DeepDroid: Dynamically Enforcing Enterprise Policy on Android Devices

***Xueqiang Wang*[1]**, Kun Sun[2], Yuewu Wang[1], Jiwu Jing[1]

[1]*Institute of Information Engineering, CAS*

[2]*College of William and Mary*

Mon, Feb. 9th, 2015

Outline

- **Introduction**
- Related Work
- DeepDroid
- Evaluation
- Discussion
- Conclusion

Introduction

- Mobile devices are widely used for work purposes.
  - "51% of end users rely on smartphones to perform daily business activities."——Cisco
  - "Android hit 84% smartphone share in Q3 2014"——IDC

Outline

- Introduction
- **Related Work**
- DeepDroid
- Evaluation
- Discussion
- Conclusion

Related Work

- Evolutionary support from Google
  - **Android Permission**
    - Coarse-grained
    - All-or-nothing
    - Lack of run-time configuration
  - **Device Administration APIs**
    - Only provide device-level control on password policy, camera, device wipe, etc.
    - Very limited interfaces (43 in KitKat *VS* 500+ in BlackBerry)

Related Work

- Evolutionary support from Google
  - Introduction of **SEAndroid**
    - Brings flexible MAC to Android
    - Middleware MAC has not been included, even in Android 5.0
    - Unavailable on legacy phones (58.7%<version 4.4)
  - Incorporation of **Knox APIs**
    - A large step towards “Android for Enterprise”
    - Introduces Knox features into AOSP except hardware-based ones
    - Unavailable on legacy phones (98.4%<version 5.0)

Related Work

- Possible solutions
  - **Device OEMs' API**, e.g., SAFE, HTC, 3LM, LG.
  - **Other solutions based on source code modification**
    - Extending permission, e.g., *Compac*[CODASPY'14].
    - Introducing MAC, *e.g., FlaskDroid*[USENIX Security'13], S*EAndroid*[NDSS'13].
    - Dynamic taint tracking, e.g., *TaintDroid*[OSDI'10].
    - Data shadowing, e.g., *AppFence*[CCS'11]
    - **Portability** issue caused by tremendous source code modification.

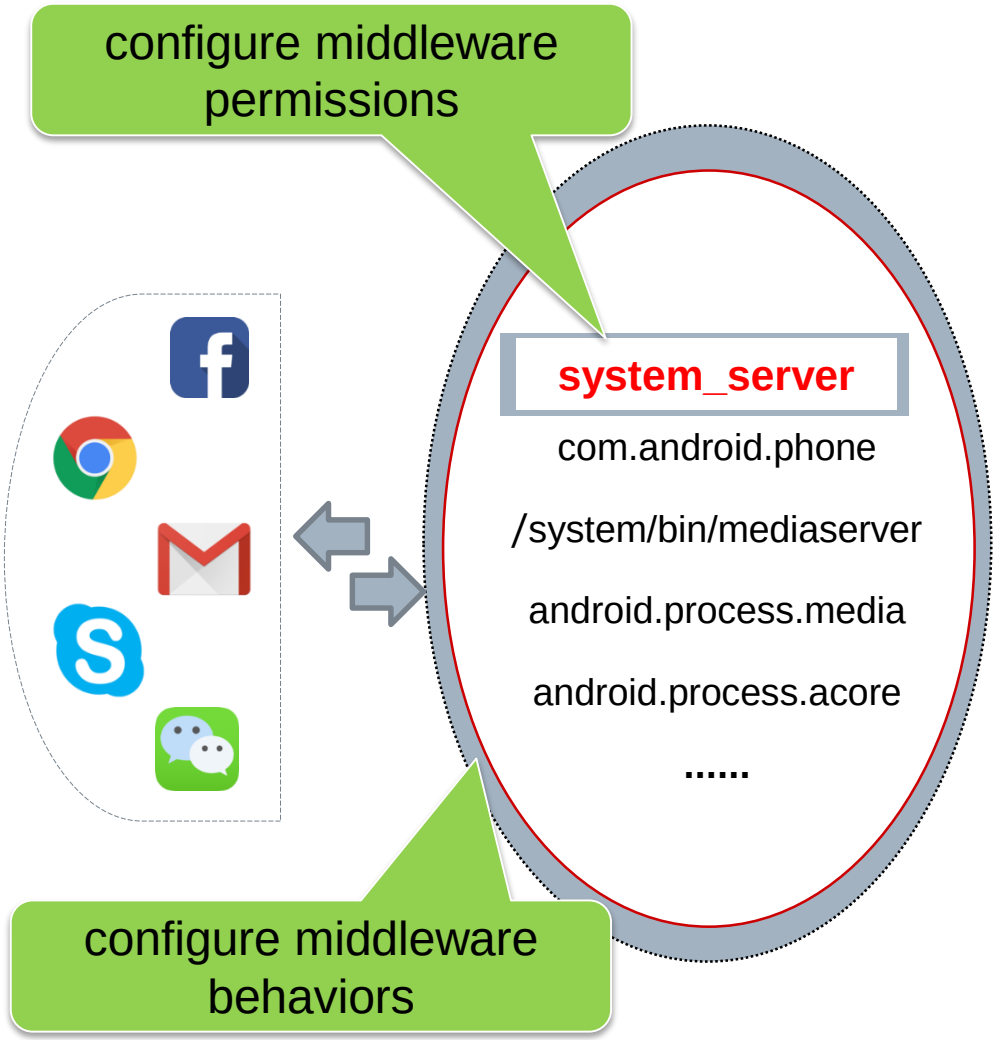
Related Work

- Possible solutions
  - **Rewriting Android apps**
    - Dalvik bytecode rewriting, e.g., *I-ARM-Droid*[MoST'12]
    - Low-level libc interposition, e.g., *Aurasium*[USENIX Security'12]
    - On-the-phone instrumentation, e.g., *AppGuard*[TACAS'13]
    - Require no modification to smartphone's firmware and require no root access
    - Lack of **isolation** between app and monitoring code.

