



**black hat**<sup>®</sup>  
EUROPE 2021

november 10-11, 2021

---

BRIEFINGS

# **A Deep Dive into Privacy Dashboard of Top Android Vendors**

Bin Ma, Zhenyu Zhu, Wei Wen, Xiangxing Qian

## About us



Bin Ma



Zhenyu Zhu



Wei Wen



Xiangxing Qian

- Security researcher at IES Red Team of ByteDance
- Privacy protection research involve Apps and Vendors
- Applied security including Mobile, Web and Cloud

# Outline

- How Privacy Dashboard Works
- Android AppOps Introduction
- Details and Flaws in Vendors' Privacy Dashboards
- Measurement and Test Methods
- Conclusion

# 1. How Privacy Dashboard Works

## Why do we need privacy dashboard ?

- **What can Apps do to invade user privacy ?**

### **Data access behavior**

- Steal Contacts
- Scan Media Files
- Read personal data
- ... ..

### **Data tamper behavior**

- Write to Calendar
- Rewrite Clipboard
- Delete Media File
- ... ..

### **Sensitive Action**

- Hidden camera
- Unauthorized recording

### **Startup Action**

- AutoStart
- Chain Start

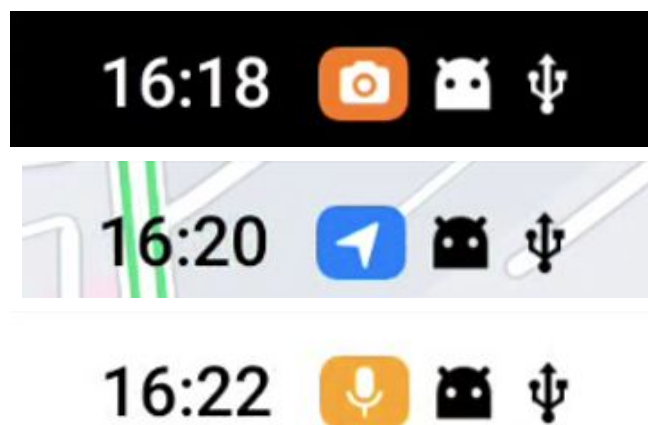
- **What can Operating System do to protect users privacy ?**

- Restrict the system API.
- Privacy dashboard.

# What is privacy dashboard ?

A permission record center that allows users to see how Apps accessing data.

- Monitor sensitive behaviors of a specific app
- Shows the timeline of a certain permission used by different apps
- Prompt some sensitive access in real time



A vertical timeline showing app permissions from 16:49 to 16:48. The events are:

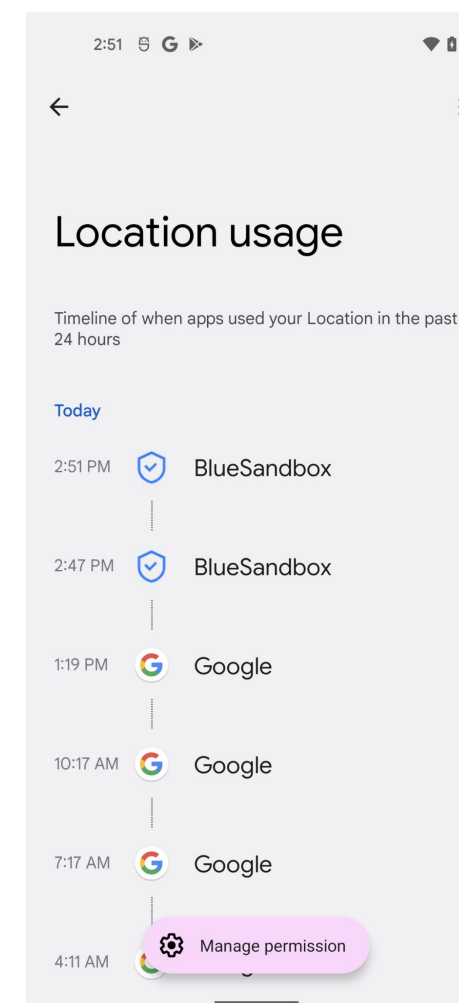
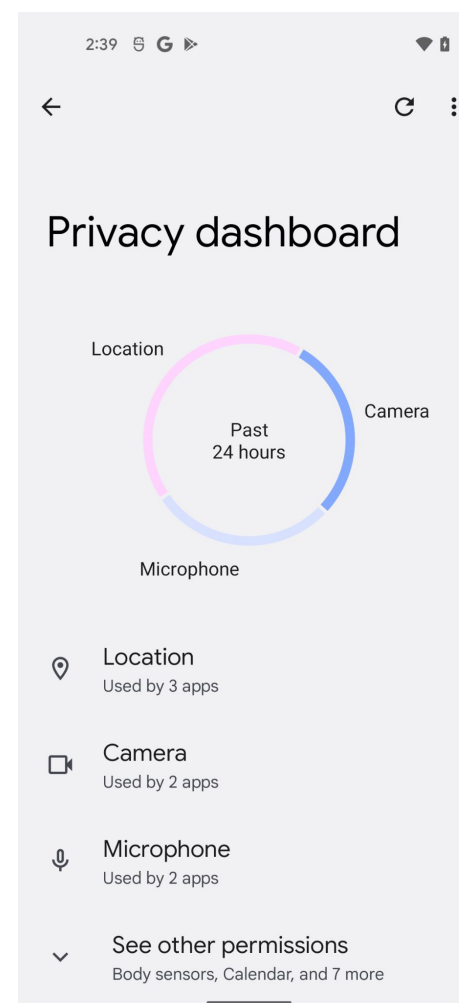
- 16:49: This app accessed your location (Allowed at 16:49)
- 16:48: This app copied items to the clipboard (Restricted at 16:48)
- 16:48: This app accessed items in the clipboard (Restricted at 16:48)
- 16:48: This app took photos or recorded video (Allowed 4 times at 16:48)
- 16:48: This app was launched from Home screen (Allowed at 16:48)

A screenshot of the 'Application behavior record' interface. It shows a list of permissions with filters for 'All', 'Startup behavior', and 'Sensitive p'. The list includes:

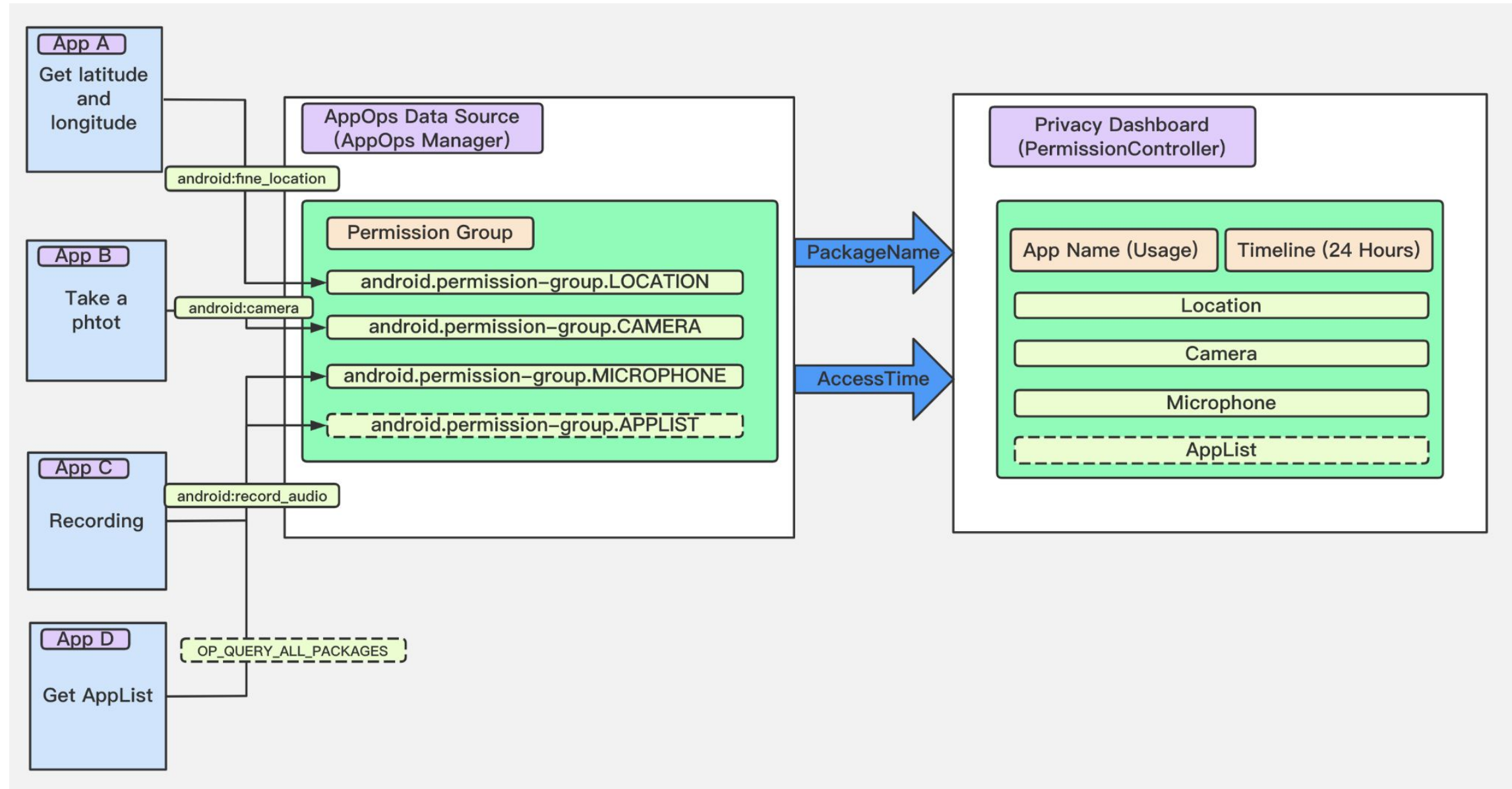
- nowadays
- Calendar (Fri 24)
- 16:30: Read calendar (Allow 1 Time)
- BlueSandbox
- 16:29: Get location information (Allow 5 Time)
- 16:25: Modify contact information (Allow 3 Time)
- Camera
- 16:24: Get location information (Allow 4 Time)
- In the background Take photos or videos (Allow 1 Time)
- Take photos or videos (Allow 1 Time)

## Privacy Dashboard & Android 12

- Monitor permissions
  - Location
  - Microphone
  - Camera
- Shows the Apps access to these permission
- Shows the timeline and frequency of these access.



# How do these privacy dashboards work ?





## 2. Android AppOps Introduction

# What is AppOps?

## Introduction

- Formally added in Android 4.3
- Unit - uid/package
- Access control and tracking
  - Permission
  - Operation

## Operations vs Permissions

- 53 permissions but 105 ops
- Clipboard and so on

```
platform/superproject > android-4.3_r0.9 > frameworks/base/core/java/android/app/AppOpsManager.java
I> AppOpsManager.java
33 /**
34  * API for interacting with "application operation" tracking. Allows you to:
35  *
36  * - Note when operations are happening, and find out if they are allowed for the current caller.
37  * - Disallow specific apps from doing specific operations.
38  * - Collect all of the current information about operations that have been executed or are not
39  *   being allowed.
40  * - Monitor for changes in whether an operation is allowed.
41  *
42  * Each operation is identified by a unique ID, and each operation is identified by the
43  * operations, enumerated by the AppOpsManager class.
44  *
45  * When checking operations, the caller must provide the operation ID and the package name
46  * for the operation under that ID. The caller must also provide the package name
47  * of the app that is performing the operation. If the caller does not provide the
48  * SecurityException back to the caller.
49  *
50  * @hide
51  */
52 public class AppOpsManager {
53     final Context mContext;
54     final IAppOpsService mService;
55     final HashMap<Callback, IAppOpsService> mCallbacks;
56     = new HashMap<Callback, IAppOpsService>();
57
58     public static final int MODE_ALLOWED = 0;
59     public static final int MODE_DENIED = 1;
60     public static final int MODE_IGNORED = 2;
}

platform/superproject > master > frameworks/base/core/java/android/app/AppOpsManager.java
I> AppOpsManager.java
1925 /**
1926  * This optionally maps a permission to an operation. If there
1927  * is no permission associated with an operation, it is null.
1928  */
1929 @UnsupportedAppUsage
1930 private static String[] sOpPerms = new String[] {
1931     android.Manifest.permission.ACCESS_COARSE_LOCATION,
1932     android.Manifest.permission.ACCESS_FINE_LOCATION,
1933     null,
1934     android.Manifest.permission.VIBRATE,
1935     android.Manifest.permission.READ_CONTACTS,
1936     android.Manifest.permission.WRITE_CONTACTS,
1937     android.Manifest.permission.READ_CALL_LOG,
1938     android.Manifest.permission.WRITE_CALL_LOG,
1939     android.Manifest.permission.READ_CALENDAR,
1940     android.Manifest.permission.WRITE_CALENDAR,
1941     android.Manifest.permission.ACCESS_WIFI_STATE,
1942     null, // no permission required for notifications
1943     null, // neighboring cells shares the coarse location perm
}
```

# Work mechanism and flow of AppOps

## Check Permission

- Befor check op
- CheckPermission()

