Authorization Logic for Mobile Ecosystems

Second Year Report

Joseph Hallett

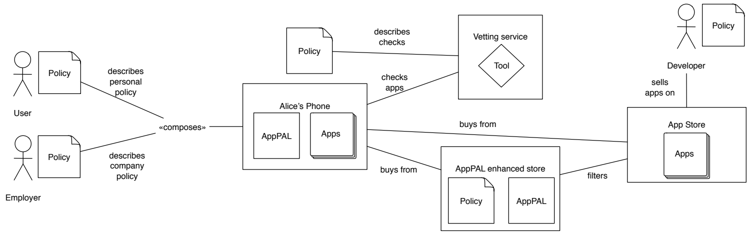
13th October 2015

Automatic tools and policy languages would provide a better means of enforcement for mobile device policies than existing mechanisms which rely on manual inspection.

AppPAL

- Authorization language for *app installation policies*
- Instantiation of *Becker et al.'s SecPAL* in Java
- Glue between user and corporate *device policies* and the *static analysis tools* and *trust relationships* used to implement them.
- Designed to model and enforce policies in *mobile ecosystems*

Mobile Ecosystems



AppPAL

$$\overbrace{\texttt{'user'}}^{\text{speaker}}\quad \texttt{says}\quad \overbrace{\overbrace{\texttt{App}}^{\text{subject}}\ \overbrace{\texttt{isRunnable}}^{\text{predicate}}}^{\text{fact}}$$

$$\overbrace{\texttt{if App isFree}}^{\text{condition}}$$

$$\overbrace{\texttt{where hasPermission(App, 'INTERNET') = True.}}^{\text{constraint}}$$

$$\texttt{'user'}\quad \texttt{says}\quad \underbrace{\underbrace{\texttt{'boss'}}_{\text{to}}\ \texttt{can-say}\ \underbrace{\texttt{inf}}_{\text{depth}}\ \texttt{App isInstallable.}}_{\text{delegation}}$$

Example AppPAL Policy

```
'user' says App isInstallable
  if App hasCategory('game')
     App isGood
  where hasPermission(App, 'IAP') = False.

'user' says 'play-store' can-say App hasCategory(Category)

'user' says 'review-site' can-say inf App isGood.

'review-site' says App isGood
  where reviewSiteScore(App) > 7.
```

Summary of Second Year Work

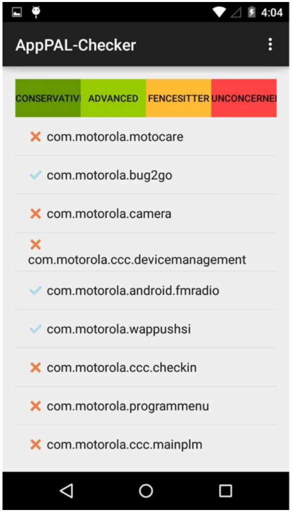
- Created a knowledge base about android apps
- Implemented AppPAL on Android
- Explored the usage policies in current app stores
- Looked at the distribution mechanisims in current app stores
- Looked at the extent privacy preferences are being followed by users using current mechanisms

Security Knowledge Base

- Needed a knowledge base to store and collect metadata about apps
- Existing tooling overly complex and couldn't be extended (easily)
- Collects metadata for around 40,000 apps
- Can run static analysis tools on the apps and collect results
- Can output AppPAL statements
- Would be nice to keep extending and to query it from AppPAL

AppPAL on Android

- Prototype from first year couldn't run on Android
- Reimplemented as a library for Java
- Created apps to scan installed apps against policies, create app stores with policies.



✗ com.motorola.motocare
✓ com.motorola.bug2go
✗ com.motorola.camera
✗ com.motorola.ccc.devicemanagement
✓ com.motorola.android.fmradio
✓ com.motorola.wappushsi
✗ com.motorola.ccc.checkin
✗ com.motorola.programmenu
✗ com.motorola.ccc.mainplm

Policies in Current Stores

- Looked at the developer and user terms of use for 4 different app markets
  - Google Play, Amazon, Yandex and Aptoide
- The policies are quite similar.
- Largest differences are to do with payment processing and age of use.
- Some stores keep modification rights
  - Moves trust from developer to store

