AppProfiler: A Flexible Method of Exposing Privacy-Related Behavior in Android Applications to End Users

Sanae Rosen\textsuperscript{1}  Zhiyun Qian\textsuperscript{2}  Z. Morley Mao\textsuperscript{1}

\textsuperscript{1}University of Michigan
Ann Arbor, MI

\textsuperscript{2}NEC Labs
The Problem

- Smartphones have lots of personal data, lots of apps: privacy concerns.
- Hard to make informed decisions about what applications to install.
  - Filtering malware not enough.
  - Privacy-intrusive applications may be acceptable for some but not others.
- Goal: Let users know what their apps do, in terms of privacy-sensitive behavior.
What about existing approaches?

- Permissions are supposed to tell users how their applications behave
  - May be vague or even incorrect
  - Many so prevalent that users are likely to ignore them
  - Inflexible to modification

- Many proposals to improve the permission system
  - We focus on immediate solutions

- Many proposals protect against smartphone-specific attacks or malware
  - We focus on legitimate apps
What about existing approaches?

- Permissions are supposed to tell users how their applications behave
  - May be vague or even incorrect
  - Many so prevalent that users are likely to ignore them
  - Inflexible to modification
- Many proposals to improve the permission system
  - We focus on immediate solutions
    - Many proposals protect against smartphone-specific attacks or malware
      - We focus on legitimate apps
What about existing approaches?

- Permissions are supposed to tell users how their applications behave
  - May be vague or even incorrect
  - Many so prevalent that users are likely to ignore them
  - Inflexible to modification
- Many proposals to improve the permission system
  - We focus on immediate solutions
- Many proposals protect against smartphone-specific attacks or malware
  - We focus on legitimate apps
Our Solution

- Automatically create profiles of application behavior offline.
  - Knowledge base mapping API calls to behaviors of interest
  - Use static analysis to find these behaviors
- Provide profiles to end users
- Also useful for more broadly understanding app behavior
- Flexible: Rules/profiles can easily be adapted

Download From Market and Decompile

Static Analysis to Identify Behaviors of Interest

Create Knowledge Base

User-Friendly Profiles

Detailed Profiles

Large-Scale Analysis

Android Application
Our Solution

- Automatically create profiles of application behavior offline.
  - Knowledge base mapping API calls to behaviors of interest
  - Use static analysis to find these behaviors
- Provide profiles to end users
- Also useful for more broadly understanding app behavior
- Flexible: Rules/profiles can easily be adapted
Our Solution

- Automatically create profiles of application behavior offline.
  - Knowledge base mapping API calls to behaviors of interest
  - Use static analysis to find these behaviors
- Provide profiles to end users
- Also useful for more broadly understanding app behavior
- Flexible: Rules/profiles can easily be adapted
Our Solution

- Automatically create profiles of application behavior offline.
  - Knowledge base mapping API calls to behaviors of interest
  - Use static analysis to find these behaviors
- Provide profiles to end users
- Also useful for more broadly understanding app behavior
- Flexible: Rules/profiles can easily be adapted
Our Solution

- Automatically create profiles of application behavior offline.
  - Knowledge base mapping API calls to behaviors of interest
  - Use static analysis to find these behaviors
- Provide profiles to end users
- Also useful for more broadly understanding app behavior
- Flexible: Rules/profiles can easily be adapted

Create Knowledge Base

Download From Market and Decompile

Static Analysis to Identify Behaviors of Interest

Detailed Profiles

Large-Scale Analysis

User-Friendly Profiles

Android Application
Our Solution

- Automatically create profiles of application behavior offline.
  - Knowledge base mapping API calls to behaviors of interest
  - Use static analysis to find these behaviors
- Provide profiles to end users
- Also useful for more broadly understanding app behavior
- Flexible: Rules/profiles can easily be adapted

Download From Market and Decompile

Static Analysis to Identify Behaviors of Interest

User-Friendly Profiles

Detailed Profiles

Android Application

Large-Scale Analysis

Create Knowledge Base

Sanae Rosen, Zhiyun Qian, Z. Morley Mao
Basic Assumptions and Limitations

- We do not attempt to detect malware or applications that otherwise subvert the Android framework API
- We do not currently address native code
- We supplement (instead of replacing) the permission system
- Our target audience is privacy-concerned users who are concerned about how apps behave
Step 1: Build the Knowledge Base

- Identify high-priority API calls
- Refine mappings with domain-specific knowledge
- Frequency analysis of classes and methods
- Mapping of API call patterns to behavior labels
Step 1: Build the Knowledge Base

- Identify high-priority API calls
- Refine mappings with domain-specific knowledge
- Frequency analysis of classes and methods
- Mapping of API call patterns to behavior labels
Step 1: Build the Knowledge Base

- Identify high-priority API calls
- Refine mappings with domain-specific knowledge
- Frequency analysis of classes and methods
- Mapping of API call patterns to behavior labels
Step 1: Build the Knowledge Base

- Identify high-priority API calls
- Refine mappings with domain-specific knowledge
- Frequency analysis of classes and methods
- Mapping of API call patterns to behavior labels
Example Knowledge Base Entry

Category:
Location - Type
Subcategory: Regional data - State

FunctionCall call:
call.function.enclosingClass.name startsWith "android.location.Address"
and call.function.name == "getAdminArea"

FunctionCall call:
call.function.enclosingClass.name startsWith "android.location.Address"
and call.function.name == "getSubAdminArea"
Step 2: Apply to applications

