Risk-based static authentication in web applications with behavioral biometrics and session context analytics

Jesus Solano, Luis Camacho, Alejandro Correa, Claudio Deiro, Javier Vargas, and Martín Ochoa

Cyxtera Technologies
first.last@cyxtera.com

Abstract. In order to improve the security of password-based authentication in web applications, it is a common industry practice to profile users based on their sessions context, such as IP ranges and Browser type. On the other hand, behavioral dynamics such as mouse and keyword features have been proposed in order to improve authentication, but have been shown most effective only in continuous authentication scenarios. In this paper we propose to combine both fingerprinting and behavioral dynamics (for mouse and keyboard) in order to increase security of login mechanisms. We do this by using machine learning techniques that aim at high accuracy, and only occasionally raise alarms for manual inspection. Our combined approach achieves an AUC of 0.957. We discuss the practicality of our approach in industrial contexts.

Keywords: Behavioral Dynamics, Static Authentication, Machine Learning

1 Introduction

With the increasing popularity of web services and cloud-based applications, we have also seen an increase on attacks to those platforms in the past decade. Several of those publicly known attacks have involved stealing of authentication credential to services (see for instance [10]). In addition to this, passwords are often the target of Malware (for instance banking related Malware such as Zeus [6] and its variants). So even if one would assume users are forced to select strong passwords (from the point of view of difficulty to guess), several studies have pointed out the challenges that password-based authentication pose for robust security [11, 2].

In order to mitigate the risk posed by attackers impersonating legitimate users by means of compromised or guessed credentials, many applications use mechanisms to detect anomalies by analyzing the connection features such as incoming IP, browser and OS type as read by HTTP headers, among others. Some of this context-based features have been also been discussed in the scientific literature [1]. However, there are some limitations of those defensive mechanisms,

for example, if the anomaly detection is too strict, there could be false positives that would harm user experience and thus hurt the webservices from a business perspective. On the other hand, if a careful attacker manages to bypass such context-related filters, for instance by manipulating HTTP parameters, using VPN services, or ultimately using a victim's machine [12], then such countermeasures fall short to provide better security.

Behavioral biometrics [17] have been proposed in the literature as a strategy to enhance the security of both web and desktop applications. They have shown to work with reasonable accuracy in the context of continuous authentication [19, 9], when the monitoring time of mouse and/or keyboard activity is long enough. In the context of static authentication, where interaction during log-in time with users is limited, such methods are less accurate and may be impractical [13], unless long static authentication interactions are assumed. However, in today's internet of services, many websites rely on third parties for security related functionality, that is integrated in the form of external javascript snippets. In domains handling highly sensitive data such as banking, those services are often only allowed to interact with a user's session during or before log-in, but not post-login. Therefore improving static risk-based authentication is a practical challenge.

Our proposed solution to address the above mentioned shortcomings of the individual context-based risk assessment techniques is to synergistically consider machine-learning based methods to detect anomalies in both context (browser type, country of origin of IP etc.) and behavioral features of a given user at login time. By considering a model that takes into account several features of browser, operating system, internet connection, connection times, keystroke and mouse dynamics one gains more confidence on the legitimacy of a given log-in attempt. Our model analyzes several previous log-in attempts in order to evaluate the risk of a new log-in attempt and is based on realistic data from customers of several major banks. In summary, the contributions made by this paper are:

- We propose a novel model that combines historical HTTP and behavioral data to detect anomalies during static authentication based part on real data from the banking domain, and part on a publicly available dataset.
- We evaluate the effectiveness of our model obtaining an AUC of 0.957 in our experimental setup.
- We discuss the practical applicability of our solution to realistic industrial scenarios based on our experience in the banking domain.

The rest of the paper is organized as follows: in Section 2 we recap some notions of context analytics and behavioral dynamics. In Section 3 we present our approach, and describe the data collected and the experimental design. In Section 4 we describe the experiments carried out in order to assess the effectiveness of the proposed approach. We discuss related work in Section 5 and conclude in Section 6.

2 Background and Attacker Model

User authentication has been traditionally based on passwords or passphrases which are meant to be secret. However, secrets can be stolen or guessed and, without further authentication mechanisms, attackers could impersonate a victim and steal sensitive information. To avoid this, the implementation of risk based authentication has allowed traditional authentication systems to increase confidence on a given user's identity by analyzing not only a pre-shared secret, but other features, such as device characteristics or user interaction which are expected to be unique [16, 14]. In the following we review some fundamental concepts related to device fingerprinting for authentication and behavioral biometrics.

2.1 Device Fingerprinting

Device fingerprinting is an identification technique used both for user tracking and authentication purposes. The main goal of this technique is to gather characteristics that uniquely identify a device. There are different ways to create this profile, the most reliable of them involves creating an identifier based on hardware signatures. However, acquiring these signatures requires high level privileges on the device, which is often hard to achieve.

Thanks to the popularization of the internet and the increased browser capabilities it is possible to also use statistical identification techniques using information gathered from the web browser [1, 18], such as browser history, installed plugins, supported mime types, user agents and also network information like headers, timestamps, origin IP and geolocation. Geolocation can be either collected using HTML5 or approximated from an IP address by using appropriate services. Gathering only browser information means these techniques identify web browsers and not necessarily devices or users. On the other hand, parameters such as HTTP request parameters are easy to spoof. Most recent techniques try to combine both hardware and statistical analysis gathering the information using the web browser capabilities, these techniques use HTML5 [3] and javascript APIs to measure the execution time of common javascript functions and the final result of rendering images as hardware signatures, these measurements are compared to a base line of time execution and rendering performed in a known hardware used as control[5, 15].