## General

- Manager/Service

AppOpsManager API

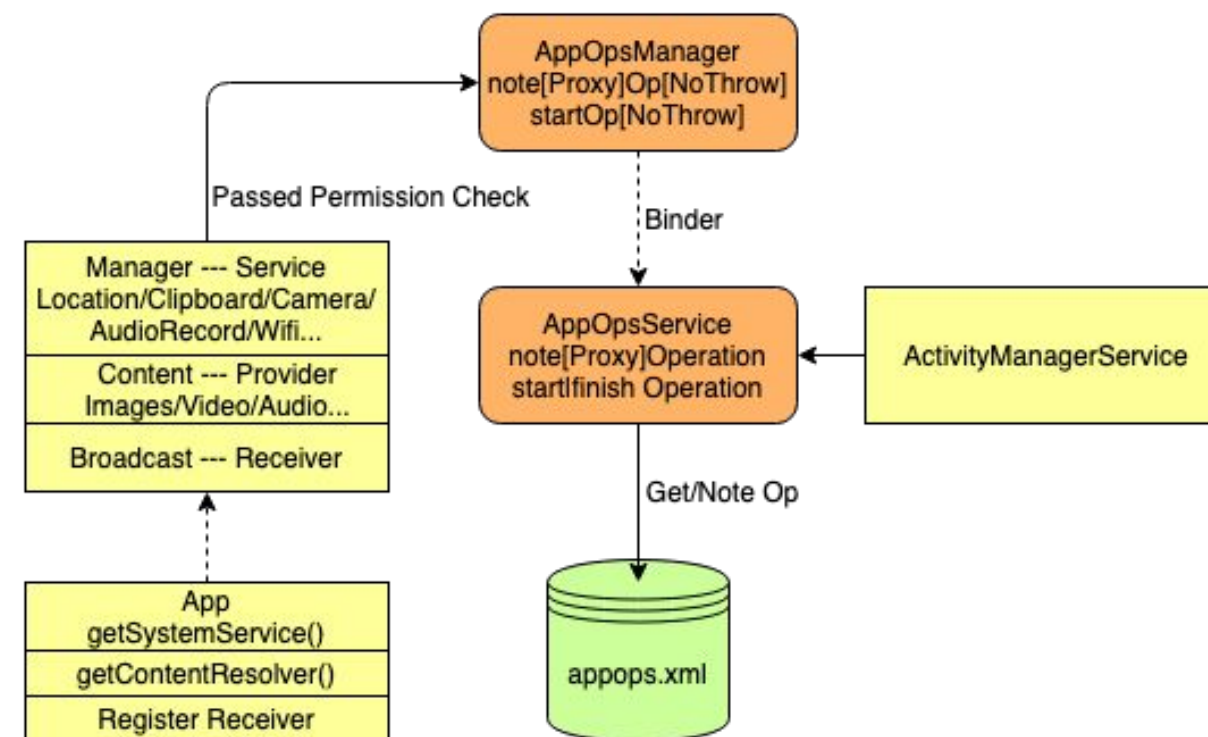
## Check and Note op

- note[Proxy]Op[NoThrow]
- startOp[NoThrow]

## Exception

- AMS

AppOpsService API



# AppOps data dump

- dumphsys appops or appops dump
  - -- package
- record detail
  - Op String
  - Allow/default/ignore...
  - Access or reject time
  - Background or foreground
  - Proxy or not
  - Duration time

```
~> adb shell appops dump | more
Current AppOps Service state:
Settings:
  top_state_settle_time=+30s0ms
  fg_service_state_settle_time=+10s0ms
  bg_state_settle_time=+1s0ms

Op mode watchers:
Op COARSE_LOCATION:
  #0: ModeCallback{11
  #1: ModeCallback{5a
  #2: ModeCallback{96
Op READ_CALL_LOG:
  #0: ModeCallback{8a
Op WRITE_CALL_LOG:
  #0: ModeCallback{8a
Op READ_SMS:
  #0: ModeCallback{8a

Package com.android.settings:
  VIBRATE (allow):
    Access: [pers-s] 2021-09-23 18:06:42.068 (-2d22h25m45s342ms) duration=+21ms proxy[uid=0, pkg=null]
  WRITE_SETTINGS (default):
    Reject: [pers-s]2021-09-26 16:03:03.710 (-29m23s700ms) proxy[uid=0, pkg=null]
  CAMERA (allow):
    Access: [pers-s] 2021-06-20 20:04:00.140 (-97d20h28m27s270ms) duration=+10s647ms proxy[uid=0, pkg=null]
  READ_CLIPBOARD (allow):
    Access: [pers-s] 2020-11-10 21:46:15.079 (-319d18h46m12s331ms) proxy[uid=0, pkg=null]
  WRITE_CLIPBOARD (allow):
    Access: [pers-s] 2021-06-01 15:13:05.613 (-117d1h19m21s797ms) proxy[uid=0, pkg=null]
  GET_USAGE_STATS (default):
    Reject: [pers-s]2021-09-26 12:14:16.780 (-4h18m10s630ms) proxy[uid=0, pkg=null]
  TOAST_WINDOW (allow):
    Access: [pers-s] 2021-08-18 12:11:42.273 (-39d4h20m45s137ms) duration=+2s543ms proxy[uid=0, pkg=null]
  WRITE_EXTERNAL_STORAGE (allow):
    Access: [pers-s] 2021-09-17 11:16:22.184 (-9d5h16m5s226ms) proxy[uid=0, pkg=null]
  REQUEST_DELETE_PACKAGES (allow):
    Access: [pers-s] 2021-06-08 12:10:37.176 (-110d4h21m50s234ms) proxy[uid=0, pkg=null]
Package com.google.android.hiddenmenu:
  WRITE_SETTINGS (default):
    Reject: [pers-s]2021-01-22 14:38:34.348 (-247d1h53m53s62ms) proxy[uid=0, pkg=null]
  REQUEST_DELETE_PACKAGES (allow):
    Access: [pers-s] 2021-08-13 16:18:48.926 (-44d0h13m38s484ms) proxy[uid=0, pkg=null]
Package com.android.wallpaperbackup:
  REQUEST_DELETE_PACKAGES (allow):
    Access: [pers-s] 2021-08-13 17:18:03.455 (-43d23h14m23s955ms) proxy[uid=0, pkg=null]
```

# Interface for AppOps data extraction

## Apps Developers

- **How**

- AppOpsManager.OnOpNotedCallback
  - logPrivateDataAccess
  - onNoted/onSelfNoted/onAsyncNoted
- AppOpsManager.setOnOpNotedCallback

- **When**

- Single Activity onCreate()
- Application attachBaseContext()

## Security Tester

- **Static inject**

- App -- modify bytecode and repack

- **Dynamic inject**

- App -- inject \*OnOpNotedCallback
- system\_server -- hook noteOperation, etc.

- **Custom rom**

- How Vendors do?

