SEApp: Bringing Mandatory Access Control to Android Apps

Speaker: Matthew Rossi

About me

- PhD student at University of Bergamo
- Research on computer security, mostly integrating security features in mobile & cloud systems
- I love to solve problems and engage with projects that require me to learn new things
- I also love sports, traveling, and hiking

Agenda

- How Android isolates applications
- Limitations
- How attackers could exploit these limitations
- SEApp
- Latest evolutions

Android platform security model

Android's security measures:

- **defense in depth** – an approach that does not immediately fail when a single assumption is violated or a single implementation bug is found
- **safe by design/default** – the default use of an operating system component or service should always protect security and privacy

Permissions

By default, an Android application can only **access** a **limited** range of system **resources**

To make use of the protected APIs, an application must define the list of Permissions it needs in its manifest

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.showcaseapp">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.INTERNET" />
```

Kernel-level isolation and containment

Android takes advantage of Linux access control mechanisms to setup a kernel-level Application sandbox which:

- **isolates** apps from each other
- **protects** apps and the system **from malicious apps**

Since the Application sandbox is in the kernel, this extends to both native code and OS applications

Unix permissions (1 of 2)

Android enforces security between apps and the system at the **process-level** through **UNIX-style** user separation of processes and file **permissions**

Each app is assigned to a **unique user and group IDs**

Unix permissions (2 of 2)

 # ps -Ao user,group,name

```
u0_a101      u0_a101  com.android.calendar
u0_a79       u0_a79   com.android.messaging
u0_a56       u0_a56   com.android.packageinstaller
u0_a58       u0_a58   com.android.permissioncontroller
system       system   com.android.localtransport
```

 # ls -l /data/data

```
drwx------ 5 u0_a101        u0_a101        3488 2021-03-10 23:32 com.android.calendar
drwx------ 4 u0_a41         u0_a41         3488 2021-03-10 23:32 com.android.calllogbackup
drwx------ 5 u0_a105        u0_a105        3488 2021-03-10 23:32 com.android.camera2
drwx------ 4 u0_a77         u0_a77         3488 2021-03-10 23:32 com.android.captiveportallogin
```

SELinux (1 of 2)

SELinux is a mandatory access control system for the Linux operating system

Android takes advantage of **SELinux** to greatly **limit** the potential **damage** of a **compromised device**

SELinux (2 of 2)

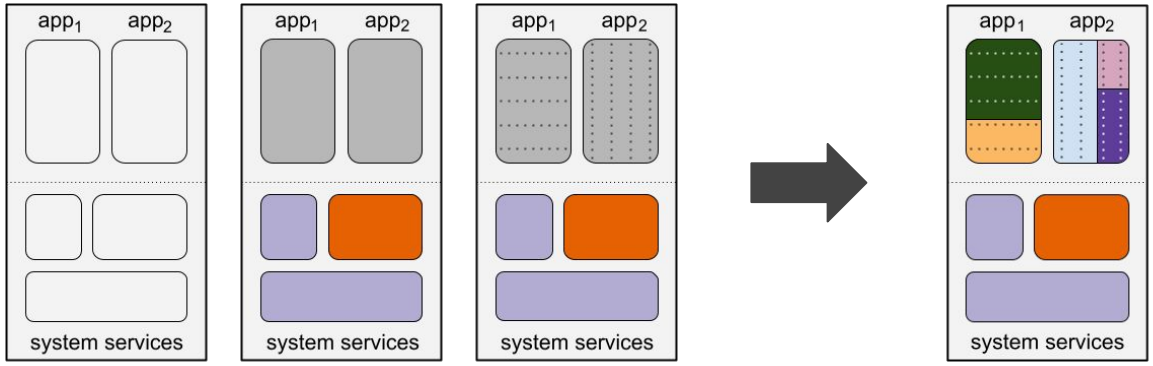
 # ps -AZo name