Outline

- Introduction
- Related Work
- **DeepDroid**
- Evaluation
- Discussion
- Conclusion

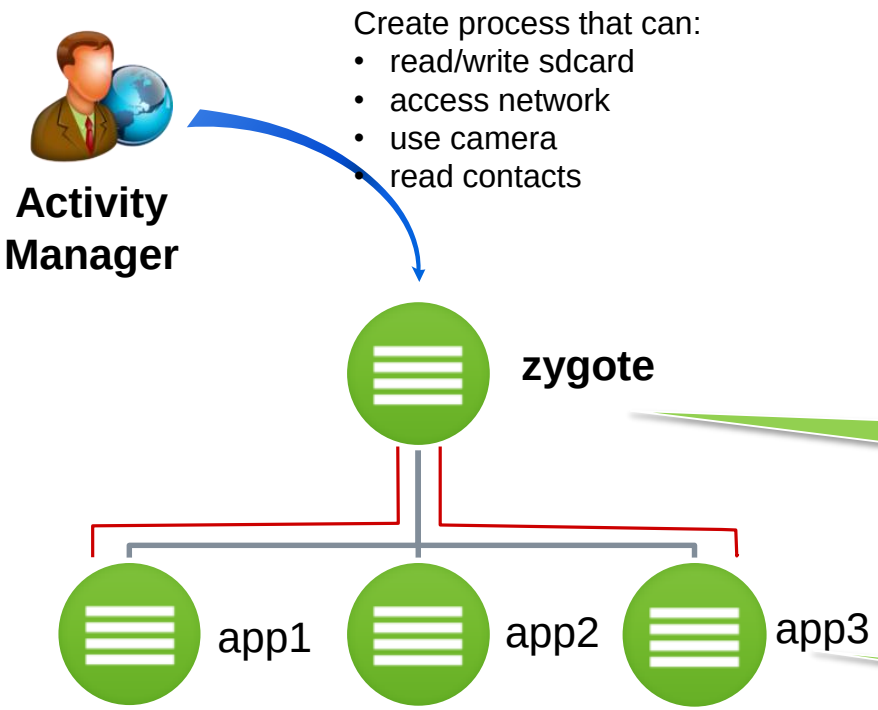
Basic Idea-Middleware



- The system_server
  - centralized controller for middleware permissions
- The client-server architecture
  - system services, content providers, etc.
- Binder IPC
  - RPC to services/content providers
  - Intent
  - Broadcast
  - Messengers
  - ashmem
  - ...

- **Dynamic Memory Instrumentation**

Basic Idea-Linux



- The zygote
  - centralized controller for Linux groups (a.k.a. Linux permissions)
- App works based on Linux system calls.

