PINPOINT: Efficient & Effective Resource Isolation for Mobile Security & Privacy

Paul Ratazzi, Ashok Bommisetti, Nian Ji, and Prof. Wenliang (Kevin) Du

Department of Electrical Engineering & Computer Science
Syracuse University, Syracuse, New York

Motivating Examples

- User likes 3rd party keyboard, but wants to ensure it will not leak sensitive information from certain apps
    - Currently, there is no way to list only trusted input methods for certain sensitive apps
- User wants some apps to use accurate sensor data, others to have less accurate data, and the rest to have no access
    - Currently, sensor access does not require permission, and all apps have same access
- User wants location-enabled coupon app to know regional location to get relevant coupons, but not coarse (10s of meters) or fine (~1 meter) locations
    - Currently, only options are no location, coarse location, or fine location
- User wants some location-enabled apps to access location data, and others to have no access
    - On/off setting is currently platform-wide
- User wants to play game, but does not want it to leak sensitive info. via requested `READ_PHONE_STATE` permission
    - Although need for permission may be legitimate, there is currently no way to allow legitimate use while making leakage impossible

Existing Isolation Approaches

- Cells
  - Leverages Linux Namespaces to allow multiple Android Virtual Phones (VP) on a single kernel
  - Hardware and kernel are shared among independent VPs

