Android Security

CS463/ECE424

University of Illinois

Agenda

- Mobile Advertising
- Permission re-delegation attacks
- Update and collusion attacks

Why In-app Advertising

Angry Birds on iPhone

- Paid $0.99
- Year 1 (one month)
  - 12,000,000 downloads
  - $8,000,000 profit (total)
- Year 4 (one month)
  - ?

Angry Birds on Android

- Free
- Year 1 (one month)
  - 8,000,000 downloads
  - $1,000,000 profit (per month)
- Year 4 (one month)
  - 100,000,000 – 500,000,000 installations
  - More than $10,000,000 profit (per month)

Why in-app advertising?

Abs workout

COST	DOWNLOADS
$1.49	10,000 – 50,000

COST	DOWNLOADS
free	10,000,000 – 50,000,000

How does it work?

- App Developer registers with ad network or ad exchange
- Receives a dev id and the ad SDK
- Includes the ad library in the application
- Includes a UI element in the app's layout
- Requests the permissions the ad network requires (Android)

Most free apps rely on it for profit

- Main app UI

* Responsible for 65%-75% of energy usage in free applications!

Ad component

AdRisk: Overview

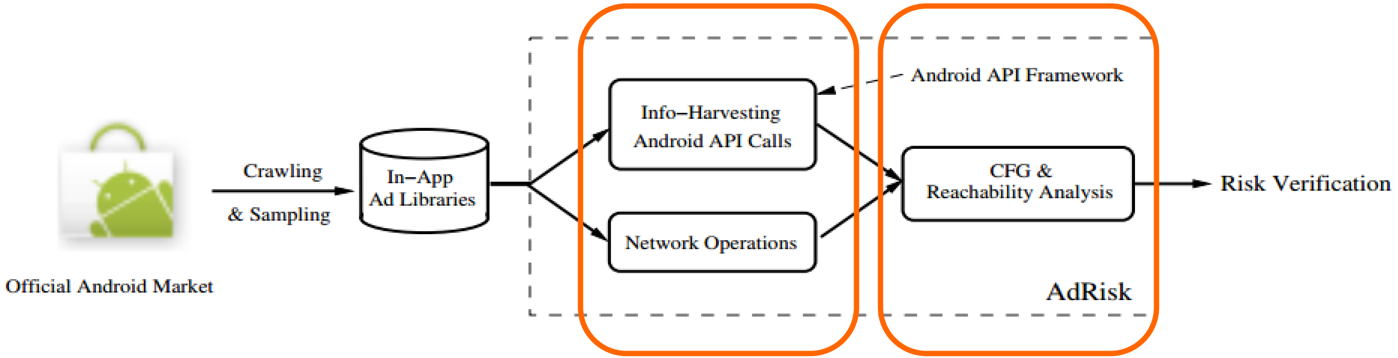
- **Problem**: assessing security and privacy risks of third-party advertising libraries embedded into apps
- **Approach**: the authors collected 100,000 apps, identified 100 ad libraries and statically analyzed them to assess their potential risk
- **Contributions**: found that ad libraries send sensitive information to remote servers and, fetch and run code dynamically

Ad Libraries Collection

- 100,000 apps from Google Play (March-May 2011)
  - Extract: permissions requested;
  - Extract: app Java class tree hierarchy
  - Candidate Set (CS) includes those apps with Internet permission; Ad Set (AS) is initially empty
    - Randomly select one app from CS and **disassemble**
    - If it contains a new ad library
      - add to AS; store its ad library class hierarchy (as a signature)
      - remove all apps in CS with this class hierarchy
    - Repeat until |AS| = 100
  - 100 ad libraries present in 52.1% of all apps

AdRisk

- Step 1
  - Identify dangerous APIs
  - Identify sinks
- Step 2
  - Identify possible risks



