



COMPUTER
SCIENCE
UNIVERSITY OF PADOVA



DIPARTIMENTO
MATEMATICA
DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

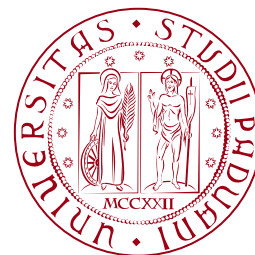
Towards Secure Virtual Apps: Bringing Android Permission Model to Application Virtualization

Master's degree in Computer Science

Candidate: Alberto Lazari

Supervisor: Prof. Eleonora Losiouk

December 12, 2024



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

- App-level virtualization: run applications inside a container app
- Permissions for virtual apps shared with container
- Android's sandbox model violated: unintended access to restricted resources
- Analyze Android's permission model → emulate it in virtual environment

- App-level virtualization: run applications inside a container app
- Permissions for virtual apps shared with container
- Android's sandbox model violated: unintended access to restricted resources
- Analyze Android's permission model → emulate it in virtual environment

- App-level virtualization: run applications inside a container app
- Permissions for virtual apps shared with container
- Android's sandbox model violated: unintended access to restricted resources
- Analyze Android's permission model → emulate it in virtual environment

- App-level virtualization: run applications inside a container app
- Permissions for virtual apps shared with container
- Android's sandbox model violated: unintended access to restricted resources
- Analyze Android's permission model → emulate it in virtual environment

Background

Motivation

Android Permission Model

Virtual Permission Model

Evaluation

Future Directions

Background

Motivation

Android Permission Model

Virtual Permission Model

Evaluation

Future Directions

Android App-Level Virtualization

Apps can load code dynamically to extend its functionalities.

Android App-Level Virtualization

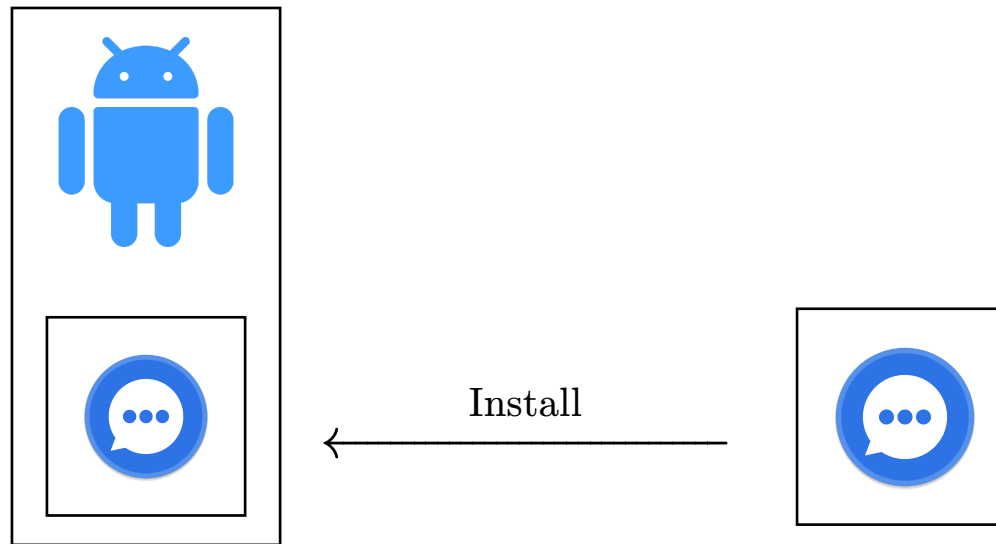
Apps can load code dynamically to extend its functionalities.

Imagine if an app loaded another app's code...

Android App-Level Virtualization

Apps can load code dynamically to extend its functionalities.

Imagine if an app loaded another app's code...



Use Cases

- Virtual apps isolation from system
- App clones (multiple instances)
- Isolated environments for testing and security
- Finer control over app's environment

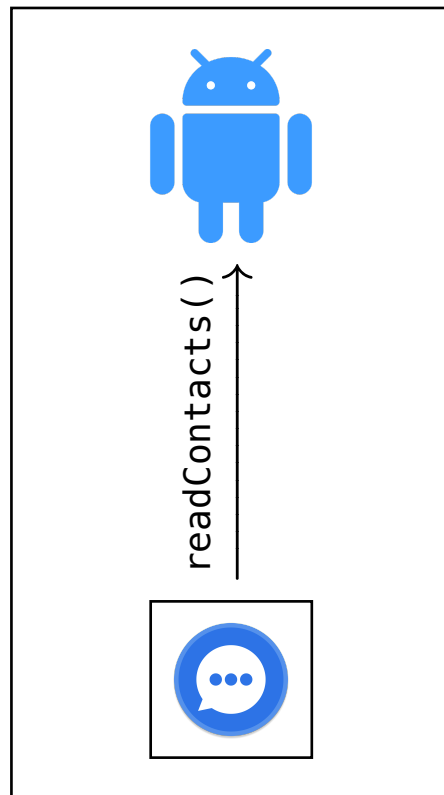
Dynamic Proxies

- Redirect virtual apps requests to the system
- Necessary because apps live inside the container
- Android knows nothing about them

Dynamic Proxies

- Redirect virtual apps requests to the system
- Necessary because apps live inside the container
- Android knows nothing about them
- Let's see an example

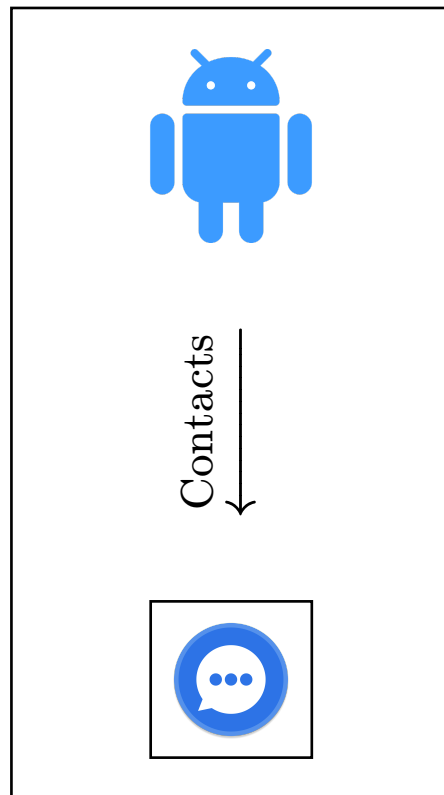
Dynamic Proxies



`(container) readContacts()` →



Dynamic Proxies



← (container) Contacts



Background

Motivation

Android Permission Model

Virtual Permission Model

Evaluation

Future Directions

Permissions in Virtual Frameworks

- The system enforces permissions normally on the container
- Android knows nothing about virtual apps
 - ⇒ container has full responsibility over virtual apps
 - ⇒ permissions are not enforced on virtual apps

Permissions in Virtual Frameworks

- The system enforces permissions normally on the container
- Android knows nothing about virtual apps
 - ⇒ container has full responsibility over virtual apps
 - ⇒ permissions are not enforced on virtual apps

Permissions in Virtual Frameworks

- The system enforces permissions normally on the container
- Android knows nothing about virtual apps
 - ⇒ container has full responsibility over virtual apps
 - ⇒ permissions are not enforced on virtual apps

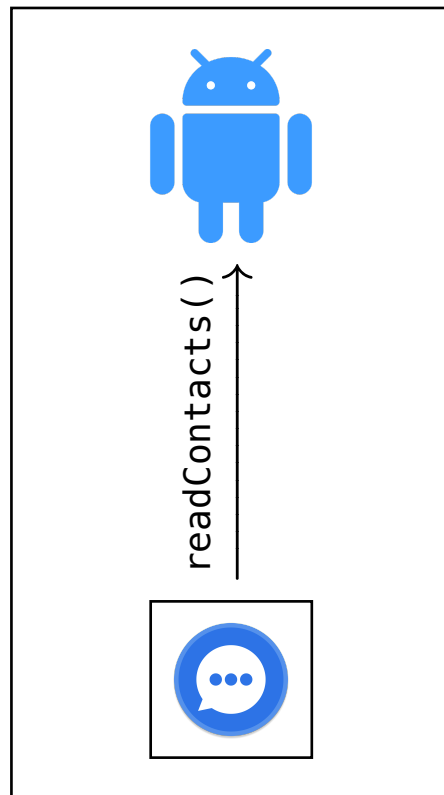
Permissions in Virtual Frameworks

- The system enforces permissions normally on the container
- Android knows nothing about virtual apps
 - ⇒ container has full responsibility over virtual apps
 - ⇒ permissions are not enforced on virtual apps

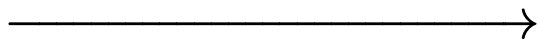
Permissions in Virtual Frameworks

- The system enforces permissions normally on the container
- Android knows nothing about virtual apps
 - ⇒ container has full responsibility over virtual apps
 - ⇒ permissions are not enforced on virtual apps
- Works fine for a single app

