Poster: Towards Continuous Authentication Based On Mobile Messaging App Usage

Eric Klieme
NovaTec Consulting GmbH
Dieselstraße 18/1
70771 Leinfelden-Echterdingen
eric.klieme@novatec-gmbh.de

Klaus-Peter Engelbrecht
Quality and Usability Lab, TU Berlin
Ernst-Reuter-Platz 7
10587 Berlin
klaus-peter.engelbrecht@telekom.de

Sebastian Möller
Quality and Usability Lab, TU Berlin
Ernst-Reuter-Platz 7
10587 Berlin
sebastian.moeller@telekom.de

1. INTRODUCTION

With the help of current messaging apps, files or location data can be exchanged in addition to traditional text messages in a convenient way. Thus, more sensitive data is stored in the apps [4] and access of adversaries becomes a higher risk in the case the device gets lost or stolen [7]. Current authentication mechanisms such as PINs or graphical passwords [8] are circumvented too easily by *shoulder-surfing* [2] or smudge-based attacks [1]. Also, authentication mechanisms may often be turned off due to usability reasons, as frequent interaction requires authentication every time. Continuous authentication has been proposed as a way to protect the data by authenticating the legitimate use in a constant manner in the background based on interaction data. In this work, a framework for collecting natural touchscreen interaction data, which is built into an open-source messaging app, is described. Two studies were conducted to collect interaction data of legitimate users and adversaries. Preliminary results show that a distinction between legitimate users and adversaries is possible based on the touch gestures alone.

2. DATA COLLECTION FRAMEWORK

To collect interaction data in the field, an existing messaging app (YAXIM) was enhanced with log functionalities. YAXIM is an instant messaging application implementing the *Extensible Messaging and Presence Protocol (*XMPP) including some *XMPP Extension Protocols (XEP)*. By that, it provides convenient features known from popular messaging apps such as *Delayed Delivery* (a message is also delivered if a recipient is currently offline) or *Message Delivery Receipt*s (a notification can be requested if a message is delivered to a recipient). Furthermore, a client-server infrastructure only accessible by the users was set up, including the app clients on the users` mobile devices and an XMPP server (Openfire). As a consequence, a high level of data privacy could be reached since, for example, the automated logging of all exchanged messages at the server side was switched off.

YAXIM (www.yaxim.org) is an open source app (GNU GPLv2), which was a requirement as no messaging app provides the needed logging features out-of-the-box. Although there exist other open source XMPP apps, YAXIM provided the needed level of provided features, a respective user interface and was not too complex since instrumentation required a deep understanding of the application. We instrumented YAXIM to retrieve and to log touch interaction on a daily basis. A central “touch listener” was implemented that received touch events from the contact list view, the chat view and the settings view. In addition, all control elements such as the ”send” button or the chat input text field were registered at this listener. For each touch interaction, the X and Y coordinate, the timestamp, the pressure and the respective action (up, down, move) are forwarded to the listener.

To log all the information on the user activity level, YAXIM’s program flow was analyzed to identify the methods, which are responsible for actions like sending a message or that are triggered if a new message is received. In terms of logging user interactions like “pushing the back button” or “switching of the screen”, the enhancement of lifecycle methods typical to the Android operation system like onCreate() or onResume() helped. Similar to a central touch listener, a central user activity listener was responsible to aggregate all events forwarded from the contact, chat and settings view. This information, which was later written to log files, included the timestamp, the action done (like opening a chat view), the reason (like choosing a contact) and the time that elapsed based on the transition that happened before (like the time elapsed between opening the contact view and choosing a contact). Apart from all activities resulting in full screen transitions, all activities within the chat view were considered, too. By that each message writing process was logged including information on the recipient, the length of the text, the duration, the speed, the corrections, and if the message was successfully sent or interrupted. In case it was interrupted, the respective reason (e.g. pushing the back or home button) was logged, too. Note that no message content was logged as all key stroke based information was dropped, and recipient related information included only hashed values.

Overall, an extensible framework of classes and interfaces was designed implementing the publish-subscribe pattern. As a result, arbitrary observers or types of events can be added in later stages of the framework like, for instance, key stroke events, events from sensors like the accelerometer, or events published by other apps. In the same manner, events or authentication scores could be published to other apps.

In order to prevent unwanted data loss, an automated data backup mechanism was implemented: All daily log files were put into a zip archive and were transmitted to the server if the device was connected to an external power source and to a Wi-Fi.

3. PRELIMINARY RESULTS

Data of legitimate use of the app was collected in the field. Eight participants (7 male; 22-26 years; M=24.3; SD=1.50) participated as part of groups of 3-4 persons chatting with each other for 9-16 days. Participants registered for the experiment as a complete chat

group. Thus, they were intrinsically motivated to use our app in their everyday live. The app was installed on the participants' own smartphones. Thus, compared to previous work where touch data were obtained from rather artificial tasks [3, 5], more natural interaction data could be collected.

