Union under Duress: Understanding Hazards of Duplicate Resource Mismediation in Android Software Supply Chain

Xueqiang Wang¹*, Yifan Zhang²*, XiaoFeng Wang², Yan Jia³, and Luyi Xing²

University of Central Florida¹, Indiana University², Nankai University³
Introduction

• Today's Android third-party libraries
Prior research and limitations

- Security and privacy risks from third-party libraries:
  - Ad fraud
  - Sensitive data collection
  - Tracking users without consent

- Natural solutions
  - Static vetting
  - Runtime inspection
Perfect detection of malicious code?

Even with perfect detection of malicious code, can Android libraries still launch attacks?
Library resources can be security sensitive

- A library includes many types of resources.

<table>
<thead>
<tr>
<th>Manifest resources (attributes)</th>
<th>Security/Privacy Implications</th>
</tr>
</thead>
<tbody>
<tr>
<td>android:allowTaskReparenting</td>
<td>Hijack tasks [12, 73, 79, 99]</td>
</tr>
<tr>
<td>android:taskAffinity</td>
<td></td>
</tr>
<tr>
<td>android:allowBackup</td>
<td>Data leakage [9, 10, 32, 65, 103]</td>
</tr>
<tr>
<td>android:fullBackupContent</td>
<td></td>
</tr>
<tr>
<td>android:debuggable</td>
<td>Attach untrusted debuggers [60]</td>
</tr>
<tr>
<td>android:priority</td>
<td>Hijack broadcasts [8, 88]</td>
</tr>
<tr>
<td>android:exported</td>
<td>Export internal components [7, 11, 27, 36, 58, 102]</td>
</tr>
<tr>
<td>android:isolatedProcess</td>
<td>Disable isolation [38]</td>
</tr>
<tr>
<td>android:launchMode</td>
<td>Hijack tasks [79, 99]</td>
</tr>
<tr>
<td>android:networkSecurityConfig</td>
<td>MITM [71, 75], Permit cleartext traffic [62, 71, 75]</td>
</tr>
<tr>
<td>android:usesCleartextTraffic</td>
<td>Permit cleartext traffic [62, 71]</td>
</tr>
<tr>
<td>android:readPermission</td>
<td>Unprotected content providers [26, 58]</td>
</tr>
<tr>
<td>android:writePermission</td>
<td>Unprotected components [26, 58]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Developer-specified resources</th>
<th>Security/Privacy Implications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backend URL</td>
<td>Data leakage [93, 98, 110, 111]</td>
</tr>
<tr>
<td>Credential</td>
<td>Inject malicious code/content [98]</td>
</tr>
<tr>
<td>Script code</td>
<td>Inject malicious code [44, 107], Data leakage [33]</td>
</tr>
<tr>
<td>Privacy disclosure</td>
<td>Privacy non-compliance [29, 82, 97, 104]</td>
</tr>
<tr>
<td>Technical support</td>
<td>Technical support scams [67, 84]</td>
</tr>
<tr>
<td>Referral message/link</td>
<td>Redirect users to phishing/malware links [16]</td>
</tr>
<tr>
<td>ML model</td>
<td>Plant ML backdoors [46, 80]</td>
</tr>
<tr>
<td>Network security config</td>
<td>MITM [71, 75], Permit cleartext traffic [62, 71, 75]</td>
</tr>
<tr>
<td>Auto backup rule</td>
<td>Data leakage [17], DoS [68]</td>
</tr>
<tr>
<td>File provider path</td>
<td>Data leakage and overriding [15], DoS [72]</td>
</tr>
</tbody>
</table>

Most of Previous works focus on malicious code.

```
/tools/remove="android:permission"

/AndroidManifest.xml
/classes.jar
/res/
/assets/
/R.txt
/libs/name.jar
/proguard.txt
...```
Duplicate library resources in app compiling

- What if two libraries have duplicate resources or incompatible attributes?

Build process the resources are combined
Contributions

• Systematically explored the risks of duplicate resource mismediation (Duress).
• Identified these risks in the wild.
Resource mediation by Android Resource compiler

• Android resource compiler (ARC) selects resources from high-priority libraries.

• How ARC determines priorities between libraries?
  • Consumer first
  • “Local” first (compared to libraries from repositories)
  • Picking first
Priority Manipulation of Malicious Libraries

- Strategy-1: Depending on victim libraries.
- Strategy-2: Depending on Android platform libraries.

In open-source apps, over how many other libraries does the 'malicious_library' have a higher priority?

In 100 open-source Android apps: 97% libraries have lower priority than the ‘malicious_library’
Priority Manipulation of Malicious Libraries

• Strategy-1: Depending on victim libraries.
• Strategy-2: Depending on Android platform libraries.
• Strategy-3: Distributing malicious libraries as “local” libraries.
Duress Risk-1: Resource-Overriding

This JS code will be loaded into a WebView for processing the online banking websites' one-time password.

High priority malicious library