- AirBag
  - Leverages Linux Namespaces to allow multiple decoupled app runtimes
  - Hardware, kernel, and native userspace are shared among independent runtimes
  - Condroid improved by restoring binder communications and increasing efficiency

Advantage: powerful general-purpose solution with many applications

Existing Isolation Approaches

- Kernel-level isolation breaks many assumptions of Android's open platform design
- Significant effort is required to fix things → 2^{nd} order complexity
- Overhead and inconvenience to end-users

Disadvantage: cost and inconvenience may be too high for many simple security and privacy scenarios

Some Key Namespace Traits

Namespace Trait	Value to Android Security
Fine-grained isolation of specific resources	Tailored isolation environment for each application; few side effects
High efficiency	Negligible performance impact; design simplicity
Share-by-default	Preserve open system design; avoid breaking things unrelated to the isolated resource
Small footprint (files, memory)	Little impact on performance & resources; OTA updates

Our Idea: PINPOINT

→Employ a Linux Namespace-like approach to Android Framework resources

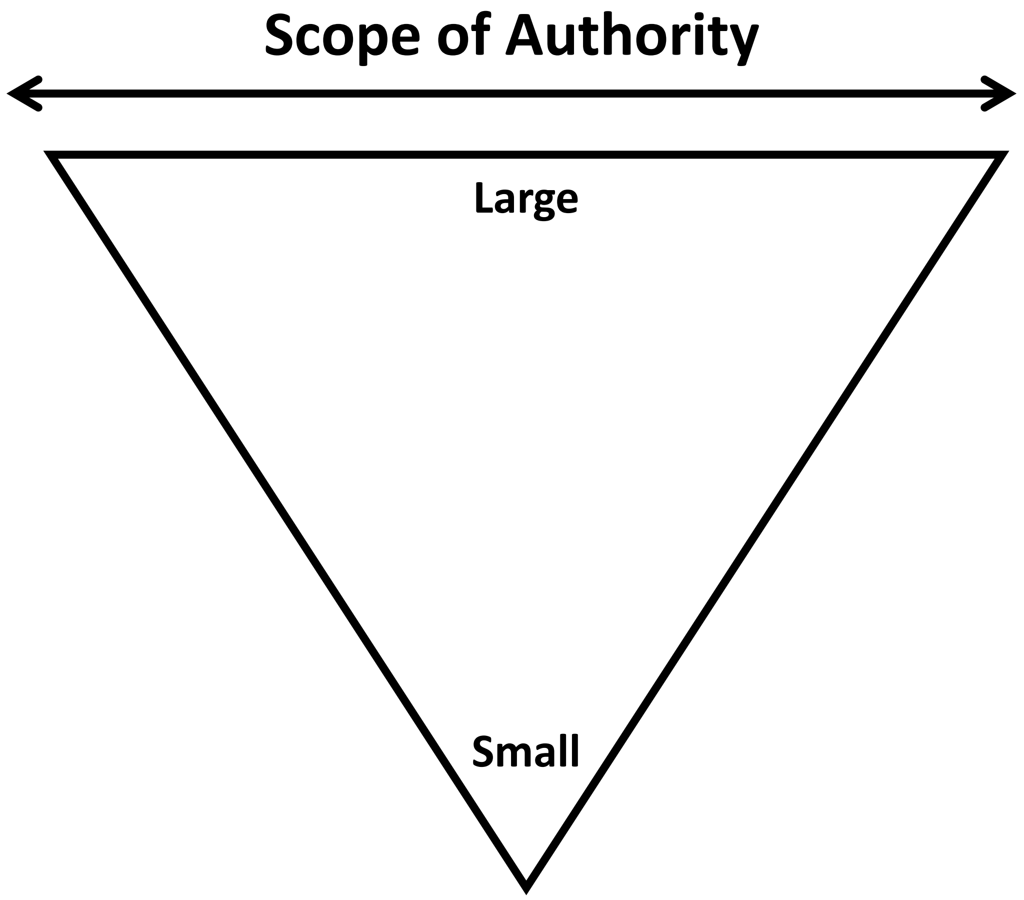
- Virtualize and isolate only what's necessary to meet stated security goal(s)
  - →Everything else is shared as Android intended
  - →Minimize or eliminate side-effects
- Provide isolation "building blocks" that can be used to create containers