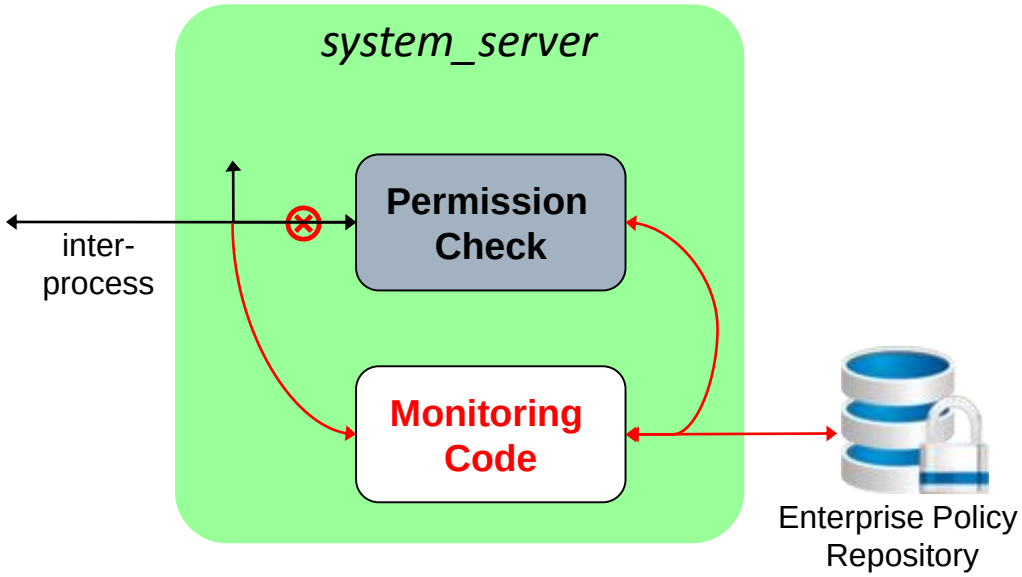
configure Linux permissions

configure Linux behaviors

- **Tracing System Calls**

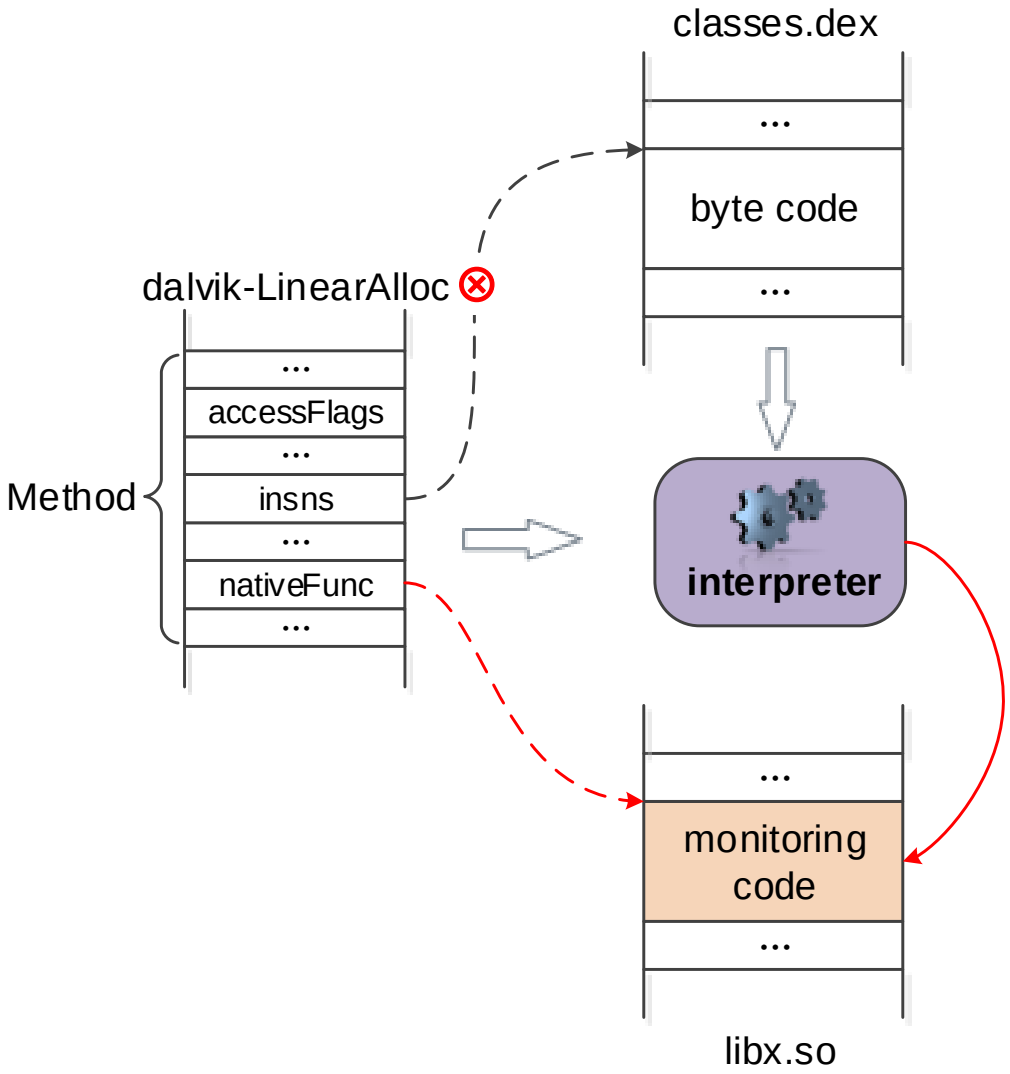
DeepDroid-Middleware Permission

- ***system_server*** opens a few interfaces for middleware permission check.



