OVERSECURED

Discovering vendor-specific vulnerabilities in Android

About me

1. #1 hacker in Google Play Security Rewards Program
2. Discovered thousands of vulnerabilities in Android apps
3. First $1M at 23 y.o. from bug bounties
4. Automated the search of Android and iOS vulnerabilities
5. Founded Oversecured in 2020

Android. Does it exist?

Every vendor patches AOSP and adds many custom codes, custom system services, and preinstalled apps

Android is very fragmented

Your app will work on any Android version because of shared APIs, but the underground is much bigger

Why Samsung?

First time faced with Samsung VDP in 2021

- 18 vulnerabilities in preinstalled apps fixed
- Write-ups on blog.oversecured.com

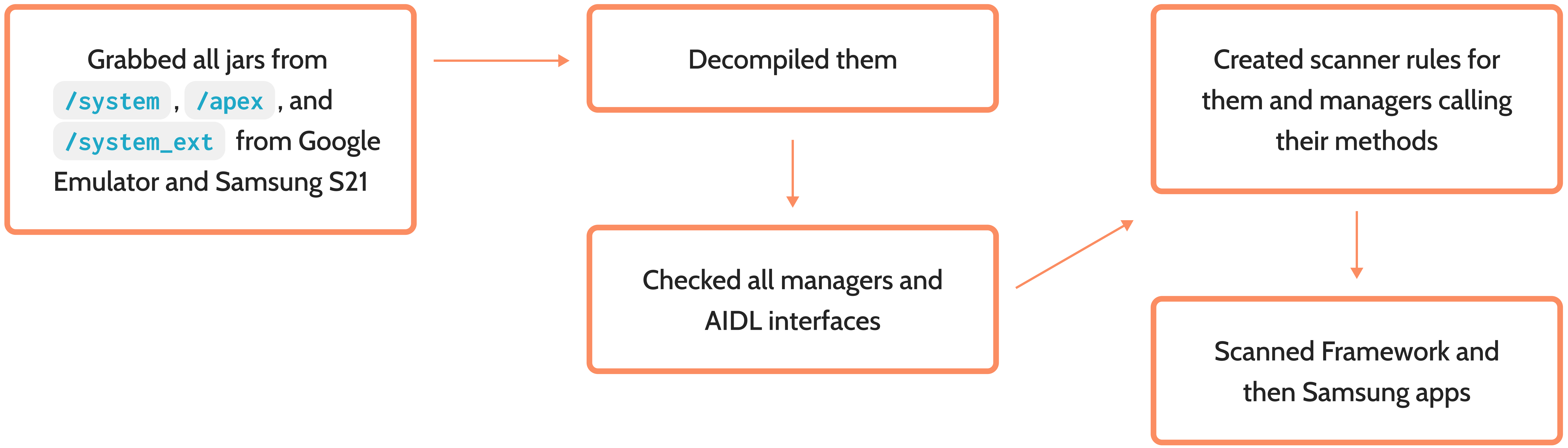
Why Samsung?

SAMSUNG

- Very nice bug bounty program
- Professional security team
- Same/bigger rewards compared to AOSP
- Huge scope containing hundreds of apps and patched Android Framework
- A few hackers focus on Samsung-ed Android
- Helping/protecting a lot more people than when you hack any other vendor

Methodology

Compared Samsung-ed and Googled Androids:



Grabbed all jars from `/system`, `/apex`, and `/system_ext` from Google Emulator and Samsung S21

