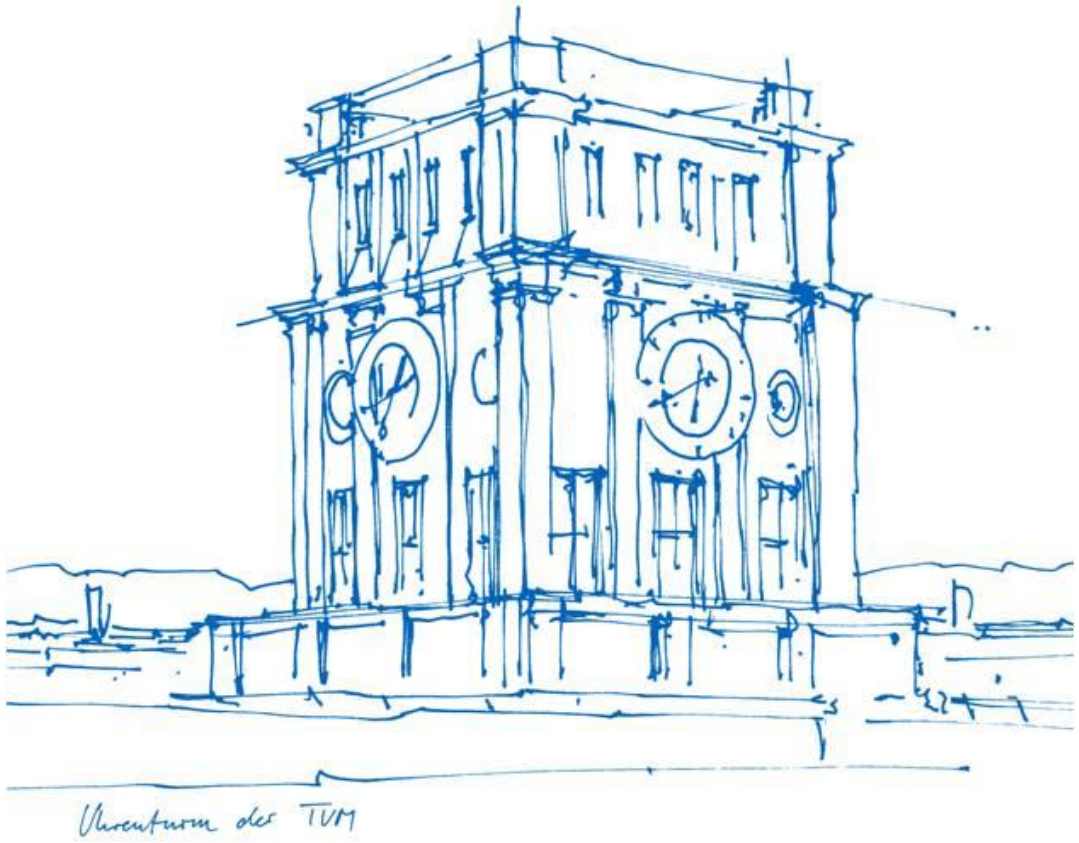
Post-GDPR Privacy Preservation in Android Ecosystem

Mark Meng Huasong, PhD
Research Fellow
Technische Universität München (TUM)



Meng Huasong Mark, PhD

Research Fellow, School of Computation, Information and Technology,
Technical University of Munich

- B.Eng.(Hons), Computer Science, Nanyang Technological University
- M.Comp., Infocomm Security, National University of Singapore
- Ph.D., Computer Science, National University of Singapore

- Research Interests

Mobile security & privacy

Trustworthy AI/LLMs

Android Ecosystem

People heavily rely on their personal mobile devices in their daily lives.

In the year 2023 alone, a total of 1.23 billion units of smartphones were purchased by end users*.

76% of newly activated devices are installed with Android OS+.

** https://www.statista.com/topics/840/smartphones*
+ https://www.idc.com/promo/smartphone-market-share

Android Ecosystem

The Android ecosystem is well-known for the great number of third-party applications (apps).

Our personal data are pervasively accessed and utilized by apps in the name of data analytics and personalized advertising.

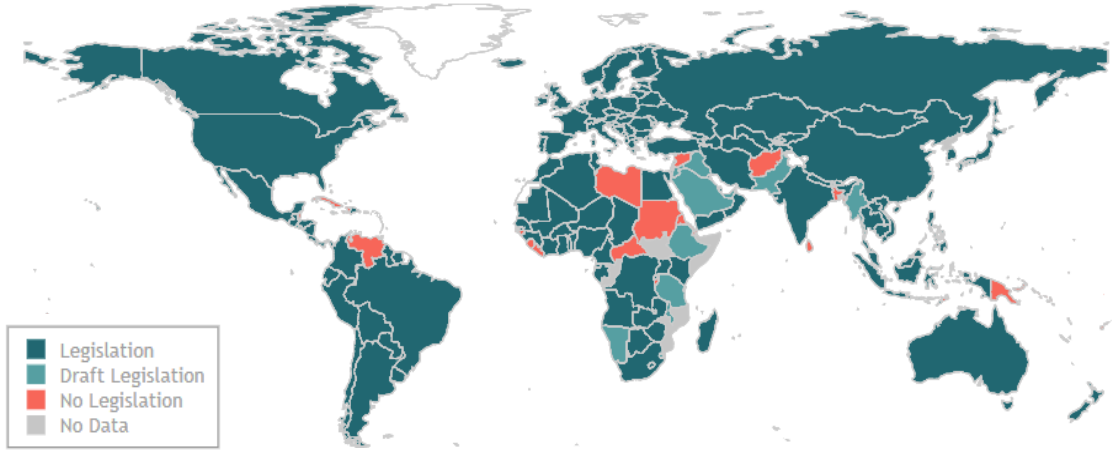
Post-GDPR Era

The protection of personal data has gained a great deal of attention around the world.

137 out of 194 countries had put in place legislation to secure the protection of data and privacy.

- EU's General Data Protection Regulation (GDPR)
- California Consumer Privacy Act (CCPA)

Our phones are the first line of defence against privacy infringement.



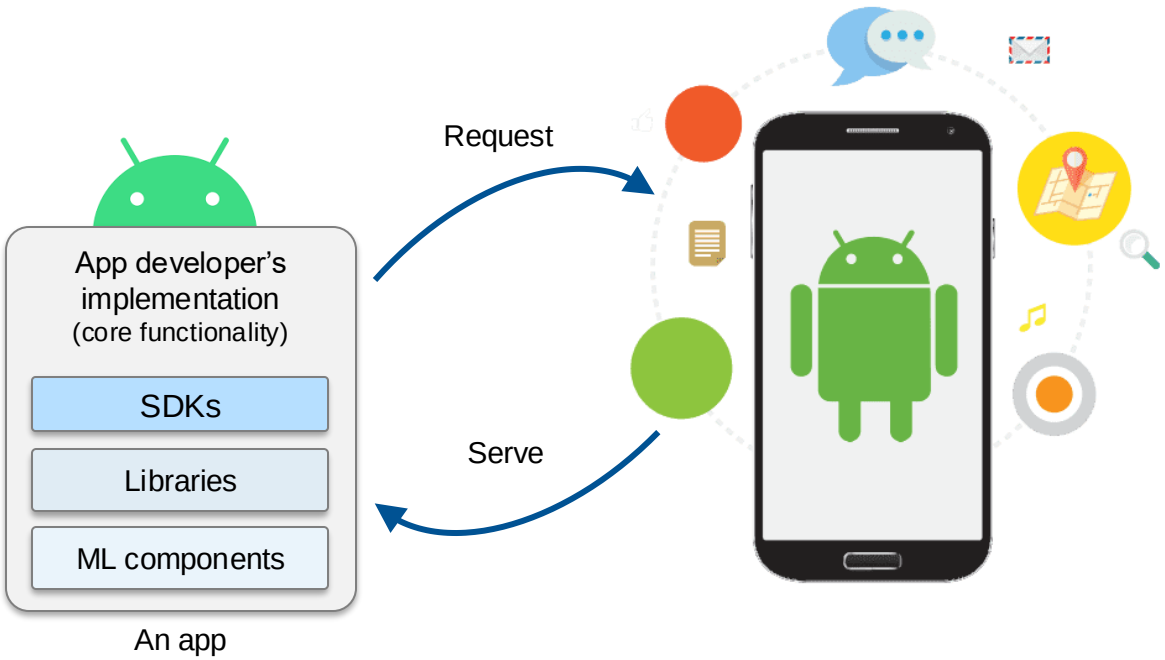
Data Protection and Privacy Legislation Worldwide (as of 2022)

Personal Data Access

Collecting personal data follows a simple routine: an app requests access to personal data, and the operating system serves the personal data.

Personal data request of an app is ***not*** necessarily initiated by app developer's implementation.

The embedding SDKs, libraries, and ML components may also collect personal data.



Privacy Preservation in the Android Ecosystem

Personal data exfiltration in third-party apps has been extensively studied since the nascent stage of Android history.

Google has deployed strict and comprehensive regulations for app developers.

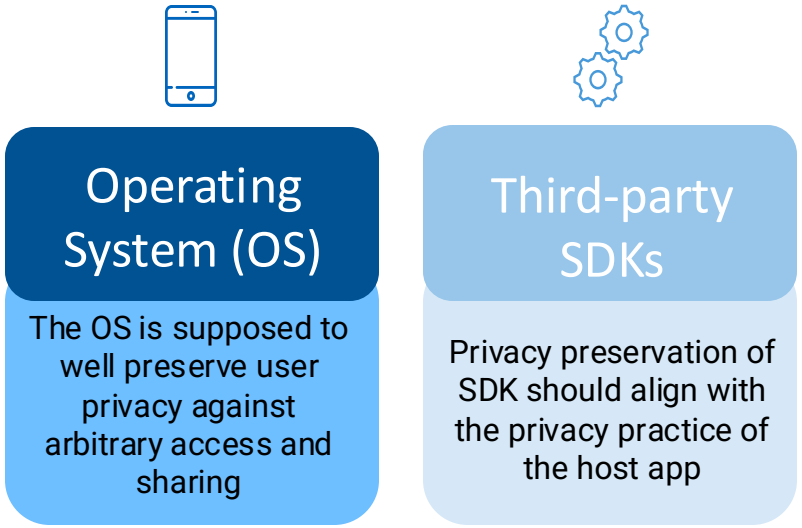
App analysis only offers a partial view of the data exfiltration in the entire Android ecosystem.

