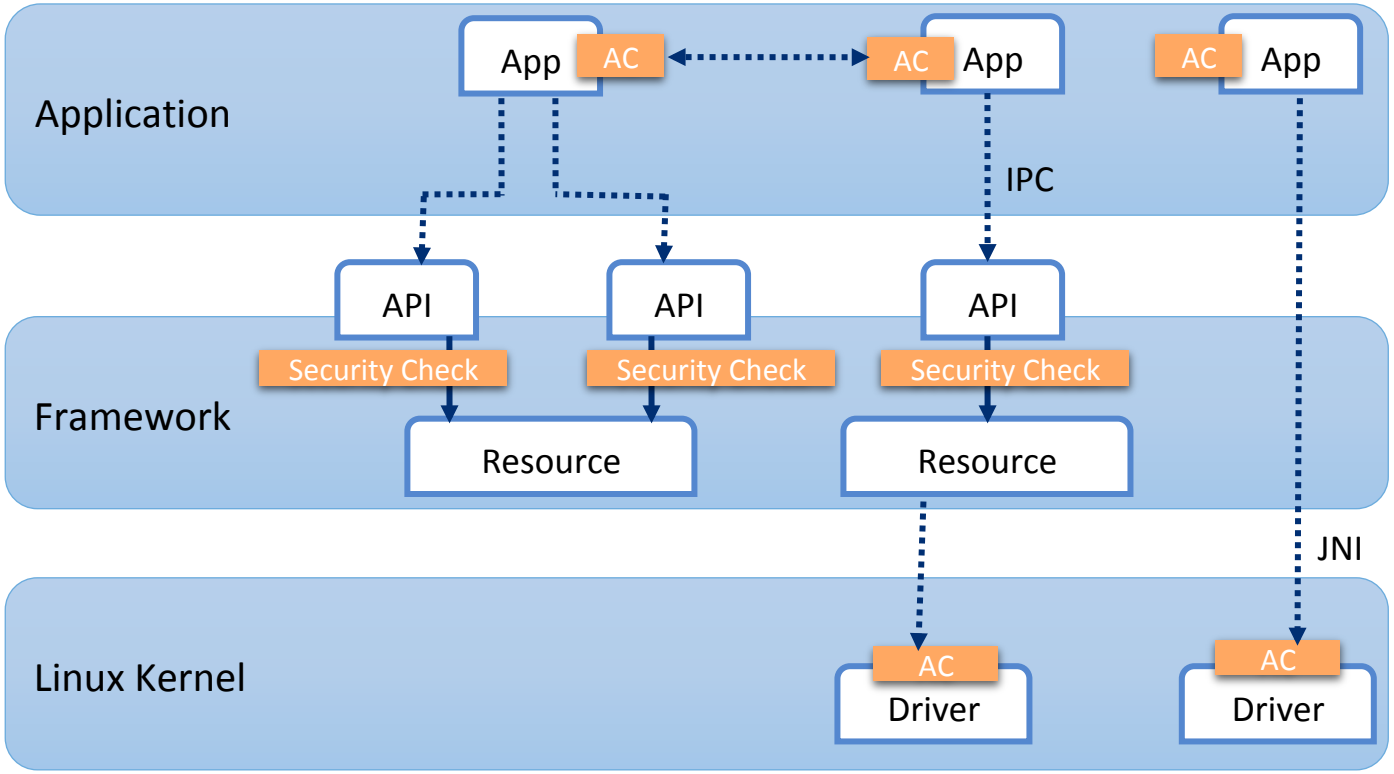
AceDroid: Normalizing Diverse Android Access Control Checks for Inconsistency Detection

Yousra Aafer*, Jianjun Huang*, **Yi Sun***, Xiangyu Zhang*, Ninghui Li* and Chen Tian†

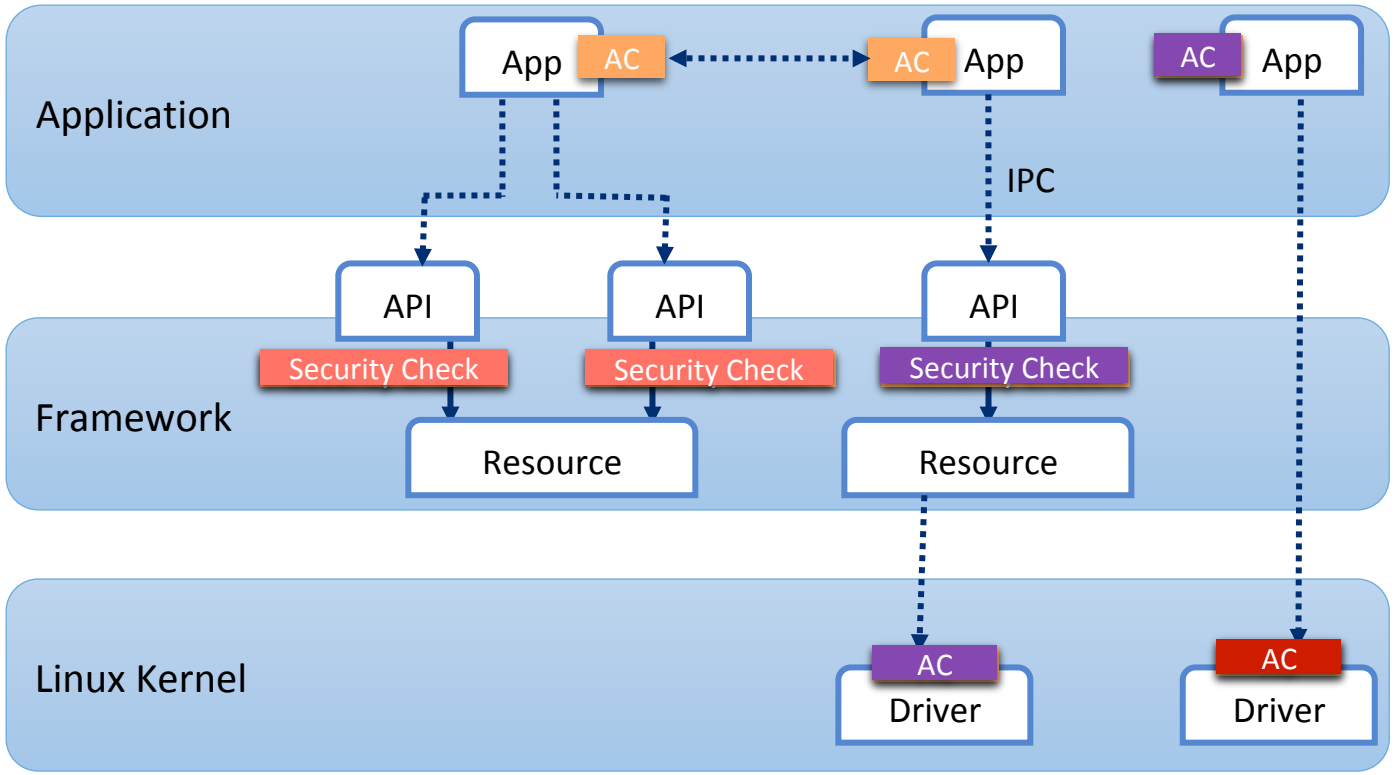
*Purdue University
†Futurewei Technologies