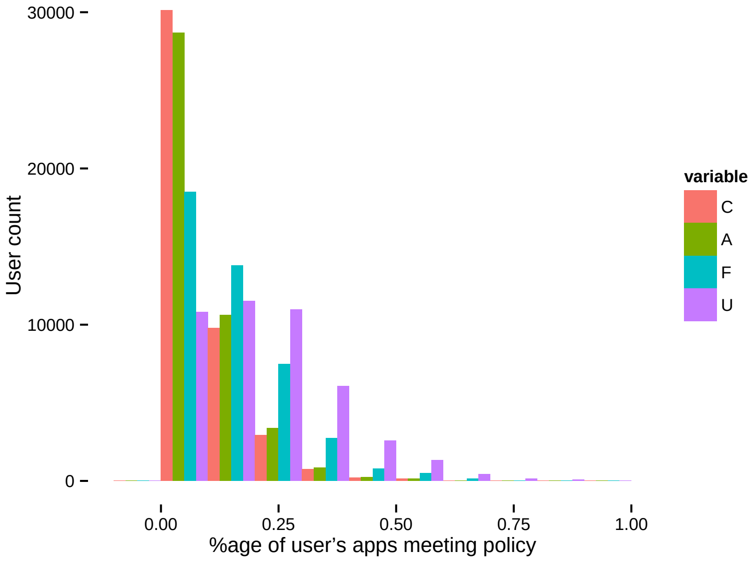
Distribution Mechanisms

- Used an SSL proxy to look at how the stores download apps
- Lots of implementation differences
- Some implementation problems
  - Certificate pinning
  - Encryption being dropped (or missing) for download
  - Being able to re-download apps

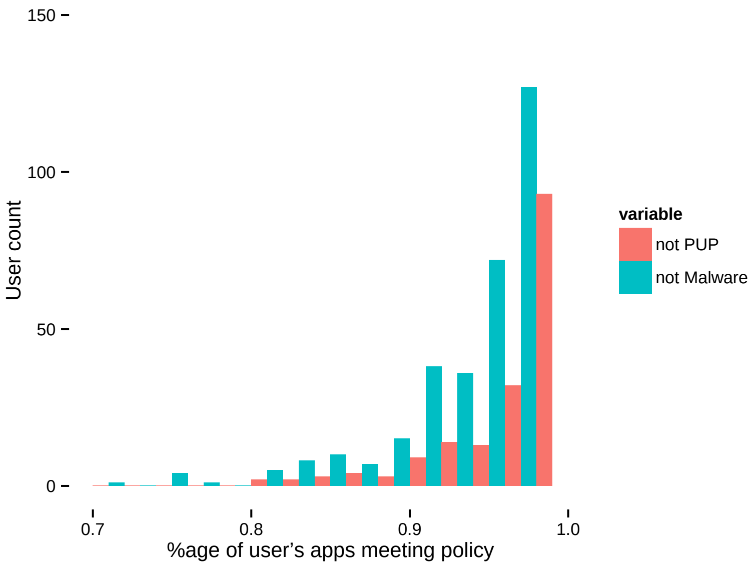
1.	$C \longrightarrow S$:	U, C, a_{d}
2.	$S \longrightarrow C$:	$a_{\mathrm{d}}, ?$
3.	$C \longrightarrow S$:	$U, !$
4.	$S \longrightarrow C$:	$a_{\mathrm{d}}, \$$
5.	$C \longrightarrow S$:	$U, a_{\mathrm{d}}, \$$
6.	$S \longrightarrow C$:	S'
7.	$C \longrightarrow S'$:	
8.	$S' \longrightarrow C$:	a

Policies in Practice

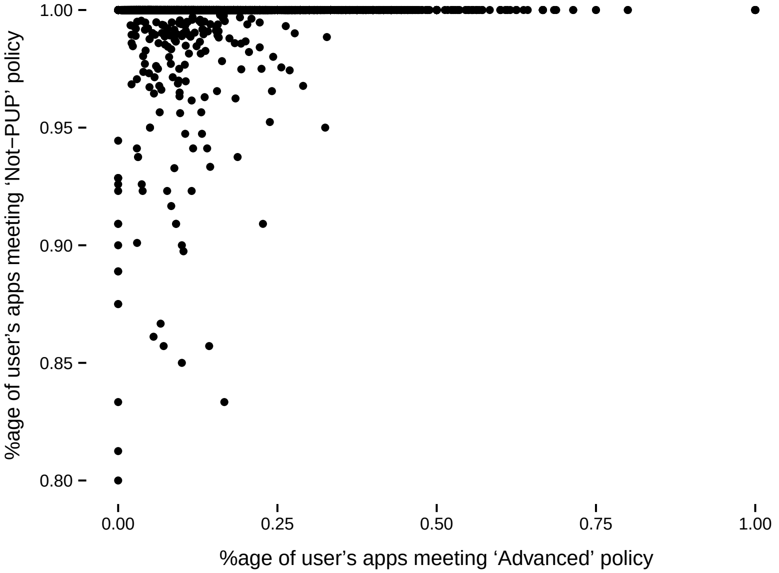
- Using *Carat* installation data we found 44,000 users for whom we new at least 20 apps they had installed
- Described 4 user privacy preference policies (*Lin et al.*) using AppPAL
- Took a list of Android malware from McAfee and created *no-malware* policies
- Ran *MalloDroid* on the apps and created *no-SSL-error* policies
- Measured the extent each user was following each policy