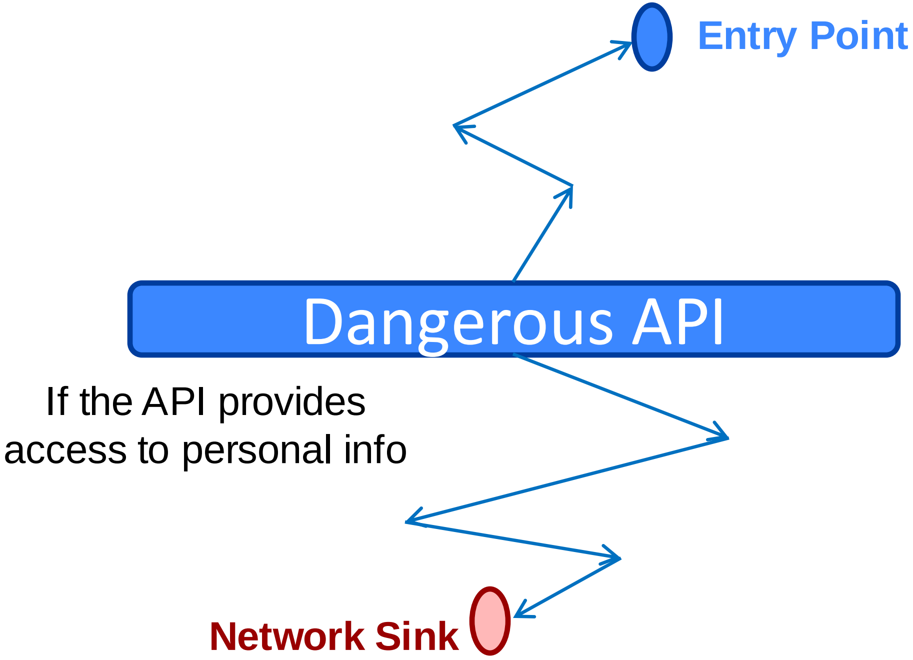
Identifying Dangerous APIs

- Analysis of
  - Android documentation
  - Android source code
- Annotate APIs with permissions they require
- ClassLoader APIs and use of java.lang.reflect package can also be dangerous
- Permissions found: 34 dangerous, 26 signature, 11 signatureOrSystem, 5 normal

Identifying possible risks (1/2)

- Dangerous behaviors
  - Can be triggered from one of many entry points
  - It is dangerous if:
    - There exists a path from an entry point to an API call that can cost the user money (e.g sending an sms) or,
    - There exists a path from entry point to an API call that allows access to personal info AND there exists a path from that API call to a sink (e.g, sending data over the Internet)

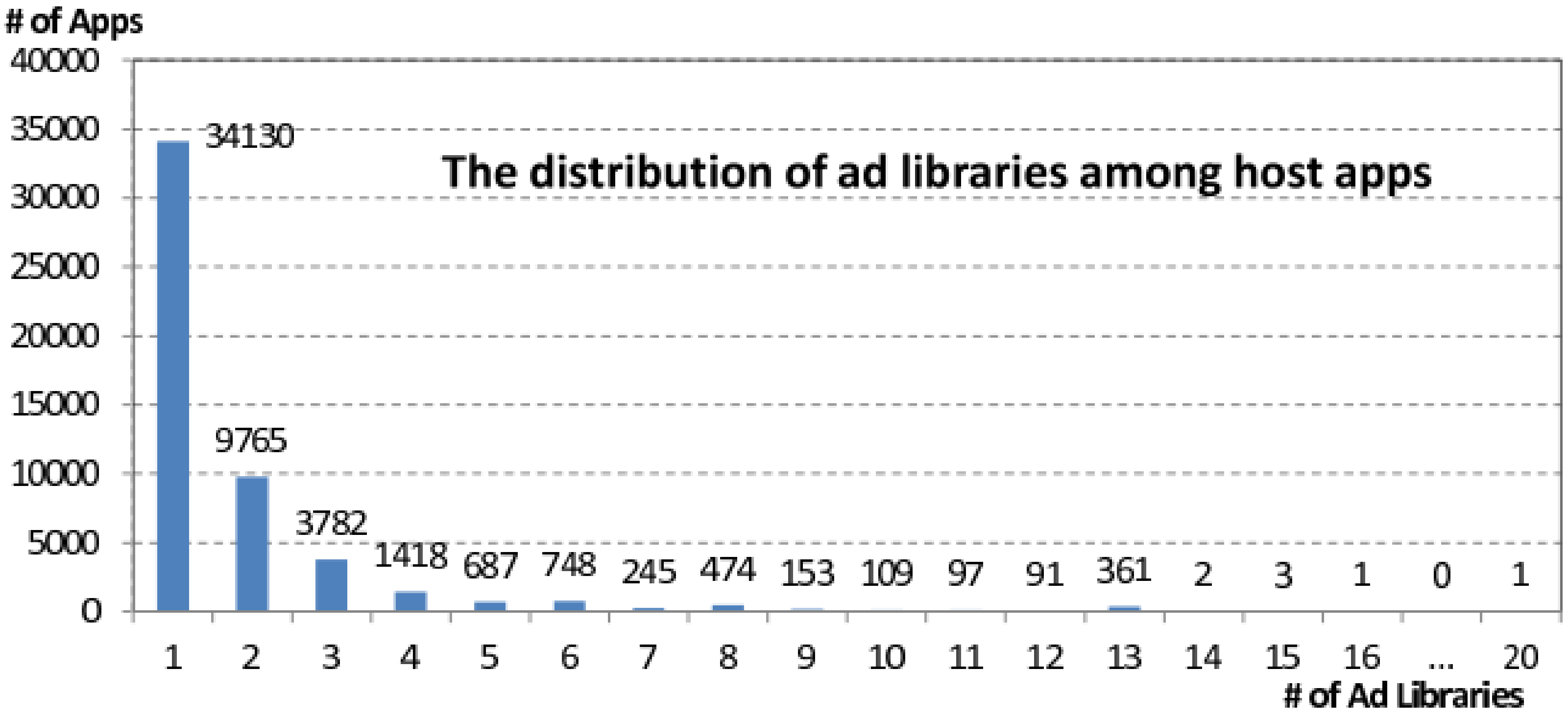
Identifying possible risks (2/2)



