LEARNING ON A BUDGET FOR USER AUTHENTICATION ON MOBILE DEVICES

Bojan Kolosnjaji ⋆ *Antonia Hüfner* ⋆ *Claudia Eckert* ⋆ *Apostolis Zarras* †

⋆ Technical University of Munich
† Maastricht University

ABSTRACT

Since the amount of sensitive information stored on smartphones expands with the growth of their popularity, the need for reliable and usable authentication on these devices increases. Constant reauthentication requests of standard methods, such as PINs, passwords, and swipe patterns, annoy many users who prefer to sacrifice the security of their mobile devices for the quest for maximum usability. An alternative to this approach is sensor-based authentication, where we fingerprint the user interaction by processing signals from sensors such as touchscreen or accelerometer. In this paper, we construct a budgeted online version of One-Class Support Vector Machine (OC-SVM) to maintain authentication performance while limiting the model size and evaluate the performance compared to an unconstrained model. The results of our experiments reveal that it is possible to correctly detect invalid smartphone users in a constrained environment using our budgeted learning methodology.

***Index Terms*—** Machine Learning, User Authentication, Security

1. INTRODUCTION

Smartphone functionality has been constantly growing in recent years, making them unavoidable tools in many aspects of our everyday life. However, this increases the amount of sensitive information stored on the mobile devices. While smartphones are easy to carry around, they are also prone to getting lost or stolen. Furthermore, once a smartphone gets unlocked, it often enables the attacker to have direct access to e-mail accounts, private files, or credit card information.

Existing authentication methods on smartphones include PINs, swipe patterns, picture codes, and fingerprint sensors. To protect the phone, entry authentication requires the user to reenter his credentials when using the phone after a predefined time interval. This is considered inconvenient, especially in cases of short lasting tasks such as answering messages. As a result, users tend to prefer weak passwords, choose highly increased lock periods, or leave their phones completely unlocked. Furthermore, entry authentication can be bypassed easily, leaving the phone content unprotected. While PINs and swipe patterns are vulnerable to both smudge attacks [1] and shoulder surfing [2], fingerprint sensors may be tricked by fingerprint copies [3].

An appealing approach to address the drawbacks of traditional methods is continuous sensor-based authentication. While using the mobile device that includes this type of authentication, the validity of the user gets continuously assessed in the background based on his current behavior. The phone gets locked on suspicious behavior, forcing the user to reauthenticate himself by means of other authentication methods such as a PIN or a password. Continuous authentication should be able to reliably protect the phone content, as user behavior has been proven difficult to mimic [4]. As opposed to entry authentication methods, continuous authentication does not rely on a secret being known, as this secret can still be gained by an attacker.

Behavior-based authentication is essentially dependent on a discriminative user model. Many works have investigated the challenge of modeling user behavior with machine learning. Apart from attempts based on device use habits [5], characteristic short-term user interactions are utilized to build a user profile. User interactions have mostly been modeled based on keystroke dynamics and swipe patterns. These gestures have been characterized in spatial and temporal terms [6–8]. Additionally, there exist works that, utilizing the sensor feedback, improve their user model [9, 10]. Further research efforts attempt to refine their behavioral model by including the application context [11].

The previous works have mostly been based on batch machine learning models which remain the same after an initial training phase. However, user behavior changes over time [8]. A good user model should therefore adapt to these changes, in order to not reject valid users from using their devices. In this paper, we aim to solve this problem by utilizing online machine learning to adapt the user model continuously to recent behavior. Yet, online machine learning suffers from model growth over time. To address the resource-constrained environment on mobile devices, we therefore use a methodology called *online learning on a budget*, thereby limiting the memory footprint of the authentication framework. Overall, our goal is to show that an online anomaly detection approach can remain efficient in the authentication scenario while dealing with a memory-constrained environment.

We leverage a well-known method: One-Class Support

Vector Machine [12] and adapt it to our scenario of learning on a budget. We design our adaptation to this method based on related work in online learning for classification [13–15]. In order to test our approach, we plan and execute an experiment with 28 live subjects performing typical user activities on smartphones in order to gather their behavioral properties. Using the traces of our subjects' behavior we train and evaluate our method of sensor-based continuous authentication.

