19th Annual Network & Distributed System Security Symposium

Towards Taming Privilege Escalation Attacks on Android

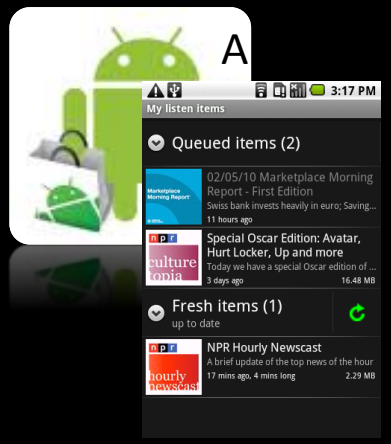
Alexandra Dmitrienko
Fraunhofer Institute for Secure Information Technology, Darmstadt, Germany

Sven Bugiel, Lucas Davi
TU Darmstadt/CASED, Germany

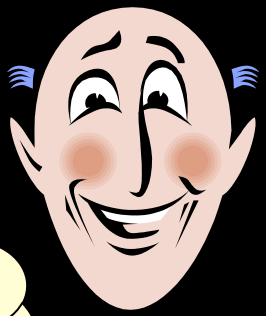
Ahmad-Reza Sadeghi, Bhargava Shastry
Fraunhofer SIT/CASED, Darmstadt, Germany

Thomas Fischer
Ruhr-University Bochum

App Installation in Android



Download App



Permissions

Install

Requested permissions are reasonable

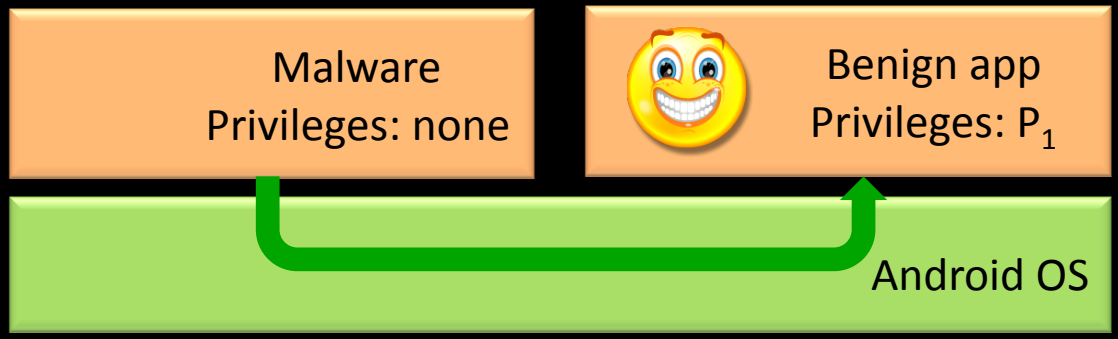
Can apps go beyond their privileges?

YES

Privilege escalation attacks

Confused Deputy Attack

Do not have a right permission? Ask your neighbor!



Examples:

1) Invoke browser to download malicious files (Lineberry et al., BlackHat 2010)
2) Invoke Phone app to perform a phone call (Enck et al., TechReport 2008)
3) Invoke Android Scripting Environment to send SMS messages (Davi et al., ISC'2010)

Collusion Attack

Two (or more) apps collude to launch the attack

1) Apps communicate directly

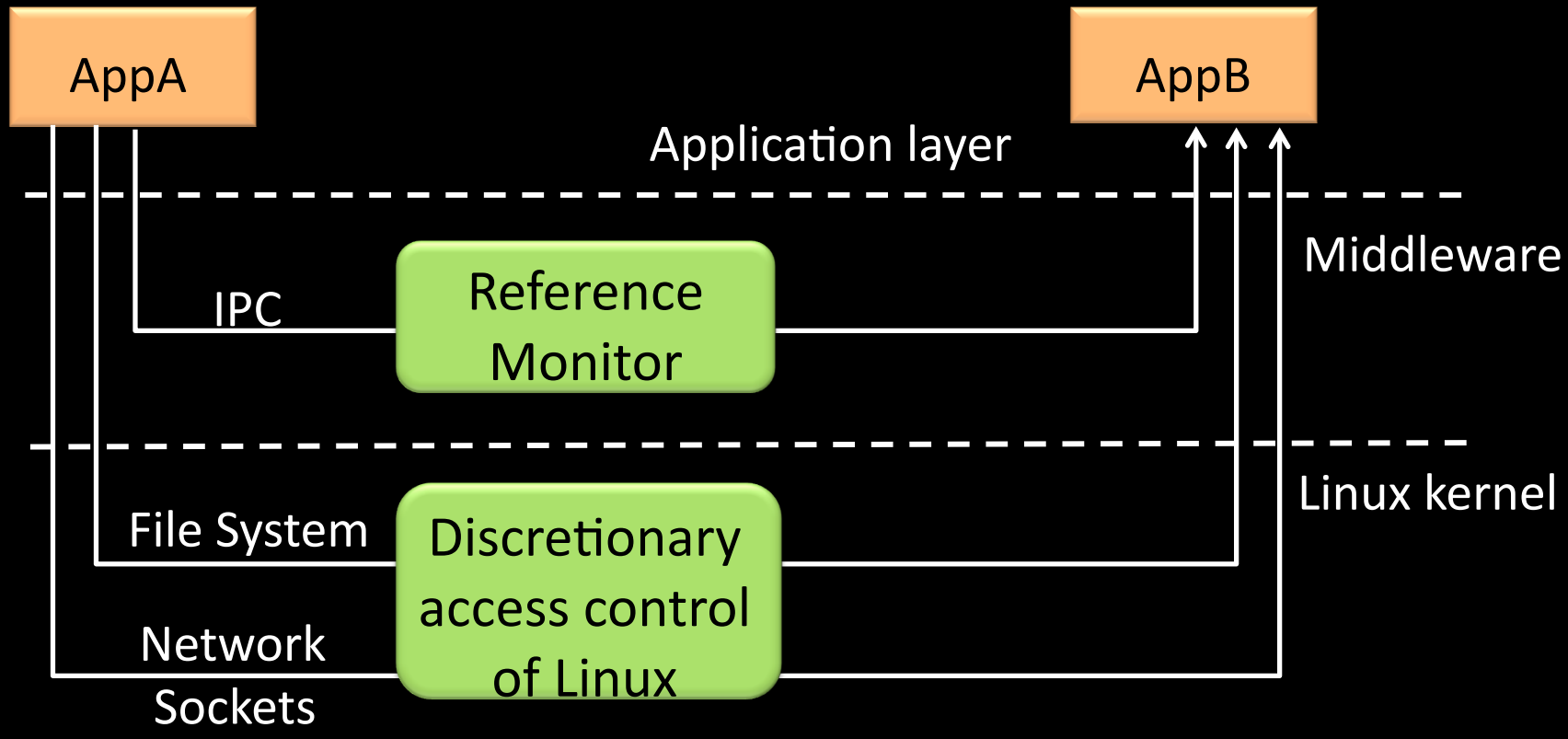
Example: Claudio Marforio et. al, TechReport ETH Zurich

2) Apps communicate via covert (e.g., volume settings) or overt (e.g., content providers) channels in Android System components

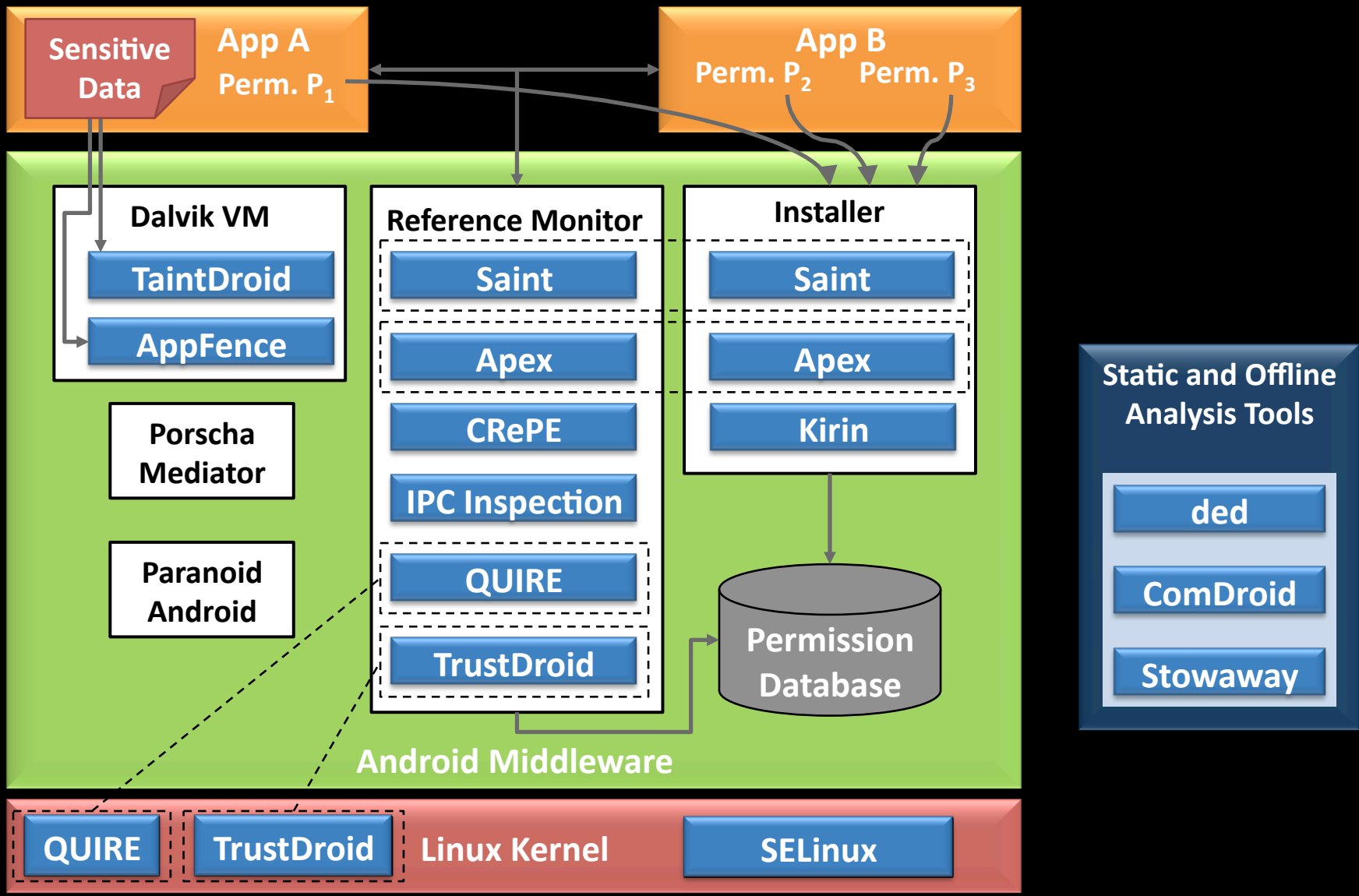
Example: Soundcomber (Schlegel et al., NDSS'2011)