AdRisk Output

- Potentially-feasible paths
- Use of reflection
  - Java.lang.reflect
- Dynamic code loading
  - Class Loader
- Permission probing
  - Ad networks opportunistically check for permissions
- JavaScript linkages
  - Wrap Android API’s with JS and expose it to rich-media apps
- Reading list of installed packages (apps)

Results



# of Apps
40000
35000
30000
25000
20000
15000
10000
5000
0
34130
9765
3782
1418
687
748
245
474
153
109
97
91
361
2
3
1
0
1
The distribution of ad libraries among host apps
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
...
20
# of Ad Libraries

Results

	Included in Apps	Probes Permissions	Uses Obfuscation	Uses Reflection	Uses JavaScript	Read Installed Packages	Location Data	Place Phone Call	Camera	List Accounts	Read Calendar	Read Contact/Call Logs	Read Browser Bookmarks	Read Phone Information	Read Phone Number	Read SMS	Send SMS	Change Calendar	Change Contacts	Use Vibrator	ClassLoader
admob/android/ads	27235	✓	✓	✓	✓	·	✓	·	·	·	·	·	·	·	·	·	·	·	·	·	·
google/ads	16323	✓	·	✓	·	·	✓	·	·	·	·	·	·	·	·	·	·	·	·	·	·
flurry	5152	✓	✓	·	✓	·	✓	·	·	·	·	·	·	·	·	·	·	·	·	·	·
google/../analytics	4551	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
millennialmedia	4228	✓	·	·	✓	·	·	·	·	·	·	·	·	✓	·	·	·	·	·	✓	·
mobclix	4190	✓	·	✓	✓	·	✓	·	·	✓	✓	✓	·	✓	·	·	·	✓	✓	✓	·
adwhirl	3915	✓	·	✓	·	·	✓	·	·	·	·	·	·	✓	·	·	·	·	·	·	·
qwapi	1745	✓	·	·	·	·	✓	·	·	·	·	·	·	✓	✓	·	·	·	·	·	·
youmi	1699	✓	✓	·	·	·	✓	·	·	·	·	·	·	✓	·	·	✓	·	·	·	·
mobfox	1524	✓	·	·	·	·	✓	·	·	·	·	·	·	✓	·	·	·	·	·	·	·
zestadz	1514	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
cauly	1249	·	·	·	·	·	✓	✓	·	·	·	·	·	✓	·	·	·	·	·	·	·
inmobi	1229	✓	·	·	·	·	✓	·	·	·	·	·	·	·	·	·	·	·	·	·	·
wooboo	1183	✓	✓	·	·	·	✓	·	·	·	·	·	·	✓	✓	·	✓	·	·	·	·
admarvel	1101	✓	·	·	✓	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
smaato	1077	✓	·	·	✓	·	✓	·	·	·	·	·	·	✓	·	·	·	·	·	·	·
mobclick	1058	✓	·	·	·	·	✓	·	·	·	·	·	·	✓	·	·	·	·	·	·	·
jp/co/nobot	995	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
airpush	945	✓	✓	·	·	·	✓	·	·	·	·	·	✓	✓	·	·	✓	·	·	·	·

Results

- Location and IMEI
  - Targeted advertising
- Place phone call, send text message, add event to calendar
  - Only through user interaction
- Other
  - Sosceo transmits call history through the Internet
  - Some of them upload the user's phone number
  - WAPS uploads the list of all installed apps
  - Mobus reads through SMSs to determine the text-messaging service center they use

Results

- Categorization of ad libraries
  - Invasively collecting Personal Info
    - Usually employed by smaller ad networks ; SMS, call logs, list of apps e.t.c.
  - Permissively disclosing data to running ads
    - JS wrapping of Android API (user interaction)
    - gpsStart(<callback>) (no user interaction)
  - Unsafely Fetching and Loading Dynamic Code
    - One ad network allows the host app to be remotely controlled!

