Giving developers the power to declare **per-component** access control **policy** for their apps

SEApp: Bringing Mandatory Access Control to Android Apps

Matthew Rossi, Dario Facchinetti, Enrico Bacis, Marco Rosa, Stefano Paraboschi

PROBLEM STATEMENT

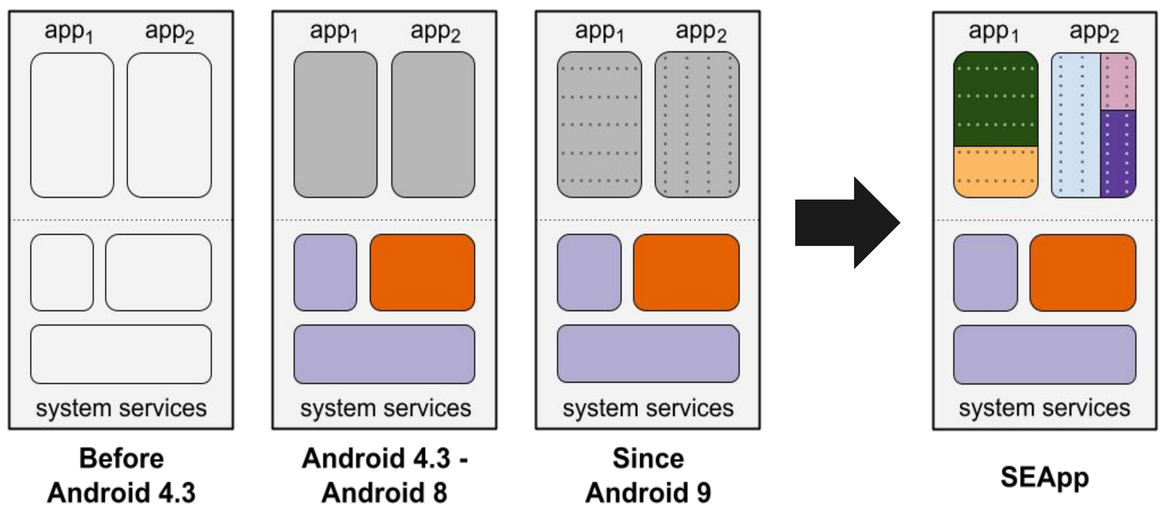
Android focuses on isolating applications from each other

There are no means to isolate components internal to the app:

- every component has **complete access** to the **internal storage**
- 3rd-party libraries may **abuse** app **privileges**
- large and complex **components** prone to bugs are **not easy to isolate**

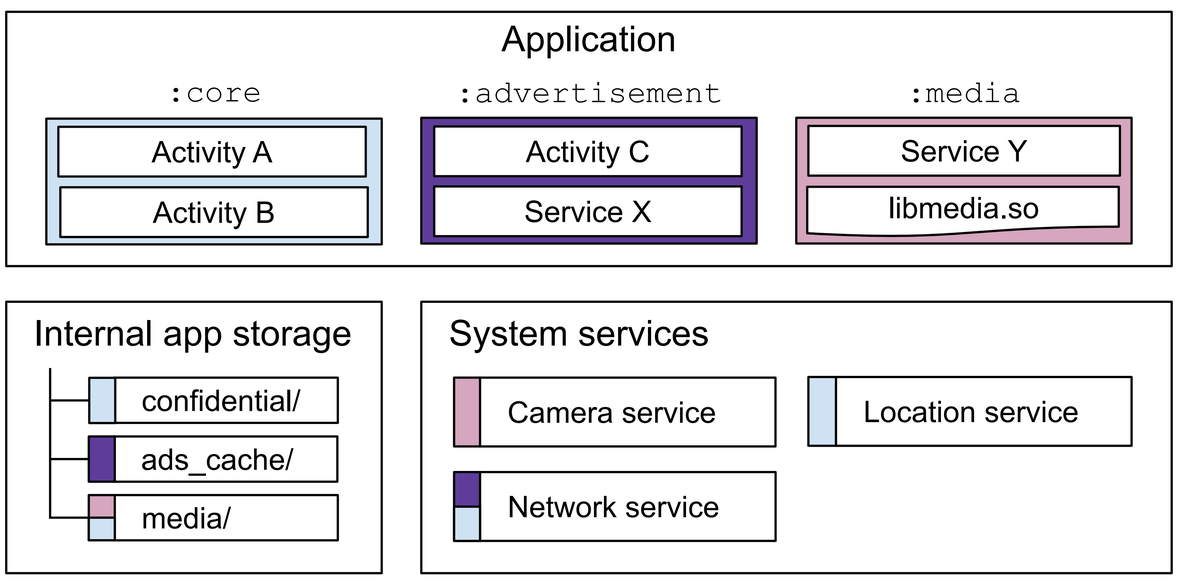
OBJECTIVE

- divide applications into multiple **security contexts**
- **restrict** security context **access** to **internal storage** and system **services**
- control **interactions** among security contexts



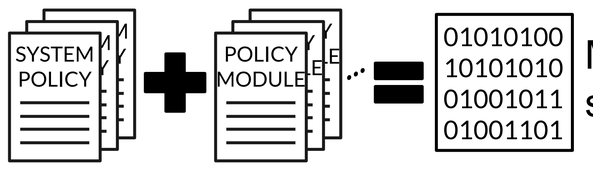
PROPOSAL

- **separate components** into **different** app **processes**
- control with **SELinux** the permissions at process level



IMPLEMENTATION

Fine-grained application **policy** module to control the permissions granted to processes



Merge all policy fragments into the same **monolithic binary policy**

A **compiler-based** approach **prohibits** the installation of policy modules that may **harm** the system or other apps

Several changes to:

- **boot** sequence
- app **installation** procedure
- **services** to support the app lifecycle (e.g., Zygote)

RESULTS

- **limited** app **installation overhead**
  Worst case: ~4s
- no deterioration of the start-up time of components running inside different processes
- running processes provide **warm start of** their **components**
  Activity: ~125 ms→~15 ms **Service:** ~105 ms→~2.5 ms
- unaltered communication overhead between components belonging to different processes
  IPC: ~200 µs
- **slow down of file creation** due to the use of a new system service to update security contexts of files
  Security context update: ~450 µs

DISCUSSION

- by mapping **security contexts** to activities and services, developers can **limit** the impact of a **vulnerability** on both the app and the end user
- our proposal is **consistent with the evolution of Android** and the desire of its designers to let app developers have access to an extensive and flexible collection of security tools
- experimental evaluation shows that the **overhead** introduced by our proposal is **limited** and **compatible with** the additional **security guarantees**