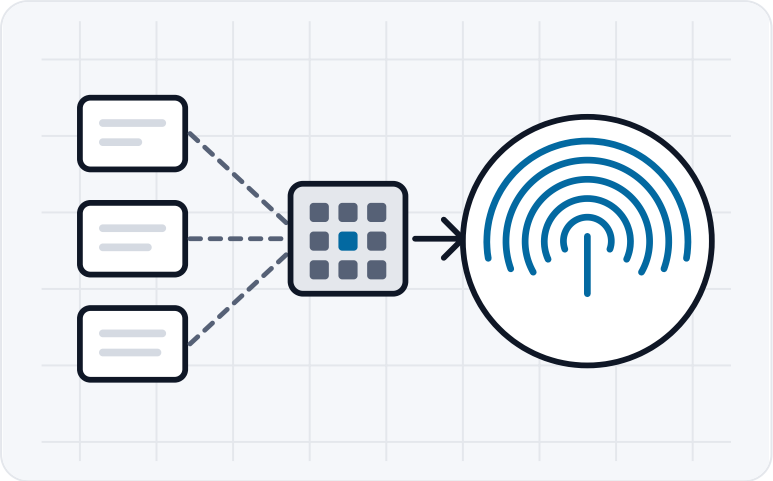
A-SIT Technology
E-Government Innovationszentrum

30. September 2024 Analyse · Mobile Sicherheit · Privatsphäre & Krypto · 11 Min. Lesezeit

Analyse von Fingerprinting-Bibliotheken auf Android

Gerald Palfinger



Autorinnen und Autoren	Gerald Palfinger
Stand	2024-09-30
Quelle	https://technology2.a-sit.at/veroeffentlichungen/analyse-von-fingerprinting-bibliotheken-auf-android/

Kernaussagen

- TrustDevice Android kombiniert Informationen aus 45 Quellen, FingerprintJS-Android aus 48.
- Beide Bibliotheken lesen Hard- und Softwaredetails, Nutzereinstellungen, die Liste installierter Apps und Akkudaten über eine versteckte Schnittstelle aus.
- TrustDevice Android sendet die Daten an ein Backend; FingerprintJS-Android berechnet den Fingerabdruck lokal per MurmurHash3 (128 Bit).
- Mit AndroPROTECT gepatchte Demo-Apps lieferten auf dem Testgerät denselben Fingerabdruck wie auf dem Referenzgerät.

Um eine Nutzerin beziehungsweise einen Nutzer über verschiedene Applikationen hinweg wiedererkennen zu können, kann ein Fingerabdruck des verwendeten Geräts erstellt werden. Dazu werden oft darauf spezialisierte Software-Bibliotheken verwendet, welche der Applikation zugängliche Daten aus verschiedenen Informationsquellen sammeln. Die gesammelten Informationen werden daraufhin kombiniert um daraus einen Fingerabdruck zu berechnen. Dieser kann direkt lokal am Gerät oder durch Übermittlung der gesammelten Daten an ein Backend berechnet werden. Im Rahmen des Projekts wird untersucht, welche Informationen Fingerprinting-Bibliotheken auf Android verwenden, um einen Fingerabdruck zu erstellen. Dazu wird untersucht, welche Methoden aufgerufen und welche Informationsquellen abgerufen werden. Darüber hinaus wird untersucht, wie die Bibliotheken die Daten verwenden, um den Fingerabdruck zu berechnen. Abschließend wird versucht, durch Änderungen der Programmierschnittstelle das Erstellen eines Fingerabdrucks zu verhindern.

1. Einleitung

In der heutigen digitalen Welt ist der Schutz der Privatsphäre von Nutzerinnen und Nutzern mobiler Anwendungen von entscheidender Bedeutung. Mobile Geräte, insbesondere Smartphones, sind aufgrund ihrer ständigen Konnektivität, der Verfügbarkeit sensibler Daten und der vielfältigen Interaktionsmöglichkeiten zu einem zentralen Ziel für Datensammler und Werbenetzwerke geworden. Eine Technik, die dabei zunehmend an Bedeutung gewinnt, ist das Fingerprinting. Fingerprinting ermöglicht es, ein Gerät und dadurch die Nutzerin beziehungsweise den Nutzer durch die Sammlung spezifischer Informationen und Eigenschaften eindeutig zu identifizieren, ohne dass eindeutige Identifizierungsmerkmale verwendet werden müssen. Dies stellt eine Herausforderung für den Datenschutz dar, da Fingerprinting schwerer zu erkennen und zu verhindern ist. Das Ziel dieses Berichts ist es, verschiedene Fingerprinting-Bibliotheken, die in Android-Applikationen verwendet werden, zu analysieren und ihre Funktionsweise zu untersuchen. Darüber hinaus soll aufgezeigt werden, welche Gegenmaßnahmen existieren, um Fingerprinting auf Android-Geräten zu verhindern beziehungsweise zu minimieren.