# 3. Privacy Dashboards of Vendors

## Privacy dashboard technique of custom vendors

- Extract AppOps records purely
  - accept/reject
  - access time
  - background/foreground
  - duration
- AppOps modification
  - self-designed opcode
  - noteOperation/noteProxyOperation
- Custom implementation
  - empty or temporary privacy data
  - album delete protection

# Privacy Dashboard Overview of Android vendors

Privacy Dashboard		A	B	C	D	E
Privacy Data Collection	Location	TBI	TBI	TBI	TBI	✓
	Camera	✓	✓	✓	✓	✓
	Microphone	✓	✓	✓	✓	✓
	Contacts	✓	✓	✓	✓	✓
	Message	✓	✓	✓	✓	✓
	Clipboard	✓	TBI	✓	✗	✗
	Device Info	TBI	✓	✓	✓	✗
	External Storage	✓	✓	✓	✓	✓
	App List	✗	✗	✓	✓	✗
Background Behavior		✓	✓	✗	✗	✗
Frequency		✓	TBI	✗	✗	TBI
Album Protection		TBI	TBI	✓	TBI	TBI

✓ Support

✗ Not Support

TBI To be Improved, Partially Support or Inaccurate



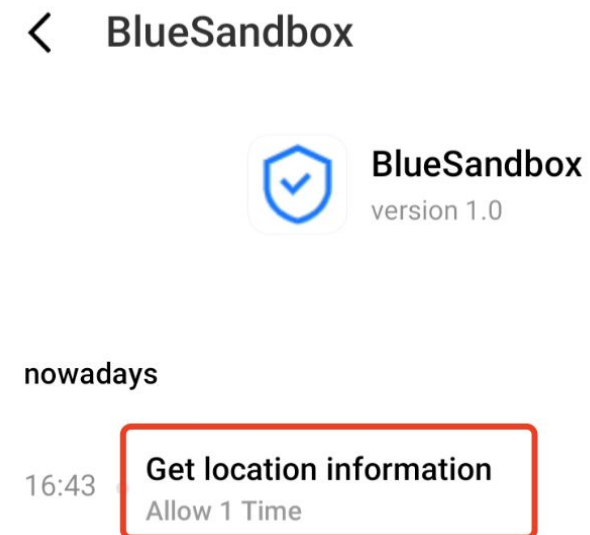
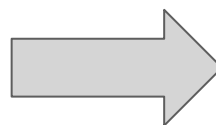
## Case 1: Get network signal but regard as location collection

```
//AndroidManifest
<uses-permission
android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>

//Code Snippet
WifiManager wifiManager = (WifiManager)
context.getSystemService(context.WIFI_SERVICE);

WifiInfo wifiInfo = wifiManager.getConnectionInfo();

int rssi = wifiInfo.getRssi();
```



Privacy Dashboard result of Vendor C

App call getConnectionInfo to get RSSI signal

## Case 1: Get network signal but regard as location collection

- Call Stack

WifiManager.getConnectionInfo()

WifiServiceImpl.getConnectionInfo()

WifiPermissionsUtil.enforceCanAccessScanResults()

WifiPermissionsUtil.checkCallersLocationPermission()

WifiPermissionsUtil.noteAppOpAllowed()

AppOpsManager.noteOp(OPSTR\_FINE\_LOCATION)

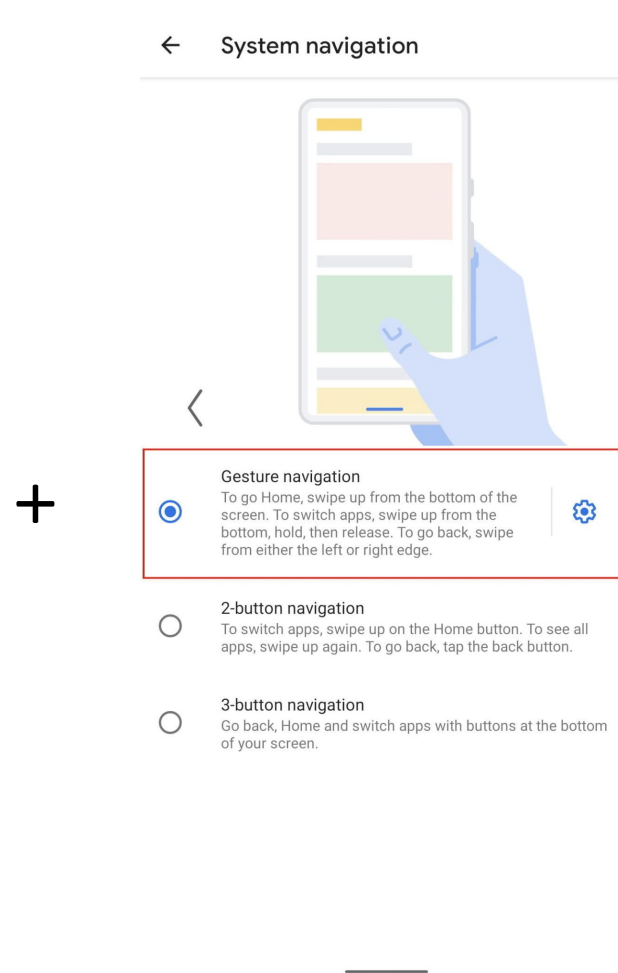
- Over-designed monitoring strategy

- location-related API(getSSID、getBSSID) but not getRssi
- Same implementation in AOSP

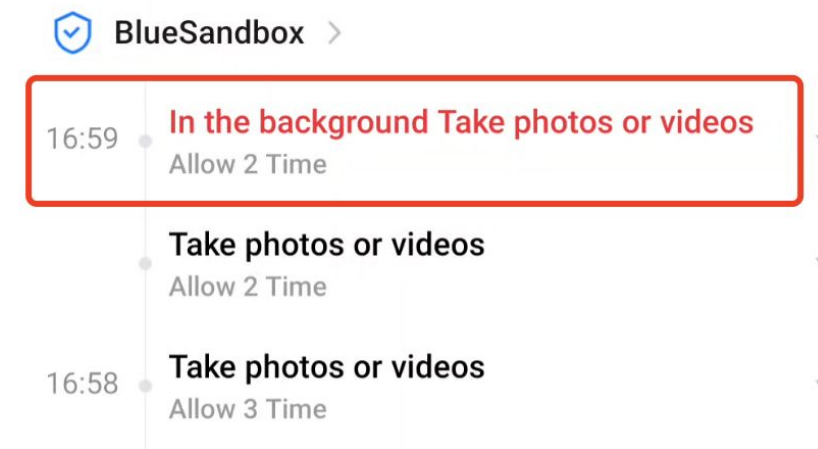
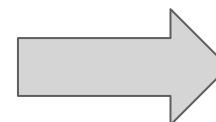
## Case 2: Background camera obtain caused by gesture navigation

```
public void onPause() {  
    super.onPause();  
    //Some recycle code  
    mCamera.release();  
}
```