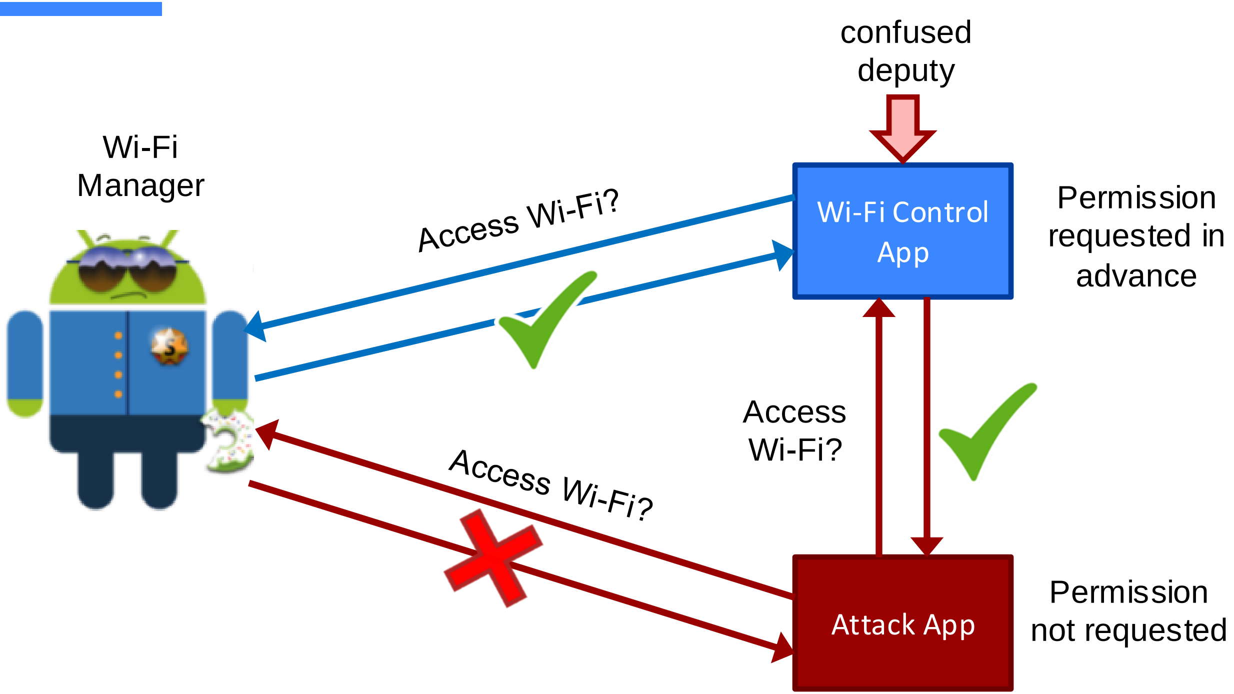
Agenda

- Mobile Advertising
- Permission re-delegation attacks
- Update and collusion attacks

Privilege Escalation Attacks on Android

- Gaining elevated access to resources that are normally protected against an unauthorized application
- 3 major classes
  - Confused deputy attacks: leveraging unprotected interfaces of benign programs
    - Permission re-delegation attacks
  - Collusion attacks: malicious applications work together to achieve their goal
  - Update attacks: vulnerabilities in software update mechanisms

Permission Re-delegation Attacks



Why Could This Happen?

- App w/ permissions exposes a public interface
  - The “deputy” app may accidentally expose privileged functionality
  - The attacker invokes it in a surprising context
    - Example: broadcast receivers in Android
  - Or intentionally expose it and but fail to correctly reduce the invoker’s authority
    - Dynamic (programmatic) permission checks
      - checkCallingPermission(), checkCallingOrSelfPermission() etc.

Public Interfaces in Android Manifest

- Via exported tag
  - <service android:name=”.WiFiService" android:exported=“true" android:permission=“com.app.MY_PERMISSION">
- Via intent filters
  - <receiver android:name=”.WiFiBroadcastReceiver">

    <intent-filter>

    <action android:name="android.intent.action.WIFI”/>

    </intent-filter>

    </receiver>

Component is still public if android:exported=“false” AND it has an intent filter!