There are still some open questions to complement existing research and unveil the complete landscape of privacy preservation.

Roadmap

- Third-party apps are just one role in the entire Android ecosystem → ***How about the operating system itself? (1st half)***

- Not all code is written by app developers → ***Third-party libraries/SDKs are widely used in apps! What is the landscape of their compliance? (2nd half)***



OS-level Safeguards of User-unresettable Identifiers (UUIs)

This work has been accepted and presented in NDSS 2023:

M. H. Meng, Q. Zhang, G. Xia, Y. Zheng, Y. Zhang, G. Bai, Z. Liu, S. G. Teo, J. S. Dong, “Post-GDPR Threat Hunting on Android Phones: Dissecting OS-level Safeguards of User-unresettable Identifiers”, The Network and Distributed System Security Symposium (NDSS), San Diego, USA, 2023.

Background

Google has taken steps to enforce new privacy features to restrict apps' use of user data.

Representative privacy changes in Android 10:

- New privileged permissions
  - *"READ_PRIVILEGED_PHONE_STATE"*
  - Only granted to privileged system apps.
- App-unique and user-resettable identifiers
  - Android advertising ID
- Compulsory disclosure of apps' *access*, *collection*, *use*, and *sharing* of user data.

Related Work & Motivation

Personally identifiable data exfiltration in Android at the *app level* or *library level* has been extensively studied by the research community (Enck et al., 2014, Qiu et al., 2015, Ren et al., 2018, Wang et al., 2020).

Privacy protection at the ***OS level*** is still an open question.

Many types of personally identifiable data served at the OS level are user-unresettable.

- Serial number, IMEI, MAC Address, etc.
- Not easily replaceable as a password once leaked.
- We refer to them as **user unresettable identifiers (UUIs).**

Whether the operating systems (OSes) themselves comprehensively safeguard UUIs?

Understanding Android UUIs

These 6 pre-identified UUIs are useful for us to form our understanding of more types of undiscovered UUIs on Android devices.

#	UUI	Category	API(s)	Permission/API changes*
1	Serial number	Chip & Cellular	android.os.Build: getSerial()	V8-9: **READ_PHONE_STATE** required. ≥10: **READ_PRIVILEDGED_PHONE_STATE** required.
2	Device ID (IMEI or MEID)		android.telephony.TelephonyManager: getImei(), getDeviceId(), getMeid()	<10: **READ_PHONE_STATE** required. ≥10: **READ_PRIVILEDGED_PHONE_STATE** required.
3	ICCID		android.telephony.TelephonyManager: getSimSerialNumber() android.telephony.SubscriptionInfo: getIccid()	
4	IMSI		android.telephony.TelephonyManager: getSubscribeId()	
5	Bluetooth MAC Address	Wireless Module	android.bluetooth.BluetoothAdapter: getAddress()	All versions: **BLUETOOTH** required. ≥10: Randomization or a fixed return value required. V6-10: **ACCESS_COARSE_LOCATION** or **ACCESS_FINE_LOCATION** required. ≥10: **ACCESS_FINE_LOCATION** becomes mandatory.
6	WiFi MAC Address		android.net.wifi.WifiInfo: getMacAddress()	All versions: **ACCESS_WIFI_STATE** required. V6-9: Randomization suggested. ≥10: Randomization becomes mandatory.

For each UUI, there is/used to be at least 1 official API for apps to access.

* The color scheme indicate the permission protection levels: **normal**, **dangerous**, and **signature**.

Goal & Challenges

We design and implement U2-I2 (short for UUI Investigator)

- Systematically investigates the OS-level UUI protection at large scale.
- Designed to assess the protection of not only the six known UUIs, but also other previously unreported ones.

Challenges of our study:

1) Identify undocumented access channels
2) Automate assessment process
3) Pinpoint customized (undiscovered) UUIs

Assessing Documented Channels

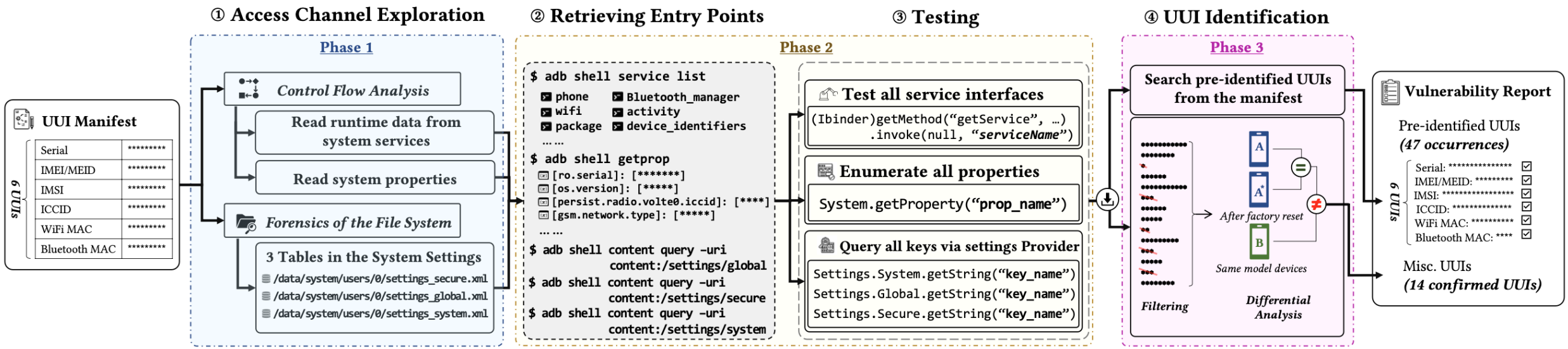
U2-I2 aims to test **two** types of errors:

1) Legacy permissions
    - e.g., additional permissions requested, or higher permissions introduced
2) Missing de-identification
    - e.g., MAC address randomization

Approach Overview

For Undocumented Channels Assessment

U2I2 adopts a three-phase approach to assess undocumented channels.



Assessing Undocumented Channels

Phase 1 - Access Channel Exploration

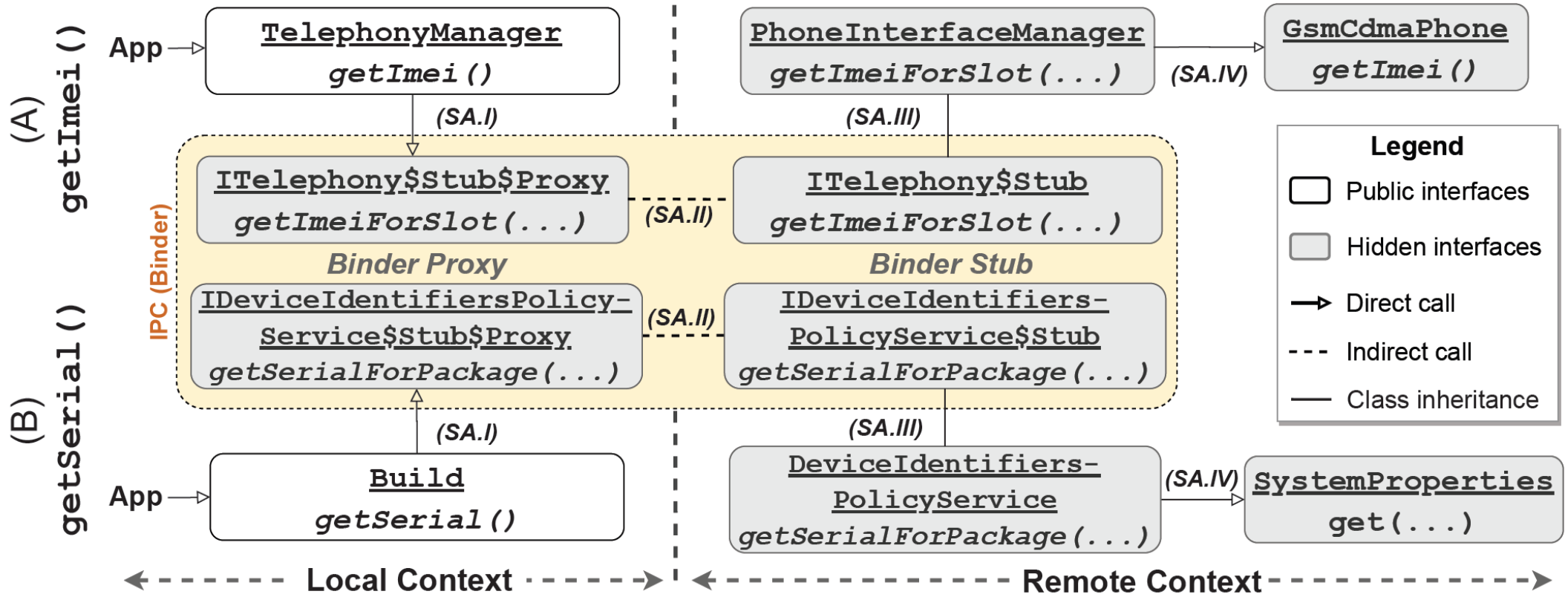
U2-I2 takes **six** pre-identified UUIs as seeds, considering that other unknown UUIs may share the same set of access channels.