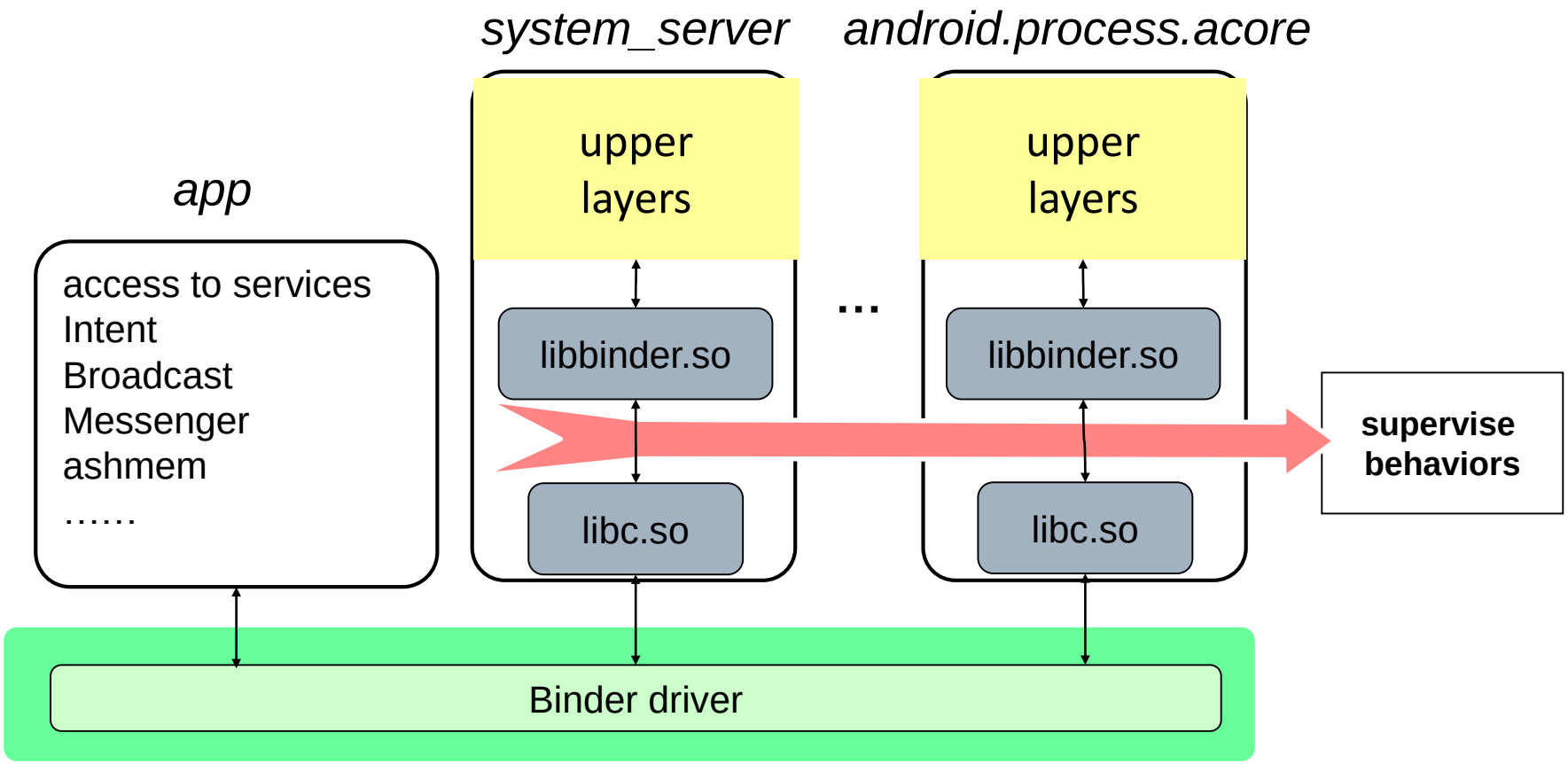
- Key: Java method interposition

DeepDroid-Middleware Permission



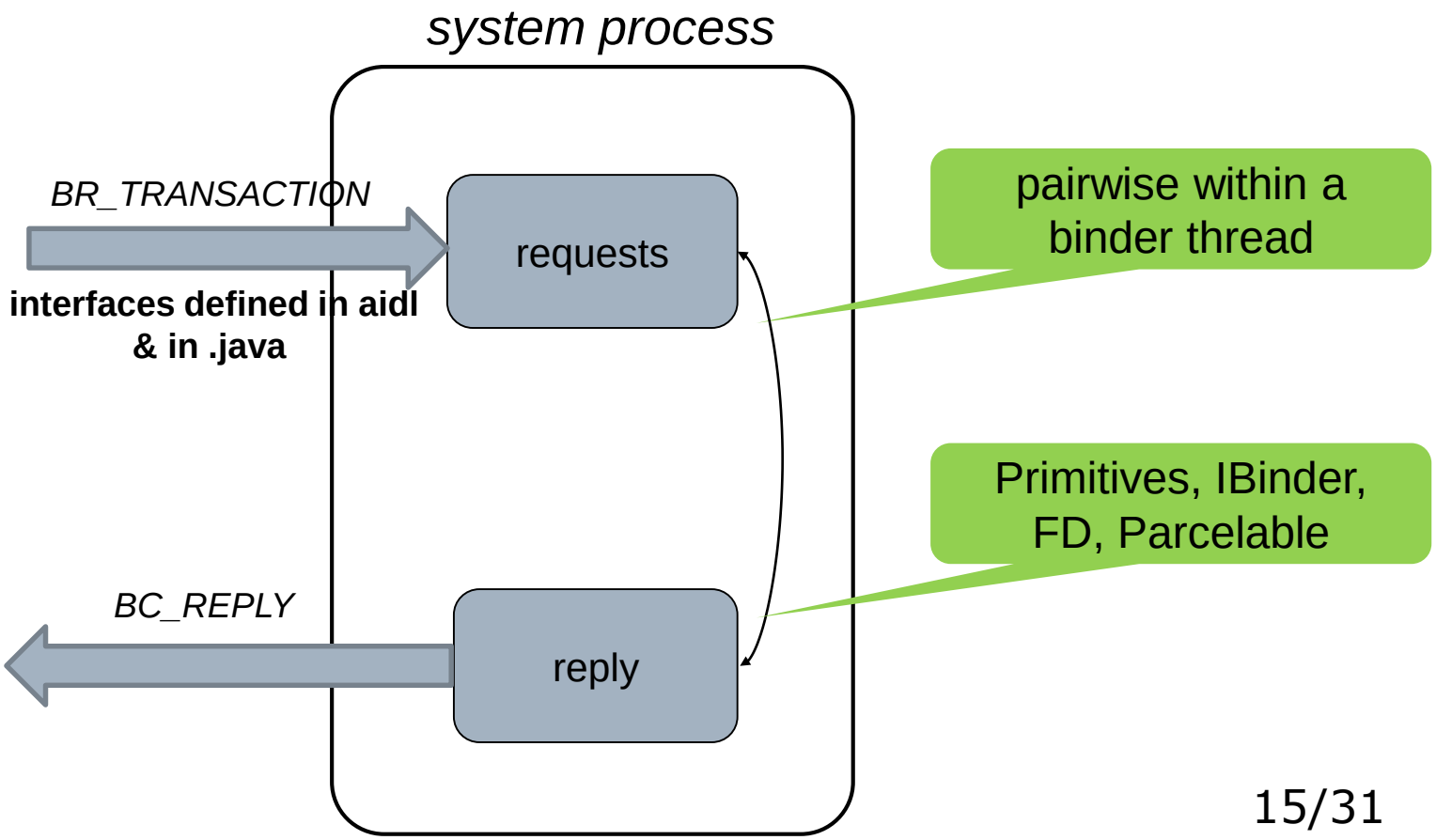
DeepDroid-Middleware Behavior

- Transactions between apps and system services
  - ioctl(binderFd, BINDER_WRITE_READ, &bwr)
  - By tampering Global Offset Table (GOT) of libbinder.so