2. Hintergrund

Das Fingerprinting von Smartphones, wie es durch Applikationen durchgeführt wird, ist eine Technik, bei der spezifische Informationen über ein Gerät gesammelt werden, um es eindeutig zu identifizieren. Dies geschieht in der Regel ohne direkte Abfrage oder Nutzung von einzigartigen Identifikatoren wie der IMEI-Nummer oder der MAC-Adresse, insbesondere, da diese auf neueren Android-Versionen nicht mehr von Drittprogrammen abrufbar sind. Stattdessen verwenden Applikationen beziehungsweise darauf spezialisierte Bibliotheken eine Vielzahl von Datenpunkten, die zusammen einen einzigartigen Fingerabdruck des Geräts erzeugen. In [1] wurde gezeigt, dass die Erstellung eines Fingerabdrucks unter Android anhand von durch Applikationen abrufbare Informationen möglich ist. In [2] wurden verschiedene Android Applikationen auf ihre Fingerprintaktivitäten untersucht. Dabei wurde festgestellt, dass Werbe- und spezialisierte Trackingbibliotheken, welche in den Applikationen eingebunden sind, eine Vielzahl an Informationen abrufen, die zur Erstellung eines Fingerabdrucks verwendet werden können. In [3] wird systematisch untersucht, welche Informationsquellen unter Android zur Verfügung stehen, um einen

Fingerabdruck zu erstellen. Daraufhin wurde in [4] AndroPROTECT vorgestellt. Dabei handelt es sich um ein Framework, welches anhand der in [3] erkannten Informationsquellen automatisiert Patches erstellt, welche durch Änderung der abrufbaren Informationen die Fingerprintbarkeit von Android-Smartphones verringern.

3. Untersuchung

In diesem Abschnitt werden die beiden Fingerprinting-Bibliotheken TrustDevice Android [5] und FingerprintJS-Android [6] näher beleuchtet. Dazu wurde der Quellcode der Bibliotheken untersucht. Im Folgenden wird darauf eingegangen, welche Informationen von den beiden Bibliotheken gesammelt werden und wie diese verwendet werden, um daraus einen Fingerabdruck zu berechnen.

3.1. Sammlung der Rohdaten zur Erstellung des Fingerabdrucks

Eine Übersicht über alle gesammelten Rohdaten ist in Tabelle 1 ersichtlich. Dabei kann festgestellt werden, dass einige der Informationsquellen von beiden Bibliotheken abgerufen werden. Insgesamt kombiniert TrustDevice Android die Informationen aus 45 Quellen, während FingerprintJS-Android Datenpunkte aus insgesamt 48 Informationsquellen abruft. Zu den Informationsquellen die beide Bibliotheken abrufen gehören die Felder in der android.os.Build Klasse. Diese beinhalten unter anderem Informationen über das verwendete Gerätemodell, den Hersteller des Geräts und Softwareversionen. Dabei liest TrustDevice Android mehr Felder dieser Klasse aus als Fingerprintjs-Android. Darunter fällt zum Beispiel der Produktname oder die Build ID des Betriebssystems, die nur von TrustDevice Android abgerufen werden. Zusätzlich zu den Informationen in der Build-Klasse lesen beide Bibliotheken weitere Hardwaredetails aus. Darunter fällt zum Beispiel die Größe des verbauten Arbeitsspeichers und des Gerätespeichers sowie die Liste der verfügbaren Gerätesensoren. Neben den Hard- und Softwaredetails des verwendeten Geräts sammeln beide Bibliotheken Informationen über durch den Nutzer beziehungsweise die Nutzerin gesetzte Einstellungen. So werden Informationen über die gesetzte Sprache, das verwendete Gebietsschema sowie Land und Zeitzone ausgelesen. Ebenso eruieren beide Bibliotheken weitere Einstellungen, wie die Verwendung des Entwicklungsmodus, den Aktivierungsstatus verschiedener Bedienungshilfen, die verwendete Eingabemethode und Netzwerkeinstellungen. Viele dieser Informationen werden aus den Settings Content Providern [7] abgerufen, welche Zugriff auf eine Vielzahl von gesetzten Einstellungen bieten. Zusätzlich versuchen beide Bibliotheken auch die Liste der installierten Applikationen sowie Systemapplikationen abzurufen.