Two strategies to explore undocumented channels:

1) Static control flow analysis
    - Extract method-level call relations until service managers
    - Map local interfaces to the corresponding remote interfaces
    - Pinpoint the components that serve the request
2) Filesystem forensics
    - Search the values of six pre-identified UUIs

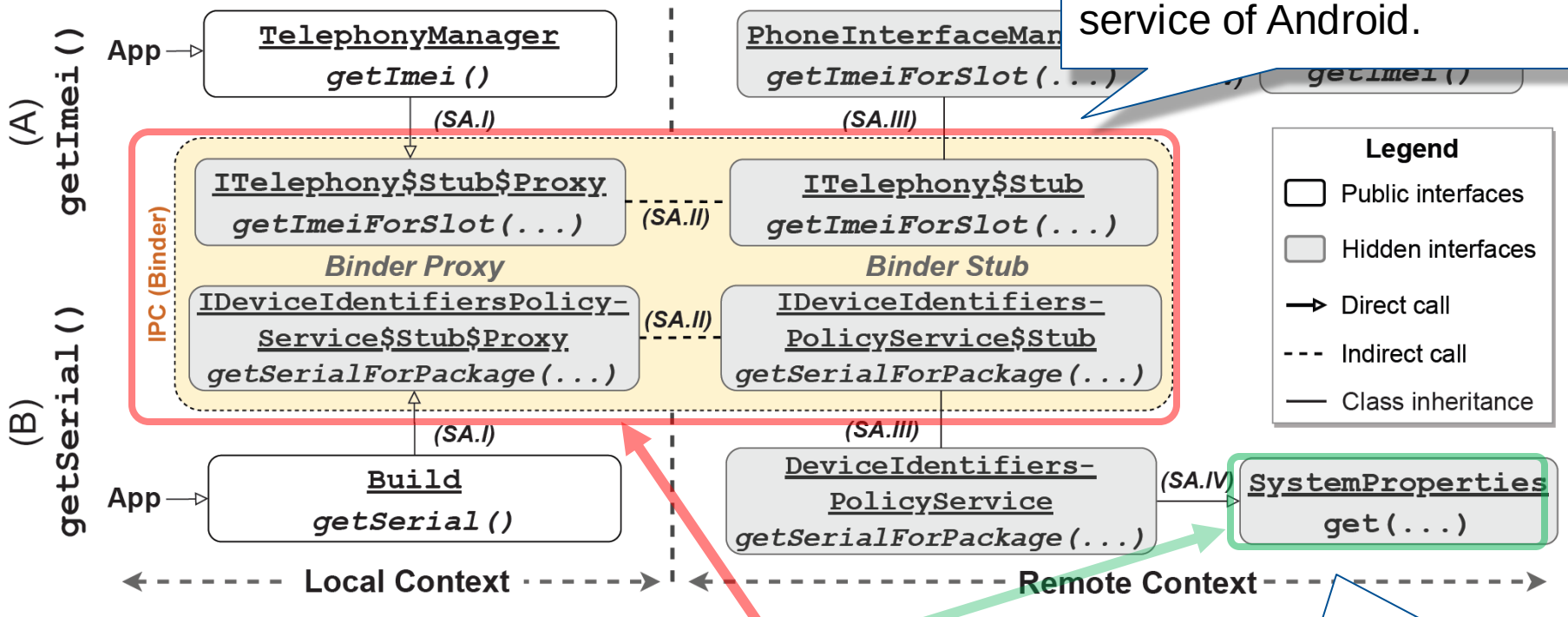
Assessing Undocumented Channels

Two Typical Static Control Flows of the API Invocation



Assessing Undocumented Channels

Two Typical Static Control Flows of the API Invocation



The invocation of API to access a UUI is usually handled by a system service of Android.

*Visualization of two typical control flows starting from documented API (**System properties** and **system services**)*

System properties are the actual component to serve the serial number.

Assessing Undocumented Channels

Phase 1 - Access Channel Exploration

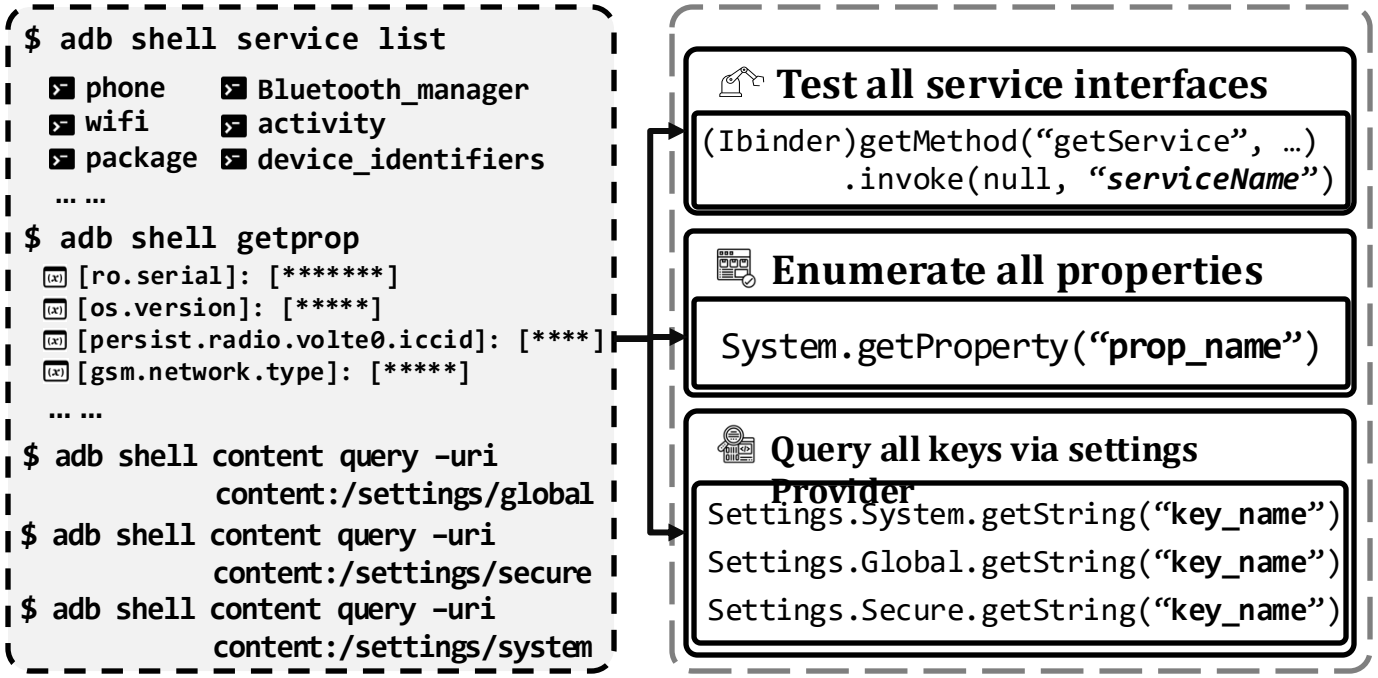
Through the exploration, U2-I2 recognizes three undocumented access channels:

1) **System settings**
   - Stored in persistent storage
   - Three key-value databases (.xml format)
2) **System properties**
   - Stored inside system directory
   - Indexed by keys
   - Initialized at boot time
3) **System services**
   - Enabled by inter-process communication
   - Serve API requests through its public interfaces

Assessing Undocumented Channels

Phase 2 – Retrieving Entry Points and Testing

U2-I2 makes use of public interfaces to query **system properties** and **system settings,** and resorts to a “hacking way” through Java reflection to bypass the permission check to invoke **system services**.



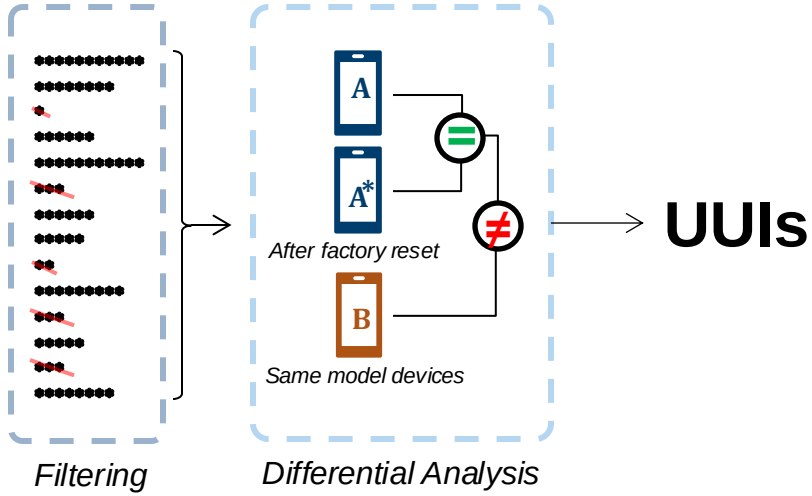
Assessing Undocumented Channels

Phase 3 – UUI Identification

U2-I2 adopts a two-step approach to pinpoint unknown or OEM-defined UUIs

1) Filtering
    - Excludes values of insufficient size (e.g., 4 hex-digit)
2) Differential Analysis
    - Excludes values that are same across devices
    - Excludes values that are changed after factory reset

Here we assume all UUIs have fixed entry names that can not be changed once created.



Evaluation

Our evaluation aims to address three research questions:

RQ1: What is U2-I2's performance in identifying UUI mis-handling issues? What is the status quo of UUI protection in the latest Android phones? Do their OSes comply with the up-to-date privacy policy of AOSP?

RQ2: Based on U2-I2's findings, what UUIs are potentially exposed to non-system apps, and what are the typical exfiltration (exposure) points?

RQ3: Have our identified access channels of UUI access already been (ab)used by apps in the wild?

Key Finding for RQ1

Landscape of OS-level UUI Safeguards

Test devices cover **13** latest models from **9** manufacturers, which represent almost **85%** of the global market share of Android devices*.

The UUI mishandling issues are pervasive in the latest Android phones.