Decompiled them

Checked all managers and AIDL interfaces

Created scanner rules for them and managers calling their methods

Scanned Framework and then Samsung apps

Methodology: AIDL interfaces

- Dumped all system services via

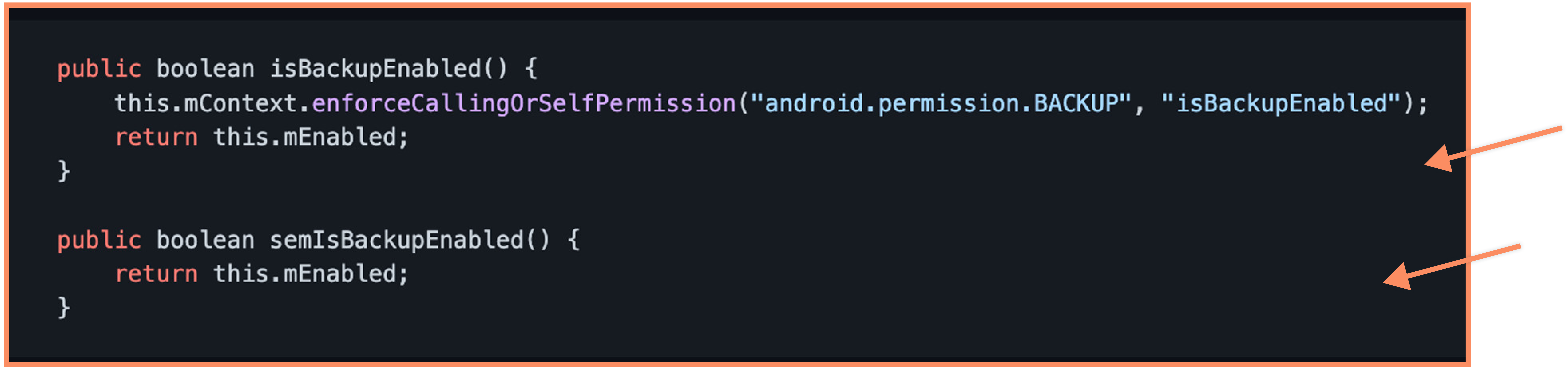
```
android.os.ServiceManager.listServices()
android.os.ServiceManager.getService(java.lang.String)
```

- Google: 221 services
- Samsung: 393 services

Samsung-owned AIDL interfaces

- Support huge Samsung ecosystem
- Interact with other Samsung devices such as S Pen and S Watch
- Used for custom device configuration (Samsung has much more features than AOSP)

AIDL interfaces: Example



```java
public boolean isBackupEnabled() {
    this.mContext.enforceCallingOrSelfPermission("android.permission.BACKUP", "isBackupEnabled");
    return this.mEnabled;
}

public boolean semIsBackupEnabled() {
    return this.mEnabled;
}
```

Invented new kind of code analysis

Control-flow:

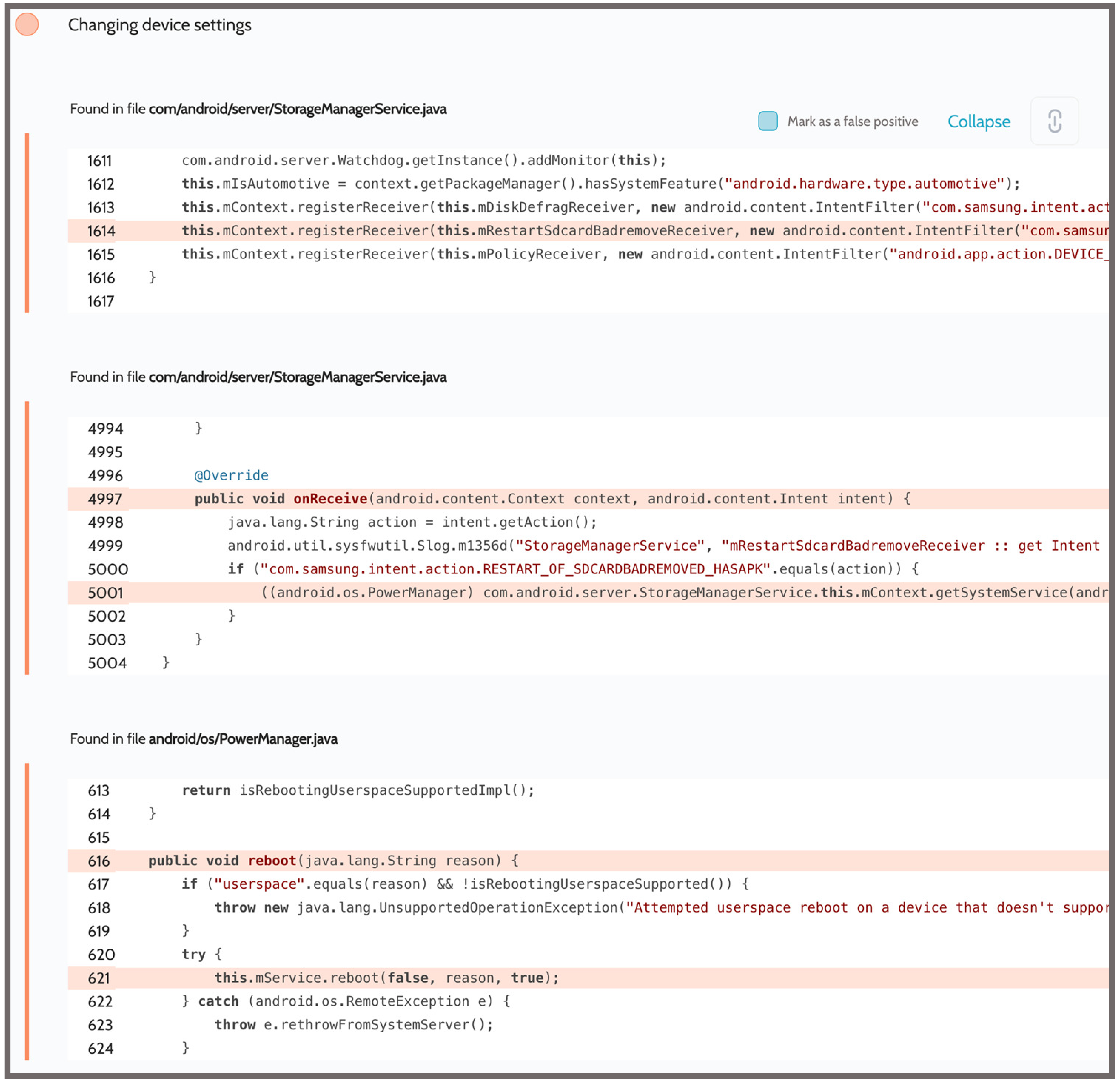
- Tracks data from exported component
- Disallows any user interaction
- Checks if the attacker can execute dangerous code via the privileged app