Prevalence of Public Interfaces

- Examine 872 apps and check their **AndroidManifest.xml**
  - 16 core system apps;
  - 756 most popular free; 100 most popular paid
- 320 of these (37%) have dangerous/signature permissions and at least one type of public component
- Only 9% of all apps perform **dynamic permission checks**
  - But typically to **protect content providers** and not services or broadcast receivers
  - Only 1 application in a random set w/ 50 apps does so to protect a service or broadcast receiver
- 11 of 16 system applications are at risk

Implementing the Attack

- Constructing the attack
  - Decompile the potentially vulnerable app
  - Build call graph of the app
  - Search the call graph to find paths from **public entry points** (sources) **to protected system APIs** (sinks)
- Likely to miss some viable paths
  - Cannot detect flow through callbacks
- Only construct attacks on API calls for verifiable side effects

Case Studies

- Build attacks for 5 system apps
  - Settings: phone's primary control panel
    - Settings UI sends intent to **Settings receiver** on user's button clicks 
    - Unprivileged app can also send **Intents** to this broadcast receiver
    - Requires CHANGE_WIFI_STATE, BLUETOOTH_ADMIN, ACCESS_FINE_LOCATION permissions
  - DeskClock: time and alarm functionality
    - Public **service** that accepts directions to play alarms
    - Send **Intent** to indefinitely vibrate the phone (prevent phone from sleeping)
    - Requires VIBRATE and WAKE_LOCK permissions

Defense: IPC Inspection

We need runtime independence and ability of reduction of privileges!

- Ideas borrowed from:
  - Stack inspection
    - When a privileged API call is made, system checks within a runtime whether the call stack includes any unprivileged application. Depends on runtime (Java vs C).
  - History-based access control (HBAC)
    - Reduces the permissions of trusted code after interactions with untrusted code. Relies on runtime mechanisms.
  - Mandatory access control (MAC)
    - Central flow control by OS enforced fixed info. flow policy
    - Apps cannot be strictly ordered in terms of integrity level

Defense: IPC Inspection

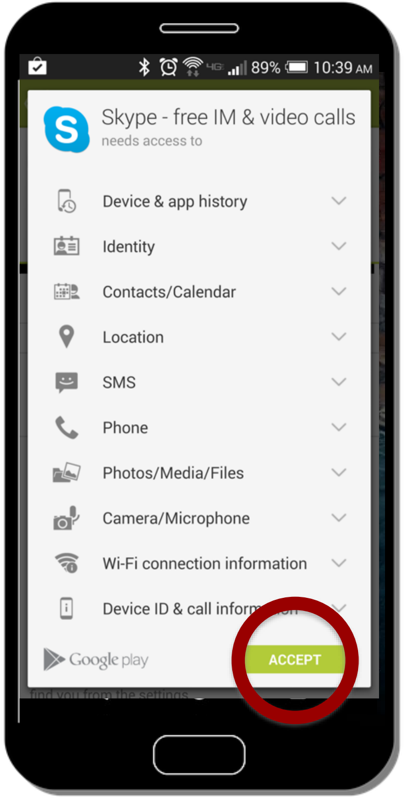
- When an app receives a message from another app, reduce the privileges of recipient to the intersection of requester's and recipient's permissions
  - Maintain a list of current permissions for each app
  - Build privilege reduction into system's IPC mechanism
  - Allow apps to accept or reject messages
    - They can register a set of acceptable requesters
    - Requesters are identified based on their permissions

Agenda

- Mobile Advertising
- Permission re-delegation attacks
- Update and collusion attacks

