Motivation

• Mobile security is becoming increasingly important
  ○ In 2013, there are over 1 billion smartphone users around the globe another billion users by 2015
  ○ F-Secure identified 275 new mobile threat families in Q1 2014, up from 149 last year

• Mobile privacy is a leading concern
  ○ Over 50% of the Android malware has some private information collection capabilities
Threat Model

● Mobile privacy: Leakage of personal or sensitive information
  ○ GPS coordinates
  ○ Audio recordings
  ○ Contacts list
  ○ SMS messages

● Not focusing on:
  ○ Attacks that tries to take over the device
  ○ Phishing, social engineering attacks
  ○ Denial of Service
Android Permission

- Coarse-grained permission system
  - Possible to hide malicious behavior
- Weak enforcement
  - All or nothing
Malware Example

Kittey Kittey

- A real Android malware, designed to evade detection tools

READ_FILESYSTEM    INTERNET_ACCESS
Approach

An enforcement tool that allows users to enforce fine-grained privacy policies on a given mobile app.

Design challenges:

- What is a easy-to-write and expressive syntax for privacy policies?
- How to build a tool that precisely and effectively enforce these policies?
Outline

● Privacy policy
● Enforcement tool
  ○ Survey of existing techniques
  ○ Static optimized dynamic enforcement
● Implementation
● Demo
● Preliminary Evaluations
What is Privacy Policy

A specification determining how sensitive information is allowed or not allowed to be used within the app.

Components:

- **Information Flow**: how sensitive data can be exfiltrated
  - Filesystem -> Internet
  - Call logs -> SMS

- **Control Flow**: specific code paths or preconditions
  - Not allowed to upload GPS coordinate till a button is pressed
A FSM that describes both the information flow and the control flow specifications

- **State**: a list of allowed or disallowed information flows
- **Edge**: a specific program instruction that causes the state change

Audio recording is only allowed after RECORD is clicked and before STOP is pressed.
Who will write the privacy policy

- **App developer**
  - Specifies how sensitive data are used in more detail
  - “Enhanced permission system”

- **Sysadmins**
  - Apply set of default “not-allowed” policies based on app’s permission

- **User**
  - All sensitive data flow is not-allowed by default
  - Ask user’s permission when a flow first occurs
  - Next time this specific flow occurs, it will be automatically allowed or blocked
Survey of existing enforcement techniques

Metrics:

- **Precision**
  - No false positive

- **Usability**
  - Small runtime overhead

- **Practicality**
  - Automated
  - Does not require modification to the runtime system
Static Analysis

Unsafe App

Privacy Policies

Static Analysis

Potential Violations

Manual Analysis

Before Runtime

INPUT
Inlined Dynamic Enforcement

Privacy Policies

Unsafe App

Inlined Dynamic Enforcement Instrumentation

Dynamically Enforced App

Execution

INPUT

Before Runtime

High Overhead
Runtime Dynamic Enforcement

Privacy Policies

Unsafe App

Next Instruction

Virtual Machine Interpreter

Runtime Dynamic Enforcement Instrumentation

Execute

Not Portable

During Runtime

INPUT

Unsafe App

Next Instruction

Virtual Machine Interpreter

Runtime Dynamic Enforcement Instrumentation

Execute

Not Portable

During Runtime

INPUT

Unsafe App
## Comparison

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Runtime Overhead</th>
<th>Portable</th>
<th>False Positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Analysis (Conservative)</td>
<td>N/A</td>
<td>N/A</td>
<td>YES</td>
</tr>
<tr>
<td>Runtime Dynamic Enforcement</td>
<td>Low</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Inlined Dynamic Enforcement</td>
<td>High</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Triceratops</td>
<td>Low</td>
<td>YES</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Key idea:** Combine static analysis and inlined dynamic enforcement
Intuition

Privacy Policies

Unsafe App

Static Analysis

Unverified Region

Verified Region

Dynamically Enforced Region

Inlined Dynamic Enforcement Instrumentation

Safe App

Unsafe App

TRICERATOPS

INPUT
Static Optimized Dynamic Enforcement

- Minimizes the instrumentation needed to enforce a set of policies by using static analysis to:
  - Apply API summaries
  - Identify unsafe code regions
  - Optimize enforcement code
API Summary

