Post-GDPR Threat Hunting on Android Phones: Dissecting OS-level Safeguards of User-unresettable Identifiers (UUIs)

*Mark Huasong Meng (National University of Singapore), Qing Zhang (ByteDance), Guangshuai Xia (ByteDance), Yuwei Zheng (ByteDance), , Yanjun Zhang (Deakin University, Australia), Guangdong Bai (University of Queensland, Australia), Zhi Liu (ByteDance), Sin G. Teo (Institute for Infocomm, A*STAR, Singapore), Jin Song Dong (National University of Singapore)*

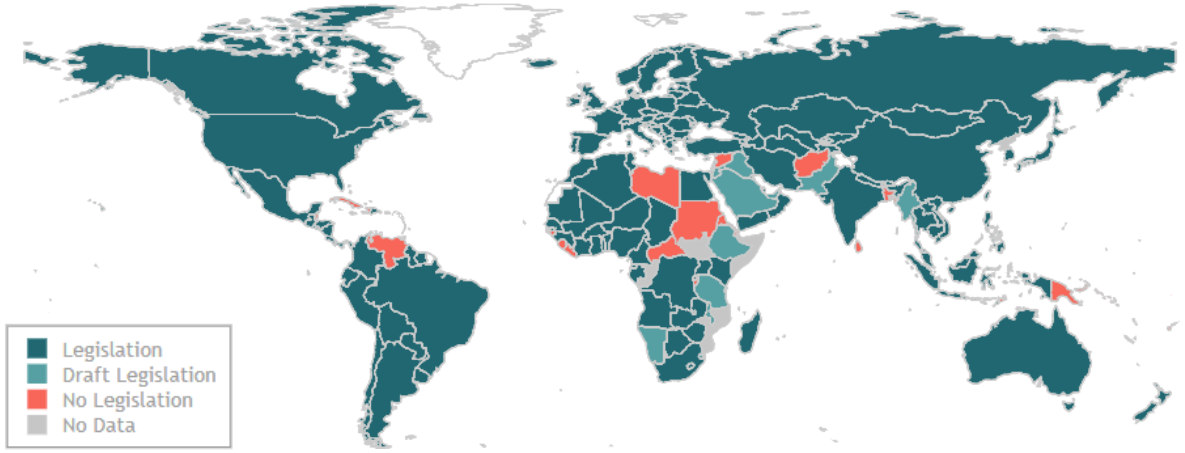
Background

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.

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



Data Protection and Privacy Legislation Worldwide (as of 2022)

Background

Android Ecosystem

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.

Motivation

OS-level UUI Safeguard

Personally identifiable data exfiltration in Android at 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 OS-level is still an open question.

Many types of personally identifiable data served at 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

List of 6 recognized Android UUIs

#	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.

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

Understanding Android UUIs

List of 6 recognized Android UUIs

#	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**.

Understanding Android UUIs

List of 6 recognized 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**.

