for high-security systems, but

also for personal computers in a general office environment. In fact, some would argue that given the (perceived) lesser criticality of computers in general office environment, they are even more vulnerable. As anyone can access the system resources if the initial user does not properly log out or the user leaves the workstation unattended to take a short break without logging out.

The requirement for the alternative authentication mechanism, then is to keep authenticating a user **continuously**, ideally without affecting user's productivity or altering the user's experience while interacting with the system.

In order to achieve this objective, we need to develop robust, reliable, and user-friendly methods for **continuous user authentication**. It is desirable that the resulting system has good usability by authenticating a user without his active cooperation.

Fortunately, continuous authentication has been an area of active research development in recent times. We delve deeper into the requirements of continuous authentication next before highlighting the most popular techniques used in practice for continuous authentication.

2.1 What is continuous authentication?

Continuous authentication (cAuth), as the name suggests, is an approach of user authentication, wherein, the current user of the system continuously (or at least very frequently). This is in contrast to the prevalent authentication approaches, which authenticate user only before logging them to the system. The motivation for continuous authentication is to a) make it harder for intruders to gain access to the system and b) also to detect a potential breach quickly so that mitigating actions to minimize losses can be initiated at the earliest.

While, the motive for adopting cAuth is sound, we note that, the prevalent authentication techniques such as asking for password, or answer to secret question cannot be leveraged for it. This is because asking users to frequently authenticate themselves will completely change the user's system interaction experience and also completely ruin their productivity.

Hence, continuous authentication can only use authentication techniques which will need very little (if any) inputs from the users. In fact, as we will see later in this paper, a key characteristic of the continuous authentication techniques used in practice is that they run in background, do not need any extra work from the user for authentication, and in a lot of cases, the user is completely oblivious to their existence.

To authenticate users with minimal interaction, cAuth techniques exploit the following insight:

Each person has multiple unique (discriminative) "attributes" which, an intruder will find extremely hard (if not impossible) to mimic.

Thus, practical cAuth approaches continuously (or with high frequency) authenticate the user over the duration of a session, by (transparently) monitoring and validating whether the current user is demonstrating the expected attributes. As mentioned before, this is in contrast to prevalent authentication systems, which rely on the fact that it shares a secret (password, OTP) with each user (and only with that user) and hence authenticates a user based on what they know or possess.

The earliest body of work involved looking into physiological biometrics features of users, such as authenticating the person based on matching the facial profile, fingerprints or voice recognition. Researchers have found that while such methods are highly accurate in identifying the user (and the attributes are hard to imitate by an intruder) are extremely intrusive (leak privacy) and costly (e.g., require an extra device like fingerprint sensor or the web-cam or microphone).

Another avenue of unique user attributes, which has gained significant focus in recent times, involves looking at the behavioral characteristics of the user. Examples of behavioral characteristics include, patterns of a user's interaction with his data input devices such as keyboard, mouse, touchscreen as well as his interaction with the UI of the application. These methods intend to passively learn discriminative features over a period of time in a user's "behavior" by applying machine learning techniques on user's interaction data. Available research literature has demonstrated that such discriminative attributes are present in a user's rate of typing keys on his keyboard, or in the speed and directionality of dragging the mouse etc. when interacting with a system.

Note that, techniques for observing and validating such features do not need any active participation from the users and also do not need any additional hardware. However, the authentication performance is significantly worse than physiological biometrics techniques (like fingerprint verification) and hence, the research recommends using a combination of the attributes to enhance the accuracy of authentication.

Yet another class of attributes corresponds to environment of the user which though can change over time but cannot change drastically. Examples of such attribute include the information regarding the type of device used, the geographical area from which the system is generally used etc. Clearly, such characteristics are expected to change over time and hence are not very discriminative. However, if monitored, they can alert against potential intrusions. For example, an user who had accessed the system from a location in USA (his usual location), cannot access it again from a location in Eastern Europe (potential intruder) at a gap of one hour and most probably signifies an intrusion into the system. Such environmental attributes are the easiest to monitor and also extremely hard to bypass by an intruder.

In the rest of this document, we will present a sampling of the popular techniques proposed in literature for carrying out continuous authentication. We will then present a modular framework, wherein we can easily plug and play various cAuth functionalities based on the demands.