```
u:r:untrusted_app_27:s0:c101,c256,c512,c768 com.android.calendar
u:r:platform_app:s0:c512,c768               com.android.messaging
u:r:platform_app:s0:c512,c768               com.android.packageinstaller
u:r:platform_app:s0:c512,c768               com.android.permissioncontroller
u:r:system_app:s0                           com.android.localtransport
```

 # ls -lZ /data/data

```
u:object_r:app_data_file:s0:c101,c256,c512,c768  3488 2021-03-10 23:32 com.android.calendar
u:object_r:privapp_data_file:s0:c512,c768        3488 2021-03-10 23:32 com.android.calllogbackup
u:object_r:app_data_file:s0:c105,c256,c512,c768  3488 2021-03-10 23:32 com.android.camera2
u:object_r:app_data_file:s0:c77,c256,c512,c768   3488 2021-03-10 23:32 com.android.captiveportallogin
```

Evolution


app1
app2
system services
Before Android 4.3
Android 4.3 - Android 8
Since Android 9
SEApp

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**
- **holds** the **app privileges**

Use case: file sharing

Many applications store both confidential data and share contents with other apps

Applications may **leak private data**

Every component of an application have the **same access** to **internal storage**, so apps may be one vulnerability away from leaking user private data

Use case: media

Most applications people interact with deal with media files (e.g., social networks)

Many applications use **media libraries**

The media library has the **same access** to **internal storage** and the **same permissions** over the **system services** as other app components

Use case: advertising

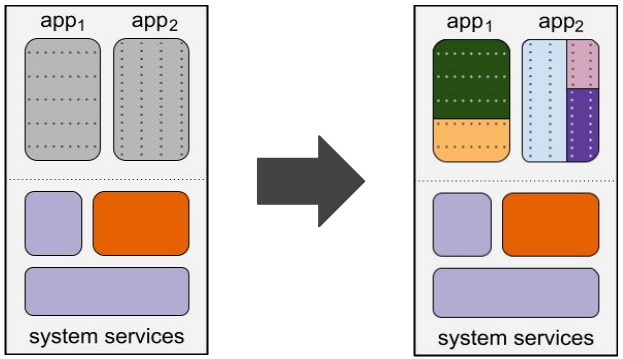
In the Android ecosystem, most applications have an ad-based revenue model

Most application import **3rd-party libraries** to display ads

The components of the ad-library have the **same access** to **internal storage** and the **same permissions** over the **system services** as other app components

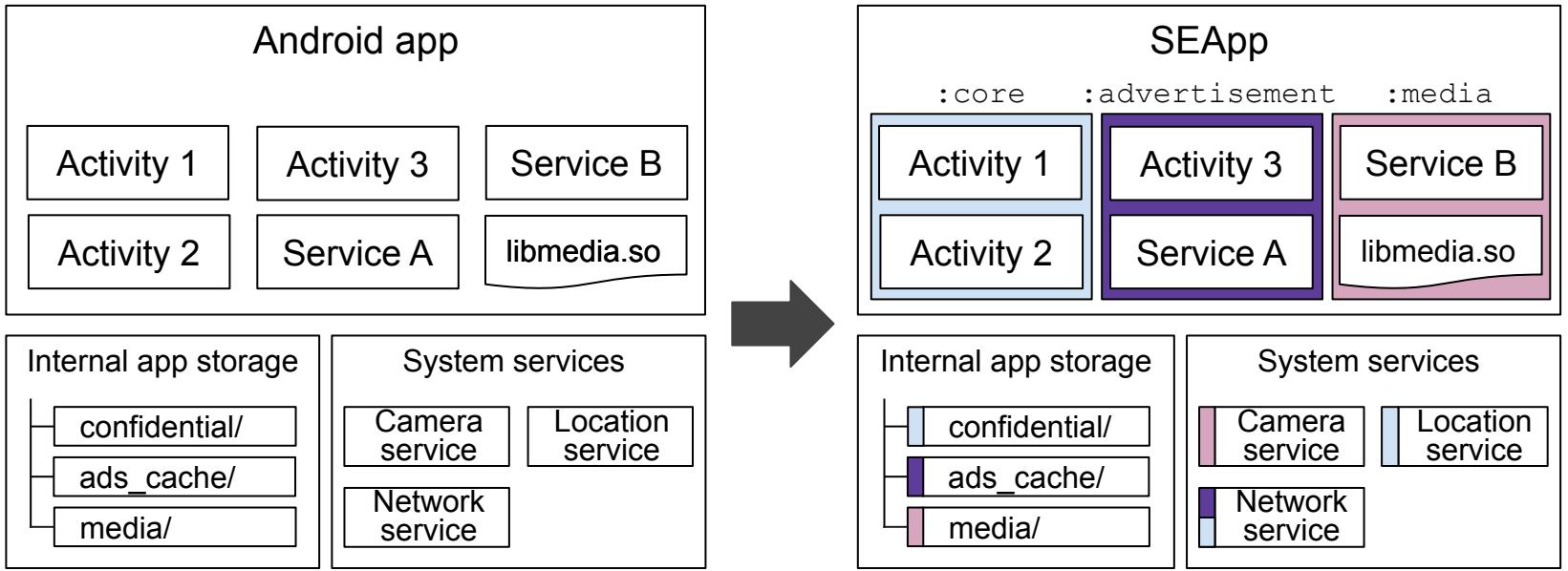
Solution: Security-Enhanced App

Improve the security of applications with the introduction of **intra-application compartmentalization**



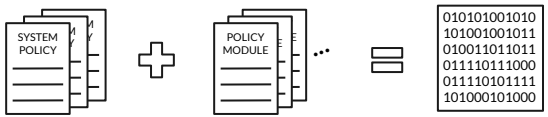
Android 9+ SEApp