Camera release normally



Gesture Navigation



Privacy Dashboard result of Vendor A

## Case 2: Background camera obtain caused by gesture navigation

```
→ /tmp adb shell dumpsys appops --package [REDACTED] --op CAMERA
Current AppOps Service state:
Settings:
  top_state_settle_time=+5s0ms
  fg_service_state_settle_time=+5s0ms
  bg_state_settle_time=+1s0ms

Uid u0a293:
  state=cch
  capability=---
  appWidgetVisible=false
  CAMERA: mode=ignore
Package [REDACTED]:
  CAMERA (allow):
    null=[
      Access: [top-s] 2021-09-24 16:59:44.783 (-29m36s214ms) duration=+2s902ms
      Access: [bg-s] 2021-09-24 16:59:47.686 (-29m33s311ms) duration=+60ms
    ]
```

bg-s: background self  
duration: obtain time

## Case 2: Background camera obtain caused by gesture navigation

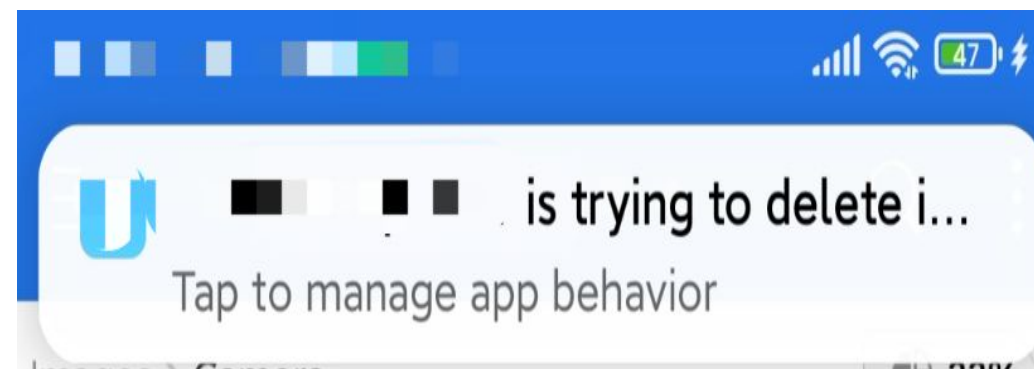
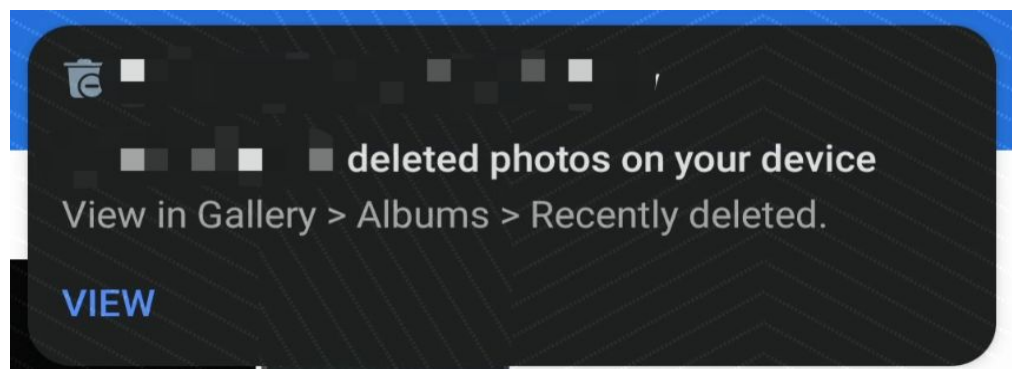
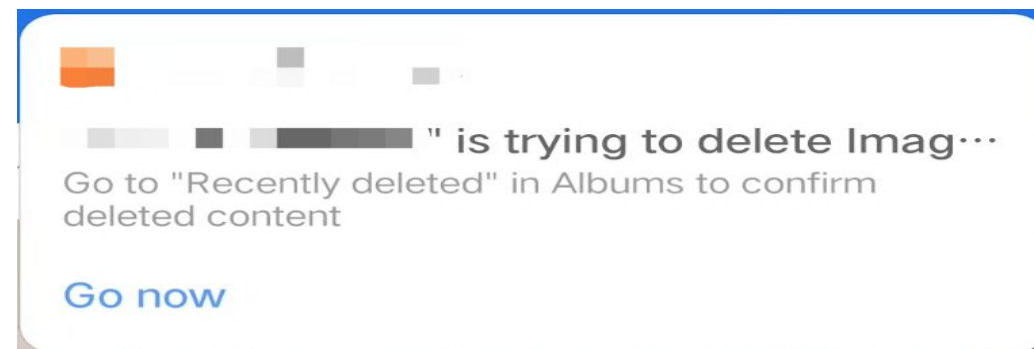
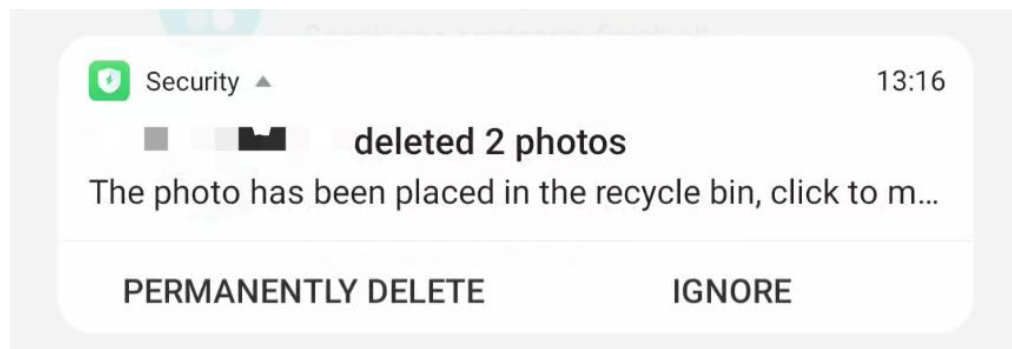
- Critical point between fg and bg - uid stage change
- Time delta from onPause to onUidStateChange

Gesture			3-button		
onPause	onUidStateChange	Time(ms)	onPause	onUidStateChange	Time(ms)
00:15:16.228	00:15:16.547	319	00:18:50.736	00:18:51.416	520
00:15:35.211	00:15:35.524	313	00:19:06.964	00:19:07.596	568
00:15:44.235	00:15:44.596	361	00:19:13.507	00:19:14.26	519
00:15:54.436	00:15:54.762	326	00:19:21.844	00:19:22.446	598
00:16:03.650	00:16:04.16	366	00:19:30.672	00:19:31.273	601