2.2 Landscape of cAuth methods

As mentioned before, the goal of cAuth techniques is to identify an user for “who they are” instead of what they know, with minimal inputs from the user. In this section, we will give and overview of three most popular classes of techniques a) physiological biometrics b) environmental biometrics and c) behavioral biometrics, recommended for continuous authentication. We will also discuss the pros and cons of the techniques in terms of expected accuracy, ease of adoption, degree of privacy violation etc.

For each of the above approach, we explain the intuition of why it will work, explain what is required in terms of input and processing (output being same authenticated/unauthenticated). We will also comment on the utility of the discussed schemes for our envisaged application setting.

Physiological Biometrics

Physiological biometric based authentication refers to the science of identifying individuals by their physiological features. All biological features eligible to be used for biometric authentication must have the following necessary characteristics:

i. The feature should be present in every person
ii. Any two persons should be discriminable based on the feature
iii. The feature should not significantly change over a period of time and
iv. Two instances of the feature should be comparable using a metric.

The most common physiological features compared for biometric authentication include, face, iris, voice and fingerprint of a person. Also, in case of adequate hardware support, biometric authentication can be conducted without any active user participation. For example, the webcam on a computer can be used to carry out facial identification while working on the system. Similarly, the microphone on the computer can be used to record voice samples of the user for voice-based authentication. Further, biometric features of a user are hard to spoof or steal at scale by fraudsters. However, biometric techniques described above also suffer from a significant disadvantage. Specifically, most biometric techniques are intrusive on the privacy of the users as they capture a rich set of information of not only the user but also their surroundings etc.

To address the above concerns, researchers have designed and benchmarked multiple “soft” biometric techniques. Such techniques capture information which in itself might not be discriminative enough to uniquely identify the user but provide enough information to discriminate the current user from others. Examples of potential softbiometric features include, gender, ethnicity, color of eye/skin/hair, height, weight, SMT (scars, marks, and tattoos) etc. Note that the inability to unique identify a person make soft biometric feature-based authentication more privacy preserving.

However, soft biometric (like regular biometric) authentication still suffer from the disadvantage that they depend on appropriate hardware support (be it webcam, or microphone or fingerprint sensors). This significantly restricts the usability of a set of biometrics authentication techniques.

The next two classes of techniques do not need any hardware support and hence are easier to adopt.

Environmental

Environmental techniques create a model of the computational 'environment' of a user over the course of their system usage. Once the system has sufficient confidence in its model for a user's computation environment, it starts using it for cAuth. Specifically, the system compares whether the environment for the user matches the model's expectation. In case there is a deviation in some features of the environment, an appropriate alarm is raised. Examples of features monitored under computational environment include the geographical location from where the user logs in. Similarly, the time of day at which the user is active and the duration of

user's activity is also part of features monitored. As an example of the kind of anomalies that can trigger the environmental monitoring system to raise an alert, consider a situation wherein a user who had last logged in from Tokyo (Japan) at around 9:00 PM, logging in from New York (USA) after one hour. Clearly, it cannot be the same user who has logged in. An environmental cAuth system, if present, will trigger an alarm when it observes such deviations. Another example of such deviation can be when a user who has been using only a desktop to access the system starts to use a smartphone instead, or, the user who has been using a specific phone on a regular basis has logged in from a different phone. An environmental system will raise an alert. The remedial action might be to ask the user whether it is indeed him who is accessing the system via some independent communication channel, for example, sending an OTP message to user's smartphone that requires the user to acknowledge it in a prescribed manner. Furthermore, the type of operating system and the device used for accessing the application usage is also monitored.

Akin to the biometric and behavioral techniques, the environmental techniques do not need active user inputs for authenticating the user. Also, like the other two classes of techniques, the environmental techniques are not foolproof as well and can raise false alarms. For example, when a user accesses the service from a different geographical location or uses a new device for access, the environmental cAuth techniques will raise a false alarm. Hence, it is critical to ensure that the thresholds for alerting a user are set optimally, so that the user does not ignore them.