Inter-Application Communication

- Inter-process communication (IPC)
  - Intents and remote procedure calls
- File system (files, Unix domain sockets)
- Network sockets

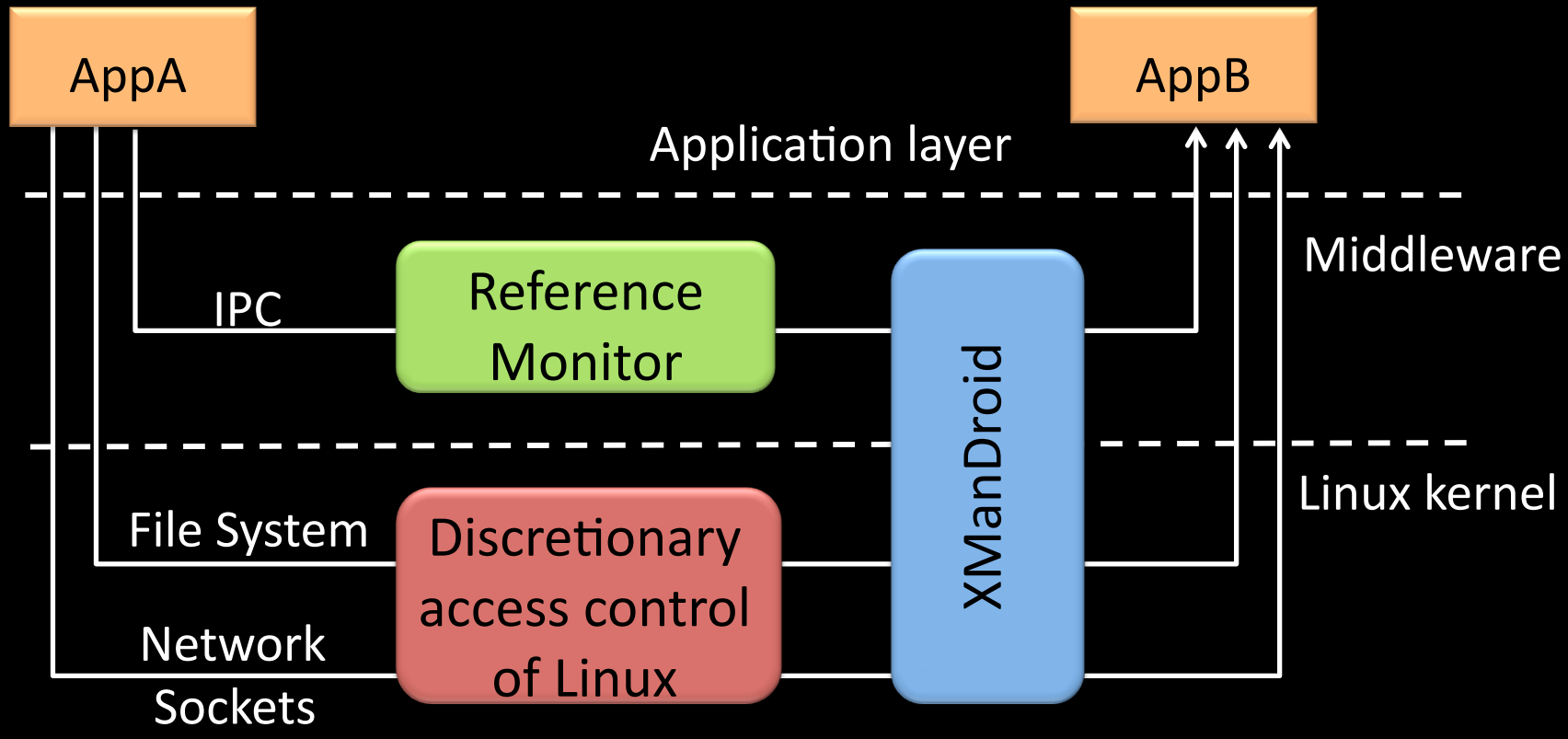


Related Work

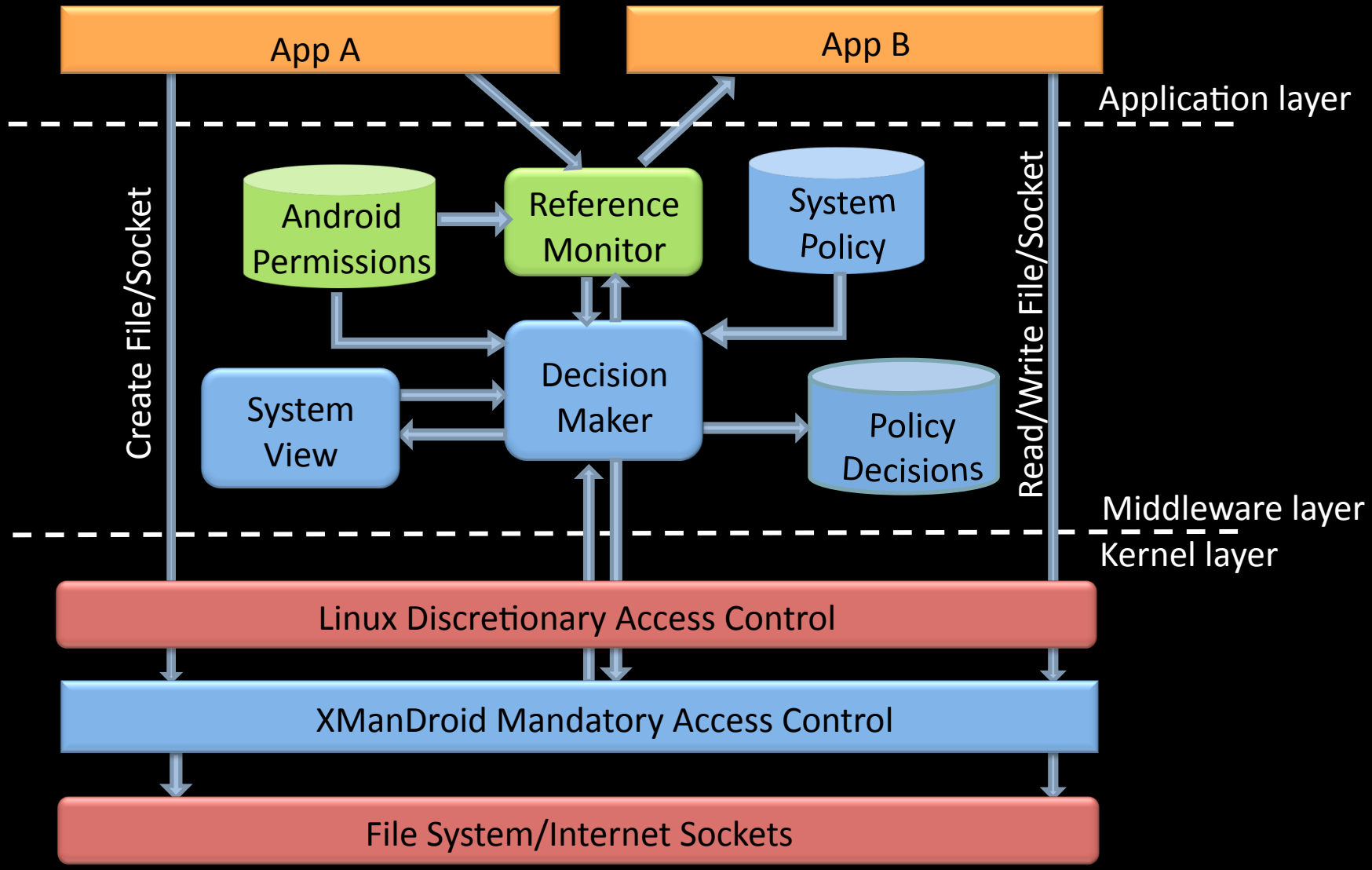


XManDroid: eXtended Monitoring on Android

