Building a Next-Generation App Store

Jason Franklin Ph.D.
jfrankli@cs.stanford.edu
Research Associate
Stanford University
App Store: Promise

Safe, Trusted, Centralized
"Permissions changed in the latest update to read my phone number. Totally unacceptable for a puzzle game. Uninstalling." [1]

"Uninstalling due to the added permissions." [1]

"Why suddenly Read phone state permission?" [1]

"Simple and challenging game but with new update there is too many Permissions for a simple game, will not be updating and once completed all levels I will be deleting it." [1]

Anatomy of an App Store

Developers

Admission System

Storage

Users

Submit

Accept

Distribute

exec()
STAMP Admission System

Static Analysis
More behaviors, fewer details

Dynamic Analysis
Fewer behaviors, more details
STAMP as a Service

Analyst

STAMPEngine

App
Summary

Logs
Updates

Custom Android Impl.
Dynamic Analysis Results
Static vs. Dynamic Analysis

- **Can be faster**
  - Google Bouncer (dynamic): 300 seconds
  - Unopt. STAMP: ~40 seconds

- **Detects more behaviors**
  - Closer to 100% coverage

- **No code execution**
  - Avoid configuration issues, VMs/emulators, and input generation while always getting results
Data Flow

Source-to-sink flows

- Sources: Location, Calendar, Contacts, Device ID etc.
- Sinks: Internet, SMS, Disk, etc.
Data Flow Analysis in Action

- **Malware/Greyware Analysis**
  - Data flow summaries enable enterprise-specific policies

- **API Misuse and Data Theft Detection**

- **Automatic Generation of App Privacy Policies**
  - Avoid liability, protect consumer privacy

- **Vulnerability Discovery**

---

Privacy Policy
This app collects your:
- Contacts
- Phone Number
- Address

---
Challenges

- Android is 3.4M+ lines of complex code
  - Uses reflection, callbacks, native code

- **Scalability**: Whole system analysis impractical

- **Soundness**: Avoid missing flows

- **Precision**: Minimize false positives
STAMP Approach

- Model Android/Java
  - Sources and sinks
  - Data structures
  - Callbacks
  - 500+ models

- Whole-program analysis
  - Context sensitive

Too expensive!
Building Models

- 30k+ methods in Java/Android API
  - Reimplement w. minimum necessary details

- Follow the permissions
  - 20 permissions for sensitive sources
    - ACCESS_FINE_LOCATION (8 methods with source annotations)
    - READ_PHONE_STATE - (9 methods)
  - 4 permissions for sensitive sinks
    - INTERNET, SEND_SMS, etc.
Identifying Sensitive Data

- Returns device IMEI in String
- Requires permission GET_PHONE_STATE

```java
android.Telephony.TelephonyManager: String getDeviceId()

@STAMP(
    SRC ="$GET_PHONE_STATE.deviceid",
    SINK ="@return"
)
```
Data We Track (Sources)

- Account data
- Audio
- Calendar
- Call log
- Camera
- Contacts
- Device Id
- Location
- Photos (Geotags)
- SD card data
- SMS

30+ types of sensitive data
Data Destinations (Sinks)

- Internet (socket)
- SMS
- Email
- System Logs
- Webview/Browser
- File System
- Broadcast Message

10+ types of exit points
Currently Detectable Flow Types

396 Flow Types

Unique Flow Types = Sources x Sink
Example: Facebook Contact Sync

Contact Sync for Facebook (unofficial)

Description:
This application allows you to synchronize your Facebook contacts on Android.

IMPORTANT:
* "Facebook does not allow [sic] to export phone numbers or emails. Only names, pictures and statuses are synced."
* "Facebook users have the option to block one or all apps. If they opt for that, they will be EXCLUDED from your friends list."

Privacy Policy: (page not found)
# Contact Sync Permissions

<table>
<thead>
<tr>
<th>Category</th>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Your Accounts</td>
<td>AUTHENTICATE_ACCOUNTS</td>
<td>Act as an account authenticator</td>
</tr>
<tr>
<td></td>
<td>MANAGE_ACCOUNTS</td>
<td>Manage accounts list</td>
</tr>
<tr>
<td></td>
<td>USE_CREDENTIALS</td>
<td>Use authentication credentials</td>
</tr>
<tr>
<td>Network Communication</td>
<td>INTERNET</td>
<td>Full Internet access</td>
</tr>
<tr>
<td></td>
<td>ACCESS_NETWORK_STATE</td>
<td>View network state</td>
</tr>
<tr>
<td>Your Personal Information</td>
<td>READ_CONTACTS</td>
<td>Read contact data</td>
</tr>
<tr>
<td></td>
<td>WRITE_CONTACTS</td>
<td>Write contact data</td>
</tr>
<tr>
<td>System Tools</td>
<td>WRITE_SETTINGS</td>
<td>Modify global system settings</td>
</tr>
<tr>
<td></td>
<td>WRITE_SYNC_SETTINGS</td>
<td>Write sync settings (e.g. Contact sync)</td>
</tr>
<tr>
<td></td>
<td>READ_SYNC_SETTINGS</td>
<td>Read whether sync is enabled</td>
</tr>
<tr>
<td></td>
<td>READ_SYNC_STATS</td>
<td>Read history of syncs</td>
</tr>
<tr>
<td>Your Accounts</td>
<td>GET_ACCOUNTS</td>
<td>Discover known accounts</td>
</tr>
<tr>
<td>Extra/Custom</td>
<td>WRITE_SECURE_SETTINGS</td>
<td>Modify secure system settings</td>
</tr>
</tbody>
</table>
Possible Flows from Permissions

Sources

- READ_CONTACTS
- READ_SYNC_SETTINGS
- READ_SYNC_STATS
- GET_ACCOUNTS
- INTERNET

Sinks

- INTERNET
- WRITE_SETTINGS
- WRITE_CONTACTS
- WRITE_SECURE_SETTINGS
- WRITE_SETTINGS
Expected Flows

Sources
- READ_CONTACTS
- READ_SYNC_SETTINGS
- READ_SYNC_STATS
- GET_ACCOUNTS
- INTERNET

Sinks
- INTERNET
- WRITE_SETTINGS
- WRITE_CONTACTS
- WRITE_SECURE_SETTINGS
- WRITE_SETTINGS
Observed Flows

FB API -> Source: FB_Data -> Write Contacts -> Sink: Contact_Book

Read Contacts -> Source: Contacts -> Send Internet -> Sink: Internet
Conclusion

- Exploring space of admission systems
- Fast, practical static data flow analysis
- Dynamic analysis collects concrete values
  - Users test drive apps, we collect data
- Warning system identifies violated assumptions
  - Dynamic code loading, reflection, and anti-analysis techniques
Interested in STAMP?

Contact:
Jason Franklin
jfrankli@cs.stanford.edu

Credits:
Alex Aiken,
John Mitchell,
Saswat Anand,
Osbert Bastani,
Lazaro Clapp,
Patrick Mutchler,
Manolis Papadakis