Android Access Control Model



Android Access Control Model: *Effective?*

- **Lack of an Oracle:** It's difficult to determine if a resource is correctly protected
- **Approximate Solution:** Compare AC enforcements across multiple instances of the same resource
  Inconsistencies are potential vulnerabilities



DroidDiff (Aafer et al., Usenix'16)

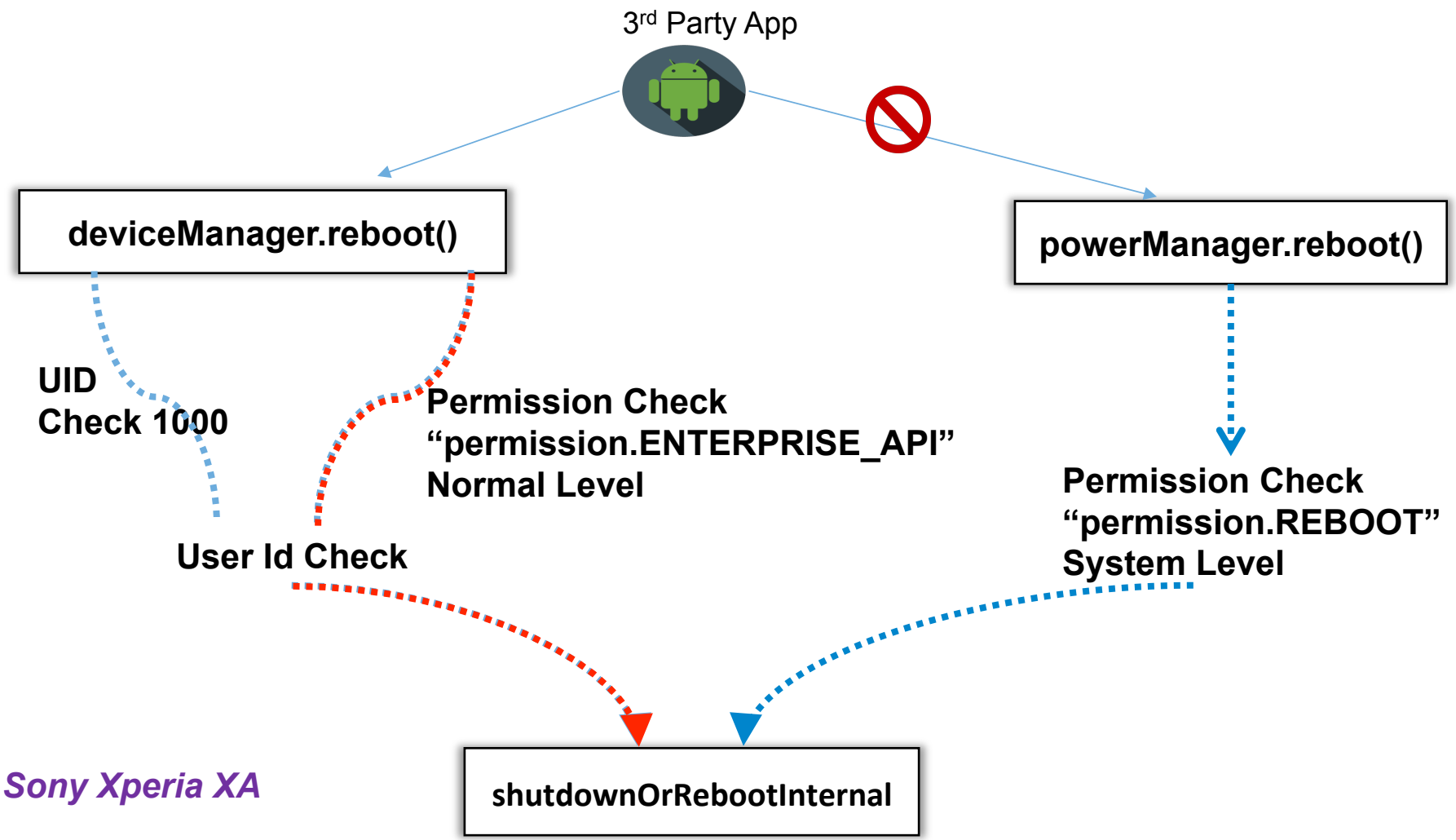
Kratos (Shao et al., NDSS'16)