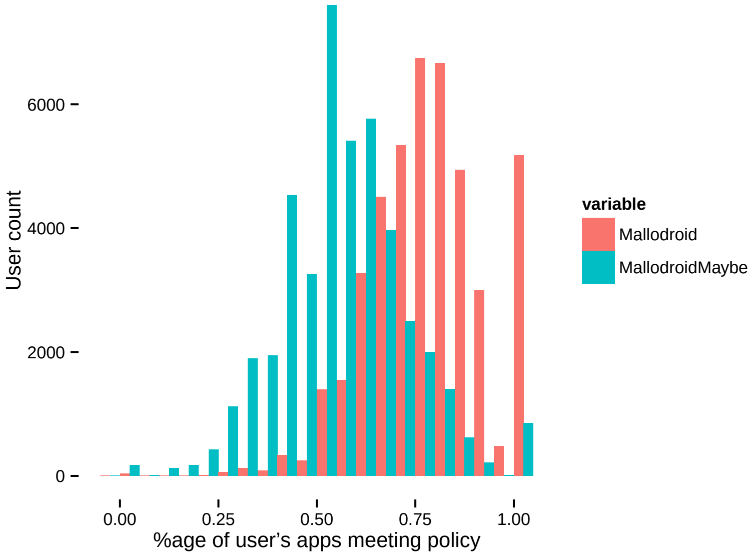


30000
20000
10000
0
User count
0.00
0.25
0.50
0.75
1.00
%age of user's apps meeting policy
variable
C
A
F
U

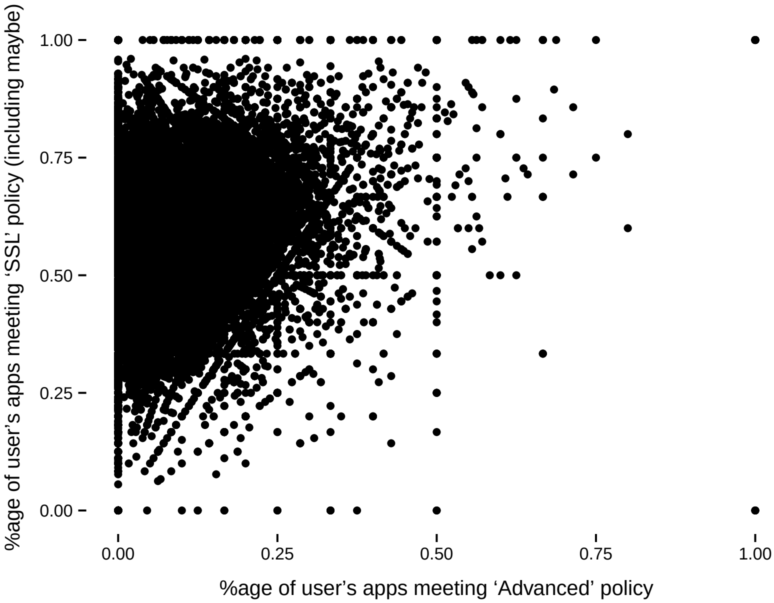


150
100
50
0
User count
0.7
0.8
0.9
1.0
%age of user's apps meeting policy
variable
not PUP
not Malware


1.00
0.95
0.90
0.85
0.80
0.00
0.25
0.50
0.75
1.00
%age of user's apps meeting 'Not−PUP' policy
%age of user's apps meeting 'Advanced' policy



User count
6000
4000
2000
0
0.00
0.25
0.50
0.75
1.00
%age of user's apps meeting policy
variable
Mallodroid
MallodroidMaybe



%age of user's apps meeting 'SSL' policy (including maybe)
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
%age of user's apps meeting 'Advanced' policy

Proposed Third Year Work

- Knowledge distribution protocols
- Case study with AppPAL

Knowledge Distribution Protocols

- We can express delegation relationships with SecPAL based languages
- It isn't clear how we should ask for more information
- Don't want delegatee to make the decision, necessarily
- Links to multi-agent knowledge distribution? (FIPA/KQML?)

- A contribution to make AppPAL more distinct from SecPAL
- Links up with security knowledge base as a source of AppPAL statements
- How do we handle timely queries?
- How do we distinguish:
  - not knowing
  - not wanting to answer (because it's false)
  - not wanting to answer (because they're unsure)
- Could lead to questions whether we can quantify the trust we have in AppPAL statements?

Case Study

- BYOD policies increasingly described informally by businesses
- Would like to implement one in AppPAL
  - alternately NIST-SP-800–46/124
- Shows the extent of what we can express in AppPAL
- Bigger use case than the hypothetical, and supposed, policies we've used so far
- May lead to other questions:
  - Can policies be composed?
  - What happens when multiple corporate policies disagree?
  - What happens when policies change over time?

Questions?