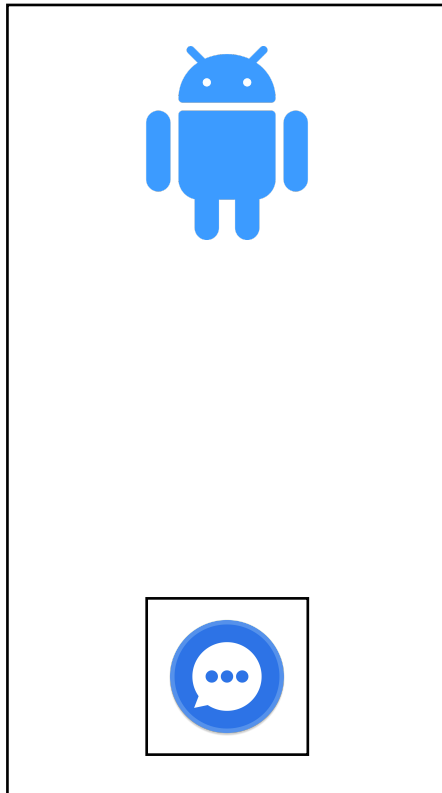
Permissions in Virtual Frameworks



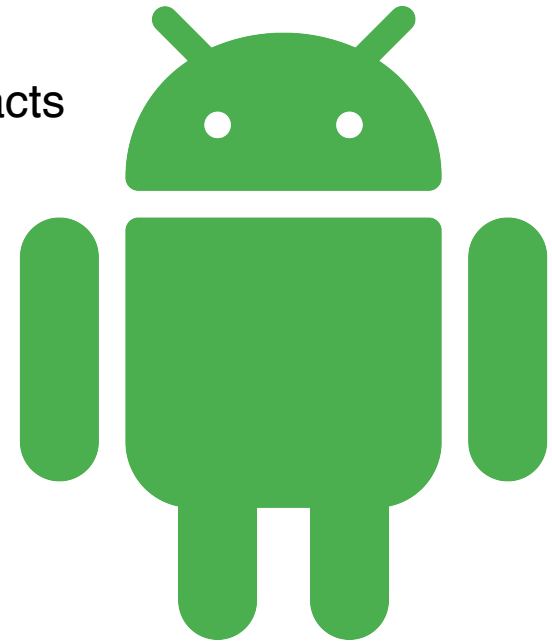
`(container) readContacts()`



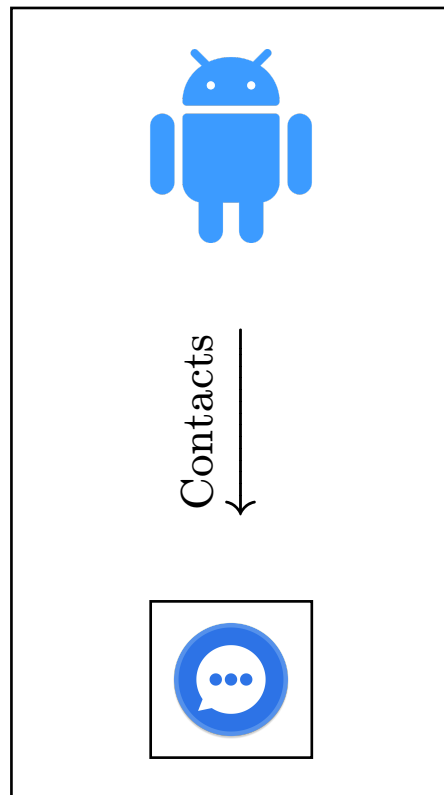
Permissions in Virtual Frameworks



Ok, container has Contacts
permission



Permissions in Virtual Frameworks

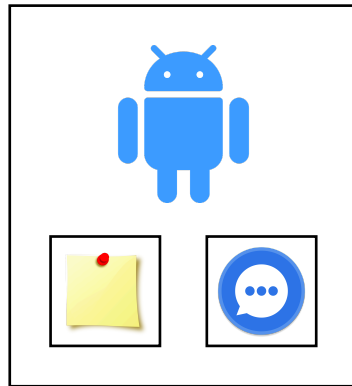


← (container) Contacts



What About Multiple Apps?

- Add notes app, declaring no permissions



What About Multiple Apps?

- Add notes app, declaring no permissions
- Messages app requests contacts



What About Multiple Apps?

- Add notes app, declaring no permissions
- Messages app requests contacts
- Notes app is granted the permission too



What About Multiple Apps?

- Containers usually have many permissions
⇒ extensive attack surface for malicious apps
- Need to isolate virtual apps → **virtual permission model**
- Consistency with Android's model → **analyze it**

What About Multiple Apps?

- Containers usually have many permissions
⇒ extensive attack surface for malicious apps
- Need to isolate virtual apps → **virtual permission model**
- Consistency with Android's model → **analyze it**

What About Multiple Apps?

- Containers usually have many permissions
⇒ extensive attack surface for malicious apps
- Need to isolate virtual apps → **virtual permission model**
- Consistency with Android's model → **analyze it**

Background

Motivation

Android Permission Model

Virtual Permission Model

Evaluation

Future Directions

Permissions Overview

Three *protection levels*, based on information sensitivity:

- Normal: minimal effects on system
- Dangerous: access user's private informations
- Signature: features available to same developer

Permissions Overview

Three *protection levels*, based on information sensitivity:

- Normal: minimal effects on system
- Dangerous: access user's private informations
- Signature: features available to same developer

Permissions Overview

Three *protection levels*, based on information sensitivity:

- Normal: minimal effects on system
- Dangerous: access user's private informations
- Signature: features available to same developer

Permissions Overview

Three *protection levels*, based on information sensitivity:

- Normal: minimal effects on system
- Dangerous: access user's private informations
- Signature: features available to same developer

Permissions Overview

Three *protection levels*, based on information sensitivity:

- Normal: minimal effects on system
- Dangerous: access user's private informations
- Signature: features available to same developer

Two categories, based on protection level:

- Install-time: normal + signature
- Runtime: dangerous

Permission Groups

Runtime permissions require direct user approval

Permission Groups

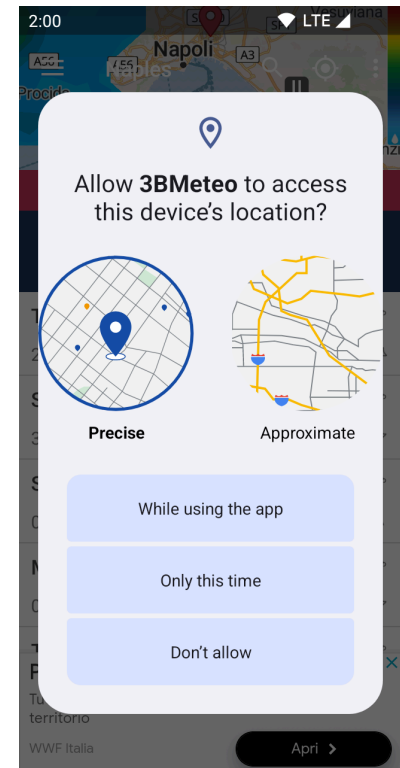
Runtime permissions require direct user approval
⇒ too many requests.

Permission Groups

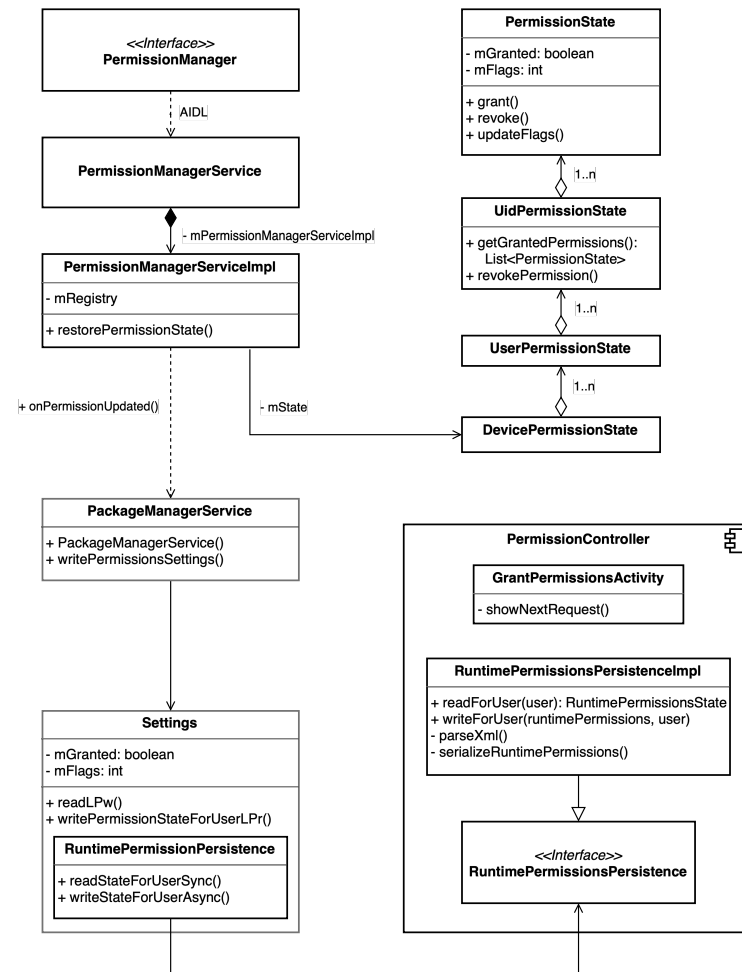
Runtime permissions require direct user approval
⇒ too many requests.