ADDICTED (Zhou et al., S&P'14)

- **Many challenges cannot be addressed by the existing work Kratos.**

Android Framework AC Features **Diversity:**

Example: Exploitable Inconsistency



Sony Xperia XA

Android Framework AC Features **Diversity:**

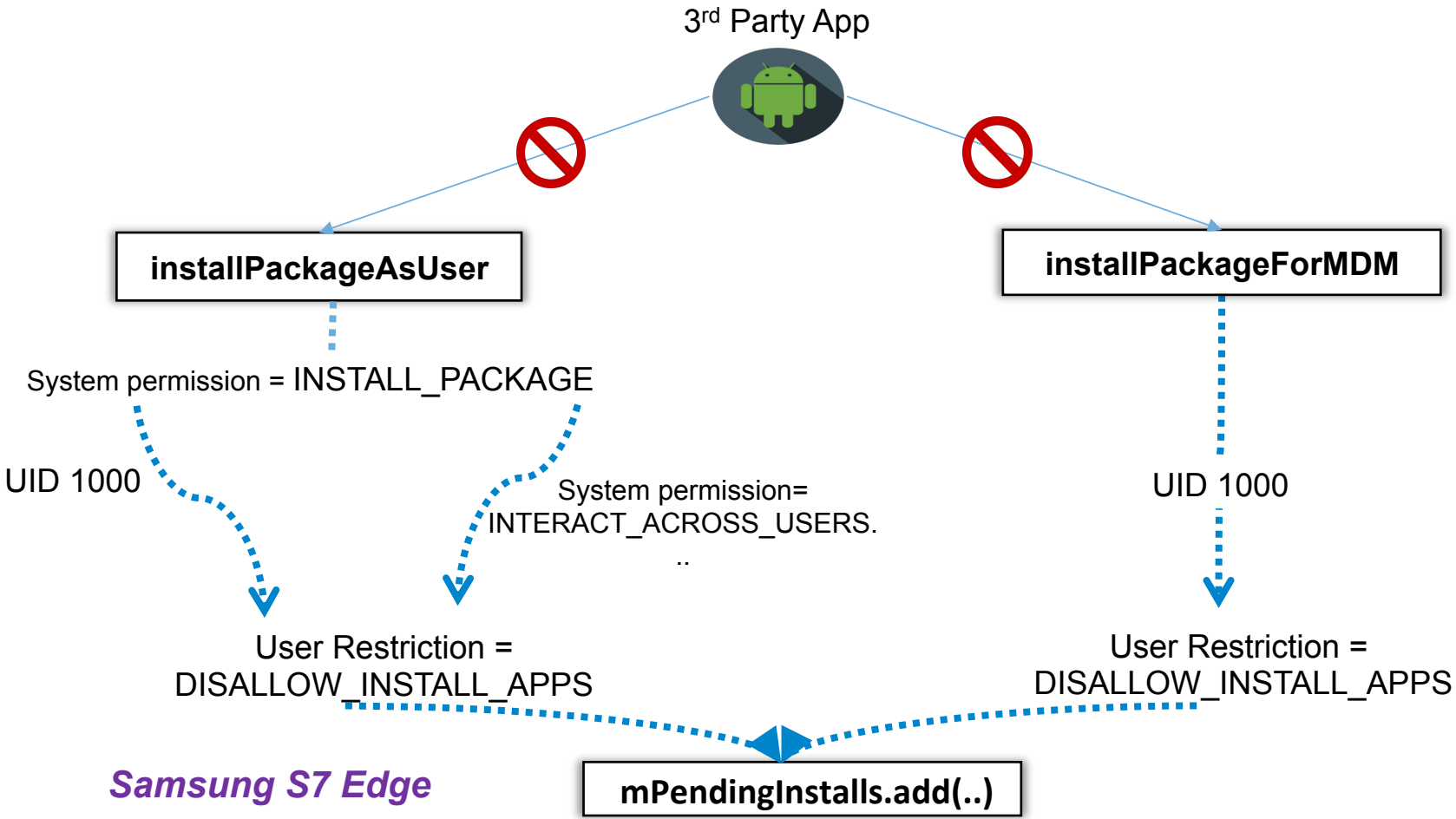
Example: Non-Exploitable Inconsistency

- ***Framework developers do not have a gold standard*** to implement appropriate access control
- ***Diverse ways*** to achieve the ***same*** protection at the framework layer
- If not taking into consideration, this diversity can lead to a ***significant number of false alarms***

Android Framework AC Features **Diversity:**

Example: Non-Exploitable Inconsistency

- ***Framework developers do not have a gold standard*** to implement appropriate access control
- ***Diverse ways*** to achieve the ***same*** protection at the framework layer
- If not taking into consideration, this diversity can lead to a ***significant number of false alarms***