2.2 User Behavior Identification

Another popular risk-based authentication technique is behavioral analysis, based on mouse and keyboard dynamic statistics. The underlying idea of measuring user behavior is to turn human-computer interactions into numerical, categorical and temporal information. The standard interactions gathered for a behavioral model are key-strokes, mouse movements and mouse clicks. For instance, common features extracted from keyboard events are key pressed and key released

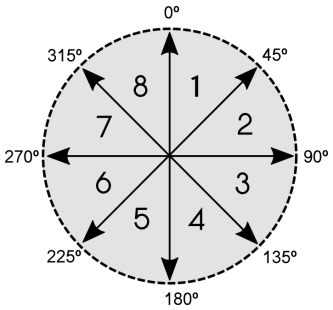


Fig. 1: Mouse directions segmentation.

events, together with their time-stamps. For mouse, cursor position, click coordinates and timestamps are commonly used [4]. Such features are processed and aggregated to profile user behavior. In this work, we will use aggregations such as the ones discussed in [7] and summarized in Tables 7 and 8. As shown in Figure 1 we used the suggested space segmentation in [7] to calculate mouse movement features.

These behavioral features give us information about very unique characteristics of each user such as how fast the user types, how many special keys the user uses, what is the proportion of use of mouse and keyboard, how long the user stops interacting before finishing an activity. The intuition behind this is that it must be easy to distinguish a user who uses mainly mouse from a user who uses mainly keyboard, also intuitively some physical conditions like hardware and user's ability with the peripheral devices makes these interactions more unique.

Behavioral models use machine learning to identify users by using these feature vectors. Notice that by recording one user's interaction in the same situation many times, it is expected that this user will interact with the computer similarly each time and also that it differs from the interactions gathered from other users.

2.3 TWOS Dataset

For the behavioral dynamics analysis, that we will illustrate in the following sections, it is important to have mouse and keyboard dynamics data, in order to evaluate our models. For this purpose, we have chosen to user data from a public data set known as The *Wolf Of SUTD (TWOS)* [4]. The data set contains realistic instances of insider threats based on a gamified competition. We have chosen this dataset since it contains both mouse and keyboard traces, among others. In [4], authors attempted to simulated user interactions in competing companies, inducing two types of behaviors (normal and malicious). The data set contains both mouse and keyboard data of 24 different users. We chose the TWOS dataset because of the large amount of behavioral patterns they recorded. In total, TWOS data set has more than 320 hours of mouse and keyboard dynamics. Data was continuously collected for volunteers during routine internet browsing activities in the context of a gamified experiment. The mouse

agent collected the position of the cursor in the screen, the action's timestamp, screen resolution, the mouse action, and user ID. The mouse actions involved in our analysis are mouse movement, button press/release and scroll. The keyboard agent logged all characters pressed by the users. The data set includes the timestamp of event, movement type (press/release), key and user ID. Both alphanumeric and special keys were recorded by the agent. Since the users typed potentially sensitive information the data is provided in an anonymized fashion. The keyboard was divided into zones to accomplish the anonymization. Figure 2 shows the mapping of the keyboard into three zones to enhance the privacy concerns.

$$\{\text{I,O,P,J,K,L,N,M}\} \rightarrow \text{RIGHT}$$
$$\{\text{R,T,Y,U,F,G,H,V,B}\} \rightarrow \text{CENTER}$$
$$\{\text{Q,W,E,A,S,D,Z,X,C}\} \rightarrow \text{LEFT}$$
$$\{\text{0,1,2,3,4,5,6,7,8,9}\} \rightarrow \text{DIGIT}$$

Fig. 2: Keyboard mapping layout to anonymize sensitive information.

2.4 Attacker model

We assume an attacker that has gained access to a victim's credentials to authenticate to a webservice (login and password). An attacker may also gain knowledge about, or try to guess, the context in which a victim uses a service: the time of the day in which a user usually connects, the operating system used, the browser used and IP range from which a victim connects. We assume that an attacker could employ one of the following strategies, or more than one in combination with others to attempt to impersonate a victim:

- *Simple attack:* The attacker connects to the webservice from a machine different than the victim's machine.
- *Context simulation attack:* The attacker connects to the webservice from a machine different than the victim's machine, but tries to replicate or guess the victim's access patterns: OS, Browser type, IP range and time of the day similar to victim's access patterns.
- *Physical access to victim's machine:* An attacker connects from the victim's machine, thereby having very faithfully replicated a victim's context, and attempts at impersonating the victim.

Note that we explicitly exclude from the attacker's capabilities that of recording and attempting to replicate a victim's behavioral dynamics (keyboard and mouse usage features). We believe that although this is an interesting attacker model, it is an extremely powerful one, and we leave its treatment to future work.

3 Approach

The goal of our approach is to overcome the shortcomings of the single risk assessment strategies (context-based analysis of HTTP connections and behavioral dynamics) by obtaining a single model that takes into account both strategies.

In Table 1 we summarize the effectiveness of various strategies in detecting the attacks discussed in the previous section, and also highlight the desired outcome of our approach. In essence, we expect a combined model to perform better in case of attacks, given that the combined model can recognize both changes in context and changes in behavior. Note that in this table we assume there is always impersonation (and thus always changes in behavior).

