Lost in Conversion: Exploit Data Structure Conversion with Attribute Loss to Break Android Systems

Rui Li*, Wenrui Diao*(✉), Shishuai Yang*, Xiangyu Liu§, Shanqing Guo*, and Kehuan Zhang‡

* Shandong University § Alibaba Group ‡ The Chinese University of Hong Kong

山东大学
SHANDONG UNIVERSITY

Alibaba

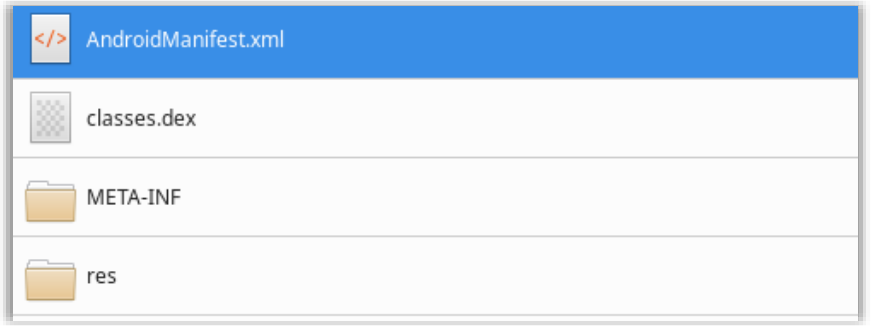
香港中文大學
The Chinese University of Hong Kong

32nd USENIX SECURITY SYMPOSIUM AUGUST 11, 2023

Android Manifest File

✪ Manifest File Functions

- Describe the essential configurations of an app.
- Highly relevant to multiple Android critical mechanisms.



Part composition of an APK file.

✪ Manifest File Composition

- A set of XML elements.
- An element contains multiple attributes.
- android:name attribute can be treated as the identifier.

```
<manifest ... >
    <activity
      android:name="com.example.TestActivity"
      android:exported="false">
    </activity>
  ...
</manifest>
```

Example of an element in a manifest file.

Manifest File-Related Security

✪ Correct Manifest Configuring (previous work)

- App developers' misconfigurations on manifest files will put apps at risk.

Declaring Duplicate Components / Misplacing Attributes / ··· ⇨ Component Protection Bypassing / App Defrauding / ···

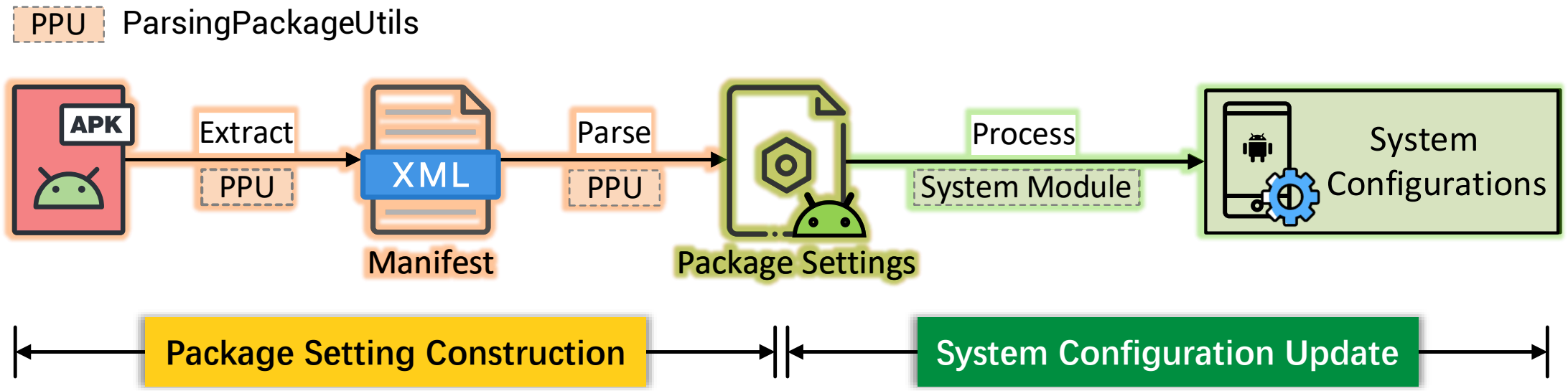
✪ Correct Manifest Data Processing (neglected by the security community)

