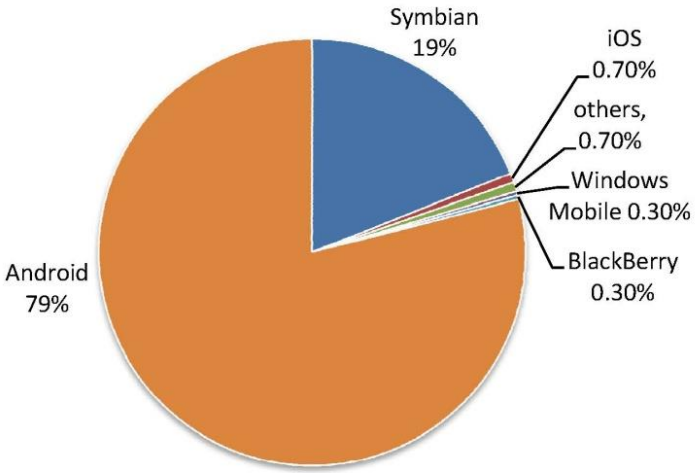
On the Need of Precise Inter-App ICC Classification for Detecting Android Malware Collusions

Karim O. Elish,
Danfeng (Daphne) Yao, and Barbara G. Ryder
Department of Computer Science
Virginia Tech

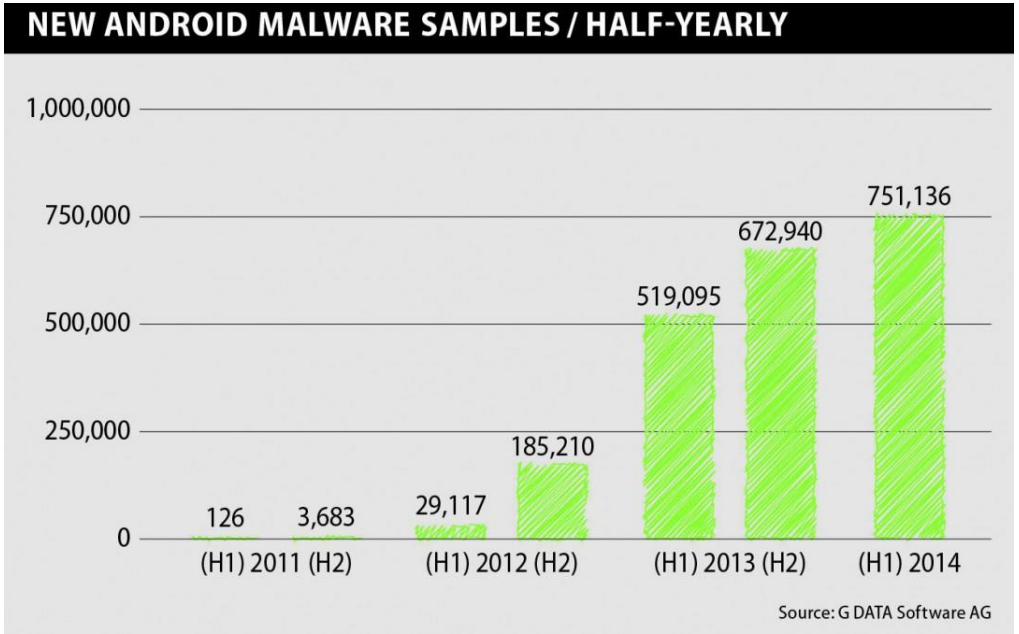
May 21, 2015

VirginiaTech
Invent the Future

Problem and Motivation



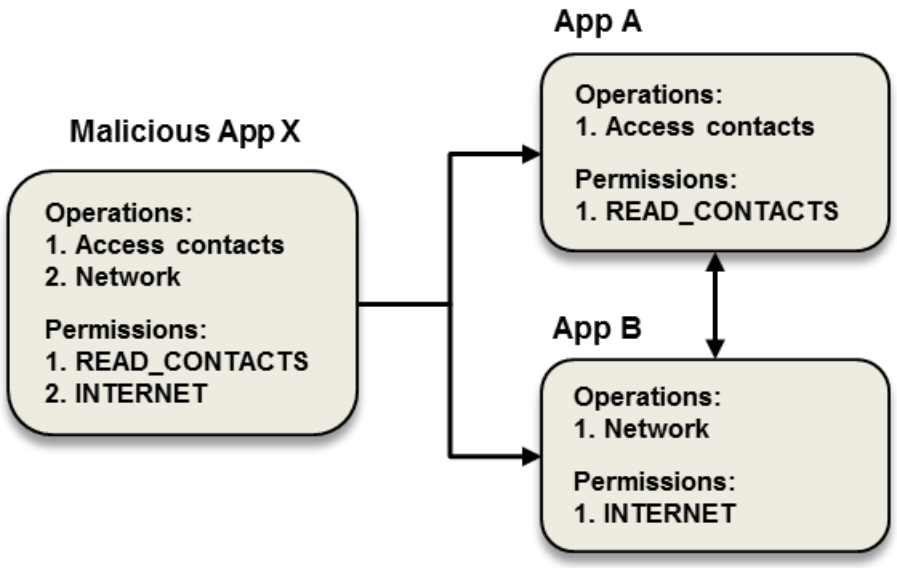
Malware Threat to Mobile OS [CIO Insight, 2012]