Approach	**Simple attack**	**Context simulation**	**Physical attack**
Context analytics	Effective	Partially effective	Ineffective
behavioral dynamics	Partially effective	Partially effective	Partially effective
Combination	Effective	More effective than single approaches.	Partially Effective

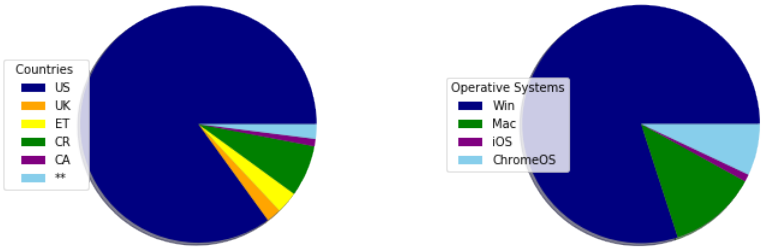
Table 1: Strategies vs. attack vectors

Moreover, we highlight the potential misclassfication of the various approaches in various scenarios in Table 2. Here, we summarize the expectation of the combination of both approaches in terms of reducing false positives. When a user uses a new device, one would expect its behavior to be similar in terms of keystrokes and mouse dynamics (although not exact). When he travels, it should remain very similar, those correcting possible false positives from the context analysis.

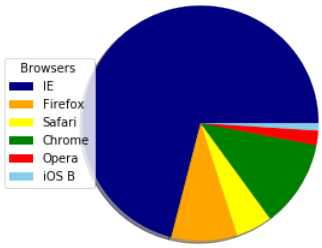
Approach	**New machine**	**User travels**
Context analytics	Likely FP	Likely FP
behavioral dynamics	Likely Accurate	Accurate
Combination	Likely Accurate	Accurate

Table 2: Strategies vs. benign context changes.

In the following we will summarize the models we used for for the single risk-based strategies, and describe how these models are used in combination to produce a combined risk-based assessment strategy. It is important to note that for the context analytics data we will assume that some users have a heterogenous access pattern (i.e. from multiple devices and locations, due to travel), as depicted in Fig. 3 for a user for which we have 338 access records. On the other hand, the time of activity considered for behavioral interaction reflects the average time of a password based log-in (which typically is a value between 25 and 30 seconds). Because of these challenges single models are not perfect within a global context attack, but can be used in synergy to produce a better model as we will show in the evaluation section.



(a) Connection per country distribution. (b) Connection per operative system distribution.



(c) Connection per browser distribution.

Fig. 3: Context of user with heterogeneous access patterns.

3.1 History-aware context analytics

In this subsection we describe the high-level construction of a session context model, based solely on session data obtained from HTTP requests. We assume users with complex access behaviors such as the ones depicted in Fig. 3, so we need to build a system that is good at detecting anomalies and potential attacks, but also it is somewhat flexible to certain changes in context that could be benign.

We assume a system that records usage statistics of the number of times that a user logs in, the day and time of the week at which the user logs in, what type of device and browser they are using, and the country and region from which the user is accessing. Currently, platform and browser data is obtained parsing the user agent, and geographic data is obtained parsing the IP address, information that can be obtained from network sessions corresponding to successful log-ins for a given user.

One of the challenges of building such a model is the fact that several categories considered are non-numerical (for instance a given browser version or operating system). This forces us to use a feature vector with connections statistics on each browser model version, each day of the week, each country and region etc. On the other hand, we must somehow assess the likelihood of a given connection context in order to decide whether a new connection is anomalous or

not. One way to do this is to simply compute the ratio of observations in a given field of a category divided by the sum of all the observations in that category.

For instance, let c the number of connections coming from a country K. Let C the total number of observations coming from all countries for a given users. Then the likelihood of an incoming connection from K could be computed as $\frac{c}{C}$. In order to assign a probability of 1 to the most likely event within a category, and a relative weight to other events in decreasing order from most likely to less likely, we normalize all values within a category as follows: order fields from most likely to less likely, define a new probability for a given field within a category as the sum of the probabilities for categories with probability equal or less to the one of the given field. For example, consider three countries with the following probabilities based on access frequency: US = $\frac{1}{2}$, UK = $\frac{1}{3}$, FR = $\frac{1}{6}$. The normalized probabilities would be: US = 1, UK = $\frac{3}{6}$ and FR = $\frac{1}{6}$.

Moreover, temporal categories (hours, days, etc.) are considered cyclical, because for instance events around midnight (before or after 24:00) should be considered relatively close to each other. Also, in order to smooth the notion of 'closeness' in discretized events such as frequencies of access in different hours of the day, we use a convolution as depicted in Figure 4. In this example, we have a distribution of discrete frequencies around the clock for a given user. In this scenario, 7PM is the hour of the day with most access. However this is close to say, 8PM, so it would be appropriate to consider an access at 8PM relatively normal for this context.

The feature vector for a session login attempt is formed using the normalized probability for each variable gathered from the HTTP request. For example, in the countries case above, a session which comes from US will have a value of 1 for variable country in the feature vector. To train the model we calculate the probability profiles for each user using the login history. Afterwards we evaluate a subset of new logins with the user probability profile and compute the feature vector for each visit. The feature vector is fed to a Random Forest model that assesses how anomalous the current event is. The impersonation records were synthesized comparing login events from one user to the history of another user. With this in mind, the model assesses the likelihood of an impersonation. Finally the statistics are updated, the idea being that the system will gradually adapt to permanent changes in the user profile.