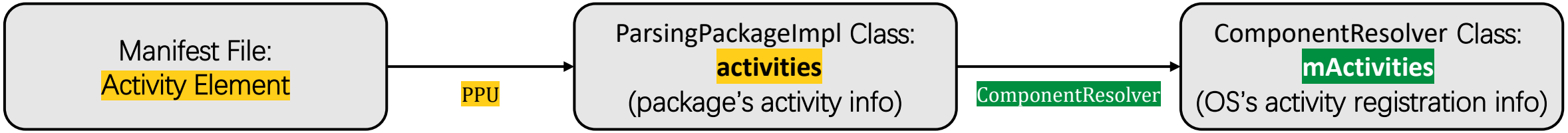
- Manifest files bridge Android apps and Android OS.

Manifest files are constructed by app developers. ⤮ Manifest files are related to multiple core mechanisms.

☹ Vulnerable manifest data processing procedures could be exploited to break the Android systems.

Manifest Processing Procedure (Android 11 & 12)





Threat Model

For Attacker

- Build a malicious app with a crafted manifest file.
- Release this app on various app markets.

For User

- Install this “apparently harmless” app on her phone.

This app exploits our discovered system vulnerabilities to conduct malicious actions (e.g., privilege escalation).

Motivation Case* – Permission Element

- Closely related to security – protect sensitive resources.
- Defined by system or third-party apps.

System Permission

Custom Permission

- Three main protection levels: normal, signature, dangerous.

  - Be granted or denied by the user when the app is running.
  - Be managed on a group basis.

- Request permissions through the <uses-permission> elements.

```
<manifest ... >
    <permission
      android:name="com.example.cp"
      android:protectionLevel="normal" />

    <uses-permission
       android:name = "com.example.cp" />
  ...
</manifest>
```

Define and request a custom permission.

Prompt of the runtime permission CALL_PHONE.

* Attack demo link: https://sites.google.com/view/eviltwins.

Motivation Case* – Privilege Escalation (Android 11)

- Crafted manifest file with twin permission declarations.

test

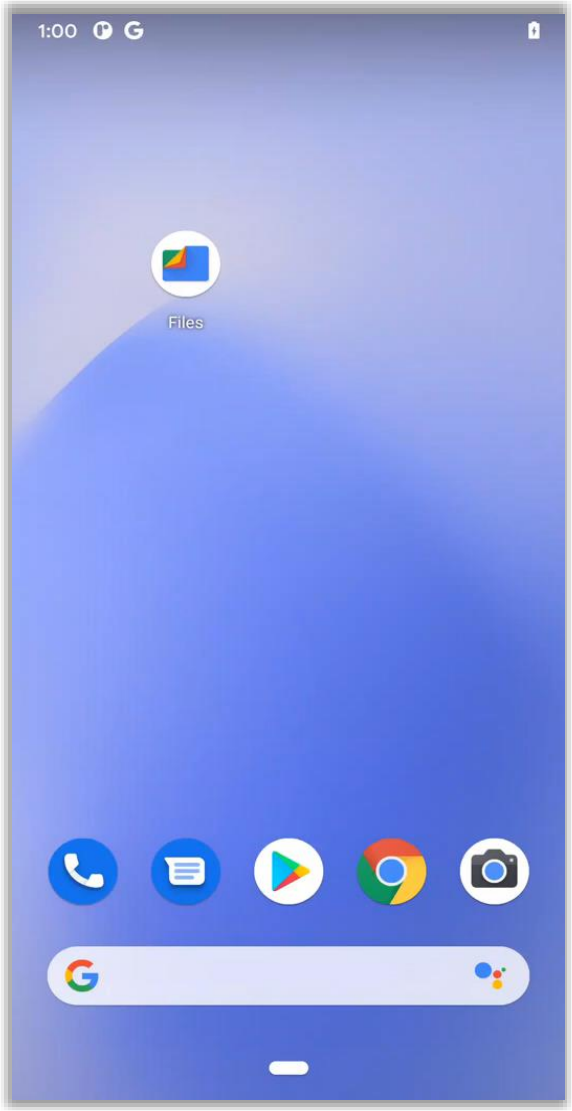
```
<manifest ... >
  <permission
    android:name="com.example.cp"
    android:protectionLevel="dangerous"
    android:permissionGroup="android.permission-group.PHONE" />
  <permission
   android:name="com.example.cp"
   android:protectionLevel="signature | development" />

  <uses-permission android:name="com.example.cp" />
  <uses-permission android:name="android.permission.CALL_PHONE" />
</manifest>
```

- This app can be built through app repackaging easily.