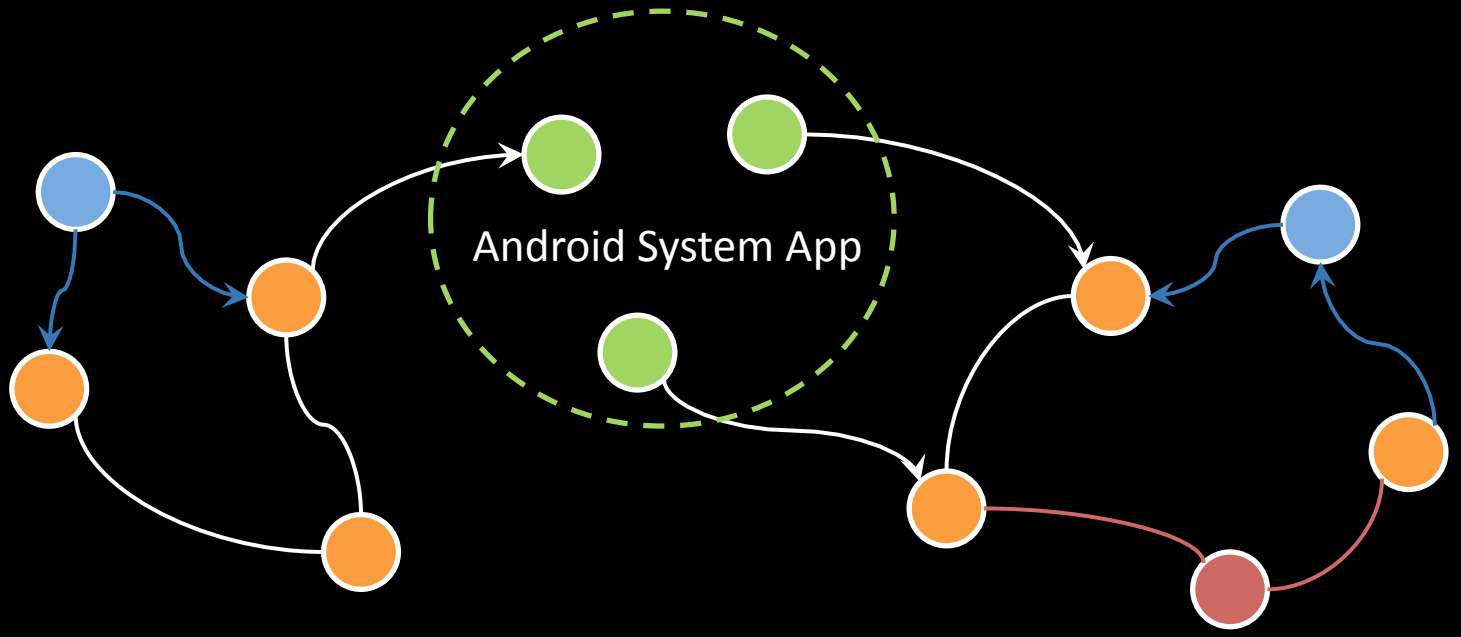
- Monitors all communication channels between apps
- Validates if the requested communication link complies to a system-centric security policy



XManDroid Architecture



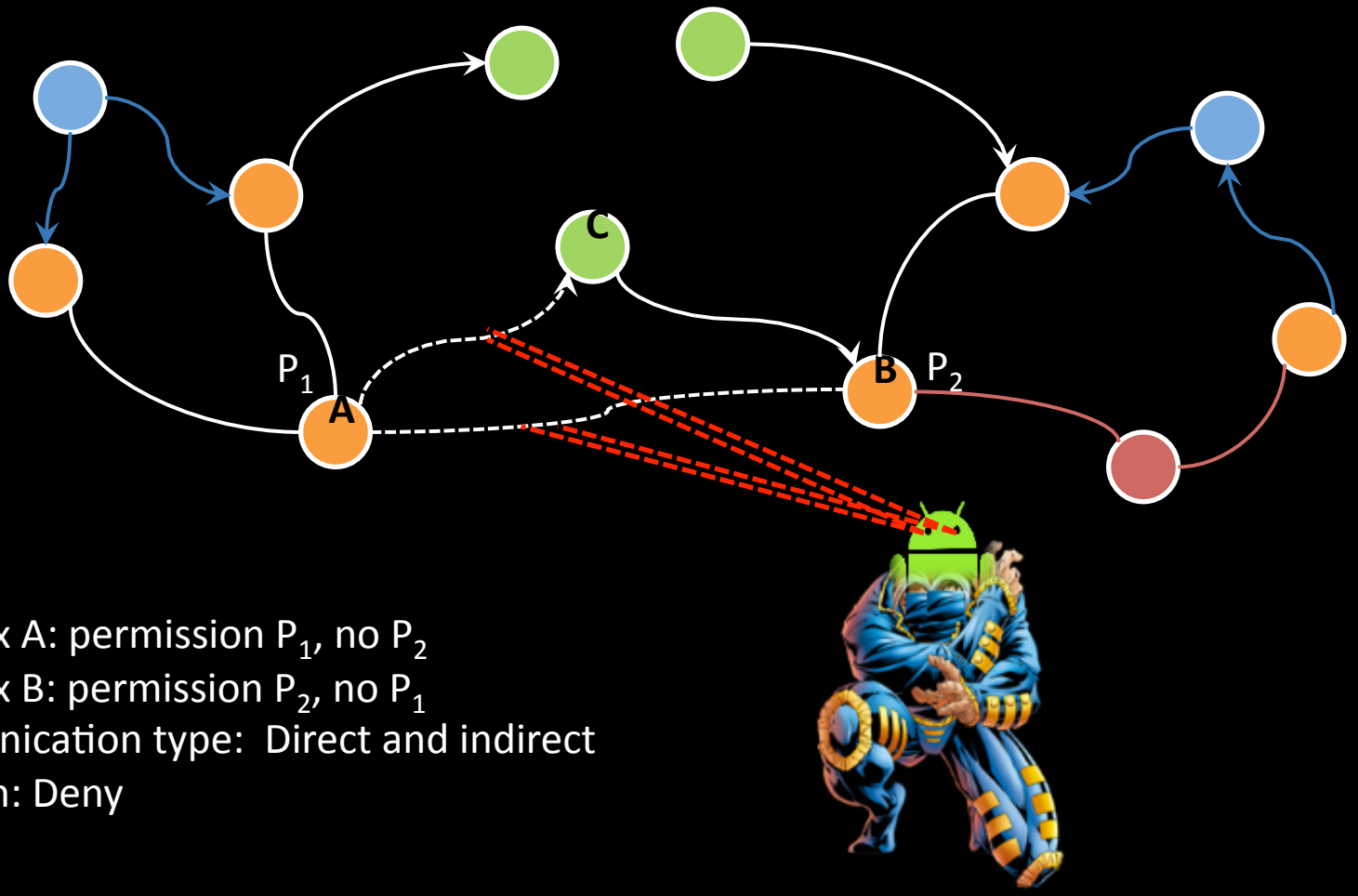
XManDroid's SystemView: Graph-based Representation



