Automated Cellphone Application Certification in Android (or) Mitigating Phone Software Misuse Before It Happens

Systems and Internet Infrastructure Security (SIIS) Laboratory
Network and Security Research Center
Department of Computer Science and Engineering
The Pennsylvania State University

William Enck, Machigar Ongtang, and Patrick McDaniel

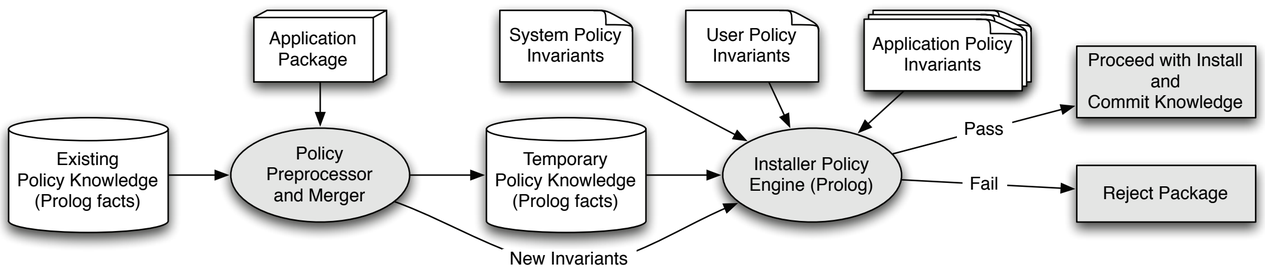
Third Party Phone Applications

- Recent smartphones platforms (e.g., Google Android, Apple iPhone, OpenMoko, etc) indicate a new trend for historically tightly controlled cellular handsets.
- New concerns are raised by the transition from close to open phones.
- A single poorly vetted program can:
  * compromise user data
  * disrupt fragile cellular networks
  * render a cell phone inoperable

Current Solution

- *Trusted Third Party Verification* (e.g., Apple AppStore, Symbian certification, etc). Emerging "AppStores" are expected to use similar tactics.
- This technique is not ideal:
  * Manual inspection is hard
  * Mixed incentives (commercial interestes frequently differ from genuine security and quality concerns)
  * No single definition of acceptible risk
  * Analysis is performed without knowledge of present applications, data, and services

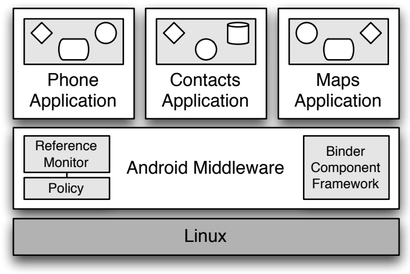
Self-Certification at Installation

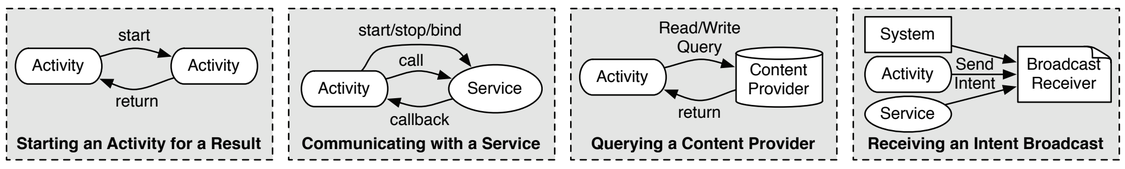


- *Idea*: Enhance the package installer with a policy engine
  * The Google Android smartphone operating system defines a framework where applications request permissions at install time
  * Applications also define access control polices for internal components
- *Challenges*:
  * Formalize Android's policy framework
  * Define appropriate "policy invariants"
  * Develop an install time policy tool

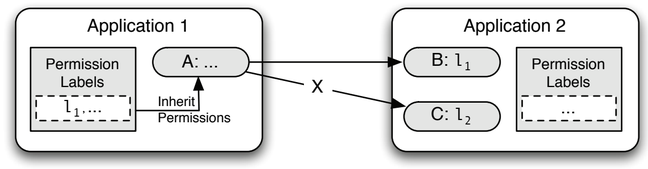
The Google Android OS

- The Android OS provides a middleware application abstraction based on components
- A reference monitor mediates the establishment of Inter-Component Communication (ICC)