Permission Model Revision

Install-time Permissions
< version 6

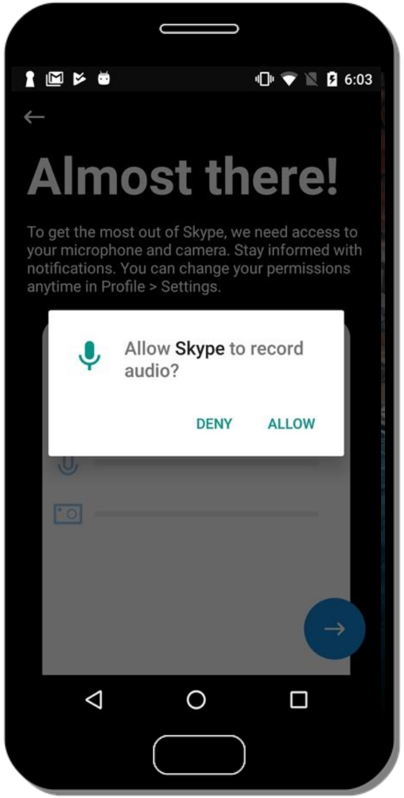


Permission Types

- Normal
- Signature
- Dangerous
- SignatureOr System

Permission Model Revision

Runtime Permissions
>= version 6

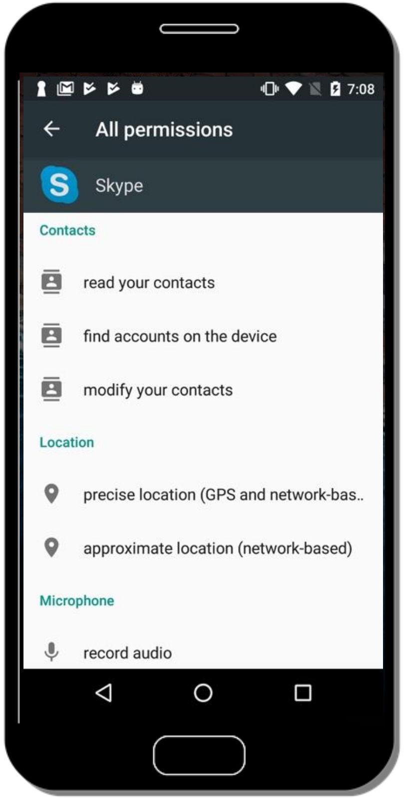


Permission Types

- Normal
- Signature
- Dangerous

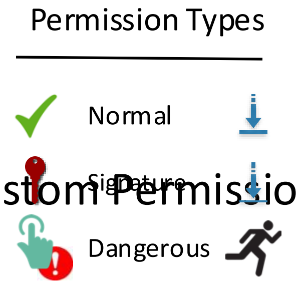
Permission Model Revision

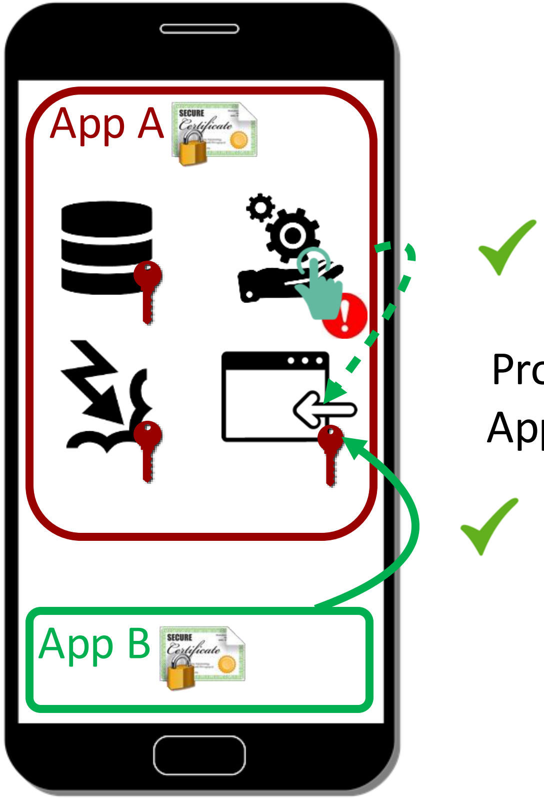
Permission Groups