In summary, we make the following main contributions:

- We combine spatio-temporal features from multiple sensors to model user behavior for mobile devices.
- We create a budgeted version of One-Class SVM method that maintains model accuracy while having a smaller memory footprint.
- We use this budgeted learning methodology in an online manner to adapt the user model continuously, achieving almost 90% classification accuracy for most of users.

2. METHODOLOGY

In our methodology we consider the situation where we gather sensor data and train our model incrementally using a valid user and then, in the test time, have both a valid user and an attacker trying to use the device. Both the attacker and the valid user leave a trace of their activity in the form of touchscreen interaction, accelerometer, and rotation sensor values. The goal of our system is to detect the attacker's behavior as a significant deviation from the model based on the valid user, while letting the valid user use the device without unnecessary interruption. Therefore, we consider our problem as an example of unsupervised anomaly detection problems.

2.1. Experiment Design

To gather user-specific data, we design a two-part experiment focusing on different typical activities on the smartphone.

Requested Tasks in our Mobile App. We request users to perform a series of tasks so we can train our model. A potential future app, delivered directly from the smartphone vendor, could offer the option for users to initially pretrain their smartphones with their behavior, if they wish to enable the continuous authentication feature. Another possibility is for the user to perform the tasks presented in our app under the hood. This means that the phone does not need to execute an application, where it requests input from the user, but it can stealthily collect data and train the models. The tasks that we use for collecting continuous authentication data in our app are the following: *Image Finding*, *Text Reading*, *Zooming*, *Message Composition*, *Entering a PIN* and *Phone Call*. We design these tasks taking into account the most often performed activities on a smartphone. Each task is repeated ten times per participant, which lowers the impact of sudden change of behavior because of external conditions (e.g., stress).

Tasks with Built-in Apps. We execute the next series of tasks using the apps provided by the mobile devices themselves. This makes the behavioral traces of users more natural, as they regularly use this kind of apps and can perform the tasks more smoothly. The tasks in this part were: *Browsing*, where a user needs to find answers to three questions using Google and Wikipedia, and *Taking a Selfie* using the smartphone's camera.

2.2. Data Collection

We used the *Samsung Galaxy S4* smartphones with Android 5.0 to execute the above described experiments with 28 participants in order to capture the discriminative features in user activity. The Android API delivers a sequence of motion events to the currently active application each time a user touches the screen. The touch sequence is started by a down event followed by a variable number of move events and an up event closing the sequence. Each motion event has an event ID, followed by its timestamp and its x and y center coordinate, as well as major and minor axis of the ellipse making up the touch contact. Apart from the standard motion event data, for keyboard typing events we can record the keycode of each key hit. For security reasons our version of Android does not allow to capture keyboard input of the standard keyboard. For this purpose we develop a keyboard that resembled the Samsung keyboard, and run it as an Android service. Users are instructed to choose the custom keyboard when opening the app such that no other keyboard would accidentally be used.

In addition to touch data, sensor data from built-in accelerometer and gyroscope (rotation sensor) are harvested. Data are polled at a rate of five times a second. The accelerometer measures acceleration in *m/s2*, while the gyroscope measures rate of rotation in *rad/s* along x, y and z axis of the phone. Gathering sensor data is done by running an Android service, same as in case of the custom keyboard.

Android only allows capturing touch events from within a custom application. Therefore, it is not possible to gather motion events from the Android API while using standard applications for the second part of the experiment. Nevertheless, it is possible to directly read from the input device driver under /dev/input/eventX, where X is a number identifying the input device.

2.3. Feature Engineering and Selection

Based on the ideas from related work, we capture a large range of spatio-temporal features to determine which subset is the most useful. Initially there are 59 features from swipe gestures and 51 features from keystrokes. The swipe feature

set contains data that characterize the trajectory of the swipe motion, the velocity of movement, and size of the trace. On the other hand, the keystroke features contain the similar velocity and stroke size patterns as well as keycode and offsets to the key center. Our feature set is based on the works of Velten et al [16], Frank et al. [17], and Buschek et al. [6]. Out of this feature set we select most relevant ones based on the sequential feature selection procedure of Ruckstiess et al. [18].

