Poster: Android Collusive Data Leaks with Flow-sensitive DIALDroid Dataset

Amiangshu Bosu
Department of Computer Science
Southern Illinois University Carbondale
Carbondale, Illinois 62901
abosu@cs.siu.edu

Fang Liu, Danfeng (Daphne) Yao, Gang Wang
Department of Computer Science
Virginia Tech.
Blacksburg, VA 24060
{fbeyond, danfeng, gangwang}@vt.edu

***Abstract*—We present DIALDroid, a scalable and accurate tool for analyzing inter-app Inter-Component Communication (ICC) among Android apps, which outperforms current state-of-the-art ICC analysis tools. Using DIALDroid, we performed the first large-scale detection of collusive and vulnerable apps based on inter-app ICC data flows among 110,150 real-world apps and identified key security insights.**

1. Introduction

Although majority of the research on Android security to-date focused on identifying stand-alone malicious apps, recent studies have demonstrated more complex android security threats associated with two or more apps [1], [4].

Android framework provides a message passing interface named, Inter-Component Communication (ICC) to facilitate data exchange between the components of the same app or different apps. However, malware writers can exploit ICC interface to develop multiple collusive apps that may look benign to single-app screening mechanisms. There are primarily two types of threats associated with inter-app ICC.

- *Privilege escalation* (aka the confused deputy problem) is a well-defined threat where the receiver app B gains unauthorized permissions or sensitive data as a result of its ICC communications with the sender app A.
- *Collusive data leak* as a threat where the receiver app B exfiltrates the sensitive data obtained from its ICC communications with the sender app A to an external destination (e.g., via disk output or network output).

The identification of inter-app ICC threats require pairwise app comparisons, which demands worst-case quadratic complexity ($O(N^2)$, where N is the total number of apps). Despite recent efforts on inter-app ICC analysis, no satisfactory solution exists that can support a *large-scale* pairwise analysis. For instance, ApkCombiner extracts suspicious inter-app ICCs by combining multiple apps into a single app, and then performs a conventional single-app data-flow analysis [4]. This approach is barely scalable, since an expensive data-flow analysis is repeated for all possible combinations of app-pairs. COVERT [1] and DidFail [3] eliminate the need for redundant data-flow analysis by analyzing each app only once. However, COVERT uses formal model checkers incurring high overhead. DidFail's experimental evaluation is small and uses an erroneous ICC intent resolution mechanism.

In this work, we develop a scalable and accurate tool DIALDroid [2][1] for inter-app ICC analysis. We use DIALDroid to perform the first systematic large-scale security analysis on inter-app data-flows among 110,150 apps, including 100,206 most popular apps from the Google Play, and 9,944 malware apps from the Virus Share. DIALDroid completes such a large-scale analysis within a reasonable time frame (6,340 total hours of program analysis and 82 minutes of ICC linking and detection). Our key design characteristics include an adaptive and pragmatic data flow analysis, highly precise ICC resolution, fast ICC matching, and ability to execute fast queries on an optimized relational database.

2. Definitions

This section defines the key security terms used in this paper, including ICC exit leak, ICC entry leak, and sensitive ICC channel. Using these terms, then we define the two types of threats mentioned Section 1.

A sensitive ICC channel refers to an ICC link between two components, from an ICC exit point (i.e., an outgoing ICC such as `startActivity` and `bindService`) to an ICC entry point (i.e., an incoming ICC such as `onActivityResult` and `getIntent`) that transfers intents[2] containing sensitive information.

An ICC exit leak indicates a data-flow path from a sensitive API call (e.g., getLatitude and getDeviceId) to the ICC exit. In the context of inter-app ICCs, ICC exit leaks describe the sender apps. Intuitively, ICC exit leaks identify sender apps that leak sensitive data via inter-app ICCs.

An ICC entry leak indicates a data-flow paths from an ICC entry point (i.e., method calls to retrieve incoming intent objects such as getIntent) to sensitive sinks that send the received data externally (e.g, via networks). In the context of inter-app ICCs, we use the ICC entry leak to describe the receiver app. Intuitively, ICC entry leaks identify receiver apps that leak received data externally.

[1] **D**atabase powered **I**CC **A**na**L**ysis for an**Droid**
[2] message objects containing data and characteristics of intended receiver(s)

Collusive data leak is a threat associated with a sensitive ICC channel between a sender component A in a sender app and a receiver component B in another app, where A has an ICC exit leak and B leaks the received data from A via an ICC entry leak.

Privilege escalation is a threat associated with a sensitive inter-app ICC channel between a sender component A in an app and a receiver component B in another app, where A has an ICC exit leak and B does not have the permission to access the data from A.

3. DIALDroid Overview

The workflow of DIALDroid involves four key operations as follows.

