GUILeak: Tracing Privacy Policy Claims on User Input Data for Android Applications

Xiaoyin Wang, Jianwei Niu, Rocky Slavin, Xue Qin, etc., Supported by NSF-SaTC 1748109, EAGER: Tracing Privacy-Policy Statements into Code for Privacy-Aware Mobile App Development

The University of Texas at San Antonio, San Antonio, TX 78249

Abstract

The Android mobile platform supports millions of users and this popularity of coupled with user data collection by Android apps has made privacy protection a well-known challenge. In practice, app producers provide privacy policies disclosing what information is collected and processed by the app. However, it is difficult to trace such claims to the corresponding app code to verify whether the implementation is consistent with the policy. Existing approaches for privacy policy analysis focus on information types directly accessed from the Android platform (e.g., location and device ID), but they are unable to trace privacy policy claims based on direct user input, a major source of private information. We propose a novel approach that automatically detects privacy leaks from user input data for a given Android app and determines whether such leakage may violate the app's privacy policy claims. The results show that Our approach was able to detect 21 strong violations and 18 weak violations from the 120 popular apps from three categories: finance, health, and dating.

Motivating Example

Figure 1: Pacer Screenshots

Virtual Invoke Example in GoalSetCheckingInReqDialog.smali

Figure 1 shows two UI screenshots from Pacer, a popular fitness app, depicting the GUI when user creates a new exercise goal. Besides editing goal descriptions and changing goal types such as steps and diet, the user also needs to set the check-in requirement by clicking the button in the red oval on the left screenshot. The right screenshot shows the pop-up window that appears after user clicking this button. Here, users will be asked to type in the desired number of check-in steps.

The right screenshot utilizes a combination of both static and dynamic layouts. Virtual invoke example shows the smali code which dynamically adds the label for "Set Check-In Requirement". The string is fetched at Line 1 as v2 with the id 0x7f07011a. In Line 2, v2 is passed as a title resulting in "Set Check-In Requirement" being dynamically defined as the title.

Just like the contexts in natural language paragraphs, input views can only be well understood with neighboring / ancestor views. **GUI context** is essential in understanding user input views and mapping the views to privacy-policy phrases, but the dynamic implementation of Android GUI makes identification of GUI context difficult.

Approach

The overview of GUILeak appears in Figure 2, which consists of three stages: (a) the ontology construction stage (blue) creates a baseline ontology by extracting reusable concepts from multiple privacy policies; (b) the app policy analysis stage (green) yields phrases that describe data types that are collected automatically or from the user directly and that are extracted from a target app's privacy policy; and (c) the app user interface (UI) analysis stage (orange) that extracts the input fields, field labels and view identifiers from the input views, which are then fed to data flow analysis as information sources.

Approach – con't

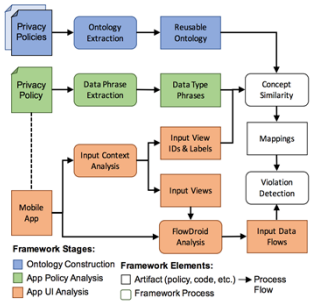

Figure 2: Approach Overview

Input Context Analysis

- Extract API method invocation that receives user input from an input view
- Extract UI labels in the context of an user input view
- Insert the dialogs into their parent activities, which allows us to identify the dialog titles and UI labels in the context of each parent.

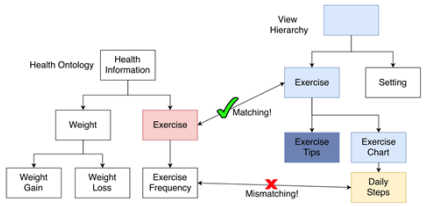

Figure 3: Illustration of Hierarchical Mapping

Hierarchical Mapping: mapping only the label / id of an input view to concept words.

As shown in Figure 3, we first collect the IDs and labels of all ancestor views for a given input view (light blue views). Next, we collect the view IDs and labels that are sibling views immediately before any collected ancestor views (the dark blue view), because sibling views may contain input view labels of their own. We refer to these collected IDs and labels, collectively, as ancestor labels. If an input view ID and label cannot be directly mapped to any ontology concept, we further map the ancestor labels to the ontology.

Evaluation

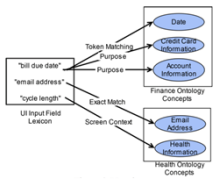

Figure 4: Mapping

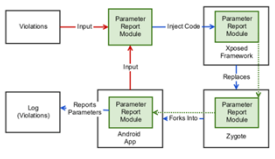

Figure 5: Validation with Xposed

Violation Detection Results

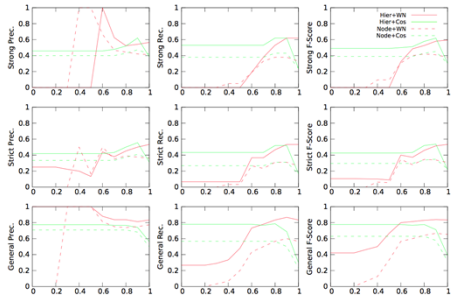

Figure 6: Comparison of Technique Variants under Different Similarity Thresholds

We use precision, relative recall and F-score as our metrics. In Figure 6, we consider two similarity measurements: WordNet (WN) and Cosine similarity (Cos). We consider two mapping strategies: node-mapping (Node), wherein only the ID and label of the input view is considered, and hierarchical-mapping (Hier), where IDs and labels of all ancestor input views are considered. This yields four approach variants by combing techniques: Hier+WN, Hier+Cos, Node+WN, Node+Cos.

Conclusions

- We developed a novel approach including UI analysis and phrase mapping techniques to detect inconsistencies between the information collection statements in an app's privacy policy and the actual collection behavior of user input data in the app's code.
- Using crowd sourcing tools, we developed privacy ontologies for three privacy-sensitive domains, health, finance and dating, which contain 333, 210 and 506 ontology terms, respectively.
- We carried out an experiment on 120 of the most popular apps in the health, finance, and dating domains and detected 21 strong violations and 18 weak violations.

References

- Rocky Slavin, Xiaoyin Wang, Mitra Bokaei Hosseini, James Hester, Ram Krishnan, Jaspreet Bhatia, Travis D. Breaux, and Jianwei Niu. 2016. Toward a Framework for Detecting Privacy Policy Violations in Android Application Code. In Proceedings of the 38th International Conference on Software Engineering (ICSE '16). ACM, New York, NY, USA, 25–36. DOI: http://dx.doi.org/10.1145/2884781.2884855
- Mitra Bokaei Hosseini, Sudarshan Wadkar, Travis D Breaux, and Jianwei Niu. 2016. Lexical Similarity of Information Type Hypernyms, Meronyms and Synonyms in Privacy Policies. In 2016 AAAI Fall Symposium Series.
- Atanas Rountev and Dacong Yan. 2014. Static Reference Analysis for GUIObjects in Android Software. In Proceedings of Annual IEEE/ACM International Symposium on Code Generation and Optimization (CGO '14). ACM, New York, NY, USA, Article 143, 11 pages. DOI:http://dx.doi.org/10.1145/2544137. 2544159