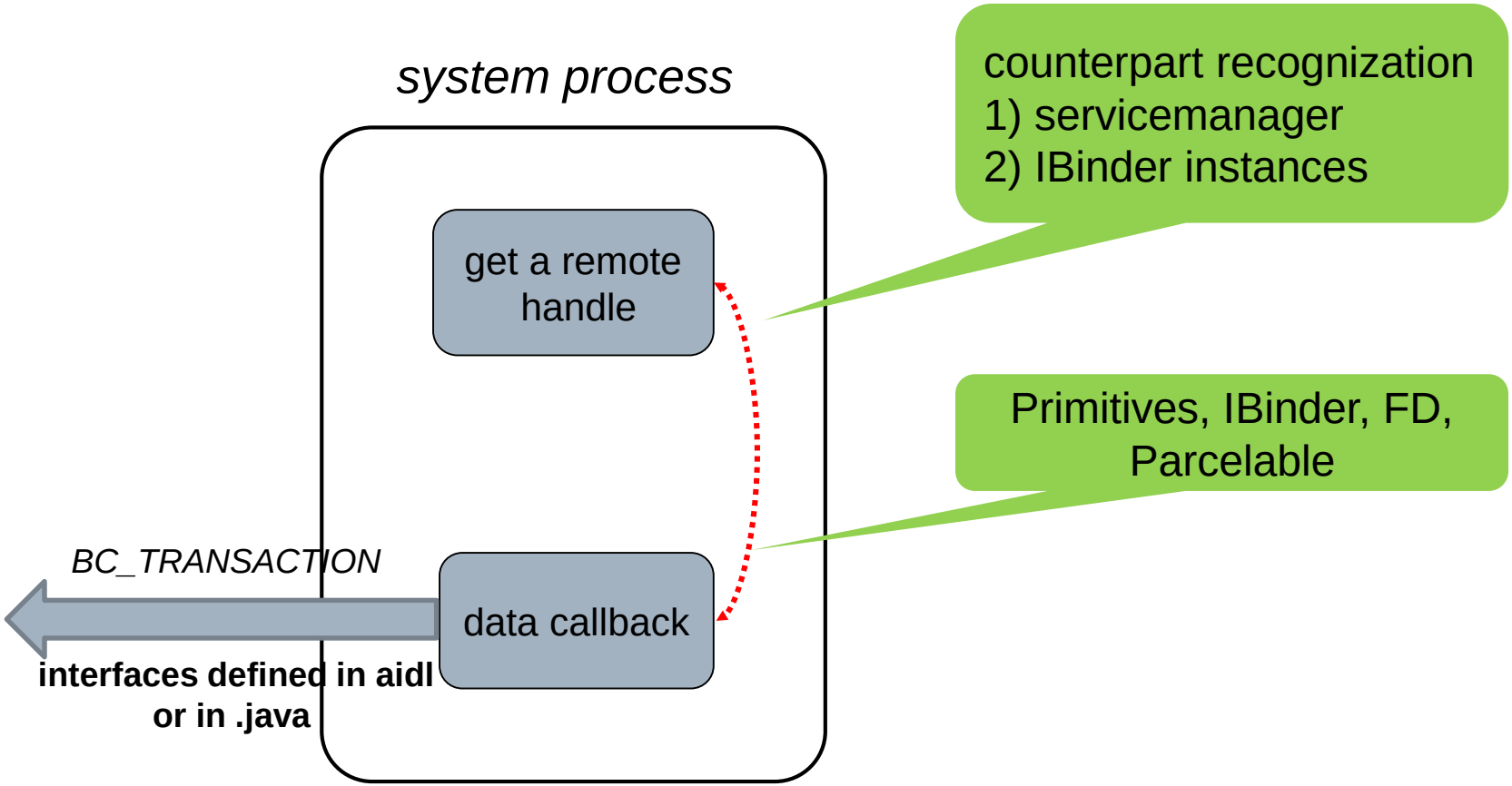
DeepDroid-Middleware Behavior

- Synchronous invocation
  - E.g., getLastKnownLocation(), getDeviceId()