2.4. Budgeted Anomaly Detection

Our goal is to detect invalid users based on the gathered data. Therefore we leverage a method of anomaly detection called One-Class Support Vector Machine (OC-SVM) [12]. OC-SVM is an anomaly detection method that enables us to determine a region where most of the data points lie, giving them the label `+1`. On the other hand, the rest of the points are considered anomalies (labeled as `−1`). We optimize the margin between these two regions of points in order to have the most accurate boundary. In the primal form this method can be represented with the following formula:

$$\min \frac{1}{2}w^2 + \frac{1}{\nu N}\sum_i \xi_i - \rho \tag{1}$$

subject to:

$$(w\Phi(x_i)) \geq \rho - \xi_i, \xi_i > 0 \tag{2}$$

The parameter w defines the position and orientation of the classification boundary, while the value of ξ is tuned to increase robustness to possible noise in the labeling process. The value of N is the size of the sample set and the constant ν is used for tuning the percentage of outliers.

In order to easier define the budget saving strategy, we use a dual form of the decision function:

$$f(x) = sgn(\sum_i \alpha_i K(x_i, x) - \rho) \tag{3}$$

where the value of ρ is tuned to get the best detection performance on a certain data distribution. The dual form enables us to use a kernelized version of the distance value ($K(x_i, x)$). This means that we can freely select the distance measure between points (K). If we use a linear distance measure, we can only optimize a linear decision boundary. Since our samples are not linearly separable, we use the Gaussian RBF kernel, in order to have a more precise separation of points.

When using a dual form of One-Class SVM, we solve the following optimization problem:

$$min_\alpha \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) \tag{4}$$

subject to:

$$0 \leq \alpha_i \leq \frac{1}{vl}, \sum_i \alpha_i = 1 \tag{5}$$

This is a quadratic problem with linear constraints. Instead of solving this problem with a usual quadratic programming procedure, we use gradient descent, where the gradient looks like the following:

$$\frac{\partial g(x)}{\partial \alpha_i} = \frac{1}{2}K_i\alpha_i \tag{6}$$

For each new point that arrives, we execute 100 gradient descent steps to update our model, with learning rate $\lambda = 0.0001$. After executing the model update, we execute the maintenance to make sure that the parameter set does not increase beyond the budget limits. There are multiple possible strategies for this maintenance. In the work of Wang et al. [19] three strategies are defined: deletion, projection, and merging. After testing all of them, we decided to use the deletion method, as it provides approximately the same results as other methods, but with faster execution, which is very important in online learning. In the deletion method, we simply remove smallest values of α_i in order to fit to the budget limit.

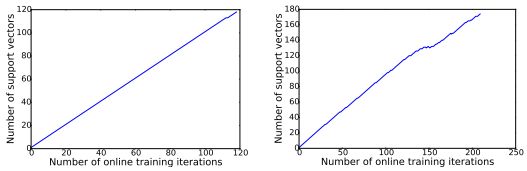
Since we have two broad feature sets (i.e., swipe gestures and keyboard writing), we build two separate models using the previously described procedures. For each new relevant event we first detect the type of event based on the values that we can read from the sensors and then run the authentication process using the appropriate model.

3. EVALUATION

We tested our procedure using the data gathered from our experiments, with a type of crossvalidation. All the gathered data from querying touch screen, keyboard, acceleration, and gyroscope sensors were turned into feature points and the order of actions from the experiment was randomized. The randomized sets of points were divided into training (70%) and test sets (30%). The results had been averaged over 5 runs to make them more reliable. At first we do this test separately for swipe and keyboard gestures in order to investigate the model behavior for the two sets of features. We test the model performance in an unconstrained scenario, followed by tests with different budget size limits. Afterwards, we execute a joint test to simulate the real-world behavior and analyze the model performance for different test users.