3.2 Behavioral dynamics combining keystrokes and mouse activity

Both keyboard and mouse events are enough to describe a human-computer interactions and turn it into behavioral features. It is obvious that a regular user uses both at the same time. However, there is no simple way to merge both keyboard and mouse dynamics features. To describe a user behavior during a session we calculate the keyboard and mouse dynamics as behavioral features, as described in Table 7 and Table 8, using all the gathered events in one single session, where a session is defined as a time frame where the user is performing any activity on the computer. Once the keyboard and mouse dynamics are calculated, we combine both set of features, resulting in only one single vector of

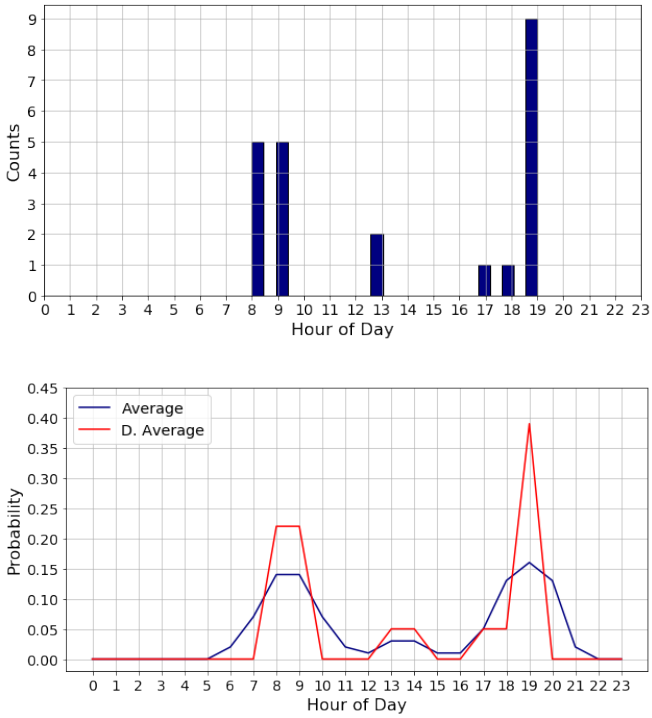


Fig. 4: Graphical representation of convolution used for temporal categories (e.g. hour of connection).

features per session. The combination of both set of features describes the use of keyboard and mouse dynamics in a single session. This process is repeated each time a new session is gathered. To compare a session behavior vector against the sessions in history we defined a maximum number of sessions to compare, in our experiment for each user we randomly chose between 10 and 30 sessions, this allows to test the algorithm performance with different history length. We calculated the history mean by using equation 1, as follows:

$$FeatureHistMean_j = \frac{\sum_J FeatureHist_j}{|J|} \tag{1}$$

Where $FeatureHistMean_j$ is defined as the the mean of one feature, $FeatureHist_j$ is the individual observation of the feature and J is the number of observations in the history. To compare the gathered session against the user sessions history we used equation 2.

$$FeatureDist_i = \frac{Feature_i - FeatureHistMean_i}{\sigma(FeatureHist_i)} \tag{2}$$

Where $FeatureHistMean_i$ is the calculated mean of the feature and $\sigma(FeatureHist_i)$ is the feature standard deviation. The resulting vectors of deviations give us the distance of a session compared to the history.

Using the previously described behavioral analysis process, we created a data set of sessions with labeled data. To create the positive labels we calculated for each user a base history. Then we calculated the behavioral features and deviation vectors. To create the negative labels for each user we randomly selected different users sessions and ran the behavioral analysis against the original user history. The resulting vectors feed a random forest algorithm to assess if a session is legitimate or not.

3.3 Overview of combined model

Assume we have a model to assess the risk of a session based on the browser context information, and another model to identify users by using behavioral patterns. As discussed in the introduction, there are however inherent limitations to each of the single models: context-based info of an incoming network (HTTP) connection cannot detect advanced impersonation attacks, whereas behavioral info is not accurate enough in short interactions such as log-ins. As a result we propose to enhance the risk-based authentication system's overall performance by combining the predictions of both models.

In principle, there are several ways to build such a combined meta-model, for instance by building a decision flowchart that takes the scores produced by the singles models and decides whether a given session should be considered suspicious or not. For simplicity's sake, in the following we propose to consider a parametric linear combination of the scores. In the evaluation section we will discuss an example instance of the parameters.

Let us to define $\hat{y}_c, \hat{y}_b \in [0, 1]$ as the prediction of context-based and behavioral model, respectively. We unify the models' prediction using a linear convex combination as we describe in equation 3.

$$\hat{y}_t \ = \alpha_c \cdot \hat{y}_c + \alpha_b \cdot \hat{y}_b \tag{3}$$

where $\alpha_c, \alpha_b \in [0, 1]$ are the coefficient parameters of each model. Note the coefficients must satisfy $\alpha_c + \alpha_b = 1$, because to be a meaningful prediction $\hat{y}_t \in [0, 1]$. As a result we expect, by considering a model that takes into account several learned features of browser context and behavioral dynamics, one gains more confidence on the legitimacy of a given log-in attempt.

Scalability of the combined model Note that the models obtained for the two risk assessment strategies involve training with a dataset of multiple users, however one model is generated that can be applied for each user (there is no need to build one model for each user). Therefore, the approach is designed to scale to millions of users, once the two respective models are trained.

4 Evaluation

In order to train and evaluate the performance of our proposed method we collect two sets of data. The behavioral data set, containing both mouse and keyboard

data, was retrieved from a public data set known as *The Wolf Of SUTD (TWOS)* [4] as we describe in subsection 2.3. Conversely, the context analytics data set was collected in house from banking web services. This data set contains information about context-based features for online banking log-in sessions. The context-based data set has ca. 13 million entries summarizing connection features when users perform a password-based authentication process. Within those features each entry has information of session timestamp, IP Address and user agent.