Behavioral Biometrics

Behavioral biometrics, as the name suggests tries to authenticate a user by his behavior when interacting with the system. Examples of specific "behavior" tracked for authentication include bias for specific mechanism when interacting with an application (for e.g., preference to type details instead of copy-pasting). device preferences (example, preferring mouse to keyboard to copy-paste) when interacting with the system, speed of typing using keyboard (slow/fast, bursty-steady), motion patterns when using mouse (straight line motion or circular motion), swiping motion patterns when using touchpad (or touch screen), press size & time on mouse, preferred font size etc. Also, physical and cognitive features such as hand tremors, left/right handedness etc. have been used for authenticating a user and are generally considered to be behavioral biometrics-based techniques.

The key insight behind the success of behavioral biometrics techniques is the observation that, there exist a set of normal behavioral patterns of an (authentic) user when interacting with the application, which are discriminative enough to distinguish from any other person. Also, the features have the property that they cannot be spoofed without a significant amount of training for each user's characteristics. Furthermore, the behavioral biometrics should not be enough to uniquely identify a user to preserve user privacy.

To intuitively understand the above observation, consider the case where a malicious entity has gained access to the login password (and other details of a user). The malicious entity, while using the system might cut-and-paste a lot of details on the system (as he has not memorized the details) instead of typing them. Similarly, an attacker who has hacked multiple systems might be proficient in using application shortcuts which the authentic user has never used. These behavioral patterns, if monitored for divergence, can easily detect fraudulent access to the system.

The most common form of behavioral biometrics involves creating models of the keystrokes of a user. A large body of research literature has shown that different users have distinct typing patterns, while some users are exceedingly proficient in using the keyboard, others are less so. Further, large variations can be observed in the

a) duration users hold down the keys,
b) how quickly they transition to the next key and
c) the frequency with which users edit text.

A machine learning model which can learn these characteristics over time can distinguish one user from any other with high accuracy (Figure 5).

Some estimates on such keystroke analysis for authentication have reported performance to the tune of 0.01% impostor pass rate (less than 1 successful attack out of 10K attempts) and a 4% false alarm rate for a user group with hundreds of participants.

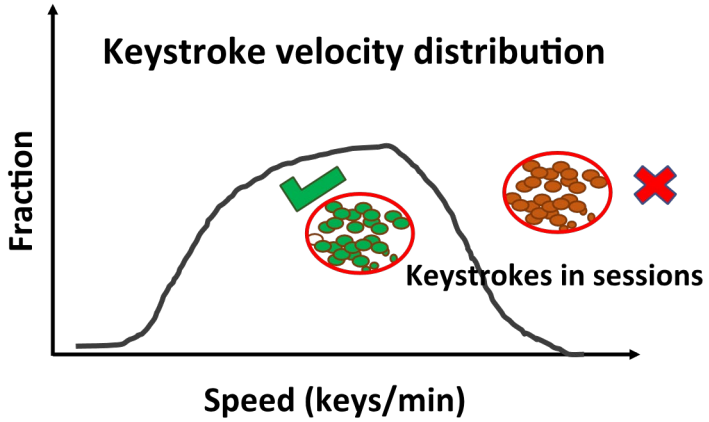

Figure 5: Detection anomalies in keystroke speed of users

Similarly, mouse motion-based authentication techniques learn models of typical patterns of the user when using a mouse (Figure 6).

A key advantage of behavioral biometrics lies in its ability to continuously map and monitor user's behavioral patterns for authentication in a manner a) oblivious to the user b) without any dependence of hardware throughout the length of the session.

The behavioral techniques fall between environmental techniques and physiological biometrics in terms of accuracy, physiological biometrics being the most accurate. However, behavioral techniques score significantly over physiological biometrics in terms of ease of deployability as they a) do not dependency on hardware availability (with attendant requirement of supporting multiple hardware specific version of data acquisition software at the client) and b) are not intrusive to the privacy of the user, as they do not collect data which can uniquely identify them.

