5-2016

Leveraging automated privacy checking for design of mobile privacy protection mechanisms

Joseph Joo Keng CHAN
Singapore Management University, joseph.chan.2012@phdis.smu.edu.sg

Lingxiao JIANG
Singapore Management University, lxjiang@smu.edu.sg

Kiat Wee TAN
Singapore Management University, williamtan@smu.edu.sg

Rajesh BALAN
Singapore Management University, rajesh@smu.edu.sg

Citation

CHAN, Joseph Joo Keng; JIANG, Lingxiao; TAN, Kiat Wee; and BALAN, Rajesh. Leveraging automated privacy checking for design of mobile privacy protection mechanisms. (2016). *CHI 2016: The 34th Annual CHI Conference on Human Factors in Computing Systems: San Jose, CA, May 7-12.* 1-4. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/3513

Leveraging Automated Privacy Checking For Design of Mobile Privacy Protection Mechanisms

Joseph Chan Joo Keng
School of Information Systems,
Singapore Management
University
joseph.chan.2012@phdis.smu.edu.sg

Lingxiao Jiang
School of Information Systems,
Singapore Management
University
lxjiang@smu.edu.sg

Tan Kiat Wee
School of Information Systems,
Singapore Management
University
williamtan@smu.edu.sg

Rajesh Krishna Balan
School of Information Systems,
Singapore Management
University
rajesh@smu.edu.sg

Acknowledgements
This research is supported by the National Research Foundation Singapore under its Interactive Digital Media (IDM) Strategic Research Programme.

Abstract

While mobile platforms rely on developers to follow good practices in privacy design, developers might not always adhere. In addition, it is often difficult for users to understand the privacy behaviour of their applications without some prolonged usage. To aid in these issues, we describe on-going research to improve privacy protection by utilizing techniques that mine privacy information from application binaries as a grey-box (*Automated Privacy Checking*). The outputs can then be utilized to improve the user's ability to exercise privacy-motivated discretion. We conducted a user study to observe the effects of presenting information on leak-causing triggers within applications in the form of privacy message overlays. We found that while users' prior usage time largely determined their usage behaviour, presenting trigger information helped users who disapproved with data use and had sufficient understanding of the implications of data leaks. Users' inherent level of privacy consciousness and surprise levels were also factors in ensuring the effectiveness of messages.

Author Keywords

mobile privacy, binary analysis, user-behavioural factors

Introduction

Mobile application privacy research has received a lot of attention in recent years. Mobile applications can gather too much about the user that he or she may be uncomfort-

Advantages in Forensics Privacy Checking

(i) Lower CPU Overheads/Improved User-Friendliness: CPU overheads reduced because leak detectors are utilized on automated app testers in the back-end, and results passed to users.

(ii) Improved Flexibility of Notifications: Provides information on privacy 'leak-causes', as well as better tailoring of notification mechanisms due to prior knowledge of data-access characteristics.

(iii) Improve Privacy Policies set by User and/or App Developer: *Automated characterization* of apps allows for formulation of *pre-deployment strategies* that can complement privacy design.

(iv) Overcome Limitation of Individual Tools: Framework allows for combination of complementary leak detectors.

(v) Ease in Set-up: Does not require phone modifications on user-side.

able with sharing, or may be unnecessary for the utility of the application. The problem is being tackled is a few main areas, for example in research that empirically finds more about users' privacy usage behaviours [4, 12], designing HCI interfaces that makes it easier for users to configure and make decisions [3, 2], as well as in engineering systems for detection and access-control of applications' usage of privacy data [5, 13].

