Android Permissions

C. Capps

November 28, 2012

Contents

Contents

What are Android Permissions?

- Apps statically request permissions in the AndroidManifest.xml file
- No support for dynamically granting apps permissions at run-time.
- The user sees a dialog at install time, and can choose to cancel installing the app based on the requested permissions
- Relies on the user's understanding of the various permissions

Permission Categories

Permissions are organized into 3 categories:

- **Normal** - API calls that could annoy but not harm the user, e.g. SET_WALLPAPER
- **Dangerous** - API calls that could be used to charge the user money or leak private information such as READ_CONTACTS
- **Signature / System** - ability to delete application packages, control backup. Only allowed by apps signed by the manufacturer.

According to Felt, et al. [4] the most commonly checked permissions by the Android API are the following (number of methods that check these permissions):

Permission	Usage
BLUETOOTH	85
B1UETOOTH_ADMIN	45
READ_CONTACTS	38
ACCESS_NETWORK_STATE	24
WAKE_LOCK	24
ACCESS_FINE_LOCATION	22
WRITE_SETTINGS	21
MODIFY_AUDIO_SETTINGS	21
ACCESS_COARSE_LOCATION	18
CHANGE_WIFI_STATE	16

Permission System

An app makes calls to the public API (and possibly hidden classes by using reflection.) This then communicates with a system process running in a Dalvik Virtual Machine. Apps can include native C code, but the native code can't directly make API calls (need a Java wrapper.)

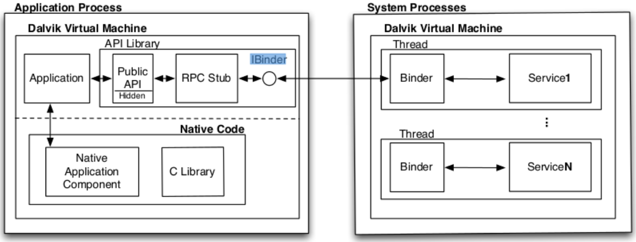


Diagram from Felt et al. [4].

Permission System

- Since permissions are checked in the system process, behavior is undefined if an app attempts to use an unauthorized permission
- Might throw a SecurityException
- Might crash the app
- Prevent a broadcast from being sent or received
- Users can create custom permissions

Contents

Study by Felt, et al.

- Not too surprisingly, users generally click past permissions warning without understanding them
- Study done by Felt et al. [5] from U.C. Berkeley, *Android Permissions: User Attention, Comprehension, and Behavior*
- Surveyed 308 Android users, and asked questions of 25 in a lab environment.
- 17% paid attentions to permissions at install-time
- 42% were completely unaware of permissions

Effective warnings

- In a paper by Baskar Sarma, et al. some guidelines for a good warning system are proposed
  1. Simple semantic meaning for users and developers
  2. Triggered by a small percentage of apps
  3. Triggered by many malicious apps
- Current system triggers too many warnings (93% of free apps have “dangerous” permissions)

Contents

Consequences of Overprivilege

- Overprivilege conditions users to accept unnecessary privileges
- Violates principle of least privilege
- Make applications more vulnerable
- More difficult to detect malicious apps with unusual permission patterns

Android Permissions Demystified

- *Android Permissions Demystified*[4]
- Experimentally determine which API calls require what permissions
  - Include private classes that developers can call using reflection
- Statically analyze Android APK files to detect overprivileged apps

Randoop

- First, they used Randoop to try calling all possible methods from a list of classes
- Modified Android kernel to log all permission checks
- Pool of input sequences, initially just primitive values
- Difficulty: generate correct input so that an exception is not thrown
  - Exception may prevent permission checks from being performed
- Difficult to get instance of every input type, seed pool of inputs with common values obtained from API
  - e.g. android.content.Context.getSystemService("wifi")

Results

- 85% coverage of API methods
- 1,259 API calls check permissions
- The API documentation only lists 78 (more at the top of classes, but very unclear)
- 6 methods are documented incorrectly

Stowaway

- Statically analyze an app, determine set of required permissions
- Examine methods that are invoked, directly or through reflection
- Many challenges, e.g. using a WebView requires the INTERNET permissions
- `android-permissions.org`