About “-visors”

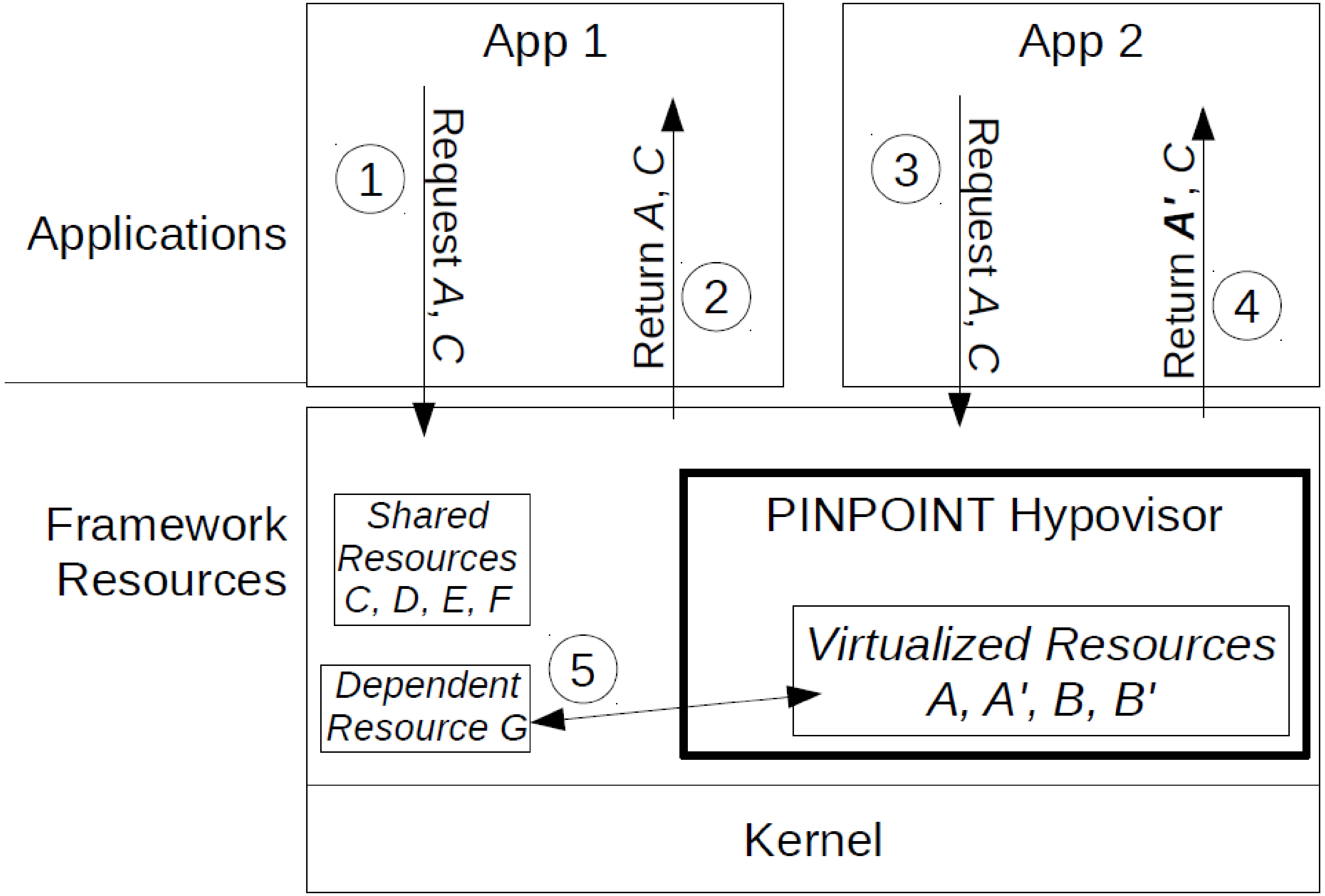
- Hypervisor (type I native)
  - Runs on “bare metal”
  - Authority over guest OS(s)