For the behavioral dynamics analysis, first we extract mouse traces and keystrokes from the TWOS dataset for all users. The next step is to correlate the mouse and keyboard data for the different sessions the users performed. For each user, a session is created by collecting all mouse entries within a time window before the final login click is observed. Afterwards, the keyboard data is correlated searching for all keystrokes in keyboard data set within the same time windows for the same user. With both mouse and keyboard session's information, we created the features described in section 3. Once the feature data sets are correlated we train the behavioral dynamics model. A random forest was trained in order to capture the behavioral patterns of each user. In order to test the performance of the model we split the full data set into 70% to train and the remaining data's entries to test model. The evaluation of the performance is done using standard classification evaluation measures. Using a confusion matrix, the following measures are calculated:

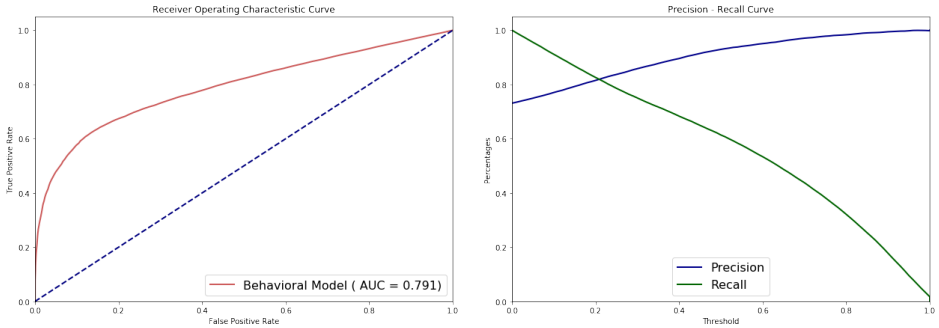
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{F1-Score} = \frac{2PR}{P + R}$$

where P, R, TP, FN, TN and FP are the precision score, recall score, the numbers of true positives, false negatives, true negatives and false positives, respectively. We define as positive the sessions with context simulation or impersonation attacks. The sessions without any attack are the negative ones. As we are facing a classification problem some performance metrics are dependent of the decision threshold λ. The λ parameter defines the minimum output probability a prediction must hold to be classified as a attack. In the table 3 we summarize the performance of the single behavioral dynamics model for different decision threshold.

Decision Threshold	**F1-Score**	**Precision**	**Accuracy**	**Recall**
0.3	0.798	0.862	0.725	0.743
0.5	0.735	0.932	0.680	0.607
0.7	0.593	0.972	0.572	0.427

Table 3: Single behavioral model performance for different classification thresholds(λ)



(a) ROC curve and AUC score. (b) Precision-Recall curve.

Fig. 5: Threshold dependent performance curves for the single model of behavioral dynamics.

From the table 3 we observe the model has a high precision independent of the decision threshold. However, the cost of high precision is the sensitivity of recall metric while the threshold is increasing. The problem addressed in this paper considers the high cost of false negative predictions because they generates a cascade of attacks which the system does not alert. For this reason we compare precision and recall curves figure 5b for the behavioral model to find out the threshold which minimizes the critical cases. Additionally, we show in figure 5a the model receiver operating characteristic(ROC) curve performance.

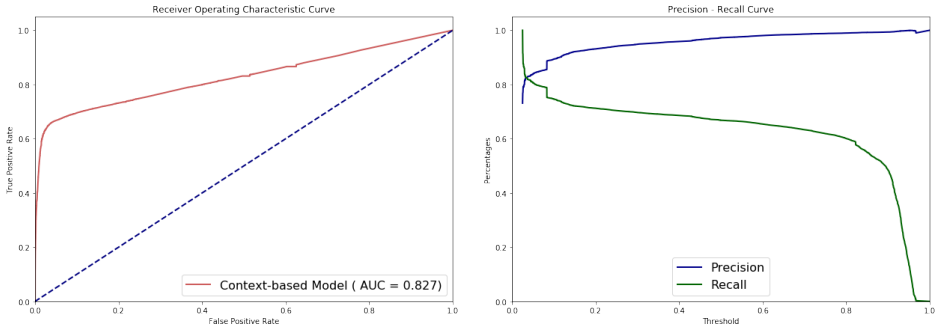
On the other hand, we train a model for the context-based information. Starting from the session timestamp, IP Address and user agent in the session start we calculate the convolutions and probability profiles described in section3. From the ca. 13 million of session log in attempts, we take the 30% of data to test the models performance and the remaining to train algorithm. The model used to predict the risk of a connection based on contextual information was a random forest. As we did with the behavioral model, we define as positive the sessions with context simulation or impersonation attacks. The sessions without any attack are the negative ones. Table 3 summarizes the performance of the single random forest model trained to alert attacks based on device context information.

The results in table 4 show the context-based model performs better than the behavioral model. Particularly, we observe that the model has a minor variation

Decision Threshold	**F1-Score**	**Precision**	**Accuracy**	**Recall**
0.3	0.803	0.948	0.750	0.697
0.5	0.792	0.972	0.743	0.668
0.7	0.771	0.986	0.725	0.633

Table 4: Single context-based model performance for different classification thresholds.

for the recall metric when the threshold is increased. We compare precision and recall curves figure 5b for the context-based model. Furthermore, we show in figure 5a the model area under the curve (AUC) metric.



(a) ROC curve and AUC score.

(b) Precision-Recall curve.

Fig. 6: Threshold dependent performance curves for the single model of context-based analysis of HTTP connections.

The AUC scores for both models are around 0.80, however, the precision and the recall metrics are not accurate enough for the problem we are addressing. For instance, the recall for context based model indicates a high rate of false negatives which in our context means a high rate of attacks are unnoticeable for the system. Moreover, F1-scores denote that each model separately has a similar performance when they try to detect a global attack. The issue is therefore that each model is not able to detect the counterpart attack: the context-based model will not detect changes in biometric features, and the behavioral model, on its own, will ignore changes in the connection context.