Permission Model Revision

Custom Permissions





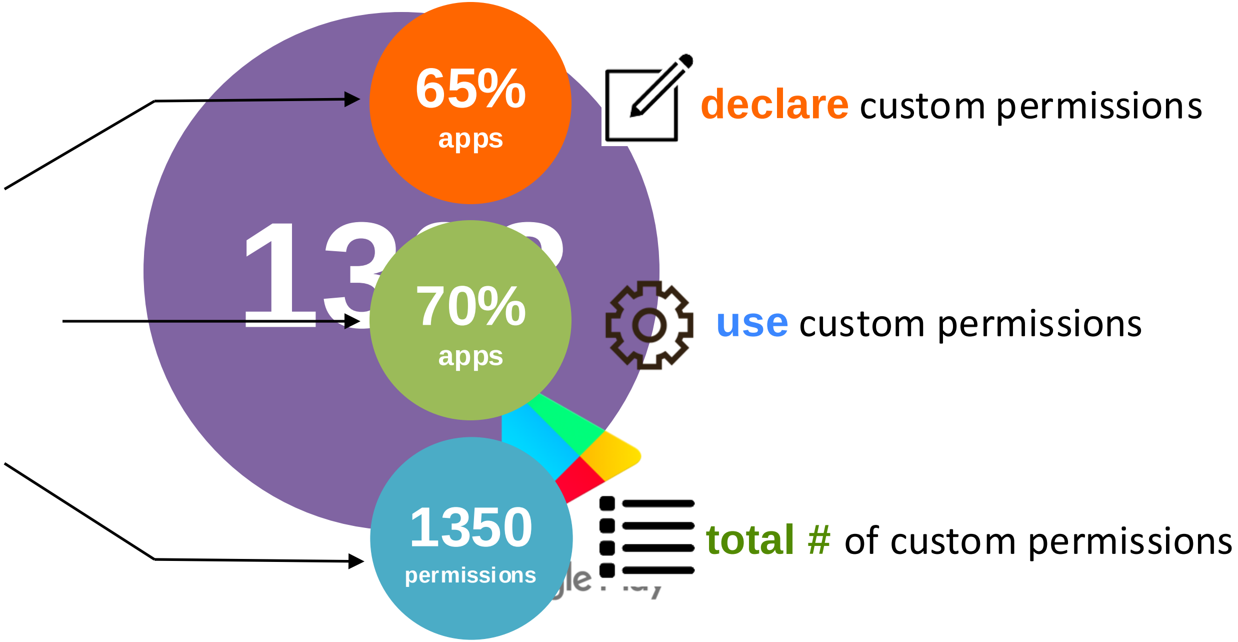
Protect Exported App Components

Prevalence of Custom Permissions

1308

Google Play

Prevalence of Custom Permissions



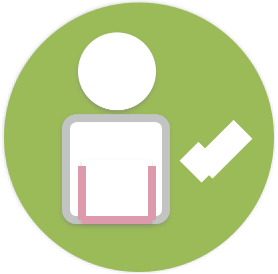
Observation 1

No clear distinction between
system permissions and custom permissions

Observation 1

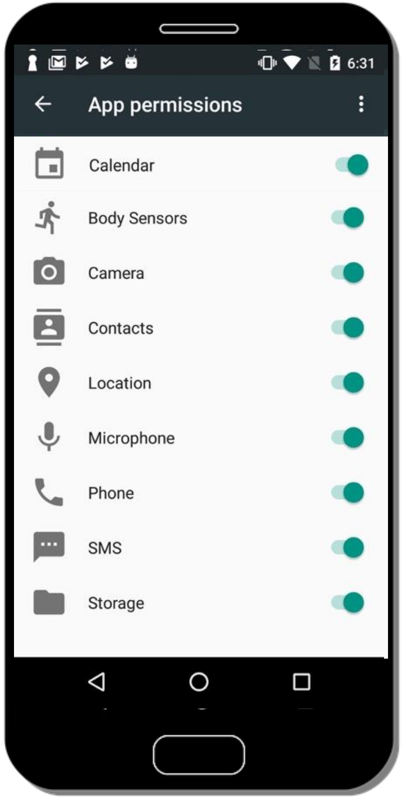
No clear distinction between
system permissions and custom permissions