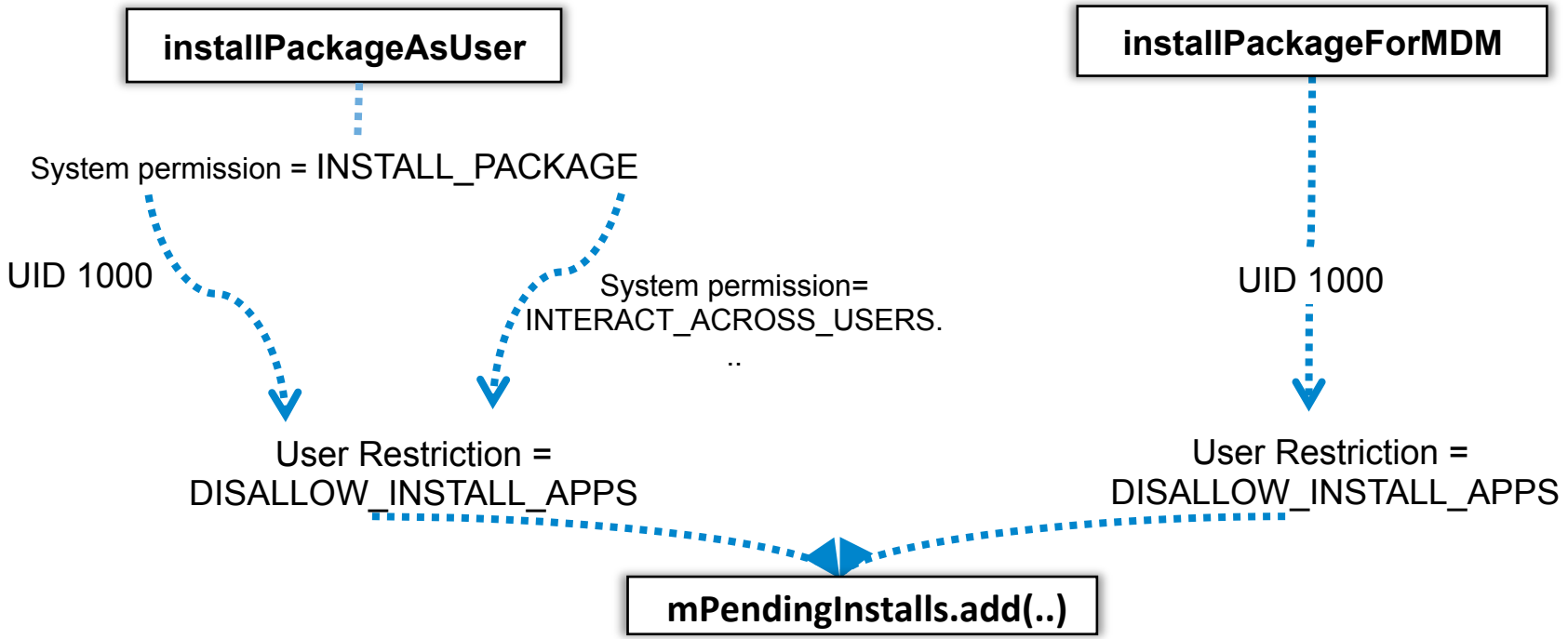


How do existing works handle this case:

- ***Unions all security checks*** from entry point to sink, regardless of their program structure.
- Only considers a number of ***explicit*** checks (e.g., permissions, UID checks)

installPackageAsUser :{permission =INSTALL_PACKAGE, UID=1000, permission = INTERACT.._USERS}

installPackageForMDM : {UID=1000}



AceDroid Solution

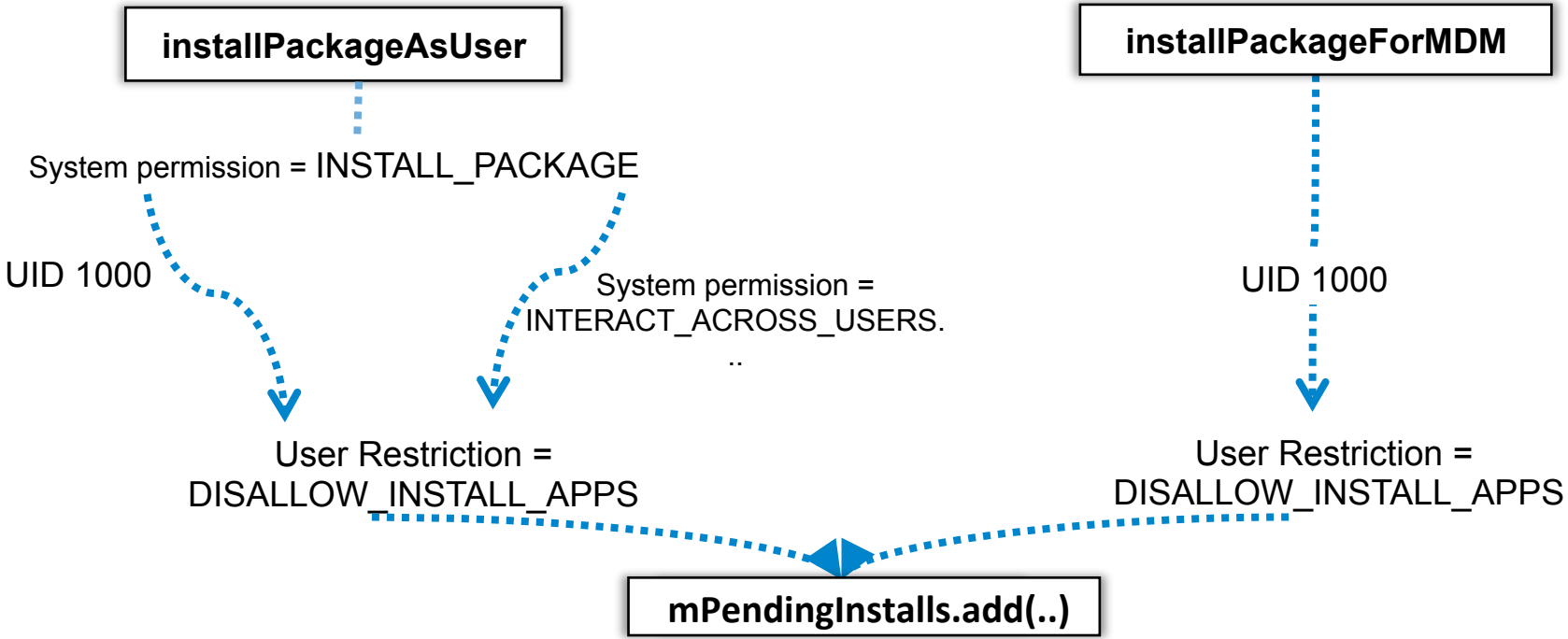
- ***Conduct Access Control Normalization to detect Exploitable Inconsistencies:***