One dialog to rule them all:

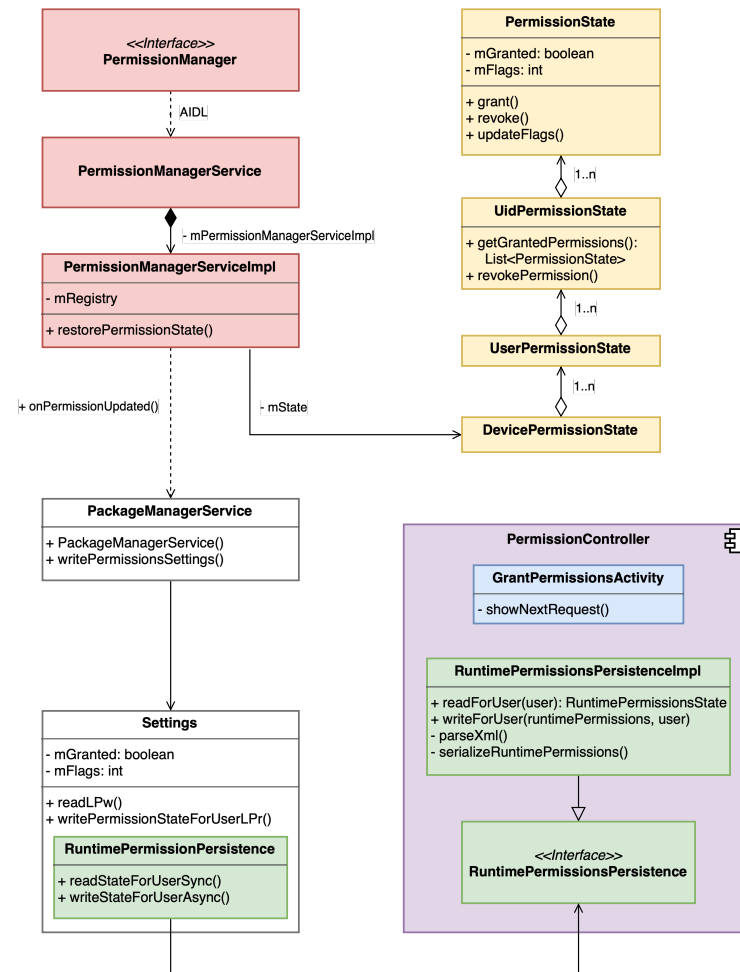
- Similar permissions are requested once
- Reduce user interaction
- Hide complexity guiding choices



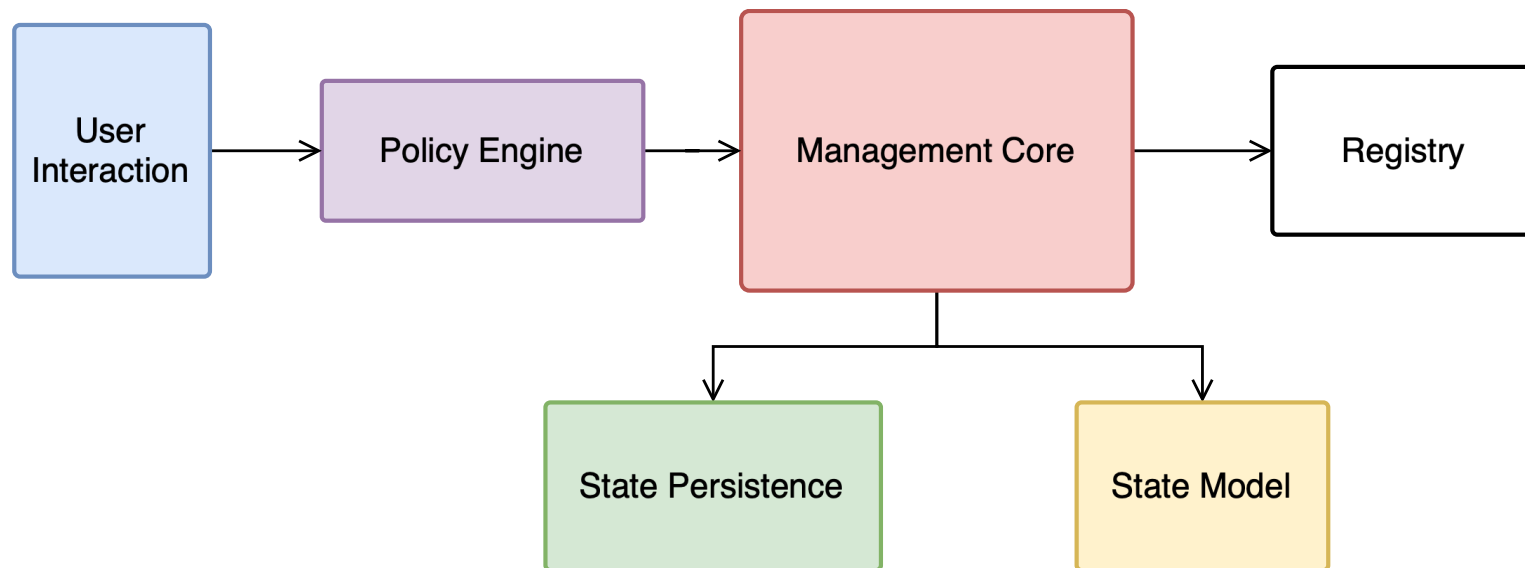
How Does It Translate to Code?



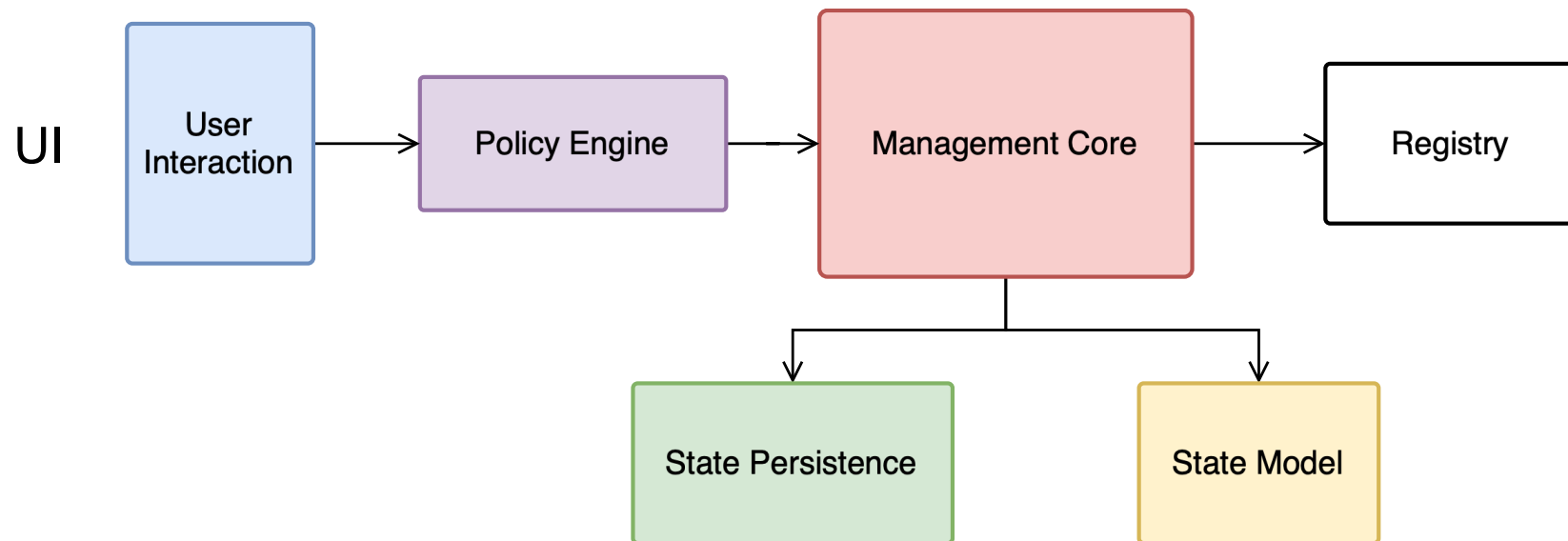
How Does It Translate to Code?