- ICC ENTRY / EXIT POINT EXTRACTION: Given an app, we extract the permissions and the attributes of the intent filters from the `AndroidManifest.xml` file. We perform static analysis using our custom ICC extractor [2] to determine the attributes of the intents passing through ICC exit points.
- DATAFLOW ANALYSIS: We use static taint analysis to determine ICC exit leaks and ICC entry leaks in an app.
- DATA AGGREGATION: We aggregate the data extracted in previous two steps to store in a relational database.
- ICC LEAK CALCULATION: Using SQL stored procedures and SQL queries, we compute ICCs with collusive data leaks and privilege escalations based on fine-grained security policies.

DIALDroid executes the first three steps once for each app (complexity $O(N)$, where N is the total number of apps being analyzed). The complexity of ICC leak calculator is $O(mN)$, where m is the number of apps with ICC exit leaks and in the worst case, $m = N$. However, for real-world apps m would several times smaller than N.

4. Evaluation and Results

Table 1 shows the benchmark comparison results of DIALDroid against four other popular Android ICC analysis tools based on three benchmark suites (i.e., DroidBench 3.0, DroidBench-IccTA, and ICC-Bench. On the inter-app testcases, DIALDroid has the highest precision (100%), the highest recall (91.2%), and the highest F-measure (0.95) among the three tools supporting inter-app analysis. On the intra-app testcases, DIALDroid has the highest precision and the highest F-measure among the five tools. The detailed results of our evaluation is reported in our upcoming publication [2].

We use DIALDroid to analyze the sensitive inter-app ICCs among 100,206 apps from the Google Play Market, and characterize them into 6 threat categories (in Table 2). Our threat categorization is based on threat types (collusive data leak or privilege escalation) and intent types (explicit, where recipient is explicitly named or implicit, where only a general action/datatype is declared).

TABLE 1. COMPARISON OF DIALDROID AGAINST FOUR ICC ANALYSIS TOOLS

	Inter-app ICC			Intra-app ICC		
	Precision	Recall	F-measure	Precision	Recall	F-measure
DIALDroid	100%	91.2%	0.95	93.1%	74.4%	0.83
IccTA	100%	12.5%	0.22	83.7%	81.4%	0.82
COVERT	3.3%	45.8%	0.06	83.3%	25.6%	0.39
DroidSafe*	-	-	-	53.7%	100%	0.70
AmanDroid*	-	-	-	77.4%	55.8%	0.65

*DroidSafe and Amandroid only supports intra-app ICC analysis.

TABLE 2. SUMMARY OF INTER-APP ICC THREATS IDENTIFIED BY DIALDROID

Categorization				Results			
Threat Type	Collusive Data Leak	Privilege Escalation	Intent Type	# of Distinct Source App	# of Distinct Receiver App	Total ICC Channels	Total App Pairs
I	Yes	Yes	Explicit	0	0	0	0
II	No	Yes	Explicit	0	0	0	0
III	Yes	No	Explicit	0	0	0	0
IV	Yes	Yes	Implicit	33	1,792	77,104	16,712
V	No	Yes	Implicit	62	44,514	1,785,102	1,032,321
VI	Yes	No	Implicit	21	1,040	34,745	6,783

We found that collusive data leaks and privilege escalations mostly use implicit intents but did not observe any explicit-intent based collusion. This findings suggest that collusive data leak research should start to examine implicit intents, rather than focusing on explicit intents.

Although the total numbers of sensitive ICCs and app pairs are extremely high, the number of sender apps initiating these ICCs is surprisingly small. E.g., 1,785,102 inter-app ICCs exhibiting privilege escalation behaviors (without collusive data leaks) originated from only 62 sender apps. We also had similar observations in other threat categories. We found that the majority of inter-app ICCs (> 99%) do not carry any sensitive data. This property implies that the typical workload of inter-app ICC analysis is much lower than the worst case workload. Our upcoming publication [2] details these results with case studies describing real-world collusive apps.

Our datasets and tools can potentially benefit the broader Android community. We have open-sourced our entire toolsuite on GitHub[3] and have made our database available[4] for other researchers. Our database contains extremely rich dataflow attributes of 100,206 apps from the Google Play and 9,944 apps from the Virus Share. We envision the database to be useful enabling more Android data analytic discoveries from the community.

References

[1] H. Bagheri, A. Sadeghi, J. Garcia, and S. Malek, "COVERT: Compositional analysis of Android inter-app permission leakage," *IEEE Transactions in Software Engineering*, 2015.

[2] A. Bosu, F. Liu, D. D. Yao, and G. Wang, "Collusive Data Leak and More: Large-scale Threat Analysis of Inter-app Communications," in *Proceedings of the ACM Asia Conference on Computer and Communications Security (AsiaCCS'17)*, 2017.

[3] W. Klieber, L. Flynn, A. Bhosale, L. Jia, and L. Bauer, "Android taint flow analysis for app sets," in *Proc. of the ACM SIGPLAN International Workshop on the State of the Art in Java Program Analysis*, 2014.

[4] L. Li, A. Bartel, T. F. Bissyandé, J. Klein, and Y. Le Traon, "ApkCombiner: Combining multiple Android apps to support inter-app analysis," in *ICT Systems Security and Privacy Protection*, 2015.

3 https://github.com/dialdroid-android/.

4 https://amiangshu.com/dialdroid/