Results

- 35.8% of applications are overprivileged
- 56% of overprivileged applications use 1 extra permissions
- 94% use 4 or fewer extra permissions
- Most common unnecessary privileges:

Permission	Usage
ACCESS_NETWORK_STATE	16%
READ_PHONE_STATE	13%
ACCESS_WIFI_STATE	8%
WRITE_EXTERNAL_STORAGE	7%
CALL_PHONE	6%
ACCESS_COARSE_LOCATION	6%
CAMERA	6%
WRITE_SETTINGS	5%
ACCESS_MOCK_LOCATION	5%
GET_TASKS	5%

Contents

Permission Patterns

- *Mining Permission Request Patterns from Android and Facebook Applications*
- Paper by Mario Frank, et al. at U.C. Berkeley, rigorous statistical analysis of permission patterns [1]
- Android and Facebook permission patterns
- Determine riskiness of an app based solely on permissions used

Permission Patterns

- Statistically find permission patterns used by high reputation apps
- Whitelist apps with ordinary patterns, warn users about unusual patterns
- Used 188,389 apps for analysis
- Web-crawled the web version of the Android market, parsed HTML to get permissions used, number of ratings, average rating, cost, etc.

Most commonly requested permissions

15 most requested Android permissions (Mario Frank, et al.) [1]

requested	permission name
69.76%	Network communication : full Internet access
43.24%	Network communication : view network state
30.26%	Storage : modify/delete USB storage & SD card contents
26.47%	Phone calls : read phone state and identity
18.34%	Your location : fine (GPS) location
16.89%	Your location : coarse (network-based) location
16.16%	Hardware controls : control vibrator
15.01%	System tools : prevent device from sleeping
8.22%	Network communication : view Wi-Fi state
8.11%	System tools : automatically start at boot
6.71%	Services that cost money: directly call phone numbers
6.27%	Your personal information : read contact data
5.59%	Hardware controls : take pictures and videos
4.61%	System tools : set wallpaper
3.9%	System tools : retrieve running applications

Boolean Matrix Factorization

- Goal: find statistically significant permission request patterns
- Input: binary matrix $\mathbf{x}$ where $\mathbf{x_{id}} = 1$ means app i requests permission d.
- Output: number of statistically significant patterns, K
- Matrix $\mathbf{z}$ — the permission patterns in each app
- Matrix $\mathbf{u}$ — the statistically significant permission request patterns

Boolean Matrix Factorization

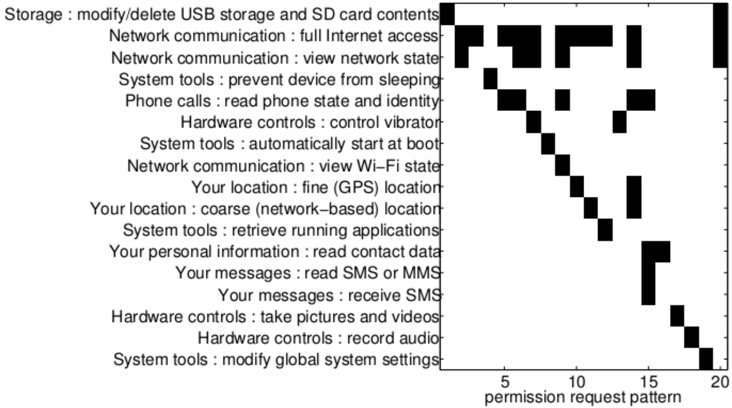
- Define boolean product $\mathbf{c} = \mathbf{a} \otimes \mathbf{b}$ of 2 matrices by:
- $c_{id} = \bigvee_{k=1}^{K} (a_{ik} \wedge b_{kd})$
- Want to find $\mathbf{z}, \mathbf{u}$ such that $\mathbf{x} \approx \mathbf{z} \otimes \mathbf{u}$
- If app i has pattern k and pattern k has permission d, then app i has permission d

Results

- They trained this model on high reputation apps (average rating of 4 or higher, at least 100 user ratings)
- $K = 30$ significant permission patterns
- Note that Permission Request Patterns are not disjoint: apps can request multiple patterns (subsets of its permissions.)
- A PRP with 1 permission indicates that a permission is requested a lot, but not always together with the same permissions