- Supervisor (a/k/a kernel)
  - Inside OS
  - Authority over userspace(s)

- NEW: Hypovisor
  - Inside userspace
  - Authority over resource(s)



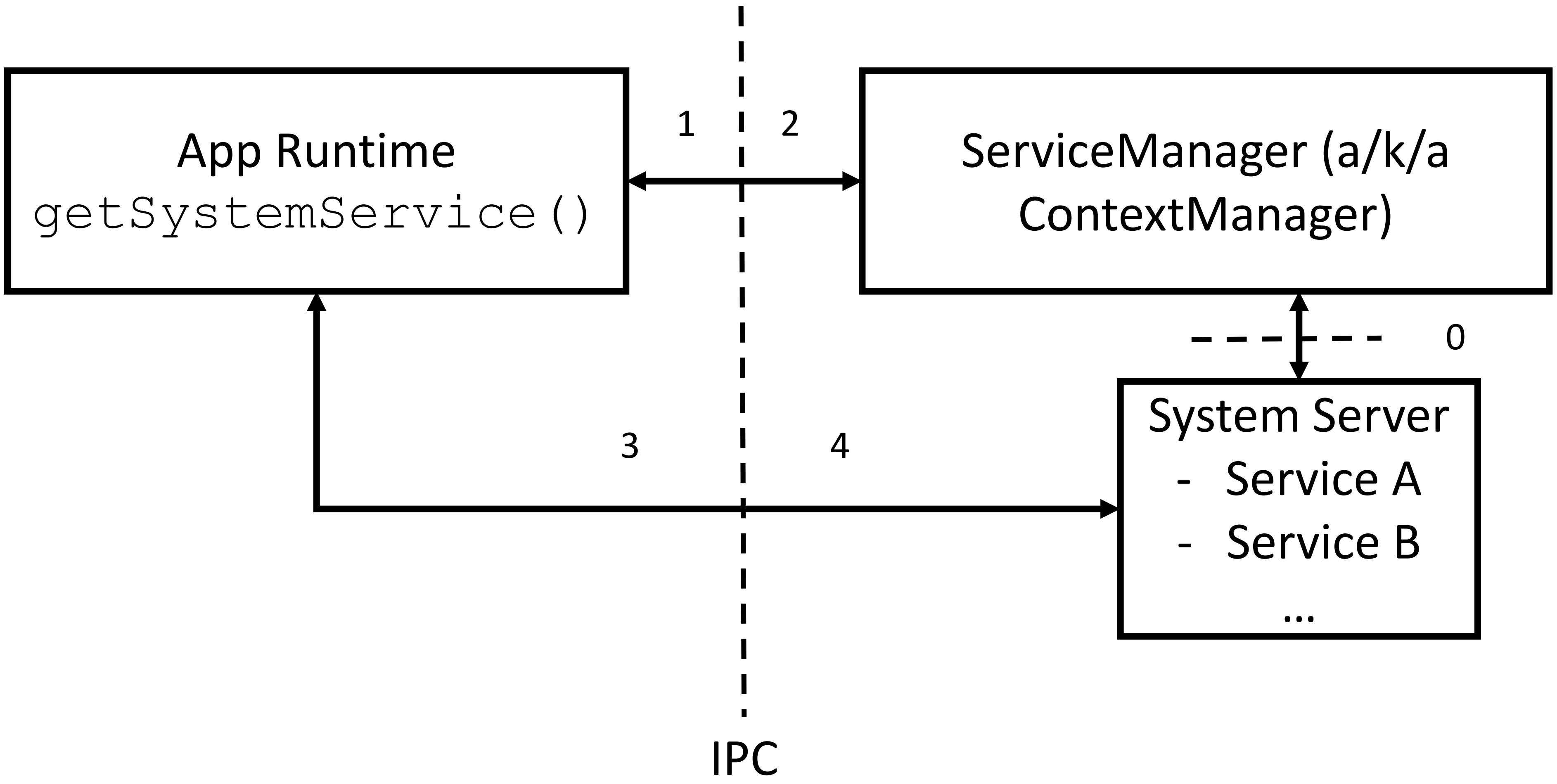
PINPOINT Concept



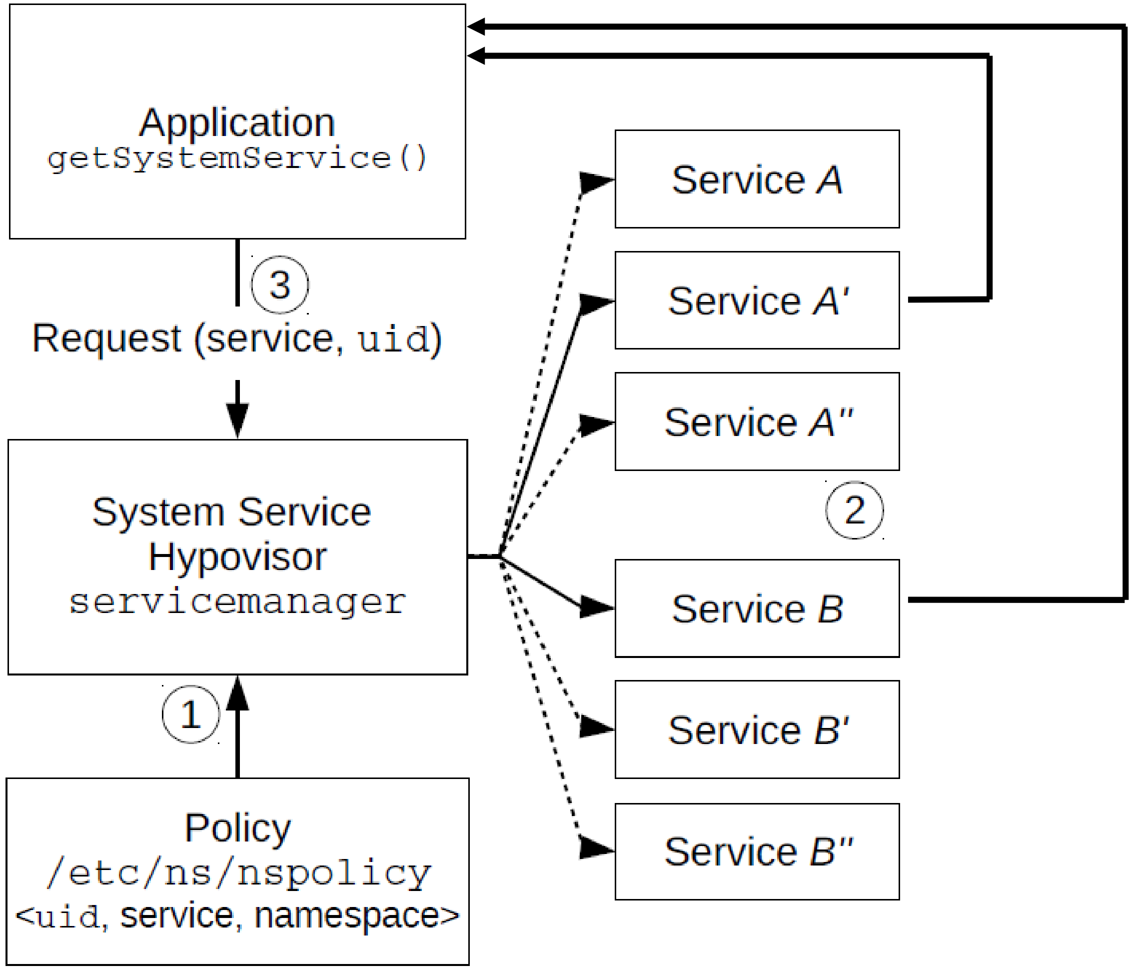
PINPOINT Methodology

Step	Description	Example
1	Define/collect security goal(s)	Protect IMEI from app A
2	Identify relevant resource(s)	`iphonesubinfo` and `phone` system services (5.1)
3	Identify point(s) of resource access / capability dispatch -> implement hypovisor(s) here	`servicemanager`
3a	Security analysis	Prevent inter-app passing of system service binder tokens (modified SEAndroid hook)
4	Identify and address dependency(ies)	`com.android.phone` and `ProxyController` (service startup)

Android System Service Basics



Case Study: System Services



Handle returned

System Service Hypovisor: `servicemanager`

`uint32_t do_find_service(struct binder_state *bs,` **`const uint16_t *s`**`, size_t len,` **`uid_t uid`**`, pid_t spid)`

1. Check `nspolicy` for entry matching caller's `uid` and service requested
2. On match, modify incoming request per `nspolicy`
3. Pass modified request to `find_svc()` for handle lookup

Example: `iphonesubinfo` → `iphonesubinfo_1` for `uid` 0010068

Hypovisor Security Analysis

- Fundamental question: “can the hypovisor be: 1) tricked or 2) bypassed?”

1) Our modifications <u>do not change</u> *how* service capabilities are dispatched, so any problems here are also a problem with stock Android

   - Subject identified by `uid` from binder driver (trusted)
   - Policy file restricted
   - Service name values validated

   → `servicemanager` cannot be tricked

Hypovisor Security Analysis

- Fundamental question: “can the hypovisor be: 1) tricked or 2) bypassed?”

2) For most normal services, `servicemanager` acts as an open capability dispatch service
    - Once obtained, apps are free to pass capabilities held to other apps
    - App-to-app transfer of system service capabilities bypasses the hypovisor

→Blocked via modified `security_binder_transfer_binder()` SEAndroid hook to disallow transfer of `u:r:system_server:s0` binders among `u:r:untrusted_app:s0`