Objective

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

Assessing Undocumented Channels

Step 1 - Access Channel Exploration

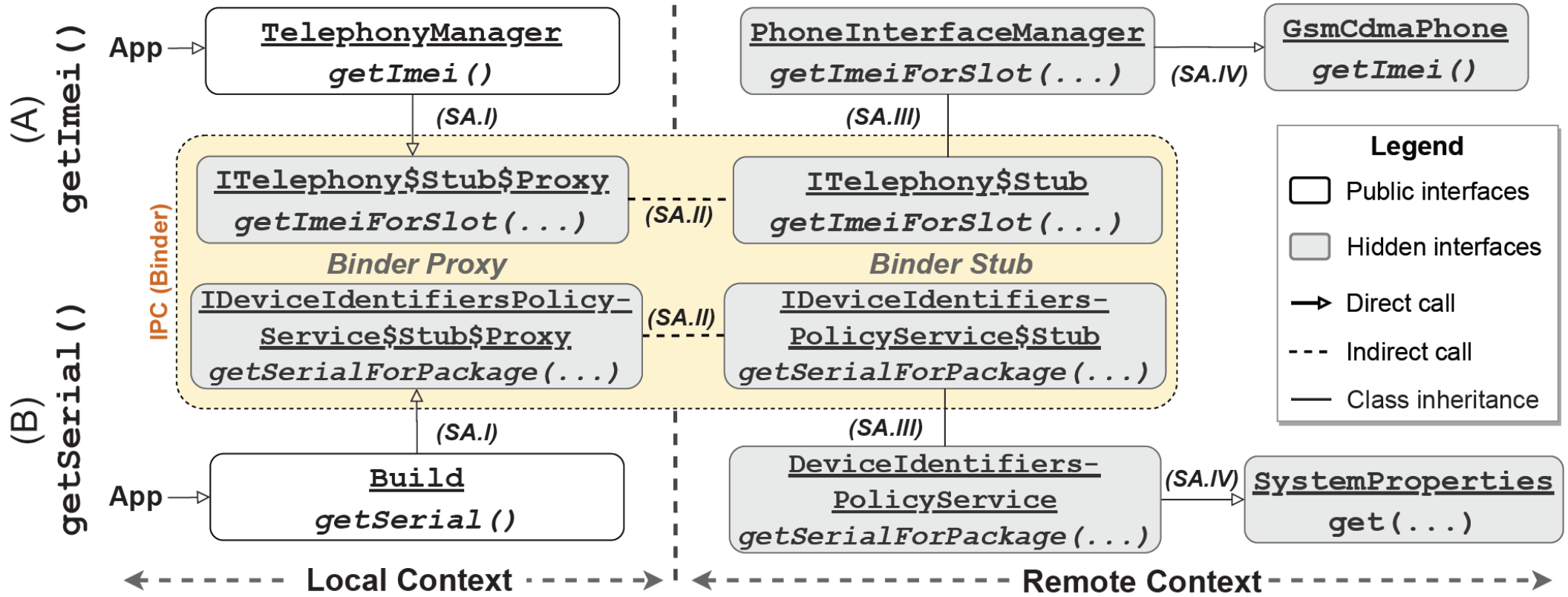
U2-I2 takes **six** pre-identified UUIs as seeds, considering that other unknown UUIs may share the same set of access channels.

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

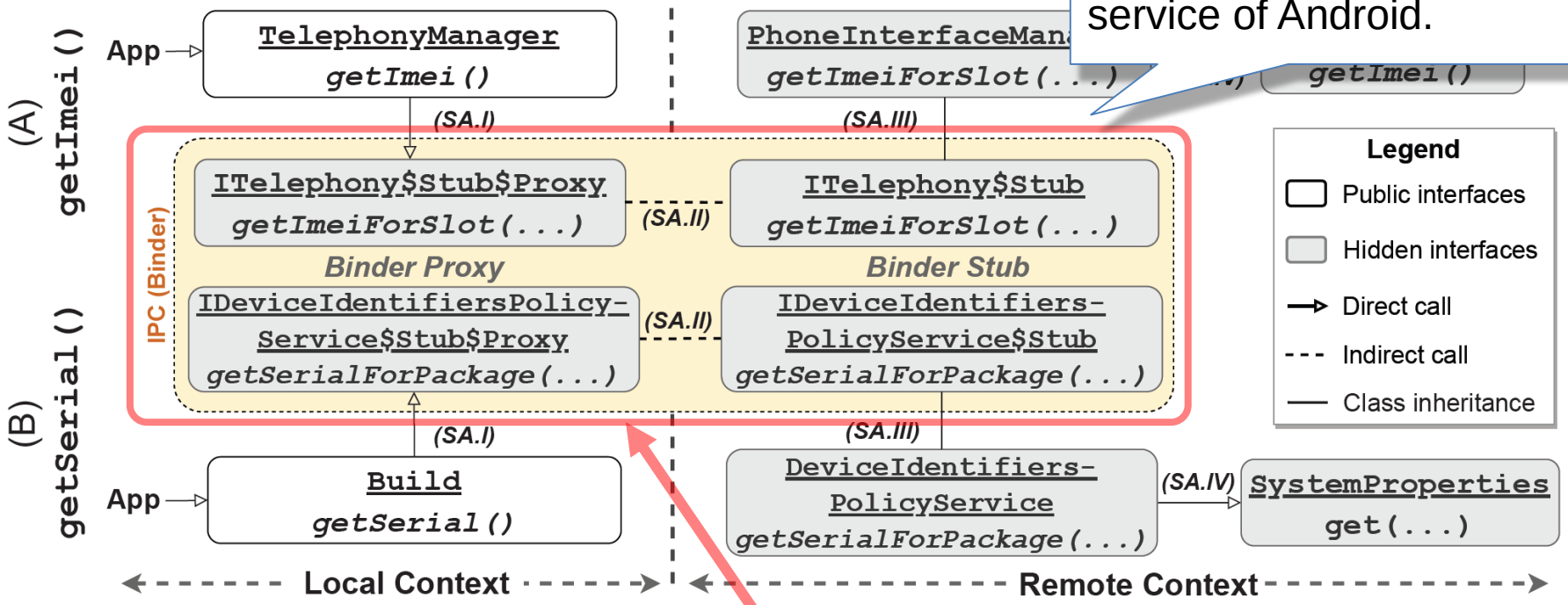
Two Typical Static Control Flows of the API Invocation