More results

Most common permission request patterns:



More results

- If an app has a permission request pattern that is not among these whitelisted patterns, then it is risky
- Can be used to predict likely reputation of new apps
- Good for detecting risky or buggy apps, but not a malware detector
- Did not analyze categories of apps in Google Play store

Network Visualization

- *Various Approaches in Analyzing Android Applications with its Permission-Based Security Models*
- Paper by Ittipon Rassameeroj and Yuzuru Tanahashi (U.C. Davis) [2]
- Visualizing related permissions per-category
- Create a network visualization based on permission data

Network Visualization

- Dataset 1: Adjacency matrix of permission concurrence
  - $M_{ij} =$ no. of apps where permission i and permission j are both requested
- Dataset 2: Adjacency matrix of distance between apps
  - Represent permissions of an app as a bit-vector
  - Distance between 2 apps is the Euclidean distance
  - Adjacency matrix of the resulting graph

Concurrent Permissions over All Apps

- Roughly divides permissions into large functional categories

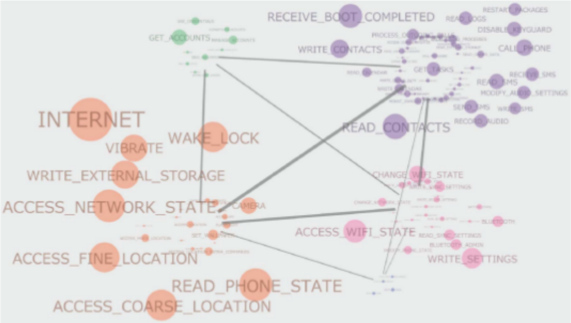


Fig. 1. Permission network of all APKs. Operationals that are granted in the permissions of each cluster represents a unique aspect of the device. The purple cluster contains many operations that a phone would perform. The orange cluster contains many operations that a web client and GPS would perform. The pink cluster contains many operations that ubiquitous devices such as a smartphone would perform.

APK Network

- Network of similar apps in Travel category

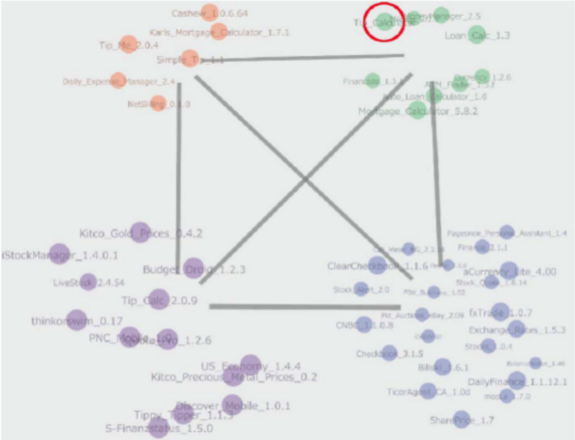


Fig. 2. APK network in the Travel category.

Results

- Suggests a method for manually finding suspicious apps
- For example, a tipping program that appears in the cluster for apps related to checking exchange rates
- Likely overprivileged or malicious
- Rank clusters by dangerous combinations of permissions

Risk Signals

- *Android Permissions: A Perspective Combining Risks and Benefits*
- Explore various techniques for giving a warning to the user
- Minimize warnings while maximizing detection of malware
- By category and sub-category

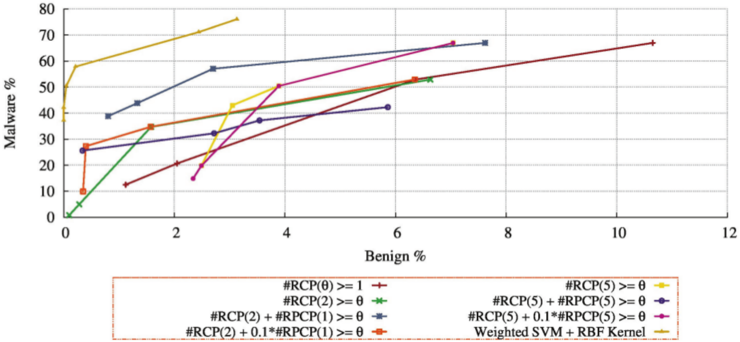
Risk Signals

- Choose 26 critical permissions, a subset of the "dangerous" permissions
- Category-based rare critical permission signal (CRCP)
- CRCP(θ) means an app uses a permission that is used by less than θ percent of the apps in the same category (theta can be an arbitrary threshold, not just percentage of apps)
- Allow user to select category for app other than its assigned category for purpose of checking if signal is raised
- Tell user what percent of apps in the category trigger signal for any permission