Taint-to-control-flow:

- The beginning for a control-flow matching is a taint event
- Currently, tracks only `registerReceiver()` -> `onReceive()` for dynamically registered broadcast receivers

Taint-to-control-flow: Example

- Impact: rebooting the device w/o any permission



Changing device settings

Found in file **com/android/server/StorageManagerService.java**

```
        com.android.server.Watchdog.getInstance().addMonitor(this);
        this.mIsAutomotive = context.getPackageManager().hasSystemFeature("android.hardware.type.automotive");
        this.mContext.registerReceiver(this.mDiskDefragReceiver, new android.content.IntentFilter("com.samsung.intent.act
        this.mContext.registerReceiver(this.mRestartSdcardBadremoveReceiver, new android.content.IntentFilter("com.samsun
        this.mContext.registerReceiver(this.mPolicyReceiver, new android.content.IntentFilter("android.app.action.DEVICE_
    }

```

Found in file **com/android/server/StorageManagerService.java**

```
        }

        @Override
        public void onReceive(android.content.Context context, android.content.Intent intent) {
            java.lang.String action = intent.getAction();
            android.util.sysfwutil.Slog.m1356d("StorageManagerService", "mRestartSdcardBadremoveReceiver :: get Intent
            if ("com.samsung.intent.action.RESTART_OF_SDCARDBADREMOVED_HASAPK".equals(action)) {
                ((android.os.PowerManager) com.android.server.StorageManagerService.this.mContext.getSystemService(andr
            }
        }
    }
```