- **51** unique vulnerabilities, leading to **65** occurrences of UUI leakages.
- **12** out of **13** tested device models contain at least **1** UUI leakage,
- **1** vulnerability found in AOSP# and is inherited in all tested OEM devices.
- **18** leakages are associated with unknown UUIs (classified as misc. UUIs)

		AOSP 10	AOSP 11	A_{10} (Huawei)	B_9 (Lenovo)	C_{10} (OnePlus)	C_{11} (OnePlus)	D_{10} (Oppo)	D_{11} (Oppo)	E_{10} (Samsung)	F_{10} (Smartisan)	G_{10} (Vivo)	H_{10} (Xiaomi)	H_{11} (Xiaomi)	**Total**
Chip & Cellular	Serial	0	0	1	1	2	1	1	1	2	0	1	1	1	***12***
	DID/IMEI/ MEID	0	0	4	2	0	0	0	1	1	0	3	0	0	***11***
	ICCID	1	0	2	0	2	0	0	4	1	1	3	1	0	***15***
	IMSI	0	0	0	2	0	0	0	0	1	0	1	0	0	***4***
	Subtotal	***1***	***0***	***7***	***5***	***4***	***1***	***1***	***6***	***5***	***1***	***8***	***2***	***1***	***42***
Wireless Module	Bluetooth MAC	0	0	0	1	0	0	0	0	0	1	0	0	0	***2***
	WiFi MAC	0	0	0	0	0	0	1	0	0	0	2	0	0	***3***
	Subtotal	***0***	***0***	***0***	***1***	***0***	***0***	***1***	***0***	***0***	***1***	***2***	***0***	***0***	***5***
	Misc. UUIs	0	0	0	3	1	2	1	1	0	1	0	7	2	***18***
	Total	***1***	***0***	***7***	***9***	***5***	***3***	***3***	***7***	***5***	***3***	***10***	***9***	***3***	***65***

Statistics of the occurrence of UUI leakages detected in our assessment, counted by UUI types and devices

* Data source https://gs.statcounter.com/vendor-market-share/mobile/worldwide

\# Google Pixel model(s). AOSP 11 stands for a Google Pixel phone installed with Android OS 11.

Key Finding for RQ1

Landscape of OS-level UUI Safeguards – Misc. UUIs

14 unique miscellaneous UUIs recognized from **18** occurrences.

Recognized miscellaneous UUIs covers identifiers of NFC module, display panel, PCB, camera, and fingerprint sensors.

Channel	Name	Format & Purpose (keywords display in Bold format)
System Settings	`Global.cplc` (C_{10})	45-byte hex string, for **NFC** module
	`Global.ro.boot.oled_wp` (H_{11})	8-byte hex string, for OLED **display panel**
	`System.ReaperAssignedDeviceId` (B_9)	30-digit decimal string, unknown type+
System Properties	`gsm.serial` ($C_{10,11}$, $D_{10,11}$)	19-digit decimal string, the device **PCB** serial number
	`ro.ril.oem.sno` ($H_{10,11}$)	8-byte hex string, unknown type+
	`vendor.camera.sensor.frontMain.fuseID` (H_{10})	64-byte alphanumeric string, ID of the front main **camera**
	`vendor.camera.sensor.rearMain.fuseID` (H_{10})	64-byte alphanumeric string, ID of the rear main **camera**
	`vendor.camera.sensor.rearUltra.fuseID` (H_{10})	64-byte alphanumeric string, ID of the rear ultra-wide **camera**
	`vendor.camera.sensor.rearTele.fuseID` (H_{10})	64-byte alphanumeric string, ID of the rear telephoto **camera**
	`persist.vendor.sys.fp.info` (H_{10})	8-digit hex string, **fingerprint** sensor related
	`persist.vendor.sys.fp.uid` (H_{10})	14-digit hex string, **fingerprint** sensor related
	`ro.qchip.serialno` (F_{10})	8-digit hex string, a serial number of an embedded **chip** module
	`ro.recovery_id` (B_9)	32-digit hex string, ID of the **boot** image
	`ro.expect.recovery_id` (B_9)	32-digit hex string, a same value as above

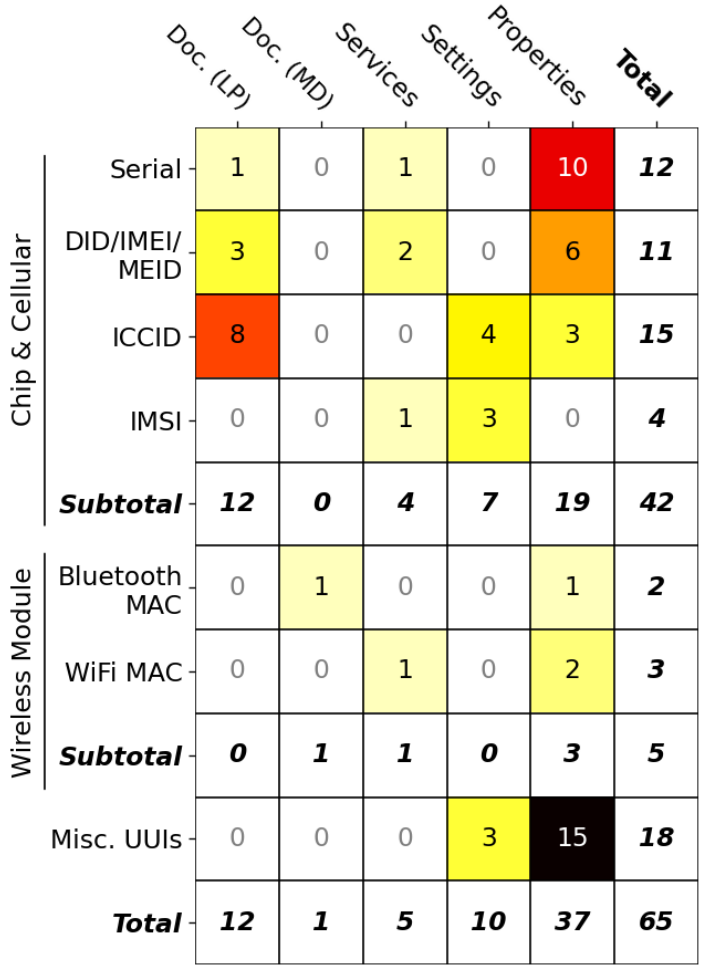
+ The UUIs labelled as “unknown type” contain insufficient information in their names and values.

Key Finding for RQ2

Exfiltration points via undocumented access channels

The **undocumented access channels** are the major exfiltration points.

- Contributes **45** out of all **51** vulnerabilities
- **5** are caught from the system services, **10** in system settings, and the remaining **30** from system properties.



		Doc. (LP)	Doc. (MD)	Services	Settings	Properties	Total
Chip & Cellular	Serial	1	0	1	0	10	***12***
	DID/IMEI/ MEID	3	0	2	0	6	***11***
	ICCID	8	0	0	4	3	***15***
	IMSI	0	0	1	3	0	***4***
	Subtotal	***12***	***0***	***4***	***7***	***19***	***42***
Wireless Module	Bluetooth MAC	0	1	0	0	1	***2***
	WiFi MAC	0	0	1	0	2	***3***
	Subtotal	***0***	***1***	***1***	***0***	***3***	***5***
	Misc. UUIs	0	0	0	3	15	***18***
	Total	***12***	***1***	***5***	***10***	***37***	***65***

Statistics of the occurrence of UUI leakages detected in our assessment, counted by UUI types and access channels

Key Finding for RQ3

Exploits in the wild

“ Have our identified undocumented channels already been (ab)used in the wild?

We test the top 150 apps from the Google Play Store and another 150 apps from an alternative app store (Xiaomi App Store).

- **12 out of 300 analyzed apps** have relevant behaviors.
- All their accesses are through **undocumented** access channels.
- The broad UUI collection by apps in the wild is not observed in our study.

List of apps found potential UUI collection

#	App package name	Downloads (by Dec 2021)	Involved UUIs	Access channels
1	com.kwai.m2u	81mil+	Device ID, Misc. UUI+	Properties
2	com.kwai.videoeditor	86mil+	Device ID, Misc. UUI+	Properties
3	com.lbe.parallel.intl	100mil+	IMSI	Services
4	com.liulishuo.engzo	116mil+	Device ID	Properties
5	com.oppo.store	3mil+	Misc. UUI+	Properties
6	com.renrendai.haohuan	25mil+	Device ID	Properties
7	com.tencent.qqimsecure	682mil+	IMSI	Settings
8	com.tencent.wifimanager	192mil+	IMSI	Settings
9	com.wuba.zhuanzhuan	234mil+	Device ID	Properties
10	com.xunmeng.merchant	50mil+	Serial	Properties
11	com.xunmeng.pinduoduo	5bil+	Serial	Properties
12	ru.yandex.searchplugin	100mil+	Device ID	Properties

+ The caught UUI access is through requesting the system property “gsm.serial”.

Responsible Disclosure & Impact

- We have reported all our findings to the relevant parties/stakeholders.
- We have received **8** CVE entries registered by Google and four other manufacturers.

CVE-listed vulnerabilities found by U2-I2

#	CVE ID	Affected devices	Involved UUIs
1	CVE-2020-12488	Vivo	Serial
2	CVE-2020-14103	Xiaomi	Serial
3	CVE-2020-14105	Xiaomi	Misc. UUI (ro.ril.oem.sno)
4	CVE-2021-0428	Pixel	ICCID
5	CVE-2021-25344	Samsung	Serial
6	CVE-2021-25358	Samsung	IMSI
7	CVE-2021-26278	Vivo	WiFi MAC address
8	CVE-2021-37055	Huawei	ICCID

Contributions

- The first comprehensive study to understand OS-level UUI protection.
- A systematic assessment approach to explore undocumented channels.
- Revealing the status quo of UUI protection on the latest versions of Android.