Therefore we develop a model that attempts to overcome the shortcomings of the single risk assessment strategies (context-based analysis of HTTP connections and behavioral dynamics individually) by proposing a single model that takes into account both strategies, as we describe in subsection 3.3. In order to test the combined model we perform a match between the session attempts in TWOS data and context-based data. First we find out the data set with less entries, for us TWOS data. Afterwards we split the TWOS data set into positive (impersonation attacks) and negative samples. As we balanced TWOS dataset before we train the behavioral model the behavioral data has as many positives as negatives entries. We take the positives entries of TWOS and split them into two sets. One of those subsets is matched with an equal number of random sessions from the context-based data set. In that vein, the remaining subset is matched with negatives samples from context-based data. The same process is performed for the positive entries in TWOS data set. As a result, the data set for the combined model is distributed as table 5 shows.

Label Behavioral	Label Context	Data Percentage	Combined Label
0	0	25%	0
0	1	25%	1
1	0	25%	1
1	1	25%	1

Table 5: Distribution of combined label to test the model that aggregates the predictions of single models.

To combine the predictions of single models we use equation 3 using $\alpha_c = 0.5$ and $\alpha_b = 0.5$. Those parameters for the convex combination were chosen following the intuition that both attacks are equally probable in our data set construction. We let as future work an analysis to define the best parameters. We show the results for combined model with a decision threshold of 0.3 and compare it with the single behavioral and context-based models in table 6.

Model	F1-Score	Precision	Accuracy	Recall
Behavioral	0.798	0.862	0.725	0.743
Context-based	0.803	0.948	0.750	0.697
Behavioral + Context-based	0.939	0.937	0.910	0.940

Table 6: Model performance comparison with a decision threshold of 0.3 for the three models we build to increase security in login attempts.

The results achieved with the combined model show an important enhancement in detection of attacks, as table 6 reveals. The high precision and recall values bring to light that the use of a combined model performs better in detecting attacks, given that the combined model can recognize both changes in context and changes in behavior. At the same time, an improved F1-Score and accuracy show that the overall classification was improved, thus also false positives caused by use of new devices or travel can be sometimes mitigated by using the information from the behavioral model.

Finally, we show in Figure 8 the model receiver operating characteristic (ROC) for all models we discuss in this work. As it is also evident from the AUC in this figure, a combined model has better performance than the individual models in the data set we have considered.

Scalability of the combined model We have measure the time it takes to evaluate a given session against the separate strategies, in order to asesss the scalability of the approach. These times were measured in a i7-7700hq processor (2.8 ghz), using a single core. For the context-based model, we obtain an execution time of 105 ms in average ($\pm$ 435 μs) per session. For the behavioral dynamics model we can classify a session within 106 ms ($\pm$ 263 μs) per session. Since the

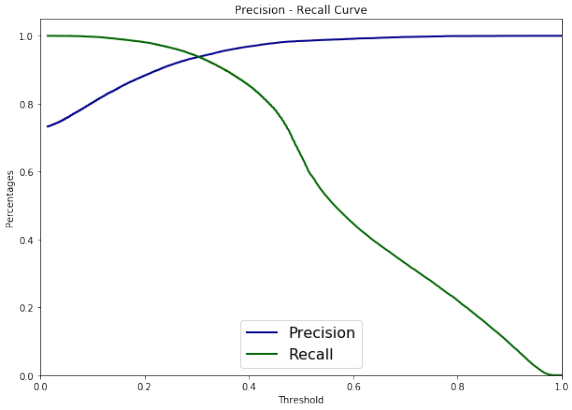


Fig. 7: Precision - Recall curves for the combined risk assessment model (i.e. context-based analysis of HTTP connections and behavioral dynamics)

combination of the scores is a simple linear combination, the risk-assessment can be completed within 1 second per each session.

4.1 Use in industrial scenarios

There is no single approach for including an anomaly detection system such as the one discussed in this paper in an operational workflow, nor a one-size-fits-all choice of parameters. Smaller entities – especially if not experiencing a high level of fraud – may want to handle assessments manually. In this case human operators in a SOC receive an alert and react to it. Actions may include blocking the user account, or contacting the user. Aggregated data can also be used to drive the decision process towards more sophisticated, and effective, solutions.

In this scenario where all alerts are handled by a human operator it is mandatory that the alert rate is reasonably small. While it is impossible to define a generic threshold, 1% may be a useful benchmark. As the user base, or the fraud level, grows the institution may decide to integrate the assessments in the application work-flow. If this is done with hard-coded rules then a low rate of false positives is critical, as each alert will generate an action and therefore an expense.

Bigger, or more security-aware, institutions will probably feed the assessment generated from this module to additional systems. While in practice there is no such clear distinction – one single system can often play the two roles – we can typify this systems in two categories:

- *Dynamic authentication systems.* Based on the assessment and other factors such as the user's risk profile and the money at stake the system can decide if additional authentication factors are to be requested to the user, or if access has to be blocked altogether, once or permanently.
- *Transaction anomaly detection systems*, that can include the information related to the transaction to decide if it can be approved, denied or further

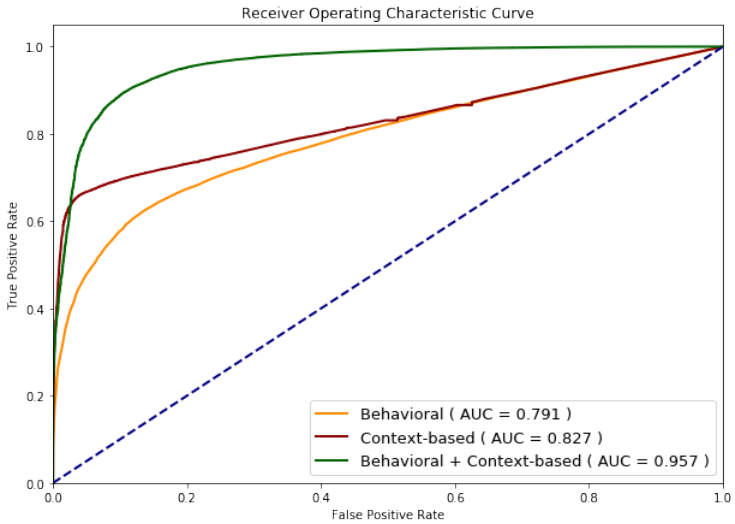


Fig. 8: ROC curves and AUC scores for the single risk assessment strategies(context-based analysis of HTTP connections and behavioral dynamics) and the model which combines both strategies.

action should be requested, including sending the transaction to a SOC for further human analysis.