- Gesture navigation with UI change may cause false report

## Case 3: App may bypass album protection to delete files

- Delete photo options blocked by vendor's protection



## Case 3: App may bypass album protection to delete files

- File delete monitor logic and bypass ways

Vendors	Protection	Bypass
B	override “delete” of MediaProvider	java.io.File.delete()
E	“DCIM” path & override java.io.File.delete()	native delete (libc remove)
A & C	“DCIM” path & “unlink” hook	rename files to other path, then delete

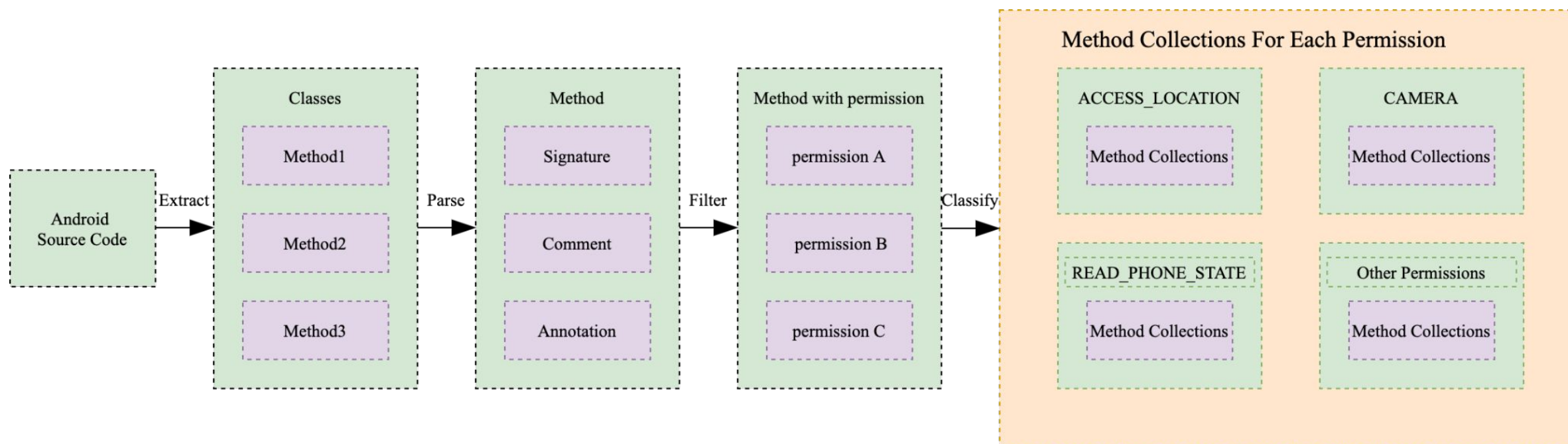
# 4. Measurement and Test Methods



## Measurement And Test Methods

- Measurement: Two Ways of Extracting AppOps API
  - AppOps APIs From Source Code Analysis
  - Sensitive APIs From Reachability Analysis
  
- Test Methods
  - BlueSandbox: A Sensitive API Verify Tool

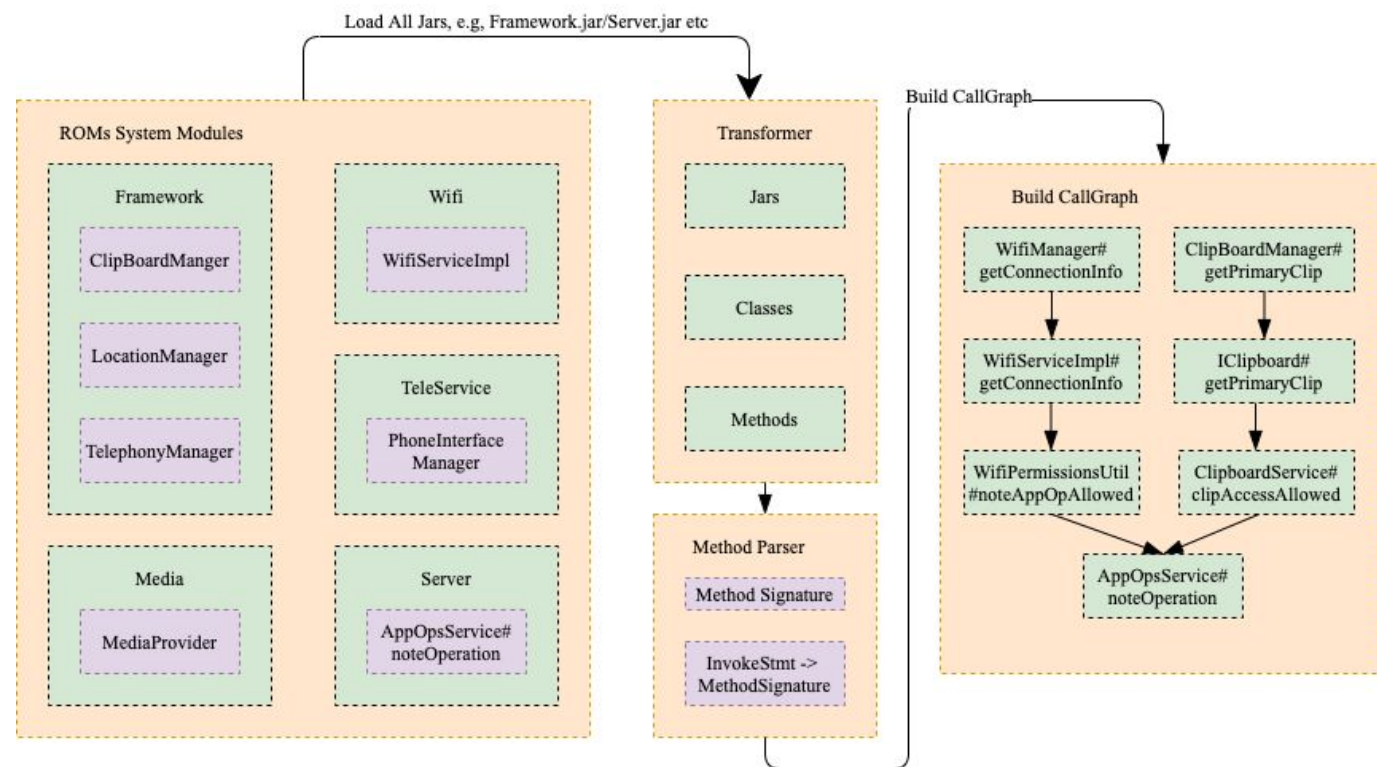
# AppOps API From Source Code Static Analysis