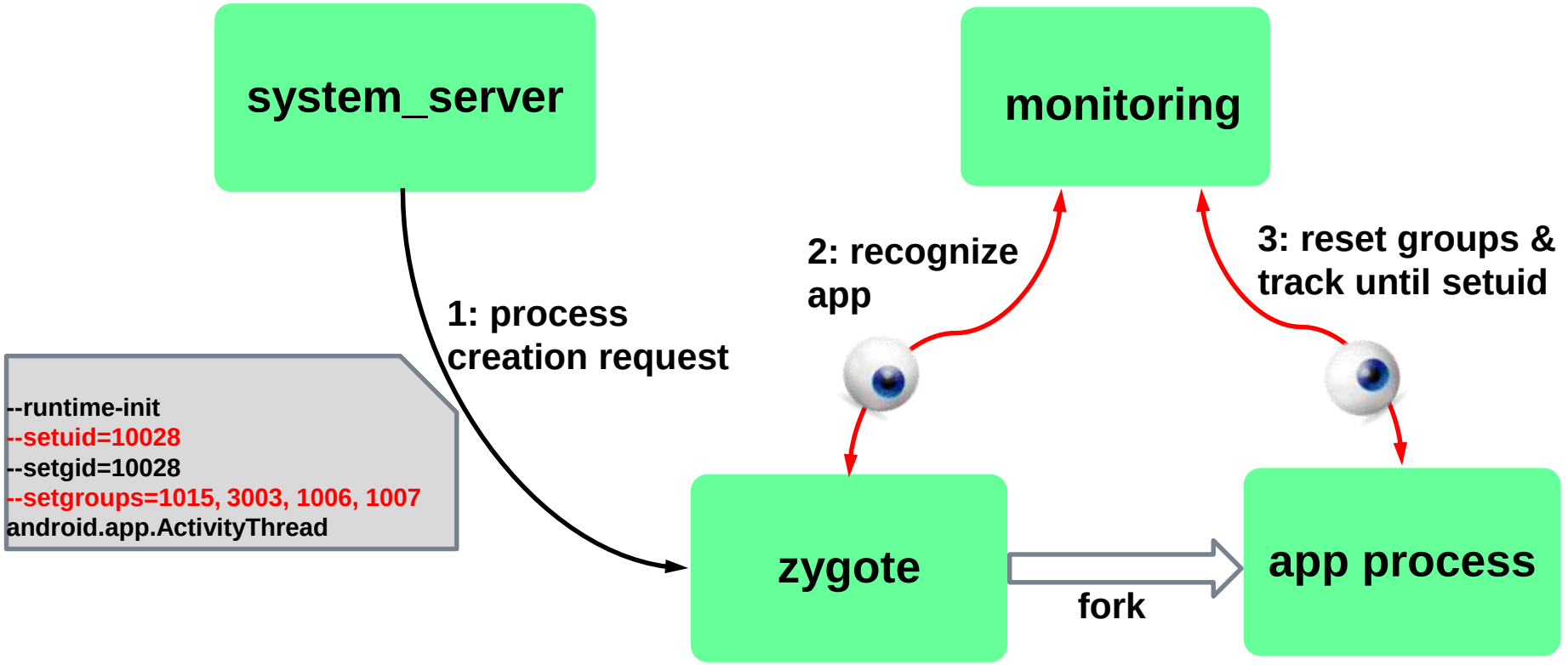
DeepDroid-Middleware Behavior

- Asynchronous invocation
  - One-way callbacks, e.g., onLocationChanged()



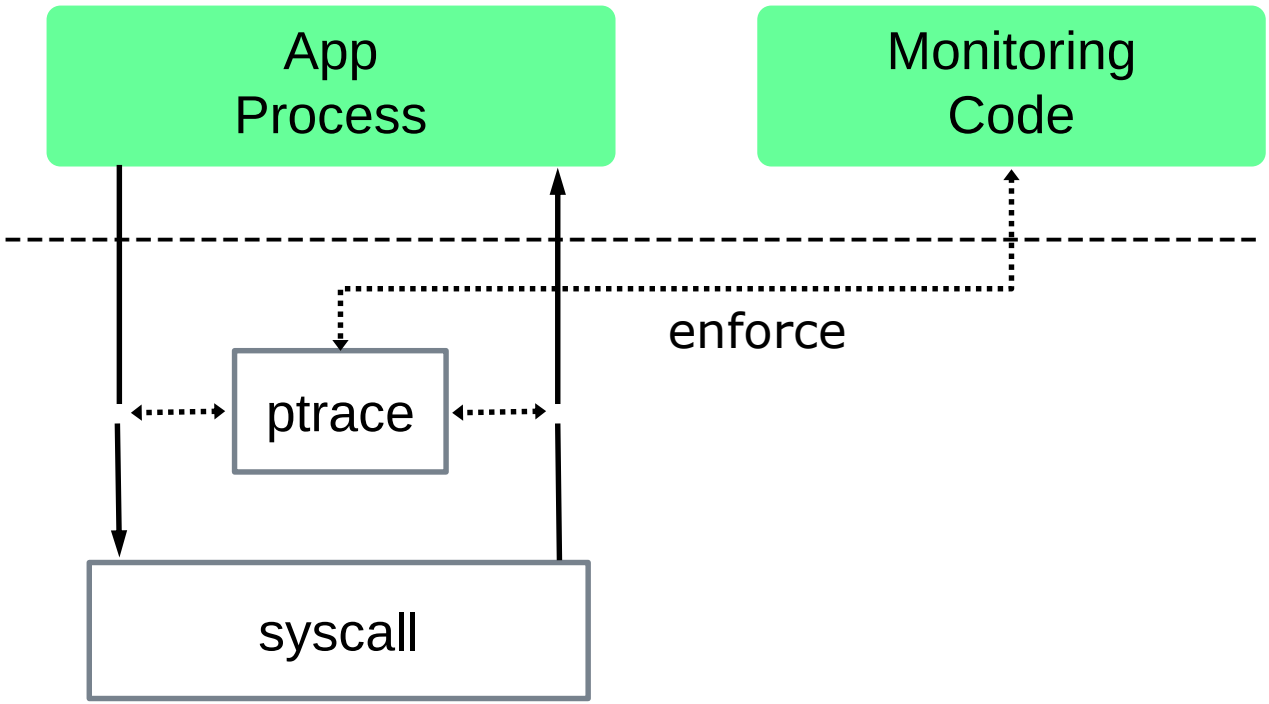
DeepDroid-Linux Permission

- Configure Linux permissions (e.g., groups)



DeepDroid-Linux Behavior

- Configuration on Linux permissions is irreversible.
  - Tracking system calls of Application



DeepDroid-Properties

- Fine-grained access control
  - Both permission and behavior level
- Portable
  - Based on stable system architecture, e.g., system services, permission mechanism, binder.
- Dynamic instrumentation
  - Reduce the work on system customization

Outline

- Introduction
- Related Work
- DeepDroid
- **Evaluation**
- Discussion
- Conclusion