Idea


Android app
Activity 1
Activity 3
Service B
Activity 2
Service A
libmedia.so
Internal app storage
confidential/
ads_cache/
media/
System services
Camera service
Location service
Network service
SEApp
:core
:advertisement
:media
Activity 1
Activity 3
Service B
Activity 2
Service A
libmedia.so
Internal app storage
confidential/
ads_cache/
media/
System services
Camera service
Location service
Network service

Changes to the Android OS (1 of 2)

Apps provide a **fine-grained policy module** to control the permissions granted to processes



All policy fragments end up in the same **monolithic binary policy**

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

Changes to the Android OS (2 of 2)

Several changes to:

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

Boot-time support

Since the introduction of **Project Treble**:

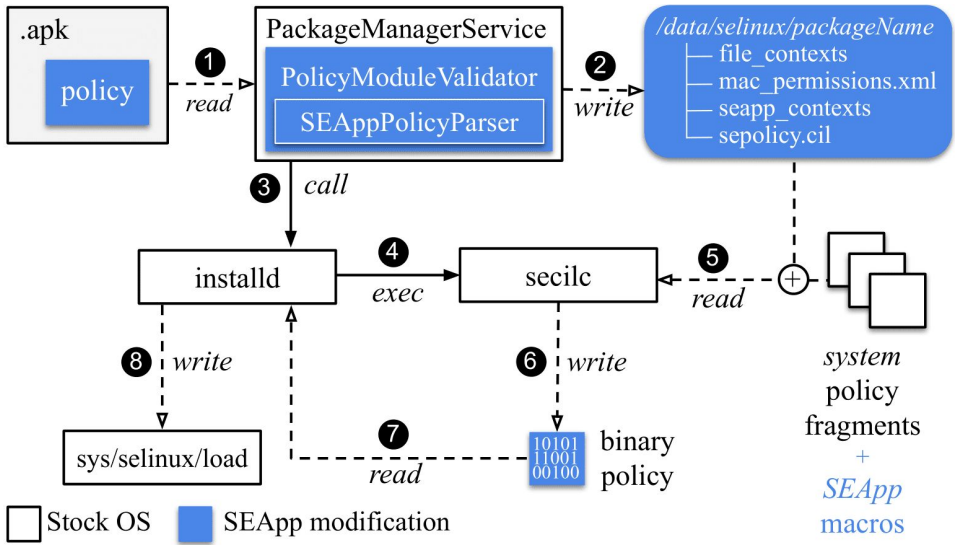
- policy segment updates → **on-device compilation**

Changes to the second stage of boot:

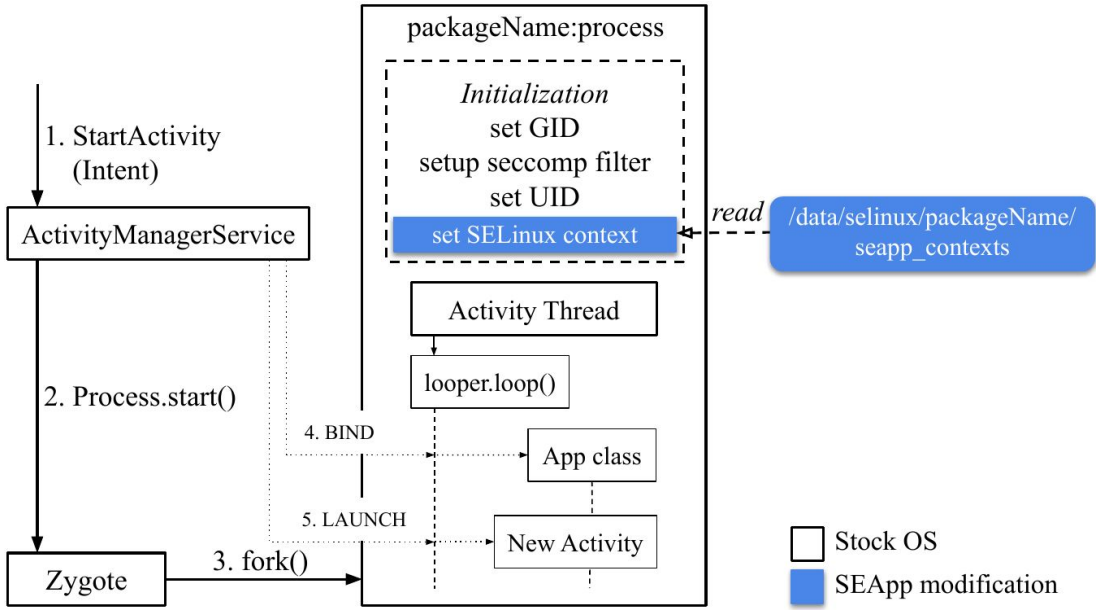
- **mount** the `/data` partition (where policy modules are stored) **early**