- System Components
- Application sandboxes
- Files
- Internet sockets

- IPC calls
- Access to files
- Socket connections

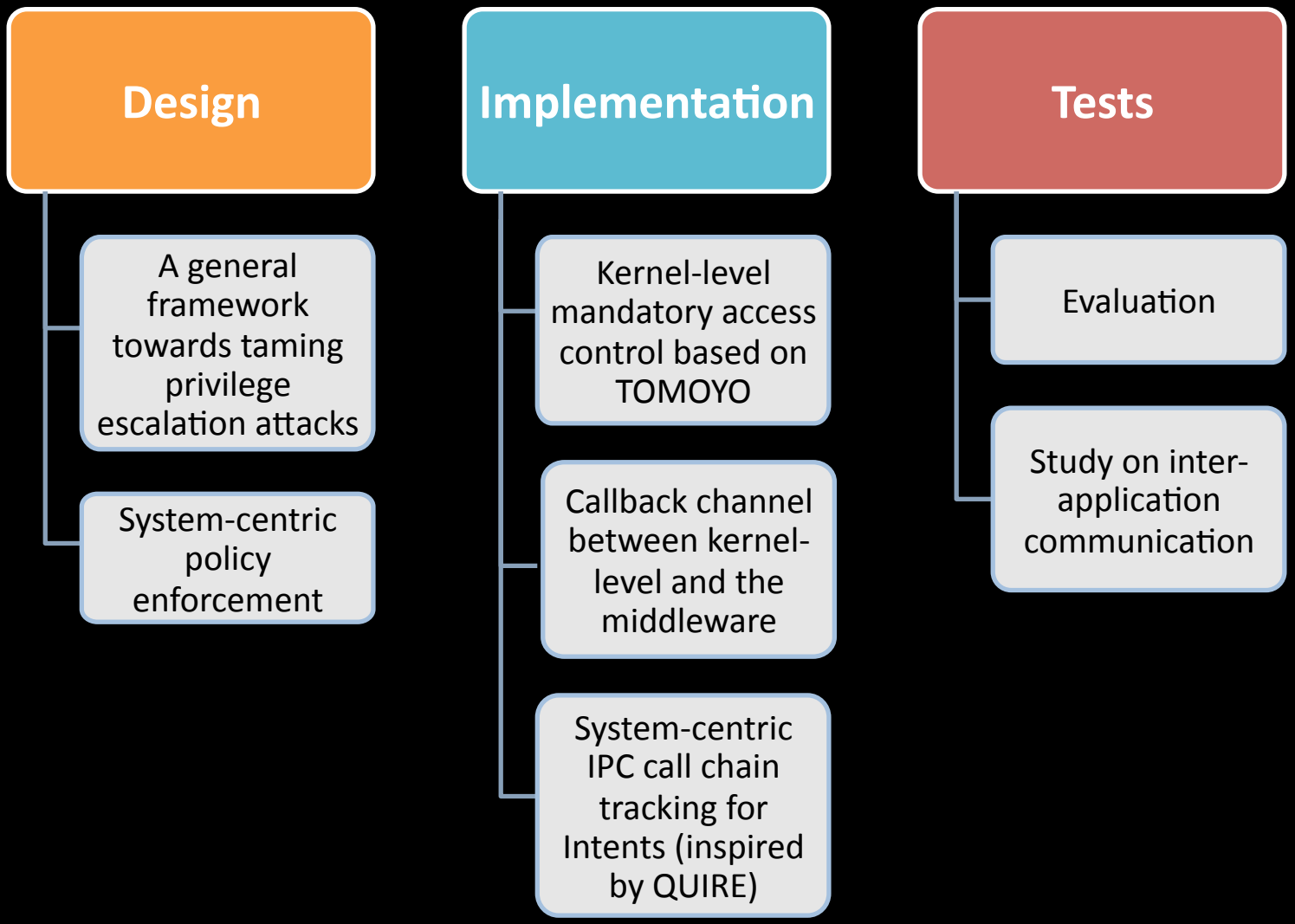
XManDroid: Simplified Example



Policy Rule:

- Sandbox A: permission P_1, no P_2
- Sandbox B: permission P_2, no P_1
- Communication type: Direct and indirect
- Decision: Deny

Contributions

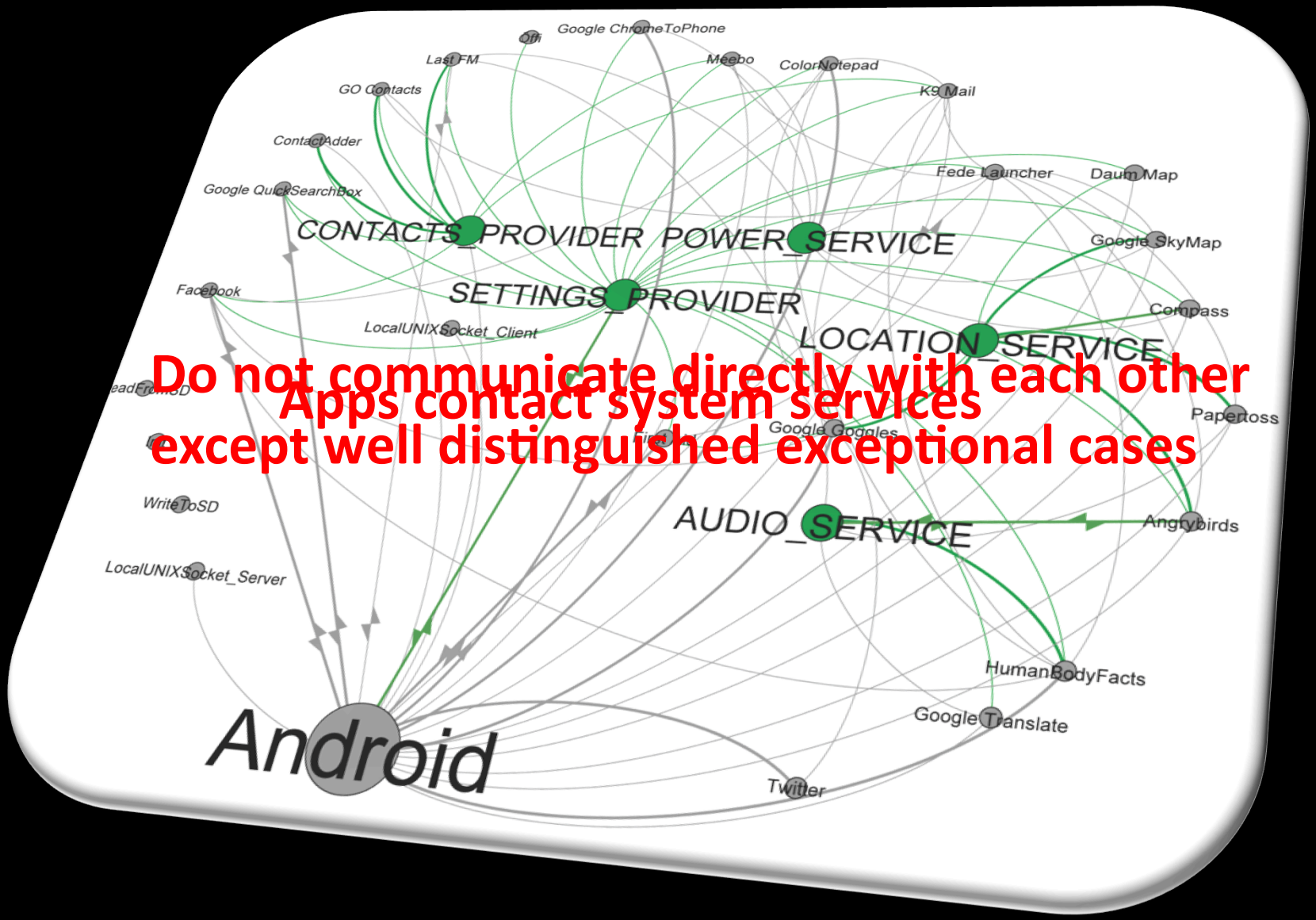


Evaluation

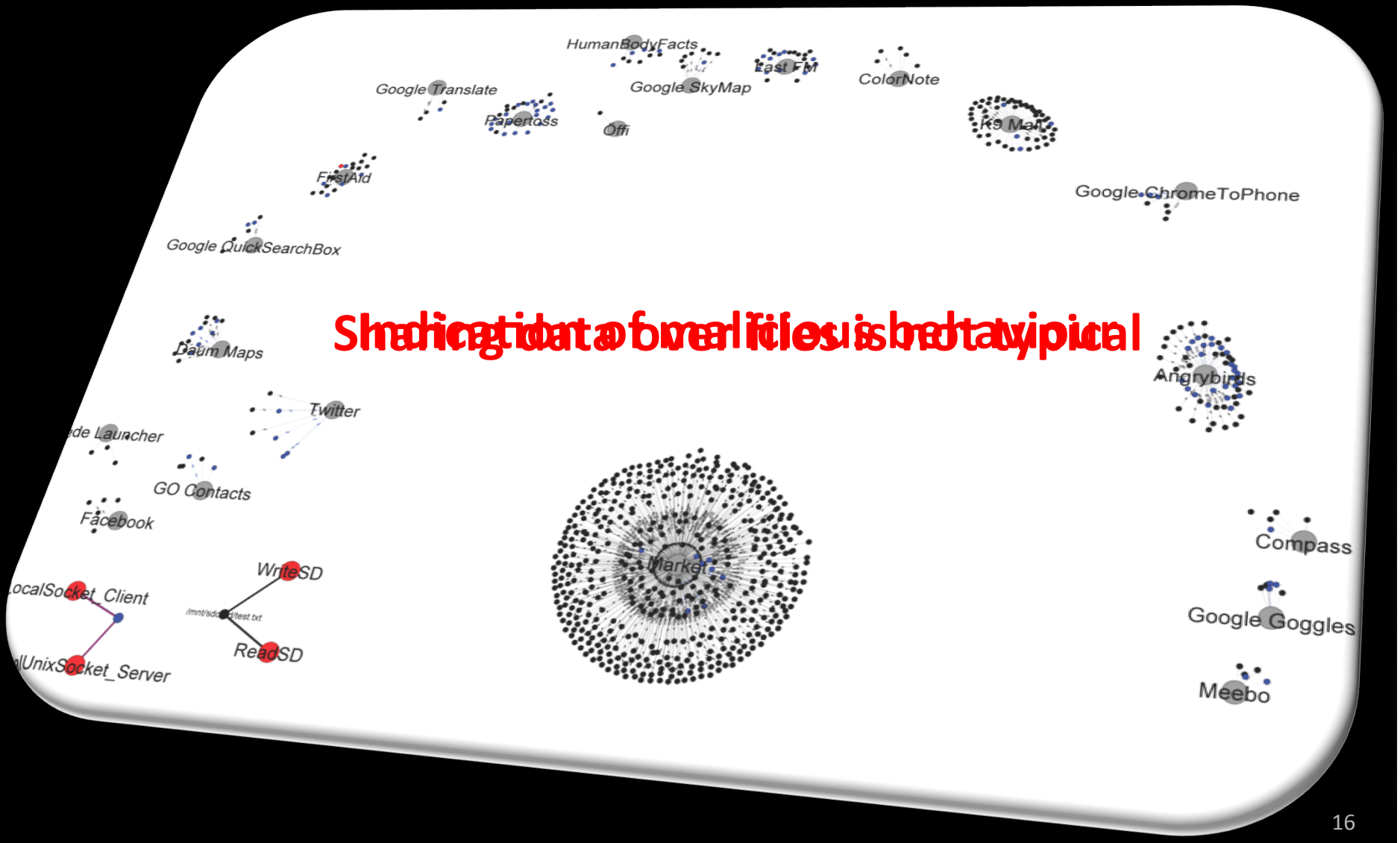
1. Effectiveness (attack prevention)
2. Performance
3. Rate of falsely denied communications