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.

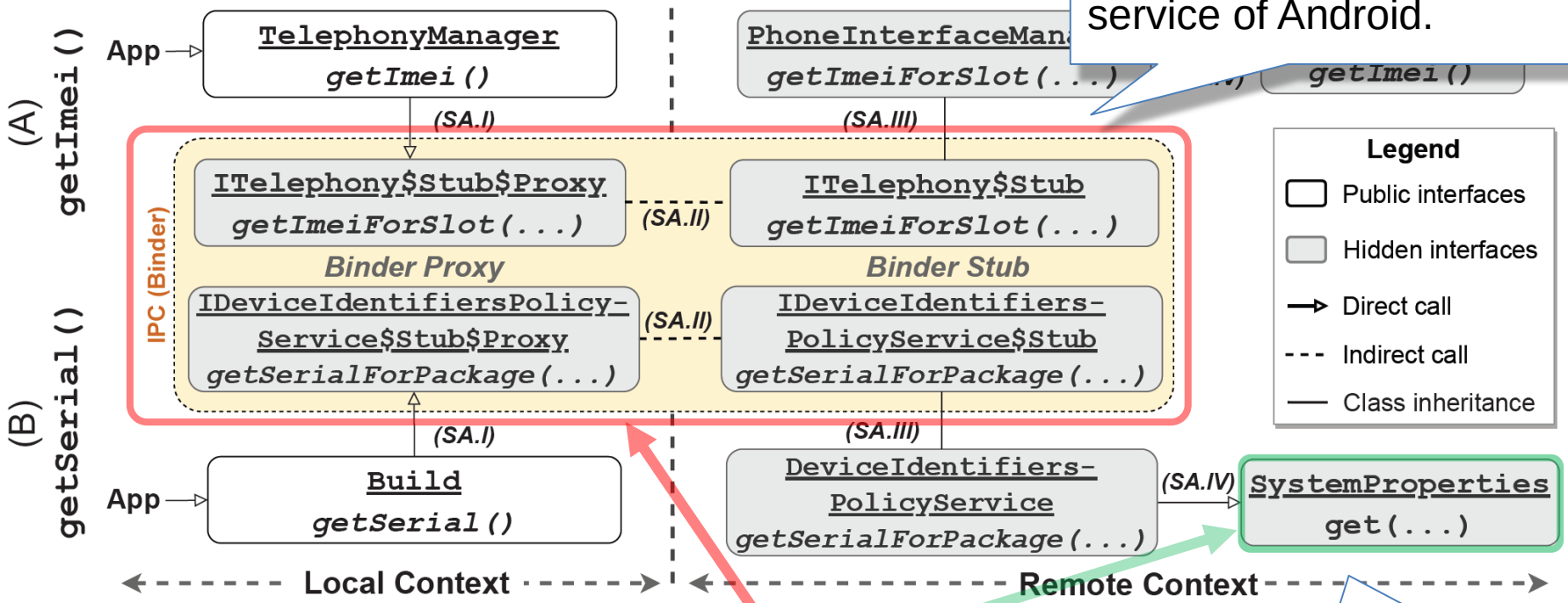


Visualization of two typical control flows starting from documented APIs.
*(**System properties** and **system services**)*

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

Step 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

Step 2 – Retrieving Entry Points and Testing

U2-I2 first retrieves entry points through the three undocumented channels, then tests through an app installed on the devices.