- Threats
  - Abuse of system resources
  - Leak of sensitive data

Malware Evolution: App Collusion

- Collusion refers to the scenario where two or more apps interact with each other to perform malicious tasks
  - Directly: Android Intent-based inter-component communication (ICC)
  - Indirectly: shared files,…etc.
- Existing solutions assume the attack model of a single malicious app, and thus cannot detect collusion



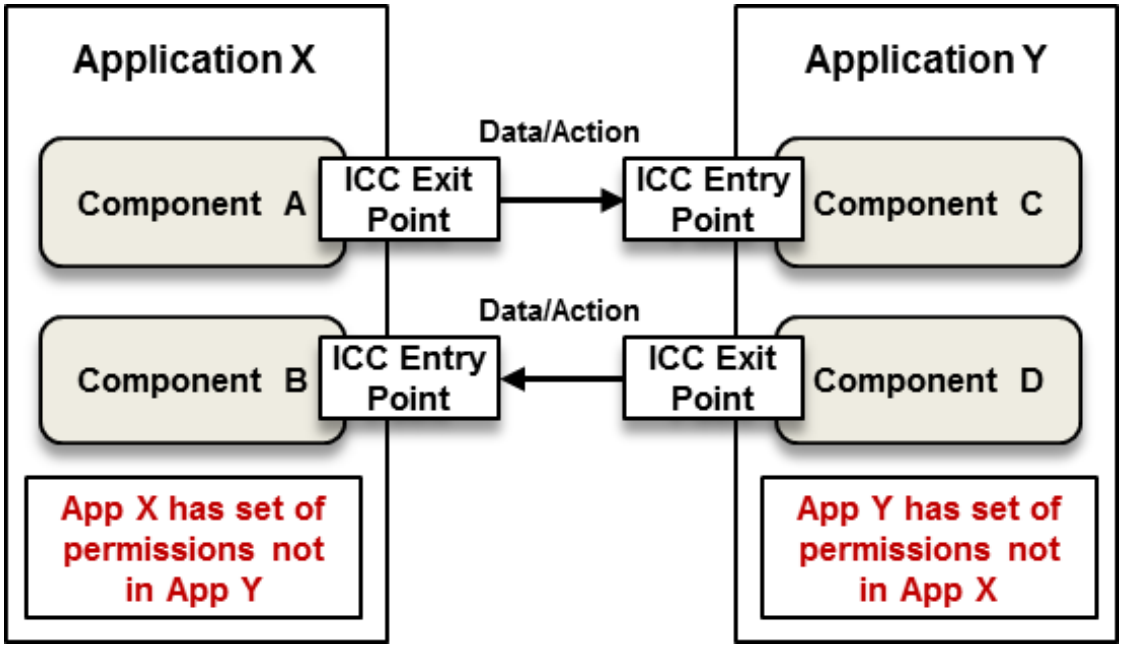
An example of permissions and operations being split between colluding apps

Existing Solutions & Limitations

Solution	Analysis Type	Collusion Classification Policies	Limitation
XManDroid [NDSS'12]	- Dynamic - Pair of apps	Permissions Combinations	- High false alerts - Scalability - Circumvented by long chain of collusion
CHEX [CCS'12]	- Static - Single app	No	- Vulnerability analysis only - Can not track data via ICC
ComDroid [MobiSys'11]	- Static - Single app	No	- Vulnerability analysis only - Can't track path from public component to critical operation -> false alerts
Epicc [USENIX13]	- Static - Single app	No	- Same as ComDroid
Amandroid [CCS'14]	- Static - Single app	No	- No analysis/info on how to connect ICC among apps