A Large-Scale Privacy Assessment at the Third-Party SDK Level

This work has been submitted and is under peer review.

M. H. Meng, C. Yan, Y. Hao, Q. Zhang, Z. Wang, K. Wang, S. G. Teo, G. Bai, J. S. Dong, “A Large-Scale Privacy Assessment of Android Third-Party SDKs”, [Preprint] https://doi.org/10.48550/arXiv.2409.10411.

Background

Third-Party SDKs

A software development kit (SDK) is a set of platform-specific building tools for developers*.

Third-party SDKs offers app developers a broad spectrum of functionalities

- User authentication
- UI enhancement
- Advertising/analytics, etc.

Android ecosystem provides an easy way to integrate SDKs in apps

A third-party app embeds 8-9 SDKs in average+.

* *Definition cited from Amazon AWS developer documentation.*

\+ *Statistical results of the top 300 apps from Google Play App Store and the top 300 apps from Xiaomi App Store.*

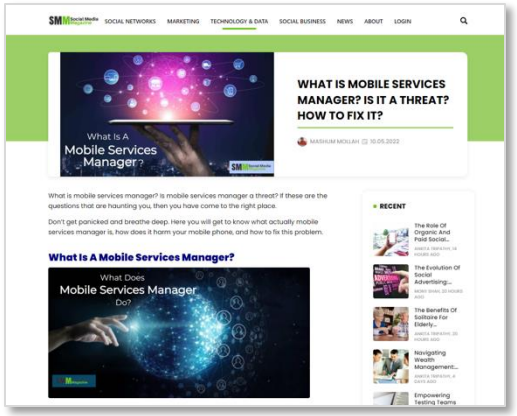
Background

A Motivating Example – Mobile Services Manager

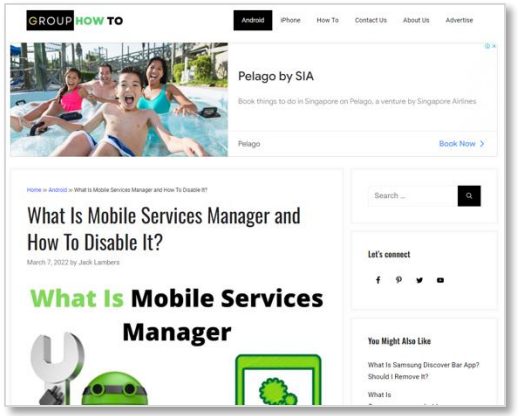
A well-known rogue SDK that downloads apps without user's consent and spams notifications without user's permission.

Embedded in a pre-installed carrier's app in the same name or named "DT Ignite".

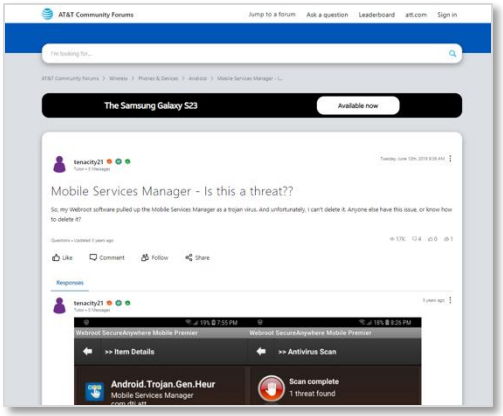
Widely covered by media.



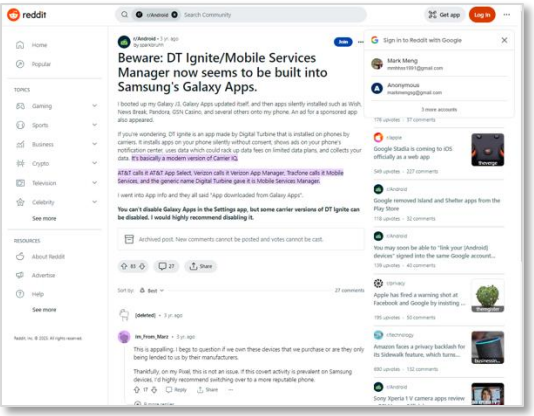
What is Mobile Services Manager? Is it a threat? How to fix it?



What is Mobile Services Manager and how to disable it?



Mobile Services Manager – is this a threat?



Beware: DT Ignite/Mobile Services Manager now seems to be built into Samsung's Galaxy Apps

Background

Privacy Protection in SDKs

Lack of Transparency

- Implementation of third-party SDKs is black-box to app developers
- Hard to manage personal data access when there is a permission requested and granted by the embedding app

Insufficient Regulations

- Markets do not impose a strict regulation (like apps)
- Privacy policies are often composed of vague and ambiguous guidelines

SDK users (developers of the embedding apps) **are often left in the darkness.**

Related Work

Third-Party SDKs Privacy Leakage Analysis

- There is a substantial body of literature addressing various aspects of privacy concerns in the Android ecosystem, which mainly focusing on the privacy preservation OS level [Meng et al., 2023] and app level [Enck et al., 2010, Zhang et al., 2009, Zhou et al., 2013].
- Various analysis techniques have been explored, including dynamic analysis [He et al., 2018, Razaghpanah et al., 2018] and static analysis [Enck et al., 2010, Meng et al., 2023].
- Privacy protection of third-party libraries has been explored in [He et al., 2018, Liu et al., 2020] although there lacks a study in large scale.

Privacy Compliance Evaluation in Other Fields

- Numerous works focus on detecting inconsistencies between the privacy practices of an app and its descriptions/privacy polices [Andow et al., 2020, Yu et al., 2021].
- Recent literature [Yan et al., 2024] resorts to large language models to assess of privacy-related documents that varies in length, structure, and writing quality.

There is no large-scale privacy measurement in the Android ecosystem at the SDK level.

Goal & Challenges

This work aims to gain insights into the landscape of privacy protection at the SDK level through a large-scale study of existing SDK packages.

Three Challenges

- Assess SDK privacy disclosure in natural language
- Investigate SDK privacy behaviors (read, collect, and share)
- Measure SDK privacy preservation on a large scale

Potential Compliance Issues

Three types of issues are investigated:

- **(Type I) Privacy leakage**
  - An SDK shares users' privacy in public spaces.

- **(Type II) Excessive collection**
  - An SDK attempts to read more types of privacy data than what it requests/claims in its privacy policy

- **(Type III) Over-claiming**
  - An SDK is found to claim its access to more types of data than what it reads

Scope of Privacy Data

We resort to the privacy-related changes in each Android OS release and existing literature [Enck et al., 2010, Zhou et al., 2013, Meng et al., 2023, etc.] to complement the list of privacy data (e.g., user-unresettable identifiers (UUIs)).

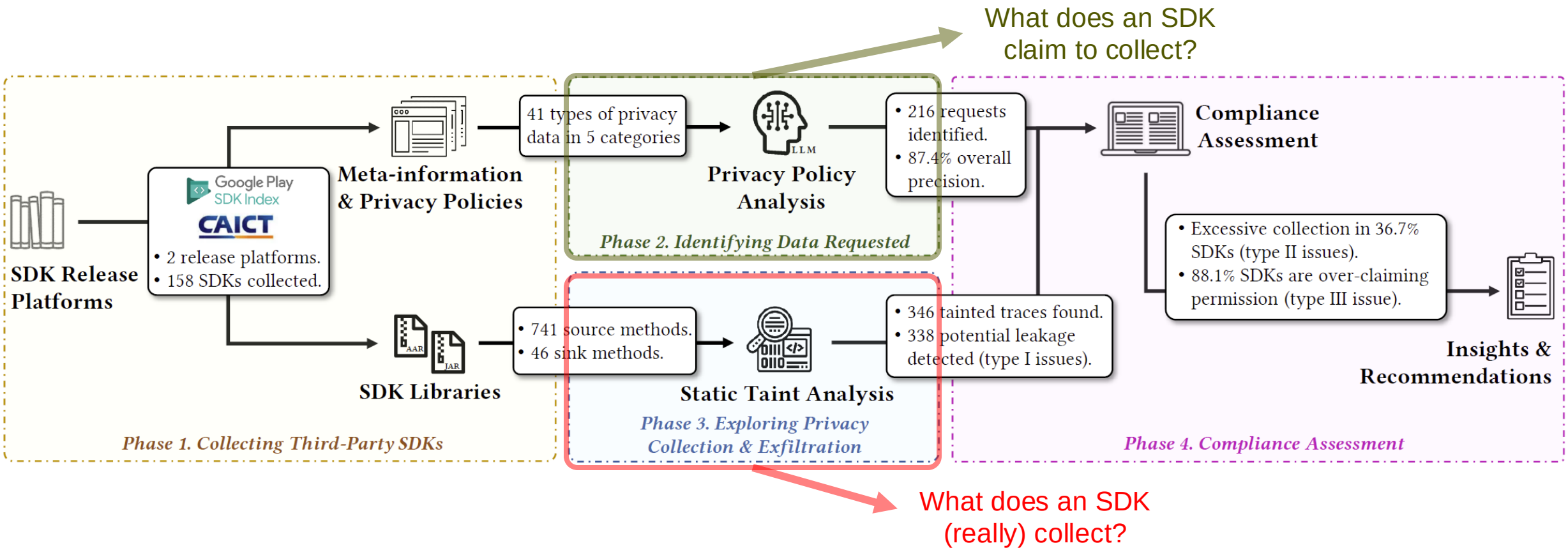
41 types of privacy data categorized into 5 classes are recognized

- **Class 1**: Chip, Cellular and Peripheral (e.g., IMEI, serial, telephone number)
- **Class 2**: Wireless Communication (e.g., Bluetooth MAC, SSID/BSSID, IP address)
- **Class 3**: Location and Sensors (e.g., camera, fine and coarse locations)
- **Class 4**: Media and Software Specific Data (e.g., app list, media location, GAID)
- **Class 5**: Personal Data (e.g., account info, contact list, browser bookmarks)

Our Approach

An overview

A large-scale study to analyze what do SDKs collect and what do SDKs claim to collect.