U2-I2 makes use of public interfaces to query **system properties** and **system settings**

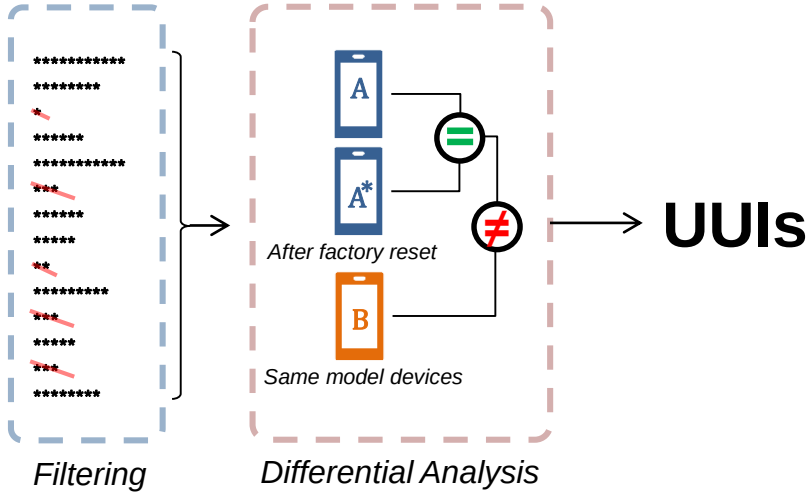
U2-I2 resorts to a “hacking way” through Java reflection to bypass the permission check to invoke **system services**.

Assessing Undocumented Channels

Step 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



Key Finding 1

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 1

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

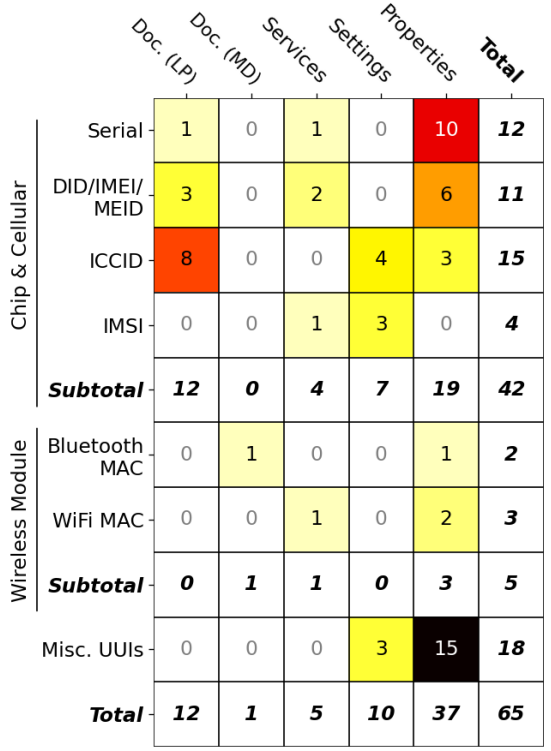
+ The UUIs labelled as “unknown type” contain insufficient information in their names and values.

Key Finding 2

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 3

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

- We have reported all our findings to the relevant parties/stakeholders.
- We also have kept them confidential for at least 90 days for them to be mended before we reported them in this paper.
- 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

Conclusion

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

Contacts

Should you have any questions, please feel free to contact us:

Guangdong Bai, UQ (g.bai@uq.edu.au)

Jin Song Dong, NUS (dcsdjs@nus.edu.sg)

*Sin Gee Teo, A*STAR I2R (teo_sin_gee@i2r.a-star.edu.sg)*

Qing Zhang, ByteDance (security@bytedance.com)

References

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 Communications (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.