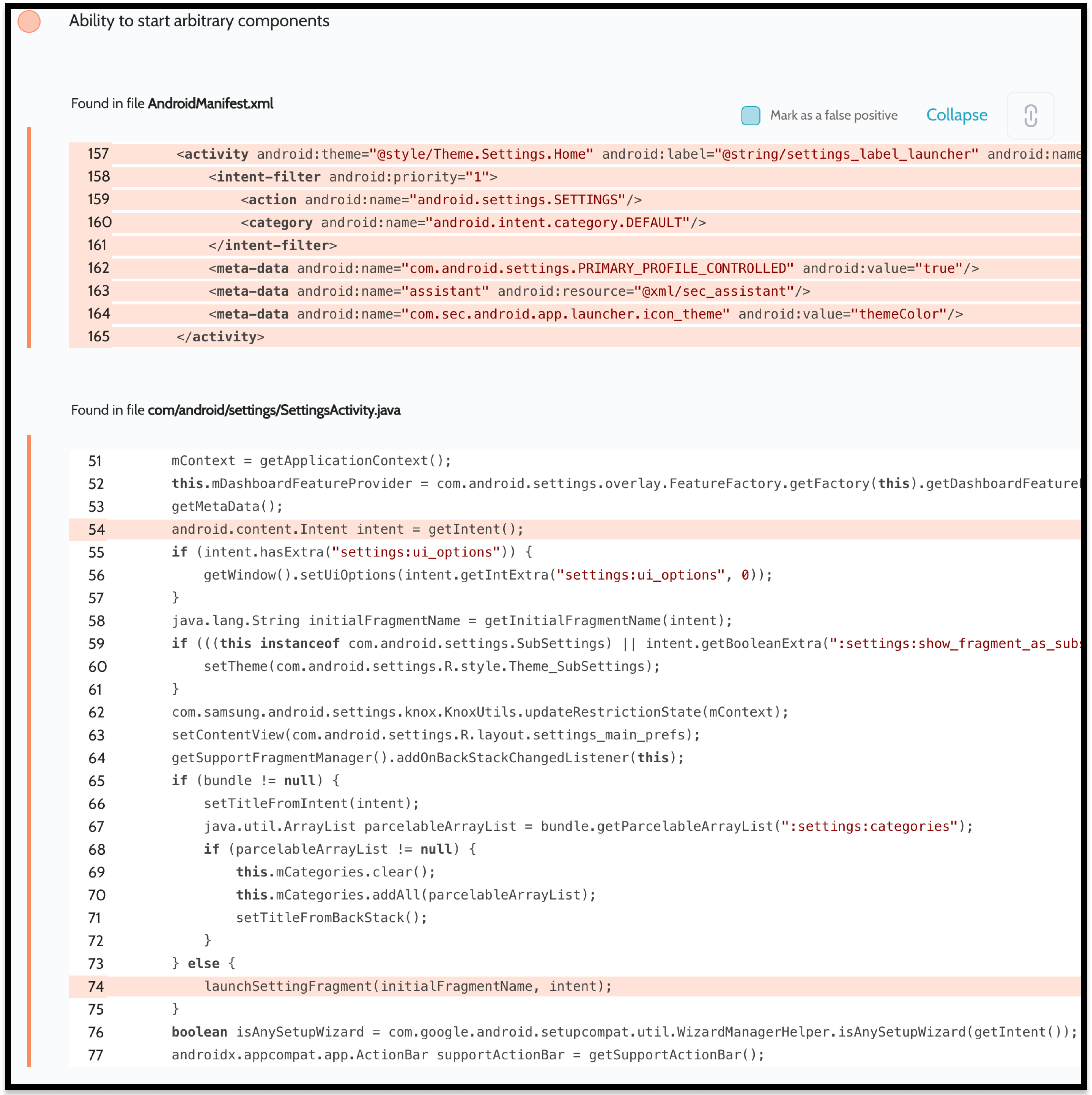
Found in file **android/os/PowerManager.java**

```
        return isRebootingUserspaceSupportedImpl();
    }

    public void reboot(java.lang.String reason) {
        if ("userspace".equals(reason) && !isRebootingUserspaceSupported()) {
            throw new java.lang.UnsupportedOperationException("Attempted userspace reboot on a device that doesn't suppor
        }
        try {
            this.mService.reboot(false, reason, true);
        } catch (android.os.RemoteException e) {
            throw e.rethrowFromSystemServer();
        }
```