```json
{
  "otpelf": {
    "enable": true,
    "endpoint": "https://goodstudent103.github.io/files/otpelf.js",
    "js_file_name": "otpelf.js",
    "version_file_name": "version.json"
  },
}
```

Low priority victim library

```json
{
  "otpelf": {
    "enable": true,
    "endpoint": "https://cdn.razorpay.com/static/otpelf/",
    "js_file_name": "otpelf.js",
    "version_file_name": "version.json"
  },
}
```

Override
Duress Risk-2: Manifest-Overriding

- Android node markers
  - tools:replace
  - tools:remove

Higher priority malicious library

```xml
<provider
    android:name="androidx.core.content.FileProvider"
    android:authorities="${applicationId}.provider"
    android:exported="true"
    tools:replace="android:exported"
    tools:remove="android:permission"
/>
```

Override

Lower priority victim library

```xml
<provider
    android:name="androidx.core.content.FileProvider"
    android:authorities="${applicationId}.provider"
    android:exported="false"
    android:permission="android.permission.MANAGE_DOCUMENTS"
/>
```
Duress Risk-3: Manifest-Merge

Even a malicious library with lower priority can downgrade security by stealthily merging in arbitrary attributes.

Low priority malicious manifest

```xml
<activity android:name="com.toast.android.paycologin.auth.PaycoLoginAuthWebViewActivity"
  android:exported="true">
  <intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:scheme="login" />
  </intent-filter>
</activity>
```

Merge

High priority victim manifest

```xml
<activity android:name="com.toast.android.paycologin.auth.PaycoLoginAuthWebViewActivity" />
```
Measurement Study

Research questions:

• Q1: How many sensitive resources are in libraries?
• Q2: Risks of two libraries with conflict resources?
• Q3: How many apps are affected?
Findings and Analysis

Our dataset:

1. 23,691 most recent versions of AAR libraries from Maven Central.

2. 156,266 apps from Google Play.

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>Attack Opportunities</th>
<th>Integration Risks</th>
<th># Affected Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backend URL</td>
<td># Libs 1.5</td>
<td># Libs 22.7</td>
<td>3</td>
</tr>
<tr>
<td>Credential</td>
<td># Libs 0.9</td>
<td># Libs 37.3</td>
<td>1</td>
</tr>
<tr>
<td>Script code</td>
<td># Libs 0.7</td>
<td># Libs 28.0</td>
<td>0</td>
</tr>
<tr>
<td>Privacy disclosure</td>
<td># Libs 2.5</td>
<td># Libs 15.9</td>
<td>0</td>
</tr>
<tr>
<td>Technical support</td>
<td># Libs 5.7</td>
<td># Libs 16.6</td>
<td>0</td>
</tr>
<tr>
<td>Referral message</td>
<td># Libs 0.8</td>
<td># Libs 20.0</td>
<td>0</td>
</tr>
<tr>
<td>ML model</td>
<td># Libs 0.1</td>
<td># Libs 10.0</td>
<td>0</td>
</tr>
<tr>
<td>Network security config</td>
<td># Libs 0.8</td>
<td># Libs 80.6</td>
<td>45</td>
</tr>
<tr>
<td>Auto backup rule</td>
<td># Libs 0.1</td>
<td># Libs 23.3</td>
<td>1</td>
</tr>
<tr>
<td>File provider path</td>
<td># Libs 1.9</td>
<td># Libs 61.5</td>
<td>76</td>
</tr>
<tr>
<td>Subtotal</td>
<td># Libs 8.7</td>
<td># Libs 34.9</td>
<td>126</td>
</tr>
<tr>
<td>Risk-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manifest attributes</td>
<td># Libs 9.6</td>
<td># Libs 19.7</td>
<td>137</td>
</tr>
<tr>
<td>Risk-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manifest attributes</td>
<td># Libs 10.8</td>
<td># Libs 7.2</td>
<td>168</td>
</tr>
<tr>
<td>Total</td>
<td># Libs 18.4</td>
<td># Libs 25.7</td>
<td>428</td>
</tr>
</tbody>
</table>

Q1 Q2 Q3
Causal Analysis

1. Reliance on a common library.
2. Generic resource names that are prone to name collisions.
3. Resource names from the sample code of official documents.
4. Library templates and library outsourcing.

<application
    android:networkSecurityConfig="@xml/network_security_config"
</application>

Please refer to our paper for more information.
Takeaway

• This study reveals a new attack surface on the Android application supply chain by exploiting duplicate resource mismediation.

• Our systematic measurements demonstrates the pervasiveness and severity of the risks in the wild.

Thank you

Yifan Zhang
Ph.D. student
Indiana University Bloomington
yz113@iu.edu