While tools have been created by platform developers as well as by the research community [8, 13], inherent drawbacks exist in terms of: (i) Significant CPU overheads required for privacy detection, ([13] evaluated that detection overheads on CPU can be up to 35% over baseline), (ii) Notifications have limited flexibility due to run-time nature; Notifications that informs users 'when' but not 'how' privacy leaks occur,(iii) Incorrect privacy policies set by the User (Inexperience or lack of knowledge) and/or App Developer (Undesirable privacy design practices), (iv) Limitations of individual detection tools that could complement each other (e.g. Ap-Ops/PMP [8, 1] detects data access but not HTTP leaks, TaintDroid [5] detects HTTP leaks but not implicit data flows in applications), as well as (v) Difficulties in set-up. (Most tools require customized phone images or modifications to the operating system)

To alleviate these drawbacks, we introduced a *Mobile App Forensics Privacy Checking Framework in the Back-end* [9, 7]. The framework advocates a similar philosophy of malware detection systems in application stores (Such as the Android and Apple store Bouncers that vet apps prior to publishing [11, 6]), where privacy behaviours and leak-causes of applications are tested in the back-end on testing devices using dynamic/static analysis techniques. The results of such testing can then be utilized in a lightweight client for notification.

Advantages in Forensics Privacy Checking

Automated Forensics Privacy Checking utilizes *back-end testing* of applications, in contrast to currently available tools which detect application data-access in *real-time* during application usage. It has advantages as it allows the creation of a taxonomy of privacy leak-causes and characterization of applications (e.g. Does the app leak data on start-up? In a periodic fashion? What are culprit buttons/widgets?), even before they are installed on user phones. Notification is then performed with a customized Android app from a 'leak-cause' database.

(Please see left Side-Bar) CPU Overheads are lowered because leak detectors do not actually run on user devices, but are utilized on automated app testers running in the back-end. Lower CPU Overheads in turn improve user-friendliness due to better UI responsiveness. Privacy Checking also improves the Flexibility of Notifications because it allows for pre-deployment tailoring of message appearances, depending on the leak characteristics of the app. For example, if it is known that privacy data is leaked in a periodic fashion, a single notification message can be made indicating this, or providing the user with a prior-use choice of control. Instead of having to constantly notify the user and letting him/her discover the characteristic over time, which has privacy costs. Information on user-triggered 'leak-causes' are also available, which can indicate to users data accessed from clicking certain buttons in the app.

Privacy checking can also potentially improve privacy policies set by users by aiding them in making decisions in the configuration of privacy profiles. An example is that auto-generated summaries might be produced on the type of app vs. its privacy behaviours. Privacy checking framework also allows for the combination of leak-detection tools, which can help to overcome the individual limitations of

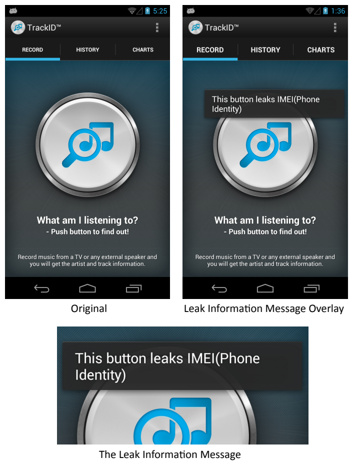
TrackID™
RECORD
HISTORY
CHARTS
This button leaks IMEI(Phone Identity)
What am I listening to?
- Push button to find out!
Record music from a TV or any external speaker and you will get the artist and track information.
Original
Leak Information Message Overlay
The Leak Information Message

Figure 1: Utilizing outputs of privacy checking - Privacy Messages overlaid on-top of culprit buttons/widgets during Application Usage, providing information that the particular application page is involved with access of the user's Phone Identifier (IMEI)

each of these tools. For example, the TaintDroid detector [5] flags data-leaks but is unable to detect implicit leaks or leaks by 3rd-party native libraries. Ap-Ops/PMP [8, 1] covers a blind-spot in data-access detection on 3rd-party native libraries.

Ongoing Research Work: We implemented an automated app testing system that utilizes Automated Model Checking [10], and parallelizes test instances over multiple test devices and emulators. Test apps are traversed automatically by triggering user-actions that cause app transitions. Using privacy leak detectors and app-usage layout logs, we mine and infer privacy characteristics of apps (Periodicity of data-leaks; Leak-causing Widgets/Views). Testing times ranged from a few minutes to 4-5 hours depending on the structure of the app. Due to the high testing times required, it became evident that an exhaustive exploration would not be a practical strategy. We thus utilized Directed Testing from the Control Flow Graph (CFG) obtained from static analysis of apps to prune the search. By identifying the UI components that are linked to data-access API calls, we directed app testing towards the app areas that are linked to privacy data accesses. We found that this approach can reduce testing times significantly.