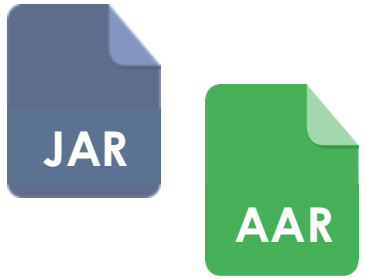
Collecting Third-Party SDKs (Phase 1)

Collecting SDKs

Collect third-party SDKs from two mainstream release platforms

- Two repositories identified
  - Google Play SDK Index
  - CAICT – An alternative source

- 158 SDKs with their latest release collected
  - 139 are from Google Play SDK Index and 19 are from CAICT
  - JAR/AAR package files
  - Meta-information and privacy policies/developer disclosures



Collecting Third-Party SDKs (Phase 1)

Collecting Meta-Information and Privacy Policies

We implement a Python crawler by Selenium

Meta information Crawling

- SDK name
- Provider (developer's identity)
- Maven IDs
- Functional Category

Privacy Polices

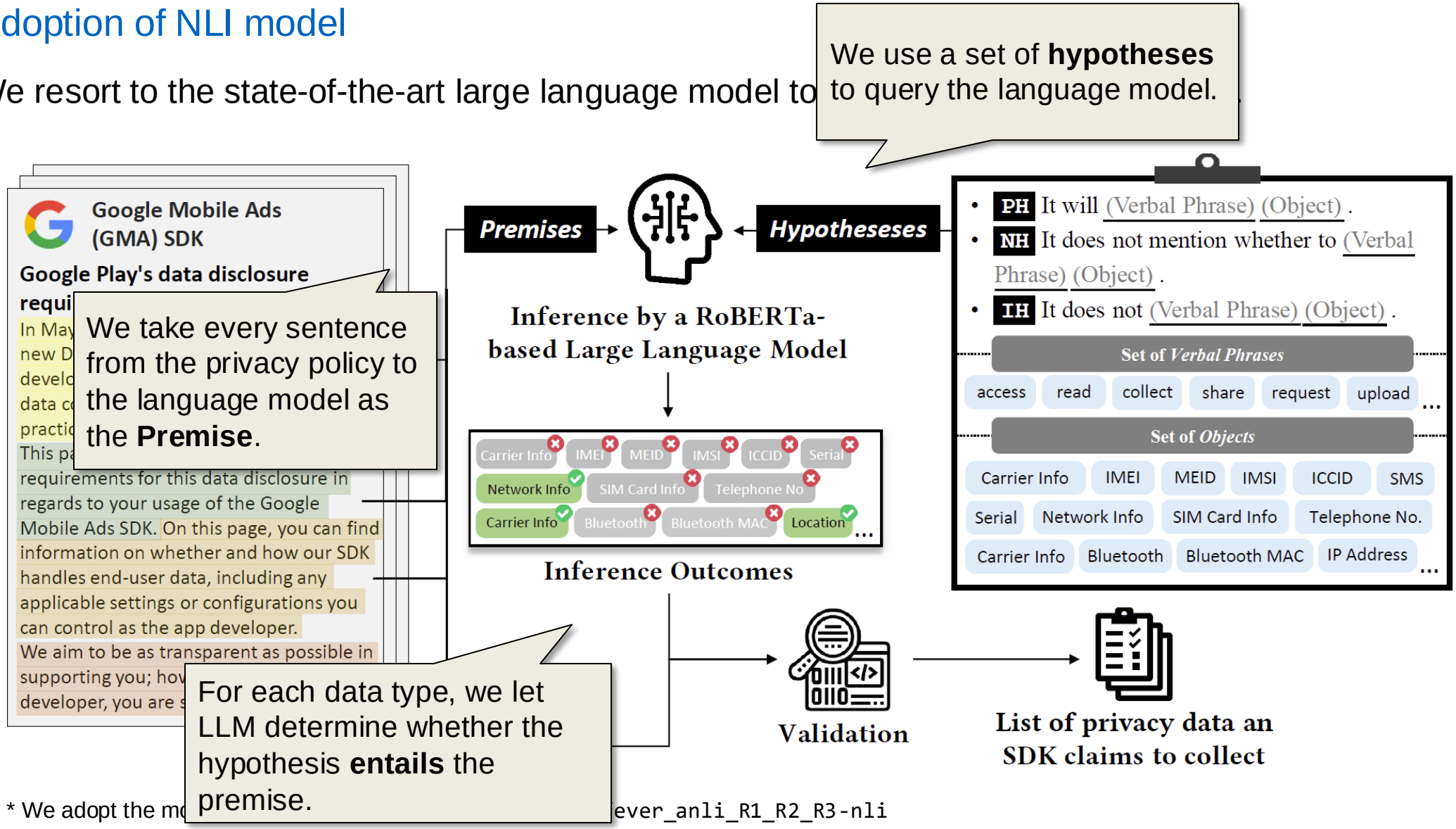
- Linked webpages are crawled*
- Only 52 SDKs' privacy policies are provided on their product pages
- 109 SDK privacy policies are collected after manual search

** indexed by data safety section guidance (Google Play SDK Index) and privacy policy (CAICT)*

Identifying Data Requested in Privacy Policies (Phase 2)

Adoption of NLI model

We resort to the state-of-the-art large language model to

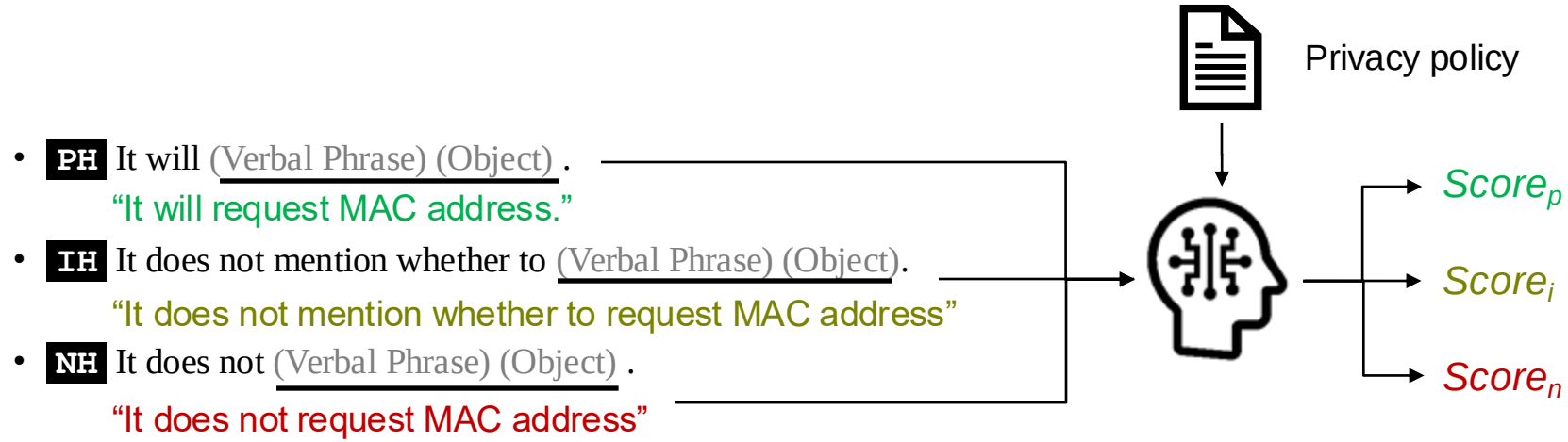


* We adopt the mo ever_anli_R1_R2_R3-nli

Identifying Data Requested in Privacy Policies (Phase 2)

Inference of Data Requested

Each hypothesis set is composed of three hypotheses, i.e. positive hypothesis (PH), negative hypothesis (NH), and irrelevant hypothesis (IH).



$$(Score_p > \boldsymbol{T}) \wedge (Score_p > Score_i) \wedge (Score_p > Score_n)$$

* We performed a small-scale preliminary study of Google in-house SDKs and through it, we adopted 0.73 as the optimal threshold (***T***) to achieve the minimal false negative rate.

Data type & class†	Count	Data type & class†	Count
(C1) Chip, Cellular & Peripheral		Wifi	38
Call phone	41	Wifi MACs(U)	5
Carrier info	5	*Subtotal*	*(245)*
Cellbroadcast	2	**(C3) Location & Sensors**	
External storage	18	Camera	54
ICCID(U)	15	Location(P)‡	87
IMEI(U)	5	Misc. sensors	140
IMSI(U)	3	*Subtotal*	*(281)*
MEID(U)	2	**(C4) Media & Software Specific**	
Network info	6	Android ID(U)	2
Phone status	1	App list(P)	7
SD card serial(U)	4	Audio record	22
Serial(U)	4	Google Ad ID§	3
SIM info	7	Media location	1
SMS	23	OAID§	9
Telephone number(U)	12	Screen record	1
Subtotal	*(148)*	*Subtotal*	*(45)*
(C2) Wireless Communication		**(C5) Personal Data**	
Bluetooth	117	Account info(P)	6
Bluetooth call	6	Browser bookmarks	1
Bluetooth ID(U)	2	Calendar	2
Bluetooth MAC(U)	36	Clipboard data	2
IP address(P)	2	Contact list	9
SIP service (VOIP)	21	Contact log	2
SSID/BSSID(P)	17	*Subtotal*	*(22)*
Ultra-wideband	1	**Total**	**741**

† We use (U) and (P) to indicate if the data is considered a UUI and a PII.
‡ Includes coarse location that is determined by network, and fine location that is jointly determined by network and GPS.
§ Google Ad ID and OAID are app differentiable identifiers, and therefore, are not considered as PIIs.