  - Normalizes various security checks to canonical values following the program semantics
  - Handles different program structures such as if-else, loops, etc.
  - Allows precise comparison across different implementation

AceDroid Solution

- ***Access Control Normalization*** technique:
- Both ***installPackageAsUser*** and ***installPackageForMDM*** have the following concise canonical value:

App:= [System] and **User** := [Restriction = DISALLOW_INSTALL_APPS]



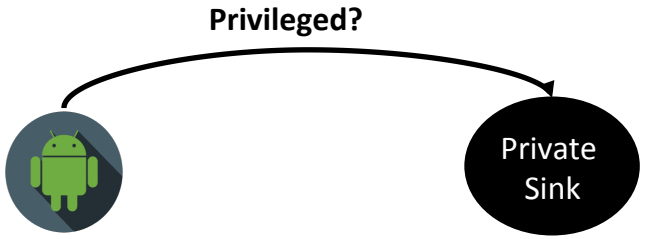
Categorization of Android Access Control

- We model each access control check as a pair consisting of ***app*** and ***user*** aspects
  - ***App aspect:*** aims to check if the app that tries to access the resource has the needed credentials
  - ***User aspect:*** determines if the user of the app that tries to access the resource has a certain role
- Each aspect is a vector of ***multiple orthogonal dimensions***:

```
public void reboot(...) {
  enforceCallingPermission("android.permission.REBOOT");
   shutdownOrRebootInternal(..);
```

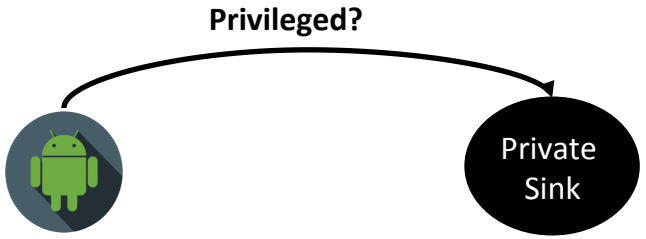
App privilege: Permissions , UID , PID , Package properties (signature..)

Normalization of Permissions: **SYSTEM > DANGEROUS > NORMAL**



Categorization of Android Access Control

- We model each access control check as a pair consisting of ***app*** and ***user*** aspects
  - ***App aspect:*** aims to check if the app that tries to access the resource has the needed credentials
  - ***User aspect:*** determines if the user of the app that tries to access the resource has a certain role
- Each aspect is a vector of ***multiple orthogonal dimensions***:
- Our Normalization handles various program structures



Multiple permissions are enforced:

```
public boolean requestRouteToHostAddress(...){
  enforceCallingPermission("permission.CHANGE_NETWORK_STATE");
  enforceCallingPermission("permission.CONNECTIVTY_INTERNAL");
  addRouteToAddress(...);
```

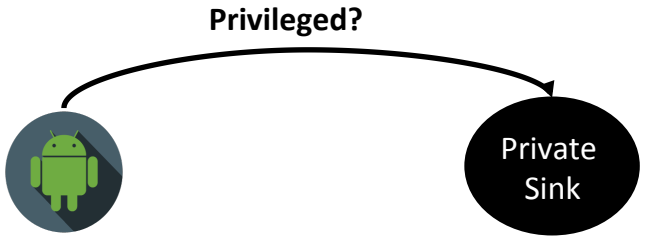
"Permission.CHANGE_NETWORK_STATE" ***= NORMAL LEVEL***

"Permission.CONNECTIVITY_INTERNAL"= ***SYSTEM LEVEL***

Normalized Value = Max(normal, system) => SYSTEM

Categorization of Android Access Control

- We model each access control check as a pair consisting of ***app*** and ***user*** aspects
  - ***App aspect:*** aims to check if the app that tries to access the resource has the needed credentials
  - ***User aspect:*** determines if the user of the app that tries to access the resource has a certain role
- Each aspect is a vector of ***multiple orthogonal dimensions***:
- Our Normalization handles various program structures



Either permission is enforced:

```
public boolean getSubscriberId(...){
  try{
    enforceCallingPermission("READ_PRIVILEGED_PHONE_STATE");
  }catch(SecurityException){
    enforceCallingPermission("READ_PHONE_STATE");
  }
  return mPhone.getSubscriberId();
```

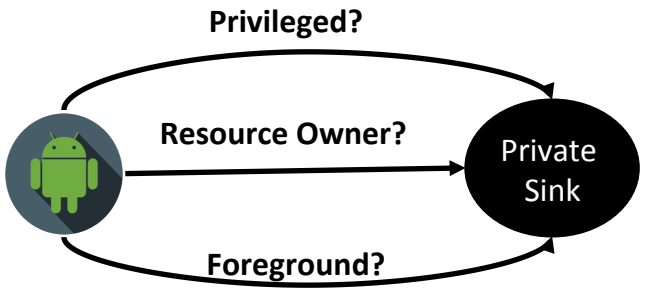
"READ_PRIVILEGED_PHONE_STATE" ***= SYSTEM LEVEL***