Study on Application Communication Patterns

IPC-based Application Communication



File and Socket-based Application Communication



Conclusion and Future Work

- First general approach towards tackling privilege escalation attacks (at application level)
- Runtime monitoring, but quite efficient
- No false negatives
- No false positives, but conceptually they are possible

- Current work
  - Large scale evaluation
  - Automatic policy engineering
  - Full IPC call chain tracking
  - Applying XManDroid framework for domain isolation on Android

BizzTrust

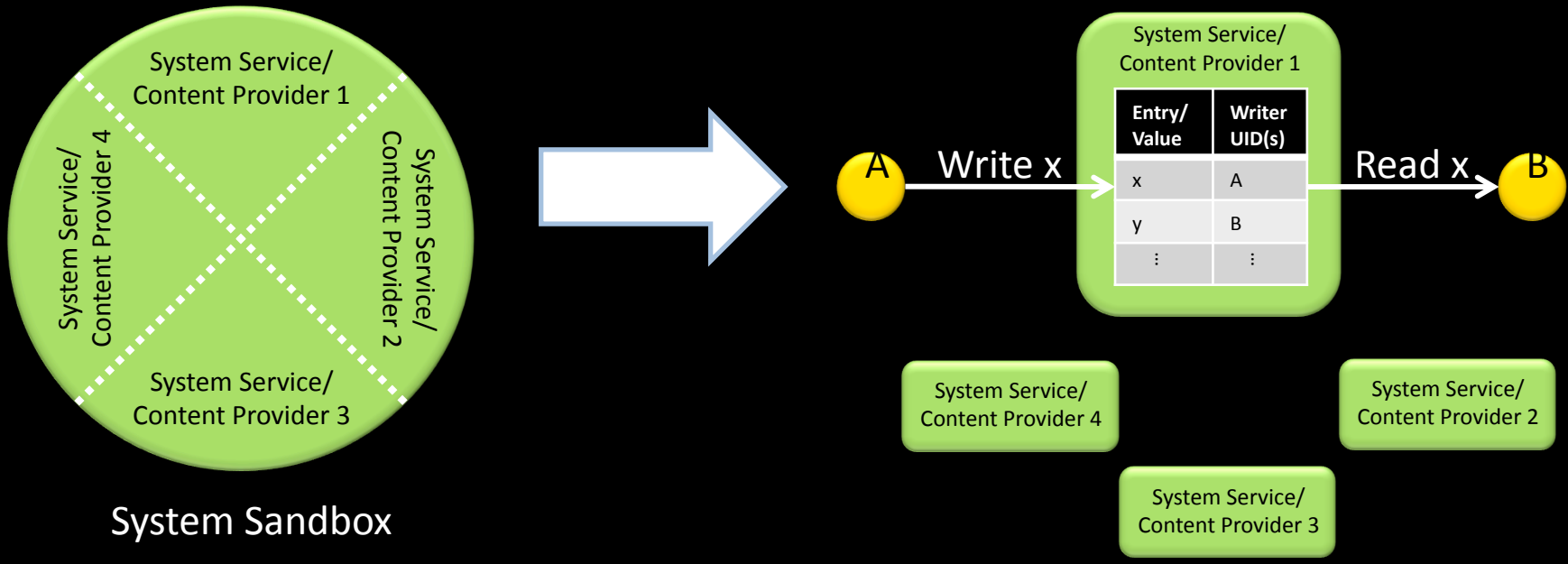
Current Work: BizzTrust

- Special case of XManDroid
- Allows dual use of phone for Private and Enterprise usage

Thank you

alexandra.dmitrienko@sit.fraunhofer.de

Fine-grained Analysis of System Sandbox



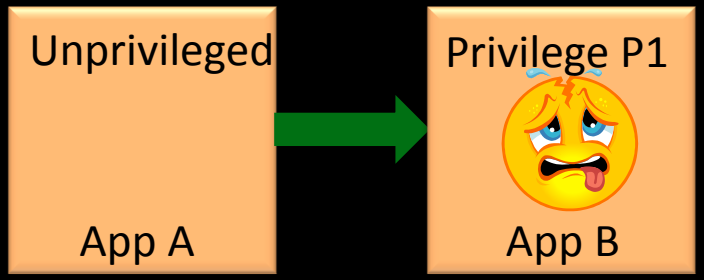
System Services:
Extend API to enforce permission check between the last writer of a value and the reader upon reading of a value

System Content Provider:
Extend API to filter all data from the response to a read access whose reader-writer(s) pair violates the system policy

Privilege Escalation Attacks

Scenario 1:

Confused deputy attack



Examples:

1) Invoke browser to download malicious files (Lineberry et al., BlackHat 2010)
2) Unauthorized phone call (Enck et al., TechReport 2008)
3) Invoke Android Scripting Environment to send SMS messages (Davi et al., ISC'2010)

Scenario 2:

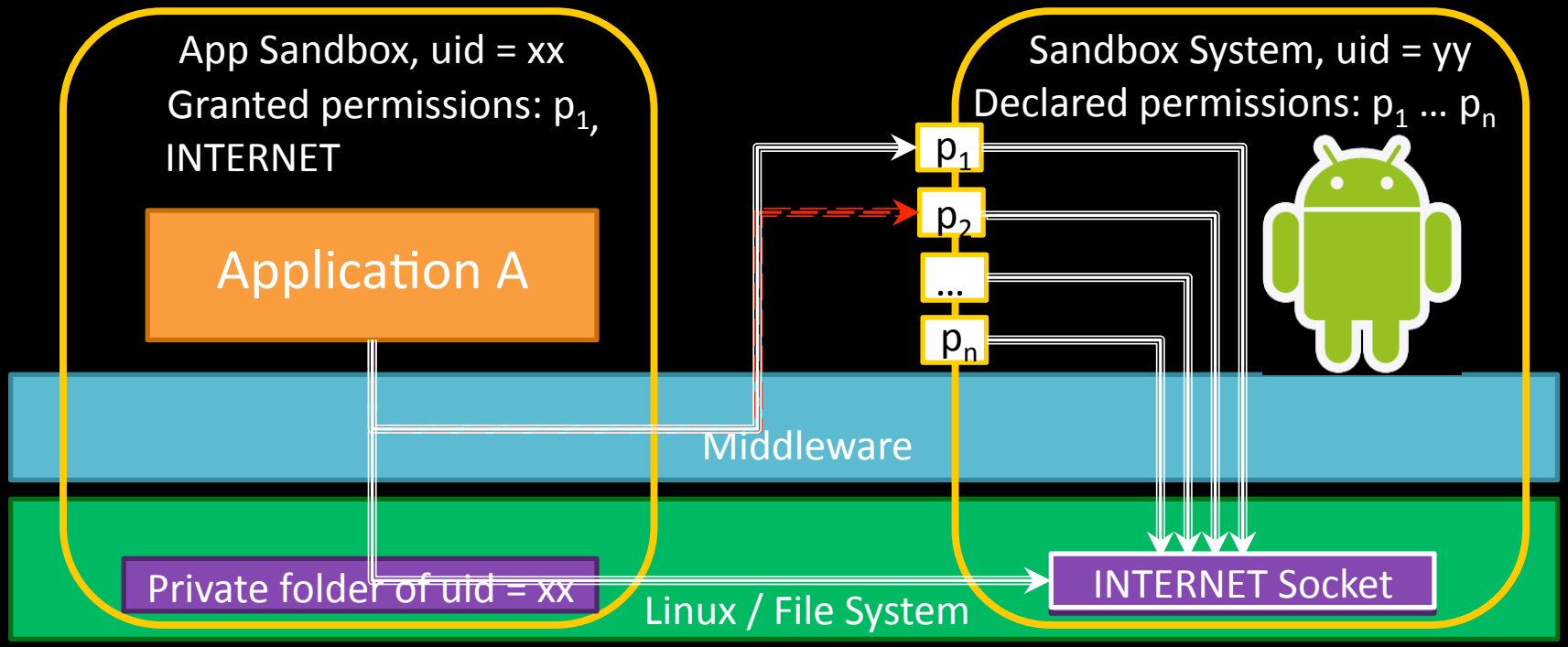
Collusion attack

1) Apps communicate directly (Claudio Marforio et. al, TechReport ETH Zurich)
2) Apps communicate via covert (e.g., volume settings) or overt (e.g., content providers) channels in Android System components