Manche der gesammelten Informationen werden nicht direkt über die Programmierschnittstelle sondern über das Dateisystem abgerufen. Insbesondere werden beispielsweise die Informationen zum verbauten Prozessor wie Name, Architektur und Anzahl der CPU-Kerne aus der Datei /proc/cpuinfo ausgelesen.

Die Informationen über den Akku, wie beispielsweise die Kapazität oder den Gesundheitszustand, werden über eine versteckte Programmierschnittstelle abgerufen. Dabei handelt es sich um Methoden beziehungsweise Klassen der Programmierschnittstelle, die nicht für die Verwendung durch Drittanbieteranwendungen gedacht sind. Sie sind jedoch nicht komplett für Drittanbieteranwendungen gesperrt und können beispielsweise über Reflection aufgerufen werden.

Weitere Informationsquellen werden nur von einer der beiden Bibliotheken abgerufen. So sammelt TrustDevice Android im Vergleich zu Fingerprintjs-Android auch die physische Größe des Bildschirms, die verwendete Bildschirmauflösung und Bildschirmhelligkeit. Ebenso wird der Pfad des Dateispeichers abgerufen. Darüber hinaus ruft TrustDevice Android Informationen zu Akkulevel und Akkutemperatur sowie

den verfügbaren Arbeitsspeicher und internen Speicher ab. Im Vergleich dazu ruft Fingerprintjs-Android weitere Informationen zu Hardwaredetails ab, wie die Liste der verbauten Kameras, die höchste unterstützte OpenGLES-Version und verfügbare Eingabegeräte ab. Die Bibliothek eruiert darüber hinaus auch Details zu den verfügbaren Mediencodes, Sicherheitsprovider und den Status des Fingerabdrucksensors. Ebenso ruft die Bibliothek auch weitere durch die Nutzerin beziehungsweise den Nutzer beeinflussbare Einstellungen ab, darunter das verwendete Datums- und Uhrzeitformat, die Schriftgröße, Verwendung der Autovervollständigung, sowie die Dateipfade des Klingeltons und des Alarmtons.

Zusätzlich zu den gesammelten Daten zur Fingerabdruck-Erstellung werden von den beiden Bibliotheken noch weitere Identifikationsmerkmale abgerufen. FingerprintJS-Android sammelt hierbei den entwicklerspezifischen ANDROID_ID, den APK-spezifischen Media DRM-Identifier (auch Widevine ID genannt) und den Google Services Framework Identifier. TrustDevice Android sammelt ebenso den ANDROID_ID, den Advertising Identifier und den Media DRM-Identifier sowie den Google Service Framework Identifier. Darüber hinaus sammelt TrustDevice Android auch den (durch den Nutzer beziehungsweise die Nutzerin deaktivierbaren) Google Advertising Identifier und den firmwarespezifischen VBMeta Digest [8].

TrustDevice Android sammelt zusätzlich zu den oben genannten Informationen noch Informationen über den Rooting-Status des Geräts. Dazu wird abgerufen, ob es sich bei dem installierten Betriebssystem um einen Debug-Build handelt, ob Root-Binärdaten gefunden wurden, ob das Rooting-Tool Magisk oder der Applikationspatcher Xposed installiert ist, oder ob Hooks erkannt wurden. Darüber hinaus wird abgefragt, ob es sich bei dem Gerät um einen Emulator handelt oder ob ein VPN verwendet wird.

3.2. Erstellung des Fingerabdrucks

TrustDevice Android bereitet die gesammelten Datenpunkte als JSON-Struktur auf und sendet diese an ein Backend, um daraus einen Fingerabdruck zu berechnen. Dadurch war es nicht möglich zu eruieren, wie aus den gesammelten Rohdaten der Fingerabdruck erstellt wird. Anders als bei TrustDevice Android wird bei FingerprintJS Android der Fingerabdruck direkt auf dem Gerät erstellt. Dazu werden die einzelnen Datenpunkte zu einer Zeichenfolge verknüpft. Diese wird dann in eine Hashfunktion gegeben, die daraus einen 128-bit großen Fingerabdruck erstellt. Dazu wird die Hashfunktion MurmurHash3 [9] verwendet. Dabei handelt es sich um eine schnelle Hashfunktion, die für den Einsatz in Hashtabellen oder Bloomfilter entwickelt wurde. Hierbei sei angemerkt, dass dieser Hash nicht für kryptographische Anwendungen geeignet ist. Die gesammelten Informationen über den ANDROID_ID, den Media DRM-Identifier und den Google Services Framework Identifier wird von FingerprintJS Android nicht in die Kalkulation des Fingerabdrucks miteinbezogen.