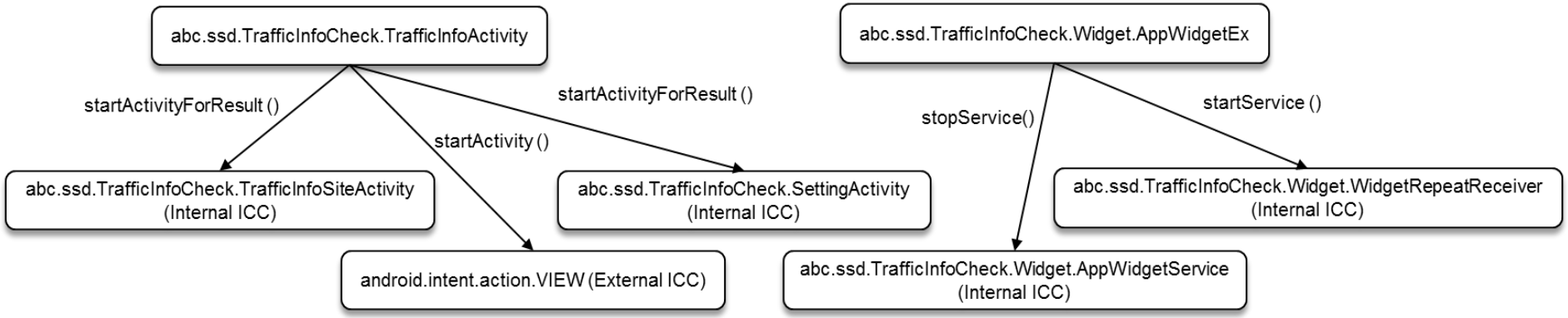
Our Goal

To characterize ICC and to experimentally demonstrate the difficulties and technical challenges associated with app collusion detection



Static Characterization of ICC

- We developed a static analysis tool (**ICC Map**) to model the Intent-based ICC of Android apps
- **ICC Map** captures all types of communication (internal and external) of an app
  - <ICCName$_k$, sourceComponent$_k$, targetComponent$_k$, typeOfCommunication$_k$>,



Partial ICC map for "abc.ssd.TrafficInfoCheck" app

Experimental Evaluation

- We statically construct **ICC Maps** of 2,644 benign apps collected from Google Play

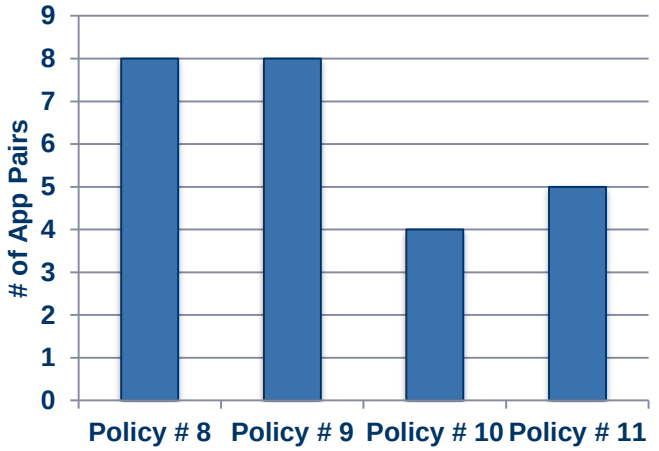
- The objectives of the study:

  1. How often do benign apps perform inter-app communications with other apps?

  2. How effective is the existing collusion detection solution (namely XManDroid) in terms of false positive rate?

Experimental Evidence

Action Used in External Implicit Intent ICC	# of Apps (%)
android.intent.action.VIEW	1870 (70.7%)
android.intent.action.SEND	943 (35.7%)
android.intent.action.DIAL	399 (15.1%)
android.intent.action.GET_CONTENT	275 (10.4%)
android.media.action.IMAGE_CAPTURE	231 (8.7%)
android.intent.action.CALL	158 (6.0%)
android.intent.action.PICK	139 (5.3%)
android.intent.action.SENDTO	122 (4.6%)
android.media.action.VIDEO_CAPTURE	62 (2.3%)
android.intent.action.DELETE	53 (2.0%)
android.intent.action.EDIT	48 (1.8%)
android.speech.action.RECOGNIZE_SPEECH	45 (1.7%)
android.intent.action.MEDIA_MOUNTED	42 (1.6%)
android.intent.action.INSERT	33 (1.2%)
android.intent.action.SEARCH	20 (0.8%)
android.intent.action.RINGTONE_PICKER	19 (0.7%)
android.intent.action.WEB_SEARCH	12 (0.5%)
android.intent.action.SYNC	3 (0.1%)
android.intent.action.ANSWER	2 (0.1%)
# of apps with external implicit Intent ICC	1932 (73.1%)
# of apps with external explicit Intent ICC	298 (11.3%)
Total # of apps with external ICC	2230 (84.4%)
Total # of apps with Internal ICC only	414 (15.6%)