The app gets the dangerous system permission automatically without user consent. 😈

- High severity: CVE-2021-39695.



* Attack demo link: https://sites.google.com/view/eviltwins.

Motivation Case – Cause Analysis (Android 11)

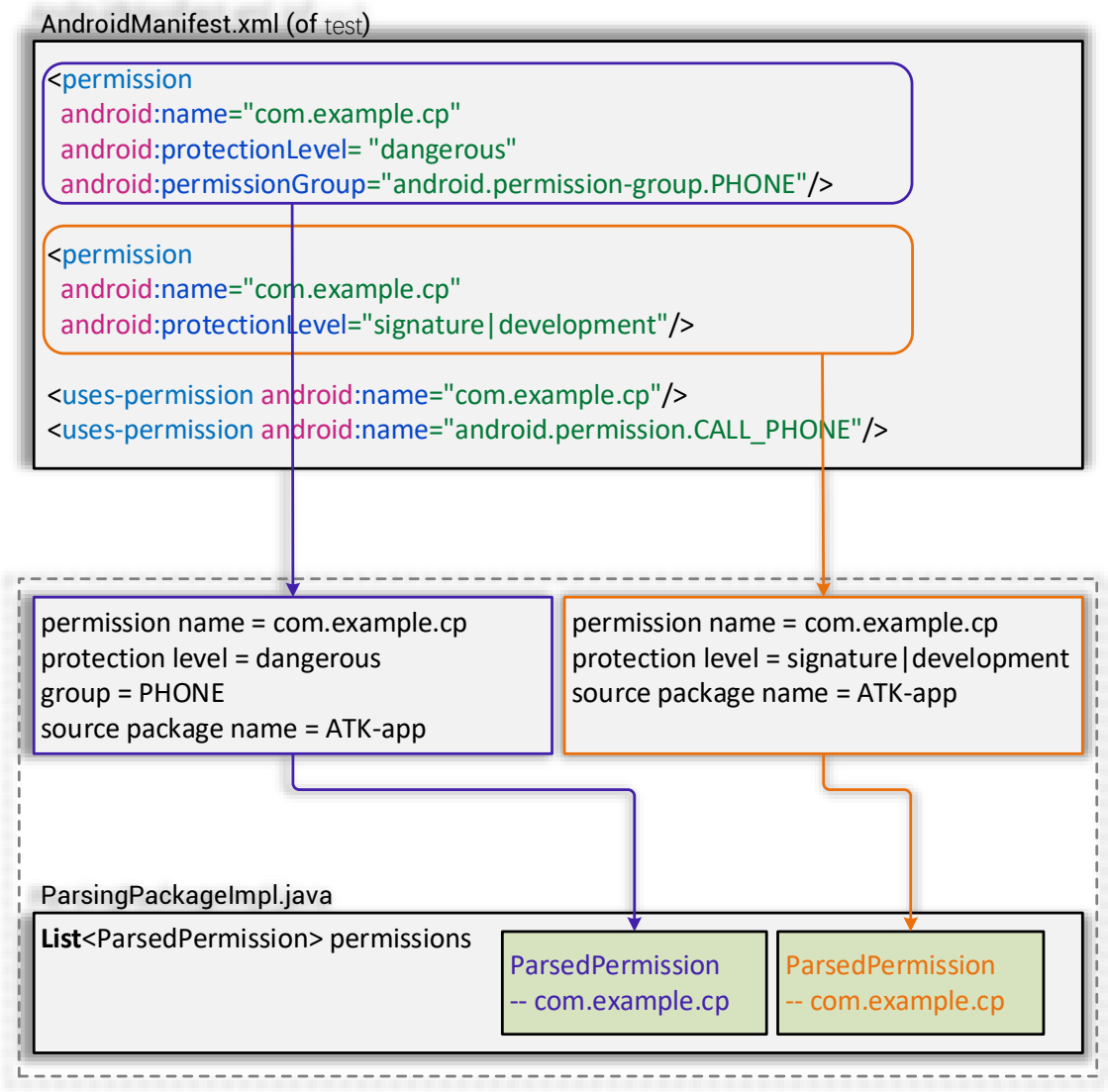
✪ Permission Element Parsing

- PPU* parses each permission element and stores the parsed data in **List<ParsedPermission> permissions**

Twin Permission Declarations

⬇

List<ParsedPermission> permissions



* The ParsingPackageUtils module.