Deployment Hurdles in Playstores: App-stores face an increase in processing time and effort. Additional resources are also required for storage and delivery of results. As privacy is usually viewed as a secondary concern, as compared to malware or virus detection, this also might discourage app-stores from increasing resources towards privacy checking.

Utilizing Outputs as Notification of 'Leak-Causes'

The outputs of our privacy checking were apps' data-leak characteristics (such as whether app leaked data on Start-Ups; or in Periodic manner), as well as the buttons/widgets on specific app layouts that leaked when clicked upon. There are many possible ways to utilize these outputs; For e.g., as additional summaries in configurable permission manager lists, notification scroll-downs or pop-up messages. We chose to utilize outputs as privacy message summaries that appear to users on app start-up, as well as by overlaying messages directly on-top of leak-causing buttons/widgets. (Figure. 1 shows an example of a privacy overlay). The privacy messages appeared when the user reaches targeted in-app views/layouts containing the involved buttons. We conducted a 2-week Field Study, to evaluate the effects of privacy messages on real users.

2-Week Field Study

The field study had 47 users, who were randomly divided into separate groups of Test (With Messages), and Control (Without Messages). (25 Users in Test, 22 Users in Control) They consisted of staff/students as well as external working adults. We utilized 10 test apps that many of the users had already been using regularly (Social Media, Utility, News as well as Game Apps). We captured the characteristics of users from pre and post-study surveys on 7-point Likert-Scale as well as Yes/No questions. These characteristics include users' *Prior app Usage-time (Familiarity)*, *Surprise level*, *Usability*, *Privacy Consciousness level* and *Disapproval of app data-usage*. In addition to privacy messages, a subset of 17 users were provided with a stimulus of additional User Education on the meaning of messages (e.g. IMEI were identifiers that could allow tracking etc.) as well as implications of allowing accesses (private data could be sent over the network and stored by developers). We conducted a multiple regression with characteristics as Independent variables, and the Duration of App Usage (Log) and No. of Leak Buttons Clicked (Log) as Dependent variables (Table. 1). We investigated these Dependent variables as they were proportional to the amount of privacy data being leaked by the application.

From Table. 1, we found that users' Prior App Usage Time (Familiarity) was the dominant individual factor in affecting the Duration of App Usage, while User Education was significant in reducing the No. of Leak Buttons that users clicked. There were significant 3-factor interactions, which indicated that the Privacy Messages were effective for users who Disapproved of the data use and had appropriate User Education, professed high Privacy Consciousness levels or were Surprised with the data accesses.

Depend. Variable	Independ. Variable	Est.	Std. Error	*p*-val
Duration of App Usage (Log) (No. of Leak Buttons Clicked (Log))	(1) Privacy Messages	-0.13 (0.04)	0.83 (1.73)	0.88 (0.98)
	(2) Disapp -roval of Data Use	-0.01 (-2.52)	0.18 (1.61)	0.95 (0.12)
	(3) User Education	0.02 **(-0.83)**	0.17 **(0.47)**	0.90 (**0.08***)
	(4) Privacy Conscious-ness	0.12 (-0.09)	0.12 (0.31)	0.33 (0.76)
	(5) Surprise Level	0.04 (-0.03)	0.04 (0.12)	0.31 (0.78)
	(6) Prior App Usage Time (Familiarity)	**0.24** (0.10)	**0.05** (0.12)	**1.23×10^{-5} ***** (0.40)
	Interaction (1:2:3)	**-1.23** **(-3.83)**	**0.32** **(0.95)**	**1.38×10^{-4} ***** **(7.86×10^{-5} ***)**
	Interaction (1:2:4)	**-0.28** (-0.19)	**0.12** (0.36)	**0.021 ***** (0.60)
	Interaction (1:2:5)	**-0.15** **(-0.44)**	**0.08** **(0.25)**	**0.07*** **(0.08*)**