- run a new built-in function to **build** and **reload** the **policy**

The policy is not bypassable, since the modules are loaded before any application starts

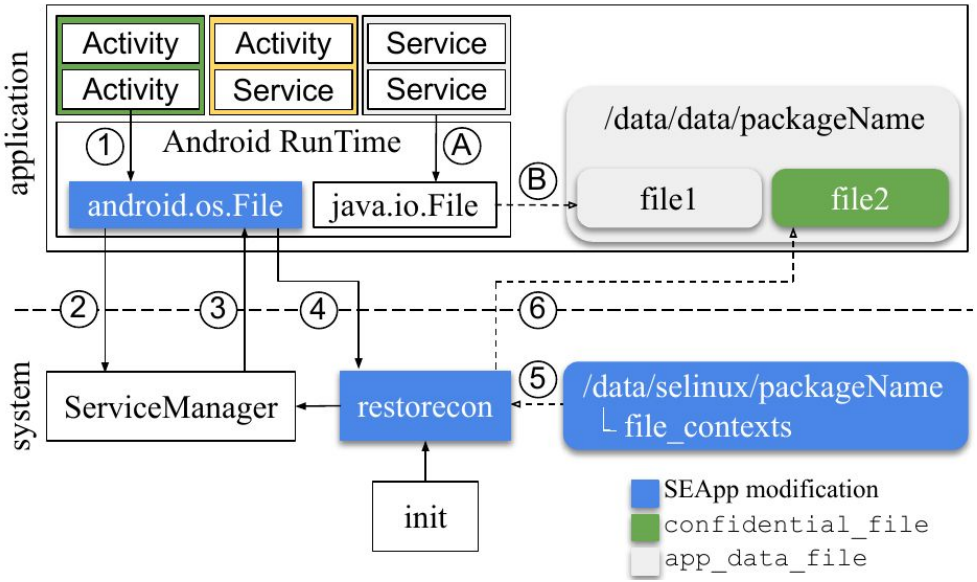
Install-time support



Runtime support: processes



Runtime support: files



Experiments

- **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

Recap

- 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**

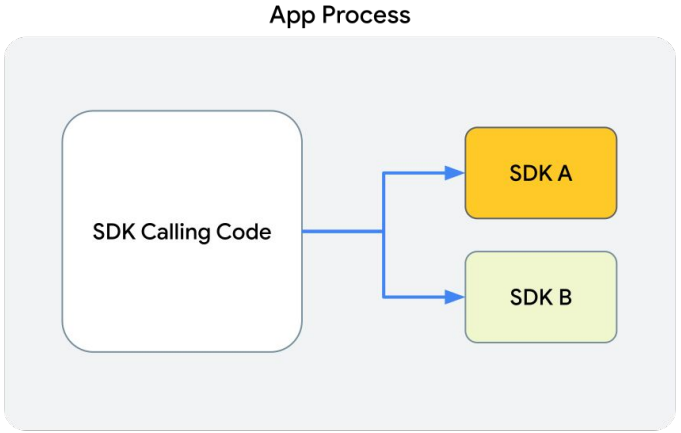
Future evolutions in app isolation (1 of 2)

> In Android 13, we plan to add a new platform capability that allows third-party SDKs to run in a dedicated runtime environment called the SDK Runtime. The SDK Runtime provides the following stronger safeguards and guarantees around user data collection and sharing:
>
> - A modified execution environment
> - Well-defined permissions and data access rights for SDKs

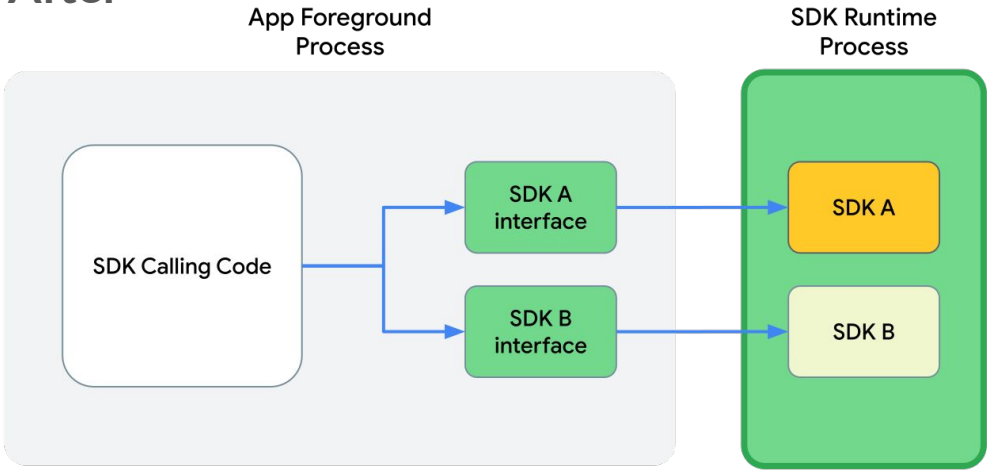
source: developer.android.com

Future evolutions in app isolation (2 of 2)

Before



After



source: [developer.android.com]

Thank you! Any questions?