Life after App Uninstallation: Are the Data Still Alive?

*Data Residue Attacks on Android*

Xiao Zhang, Kailiang Ying, Yousra Aafer, Zhenshen Qiu, and Wenliang Du
App Life

Installation

Interaction

Uninstallation
But, what if ... 

Windows Residue

Android App Uninstallation

Are there any data left after application uninstallation on Android?
In Details

Are the data still alive after application uninstallation on Android?
What can go wrong?

Are the data still alive in Android system services after application uninstallation?
Methodology

System Service Collection \(\rightarrow\) Filtering \(\rightarrow\) Candidate Database \(\rightarrow\) Manual Analysis \(\rightarrow\) Residue Instances

Protection Examination \(\rightarrow\) Exploit Attempts \(\rightarrow\) Attack Design \(\rightarrow\) Damage Measurement

Saving data to files, databases? Or Saving data in memory?

Candidate Service \(\rightarrow\) Data Residue Harvest \(\rightarrow\) Data cleanup (flaw)?

Yes \(\rightarrow\) Data Residue \(\rightarrow\) exploits Vulnerability
No \(\rightarrow\)
Findings

<table>
<thead>
<tr>
<th>Samples</th>
<th>Category</th>
<th>Service Instances</th>
<th>Residues</th>
<th>Exploitable</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Services (96/96/10)</td>
<td>Credential Residue</td>
<td>AccountManager</td>
<td>User Credentials</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Keystore</td>
<td>Public/Private Keypairs</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Capability Residue</td>
<td>Clipboard</td>
<td>URI</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ActivityManager</td>
<td>PendingIntent</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td>System-app Services (161/26/2)</td>
<td>TextService</td>
<td>User Selected Components</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DebugService</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DreamService</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TrustAgent</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LocationManager</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Settings Residue</td>
<td>PrintService</td>
<td>Print/Download Information</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DownloadService</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>History Residue</td>
<td>PackageManager</td>
<td>Permissions</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Permission Residue</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

* Resolved on Android Lollipop, but reproducible on KitKat and prior versions

- 7 security vulnerabilities acknowledged by Google with Medium priority
Sample Exploits - I

- **Credential Stealing**
Sample Exploits - II

- **Settings Impersonating**
Sample Exploits - II

- Settings Impersonating
Even More ...

Details are available at:
https://sites.google.com/site/droidnotsecure/
## Evaluation

- 2,373 apps
- 10 devices
- 8 Android versions
- 3 play stores

<table>
<thead>
<tr>
<th></th>
<th>package</th>
<th>account type</th>
<th>authority</th>
</tr>
</thead>
<tbody>
<tr>
<td>GooglePlay</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Amazon Appstore</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Samsung Appstore</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attack Instances</th>
<th>Account</th>
<th>Clipboard</th>
<th>Download</th>
<th>Dream</th>
<th>Keystore</th>
<th>Permission</th>
<th>Print</th>
<th>Spell Checker</th>
</tr>
</thead>
<tbody>
<tr>
<td># Targets</td>
<td>131</td>
<td>92</td>
<td>17</td>
<td>24</td>
<td>63</td>
<td>55</td>
<td>49</td>
<td>16</td>
</tr>
</tbody>
</table>

### I: Analysis on Real-world Applications

### II: Examination on Essential Attributes

### III: Measurement on Device Customization Influence†

† N/A¹: feature Not Available because of the low Android version; N/A²: feature Not Available because of the vendor customization.
Fundamental Causes

• Data Residue Instances <-> *Mandatory Design Principle in Backend*

• Exploits <-> *Signature-based Frontend*

<table>
<thead>
<tr>
<th>Layers</th>
<th>Attributes</th>
<th>Assumptions</th>
<th>Protection Effectiveness</th>
<th>Breaking Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>PID</td>
<td>process isolation</td>
<td>Hard Isolation</td>
<td>—</td>
</tr>
<tr>
<td>Framework</td>
<td>UID</td>
<td>UID exclusion</td>
<td>individual device cycle</td>
<td>device rebooting</td>
</tr>
<tr>
<td>Application</td>
<td>package</td>
<td>package exclusion</td>
<td>individual device state</td>
<td>(un)installation</td>
</tr>
<tr>
<td>Component</td>
<td>account type</td>
<td>customized-id exclusion</td>
<td>Invalid</td>
<td>(un)installation</td>
</tr>
<tr>
<td></td>
<td>authority</td>
<td>customized-id exclusion</td>
<td>individual device state</td>
<td>(un)installation</td>
</tr>
</tbody>
</table>

TABLE IV: Security Examination of Android Attributes Used in Protecting Data Residue
Limitation

- Manual Analysis
- Static Analysis
  - App Level
  - Intelligence
- Dynamic Analysis
  - App Level
  - Exploit Conditions

```java
private class TextServicesMonitor extends PackageMonitor {
    @Override
    public void onSomePackagesChanged() {
        synchronized (mSpellCheckerMap) {
            buildSpellCheckerMapLocked(mContext, mSpellCheckerList, mSpellCheckerMap);
            // TODO: Update for each locale
            SpellCheckerInfo sci = getCurrentSpellChecker(null);
            if (sci == null) return;
            final String packageName = sci.getPackageName();
            final int change = isPackageDisappearing(packageName);
            if ((change == PACKAGE_PERMANENT_CHANGE || change == PACKAGE_TEMPORARY_CHANGE) {
                // Package disappearing
                sci = findAvailSpellCheckerLocked(null, packageName);
                if (sci != null) {
                    setCurrentSpellCheckerLocked(sci.getId());
                }
            }
        }
    }
}
```
Conclusion

• Data Residue Vulnerability
• Systematic Study
• Comprehensive Evaluation

• Trigger more research efforts
Questions?

xzhang35@syr.edu

https://sites.google.com/site/droidnotsecure/