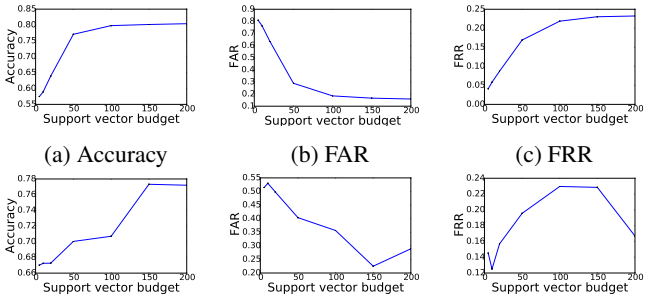
3.1. Model Growth in an Unconstrained Scenario

First, we show a graph of the model growth suitable to our scenario of using One-Class Support Vector Machine. Figure 1 shows that in an unconstrained scenario the model grows almost linearly. While we can only show growth in the support vector set for the limited sequence of training samples we used in crossvalidation, the model would keep receiving a continuous stream of samples in reality, thereby growing unlimitedly. This clearly shows demand for a limit on model size as we would otherwise gradually reserve more



(a) Swipe model (b) Keyboard model

Fig. 1. Increase in the model size for increase in training samples presented to the model for keyboard features.



(a) Accuracy (b) FAR (c) FRR

(d) Accuracy (e) FAR (f) FRR

Fig. 2. Results for different budget constraints with swipe (a,b,c) and keyboard (d,e,f) features

and more memory on users' phones. Not only is this inconvenient in terms of memory footprint, but it would also build up a very complex model, thereby increasing computational cost, which might in turn yield a reduced operating speed. Besides, old support vectors kept in the model may not be reflecting current user behavior. While they unnecessarily reserve memory, they may also degrade performance of the model.

3.2. Budget Size

The next group of plots, displayed on Figure 2 shows the change of results for swipe features with the increase in the available support vector budget. On the first plot the accuracy grows very fast with the number of available support vectors, achieving 95% of the maximum accuracy with only a half of maximum support vectors as a limitation. This shows that we can place a demanding limit on the size of the model and still achieve high accuracy. Furthermore, the next plot shows fast decrease in false acceptance rate with the number of available support vectors. This means that even for a small budget we can get an authentication system where attacks will have a high detection rate. On the other hand, false rejection rate is actually growing with the number of available support vectors. This means that in terms of this particular criteria, the performance does not improve with the available budgets. Possible reason for that is that more complex model tends to have high bias towards the training set. This means that more initial training data may be necessary for more robust models.

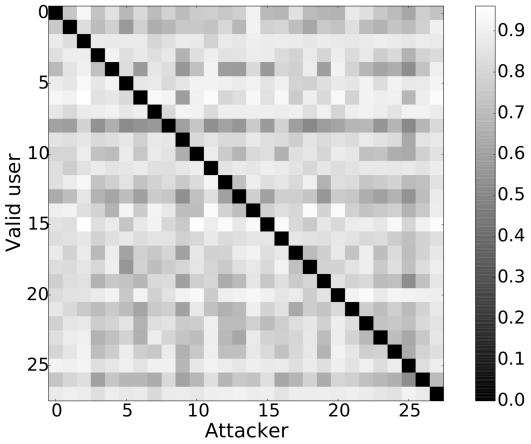


Fig. 3. Confusion Matrix.

We also show similar results for the keyboard features. However, in the case of keyboard features the growth of accuracy is not as fast and we need around 150 vectors to achieve the maximum performance, as a difference from the swipe features, where we only need around 100. In terms of FAR and FRR, the test shows similar trends to the results with swipe features.

3.3. Confusion Matrix

The confusion matrix on Figure 3 contains the information on the mean accuracy when using various pairs of training users and attackers. This figure shows that for the most pairs we can achieve high accuracy in recognizing the attack. However, for some users the performance is still problematic. In particular, for the users 8, 13, and 26 we cannot properly differentiate the attackers, because the models are not discriminative enough. By gathering more data this performance can be improved.

4. CONCLUSION

In this paper we propose a budget-efficient model for continuous authentication on mobile devices, using the data from touchscreen interaction as well as the values from accelerometer and rotation sensor. We develop an online budgeted version of the well-known OC-SVM algorithm and train it using gradient descent in an online manner. We execute the training and testing experiment on a dataset from behavioral traces that we gather from 28 subjects. This enables us to achieve performance highly comparable to the results using unlimited models, while reducing the model size by more than 50%.