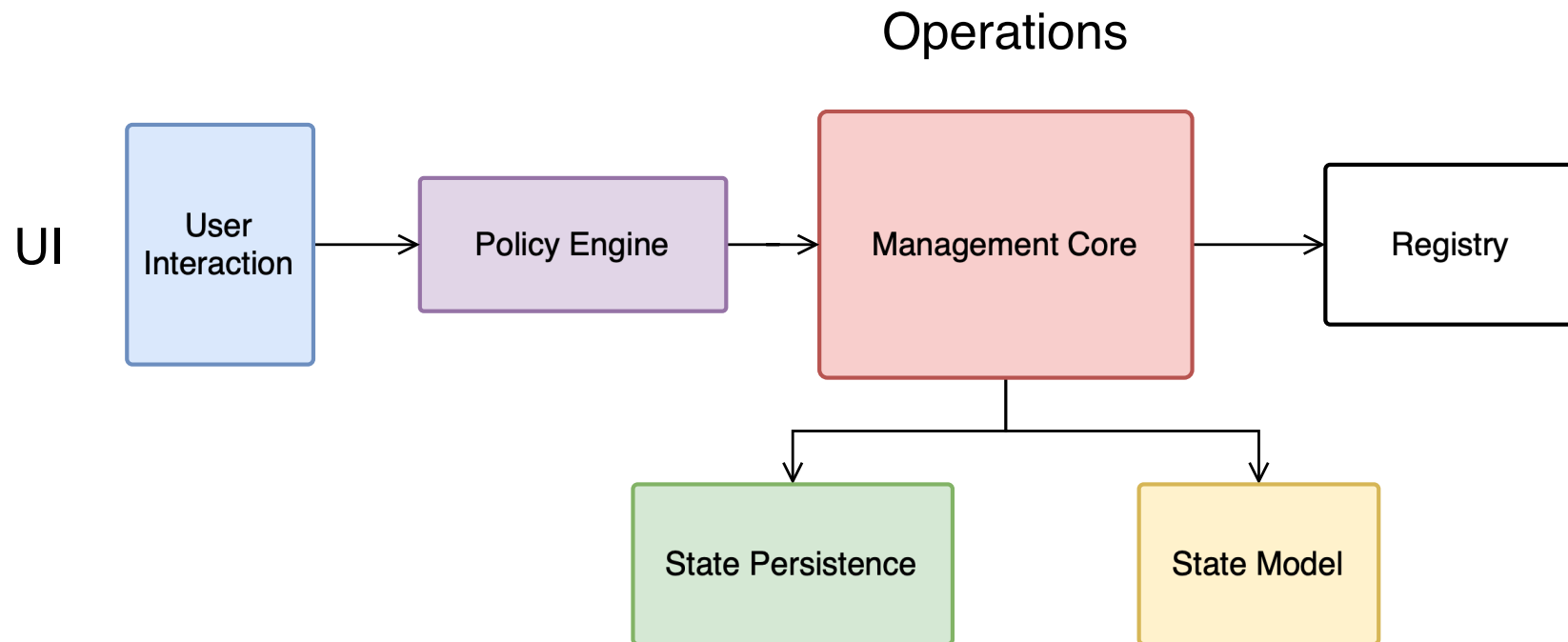
High-Level Components



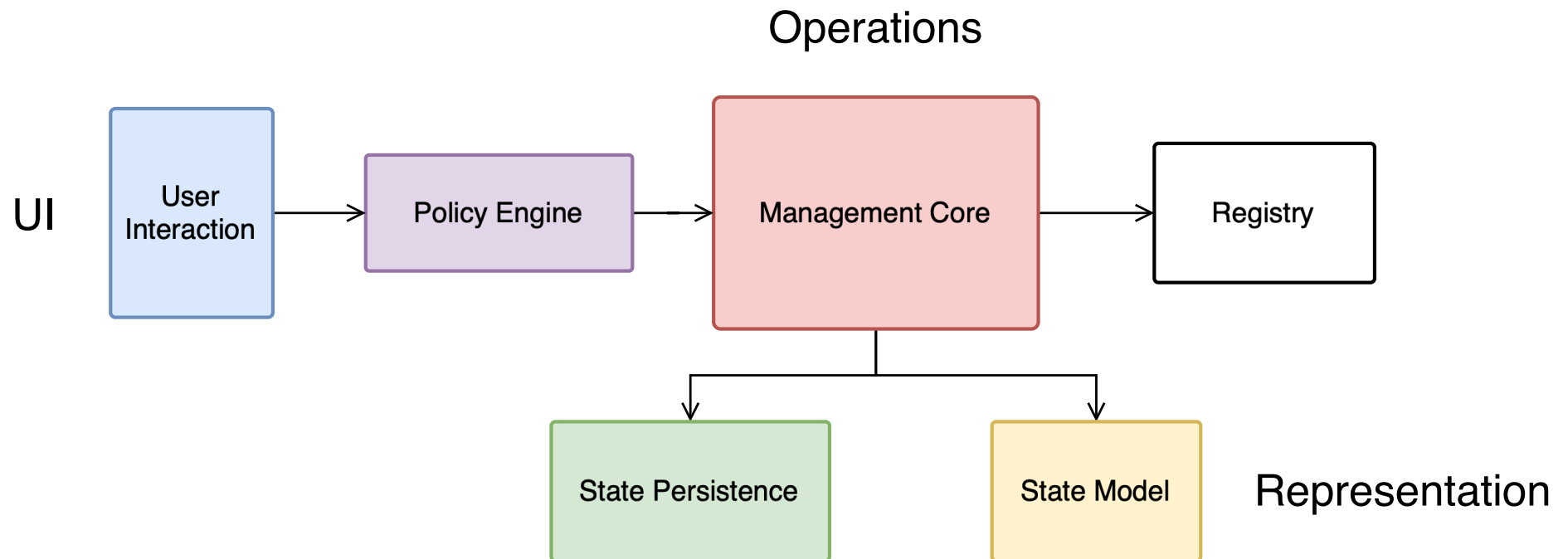
High-Level Components



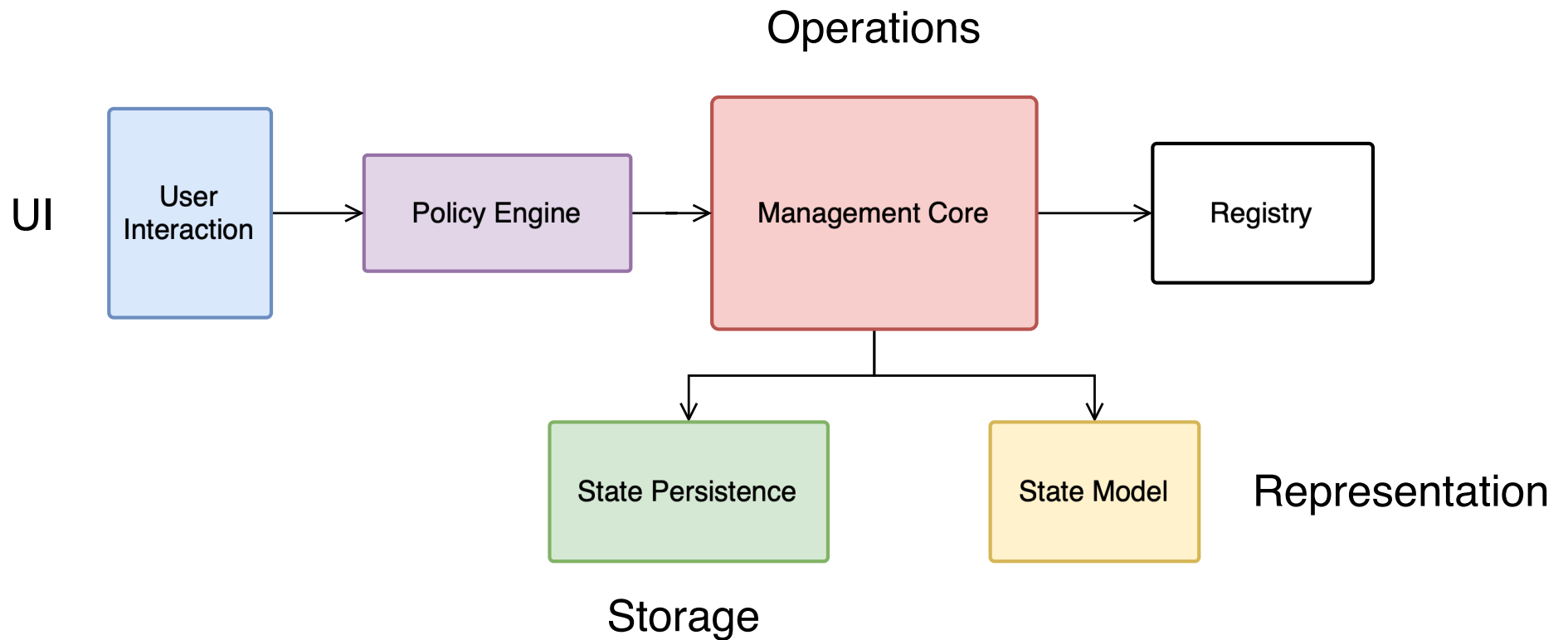
High-Level Components



High-Level Components



High-Level Components



Background

Motivation

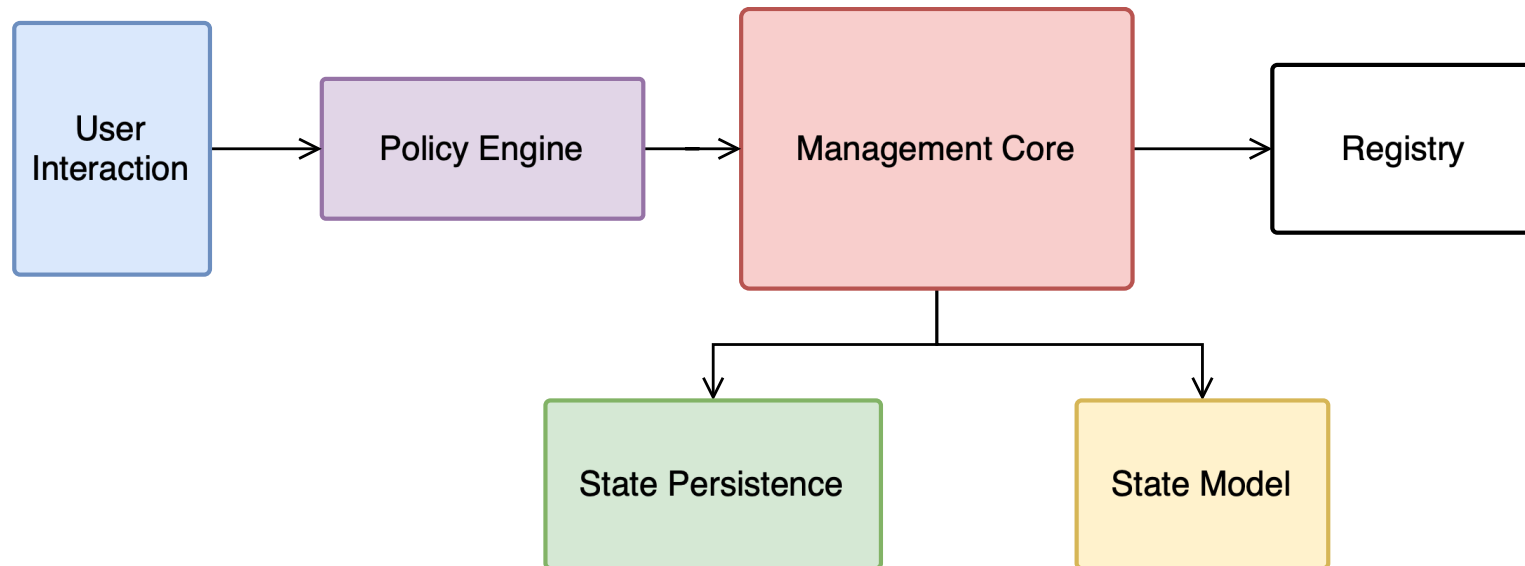
Android Permission Model

Virtual Permission Model

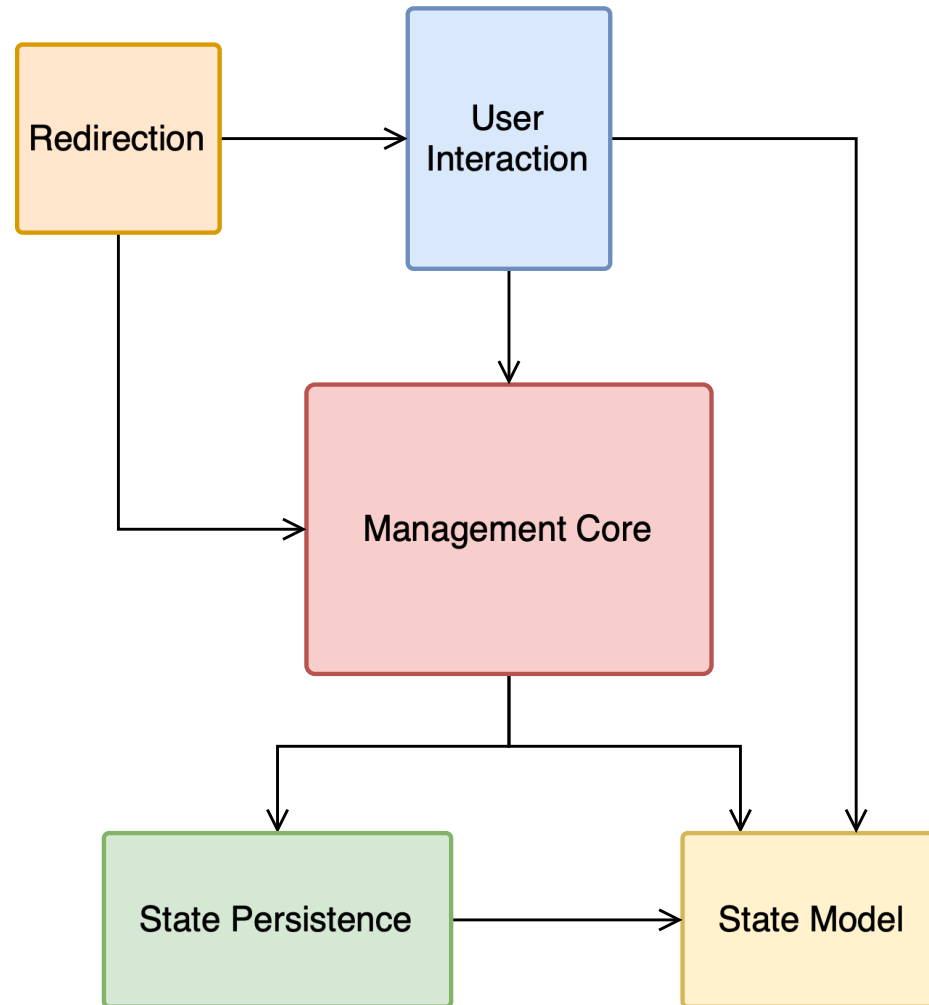
Evaluation

Future Directions

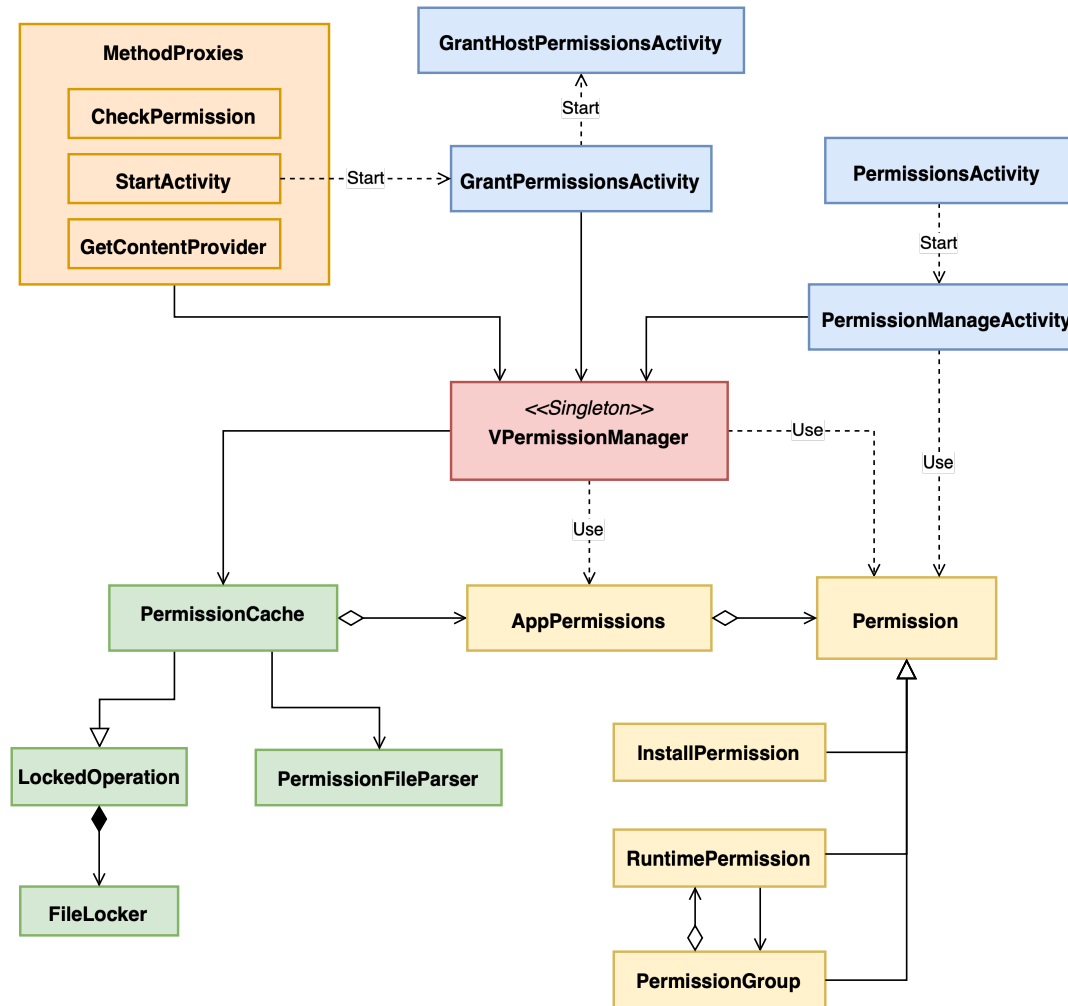
High-Level Components



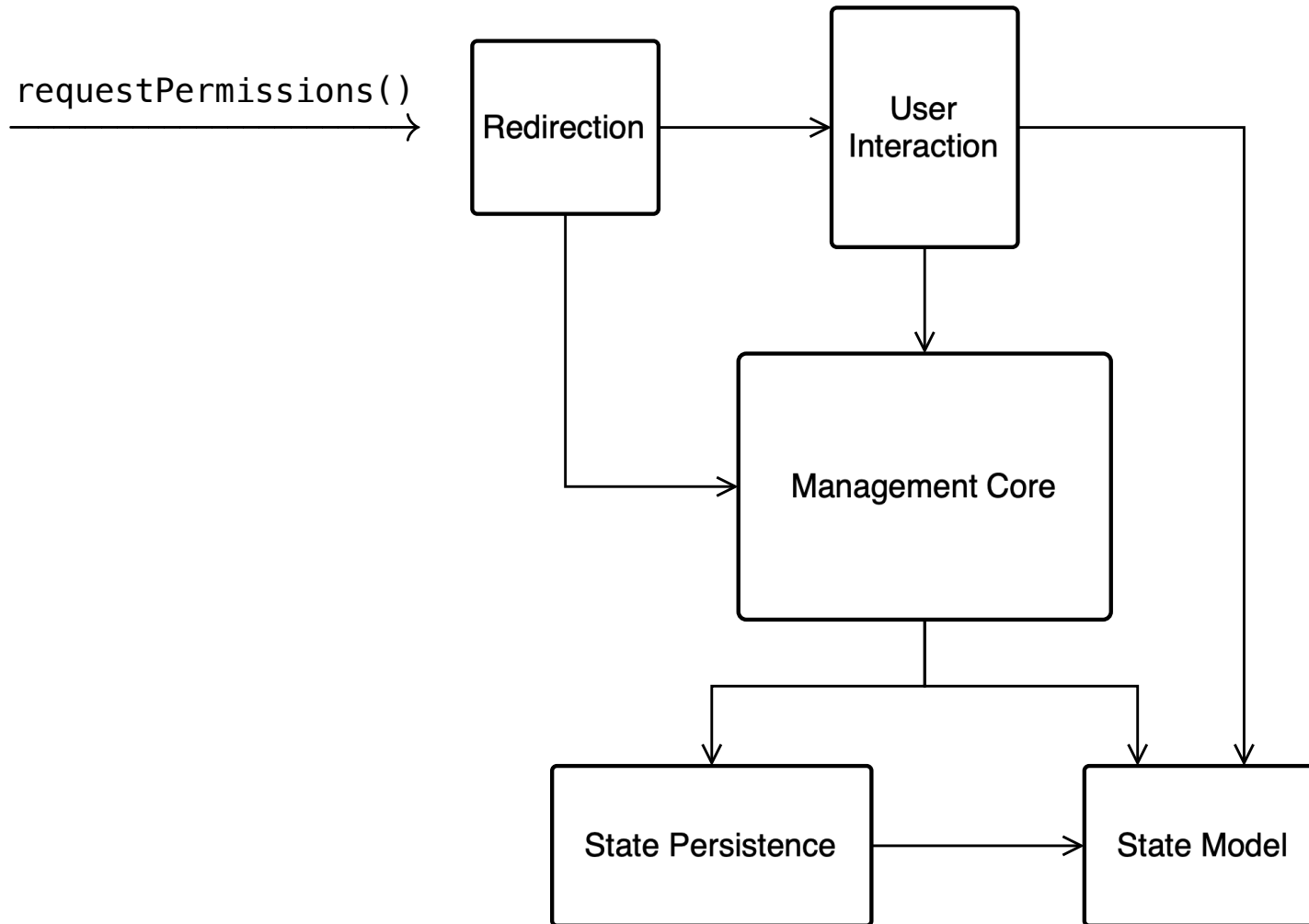
High-Level Components



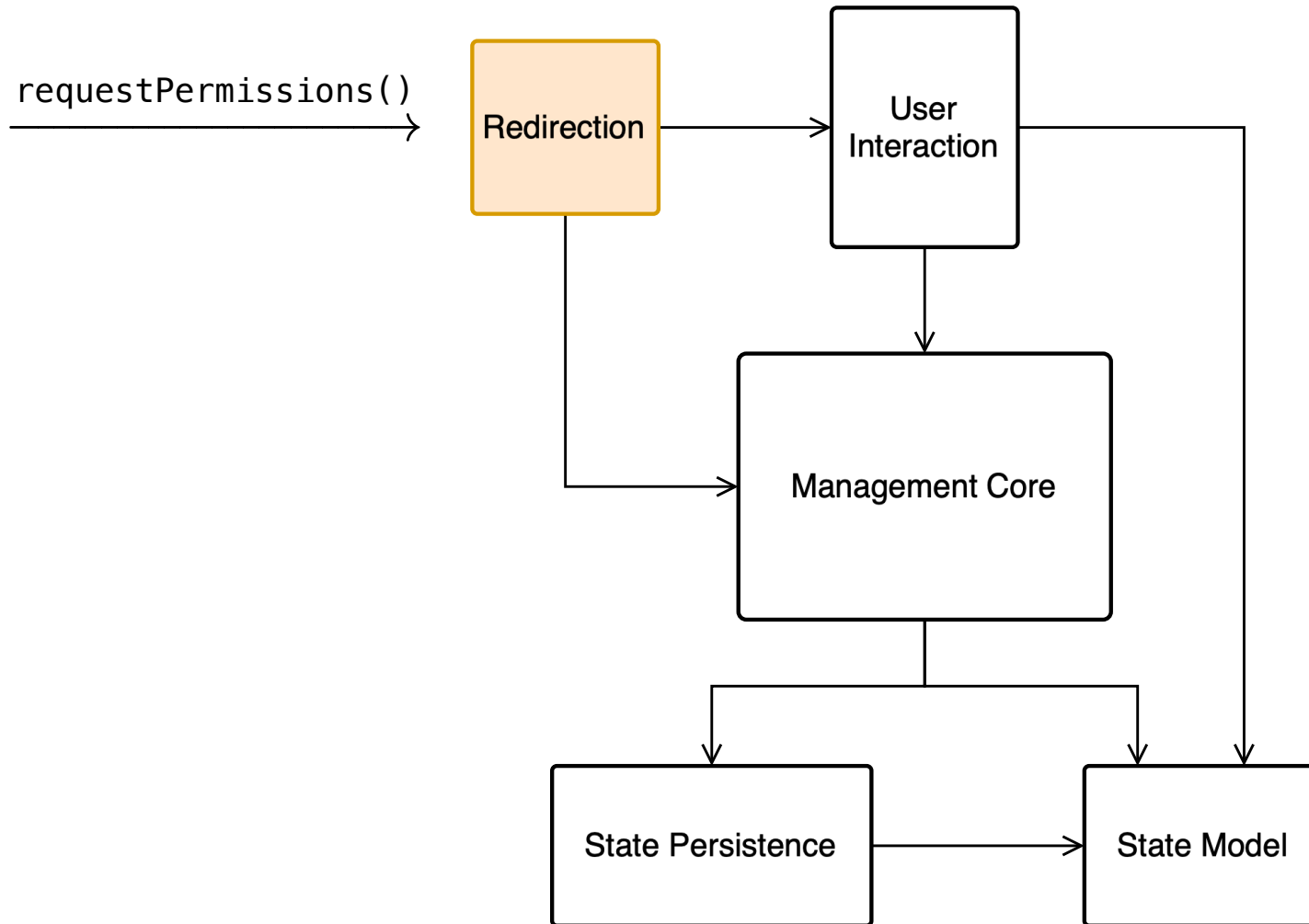
Final Architecture



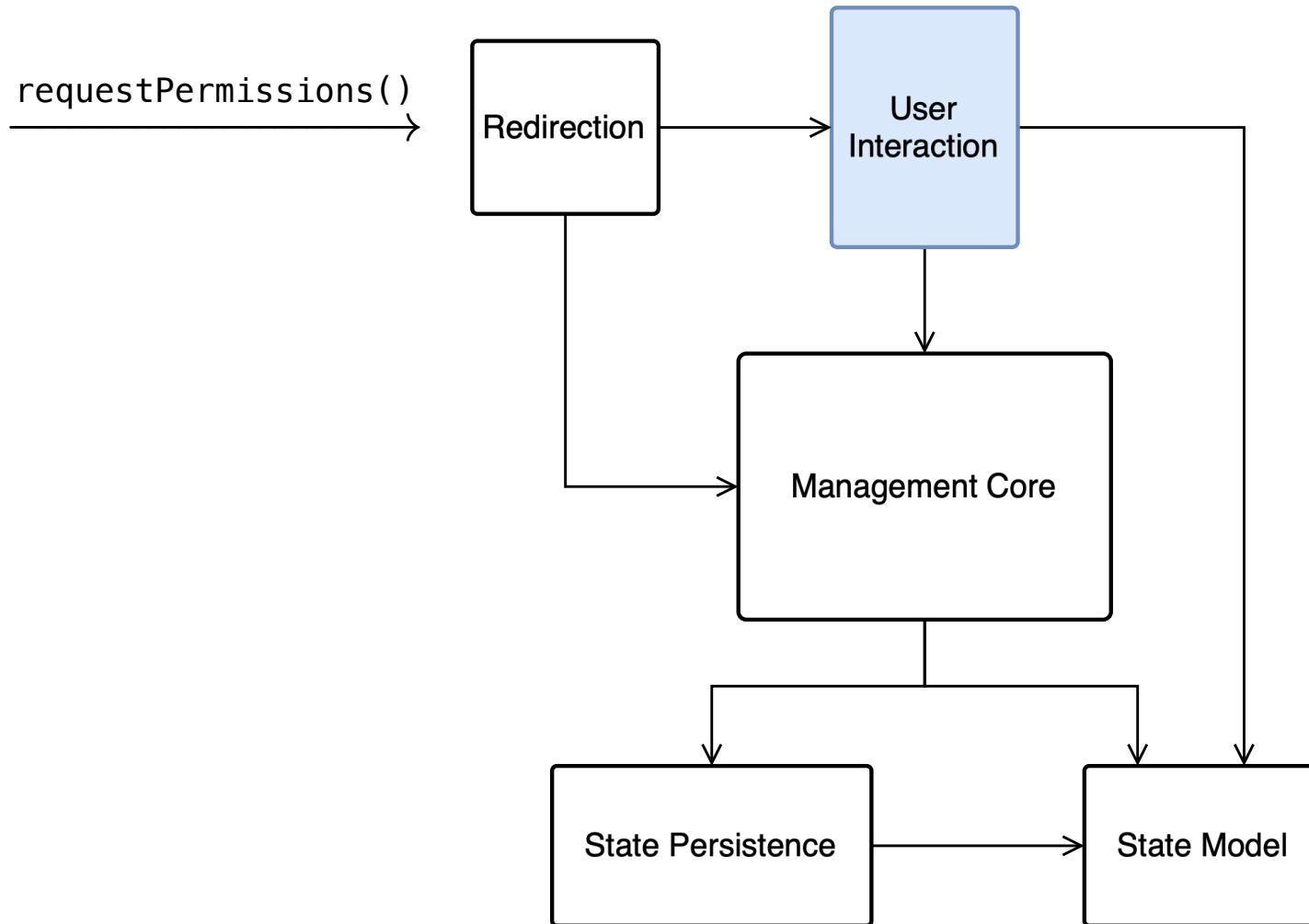
Components Flow



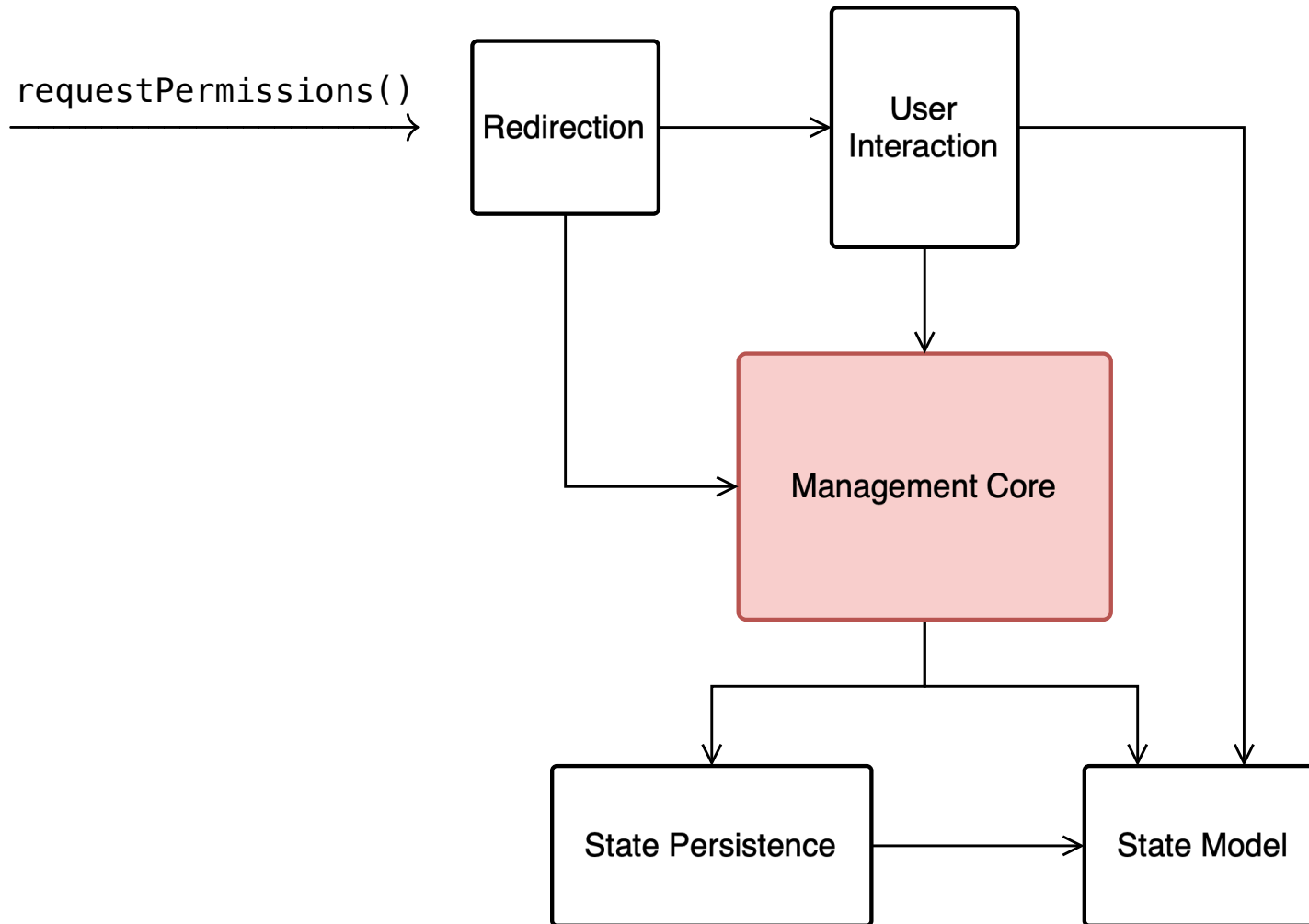
Components Flow



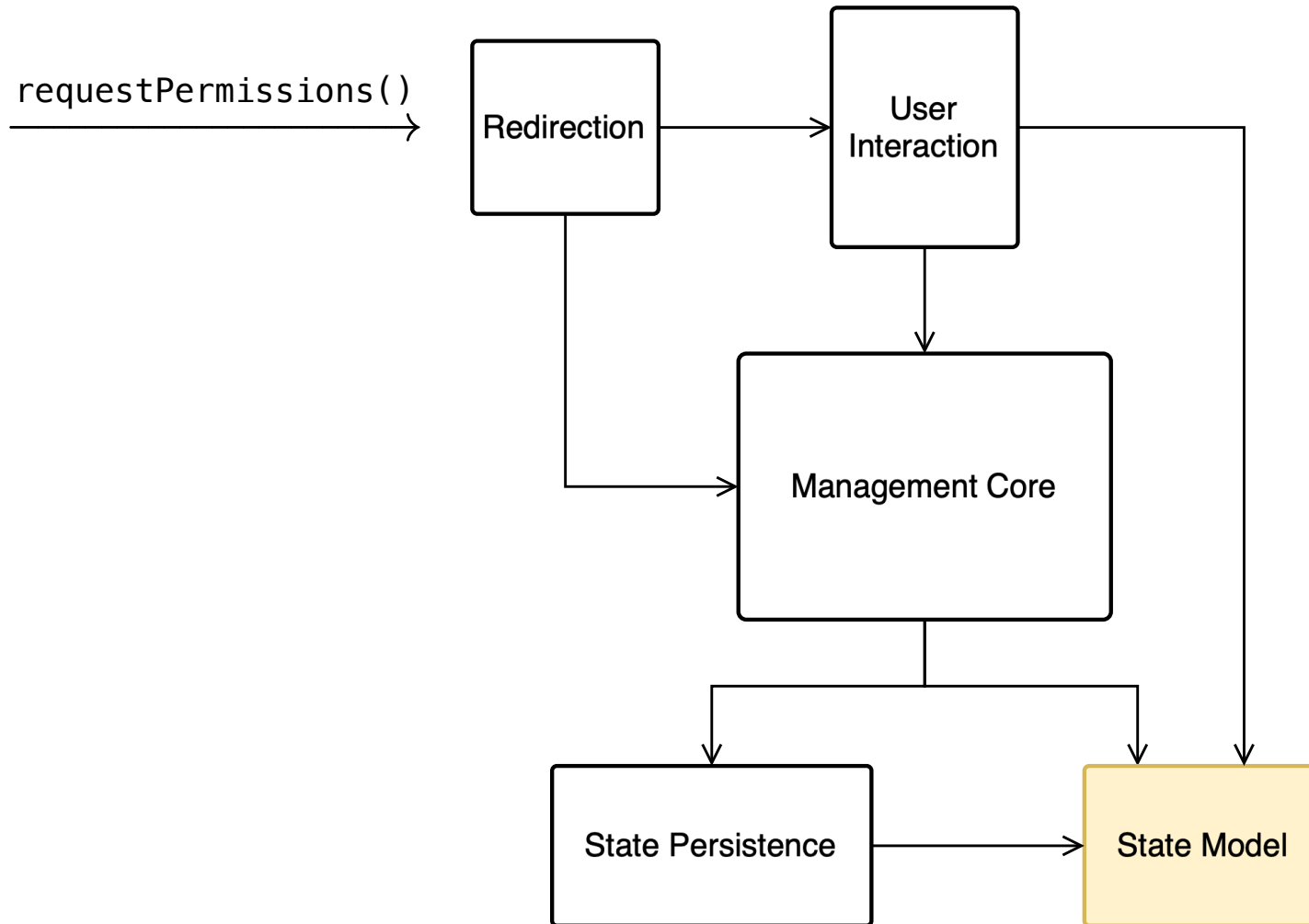