Table 1: Results: Multiple Linear Regression of Field Study (***99%,**95%,*90% Confidence Intervals)

Discussion & Implications of Findings

Our notifications can be described as *'Non-Enforcing'*, in which messages appear to the user in the background. This is in contrast to *'Enforcing'* mechanisms, in which message controls appear with a GUI freeze, where the user is required to make a selection on the data access before app functionality can be restored. Our study highlights the conditions under which a 'Non-Enforcing' mechanism can be effective. In particular, we uncovered that user-disapproval with accesses, user-education, inherent levels of privacy consciousness as well as high surprise-levels play a part in effectiveness. For strong overall effects, especially for users who have a longer history of usage of the particular app as well as in warning of more serious issues such as malware, 'Enforcing' mechanisms would be advocated.

An observation is that privacy messages is unlikely to cause overall usage durations to drop, and this supports the notion that mobile platforms can readily deploy privacy messages in an app without significant decrease in overall usage rates. Another implication is that it is greatly beneficial for platforms to include educational aspects, as well as utilize message designs that increase the contrasts between legitimate and questionable usage for increased surprise to users.

References

[1] Yuvraj Agarwal and Malcolm Hall. 2013. ProtectMyPrivacy: detecting and mitigating privacy leaks on iOS devices using crowdsourcing. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 97–110.

[2] Hazim Almuhimedi, Florian Schaub, Norman Sadeh, Idris Adjerid, Alessandro Acquisti, Joshua Gluck, Lorrie Faith Cranor, and Yuvraj Agarwal. 2015. Your Location has been Shared 5,398 Times!: A Field Study on Mobile App Privacy Nudging. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 787–796.

[3] Rebecca Balebako, Jaeyeon Jung, Wei Lu, Lorrie Faith Cranor, and Carolyn Nguyen. 2013. Little brothers watching you: Raising awareness of data leaks on smartphones. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*. ACM, 12.

[4] Rebecca Balebako, Florian Schaub, Idris Adjerid, Alessandro Acquisti, and Lorrie Cranor. 2015. The Impact of Timing on the Salience of Smartphone App Privacy Notices. In *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*. ACM, 63–74.

[5] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. 2014. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)* 32, 2 (2014), 5.

[6] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner. 2011. A survey of mobile malware in the wild. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*. ACM, 3–14.

[7] Swapna Gottipati, Jeena Sebastian, Luong Trung Tuan, Tan Kiat Wee, Jeffrey Chan, Joo Keng, Kartik Muralidharan, Tadashi Okoshi, Youngki Lee, Abhishek Misra, and others. 2014. Mobile platform and application research at SMU LiveLabs. In *Communication Systems and Networks (COMSNETS), 2014 Sixth International Conference on*. IEEE, 1–4.

[8] Russell Holly. 2015. Using App Permissions in Android M. (June 2015). http://www.androidcentral.com/using-app-permissions-android-m

[9] Joseph Chan Joo Keng, Tan Kiat Wee, Lingxiao Jiang, and Rajesh Krishna Balan. 2013. The case for mobile forensics of private data leaks: towards large-scale user-oriented privacy protection. In *Proceedings of the 4th Asia-Pacific Workshop on Systems*. ACM, 6.

[10] Kyungmin Lee, Jason Flinn, Thomas J Giuli, Brian Noble, and Christopher Peplin. 2013. Amc: Verifying user interface properties for vehicular applications. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 1–12.

[11] Jon Oberheide and Charlie Miller. 2012. Dissecting the android bouncer. *SummerCon2012, New York* (2012).

[12] Joshua Tan, Khanh Nguyen, Michael Theodorides, Heidi Negrón-Arroyo, Christopher Thompson, Serge Egelman, and David Wagner. 2014. The effect of developer-specified explanations for permission requests on smartphone user behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 91–100.

[13] Rubin Xu, Hassen Saïdi, and Ross Anderson. 2012. Aurasium: Practical Policy Enforcement for Android Applications.. In *USENIX Security Symposium*. 539–552.