1. Find rule matches in decompiled app
2. Identify behaviors from multiple rules (e.g. photo with no preview)
3. Identify which rules belong to ad libraries
4. Convert data into user-readable profiles
5. Simplify and focus on key behaviors
6. Technical Profiles
7. User Profiles
Step 2: Apply to applications

1. Find rule matches in decompiled app
2. Identify behaviors from multiple rules (e.g., photo with no preview)
3. Identify which rules belong to ad libraries
4. Convert data into user-readable profiles
5. Simplify and focus on key behaviors
6. Technical Profiles
7. User Profiles
Step 2: Apply to applications

- Find rule matches in decompiled app
- Identify behaviors from multiple rules (e.g. photo with no preview)
- Identify which rules belong to ad libraries
- Convert data into user-readable profiles
- Simplify and focus on key behaviors
- Technical Profiles
- User Profiles
Step 2: Apply to applications

- Find rule matches in decompiled app
- Identify behaviors from multiple rules (e.g. photo with no preview)
- Identify which rules belong to ad libraries
- Convert data into user-readable profiles
- Simplify and focus on key behaviors
- Technical Profiles
- User Profiles
Step 2: Apply to applications

- Find rule matches in decompiled app
- Identify behaviors from multiple rules (e.g. photo with no preview)
- Identify which rules belong to ad libraries
- Convert data into user-readable profiles
- Simplify and focus on key behaviors
- Technical Profiles
- User Profiles
Example Profile Excerpt

Technical Profiles

- Use GPS or network
- Latitude/longitude (broadcast receiver)
- Updates every 1s or less (activity) (Jumptap library)
- Proximity to location (activity, on click)

User-Friendly Profiles

- Gathers fairly precise location data (e.g. GPS)
- Uses more of your phone’s resources than recommended to gather location data
- Concerned with your proximity to a given location
Step 3: Make available to users (+ request feedback)

We have about 50 000 profiles available!
What benefit do they provide over permissions?

Based off the feedback submitted:

- Provide *more specific* information that is relevant to users. Examples:
  - Cookies flagged as objectionable 23% of the time vs 6% for Internet use in general
  - Behavior not covered by permissions can be objectionable
    - E.g. accelerometer (info about user movement patterns?)
  - Users care about non-malicious but privacy-intrusive behavior
    - Users not that concerned about SMS messages
    - Users mostly concerned about behavior affecting privacy
What benefit do they provide over permissions?

Based off the feedback submitted:
- Provide *more specific* information that is relevant to users. Examples:
  - Cookies flagged as objectionable 23% of the time vs 6% for Internet use in general
- Behavior not covered by permissions can be objectionable
  - E.g. accelerometer (info about user movement patterns?)
- Users care about non-malicious but privacy-intrusive behavior
  - Users not that concerned about SMS messages
  - Users mostly concerned about behavior affecting privacy
What benefit do they provide over permissions?

Based off the feedback submitted:

- Provide *more specific* information that is relevant to users. Examples:
  - Cookies flagged as objectionable 23% of the time vs 6% for Internet use in general
  - Behavior not covered by permissions can be objectionable
    - E.g. accelerometer (info about user movement patterns?)
  - Users care about non-malicious but privacy-intrusive behavior
    - Users not that concerned about SMS messages
    - Users mostly concerned about behavior affecting privacy

Sanae Rosen, Zhiyun Qian, Z. Morley Mao
Case Study - Facebook

Findings (manually confirmed):
- Intrusive and frequent location data, phone number, info about the carrier
- However, mostly occurs only in response to direct user input
- Did *not* detect SMS-related behavior, even though it asks for related permission

In short, permissions make it look worse than it really is!

More detailed/more contextual information is important to really understand what is happening
Case Study - Facebook

- Findings (manually confirmed):
  - Intrusive and frequent location data, phone number, info about the carrier
  - However, mostly occurs only in response to direct user input
  - Did *not* detect SMS-related behavior, even though it asks for related permission

- In short, permissions make it look worse than it really is!
- More detailed/more contextual information is important to really understand what is happening
Accuracy

- False negatives often occurred due to decompilation errors; better tools will have a big impact
  - 15% in random applications
  - 10% in popular applications
- False positives occur primarily due to inactive third-party libraries
  - 16% rate in random applications
  - 23% in popular applications
- Although we do not attempt to address malware, detected 59% of malicious behaviors
Accuracy

- False negatives often occurred due to decompilation errors; better tools will have a big impact
  - 15% in random applications
  - 10% in popular applications
- False positives occur primarily due to inactive third-party libraries
  - 16% rate in random applications
  - 23% in popular applications
- Although we do not attempt to address malware, detected 59% of malicious behaviors
**Accuracy**

- False negatives often occurred due to decompilation errors; better tools will have a big impact
  - 15% in random applications
  - 10% in popular applications
- False positives occur primarily due to inactive third-party libraries
  - 16% rate in random applications
  - 23% in popular applications
- Although we do not attempt to address malware, detected 59% of malicious behaviors
Large-Scale Trends
Large-Scale Trends
Large-Scale Trends
Summary

- Behavior profiles of applications can be automatically created using a knowledge base of API calls.
- These profiles allow users (as well as researchers) to better understand application behavior.
- For more information: http://appprofiles.eecs.umich.edu
- To download the app: https://play.google.com/store/apps/details?id=com.appdescriber
Questions?