System apps are also affected

- Settings app (`com.android.settings`)
- Known as LaunchAnyWhere


Ability to start arbitrary components

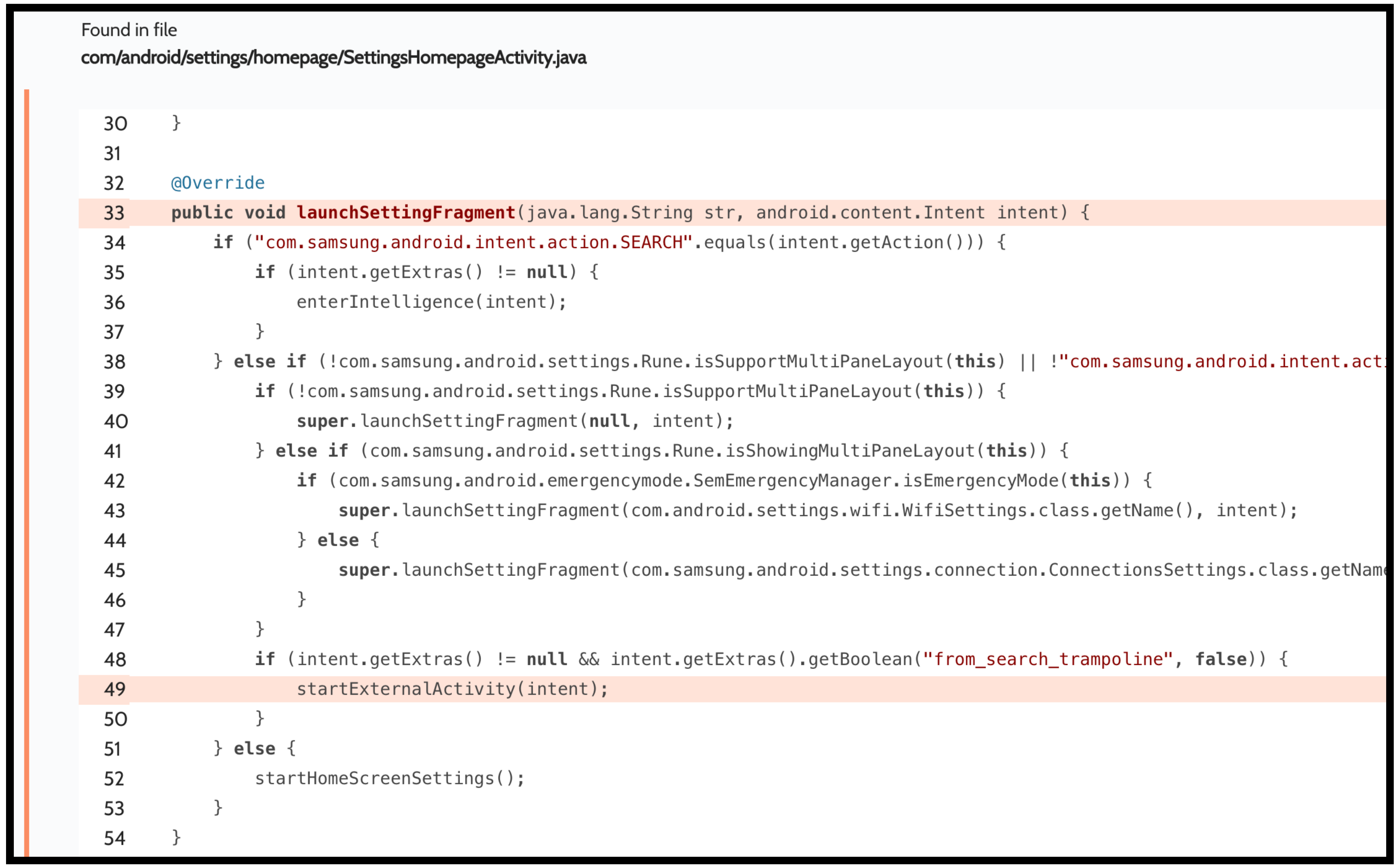
Found in file AndroidManifest.xml

```
    <activity android:theme="@style/Theme.Settings.Home" android:label="@string/settings_label_launcher" android:name
        <intent-filter android:priority="1">
            <action android:name="android.settings.SETTINGS"/>
            <category android:name="android.intent.category.DEFAULT"/>
        </intent-filter>
        <meta-data android:name="com.android.settings.PRIMARY_PROFILE_CONTROLLED" android:value="true"/>
        <meta-data android:name="assistant" android:resource="@xml/sec_assistant"/>
        <meta-data android:name="com.sec.android.app.launcher.icon_theme" android:value="themeColor"/>
    </activity>
```