   Example: Soundcomber (Schlegel et al., NDSS'2011)

Permission Framework

- Applications must be granted corresponding permissions to be able to access protected interfaces
- Permission assignments are monitored by middleware reference monitor
- Some exceptional cases: INTERNET and EXTERNAL_STORAGE permissions
  - Enforced by Linux kernel rather than middleware
  - These permissions are mapped to Linux groups (e.g., each app granted Internet permission is a member of a group which is allowed to access Internet driver)



Policy Language

Based on VALID [Bleikertz et al., POLICY 2011] – a formal security assurance language for graph-based virtualized topologies

- Expresses properties of graph vertices and paths
- Defines high-level security goals in a form of attack states

Policy rule for our simplified example:

A,B – application sandboxes

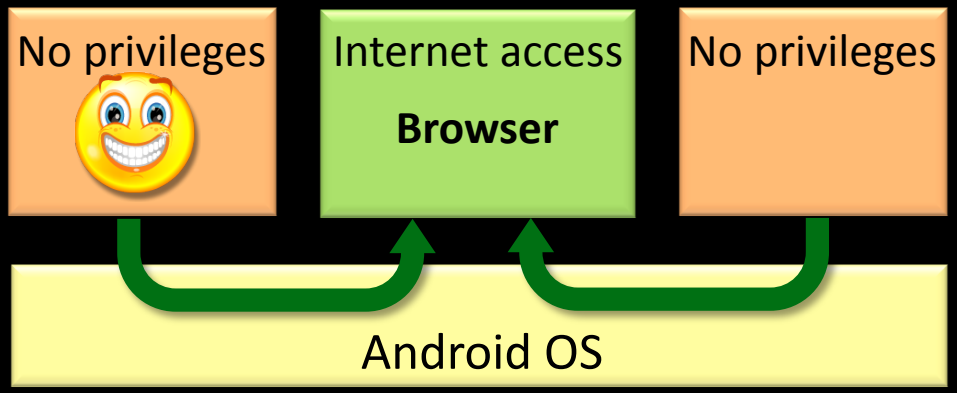
L(A,B) – path

goal RuleName(deny) := A.hasPermission(p1) ^ ¬(A.hasPermission(p2) ^ B.hasPermission(p2) ^ ¬(B.hasPermission(p1)) ^ L.connects(A,B) ^ L.type(direct I indirect)

Challenges: False Positives

Problem: The same communication channels can be used for both, legitimate purposes and attacks

App invokes the browser to open the link to advertisements



App invokes the browser to download malicious files

(Lineberry et al., BlackHat 2010)

Performed a study on third party application communication to discover possible sources of false positives

Tchnical Challenges to Mitigate Falselz Denied

1. Fine-grained analysis of communication links via system components
2. Directed edge representation upon file access and upon read/write access to system databases and system services
3. Establishment of the semantic links for IPC calls (inspired by QUIRE, (Dietz et al., USENIX Security 2011)
4. Deployment of exceptional policy rules to handle exceptional cases of application-to-application communication

System Sandbox: Fine-grained Analysis

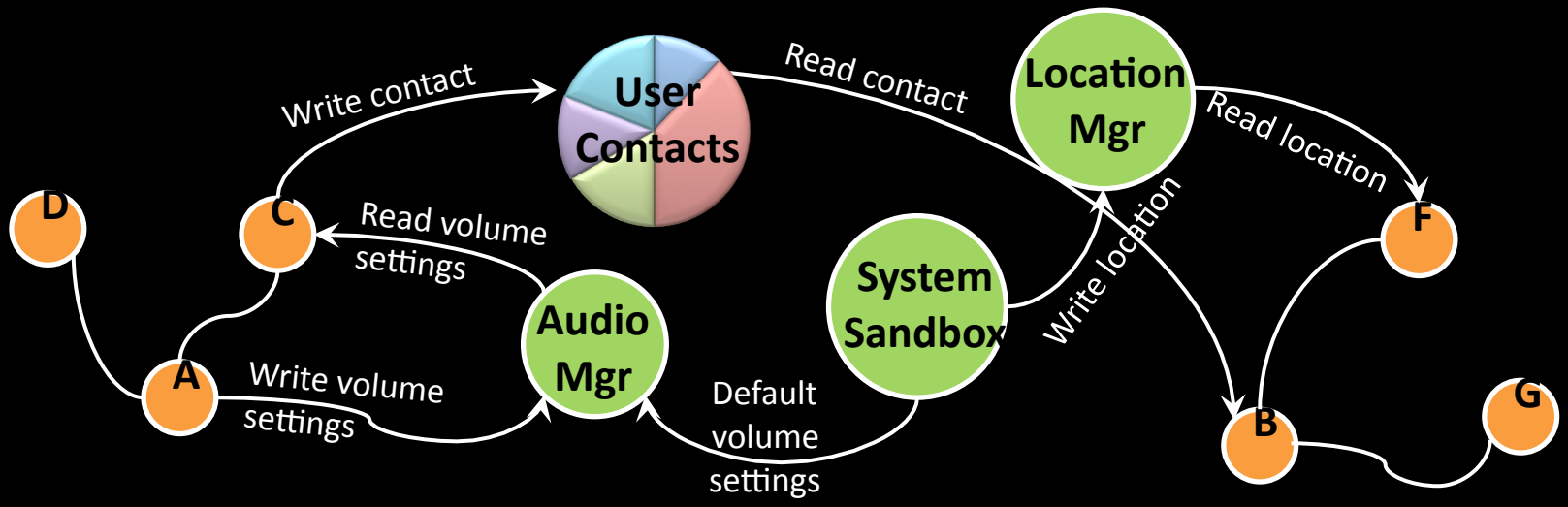
- Extract System Services (SS) and Content Providers (CP) from monolithic Android System Sandbox as virtual nodes
- Distinguishing read/write access to SSs and CPs (directed edges)

SSs (e.g., location manager or audio manager):

- Check privileges of the last writer of a value and the current reader

CPs (SQL-like databases, e.g., user contacts):

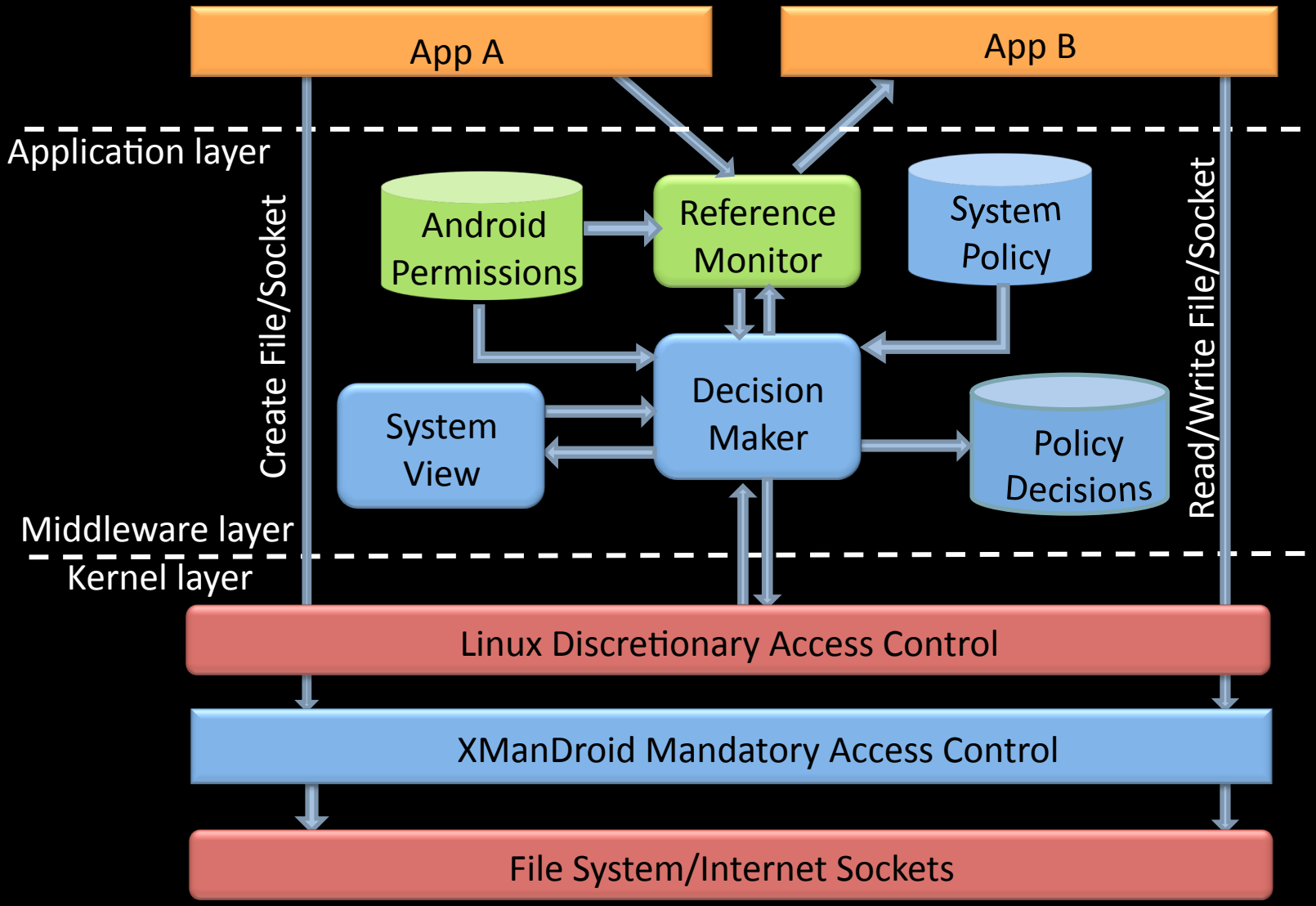
- Coloring data
- Filtering query response



