Language-Based Security on Android
(call for participation)

Avik Chaudhuri
What is Android?

- Open-source platform for mobile devices
- Designed to be a complete software stack
  - Operating system
  - Middleware
  - Core applications
- Accompanied with an SDK
  - Tools and APIs to develop new applications
  - Based on the Java programming language
  - Compiled to run on a VM designed to optimize memory and hardware resources in a mobile environment
Why should I care?

- **Open Handset Alliance**
  - major mobile operators (T-Mobile, Vodafone, …)
  - semiconductor companies (Intel, NVIDIA, …)
  - handset manufacturers (LG, Motorola, Samsung, Sony Ericsson, …)
  - software companies (**Google**, …)

- **Free** and **open-source** software

- **Core applications** and **new applications** on par
  - All applications can access the underlying mobile device
  - All applications have similar structure, and can share functionality
How can I join in?

- Maturing into a major platform in the mobile phone industry
  - Some Android-based phones available
    - More planned for release
  - Many Android applications available
    - Many more developed and installed every day

- Actually a source of concern!

What do we understand about security on Android?

- Costs of ignorance? (Same as in other settings)

- Benefits of knowledge? (Perhaps more than in other settings)
This reminds me of PCC!

- Alice downloads on her phone a new app developed by Bob
- How does she know whether it is safe to run the app?
  - Can she trust the app to safely access her data?
  - If she cannot, is there still a way to safely run the app?

These concerns are important for Alice (user).

These concerns are also important for Bob (developer).

- How does Bob convince Alice that his app is safe to run?
How about certified installation?

- Bob constructs a safety proof for his app
  - Sound static analysis of the Java code

- Alice verifies the proof before installing the application

Ingredients?

- **Operational semantics** for applications on Android
  - Formal specifications of the APIs provided by the SDK

- **Static checking** for safety of Android applications
  - Formalized as a security type system for application code
  - Soundness of type system provides proofs
What are Android applications?

- Package of components
  - Can be run as necessary (possibly even by other apps)

- Components follow structure

  **Activity** controls some **window** of the app

  **Service** runs in **background** (even if window is switched), exposes interface for communication with other apps

  **Receiver** reacts asynchronously to **messages** from other apps

  **Provider** stores **content** relevant to the app (usually in database), allows sharing of data with other apps
Show me an example.

Music player app

Activities
- View all songs
- Edit details of a song

Services
- Play song in background

Receivers
- Pause song on incoming call
- Resume song when call ends

Providers
- Share songs
How do I develop this in the SDK?

Inherit built-in classes, override their methods!

Activity { ..., onCreate(x), onActivityResult(x), ... }
Service { ..., onBind(x), ... }
Receiver { ..., send(x), onReceive(x), ... }
Provider { ..., query(x), update(x), ... }

SongsViewActivity
SongEditActivity
PlayService
IncomingCallReceiver
CallEndReceiver
SongsProvider
What about security?

- Apps require **permissions** to access components of other apps
  - To **call** activities
  - To **bind** to services
  - To **send** messages to receivers, and to **receive** messages from apps
  - To **query** and to **update** content stored by providers

**Permissions are set at install time, not at run time**

- Required permissions are **declared** in a manifest

- Permissions are set **statically** by the package installer

- The app **blocks** if it requires other permissions at run time
I don’t know how my app is run!

Semantics

Components

Stack of windows

method definitions + access controls

(content)

Pool of listeners

[perms] current_expr

\(\text{x. [perms']}\) callback_expr

(\(\text{x. [perms'']}\) listener_expr)
A formal abstract syntax

Program syntax

\[ v ::= \]
\[ n \]
\[ x \]
\[ \text{void} \]
\[ i ::= (n, v) \]
\[ t ::= \]
\[ \text{call}(i) \]
\[ \text{return}(v) \]
\[ \text{bind}(i, \lambda x.t) \]
\[ \text{register}(\text{SEND}, \lambda x.t) \]
\[ \text{send}(\text{RECEIVE}, i) \]
\[ !n \]
\[ n ::= v \]
\[ \text{let } x = t \text{ in } t' \]
\[ \text{fork} \]
\[ t + t' \]
\[ v \]

value
name
variable
void
intent
code
call activity
return from activity
bind to service
register new receiver
send to receiver
read from provider
write to provider
evaluate
fork
choice
result