Found in file com/android/settings/SettingsActivity.java

```
        mContext = getApplicationContext();
        this.mDashboardFeatureProvider = com.android.settings.overlay.FeatureFactory.getFactory(this).getDashboardFeature
        getMetaData();
        android.content.Intent intent = getIntent();
        if (intent.hasExtra("settings:ui_options")) {
            getWindow().setUiOptions(intent.getIntExtra("settings:ui_options", 0));
        }
        java.lang.String initialFragmentName = getInitialFragmentName(intent);
        if (((this instanceof com.android.settings.SubSettings) || intent.getBooleanExtra(":settings:show_fragment_as_sub
            setTheme(com.android.settings.R.style.Theme_SubSettings);
        }
        com.samsung.android.settings.knox.KnoxUtils.updateRestrictionState(mContext);
        setContentView(com.android.settings.R.layout.settings_main_prefs);
        getSupportFragmentManager().addOnBackStackChangedListener(this);
        if (bundle != null) {
            setTitleFromIntent(intent);
            java.util.ArrayList parcelableArrayList = bundle.getParcelableArrayList(":settings:categories");
            if (parcelableArrayList != null) {
                this.mCategories.clear();
                this.mCategories.addAll(parcelableArrayList);
                setTitleFromBackStack();
            }
        } else {
            launchSettingFragment(initialFragmentName, intent);
        }
        boolean isAnySetupWizard = com.google.android.setupcompat.util.WizardManagerHelper.isAnySetupWizard(getIntent());
        androidx.appcompat.app.ActionBar supportActionBar = getSupportActionBar();
```