Components Flow



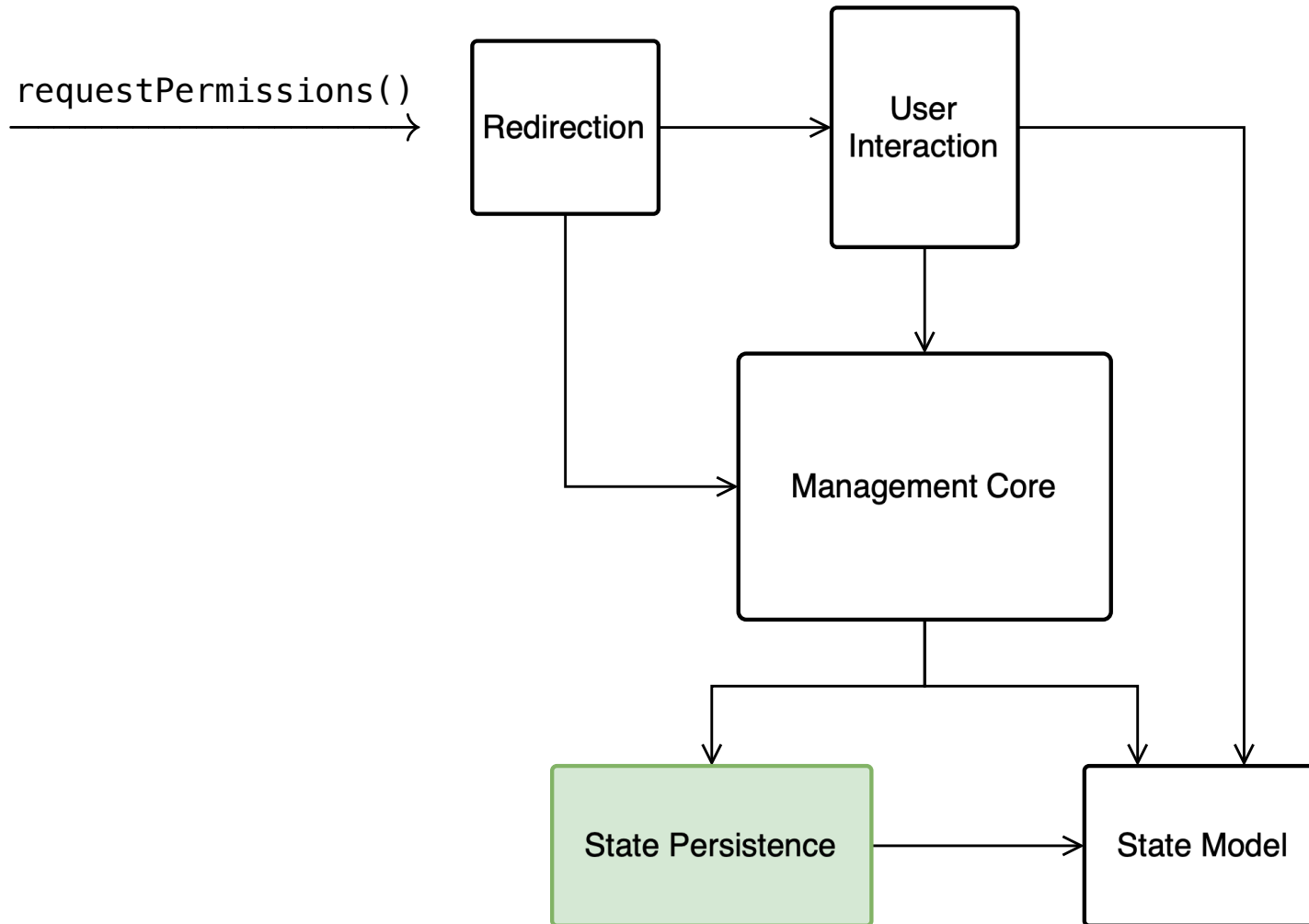
Components Flow



Components Flow



Components Flow



Implementation



- VirtualXposed container app
- VirtualApp underlying virtualization framework
- Android 14 and technological update
- Actual components implementation

Implementation



- VirtualXposed container app
- VirtualApp underlying virtualization framework
- Android 14 and technological update
- Actual components implementation

Implementation



- VirtualXposed container app
- VirtualApp underlying virtualization framework
- Android 14 and technological update
- Actual components implementation

Background

Motivation

Android Permission Model

Virtual Permission Model

Evaluation

Future Directions

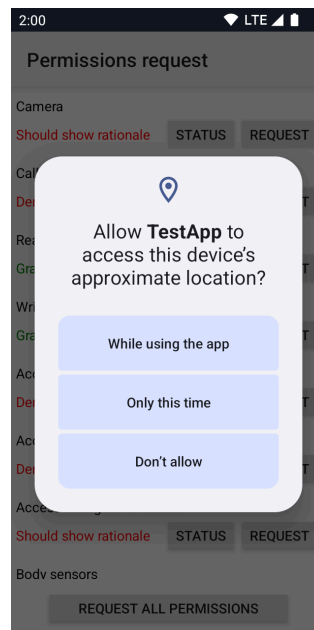