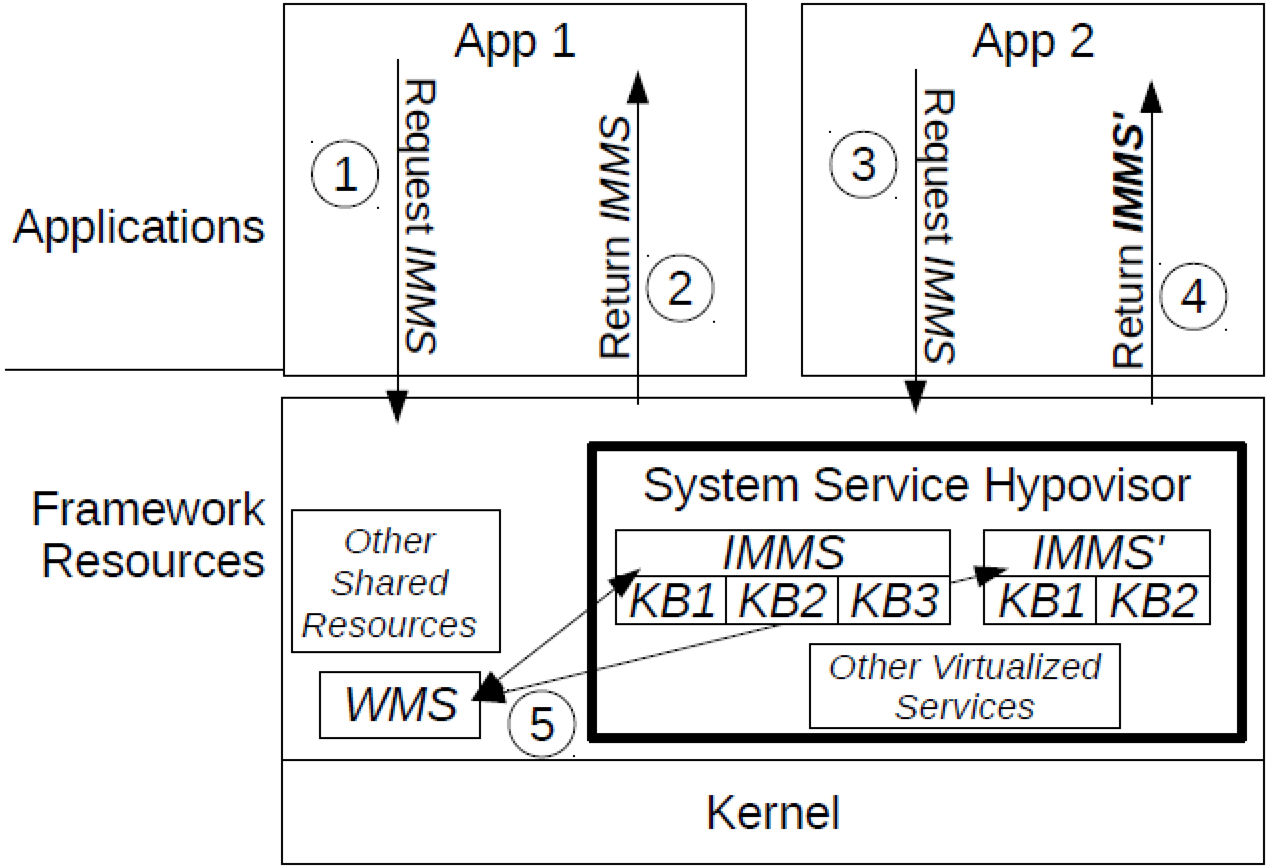
→`task_struct` of `binder_ref`/`binder_node` contains owner’s SELinux security identifier (SID)

Four Sample Applications

- Security goal: prevent untrusted apps from obtaining accurate location information
  - `LocationManagerService`
- Security goal: prevent critical apps from leaking information through untrusted input methods
  - **`InputMethodManagerService`**
- Security goal: prevent untrusted apps from obtaining sensitive subscriber information
  - `IPhoneSubInfo`
- Security goal: prevent untrusted apps from obtaining accurate sensor data to steal data, eavesdrop, or track movement/location
  - `SensorService`

InputMethodManagerService

- Goal: protect app from untrusted input method
- One additional namespace
  - Stock input methods only
- Dependency: WindowManagerService
  - Modified to update all input_method services with current activity