In this last scenario a higher rate of false positives is acceptable, as the alerts will be filtered using independent criteria. Furthermore, a numeric assessment is preferable with respect to a binary value, letting the institution fine tune, possibly in real time, how to react to the assessment.

In the context of web-application static authentication, we believe that optimizing the choice of parameters in the model to minimize false negatives (i.e. undetected attacks), is acceptable if in those cases, users can be prompted for a 2-Factor-Authentication such as an OTP sent to their mobile phones. In our model, setting the threshold between 0.2 and 0.25 will yield between 35% and 24.6% false-positive rate against $1,9\%$ and 3.5% false-negative rate respectively. This means roughly one out four users is prompted for 2-FA, whereas between one out 30 to 50 attacks goes undetected. Note that these number hold for our experiments, where 75% of the data consists of attacks, in practice attacks are much less common.

For very sensitive customers, further manual action can be taken depending on the transactions performed in the application. For instance, in the banking domain, further filters depending on transaction amounts can be applied, given a suspicion on context and behavior.

4.2 Discussion and limitations

We have shown that in principle the combination of both risk-based authentication strategies indeed improves the performance of the single models in isolation. There are a number of limitations to our evaluation. First, the data from the HTTP contextual model and the behavioral model do not belong to the same users. Although in principle there should be no strong correlation between context and behavior, a more accurate model could be built if variations in behavior from the same user across devices are taken into account.

On the other hand, experiments were built under the assumption that the different scenarios combinations (between contextual and behavioral attacks) were equally likely. In practice, attacks are rare, and this aspect should be considered in future work. Last, we have considered behavioral data that has been adapted to simulate static authentication, but that in reality may belong to other activities in the context of the competition where it was gathered. In future work, we plan to consider ad-hoc gathered data from user log-ins. To the best of our knowledge, there is no public database containing both mouse and keyboard data for static authentication, although there are some datasets containing either of them.

Last, regarding Tables 1 and 2, although we have shown implicitly that the overall combined model performs better in both senses (FP and FN), we have not evaluated concretely the single combinations pointed out in the tables, which is also left for future work.

5 Related work

Risk-based authentication has seen popularity in web applications due to the limitations of password authentication. In [2] Bonneau et al. give a historical overview of the introduction of risk-based authentication in practical systems in order to complement password-based authentication. In [1] Alaca et al. classify and survey several device fingerprinting mechanisms that can be used as the basis for authentication, and discuss different ways in which authentication can be complemented by them. In [8] Misbahuddin et al. study the application of machine learning techniques for risk-based authentication using HTTP and network patterns, in a similar spirit of our technique, but do not take into account behavioral biometric patterns from mouse and keyboard, that as we have shown, improve the accuracy of risk-based authentication.

On the other hand, there are several works exploring applications of behavioral biometrics for static and continuous authentication. In the general context of desktop based applications, Mondal et al. [9] have studied the combination of keyboard and mouse for continuous authentication. Different from them, we focus on static authentication for web applications. In [13] Shen et al. study the applicability of mouse-based analytics for static authentication and conclude that longer than typical log-in interactions would be necessary in order to obtain high accuracy in such models. Traore et al. [16] explore the combination

of both mouse and keyboard for risk-based authentication in web applications, however they assume the behavior monitor to be in the application after log-in as well (continuous authentication), and obtain an equal error rate of around 8% (even when considering full web sessions).

To the best of our knowledge the combination of traditional risk-based authentication based on HTTP and network information and behavioral biometrics for static (log-in time) authentication, as proposed in this work, has not been discussed in the literature.

6 Conclusions

The results of our proposed method demonstrates that device identification and behavioral analytics are complementary methods of risk measurement thus by combining both of them, efficacy and performance are never lower than single method approach. Moreover, our approach seams to be more resilient to changes, for instance when a user changes his/her device, an only device identification approach will alert event though there is no attack and an only behavioral approach will not notice the change at all.

In this work we also have shown that, by combining both device identification and behavioral identification risk assessment methods during login time, static web authentication performance can be enhanced by detecting single and mixed attack models with higher or equal accuracy in each case. This also makes web authentication systems more robust and may give the user a better security experience.

We have also discussed the practical applicability of our solution in industrial scenarios. In the future, we plan to consider a more powerful attacker model that is aware of a behavioral risk assessment component and attempts to bypass it, as well as reproducing this experiments on novel datasets that collect both session information and behavioral dynamics simultaneously.