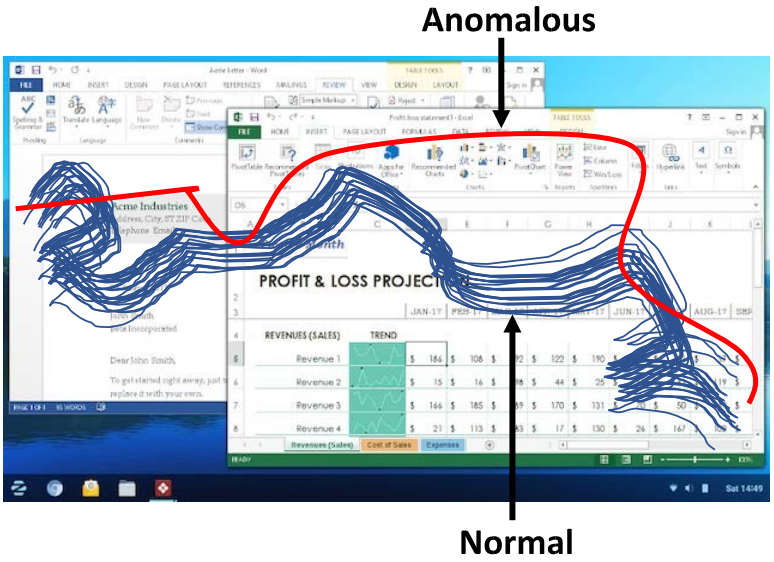

Figure 6: Detecting anomalous activities through mouse-tracking

3. Stratus Approach to Continuous Authentication

There is no panacea when it comes to user authentication, as we have highlighted over the course of our report each and every technique has its pros and cons. *Hence, creating authentication systems which deploy more than one technique, where each technique can negate the weakness of the other techniques, is the best engineering option.* In this section, we describe our plug-and-play solution to enable continuous authentication. The solution is API based, which is straightforward to integrate with any authentication service already in place.

The continuous authentication service will provide a much needed second line of defense against malicious entities who have gained access to a user's credentials by

a) passively monitoring of the actions undertaken by an entity (authenticated to be a known user) over the duration of a session,
b) ensuring that they match the known behavioral, environmental and physiological patterns of the known user and
c) triggering remedial actions on detecting significant divergence in an entities "behavior" to minimize potential devastation.

Our proposed solution also has the salient property that it does not alter user experience for the most part. Except for the initial installation and minor inputs, the continuous authentication system runs in background.

System model

Our system for continuous authentication consists of five main modules.

Client data logger: This component is tasked with collecting all the data necessary to carry out continuous authentication of the user. The user is prompted to install the logger on his local machine if the user plans to use remote access technologies such as Remote Desktop or VNC to access the system. Alternately, the client data logger can be a simple JavaScript plugin embedded in HTML web pages in case the access to the system is via a web-browser. The client logger will use standard encryption and SSL based connections to ensure that the user data remains safe. Further, compression and proactive cleaning of collected data will be undertaken to optimize resource footprint of the logging module.

Session manager: The session manager module is responsible for orchestrating the end-to-end authentication process for each active user session. The session manager will receive data from the client logger and then pass it on to a set of continuous authentication monitors. The module might also trigger a learning module in case a significant amount of time has evolved since the last training.

Profile training manager: The module learns the feature for each user using appropriate machine learning algorithms. Sample features learnt include, left/right handedness, spatial zone of operation, preference between keyboard or mouse, typing speed and characteristics etc. The training manager has two modules:

a) Trainer: Trains various continuous authentication models, on incoming user data. Note, the specific features trained have the property that they are universal (all users will have the feature) and collectable. Also, the selected features must be recognizable quickly with low overhead. Further the user's feature should not be highly dependent on other factors such as his access geography or environment. For example, if a predominantly used keyboard is old and does not register keystrokes correctly, the feature signature generated by the training algorithms will be biased (and hence not useful).

b) Model selector: Identifies the right set of features from a large library of behavioral, environmental and physiological features which are most discriminative for the user, while being relatively stable (i.e., do not change significantly over time). Selects models for each user that have low false reject rate (inability to identify the user) and false acceptance rate (identifying someone else as the user).

The training manager is triggered in the initial phases, and also periodically thereafter, to update the trained models for (natural) changes in user's feature signatures.

Session monitor: The session monitor performs the central task of applying learned user feature models on the incoming data from client logger to detect divergences. The session monitor takes a weighs each feature detector's observation (on divergence) with its reliability factor (function of false accept and reject rates). In case the feature detection modules decide that there is a significant divergence from the norm in a user's session, the module reports the incident to the Redressal manager.

Redressal manager: This module is tasked with triggering a pre-decided action based on the severity of divergence and the application/system/user account in question. The redressal actions can be as simple as logging the divergence in a log, sending emails to the account

holder and system administrator to asking the user to re-authenticate (using entry-point based mechanism) or suspending the user session.