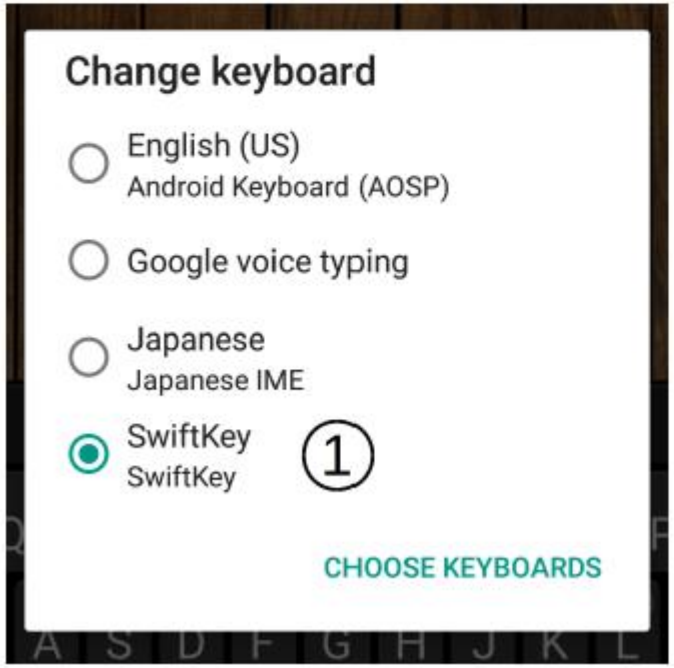
InputMethodManagerService

Unmodified non-critical app (uid 10083) with 3rd party IME available

Global IME

nspolicy: <no entry>

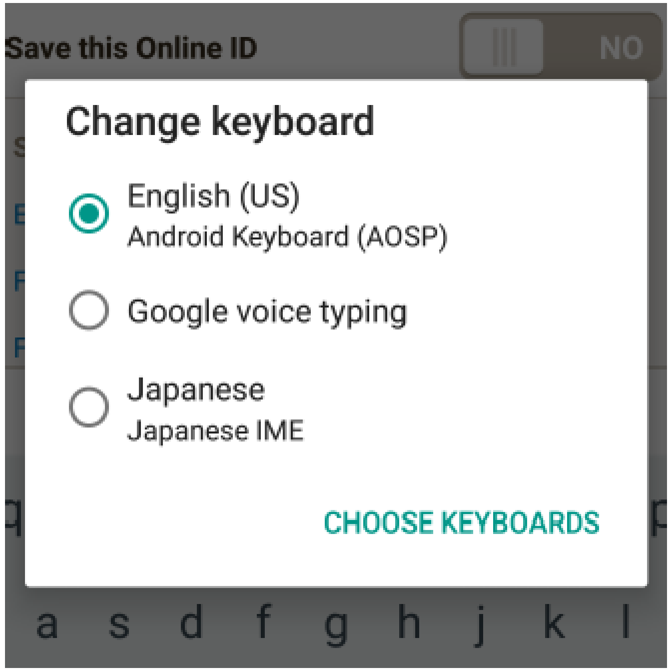
Requests: input_service; receives input_service



Alternate IME

nspolicy: 10084 input_service 1

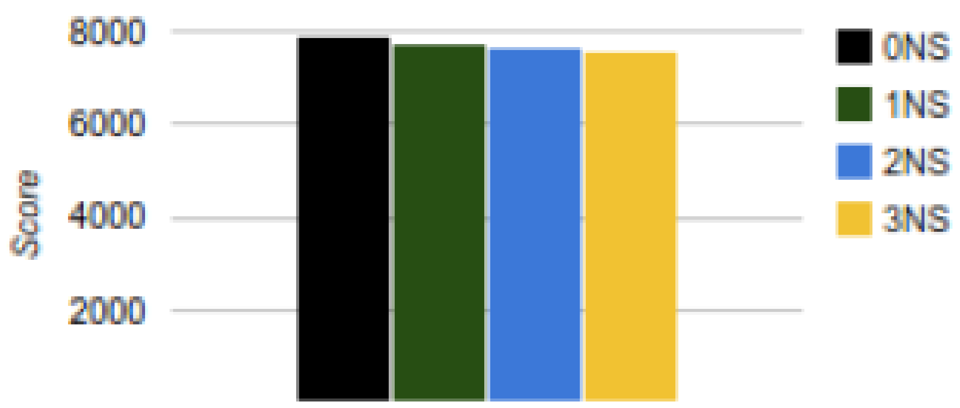
Requests: input_service; receives input_service_1



Unmodified banking app (uid 10084) with only stock IMEs available

Performance Impacts

Quadrant 2.1.1 File I/O score vs. # namespaces



~1.6% loss per namespace

`system_server` process memory footprint vs. # namespaces

~0.6% increase per namespace

Limitations

- Our approach does not provide security domain isolation
  - Apps can pass high level information among namespaces
- Alternate services must be configured and running even if not used
  - Additional `system_server` memory footprint
- Alternate services must be defined at build time

Future Directions

- Formalize methodology (esp. security analysis)
- Implement other hypovisors

- Provide sample device images
- Open source

PINPOINT

Thank You

Questions?

ratazzi@ieee.org