Motivation Case – Cause Analysis (Android 11)

✪ Permission Registering

- Based on permissions, PMS* further updates **ArrayMap<permission-name, BasePermission> mPermissions**.

List<ParsedPermission> permissions

↓

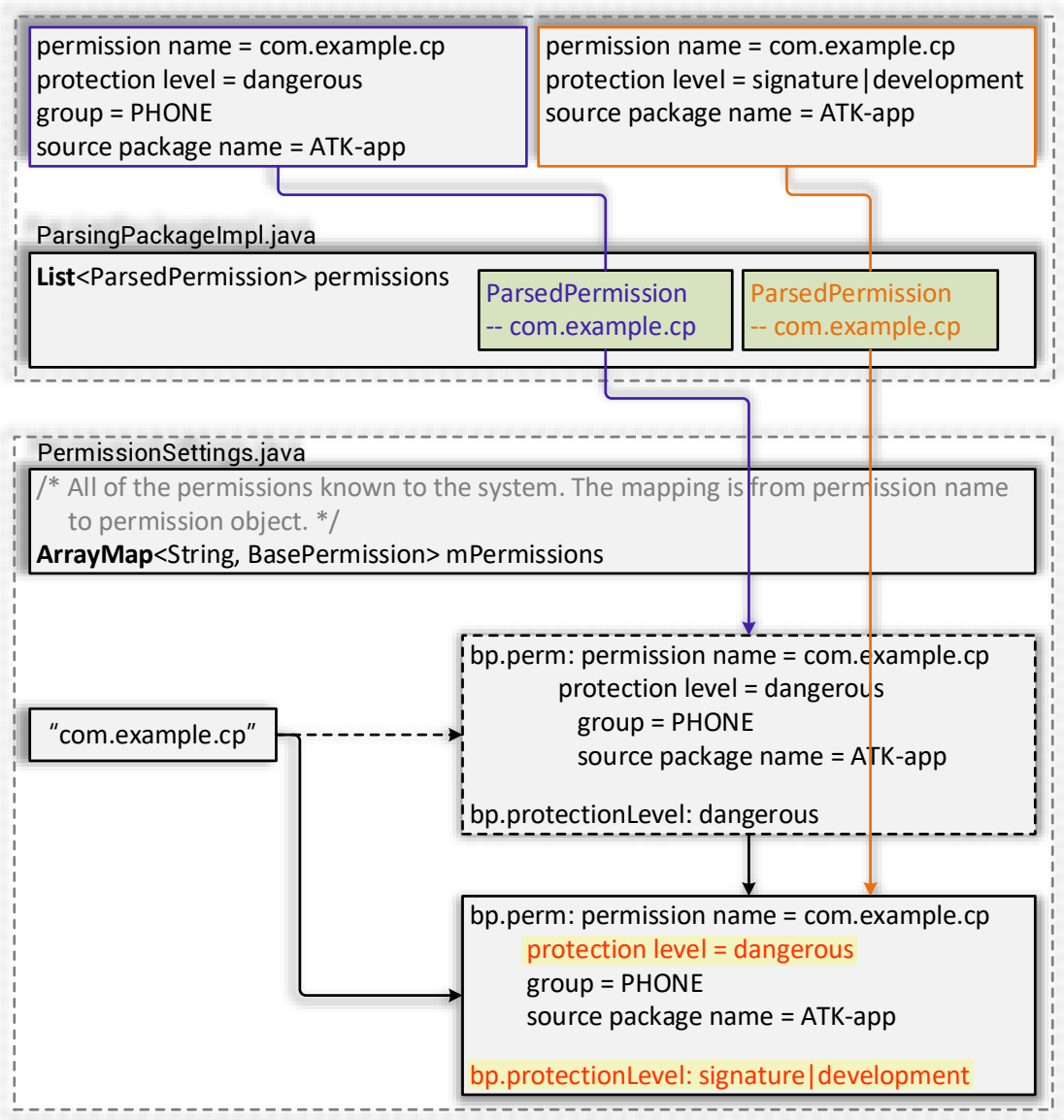
ArrayMap<String, BasePermission> mPermissions

↓

Protection level attributes are lost.