In a second study dedicated to collecting attacker interaction data, 21 technically skilled participants (10 male; 20-49 years; M=27.9; SD=6.74) performed two types of attacks on the app. For the study, they were invited to a test lab and provided with a smartphone with the app installed and a fictional contact list and message history stored on the phone. First, participants reconstructed a fictional social network. To do so, they had to assign each contact stored in the phone to groups like "family", "friends", or "colleagues", based on the content of the messages stored in the phone. Afterwards, they distributed a fictional malicious URL to as many contacts as possible, making sure that the link will be clicked by as many recipients as possible. Half of the participants had 1 minute for the first task, and 3 minutes for the second; the other half vice versa.

Afterwards, more than 20 features were extracted out of typical touch movements such as scrolling. So far, we only studied how well attack detection is possible based on movement touch events such as scrolling within the contact view, scrolling within the chat view to read messages, long-click in text field for writing messages, long-click on send button, long-click somewhere else. Apart from simple features, like the start or end coordinates of a movement, or its maximum X- or Y-coordinate, higher order features like direction, mean resultant length, acceleration, and velocity were extracted based on the work of Frank et al. [3]. To allow for a comparison of data collected with different smartphones, all coordinates and distances were normalized by the screen size. The final number of data points obtained from the legitimate user test was 18095. Due to differences between the users in using the app, different amounts of data points were obtained for each user (M_{user} = 2261.9; MIN_{user}=100; MAX_{user}=6659). The overall number of data points for adversaries was 12637 (M_{user}=631.9; MIN_{user}=46; MAX_{user}=1905).

To evaluate how accurately attacks can be distinguished from legitimate use, we trained 8 Naïve Bayes classifiers on the data, one for each user. Evaluation was done with data of the respective user and of all attackers. We cross-validated (CV) over the attackers, leaving out the most recent legitimate user data for testing in each CV round.

Table 1 shows that the true-negative rate is approximately 91% on average and thus provides a good recognition of imposters, missing only 9% of all attacks. In terms of the true-positive, approximately 69% are reached. That is ok, but too low for a direct deployment in real applications. Based on a false-negative rate of ~31%, every third sample swipe would identify him or her as an attacker despite being the legitimate user.

In order to judge the strength of the proposed authentication approach, it should be considered that multiple user actions could be analyzed to detect an attack, leading to higher true positive and true negative rates. However, contrary to some previous approaches, where the user is authenticated before getting access to the app (e.g. [6]), our algorithm detects an attack after the adversary already had access to the app and the data stored within it. Thus, the detection should happen as early as possible. Because of that, the final model should process as many types of actions as possible (e.g., also taps) and should return reliable results after as few user actions as possible. We have collected such data and will investigate them in the future.

Table 1 Classification Results

True positive	True negative	False positive	False negative	Area under ROC
69.33	90.98	9.02	30.67	0.91

Finally, in order to mimic the legitimate user's behavior, an attacker could watch him using the application. However, due to mobile internet subscription and the "always-on" characteristics of messaging services, interactions with the app can happen at random places and times during the day. This makes it very hard for the attacker to observe enough interaction without being noticed. Also, mimicking the user behavior would constrain the actions an adversary could take, making it hard to reach goals which the legitimate user would never pursue.

4. ACKNOWLEDGMENTS

We are grateful for the many comments and suggestions we received from usable privacy and security group of our lab.

5. REFERENCES

[1] Aviv, A.J., Gibson, K., Mossop, E., Blaze, M., Smith, J.M. Smudge attacks on smartphone touch screens. In *Proc. of 4th USENIX conference on Offensive technologies* (2010), 1–7.

[2] Forget, A., Chiasson, S., Biddle, R. Shoulder-surfing resistance with eye-gaze entry in cued-recall graphical passwords. In *Proc. of SIGCHI Conference on Human Factors in Computing Systems* (2010), 1107–1110.

[3] Frank, M., Biedert, R., Ma, E., Martinovic, I., Song, D. Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication. IEEE Transactions on Information Forensics and Security, 8, 1 (2013), 136–148.

[4] Karlson, A.K., Brush, A.J.B., Schlechter, S. Can I borrow your phone? Understanding concerns when sharing mobile phones. In *Proc. of SIGCHI Conference on Human Factors in Computing Systems* (2009), 1647–1650.

[5] Kolly, S.M., Wattenhofer, R., Welten, S. A personal touch: recognizing users based on touch screen behavior. In *Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones* (2012), 1:1–1:5.

[6] De Luca, A., Hang, A., Brudy, F., Lindner, C., Hussmann, H. Touch me once and I know it's you! Implicit authentication based on touch screen patterns. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems* (2012), 987–996.

[7] Symantec Smartphone Honey Stick Project | Symantec: http://www.symantec.com/about/news/resources/press_kits/detail.jsp?pkid=symantec-smartphone-honey-stick-project Retrieved March 6, 2014.

[8] United States Patent: 5559961: http://patft.uspto.gov/netacgi/nph-Parser?Sect2=PTO1&Sect2=HITOFF&p=1&u=/netahtml/PTO/search-bool.html&r=1&f=G&l=50&d=PALL&RefSrch=yes&Query=PN/5559961 Retrieved March 6, 2014.