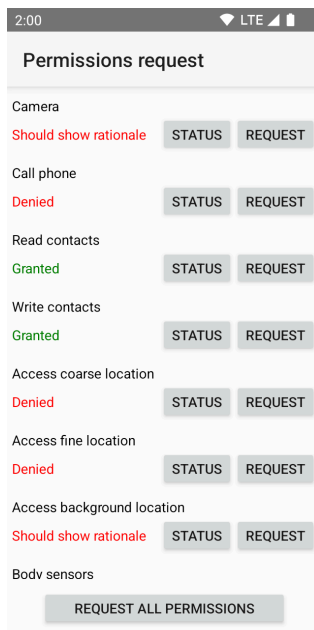
Permission Type Focus

- General model, different behaviors
- Protection levels:
 - Normal: less testing involved
 - Signature: not even available to container
 - Dangerous: most complicated, many edge cases

TestApp

App designed to test key aspects of the model.

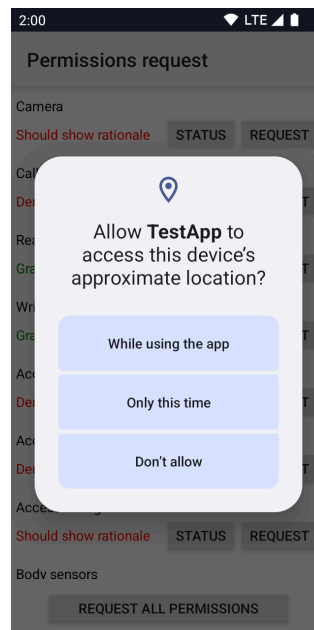
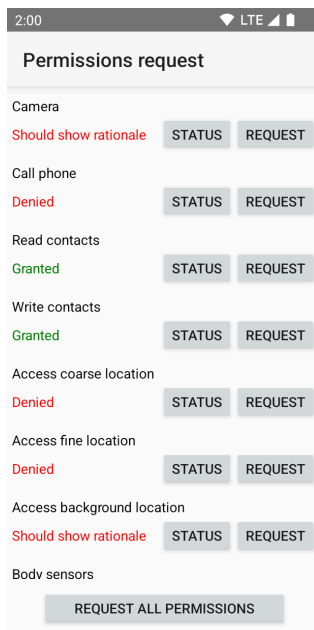
PermissionRequestActivity



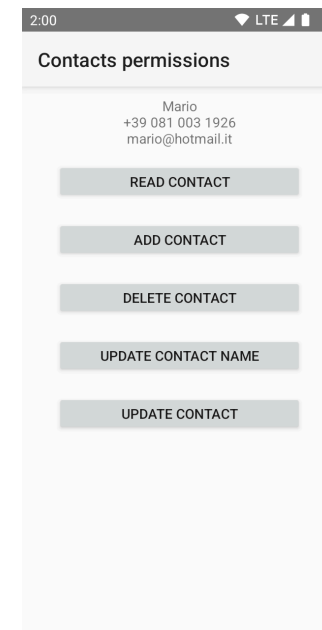
TestApp

App designed to test key aspects of the model.

PermissionRequestActivity

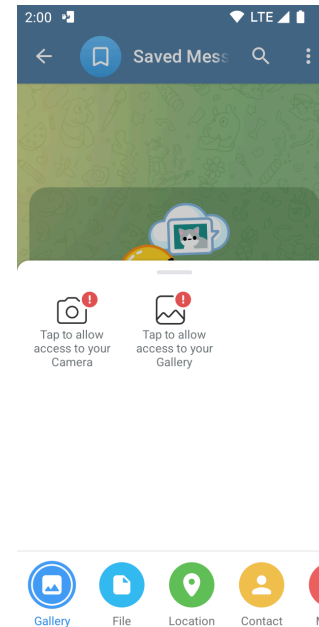


ContactsActivity



Telegram

- Test real-world scenario
- Checks and requests work fine
- Camera patch works



Overall Result

- Proof of concept of general model
- Effective for basic use cases

Overall Result