↓

Inconsistent permission protection levels.



* The PermissionManagerService module.

The Evil Twins Flaw

✪ Root Cause

During the processing of twin manifest elements, the ill-considered data structure conversion merges them into one item with attribute loss, further resulting in system configuration inconsistency.

- Twin elements: have the same identifier (e.g., name) but different attributes.

✪ Flaw Detection

- The manifest file is a practical attack surface & correlates to core mechanisms.

An automated tool to detect Evil Twins flaw-related vulnerabilities is needed.

TwinDroid* – Data Flow Analysis Tool

✪ Design Idea

Analyze Android OS's manifest processing procedure implementations to identify the data structure conversions with attribute loss.

- The output is a candidate set of suspicious methods causing attribute loss.
- Confirm vulnerabilities manually based on method info and the Android source code.

* Source code: https://github.com/little-leiry/TwinDroid.

TwinDroid* – Data Flow Analysis Tool

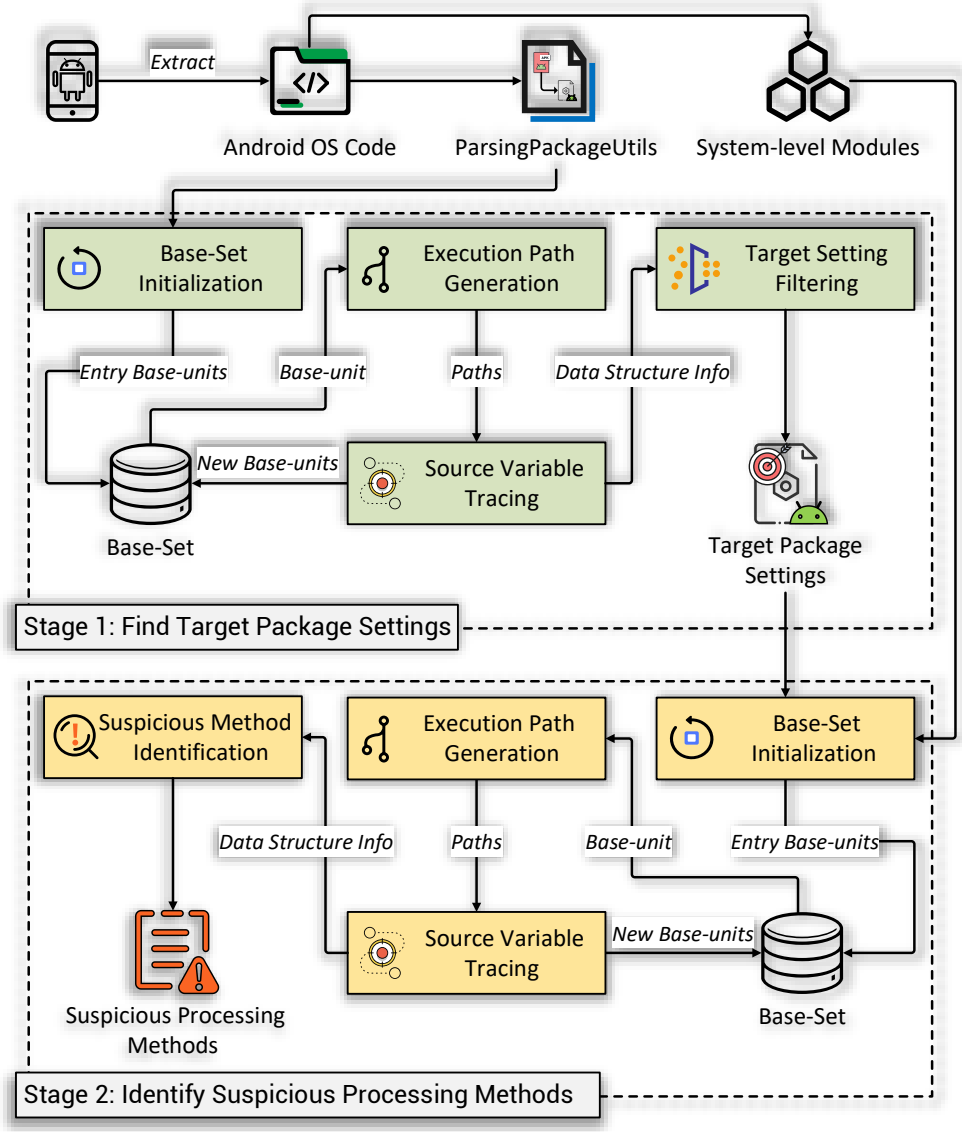
✪ Find Target Package Settings

- Find the package settings supporting duplicate elements, e.g., **List<ParsedPermission> permissions.**

✪ Identify Suspicious Processing Methods

- Identify the methods that 1) access the target package settings and 2) cause attribute loss due to data structure conversion.

Output: methods' names, associated element types, and saved data structures.



* Source code: https://github.com/little-leiry/TwinDroid.

Evaluation & Results

Implementation

- 6100 lines of Java code.
- Integrate Soot.
- Optimization strategies, e.g., path explosion avoiding & incorrect path pruning.

Execution Environment

- Intel Xeon Gold 6226R CPU @ 2.90GHz and 256G RAM.
- Ubuntu 20.04.3 LTS.

Evaluation & Results

✪ Result Overview

- Input: DEX files under the /system/framework directory from Android 11 & 12.
- Output: 47 suspicious methods, 6 of which are false positives.

8 Target Package Settings

41 Suspicious Methods

10 Element Types

4 Exploitable Inconsistencies

No.	Suspicious Processing Method	Target Package Setting	Element Type	Vul#	Ver.‡
1	addAllPermissions	List<ParsedPermission> permissions	permission, permission-tree	**1, 2**	11