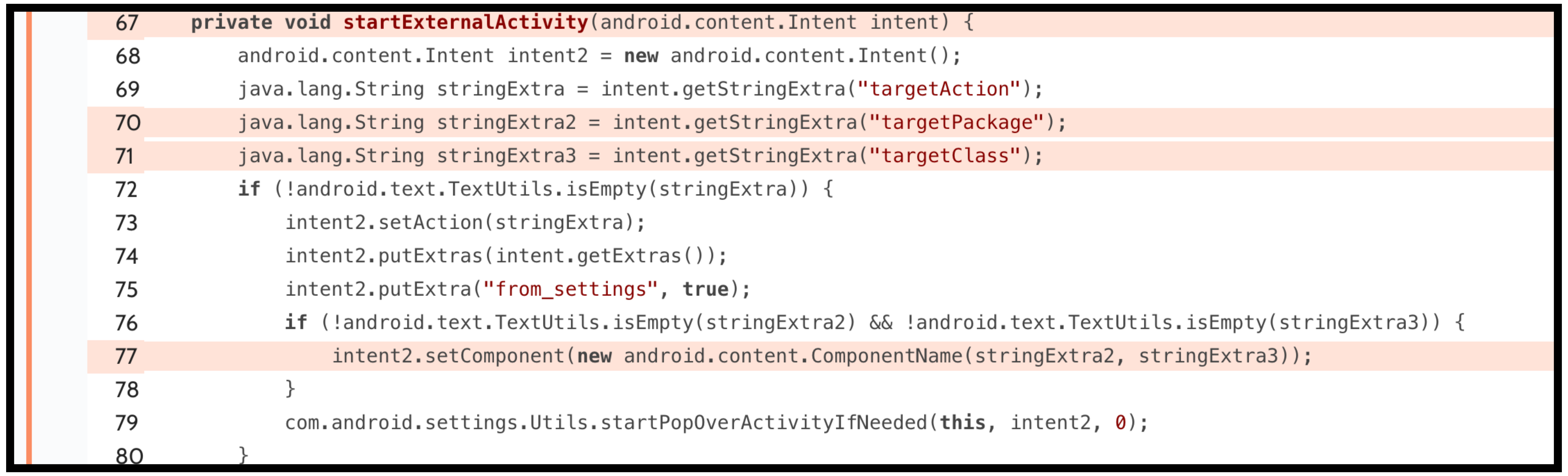
System apps are also affected



Found in file
com/android/settings/homepage/SettingsHomepageActivity.java

```java
}

@Override
public void launchSettingFragment(java.lang.String str, android.content.Intent intent) {
    if ("com.samsung.android.intent.action.SEARCH".equals(intent.getAction())) {
        if (intent.getExtras() != null) {
            enterIntelligence(intent);
        }
    } else if (!com.samsung.android.settings.Rune.isSupportMultiPaneLayout(this) || !"com.samsung.android.intent.act
        if (!com.samsung.android.settings.Rune.isSupportMultiPaneLayout(this)) {
            super.launchSettingFragment(null, intent);
        } else if (com.samsung.android.settings.Rune.isShowingMultiPaneLayout(this)) {
            if (com.samsung.android.emergencymode.SemEmergencyManager.isEmergencyMode(this)) {
                super.launchSettingFragment(com.android.settings.wifi.WifiSettings.class.getName(), intent);
            } else {
                super.launchSettingFragment(com.samsung.android.settings.connection.ConnectionsSettings.class.getName
            }
        }
        if (intent.getExtras() != null && intent.getExtras().getBoolean("from_search_trampoline", false)) {
            startExternalActivity(intent);
        }
    } else {
        startHomeScreenSettings();
    }
}
```



```java
private void startExternalActivity(android.content.Intent intent) {
    android.content.Intent intent2 = new android.content.Intent();
    java.lang.String stringExtra = intent.getStringExtra("targetAction");
    java.lang.String stringExtra2 = intent.getStringExtra("targetPackage");
    java.lang.String stringExtra3 = intent.getStringExtra("targetClass");
    if (!android.text.TextUtils.isEmpty(stringExtra)) {
        intent2.setAction(stringExtra);
        intent2.putExtras(intent.getExtras());
        intent2.putExtra("from_settings", true);
        if (!android.text.TextUtils.isEmpty(stringExtra2) && !android.text.TextUtils.isEmpty(stringExtra3)) {
            intent2.setComponent(new android.content.ComponentName(stringExtra2, stringExtra3));
        }
        com.android.settings.Utils.startPopOverActivityIfNeeded(this, intent2, 0);
    }
```

Top vulnerability types discovered

- Unprotected AIDL interface methods
- Custom broadcast actions without `protected-broadcast` protection
- Privileged apps insecurely using Samsung-owned code
- Access-control issues to components (services, receivers, activities, etc) of Samsung apps

Results?

2021:

CVE-2021-25388, CVE-2021-25356, CVE-2021-25391, CVE-2021-25393, CVE-2021-25392, CVE-2021-25397, CVE-2021-25390, CVE-2021-25426, CVE-2021-25410, CVE-2021-25413, CVE-2021-25414, CVE-2021-25440, CVE-2021-25514, CVE-2021-25377, CVE-2021-25379, CVE-2021-25400, CVE-2021-25401, CVE-2021-25404

2022:

CVE-2022-28781, CVE-2022-28783, CVE-2022-28784, CVE-2022-30727, CVE-2022-30754, CVE-2022-33689, CVE-2022-33690, CVE-2022-33694, CVE-2022-33726, CVE-2022-33722, CVE-2022-33721, CVE-2022-33732, CVE-2022-33731, CVE-2022-33715, CVE-2022-33725, CVE-2022-36852, CVE-2022-36853, CVE-2022-36850, CVE-2022-24003, CVE-2022-28544, CVE-2022-28790, CVE-2022-30745, CVE-2022-30746, CVE-2022-30747, CVE-2022-30748, CVE-2022-33705, CVE-2022-33713, CVE-2022-33710, CVE-2022-33709, CVE-2022-33708, CVE-2022-36835, CVE-2022-36839, CVE-2022-36832, CVE-2022-36833, CVE-2022-36834, CVE-2022-36836, CVE-2022-36830, CVE-2022-36829, CVE-2022-33734, CVE-2022-33733, CVE-2022-36837, CVE-2022-36838, CVE-2022-36831, CVE-2022-36865, CVE-2022-36866, CVE-2022-36867, CVE-2022-36872, CVE-2022-36871, CVE-2022-36870, CVE-2022-39858, CVE-2022-39859, CVE-2022-39860, CVE-2022-39861, CVE-2022-39863, CVE-2022-39864, CVE-2022-39871, CVE-2022-39870, CVE-2022-39869, CVE-2022-39868, CVE-2022-39867, CVE-2022-39866, CVE-2022-39865

OVERSECURED

THANK YOU!

QUESTIONS?