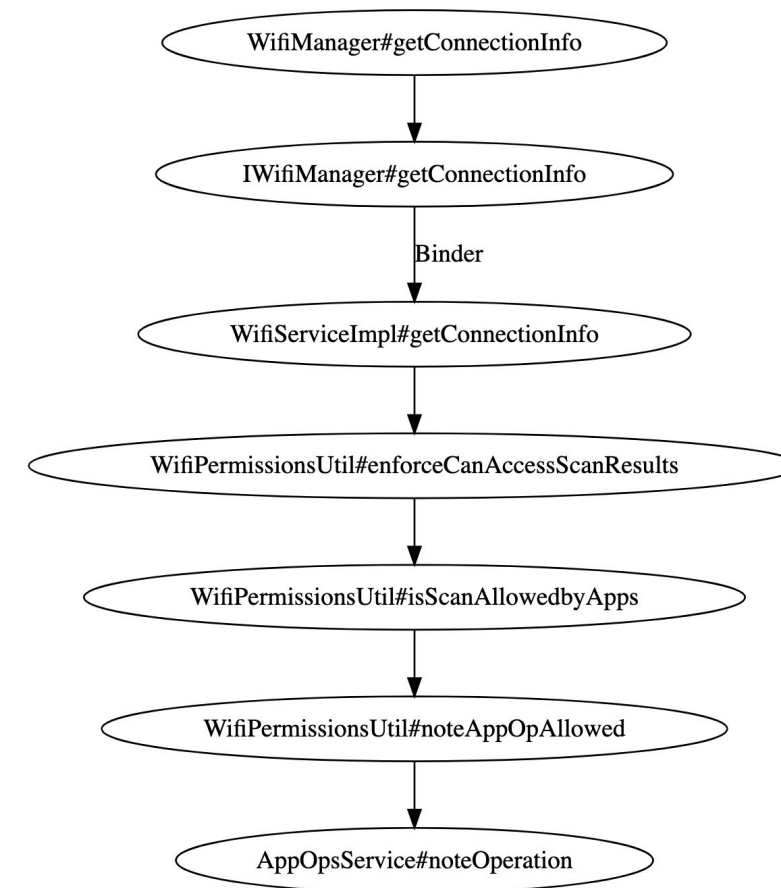
## Steps:

1. Extract All Classes from Android Source Code
2. Extract All Methods from Classes, and build **<Signature, Comment, Annotation>** For Each Method
3. Filter Methods with **Comment or Annotation Marked Permissions**
4. Classify Method Collections For Each Permission

# AppOps API From Reachability Analysis



Reachability Analysis



Steps:

1. Preparation: Framework/Wifi/Media/Server/TeleService Jars
2. Transformer: extract methods, construct <Method, InvokeMethod> Collections
3. CallGraph Builder: build a fully CallGraph within system modules
4. Method Collections: find all reachable paths to AppOps through Reachability Analysis

# Difficulty: Build A Fully CallGraph Within System Modules

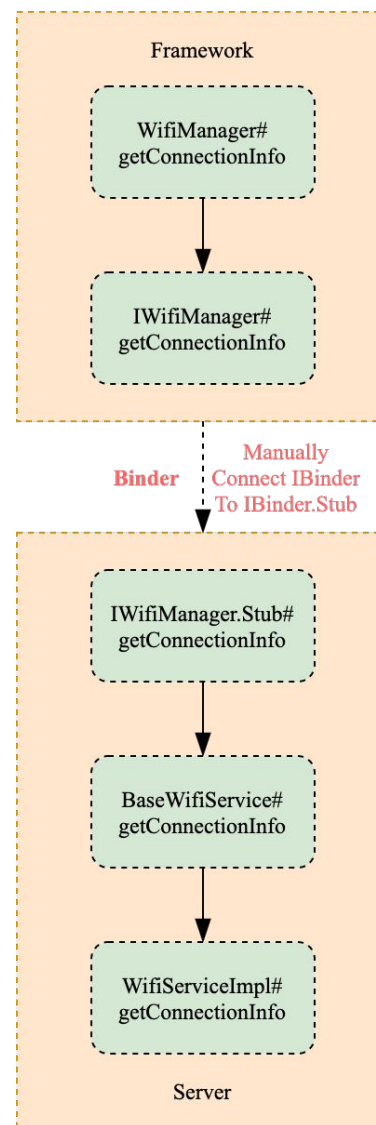
```

# Framework android.net.wifi.WifiManager
public WifiInfo getConnectionInfo() {
    // invoke statements
    return mService.getConnectionInfo();
}

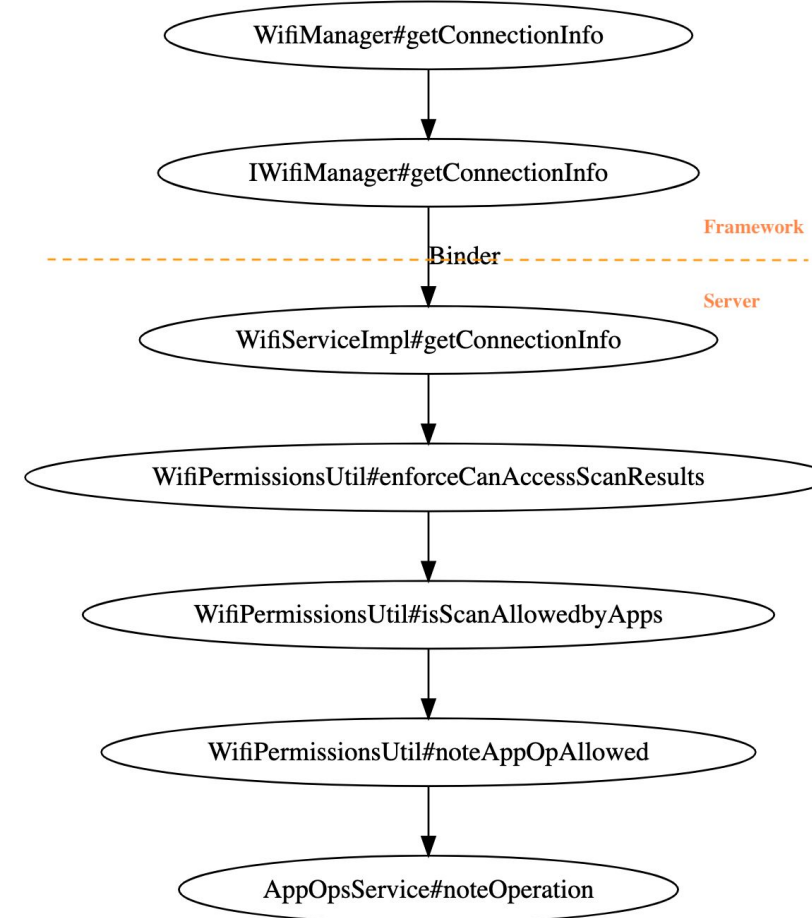
# Server com.android.server.wifi.WifiServiceImpl
public class WifiServiceImpl extends IWifiManager.Stub {
    public WifiInfo getConnectionInfo(String, String) {
        mWifiPermissionsUtil.enforceCanAccessScanResults();
    }
}