References

1. Alaca, F., Van Oorschot, P.C.: Device fingerprinting for augmenting web authentication: classification and analysis of methods. In: Proceedings of the 32nd Annual Conference on Computer Security Applications. pp. 289–301. ACM (2016)
2. Bonneau, J., Herley, C., Stajano, F.M., et al.: Passwords and the evolution of imperfect authentication. Communications of the ACM (2014)
3. Gabi Nakibly, Gilad Shelef, S.Y.: Hardware fingerprinting using html5 pp. 1–13 (2015)
4. Harilal, A., Toffalini, F., Homoliak, I., Castellanos, J., Guarnizo, J., Mondal, S., Ochoa, M.: The wolf of SUTD (twos): A dataset of malicious insider threat behavior based on a gamified competition. Journal of Wireless Mobile Networks **9** (03 2018). https://doi.org/10.22667/JOWUA.2018.03.31.054
5. Iskander Sanchez-Rola, I.S., Balzarotti, D.: Clock around the clock: Time-based device fingerprinting pp. 1–13 (2018)

6. Kaspersky: Zeus malware. Online (2019), https://usa.kaspersky.com/resource-center/threats/zeus-virus
7. Kyle O. Bailey, J.S.O., Peterson, G.L.: User identification and authentication using multi-modal behavioral biometrics. Computers & Security (2014)
8. Misbahuddin, M., Bindhumadhava, B.S., Dheeptha, B.: Design of a risk based authentication system using machine learning techniques. In: 2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation. pp. 1–6 (2017)
9. Mondal, S., Bours, P.: Combining keystroke and mouse dynamics for continuous user authentication and identification. In: 2016 IEEE International Conference on Identity, Security and Behavior Analysis (ISBA). pp. 1–8. IEEE (2016)
10. Newman, L.: Hacker lexicon: What is credential stuffing? Wired Magazine (2019), https://www.wired.com/story/what-is-credential-stuffing/
11. Perrig, A.: Shortcomings of password-based authentication. In: 9th USENIX Security Symposium. ACM. vol. 130 (2000)
12. Salem, M.B., Hershkop, S., Stolfo, S.J.: A survey of insider attack detection research. In: Insider Attack and Cyber Security, pp. 69–90. Springer (2008)
13. Shen, C., Cai, Z., Guan, X., Wang, J.: On the effectiveness and applicability of mouse dynamics biometric for static authentication: A benchmark study. In: 2012 5th IAPR International Conference on Biometrics (ICB) (2012)
14. Swati Gurav, R.G., Mhangore, S.: Combining keystroke and mouse dynamics for user authentication. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS) (2017)
15. Tadayoshi Kohno, Andre Broido, k.c.: Remote physical device fingerprinting pp. 1–13 (2004)
16. Traore, I., Woungang, I., Obaidat, M.S., Nakkabi, Y., Lai, I.: Combining mouse and keystroke dynamics biometrics for risk-based authentication in web environments. In: 2012 Fourth International Conference on Digital Home (2012)
17. Yampolskiy, R.V., Govindaraju, V.: Behavioural biometrics: a survey and classification. International Journal of Biometrics **1**(1), 81–113 (2008)
18. Yinzhi Cao, S.L., WijmansâĂă, E.: (cross-)browser fingerprinting via os and hardware level features pp. 1–15 (2017)
19. Zheng, N., Paloski, A., Wang, H.: An efficient user verification system via mouse movements. In: Proceedings of the 18th ACM conference on Computer and communications security. pp. 139–150. ACM (2011)

7 Appendix

Variable Name	Variable Type	Description
Average Key Press	Float	General time average of key press event.
Standard Deviation Key Press	Float	General time standard Deviation of key press event.
Median Key Press	Float	General time median of key press event.
Count Key Press	Float	General count of pressed keys.
Average Key Press per Second	Float	Average time of key press event per second.
Average Hold key RIGHT	Float	Average hold time of a right side key.
Standard Deviation Hold key RIGHT	Float	Standard deviation of hold time of a right side key.
Median Hold key RIGHT	Float	Median hold time of a right side key.
Count Hold key RIGHT	Float	Count of pressed keys in the right side.
Average Hold key LEFT	Float	Average hold time of a left side key.
Standard Deviation Hold key LEFT	Float	Standard deviation of hold time of a left side key.
Median Hold key LEFT	Float	Median hold time of a left side key.
Count Hold key LEFT	Float	Count of pressed keys in the left side.
Average Hold key CENTER	Float	Average hold time of a center side key.
Standard Deviation Hold key CENTER	Float	Standard deviation of hold time of a center side key.
Median Hold key CENTER	Float	Median hold time of a center side key.
Count Hold key CENTER	Float	Count of pressed keys in the center side.
Average Hold key ctrl left	Float	Average hold time of a left control key.
Standard Deviation Hold ctrl left	Float	Standard deviation of hold time of a left control key.

Table 7: List of behavioral features from keyboard dynamics

Variable Name	Var Type	Description
Average Speed Dir1	Float	Average speed in px per second in Direction 1.
Average Speed Dir2	Float	Average speed in px per second in Direction 2.
Average Speed Dir3	Float	Average speed in px per second in Direction 3.
Average Speed Dir4	Float	Average speed in px per second in Direction 4.
Average Speed Dir5	Float	Average speed in px per second in Direction 5.
Average Speed Dir6	Float	Average speed in px per second in Direction 6.
Average Speed Dir7	Float	Average speed in px per second in Direction 7.
Average Speed Dir8	Float	Average speed in px per second in Direction 8.
Percentage in Dir1	Float	Average speed in px per second in Direction 1.
Percentage in Dir2	Float	Average speed in px per second in Direction 2.
Percentage in Dir3	Float	Average speed in px per second in Direction 3.
Percentage in Dir4	Float	Average speed in px per second in Direction 4.
Percentage in Dir5	Float	Average speed in px per second in Direction 5.
Percentage in Dir6	Float	Average speed in px per second in Direction 6.
Percentage in Dir7	Float	Average speed in px per second in Direction 7.
Percentage in Dir8	Float	Average speed in px per second in Direction 8.

Table 8: List of behavioral features from mouse dynamics