"READ_PHONE_STATE"= ***NORMAL LEVEL***

Normalized Value = Min(normal, system) => NORMAL

Categorization of Android Access Control

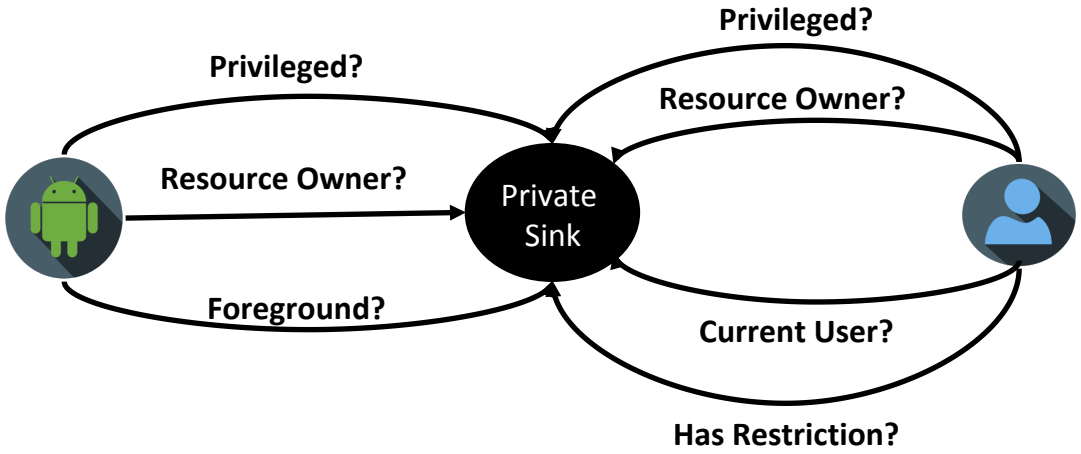
- We model each access control check as a pair consisting of ***app*** and ***user*** aspects
  - ***App aspect:*** aims to check if the app that tries to access the resource has the needed credentials
  - ***User aspect:*** determines if the user of the app that tries to access the resource has a certain role
- Each aspect is a vector of ***multiple orthogonal dimensions***:




```
public void clearApplicationUserData(String packageName,..){
   pkgUid = pm.getPackageUid(packageName, ..);
   if (Binder.getCallingUid() == pkgUid)
     pm.clearApplicationUserData(..);
```

Categorization of Android Access Control

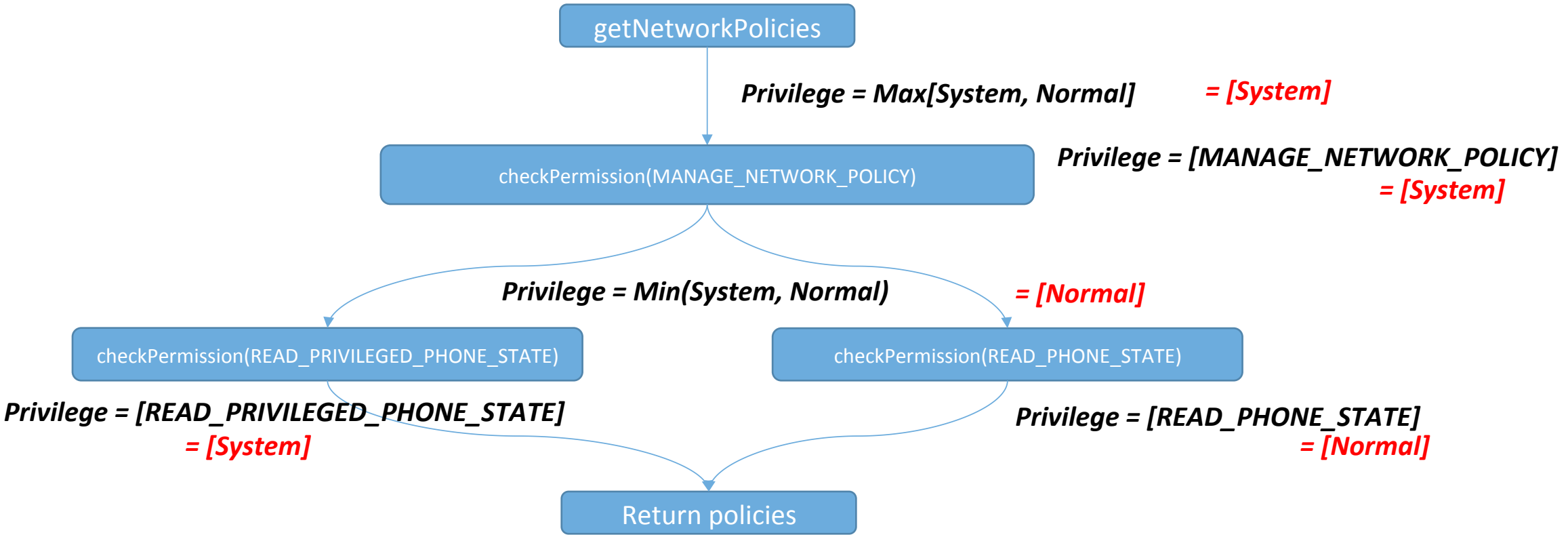
- We model each access control check as a pair consisting of ***app*** and ***user*** aspects
  - ***App aspect:*** aims to check if the app that tries to access the resource has the needed credentials
  - ***User aspect:*** determines if the user of the app that tries to access the resource has a certain role
- Each aspect is a vector of ***multiple orthogonal dimensions***:



System Design: *Modeling Security Checks*

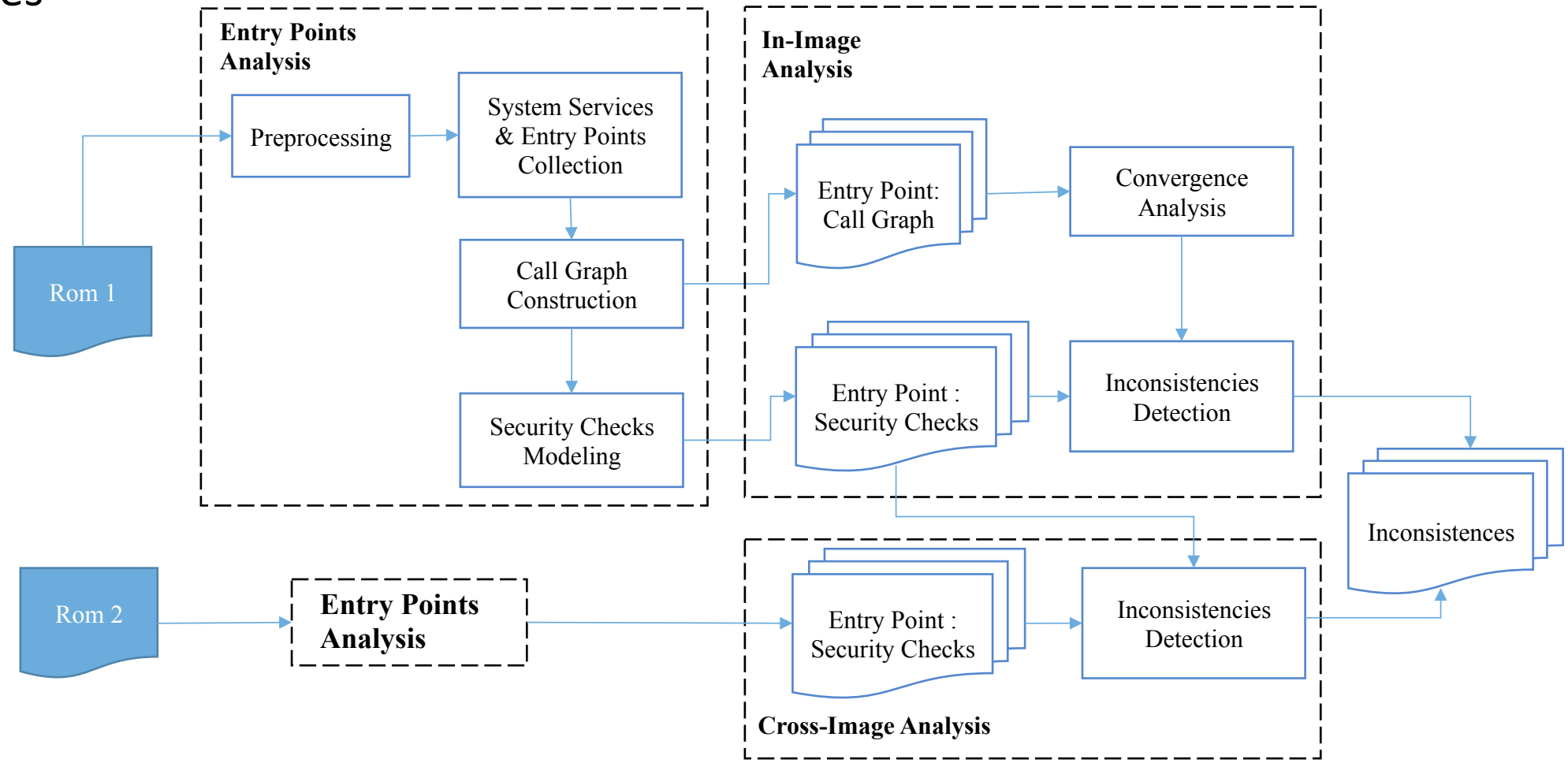
- Modeling security checks for a given API:

```
public NetworkPolicy[] getNetworkPolicies(..) {
  enforceCallingOrSelfPermission("MANAGE_NETWORK_POLICY");
  if(checkPermission("READ_PRIVILEGED_PHONE_STATE")
  == PERMISSION_GRANTED ||
     checkPermission("READ_PHONE_STATE")
  == PERMISSION_GRANTED)
    return policies;
```



System Design

- ***In-Image Analysis:*** compares access control to a same resource to discover inconsistencies
- ***Cross-Image Analysis:*** identifies inconsistencies along similar APIs across two Android images



Evaluation: Inconsistencies Landscape: *in-image*

In-Image (TP/ Reported Inconsistencies)

Image	Nexus 5.0.2	Nexus 6.0	Nexus 6.0.1	Samsung S6 Edge 6.0.1	Samsung Tab S 8.4 6.0.1	Samsung S7 Edge 7.0	LG G3 5.0.2	LG G4 6.0	HTC M8 5.0.2	HTC M8 6.0	Sony Xperia XA 6.0	Sony Xperia XZ 7.0
Nexus 5.0.2	21/32											
Nexus 6.0		15/29										
Nexus 6.0.1			12/26									
S6 Edge 6.0.1				36/64								
Tab S 8.4 (6.0.1)					30/53							
S7 Edge 7.0						39/68						
LG G3 5.0.2							23/41					
LG G4 6.0								28/41				
HTC M8 5.0.2									30/47			
HTC M8 6.0										29/46		
Xperia XA 6.0											32/48	
Xperia XZ 7.0												34/54

- On Average, we achieve ***63% increase*** over Krato's TP.
- Customized ROMs exhibit a higher number of inconsistencies.