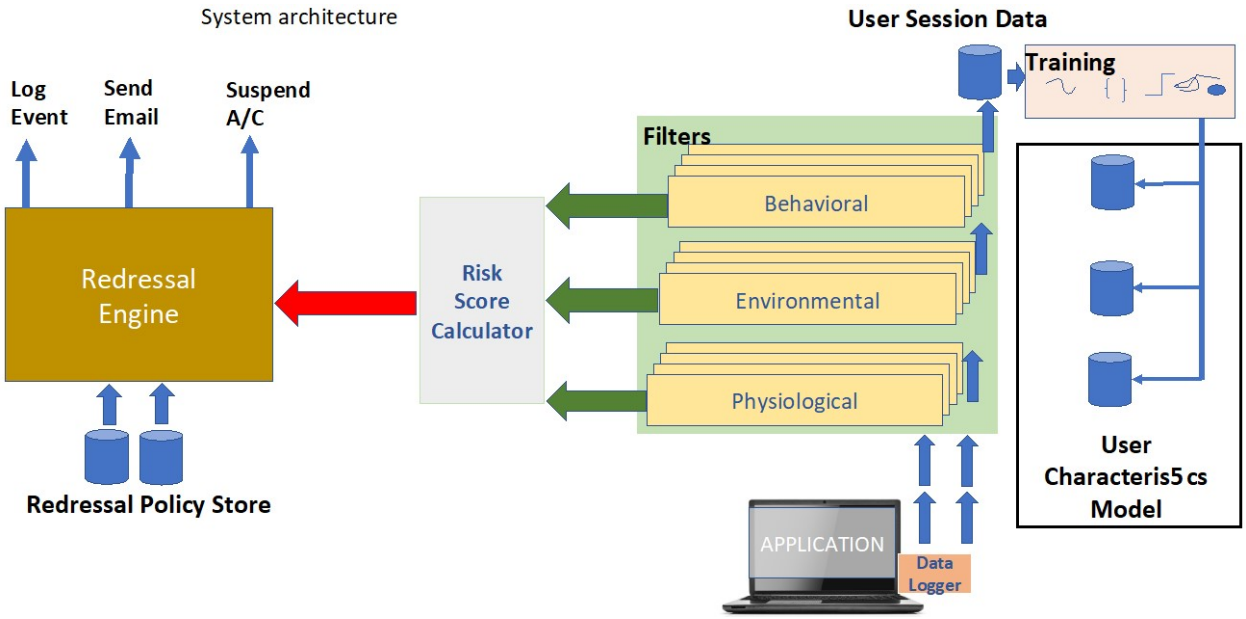

Figure 7: Stratus CAuth architecture and system overview

The overall architecture of the continuous authentication is detailed in Figure 7. As can be seen from the figure, client logger component is responsible for collecting the relevant data from the user and transmitting to the continuous authentication system. The incoming data is run through various filters, which try and detect any deviation from the norm. Any divergence is notified to the redressal engine. The redressal engine looks up the appropriate action based on the previously defined policies and then performs that action. Every new algorithm added will first train on the user's features and once trained will start monitoring the user's activities. The system will connect into an already present authentication system to initiate monitoring of client sessions. To this end, we will provide REST APIs which can be easily connected to any authentication system supporting API access.

To ensure that the system scales to a large number of users, the authentication service will be run as a set of inter-connected micro-services running in cloud such that each micro-service can be transparently replicated on demand.

This architecture provides API based support (and also a dashboard UI) to the system administrators to monitor the state of each user's session activity and also to look at prior activity and authentication event logs. Further, for each alert raised, the system will provide data explaining the reason for alert.

4. Summary

Authentication is a necessary requirement to ensure access to systems, and especially sensitive systems. Traditional methods of authentication tend to identify the user just once --- at the beginning of a session, or after a significant amount of time has elapsed since a period of activity. Such approaches do not address scenarios where an attacker may have stolen the credentials of a valid user and is therefore masquerading as an authorized user. Therefore, this document proposes the need and overall design of a continuous authentication system that provides additional safeguards and checks against attackers who might have found a way to authenticate against standard onetime authentication systems. This brings a whole new class of security properties that are not realizable otherwise. For example, the impact of phishing attacks is greatly mitigated because there is less ‘value’ for hackers to obtain one-time username/password information once continuous authentication is implemented.