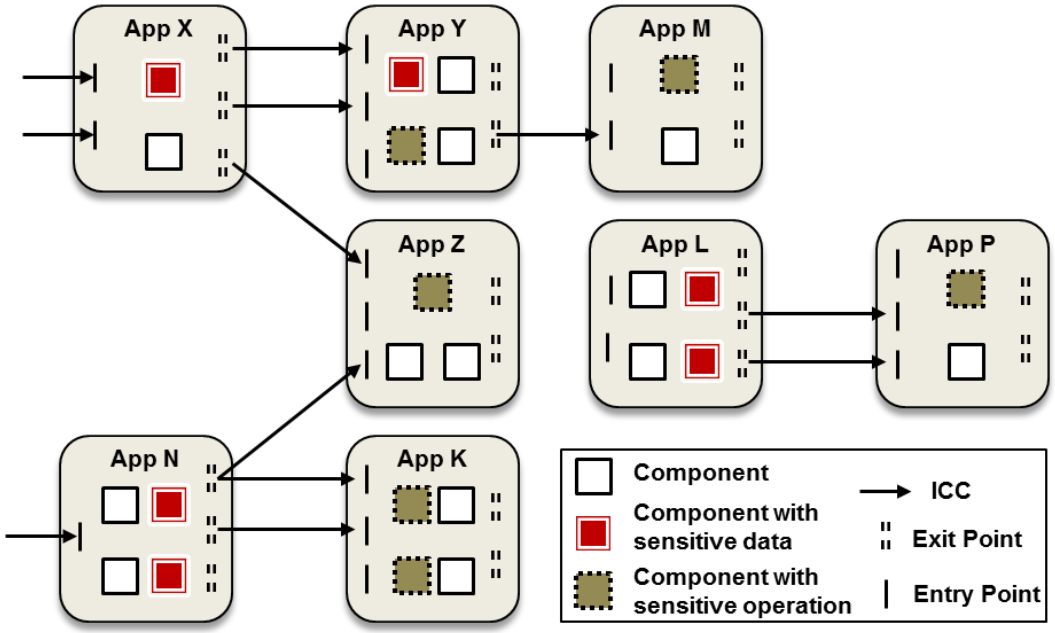


Existing collusion detection solution (XManDroid) triggers a large number of false alerts in benign app pairs (11 out of 20 benign app pairs are misclassified as collusion)

Policy	Description
8	¬ communicate(A,B) if (A has INTERNET ∧ B has ACCESS_FINE_LOCATION)
9	¬ communicate(A,B) if (A has INTERNET ∧ B has READ_CONTACTS)
10	¬ communicate(A,B) if (A has INTERNET ∧ B has READ_SMS)
11	¬ communicate(A,B) if (A has INTERNET ∧ B has RECORD_AUDIO ∧ PHONE_STATE)

Subset of XManDroid's policy

Collusion Detection: Challenges



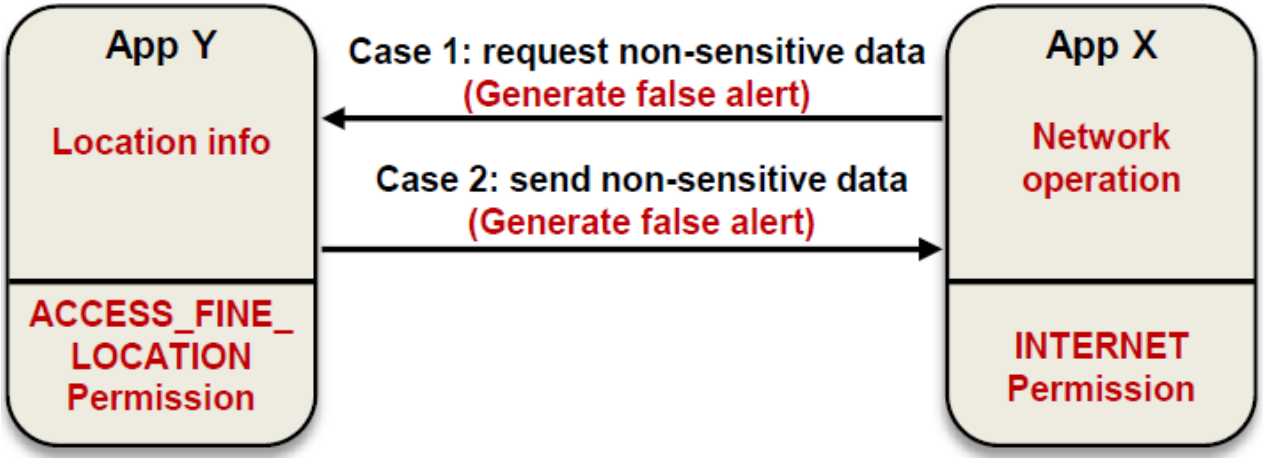
Challenges & Problems:

- Many benign apps interacts with other apps
- Analysis scalability with minimum complexity
- Existing solution produces large number of false alerts

Solution for detecting malware collusion needs:

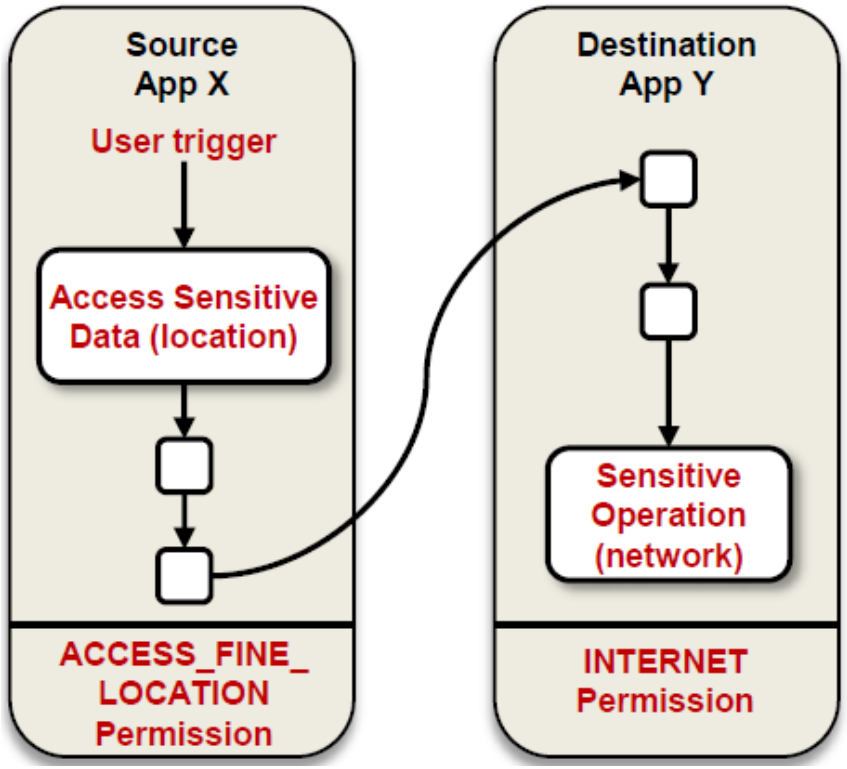
- To be able to characterize the context associated with communication channels with fine granularity
- To define security policies to classify benign ICC flows from colluding ones with low false alerts
- To be scalable to a large number of apps (e.g., tens of thousands of apps)

Improving Collusion Detection with Deep Cross-App Data-flow Analysis



- ICC involving non-sensitive data or request should NOT be alerted, despite of the sensitive permission combination (ACCESS_FINE_LOCATION and INTERNET)
- We argue that there is a need for a more practical solution based on in-depth static flow analysis that captures the context associated with the ICC

Improving Collusion Detection with Deep Cross-App Data-flow Analysis



Source App:
- ✓Access sensitive data
- ✓Send sensitive data
- ✓User triggers for accessing & sending

Destination App:
- ✓Sensitive operation

Deep static data-flow analysis in both source and destination apps (requires new program analysis algorithms and data structures)

Conclusions & Future Work

- This work demonstrates experimentally the challenges to detect malware collusion
- Future work
  - We plan to utilize our ICC Map for app collusion detection and define more fine-grained security policies to reduce false alerts
- App collusion analysis has many useful applications:
  - Enable app store to perform massive screening of the apps to detect possible collusion
  - Enable the user to check apps before installing to detect possible collusion with the pre-installed apps

Thank You…