Exploring Privacy Collection (Phase 3)

Configuring Taint Analysis

We resort to **static taint analysis*** to identify “What do SDKs collect?”

We recognize APIs that collect the previously identified privacy data as **source** methods and identify the APIs that could save or share those privacy data into public spaces as **sink** methods.

741 source methods are selected to cover the 41 identified privacy data

** We resort to the latest open-sourced taint analysis toolkit for Android apps named “AppShark” to perform our analysis.*

Exploring Privacy Collection (Phase 3)

Configuring Taint Analysis

46 methods are identified as the sink methods, with their functionalities covering ***network transfer***, ***file saving***, and ***writing into the system settings***.

All the sinks are considered public locations that an arbitrary third-party app can access*.

Purpose	Class name	Count
Network Transfer	`java.net.HttpURLConnection`	1
	`java.net.URL`	2
	`java.net.URLConnection`	2
	`org.android.spdy.SpdyRequest`	12
	`org.android.spdy.SpdySession`	3
	`org.apache.http.client.methods.HttpGet`	8
	`org.apache.http.client.methods.HttpPost`	8
	`org.apache.http.params.HttpParams`	1
File Saving	`java.io.Writer`	5
	`java.io.FileOutputStream`	1
System Settings	`android.provider.Settings`	3
Total		**46**

** We remark that some access actions may need a non-dangerous permission (e.g., access Internet, read public directories, etc)*

Evaluation

Our evaluation aims to address three research questions:

RQ1: Can our privacy policy analysis identify the privacy data claimed to be collected by SDKs in a large scale? What is the performance of our NLI model-based approach?

RQ2: Can our taint analysis find privacy data collection and sharing behaviors from the SDKs? How many tainted traces are detected? Can those tainted traces (really) leak users' privacy?

RQ3: By cross-checking their privacy disclosure, what is the status quo of personal data protection at SDK-level? Do the collected SDKs comply with their privacy policies?

Evaluation
RQ1 Privacy Policy Analysis

Our analysis managed to identify 214 data requests from privacy policies.

Data type & class	#SDKs detected to collect	TP	FP	FN	Precision	Recall	F1-score
C1 Carrier info	12	10	2	3	83.3%	76.9%	80.0%
Device identifiers*	21	21	0	3	100.0%	87.5%	93.3%
Network info	30	25	5	0	83.3%	100.0%	90.9%
SIM card info	7	7	0	0	100.0%	100.0%	100.0%
SMS	7	3	4	0	42.9%	100.0%	60.0%
Telephone No	5	3	2	2	60.0%	60.0%	60.0%
C2 BSSID/SSID	4	4	0	0	100.0%	100.0%	100.0%
IP address	47	45	2	1	95.7%	97.8%	96.8%
WiFi	4	3	1	0	75.0%	100.0%	85.7%
C3 Camera	7	5	2	0	71.4%	100.0%	83.3%
C3 Location	28	25	3	4	89.3%	86.2%	87.7%
Misc sensors	1	1	0	0	100.0%	100.0%	100.0%
Android ID	8	8	0	0	100.0%	100.0%	100.0%
App list	4	4	0	0	100.0%	100.0%	100.0%
Audio record	10	10	0	0	100.0%	100.0%	100.0%
Google Ad ID	6	6	0	0	100.0%	100.0%	100.0%
OAID	2	2	0	0	100.0%	100.0%	100.0%
C4 Account info	10	4	6	4	40.0%	50.0%	44.4%
Clipboard	1	1	0	0	100.0%	100.0%	100.0%
Total	**214**	**187**	**27**	**17**	**87.4%**	**91.7%**	**89.5%**

- The **IP address (21.9%)**, **network information (14.0%)**, and **locations (13.0%)** are the most requested privacy data.
- Our NLI model achieves a promising precision of **87.4%** and F1-score of **89.5%**.

Evaluation

RQ2 Privacy Collection and Potential Leakages at SDK-Level packages

Through analyzing the 158 SDKs, we manage to identify 2,318 occurrences of privacy data collection in their implementation.

- The **location and sensor data (C3, 43.2%)** are the most **read** privacy data type.
- **34 (21.5%)** analyzed SDKs contain at least one trace tainted.
- **346** sharing actions were observed from the 158 SDKs.
- The **wireless communication data (C2, 49.7%)** are the most **shared** privacy data type.
- **338 out of 346 (97.7%)** tainted traces are logically valid traces (Type I issue).

	Caught access traces		
Data type & class	**Not tainted**	**Tainted (Type I)**	**Total**
C1 Carrier info	106	0	106
IMEI	92	26	118
MEID	0	4	4
IMSI	15	5	20
ICCID	20	4	24
Network info	2	0	2
Serial	82	5	87
SIM card info	14	0	14
SMS	6	0	6
Telephone no	33	0	33
Subtotal	*(370)*	*(44)*	*(414)*
C2 Bluetooth	20	0	20
Bluetooth MAC	10	0	10
BSSID/SSID	220	107	327
IP Address	23	1	24
WiFi	21	0	21
C2 WiFi MAC	62	64	126
Subtotal	*(356)*	*(172)*	*(528)*
C3 Camera	3	0	3
Location	486	130	616
Misc sensors	383	0	383
Subtotal	*(872)*	*(130)*	*(1002)*
C4 Android ID	83	0	83
App list	128	0	128
Audio record	11	0	11
Google Ad ID	20	0	20
OAID	68	0	68
Subtotal	*(310)*	*(0)*	*(310)*
C5 Account info	2	0	2
Clipboard data	57	0	57
Contact list	5	0	5
Subtotal	*(64)*	*(0)*	*(64)*
Total	**1972**	**346**	**2318**

Evaluation

RQ3 SDK-Level Privacy Assessment

We resort to cross-checking the privacy policy with the caught data collection behaviors of the collected SDKs in their implementation.

- Excessive collection of users' privacy (**36.6%**) and over-claiming collected data types in privacy policies (**88.1%**) are prevailing issues.
- **App list, clipboard data**, and **location** are the top 3 excessively collected data types .
- **IP address**, **IMEI**, and **Google Ad ID** are the top 3 data types over-claimed by SDKs.

Data type and class	Type II (EC)	Type III (OC)	Data type and class	Type II (EC)	Type III (OC)	Data type and class	Type II (EC)	Type III (OC)
C1 Carrier info	10	7	**C2** Bluetooth	4	0	**C4** Android ID	10	0
IMEI	1	6	Bluetooth MAC	2	0	App list	22	1
MEID	0	21	BSSID/SSID	8	2	Audio record	0	9
IMSI	1	18	IP Address	2	44	Google Ad ID	2	3
ICCID	4	16	WiFi	3	0	OAID	0	3
Network info	0	24	WiFi MAC	2	2	Screen record	0	1
Serial	1	6	*Subtotal*	*(21)*	*(48)*	*Subtotal*	*(34)*	*(17)*
SIM card info	1	0	**C3** Camera	0	4	**C5** Account info	1	4
SMS	3	2	Location	11	14	Clipboard data	11	0
Telephone no	1	3	Misc. sensors	7	0	Contact list	1	10
Subtotal	*(22)*	*(103)*	*Subtotal*	*(18)*	*(18)*	*Subtotal*	*(13)*	*(14)*
						Total	**108**	**200**

System Settings Injection Issues

Findings and Impact Analysis

System settings injection issues can be exploited by malicious SDKs when it is embedded in a pre-installed app or system app.

Successful exploitation can bypass the privacy mechanism of Android OS and share a UUI in the public space.

Our reserve engineering identifies 4 exploits of system setting injection issues from 2 SDKs.

SDK Name	Entry Name	Description of the value
Alicloud Push	dxCRMxhQkdGePGnp	IMEI/MEID, serial, or Android ID, AES encrypted, base64 encoded
	mqBRboGZkQPcAkyk	IMEI/MEID, serial, or Android ID, AES encrypted, base64 encoded
Baidu LBS	com.baidu.deviceid.v2	App's package name and Android ID, MD5 encrypted
	com.baidu.uuid	App's package name and serial or a random UUID, MD5 encrypted

The caught 2 SDKs have been found embedded in pre-install apps of devices from 9 OEM manufacturers, reserving over 94% Android smartphone market share in China.

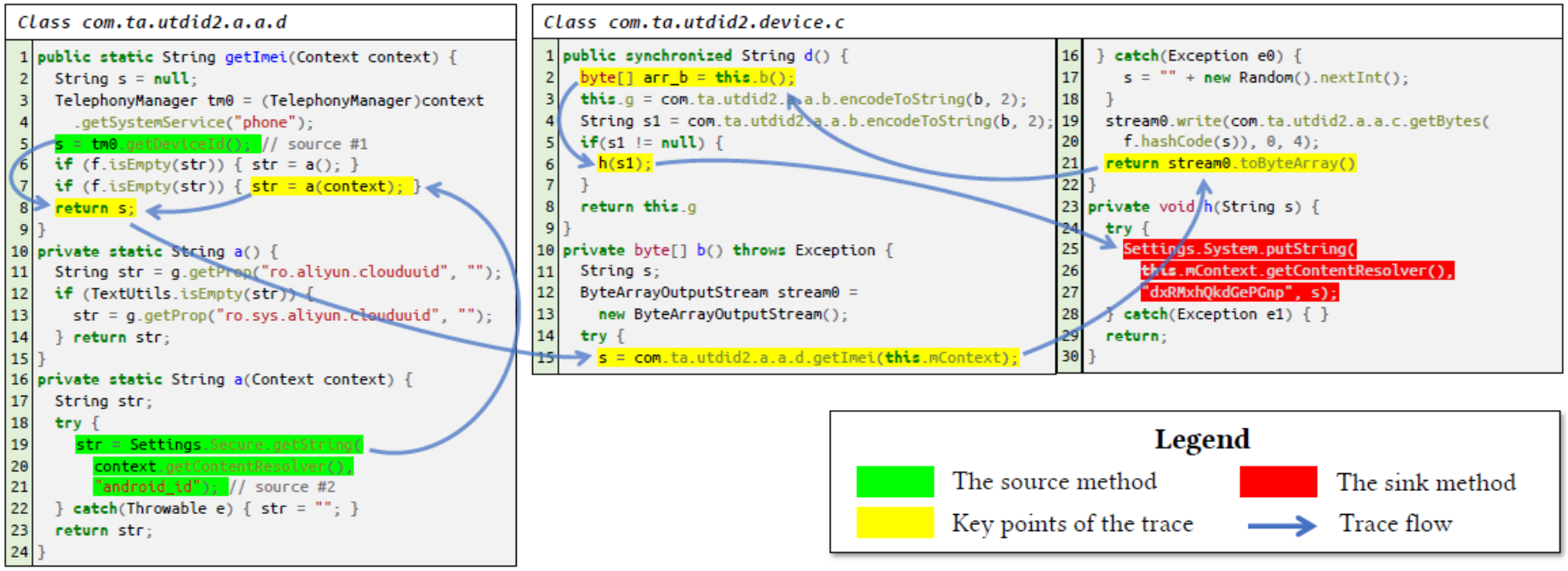
System Settings Injection Issues

A Real-World Example

System settings injection issues can be exploited by malicious SDKs when it is embedded in a pre-installed app or system app.

