**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

Application

Framework

Linux Kernel

API

Security Check

Resource

API

Security Check

Resource

API

Security Check

JNI

Driver

AC

App

IPC

Security Check

Driver

AC

App

AC

App

AC

App
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*

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

---

- **DroidDiff** (Aafer et al., Usenix’16)
- **Kratos** (Shao et al., NDSS’16)
- **ADDICTED** (Zhou et al., S&P’14)
Android Framework AC Features Diversity:

Example: Exploitable Inconsistency

```
deviceManager.reboot()
```

UID Check 1000

User Id Check

Permission Check “permission.ENTERPRISE_API”
Normal Level

```
shutdownOrRebootInternal()
```

```
powerManager.reboot()
```

Permission Check “permission.REBOOT”
System Level

3rd Party App

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*

---

**Diversity:**

- 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

---

**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

---

**Diagram:**

- **3rd Party App**
  - **installPackageAsUser**
    - System permission = INSTALL_PACKAGE
    - UID 1000
    - User Restriction = DISALLOW_INSTALL_APPS
  - **installPackageForMDM**
    - System permission = INTERACT_ACROSS_USERS
    - UID 1000
    - User Restriction = DISALLOW_INSTALL_APPS

---

**Samsung S7 Edge**

- mPendingInstalls.add(())
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:

  \[
  \text{App} := \text{[System]} \text{ and } \text{User} := \text{[Restriction = DISALLOW_INSTALL_APPS]}
  \]

### Diagram

- **installPackageAsUser**
  - System permission = INSTALL_PACKAGE
  - UID 1000
  - User Restriction = DISALLOW_INSTALL_APPS

- **installPackageForMDM**
  - UID 1000
  - User Restriction = DISALLOW_INSTALL_APPS

- `mPendingInstalls.add(..)`
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:

```java
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

```
public boolean requestRouteToHostException(...){
    enforceCallingPermission("permission.CHANGE_NETWORK_STATE");
    enforceCallingPermission("permission.CONNECTIVITY_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:

```java
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**:

```java
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**:

```
Privileged?
Resource Owner?
Foreground?

Private Sink

Privileged?
Resource Owner?
Current User?
Has Restriction?
```
System Design: **Modeling Security Checks**

- Modeling security checks for a given API:

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

- **checkPermission(MANAGE_NETWORK_POLICY)**
  - Privilege = [MANAGE_NETWORK_POLICY] = [System]

- **checkPermission(READ_PRIVILEGED_PHONE_STATE)**
  - Privilege = [READ_PRIVILEGED_PHONE_STATE] = [System]
  - Privilege = Min(System, Normal) = [Normal]

- **checkPermission(READ_PHONE_STATE)**
  - Privilege = [READ_PHONE_STATE] = [System]
  - Privilege = [READ_PHONE_STATE] = [Normal]

- **Return policies**
  - Privilege = Min(System, Normal) = [Normal]
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)**

<table>
<thead>
<tr>
<th>Image</th>
<th>Nexus 5.0.2</th>
<th>Nexus 6.0</th>
<th>Nexus 6.0.1</th>
<th>Samsung S6 Edge 6.0.1</th>
<th>Samsung Tab S 8.4 6.0.1</th>
<th>Samsung S7 Edge 7.0</th>
<th>LG G3 5.0.2</th>
<th>LG G4 6.0</th>
<th>HTC M8 5.0.2</th>
<th>HTC M8 6.0</th>
<th>Sony Xperia XA 6.0</th>
<th>Sony Xperia XZ 7.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nexus 5.0.2</td>
<td>21/32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nexus 6.0</td>
<td>15/29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nexus 6.0.1</td>
<td>12/26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S6 Edge 6.0.1</td>
<td></td>
<td>36/64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tab S 8.4 (6.0.1)</td>
<td>30/53</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S7 Edge 7.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LG G3 5.0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LG G4 6.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTC M8 5.0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTC M8 6.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xperia XA 6.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xperia XZ 7.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- On Average, we achieve **63% increase** over Krato’s TP.
- Customized ROMs exhibit a higher number of inconsistencies.
Evaluation: Inconsistencies Landscape: \textit{cross-image}

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

• Inconsistencies are prevalent:
  • Across different vendors
  • Even within the same vendor.
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>
<thead>
<tr>
<th>Security Impact</th>
<th>Description</th>
<th>Victim Device(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privilege Escalation</td>
<td>Eavesdropping on input events such as screen taps</td>
<td>LG G4 6.0</td>
</tr>
<tr>
<td>Privilege Escalation</td>
<td>Intercepting and injecting input events such as screen taps</td>
<td>LG G4 6.0</td>
</tr>
<tr>
<td>Privilege Escalation</td>
<td>Sending SMS messages including premium messages</td>
<td>S6 Edge 6.0.1</td>
</tr>
<tr>
<td>DoS</td>
<td>Denying receiving of SMS messages</td>
<td>S6 Edge (6.0.1)</td>
</tr>
<tr>
<td>DoS</td>
<td></td>
<td>HTC M8 6.0</td>
</tr>
<tr>
<td>Privilege Escalation</td>
<td>Enabling Bluetooth Quietly</td>
<td>S6 Edge (6.0.1)</td>
</tr>
<tr>
<td>Privilege Escalation</td>
<td></td>
<td>LG G4 6.0</td>
</tr>
<tr>
<td>Privilege Escalation</td>
<td>Persist Bluetooth Settings</td>
<td>LG G4</td>
</tr>
<tr>
<td>Privilege Escalation</td>
<td>Bypassing and Forging User Restrictions</td>
<td>S6 Edge 6.0.1</td>
</tr>
<tr>
<td>Privilege Escalation</td>
<td></td>
<td>S7 Edge (7.0)</td>
</tr>
<tr>
<td>PrivilegeEscalation</td>
<td>Injecting Hard Key Events such as Volume Up, Power Off, Screen Off</td>
<td>Sony Xperia 6.0</td>
</tr>
<tr>
<td>Privilege Escalation</td>
<td>Rebooting the phone into Recovery Mode</td>
<td>Sony Xperia 6.0</td>
</tr>
<tr>
<td>Privilege Escalation</td>
<td>Phone Shutdown</td>
<td>Sony Xperia XA 6.0</td>
</tr>
<tr>
<td>Privilege Escalation</td>
<td>Turning Radio On / Off</td>
<td>LG G3 5.0.2</td>
</tr>
<tr>
<td>DoS</td>
<td>Unmounting SD Card persistently</td>
<td>HTC M8 6.0</td>
</tr>
<tr>
<td>DoS</td>
<td>Turning-Off Wifi persistently</td>
<td>HTC M8 6.0</td>
</tr>
<tr>
<td>DoS</td>
<td>Turning-Off Bluetooth persistently</td>
<td>LG G3 5.0.2</td>
</tr>
<tr>
<td>Privilege Escalation</td>
<td>Manipulating Network Firewall Rules</td>
<td>Xperia XA 6.0</td>
</tr>
</tbody>
</table>
Thank you!