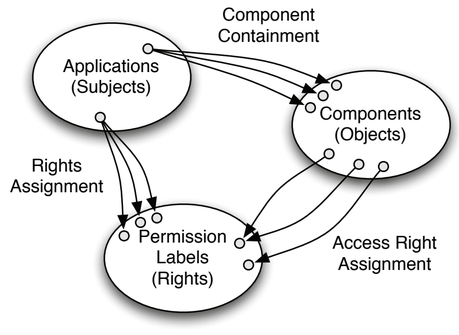
- Android uses a relatively straightforward security model where application assigned "permission labels" are inherited by components for use when performing ICC



- However, there are many practical exceptions ...

A Formal Logic

- We express Android's security policy as a traditional Subject-Objects-Rights (SOR) Access Matrix
- The policy allows a component in application *s* to access a component *o* that requires right *r* if the following evaluates to true:



$$P(s,o,r) = \texttt{requires}(o,r) \wedge \texttt{has\_perm}(s,r)$$

- We model additional policy semantics (caveats) through preprocessing that increases the number of objects/rights
- **Default Allow Policy**: Components with unspecified access rights can be accessed by anyone. We model this by creating a reserved "open" right assigned to all applications and components with unspecified access conditions.
- **Protected APIs**: Framework interfaces requiring special rights are modeled as objects.
- **Content Providers**: The read and write semantics of Content Providers are modeled by creating two objects
- **Broadcast Permissions**: Intent broadcasts allow "reverse permission checks," therefore Intents are modeled as objects.
- **Permission Protection Levels**: Not all requested permissions are granted, therefore preprocessing creates "virtual rights."

Example Policy Invariants

"An application must request permission to make an outgoing call"

$$\begin{aligned}\texttt{invariant}_{1i}(s) \;=\; & \forall r_1 \in R, \exists r_2 \in R.\, \neg P(s, \texttt{IF}_i, r_1) \\ & \vee (P(s, \texttt{IF}_i, r_2) \wedge (\texttt{has\_perm}(s, \texttt{CALL\_PHONE}) \vee \texttt{has\_perm}(s, \texttt{CALL\_PRIVILEGED}))\end{aligned}$$

"Applications that can perform audio record must not have network access or pass data to an application that has network access"

$$\begin{aligned}\texttt{invariant}_5(s) \;=\; & \forall r_1 \in R, \exists r_2 \in R, \forall r_3 \in R, \forall s_1 \in S, \forall o \in O, \forall \{r_4, r_5\} \in R.\, \neg P(s, \texttt{record\_audio}, r_1) \\ & \vee (P(s, \texttt{record\_audio}, r_2) \wedge \neg P(s, \texttt{network}, r_3) \\ & \wedge \neg (P(s, o, r_4) \wedge \texttt{contains}(s_1, o) \wedge P(s_1, \texttt{network}, r_5)))\end{aligned}$$

"An application can only receive SMS from trusted components"

$$\texttt{invariant}_{7i}(s) \;=\; \forall r_1 \in R, \exists r_2 \in R.\, \neg P(s, \texttt{BR}_i, r_1) \vee (P(s, \texttt{BR}_i, r_2) \wedge (s \equiv \texttt{PhoneApp}))$$

Kirin: An Enhanced Installer

- We developed Kirin, an enhanced installer that tests invariants.
- An evaluation of applications bundled with the SDK identifed multiple flaws that lead to vulnerabilities in the smartphone.
  * Unprivileged phone calls
  * Forged SMS messages within the system
  * Forged location updates to applications not following "best practices"
- Install time "self-certification" of applications provides flexibility and expressibility not achievable with "AppStore"-like application distribution models. This technique will be valuable for protecting next generation technology from malicous applications, before they are installed.

RESEARCH SUPPORTED BY NSF