Rare Pairs of Critical Permissions

- A pair of permissions triggers RPCP(y) if:
- The individual permission's frequency is greater than y, but the frequency of the 2 permissions together is below y
- i.e. the permissions are relatively common, but they are not seen together frequently
- Trigger warning if $RPCP(y) \geq \theta$

Results


Malware %
Benign %
80
70
60
50
40
30
20
10
0
0
2
4
6
8
10
12
#RCP(θ) >= 1
#RCP(2) >= θ
#RCP(2) + #RPCP(1) >= θ
#RCP(2) + 0.1*#RPCP(1) >= θ
#RCP(5) >= θ
#RCP(5) + #RPCP(5) >= θ
#RCP(5) + 0.1*#RPCP(5) >= θ
Weighted SVM + RBF Kernel

- The SVM performed the best
- However, trained only on specific set of apps
- Linear combination of RPCP (pair-wise) and RCP (all apps) performed second-best
- CRCP (by category) performed better than RCP

- Frank, M. and Dong, B. et al. 2012
  Mining Permission Request Patterns from Android and Facebook Applications. arxiv.org, submitted 8 October 2012.
- Rassameeroj, I. and Tanahashi, Y. 2011
  Various Approaches in Analyzing Android Applications with its Permission-Based Security Models. IEEE International Conference on Electro/Information Technology.
- Sarma, B. and Li, N. et al. 2012
  Android Permissions: A Perspective Combining Risks and Benefits. SACMAT '12: Proceedings of the 17th ACM symposium on Access Control Models and Technologies
- Felt, A. and Chin, E. et al. 2011

Android Permissions Demystified. CCS '11: Proceedings of the 18th ACM conference on Computer and communications security

Felt, A. and Ha, E. et al.
Android Permissions: User Attention, Comprehension, and Behavior. Symposium on Usable Privacy and Security (SOUPS) 2012, July 11-13

Chia, P., Yamamoto, Y., and Asokan, N.
Is this App Safe? A Large Scale Study on Application Permissions and Risk Signals. International World Wide Web Conference April 16-20 2012.

Bartel, A. and Klein, J. et al. 2012
Automatically Securing Permission-Based Software by Reducing the Attack Surface: An Application to Android. ASE 2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering Pages 274-277

- Sarma, Baskar et al. 2012.
Android Permissions: A Perspective Combining Risks and Benefits. SACMAT '12: Proceedings of the 17th ACM symposium on Access Control Models and Technologies
- Yang, N. and Boushehrinejadmoradi, N et al. 2012
Enhancing Users Comprehension of Android Permissions. SPSM '12: Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices
- Mann, C. and Starostin, A. 2012
A Framework for Static Detection of Privacy Leaks in Android Applications. SAC '12: Proceedings of the 27th Annual ACM Symposium on Applied Computing
- Grace, M. and Zhao, Y. et al. 2012

RiskRanker: Scalable and Accurate Zero-day Android Malware Detection. MobiSys '12: Proceedings of the 10th international conference on Mobile systems, applications, and services

- Kelley, Patrick et al. 2012. *A conundrum of permissions: Installing applications on an Android smartphone*. UbiComp '12: Proceedings of the 2012 ACM Conference on Ubiquitous Computing

- Gold, Steve. 2012 *Android: A Secure Future at Last?*. Engineering & Technology Magazine, March 2012.

- Shabtai, Asaf., Fledel, Yuval et al. 2010 *Google Android: A Comprehensive Security Assessment*. Security & Privacy, IEEE. March-April 2010

- Pieterse, Heloise and Olivier, Martin. 2012.

Android Botnets on the Rise: Trends and Characteristics.
Information Security for South Africa (ISSA), 2012

Wikipedia Article. *Google Play*