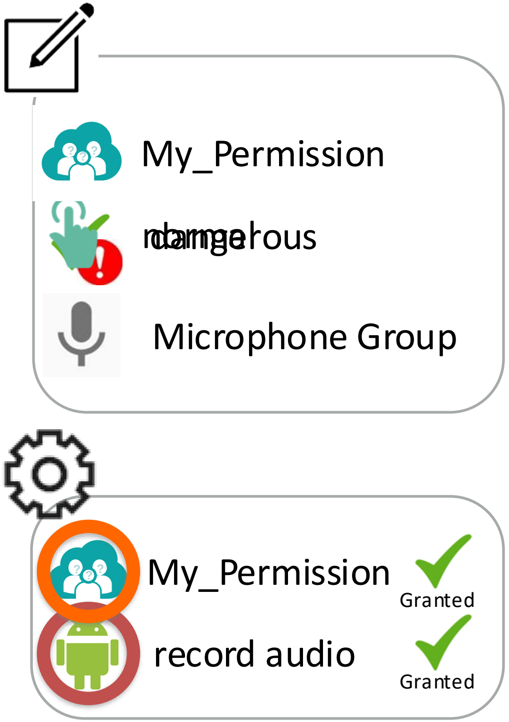
declared by
the system



declared by
3rd party apps

Privilege (Permission) Escalation Attack

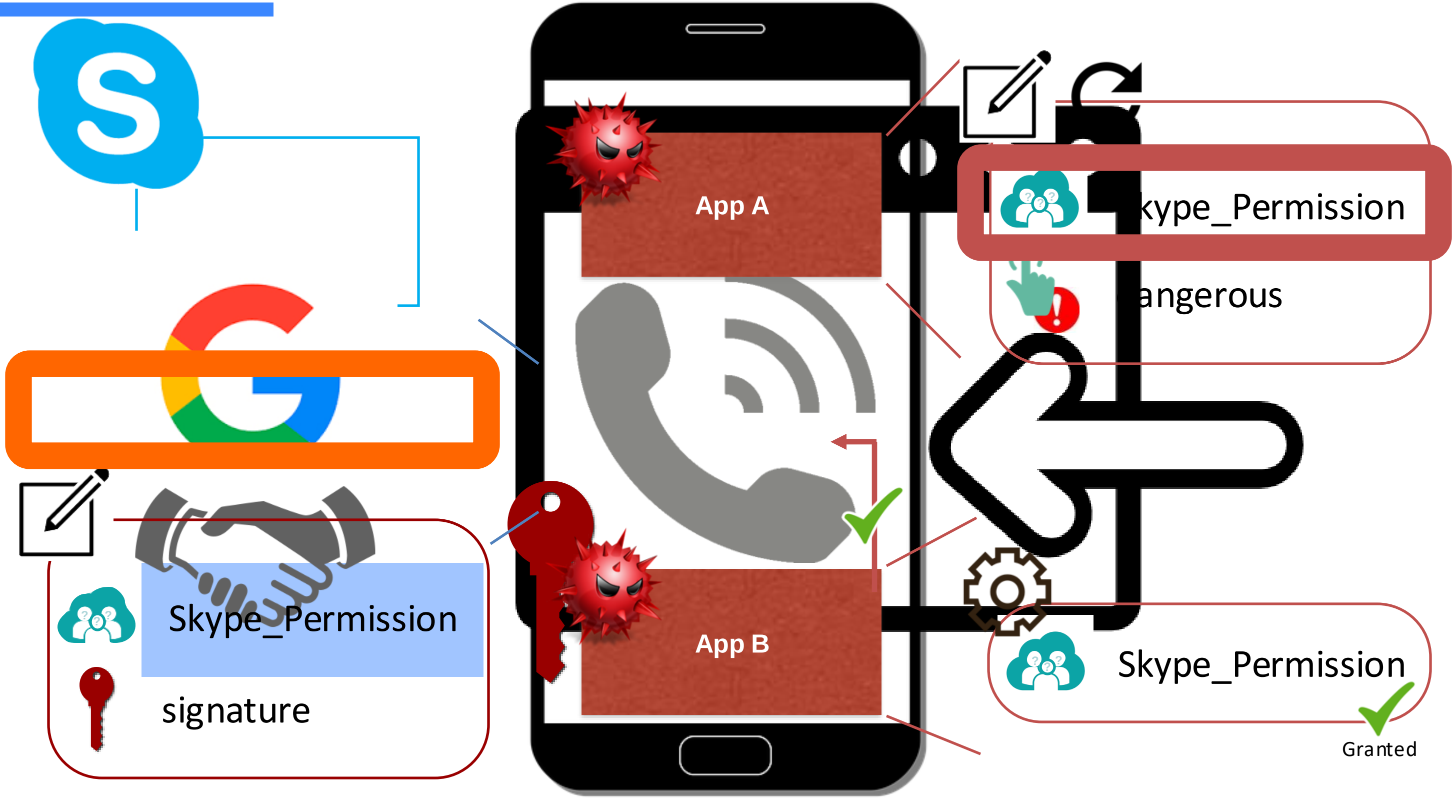




Observation 2

No distinction between
custom permissions owners

Collusion + Confused Deputy Attack



Defense

android	cusper
Decisions made by principals outside the framework's Trusted Compute Base affect enforcement at runtime —> privilege escalation	Systematically addresses the lack of **separation** of trust by decoupling system from custom permissions
Custom permissions are claimed on a FCFS basis —> spoofing	Provides a backward-compatible OS-level naming convention for tracking **ownership** of custom permissions
Software testing	**Formally verified** to be correct

Reading

- Advertising Attacks
  - [Grace, Michael C., et al. 2012] Grace, Michael C., et al. "Unsafe exposure analysis of mobile in-app advertisements." *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 2012.
  - [DemetriouNDSS16] Demetriou, Soteris, et. al. “Free for all! Assessing User Data Exposure to Advertising Libraries on Android” *Proceedings of the 23rd Annual Network and Distributed System Security Symposium (NDSS)*. 2016.
- Permission Re-Delegation Attacks
  - [FeltUSENIX11] Felt, Adrienne Porter, et al. "Permission Re-Delegation: Attacks and Defenses." *USENIX Security Symposium*. 2011.
  - [BugielNDSS12] Bugiel, Sven, et al. "Towards Taming Privilege-Escalation Attacks on Android." *NDSS*. 2012.
- Update and Collusion Attacks
  - L. Xing, X. Pan, R. Wang, K. Yuan, and X. Wang, “Upgrading your Android, Elevating my Malware: Privilege Escalation through Mobile OS Updating,” in IEEE Security and Privacy, 2014.
  - [Tuncay, Güliz Seray et al. 2018] Tuncay, Guliz Seray, Soteris Demetriou, Karan Ganju and Carl A. Gunter. ”Resolving the Predicament of Custom Permissions." *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS)*. 2018.

Discussion Questions

- Why are we not making all app components private to protect apps from privilege escalation attacks?
- Does IPC inspection have an impact on application developers?
- What kind of **apps** would you be more comfortable sharing your data with? Are there any apps you are not comfortable sharing your data with?
- Other kinds of attacks on smartphones?