Acknowledgements

This research was supported by the German Federal Ministry of Education and Research under grant 16KIS0327 (IUNO).

5. REFERENCES

[1] Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith, "Smudge attacks on smartphone touch screens," in *Proceedings of the 4th USENIX Conference on Offensive Technologies*, Berkeley, CA, USA, 2010, WOOT'10, pp. 1–7, USENIX Association.

[2] Florian Schaub, Ruben Deyhle, and Michael Weber, "Password entry usability and shoulder surfing susceptibility on different smartphone platforms," in *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, New York, NY, USA, 2012, MUM '12, pp. 13:1–13:10, ACM.

[3] Dan Goodin, "Fingerprint lock in Samsung Galaxy 5 easily defeated by whitehat hackers," `https://arstechnica.com/security/2014/04/fingerprint-lock-in-samsung-galaxy-5-easily-defeated-by-whitehat-hackers/`, Apr 2014.

[4] Cheng Bo, Lan Zhang, Xiang-Yang Li, Qiuyuan Huang, and Yu Wang, "Silentsense: Silent user identification via touch and movement behavioral biometrics," in *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, New York, NY, USA, 2013, MobiCom '13, pp. 187–190, ACM.

[5] Elaine Shi, Yuan Niu, Markus Jakobsson, and Richard Chow, "Implicit authentication through learning user behavior," in *International Conference on Information Security*. Springer, 2010, pp. 99–113.

[6] Daniel Buschek, Alexander De Luca, and Florian Alt, "Improving accuracy, applicability and usability of keystroke biometrics on mobile touchscreen devices," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, New York, NY, USA, 2015, CHI '15, pp. 1393–1402, ACM.

[7] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *CoRR*, vol. abs/1207.6231, 2012.

[8] Hui Xu, Yangfan Zhou, and Michael R Lyu, "Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones," in *Symposium On Usable Privacy and Security, SOUPS*, 2014, vol. 14, pp. 187–198.

[9] Hugo Gascon, Sebastian Uellenbeck, Christopher Wolf, and Konrad Rieck, "Continuous authentication on mobile devices by analysis of typing motion behavior.," in *Sicherheit*. Citeseer, 2014, pp. 1–12.

[10] Michael Velten, Peter Schneider, Sascha Wessel, and Claudia Eckert, "User Identity Verification Based on Touchscreen Interaction Analysis in Web Contexts," in *Information Security Practice and Experience*, Javier Lopez and Yongdong Wu, Eds., vol. 9065 of *Lecture notes in computer science*, pp. 268–282. Springer International Publishing, Cham, 2015.

[11] Rahul Murmuria, Angelos Stavrou, Daniel Barbará, and Dan Fleck, "Continuous authentication on mobile devices using power consumption, touch gestures and physical movement of users," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2015, pp. 405–424.

[12] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt, "Support vector method for novelty detection," in *Advances in neural information processing systems*, 2000, pp. 582–588.

[13] Francesco Orabona, Joseph Keshet, and Barbara Caputo, "Bounded kernel-based online learning," *Journal of Machine Learning Research*, vol. 10, no. Nov, pp. 2643–2666, 2009.

[14] Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile, "Tracking the best hyperplane with a simple budget Perceptron," *Machine Learning*, vol. 69, no. 2-3, pp. 143–167, 2007.

[15] Zhuang Wang and Slobodan Vucetic, "Online Passive-Aggressive Algorithms on a Budget," in *AISTATS*, 2010, pp. 908–915.

[16] Michael Velten, Peter Schneider, Sascha Wessel, and Claudia Eckert, "User identity verification based on touchscreen interaction analysis in web contexts," in *Information Security practice and experience*, pp. 268–282. Springer, 2015.

[17] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 136–148, 2013.

[18] Thomas Rückstieß, Christian Osendorfer, and P Patrick van der Smagt, "Sequential feature selection for classification.," in *Australasian conference on artificial intelligence*. Springer, 2011, pp. 132–141.

[19] Zhuang Wang, Koby Crammer, and Slobodan Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training," *Journal of Machine Learning Research*, vol. 13, no. Oct, pp. 3103–3131, 2012.