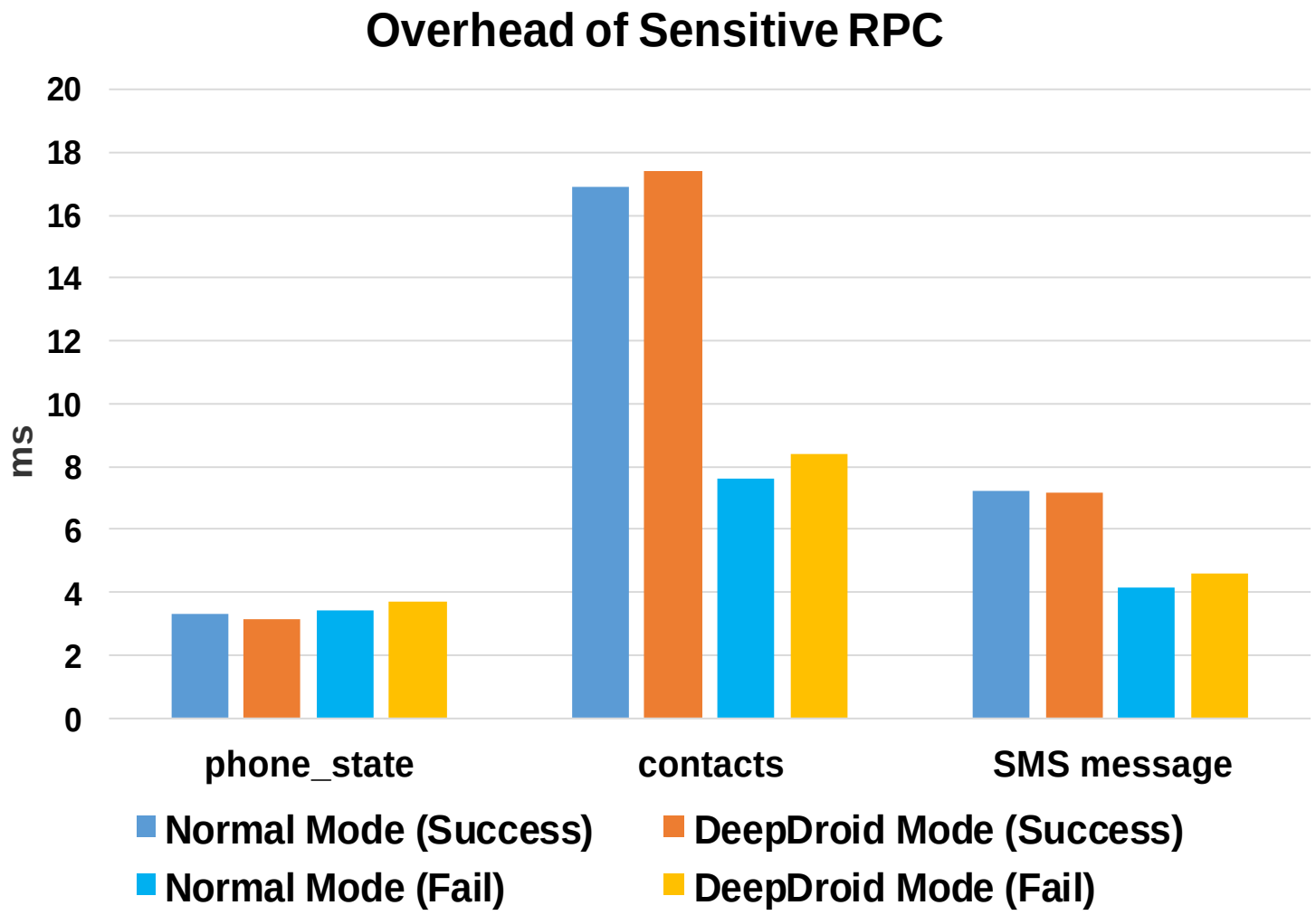
Evaluated Resources

Resource	Permission	Group	Permission Enforcement	Behavior Enforcement
IMEI	READ_PHONE_STATE		*package*	com.android.phone
Phone #	READ_PHONE_STATE		*package*	
location	ACCESS_FINE_LOCATION		*package*	system_server
contacts	READ_CONTACTS		*package*	android.process.acore
camera	CAMERA	camera	*package/ Process Creation*	mediaserver
account	GET_ACCOUNTS		*package*	system_server
logs	READ_LOGS	log	*Process Creation*	app process
network	INTERNET	inet	*package/ Process Creation*	
SMS	SEND_SMS		*package*	com.android.phone