Contribution

- A general framework to prevent both classes of privilege escalation attacks
  - Middleware: Runtime monitoring of IPC calls, establishment of the semantic links for IPC calls (inspired by QUIRE [Dietz et al., USENIX Security 2011])
  - Kernel level: Mandatory access control on file system and Internet sockets (based on TOMOYO)
  - Runtime mapping of middleware policies to kernel level
- Policy
  - System-centric policy enforcement
  - Expressed on adopted VALID policy language [Bleikertz et al., POLICY 2011]
- Evaluation
  - Performance, effectiveness, rate of falsely denied communications

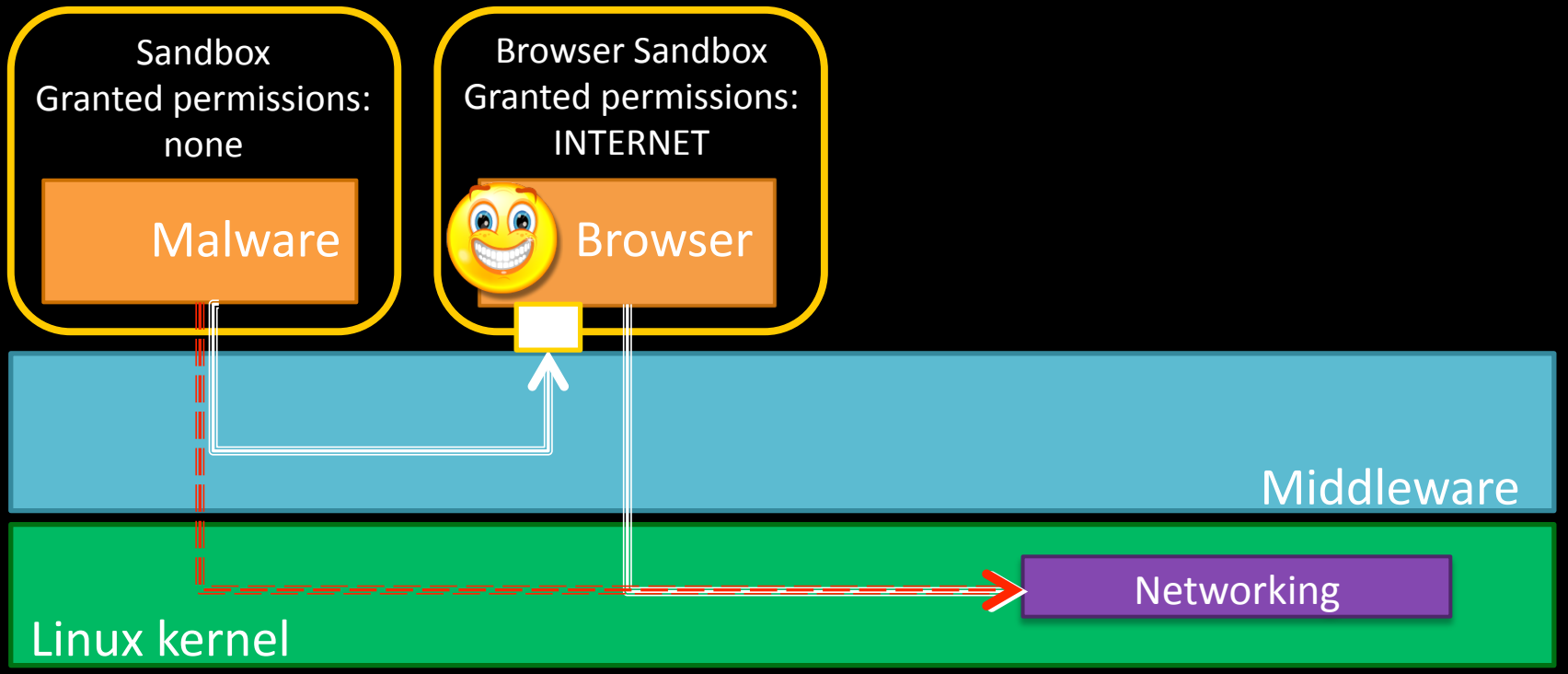
XManDroid Architecture



Confused Deputy Attack

Do not have a right permission? Ask your neighbor!

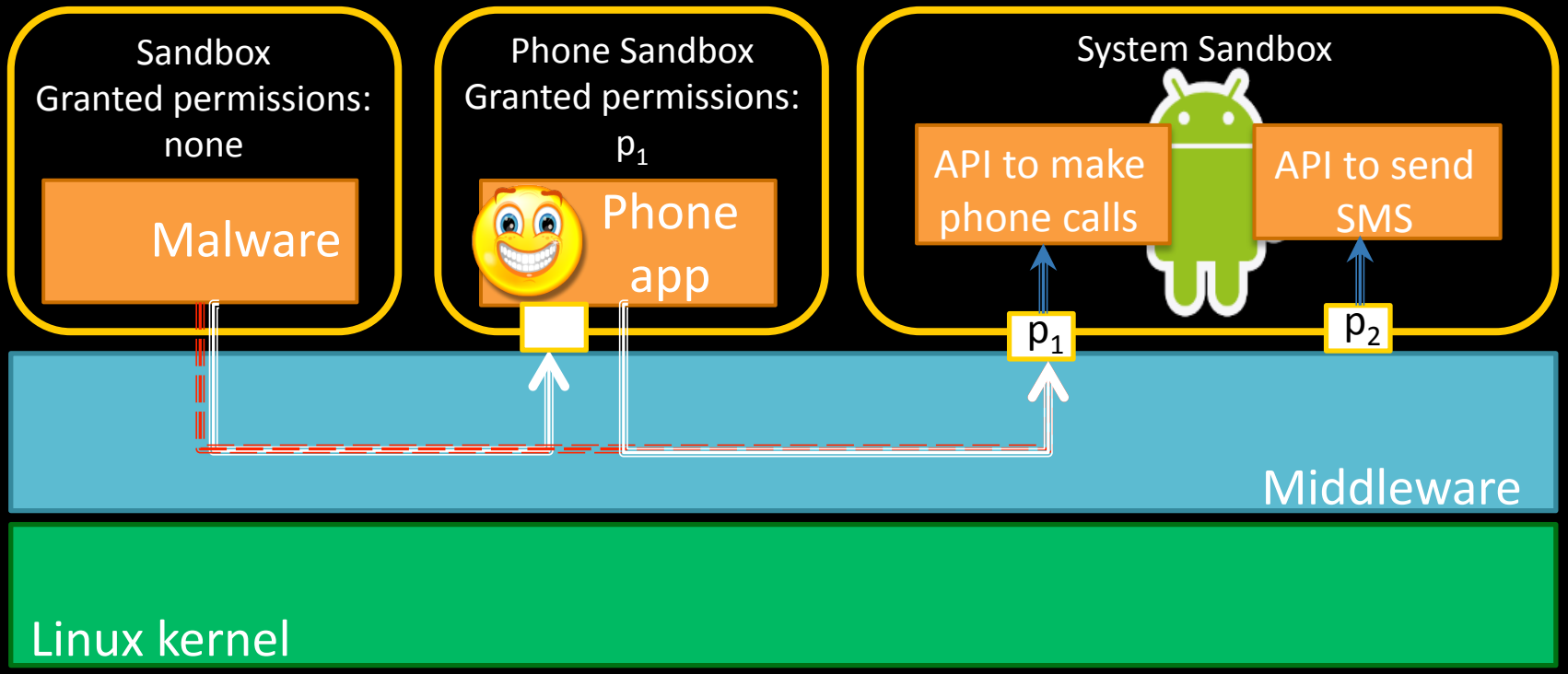
- Invoke browser to download malicious files (Lineberry et al., BlackHat 2010)



Confused Deputy Attack

Do not have a right permission? Ask your neighbor!

