PVDetector: A Detector of Privacy-Policy Violations for Android Apps

Rocky Slavin[1], Xiaoyin Wang[1], Mitra Bokaei Hosseini[1], James Hester[2], Ram Krishnan[1], Jaspreet Bhatia[3], Travis D. Breaux[3], and Jianwei Niu[1]
[1]University of Texas at San Antonio, San Antonio, TX, USA
[2]University of Texas at Dallas, Dallas, TX, USA
[3]Carnegie Mellon University, Pittsburgh, PA, USA
{rocky.slavin, xiaoyin.wang, mitra.bokaeihosseini, ram.krishnan, jianwei.niu}@utsa.edu
william.hester@utdallas.edu, {jbhatia, breaux}@cs.cmu.edu

ABSTRACT

Many Android apps heavily depend on collecting and sharing sensitive privacy information, such as device ID, location, and postal address, to provide service and value. To protect user privacy, apps are typically required by market places to provide privacy policies informing users about how their private information will be processed. In this paper, we present PVDetector, an automatic tool that analyzes Android apps to detect privacy-policy violations, i.e., inconsistencies between an app's data collection code and the corresponding description in its privacy policy.

1. INTRODUCTION

Mobile apps become increasingly prevalent as they provide timely, user-friendly services such as shopping, instant messaging, travel, and gaming in context and on-demand. To meet users' requirement and provide value, these services heavily depend on collecting and sharing sensitive personal information, such as user location, contact lists, and app usage patterns. The increased growth in app markets has out-paced the development of mechanisms to guarantee user privacy. Privacy policies are the primary means to inform users about how apps access and process their personal information. Since privacy policies are legal documents, which may not be written by developers, and the code can change while the policy remains static, there is a risk that the privacy polices become misaligned with actual code in the software product. In addition to misinforming users, such inconsistencies between policies and code may have severe legal repercussions, resulting in significant fines or burdensome security audits.

In this paper, to bridge the gap between policy and practice, we present a tool named PVDetector (Privacy-policy Violation Detector) to map the descriptive phrases in privacy policies expressed in natural language and privacy-related API invocations implemented in the corresponding code. This mapping will provide the semantics needed to check code for misalignment with privacy policy, and to suggest where the code or policy may be changed to fit the functional and legal requirements of apps.

MobileSoft '16 May 16-17 2016, Austin, TX, USA
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4178-3/16/05.
DOI: http://dx.doi.org/10.1145/2897073.2897720

2. TOOL CONSTRUCTION AND USAGE

The overview of the construction process and the structure of PVDetector is presented in Figure 1. In the figure, we use the annotations of data flow chart, and the scope of PVDetector tool is presented as a box with dashed border and label "PVDetector".

As shown in the figure, to construct PVDetector, we need to prepare two resource files: the policy phrase ontology that describes the conceptual inclusion relationship between policy phrases, and the API-Phrase mapping file that maps each privacy-related Android API method to a number of policy phrases. In the preparation of policy phrase ontology, we first leveraged crowd sourcing to annotate phrases related to collected privacy-data from 50 privacy policies of 50 top Android apps (used as seed apps) in the Google Play Market. After building the policy lexicon with annotation, we manually find out all conceptual inclusion relationships between policies, and with these relationships, we build a policy phrase ontology with the ontology building tool Protégé [1]. In the preparation of API-Phrase mapping, we also performed crowd sourcing to annotation privacy-collection related API methods from the official Android API documents. After that, for each annotated API method, we manually map it to one or multiple phrases in the privacy lexicon, which forms the final API-Phrase mapping.

After we constructed the two resource files, our PVDetector tool is fully automatic for any given Android app. The input of PVDetector is the byte code (i.e., apk file) and the privacy policy of an app, and the output of PVDetector is the list of detected violations. A violation can be either a strong violation (no phrases related to an API method invocation is found) or a weak violation (no phrases directly mapped to an API method invocation is found, but some related abstract phrases are found).

The PVDetector tool mainly consists of three components: FlowDroid [2], the phrase extraction component, and the violation detection component. For information flow analysis, we leverage the state-of-the-art tool FlowDroid to analyze the byte code of a given app, and identify all the privacy-data-collecting API invocations whose return values are sent to network sinks. Our phrase extraction component first identifies data-collection-related paragraphs in privacy policies according to whether a paragraph contains any collection verbs (e.g., collect, access), and then extracts the policy phrases occurrences from these paragraphs. Finally, our violation detection component matches the network-targeting privacy-data-collecting APIs and mentioned policy phrases, to detect and output all violations.