It alters the temporal validity and visibility of an app-specific resettable ID to a *de facto* UUI.

Successful exploitation can bypass the privacy mechanism of Android OS and share a UUI in the public space.



Privacy Compliance Re-Inspection

Based on our large-scale analysis of the 158 SDKs collected in October 2022, we re-visit the two sources (Google SDK Index and CAICT) and collect the same group of SDKs in their latest release version as of October 2023.

- **35 fewer traces** that attempt to collect and share privacy data.
- A drift of data collection pattern in data categories.
- Our concern about excessive privacy collection has not eased amidst the evolution of SDKs.

Data type & class	Changes in tainted traces	Data type & class	Changes in tainted traces
C1 Carrier info	▲ +31 (new)†	**C2** *Subtotal*	▼ *-106 (-61%)*
IMEI	▼-17 (65%)	**C3** Camera	▲ +2 (new)
MEID	▼-2 (50%)	Location	▼-94 (72%)
IMSI	▼-2 (40%)	Misc sensors	▲ +20 (new)
ICCID	● +0 (0%)	*Subtotal*	▼ *-72 (55%)*
Serial	▲ +3 (60%)	**C4** Android ID	▲ +34 (new)
SIM card info	▲ +1 (new)	App list	▲ +53 (new)
SMS	▲ +4 (new)	Audio record	▲ +5 (new)
Telephone no	▲ +2 (new)	Google Ad ID	▲ +9 (new)
Subtotal	▲ *+20 (45%)*	OAID	▲ +3 (new)
C2 Bluetooth	▲ +5 (new)	Screen record	▲ +1 (new)
Bluetooth MAC	▲ +3 (new)	*Subtotal*	▲ *+105 (new)*
BSSID/SSID	▼-73 (68%)	**C5** Clipboard data	▲ +17 (new)
IP Address	▲ +9 (900%)	Contact list	▲ +1 (new)
WiFi	▲ +9 (new)	*Subtotal*	▲ *+18 (new)*
WiFi MAC	▼-59 (92%)	**Total**	▼-35 (10%)

† We use "new" to indicate data types observed re-inspection for the first time.

Contributions

- Detecting potential privacy leakage in a large-scale
- Revelation of landscape of privacy protection at the SDK level
  - 338 potential privacy leakages detected from 158 SDKs
  - Less than 70% of the examined SDKs provide privacy policies
  - Approximately 37% of SDKs are found indulging in the over-collection of privacy data, 88% of SDKs falsely claim their data request.
- Amplified privacy risk caused by pre-installed privileged

Future Research Directions

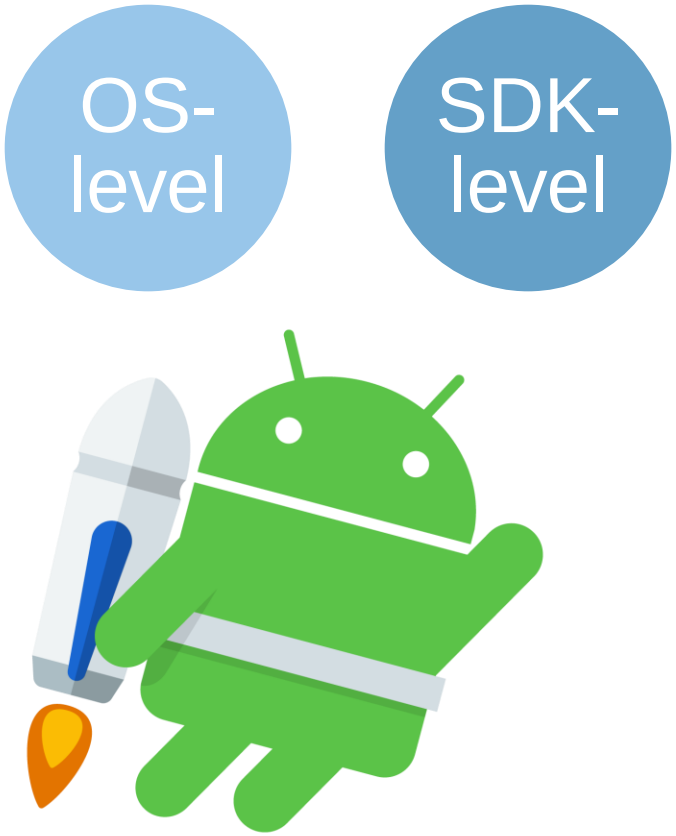
- **Exploring the latest causal LLMs in privacy policy analysis**
  - Latest causal LLMs (e.g., Llama, GPT) are more powerful in natural language comprehension.
  - Finding the optimal threshold may no longer needed and can be replaced by prompt engineering.

- **Improvement of the static analysis**
  - Complement with dynamic analysis to minimize false positives

Conclusion

Today's talk includes two research works:

1) We investigated whether the operating systems (OSes) themselves comprehensively safeguard users' personal data
2) We performed a large-scale measurement of the third-party SDKs' privacy preservation and their compliance with the ever-strengthening privacy protection regulations



Ongoing Work

Mobile Security & Privacy

- App signature leakage

Software Engineering for Trustworthy AI/LLM

- The broad sense of glitch tokens and hallucination of large generative AI models
- “How reasoning LLMs reason” — Comprehensive evaluation of reasoning quality

Selected References of the 1st Work

W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, “Taintdroid: an information- flow tracking system for realtime privacy monitoring on smartphones,” ACM Transactions on Computer Systems (TOCS), vol. 32, no. 2, pp. 1–29, 2014.

L. Qiu, Z. Zhang, Z. Shen, and G. Sun, “Apptrace: Dynamic trace on android devices,” in 2015 IEEE International Conference on Commu- nications (ICC), 2015, pp. 7145–7150.

J. Ren, M. Lindorfer, D. J. Dubois, A. Rao, D. Choffnes, and N. Vallina- Rodriguez, “A longitudinal study of pii leaks across android app versions,” in The Network and Distributed System Security Symposium (NDSS), 2018.

Z. Wang, Z. Li, M. Xue, and G. Tyson, “Exploring the eastern frontier: A first look at mobile app tracking in china,” in International Conference on Passive and Active Network Measurement. Springer, 2020, pp. 314–328.

Selected References of the 2nd Work

B. Andow, S. Y. Mahmud, J. Whitaker, W. Enck, B. Reaves, K. Singh, and S. Egelman, “Actions Speak Louder than Words: Entity-Sensitive Privacy Policy and Data Flow Analysis with PoliCheck,” in USENIX Security, 2020, pp. 985–1002.

W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, “Taintdroid: an information- flow tracking system for realtime privacy monitoring on smartphones,” ACM TOCS, vol. 32, no. 2, pp. 1–29, 2014.

Y. He, B. Hu, and Z. Han, “Dynamic privacy leakage analysis of android third-party libraries,” in ICDIS, 2018, pp. 275–280.

X. Liu, J. Liu, S. Zhu, W. Wang, and X. Zhang, “Privacy Risk Analysis and Mitigation of Analytics Libraries in the Android Ecosystem,” TMC, vol. 19, no. 5, pp. 1184–1199, 2020.

M. H. Meng, Q. Zhang, G. Xia, Y. Zheng, Y. Zhang, G. Bai, Z. Liu, S. G. Teo, and J. S. Dong, “Post-gdpr threat hunting on android phones: dissecting os-level safeguards of user-unresettable identifiers,” in NDSS, 2023.

L. Qiu, Z. Zhang, Z. Shen, and G. Sun, “Apptrace: Dynamic trace on android devices,” in 2015 IEEE ICC, 2015, pp. 7145–7150.

A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill, “Apps, Trackers, Privacy, and Regulators: A Global Study of the Mobile Tracking Ecosystem,” in NDSS, 2018.

L. Yu, X. Luo, J. Chen, H. Zhou, T. Zhang, H. Chang, and H. K. N. Leung, “PPChecker: Towards Accessing the Trustworthiness of Android Apps’ Privacy Policies,” TSE, vol. 47, p. 221–242, 2021.

K. Zhang and X. Wang, “Peeping tom in the neighborhood: Keystroke eavesdropping on multi-user systems,” in USENIX Security, 2009.

X. Zhou, S. Demetriou, D. He, M. Naveed, X. Pan, X. Wang, C. A. Gunter, and K. Nahrstedt, “Identity, location, disease and more: Inferring your secrets from android public resources,” in Proceedings of the 2013 ACM SIGSAC CCS, 2013, pp. 1017–1028.

Thank you

Feel free to contact me

mark-h-meng@github.io
mark.meng@tum.de