Application syntax

\[ d ::= \]
\[ \text{activity}(\text{CALL}, \text{PERMS}, \lambda x.t, \lambda x.t') \]
\[ \text{service}(\text{BIND}, \text{PERMS}, \lambda x.t) \]
\[ \text{receiver}(\text{SEND}, \text{PERMS}, \lambda x.t) \]
\[ \text{provider}(\text{READ}, \text{WRITE}, v) \]
\[ D ::= \emptyset \mid D, n \mapsto d \]

definition
activity
service
receiver
provider
hash of definitions

Encode away services

Enencodings

\[ \text{service}(\text{BIND}, \text{PERMS}, \lambda x.t) \triangleq \text{receiver}(\text{BIND}, \text{PERMS}, \lambda x.t) \]
\[ \text{bind}((n, v), \lambda x.t) \triangleq \text{let } x = \text{send}(\perp, (n, v)) \text{ in } t \]
A formal operational semantics

Local reduction \( D; e; E \rightarrow D'; e'; E' \)

(\text{Red let-distr}) \( D; [\text{PERMS}] \text{let } x = t \text{ in } t'; E \rightarrow D; \text{let } x = \text{PERMS} \text{ in } t' \text{ in } \text{PERMS} \text{ in } t'; E \)

(\text{Red let-eval}) \( D; \text{let } x = e \text{ in } \text{PERMS} \text{ in } t; E \rightarrow D'; \text{let } x = e' \text{ in } \text{PERMS} \text{ in } t; E' \)

(\text{Red let-return}) \( D; \text{let } x = [\text{PERMS}'] \text{ in } \text{PERMS} \text{ in } t; E \rightarrow D; [\text{PERMS} \text{ in } t \{v/x\}; E \)

\( E' = E; [\text{PERMS} \text{ in } t \{v/x\}; E \)

(\text{Red fork}) \( D; \text{PERMS} \text{ in } t \text{ in } E \rightarrow D; \text{PERMS} \text{ in } \text{void}; E' \)

(\text{Red read}) \( D(n) = \text{provider(READ, v)\ PERMS \sqsubseteq \text{READ} \rightarrow D; \text{PERMS} \text{ in } n; E \rightarrow D; \text{PERMS} \text{ in } v; E \)

(\text{Red write}) \( D(n) = \text{provider(READ, WRITE, \_)} \ PERMS \sqsubseteq \text{WRITE} \rightarrow D = D[n \mapsto \text{provider(READ, WRITE, v)}] \)

(\text{Red register}) \( n \text{ fresh} \rightarrow D' = D, n \mapsto \text{receiver(SEND, PERMS, \_x.t)} \)

(\text{Red send}) \( D(n) = \text{receiver(SEND, PERMS', \_x.t)} \ PERMS \sqsubseteq \text{SEND} \rightarrow D = D[n \mapsto \text{receiver(SEND, \_x.t)}] \)

(\text{Red choice-I}) \( D; [\text{PERMS} \text{ in } t + t'; E \rightarrow D; [\text{PERMS} \text{ in } t; E \]

(\text{Red choice-r}) \( D; [\text{PERMS} \text{ in } t + t'; E \rightarrow D; [\text{PERMS} \text{ in } t'; E \)

Global reduction \( D; S; E \rightarrow D'; S; E' \)

(\text{Red local}) \( D; e; E \rightarrow D'; e'; E' \)

(\text{Red call}) \( D(n) = \text{activity(CALL, PERMS', \_x.t, \_x.t')} \ PERMS \sqsubseteq \text{CALL} \rightarrow S' = ([\text{PERMS}] \text{ void, \_x.e}) \rightarrow S \)

(\text{Red return}) \( D; ([\text{PERMS}] \text{ void, \_x.e}) \rightarrow S \rightarrow D; e' \rightarrow D'; [\text{PERMS} \text{ in } \text{void}; \_x.e']) \rightarrow S'; E \)

(\text{Red thread}) \( D; e; E \rightarrow D'; e'; E' \)

data flows + control flows + permission checks
A security type system

Stack types

- Call activities from the stack
- Send to receivers from the stack or the pool
- Data that READ can read and WRITE can write

Blocking due to access control
What guarantees do I get?

- Sub-typing captures **data flow**
- **Type preservation** implies **data-flow security**

Well-typed programs preserve their types [PLAS 09]

- Focus on data flow rather than information flow
- Guarantees should extend under refinement

**Future work**

**Type-check concrete application code**

- Build on existing static analysis tools for Java (e.g., WALA)
How does this help?

- Alice can safely run any well-typed app on her phone
  - Guaranteed to preserve the secrecy and integrity of her data
- Any app type-checks if it has no permission
  - Alice can safely run such an app on her phone (w/o permissions)
- Bob can convince Alice that his app is safe if it type-checks

**Certified installer for Android applications**

Install apps that type-check, or are certified to type-check

**Opportunity to bring language-based security to the mainstream!**
Questions?