- Allows static analysis to reason about API’s effect without executing the app
- Remove the need to instrument API bodys

File f=sensitiveFile
String x= Long.toString(f.lastModified()) ➡️ String x= f
uploadToInternet(x)

Long.toString(long) ➡️ if (parameter.isSensitive)
File.lastModified() return Sensitive
else
return NotSensitive
Identify Unsafe Code Regions

- Because the tool knows exactly what data flow it needs to track
- Conservatively identify code regions that help compute or propagate data from the source
Example

Filesystem -> Internet

c=getFile() → c=null → fieldA=c → fieldA=null → fieldA=b → b="b"

x="c" → x=fieldA → uploadToInternet(x)
Example

Filesystem -> Internet

c=getFile()
c=null
b="a"
b="b"
felda=c
felda=null
felda=b
x="c"
x=felda

uploadToInternet(x)
Enforcement Code Optimization

- Static taint propagation
- Constant folding
- Copy propagation
- Dead code elimination
Static Taint Propagation Example

Filesystem -> Internet

c = getFile()

fieldA = c

fieldA = null

c = null

b = "a"

b = "b"

x = "c"

x = fieldA

uploadToInternet(x)
Static Taint Propagation Example

Filesystem -> Internet

c=getFile()
c=null

fieldA=c
fieldA=null
fieldA=NotSensitive

x="c"

x=fieldA

uploadToInternet(x)
Implementation

● Mainly built on top of Wala analysis framework
● Directly perform analysis on Dalvik bytecode (no need for source code)
● Use smali assembler and disassembler toolchain for instrumentation
● Existing API summary from SPARTA project
Triceratops Demo

Untrusted App

Privacy Policy

Triceratops

Safe App

Android Device
Preliminary Evaluations

- **Kittey Kittey**
  - No Filesystem -> Internet
- **SMS replicator**
  - No SMS -> SMS before a button is clicked

<table>
<thead>
<tr>
<th>App</th>
<th>No Optimization</th>
<th>API Summary Relevant Code</th>
<th>Full Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kittey Kittey</td>
<td>2757</td>
<td>75/61</td>
<td>6/4</td>
</tr>
<tr>
<td>SMS replicator</td>
<td>886</td>
<td>20/13</td>
<td>4/3</td>
</tr>
</tbody>
</table>

Enforcement Overhead (# of additional instructions)

Very low runtime overhead!
Preliminary Evaluations

<table>
<thead>
<tr>
<th>Tools</th>
<th>Kittey</th>
<th>SMS Replicator</th>
<th>Root Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android Permission System</td>
<td>![X mark]</td>
<td>![X mark]</td>
<td>No IF, CF</td>
</tr>
<tr>
<td>Pegasus [Chen’13]</td>
<td>![Sad face]</td>
<td>![Sad face]</td>
<td>Multiple code path to potential violation</td>
</tr>
<tr>
<td>TaintDroid [Enck’10]</td>
<td>![Green face]</td>
<td>![Sad face]</td>
<td>No CF</td>
</tr>
<tr>
<td>Aurasium [Xu’12]</td>
<td>![X mark]</td>
<td>![Sad face]</td>
<td>No IF</td>
</tr>
<tr>
<td>Triceratops</td>
<td>![Green face]</td>
<td>![Green face]</td>
<td>Finer-grained privacy policy IF+CF</td>
</tr>
</tbody>
</table>

Supports more types of malware
Limitations

- Classical Java static analysis challenges
  - Reflection
  - Precision of points-to analysis
- Static modeling of Android runtime behavior
  - Dynamically register a callback function to a button
- Completeness of the API summary
- Native code

- Can be addressed by other research
Future Work

- **Implicit Flow**
  - Static analysis assisted dynamic analysis can be used to track implicit flow while achieving high precision
- **Data tracking mechanisms for persistent storage mediums and side channels**
  - Databases and file systems
  - Displaying sensitive information on screen, then take a screenshot
Conclusion

A powerful **enforcement tool** that allows users to enforce **fine-grained privacy policies** on a given mobile app

- Finer-grained privacy policy (IF+CF)
  - Defend against more types of malicious apps
- Static optimized dynamic enforcement
  - Portable, low runtime overhead, and no false positives