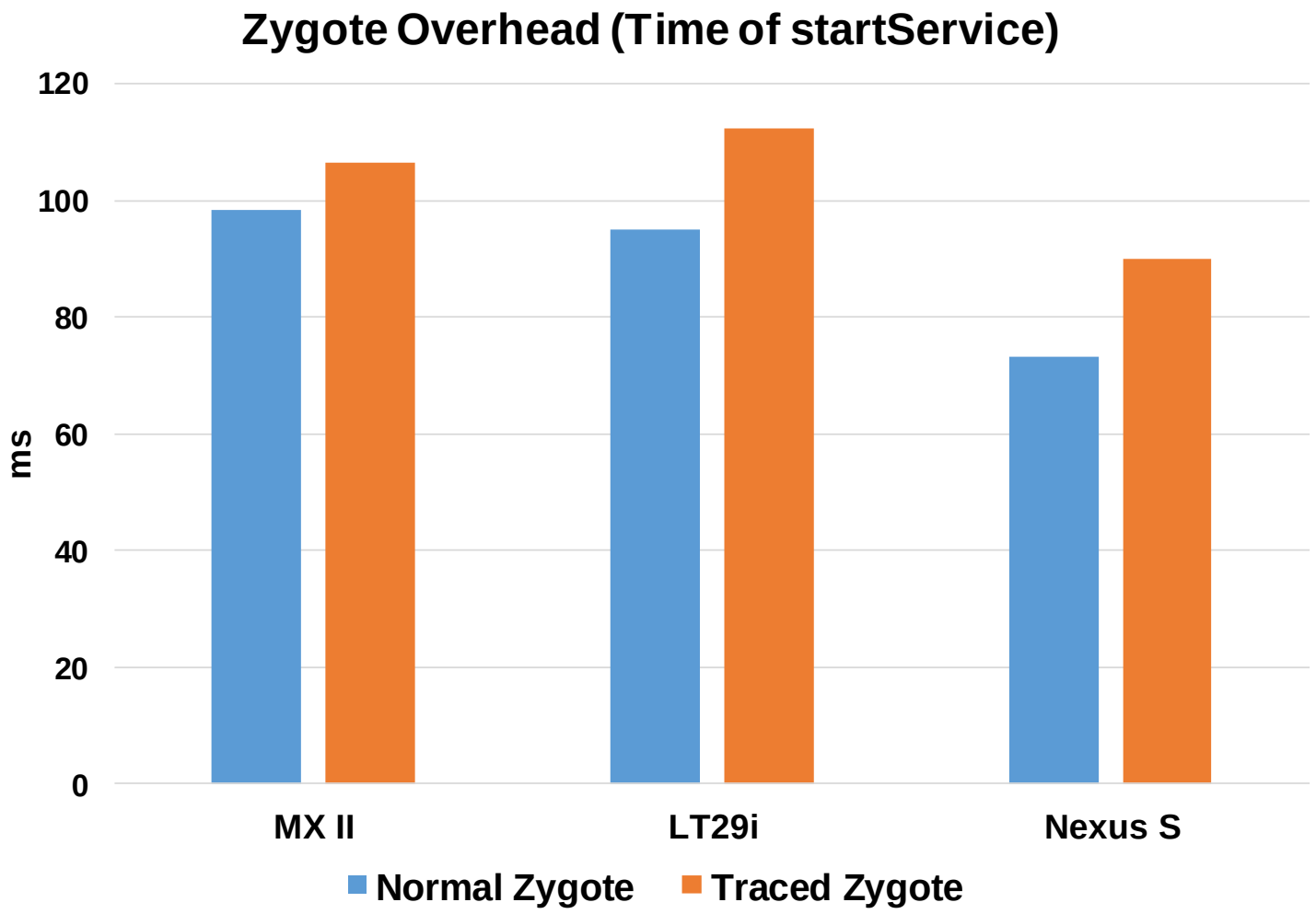
Evaluated Devices

Device	Android OS
Nexus S(Samsung)	2.3.6
Sony LT29i	4.1.2 4.2.2
Galaxy Nexus(Samsung)	4.0
Samsung Galaxy Note II	4.1
Samsung Galaxy Note 3	4.3
Nexus 5(LG)	4.4
Meizu MX II	Flyme 3.2 (4.2.1)
Huawei Honor 3c	4.2

Performance



Performance



Performance

Quadrant Scores

	Normal Quadrant	Traced Quadrant
MX II	2508.5	2507.6
LT29i	4653.8	4553.6
Nexus S	1750.0	1705.6

CaffeineMark Scores

	Normal CaffeineMark	Traced CaffeineMark
MX II	6367.2	6207.5
LT 29i	14125.5	13998.5
Nexus S	5982.8	5959.9

Outline

- Introduction
- Related Work
- DeepDroid
- Evaluation
- **Discussion**
- Conclusion

Discussion

- Root access
  - Required to instrument system components and trace *zygote.*
  - DeepDroid is a *self-contained* app and can be easily inserted as a system component.
  - DeepDroid carries little burden on vendor customization.
- Compared to other solutions
  - SEAndroid is enforced on Android 4.4.
  - Knox is fully supported only on some Samsung devices.
  - DeepDroid is based on stable architecture of Android, therefore, it can be easily adopted on *phones from other OEMs* and *legacy phones*.

Discussion

- policy misuse
  - We used software-based scheme to protect policies.
  - On future devices, we can adopt some hardware-based schemes (e.g., TrustZone-based integrity checking scheme).

Outline

- Introduction
- Related Work
- DeepDroid
- Evaluation
- Discussion
- **Conclusion**

Conclusion

- We propose a dynamic security policy enforcement scheme named DeepDroid.
- DeepDroid enables fine-grained control on both permission and apps' behavior.
- DeepDroid is relatively portable on different devices compared to direct system customization.

Thank You

References

- **Compac**[CODASPY'14]: "Compac: enforce component-level access control in android"
- **FlaskDroid**[USENIX Security'13]: "Flexible and Fine-Grained Mandatory Access Control on Android for Diverse Security and Privacy Policies"
- ***SEAndroid***[NDSS'13]: "Security Enhanced (SE) Android: Bringing Flexible MAC to Android"
- **TaintDroid**[OSDI'10]: "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones"
- **AppFence**[CCS'11]: "These aren't the droids you're looking for: retrofitting android to protect data from imperious applications"
- **I-ARM-Droid**[MoST'12]: "I-ARM-Droid: A Rewriting Framework for In-App Reference Monitors for Android Applications"
- **Aurasium**[USENIX Security'12]: "Aurasium: Practical Policy Enforcement for Android Applications"
- **AppGuard**[TACAS'13]: "AppGuard: Enforcing User Requirements on Android Apps"