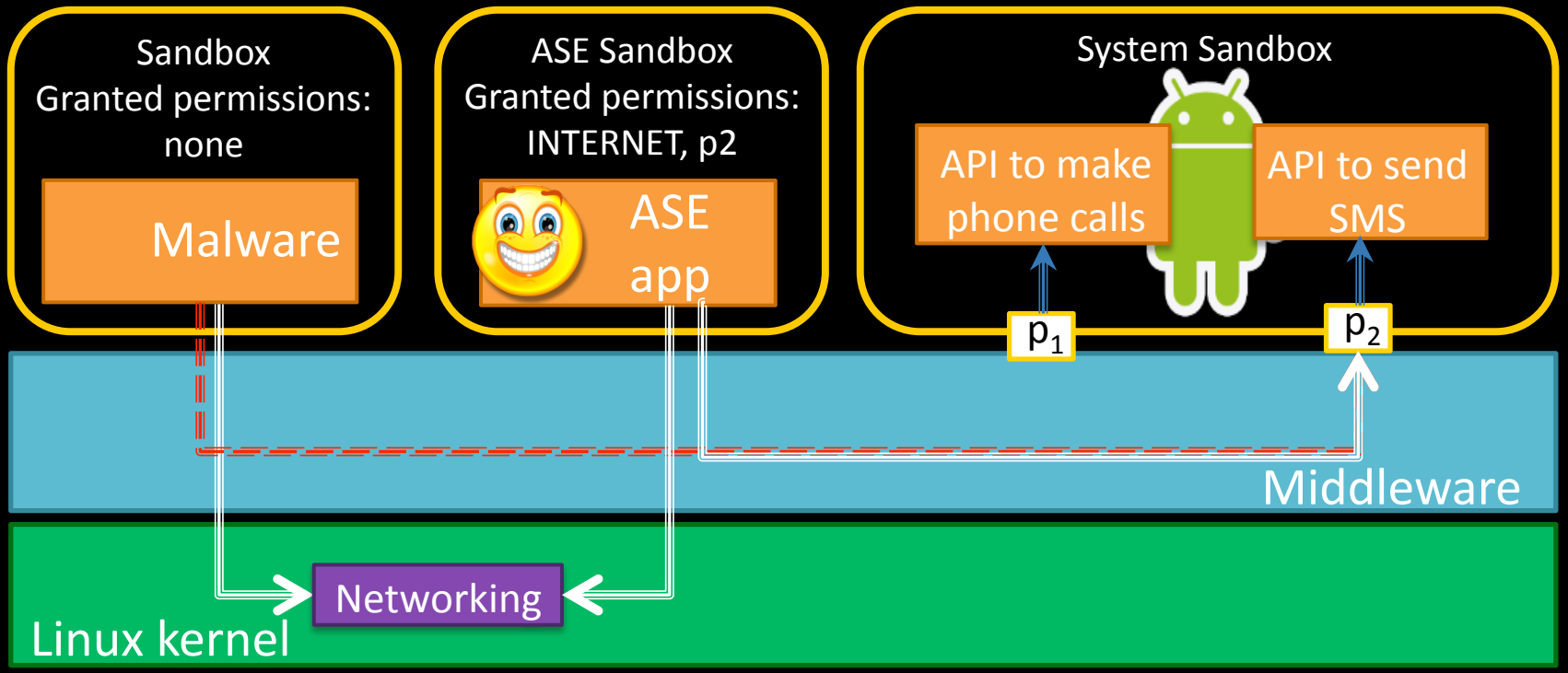
- Invoke browser to download malicious files (Lineberry et al., BlackHat 2010)
- Invoke Phone app to perform a phone call (Enck et al., TechReport 2008)



Confused Deputy Attack

Do not have a right permission? Ask your neighbor!

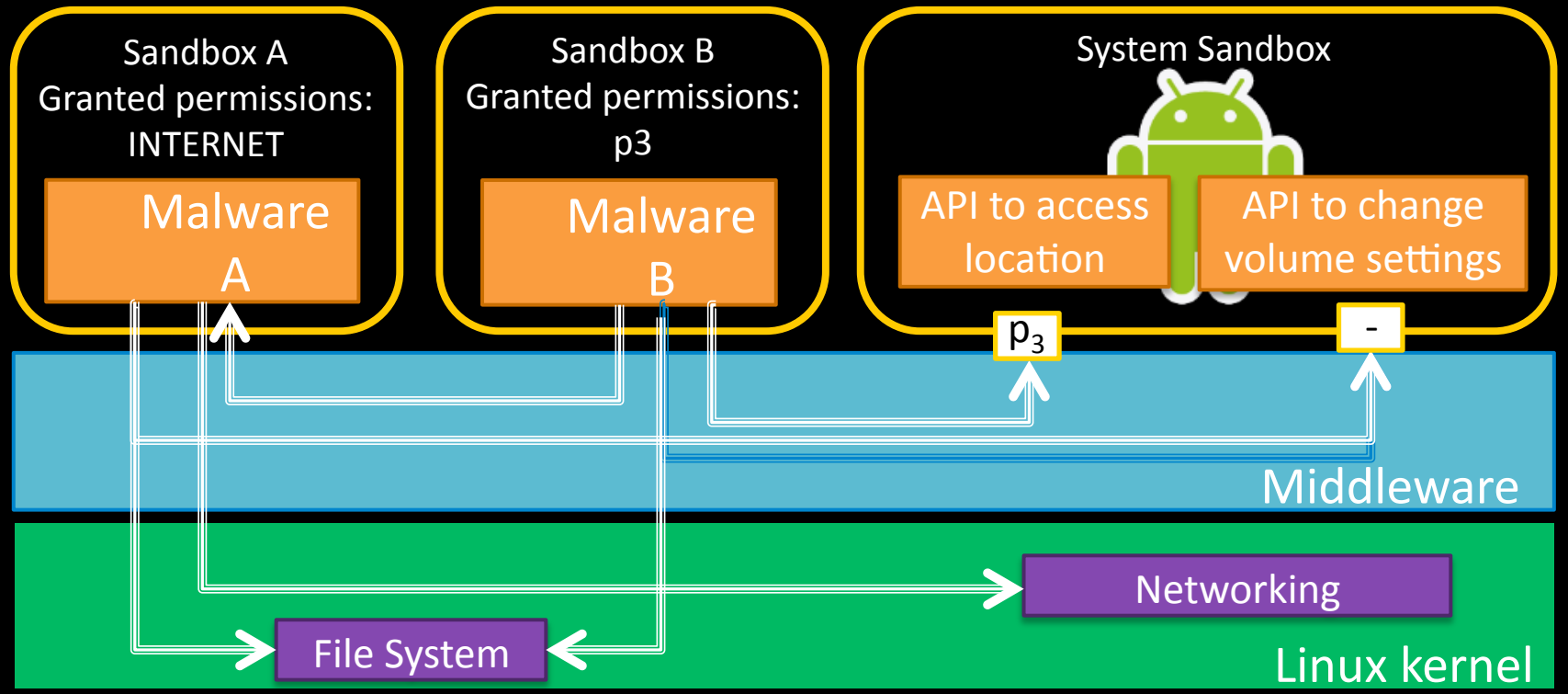
- Invoke browser to download malicious files (Lineberry et al., BlackHat 2010)
- Invoke Phone app to perform a phone call (Enck et al., TechReport 2008)
- Invoke Android Scripting Environment (ASE) to send SMS messages (Davi et al.,ISC 2010)



Collusion Attack

Divide necessary permissions among two (or more) apps

- 1st app has access to user location; 2nd is granted Internet access
  - Apps communicate directly
  - Apps communicate via overt/covert channels in a System Sandbox (e.g., write/read data to system databases, or write-read system settings) (Soundcomber, NDSS`2011)
  - Apps communicate via file sharing



Evaluation

1. Effectiveness (attack prevention)
2. Rate of falsely denied communications
3. Performance

Evaluation

1. Effectiveness (attack prevention)
2. Evaluation of application communication patterns
3. Rate of falsely denied communications
4. Performance

Performance

- Small runtime overhead
  - Non-Intents: overhead is low, especially considering ratio for cached/uncached decisions, low standard deviation
  - Intents: overhead is higher due to analysis of data included into Intent

IPC Type	Number of Calls	Average	Std. Dev.
Runtime for IPC without XManDroid			
All types	11003	0.184 ms	2.490 ms
XManDroid overhead for IPC			
Uncached, except Intents	312	6.182 ms	9.703 ms
Cached, except Intents	10691	0.367 ms	1.930 ms
Intents (never cached)	1821	8.621 ms	29.011 ms
XManDroid overhead for file read			
File read	389	3.320 ms	4.088 ms

Performance (ctd.)

- Access to System Content Providers imposes higher overhead, because multiple reader-writer pairs have to be checked

	Number of Calls	Average	Std. Dev.
Access to System Content Provides			
Without XManDroid	591	10.317 ms	41.224 ms
Overhead of XManDroid	591	4.984 ms	36.441 ms
Access to System Services			
Without XManDroid	87	8.578 ms	20.241 ms
Overhead of XManDroid	87	0.307 ms	0.4318 ms

Evaluation: Attack Prevention

- Malware test suite
  - Misusing Phone app (Enck et al. [TechReport 2008])
  - Misusing Android Scripting Environment (Davi et al. [ISC'2010])
  - Collusion over covert channels (Schlegel et al. [NDSS 2011])

- Deployed policy
  - 7 rules target confused deputy attacks
  - 4 more general rules target collusion attacks

=> All attacks were successfully prevented

Expected result, as we had policy rules against all these attacks

Rate of Falsely Denied Communications

- Test setup
  - Manual tests by 25 users
  - 50 third party applications
- Policy
  - 7 rules targeting confused deputy attacks
  - 4 rules targeting collusion attacks
  - 1 exceptional policy rule to allow launching other apps by sending Intents which do not include any data

=> No false positives were detected

Result confirms our observations on inter-application communication patterns

Evaluation

1. Effectiveness on malware testbed
2. Performance
3. Rate of falsely denied communications
4. Evaluation of application communication patterns