Image	**AceDroid**		**Simkratos**		**TP**
	# Inc*	**TP**	**# Inc***	**TP**	**%↑**
Nexus 5.0.2	32	21 (65.6 %)	53	13 (24.5 %)	62
Nexus 6.0	29	15 (51.7 %)	47	7 (14.9 %)	114
Nexus 6.0.1	26	12 (46.2 %)	45	7 (15.6 %)	71
S6 Edge 6.0.1	64	36 (57.8 %)	98	26 (26.5 %)	42
Tab S 8.4 6.0.1	53	30 (56.6 %)	92	21 (22.8 %)	43
S7 Edge 7.0	68	39 (57 %)	103	18 (17.5 %)	56
LG G3 5.0.2	41	23 (57 %)	71	16 (22.1 %)	44
LG G4 6.0	41	28 (63.3 %)	71	17 (23.6 %)	43
HTC M8 5.0.2	47	30 (63.8 %)	71	18 (25.4 %)	67
HTC M8 6.0	46	29 (63 %)	68	18 (26.5 %)	61
Xperia XA 6.0	48	32 (66 %)	69	18 (26.1 %)	72
Xperia XZ 7.0	54	34 (62 %)	75	20 (26.7 %)	70

Evaluation: Inconsistencies Landscape: *cross-image*

Cross-Image (TP / Reported Inconsistencies)

Image	Nexus 5.0.2	Nexus 6.0	Nexus 6.0.1	Samsung S6 Edge 6.0.1	Samsung Tab S 8.4 6.0.1	Samsung S7 Edge 7.0	LG G3 5.0.2	LG G4 6.0	HTC M8 5.0.2	HTC M8 6.0	Sony Xperia XA 6.0	Sony Xperia XZ 7.0
Nexus 5.0.2		13/17	17/19	38/47	35/45	39/51	7/9	28/36	9/12	24/32	26/35	26/34
Nexus 6.0			6/9	27/37	24/35	34/38	20/26	15/19	22/28	11/15	13/18	36/53
Nexus 6.0.1				21/28	18/26	32/40	24/28	21/28	24/31	15/22	19/27	19/26
S6 Edge 6.0.1					12/16	26/33	40/51	35/48	37/49	26/40	28/43	36/52
Tab S 8.4 (6.0.1)						29/35	34/48	31/46	34/47	22/37	23/39	31/46
S7 Edge 7.0							43/51	35/52	37/49	26/40	28/43	36/52
LG G3 5.0.2								19/26	13/17	31/41	28/39	33/43
LG G4 6.0									28/37	26/32	23/32	28/36
HTC M8 5.0.2										23/33	21/31	35/46
HTC M8 6.0											26/41	24/32
Xperia XA 6.0												16/21
Xperia XZ 7.0												

- Inconsistencies are prevalent:
  - Across different vendors
  - Even within the same vendor.

Evaluation: Inconsistencies Landscape

Image	Nexus 5.0.2	Nexus 6.0	Nexus 6.0.1	Samsung S6 Edge 6.0.1	Samsung Tab S 8.4 6.0.1	Samsung S7 Edge 7.0	LG G3 5.0.2	LG G4 6.0	HTC M8 5.0.2	HTC M8 6.0	Sony Xperia XA 6.0	Sony Xperia XZ 7.0
Nexus 5.0.2						39/51						
Nexus 6.0	101											
Nexus 6.0.1	133	55										
S6 Edge 6.0.1	546	446	410									
Tab S 8.4 (6.0.1)	503	422	379	212								
S7 Edge 7.0	562	457	498	289	314							
LG G3 5.0.2	115	96	188	338	305	468						
LG G4 6.0	209	198	222	403	378	313	215					
HTC M8 5.0.2	68	183	198	331	298	325	233	401				
HTC M8 6.0	186	87	119	268	243	366	274	333	264			
Xperia XA 6.0	183	89	123	284	252	369	274	340	271	312		
Xperia XZ 7.0	246	186	221	410	389	491	305	238	294	213	247	

Number of inconsistencies if no Normalization is performed

- On average, we can reduce the false alarms from ***229*** to ***13 instances***.

Findings: Confirmed Attacks

- 27 confirmed attacks.
- 2 ranked as ***critical*** by LG.

TABLE V. CONFIRMED ATTACKS

Security Impact	Description	Victim Device(s)
Privilege Escalation	Eavesdropping on input events such as screen taps	LG G4 6.0
Privilege Escalation	Intercepting and injecting input events such as screen taps	LG G4 6.0
Privilege Escalation	Sending SMS messages including premium messages	S6 Edge 6.0.1
DoS	Denying receiving of SMS messages	S6 Edge (6.0.1) HTC M8 6.0
Privilege Escalation	Enabling Bluetooth Quietly	S6 Edge (6.0.1) LG G4 6.0
Privilege Escalation	Persist Bluetooth Settings	LG G4
Privilege Escalation	Bypassing and Forging User Restrictions	S6 Edge 6.0.1 S7 Edge (7.0)
Privilege Escalation	Injecting Hard Key Events such as Volume Up, Power Off, Screen Off	Sony Xperia 6.0
Privilege Escalation	Rebooting the phone into Recovery Mode	Sony Xperia 6.0
Privilege Escalation	Phone Shutdown	Sony Xperia XA 6.0
Privilege Escalation	Turning Radio On / Off	LG G3 5.0.2
DoS	Unmounting SD Card persistently	HTC M8 6.0
DoS	Turning-Off Wifi persistently	HTC M8 6.0
DoS	Turning-Off Bluetooth persistently	LG G3 5.0.2
Privilege Escalation	Manipulating Network Firewall Rules	Xperia XA 6.0

Q&A

Thank you!