- Proof of concept of general model
- Effective for basic use cases
- Limitations for actual applications

Background

Motivation

Android Permission Model

Virtual Permission Model

Evaluation

Future Directions

Limitations

- Framework hands off operations to system
- System services perform actual permission checks internally
- Model cannot redirect system's calls

Solutions

- Manual hooks on all possible methods (performed for contacts and camera)
- Automated permission analysis: block every call lacking permissions.
→ Problem: lack of exhaustive mapping
- Re-implement Android's services: include Android's source in virtual framework.
→ Problem: probably still requires adaptation effort

Solutions

- Manual hooks on all possible methods (performed for contacts and camera)
- Automated permission analysis: block every call lacking permissions.
→ Problem: lack of exhaustive mapping
- Re-implement Android's services: include Android's source in virtual framework.
→ Problem: probably still requires adaptation effort

Solutions

- Manual hooks on all possible methods (performed for contacts and camera)
- Automated permission analysis: block every call lacking permissions.
→ Problem: lack of exhaustive mapping
- Re-implement Android's services: include Android's source in virtual framework.
→ Problem: probably still requires adaptation effort

Question Time?



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Thank you for your attention!



COMPUTER
SCIENCE
UNIVERSITY OF PADOVA



DIPARTIMENTO
MATEMATICA
DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"