2	addAllPermissionsInternal	List<ParsedPermission> permissions	permission, permission-tree	**2**	12
3	revokeRuntimePermissionsIfGroup-Changed	List<ParsedPermission> permissions	permission, permission-tree	**2**	11
4	revokeRuntimePermissionsIfGroup-ChangedInternal	List<ParsedPermission> permissions	permission, permission-tree	**2**	12
5	hasPermission	List<ParsedPermission> permissions	permission, permission-tree	**3, 4**	11, 12
6	addActivitiesLocked	List<ParsedActivity> activities	activity, activity-alias	bug	11, 12
7	queryIntentActivitiesInternalBody	List<ParsedActivity> activities	activity, activity-alias	bug	12
8	addReceiversLocked	List<ParsedActivity> receivers	receiver	bug	11, 12
9	addServicesLocked	List<ParsedService> services	service	bug	11, 12

Discovered suspicious processing methods* with security issues.

* Complete results are provided at https://github.com/little-leiry/TwinDroid/blob/main/Results.pdf.

Vulnerabilities*

Vulnerability	Severity	CVE	Exploit
1 - Break Permission Protection Levels	High	CVE-2021-39695	Permission Escalation
2 - Break Permission-Group Mapping	High	CVE-2022-20392	Permission Escalation
3 - Break Permission Registration Status	Low	N/A	Permission Revoking Prevention
4 - Break Permission Granting Status	Moderate	CVE-2023-20971	Permission Escalation

N/A: not assign a CVE ID.

- **Vul#1 and Vul#2 have been acknowledged by Samsung, Huawei, Honor, realme, and LG.**

* Attack demo link: https://sites.google.com/view/eviltwins.

Mitigation

✪ Lessons Learned

- For Google & OEM:

Avoid potential information loss during processing configuration data.

- For App Developers:

Avoid defining multiple elements with the same name, even for different element types.

One-page Takeaway

✪ Flaw Discovery

Discover a new category of vulnerabilities — the Evil Twins flaw.

✪ Flaw Detection

Develop an automated tool, TwinDroid, to detect the flaw-related vulnerabilities.

Identify a series of severe vulnerabilities in the real-world evaluation.

- Rui Li (Shandong University): leiry@mail.sdu.edu.cn

- TwinDroid: https://github.com/little-leiry/TwinDroid
- Attack demo link: https://sites.google.com/view/eviltwins