University at Buffalo The State University of New York | REACHING OTHERS

Flow Permissions for Android

Feng Shen, Namita Vishnubhotla, Mohit Arora, Eric John Lehner
Steven Y. Ko, Lukasz Ziarek
Computer Science and Engineering, University at Buffalo, The State University of New York, USA

Motivation

Android Malware has been increasing dramatically

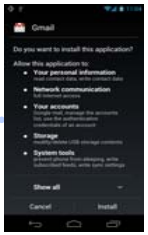
- Data leakage on Android has became a big problem.
- Concern about Android privacy has drawn more and more attention.
- What should we do with this?

How does Android protect sensitive data?

Permission Mechanism

- Data access control on Android
- Developers must declare all required permission in Manifest file.
- Users must grant at the installation time.
- The goal is to protect users' sensitive data.

- Is this good enough?
- Do it reveal how the app leverage the sensitive data?



What does Android tell us?

Requested Permissions
1.Reading phone number
2.Network access
Read data
Send data

Does it transmit the phone number over network?

Do you know apps can acquire an implicit permission from another app?

File Browser

SD Card
Requested permissions
1.Reading SD Card
Read data

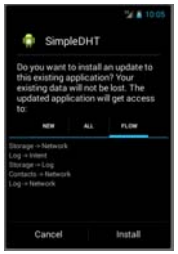
IPC
Implicit network permission

Network
Requested permissions
1.Network access
Send data

- The File Browser app does not have network access.
- The Gmail app provides public network access interface.
- The File Browser application can send data to the network via Anroid IPC.

Our approach: Flow Permissions

- ***Goal:*** Our Flow Permission mechanism is to show whether or not an application contains a flow.
- ***Structure of a Flow Permission:*** *source → sink*. For example, the following Flow Permission: *contacts → network* means the application reads the contact information and sends out over network as shown on the right.
- ***Purpose:*** It provides the user with additional context on how the sensitive resources/data are leveraged by the apps. Nevertheless, it is up to the user to decide if these behaviors should be allowed. The existence of a flow does not indicate that the app is necessarily malicious.
- ***Tool: BlueSeal,*** an automated compiler tool to analyze an apk dex file and generate Flow Permissions.



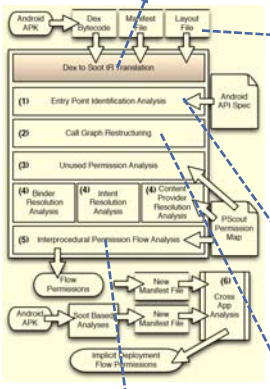
BlueSeal System Overview

1. Dex decompiler

- You cannot obtain the source code of a published application.
- BlueSeal takes an app binary and performs analysis.

2. LayoutParser

- In Android, callbacks can be defined in layout files.
- All these callbacks must be analyzed to perform precise static analysis.
- BlueSeal can automatically detect all these methods by parsing the layout files.



3. Entry Points Discovery

- An entry point is a starting point of a program.
- As show in the right figure, all the buttons can trigger some part of the code to execute.
- We can automatically find all the entry points in the program.

Fig. 1 The BlueSeal Android app analysis framework architecture. Shaded boxes represent components already present in Soot.

4. Call Graph Restructuring

- Android has methods that get executed implicitly, e.g., AsyncTask, Messenger, Handler, etc.
- Problem: no call sites for these methods

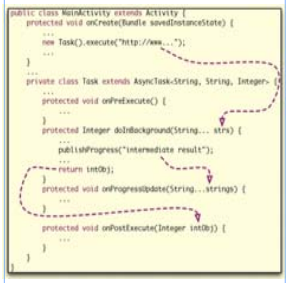


Fig. 2 A code snippet illustrating the methods that comprise the control flow of an AsyncTask in Android and the implicit flow of arguments provided by the Android framework.

5. InterProcedural flow analysis

- Per-method Summary
  - Intraprocedual forward flow analysis
  - Summary is a flow graph.
- Apply summary
  - Connect related flows between different methods
- Synthesize a global flow graph
  - Find all the flows from sources to sinks

Generated flow graph:

Fig. 3 Example for generated flow graph

BlueSeal Results

Information flows VS Flow Permissions

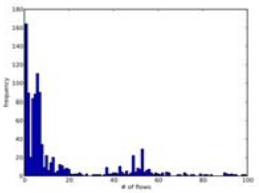


Fig. 4 Distribution of Flows generated for each app.

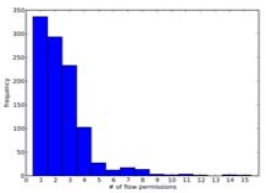


Fig. 5 Distribution of Flows Permissions generated for each app.

- As shown, most of the apps contain less than 10 Flow Permissions with a maximum of 15, which makes it practical for a user to examine.
- From these two figures, we can see that even though there are many flows in the apps, the number of Flow Permissions stays low.
- For example, there are some APKs that contain more than 1,000 flows but the number of generated Flow Permissions for those apps is less than 10.

BlueSeal Performance Results

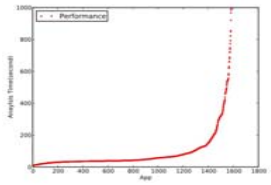


Fig. 6 Scatter plot showing the time taken to analyze all apps in seconds.

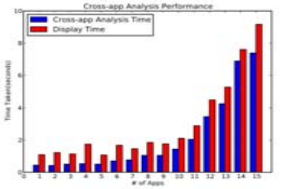


Fig. 7 Performance of our cross-app analysis.

- As shown, BlueSeal is able to analyze and synthesize Flow Permissions for all but the largest apps in under ten minutes. Only 64 apps require an analysis time greater than ten minutes.
- Cross-app analysis is performed on real Galaxy Nexus smartphone. As shown, it takes less than or around 1 second up to 9 apps. Fig. 7 shows that our cross-app analysis is feasible and practical.

Future Work

- Collaboration with RIT group
  - The RIT group is working on Android related code analyzing framework.
  - We believe their work can improve our BlueSeal with path and context sensitivity built call graph.
- Android secure app store
  - We hope to build a secure app store for Android applications.
  - It will analyze application when they submit into the app store to make sure the application is not malicious.
- Data Provenance on Android

Contact us

Information http://blueseal.cse.buffalo.edu
Mail: fengshen@buffalo.edu