```

Binder: Code Snippet

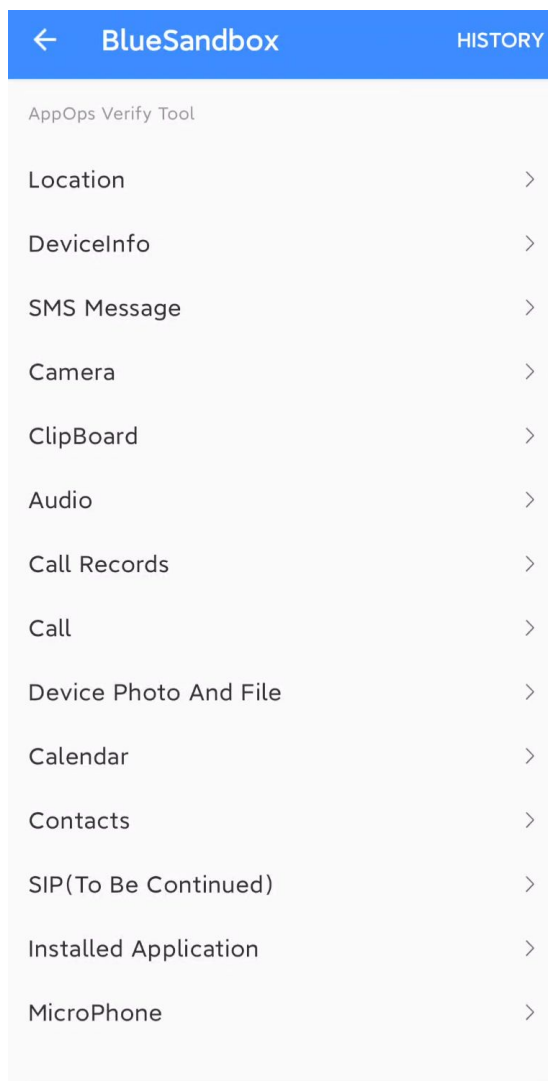


Binder Connection In Static Analysis

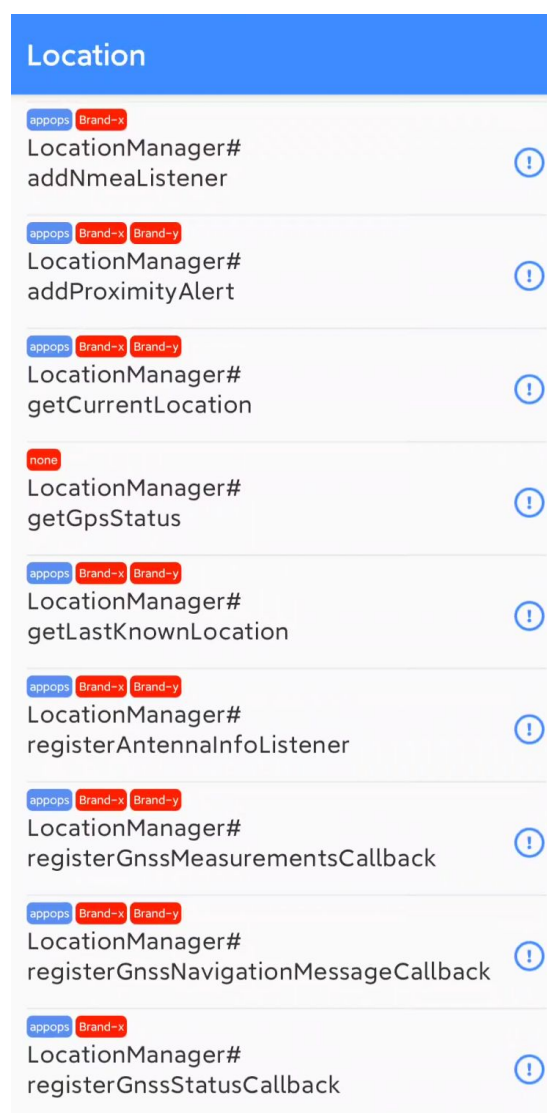


A Connected CallGraph

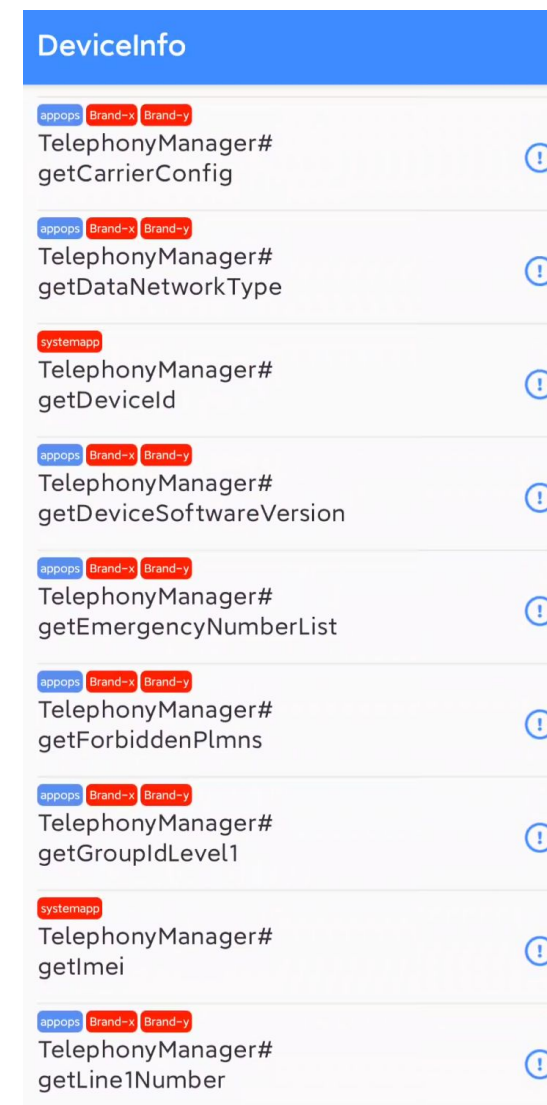
# BlueSandbox: Sensitive API Verify Tool



BlueSandbox



e.g, Location APIs



e.g, DeviceInfo APIs

# 5. Conclusion

- Privacy Dashboard is a new feature on Android 12 aimed to keep track of apps' sensitive behaviors, but several android vendors achieved their own before
- AppOps provide interfaces to track sensitive permission access but not accurate or comprehensive for Vendors to present on privacy dashboard directly
- Vendor's custom privacy protection mechanism need to be improved and verified fully to protect billions of users' privacy