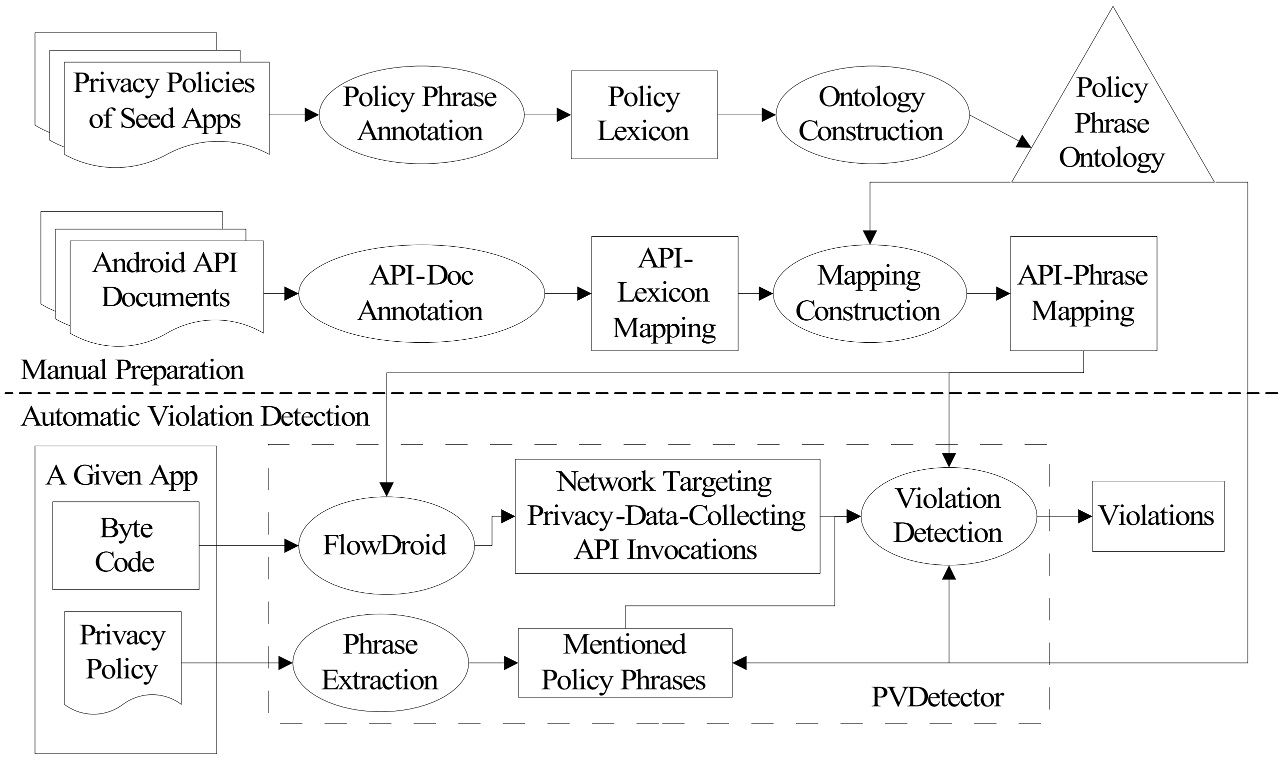

Figure 1: Tool Construction and Structure

3. RELATED WORK

On tool support for privacy-policy-related tasks, Rowen et al. [8] have developed an IDE plugin, Privacy Policy Auto-Generation in Eclipse (PAGE), for generating privacy policies along side the development of the app. PAGE works by guiding the user through a series of questions about the implementation of the app. Based on the answers, PAGE uses existing policy templates to generate a privacy policy for the app. In another research, Kelley et al. discussed the importance of privacy policy and permission representation in Android apps marketplaces conducted through a user study [6]. They designed a privacy fact checklist for a set of 12 Android applications. Their goal was studying the effects of privacy and permission representation on user decision making for downloading an Android application. Much research has been carried out on ontology implementation and usage in computer and information systems in recent years [3, 5]. However, the majority of these works are regarding permission based systems, firewalls, and pervasive systems. The research work most close to this paper is Eddy [4]. In the paper, Breaux et al. presented an ontology to analyze the privacy policy of multi-tier systems to find the conflicts between the policies regarding data collection, usage, retention and transfer. There has been also a number of works [2] [7] on information flow analysis and FlowDroid [2] is a state-of-art static information analysis tool for Android apps.

Acknowledgement

The authors are supported in part by NSF Awards CNS-0964710, CNS-1330596, CCF-1464425, NSA Grant on Science of Security, and DHS grant DHS-14-ST-062-001.

4. REFERENCES

[1] Protégé. http://protege.stanford.edu/, 2015.

[2] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 259–269, 2014.

[3] J. Bradshaw, A. Uszok, R. Jeffers, N. Suri, P. Hayes, M. Burstein, A. Acquisti, B. Benyo, M. Breedy, M. Carvalho, et al. Representation and reasoning for daml-based policy and domain services in kaos and nomads. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 835–842. ACM, 2003.

[4] T. D. Breaux, H. Hibshi, and A. Rao. Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements. *Requirements Engineering*, 19(3):281–307, 2014.

[5] H. Chen, F. Perich, T. Finin, and A. Joshi. Soupa: Standard ontology for ubiquitous and pervasive applications. In *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, pages 258–267. IEEE, 2004.

[6] P. G. Kelley, L. F. Cranor, and N. Sadeh. Privacy as part of the app decision-making process. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3393–3402. ACM, 2013.

[7] L. Lu, Z. Li, Z. Wu, W. Lee, and G. Jiang. Chex: Statically vetting android apps for component hijacking vulnerabilities. In *CCS*, pages 229–240, 2012.

[8] M. Rowan and J. Dehlinger. Encouraging privacy by design concepts with privacy policy auto-generation in eclipse (page). In *Proceedings of the 2014 Workshop on Eclipse Technology eXchange*, pages 9–14. ACM, 2014.