Tabelle 1 Übersicht über die abgefragten Eigenschaften der beiden untersuchten Fingerprinting Bibliotheken.

Eigenschaft	TrustDevice	Fingerprintjs-Android
Gerätemodell	✓	✓
Hersteller	✓	✓
Android Version	✓	✓
SDK Version	✓	✓
Kernel Version	✓	✓
Build Fingerprint	✓	✓
Hardwarename	✓	
Produktname	✓	
Marke	✓	
Build ID (Build.DISPLAY)	✓	
Build Host	✓	
Gesundheitszustand des Akkus	✓	✓
Status des Akkus	✓	
Akkulevel	✓	
Akkutemperatur	✓	
Kapazität des Akkus	✓	✓
Prozessortyp	✓	✓
Prozessor	✓	✓
Anzahl der CPU-Kerne	✓	✓
Prozessorarchitektur	✓	✓
Land	✓	✓
Sprache	✓	✓
Zeitzone	✓	✓
Gesamter Arbeitsspeicher	✓	✓
Verfügbarer Arbeitsspeicher	✓	
Gesamter Speicher	✓	✓
Verfügbarer Speicher	✓	

Eigenschaft	TrustDevice	Fingerprintjs-Android
Installierte Apps	✓	✓
Installierte Systemapplikationen	✓	✓
Sensoren	✓	✓
Android Debug Bridge aktiviert	✓	✓
Entwicklermodus aktiviert	✓	✓
HTTP Proxy gesetzt	✓	✓
Datenroaming aktiviert	✓	✓
Attrapenstandort aktiviert	✓	
Bedienungshilfen aktiviert	✓	✓
Standard Eingabemethode	✓	✓
Touch Exploration aktiviert	✓	✓
Bildschirmhelligkeit	✓	
Ausschaltzeit des Bildschirms	✓	✓
Bildschirmauflösung	✓	
Bildschirmgröße	✓	
Pfad des Dateispeichers	✓	
Name des Applikationspackages	✓	
Gerät verwendet Harmony OS	✓	
Eingabegeräte		✓
Liste der Kameralinsen		✓
Maximal unterstützte OpenGL ES Version		✓
Verschlüsselungsstatus des Speichers		✓
Liste der Mediencodes		✓
Sicherheitsprovider		✓
Animationsmaßstab der Übergänge (Bedienungshilfeneinstellung)		✓
Animationsmaßstab der Fenster (Bedienungshilfeneinstellung)		✓
Echtzeittext (RTT) Status		✓

Eigenschaft	TrustDevice	Fingerprintjs-Android
Dateipfad des Alarms		✓
Datumsformat		✓
Verhalten der Auflegen-Taste		✓
Schriftgröße		✓
Autovervollständigung für Text		✓
Automatische Interpunktion		✓
Uhrzeitformat (12/24h)		✓
Verwendung eines Geräte-PINs		✓
Status des Fingerabdrucksensors		✓
Pfad des gesetzten Klingeltons		✓
Verfügbare Gebietsschema		✓
Gesamtanzahl der verwendeten Informationsquellen	45	48

4. Verhinderung der Fingerabdruck-Erstellung

In diesem Abschnitt wird getestet, ob das AndroPROTECT-Framework [4], ein Framework zur Reduzierung der Fingerprintbarkeit von Android-Geräten, die Erstellung eines Fingerabdrucks verhindern kann. Dazu wird das durch AndroPROTECT erstellte Patch-Paket auf den Demo-Applikationen der zwei untersuchten Fingerprinting-Bibliotheken angewendet. Die gepatchten Anwendungen wurden auf einem Google Pixel 7a (als Testgerät) sowie auf einem Pixel 6 Pro (als Referenzgerät) mit Android 14 installiert. Anhand des Referenzgeräts wird zuerst mit AndroPROTECT ein Patch-Paket erstellt. Dazu durchforstet AndroPROTECT die Programmierschnittstelle des Geräts und ruft alle durch eine Drittanbieteranwendung abrufbaren Informationsquellen ab. Für Informationsquellen, welche potenziell fingerprintbar sind, werden daraufhin Patches erstellt, welche die abrufbaren Informationen vereinheitlichen. Diese werden in einem Patch-Paket zusammengefasst. Dieses Paket kann dann auf Android-Applikationspakete angewendet werden, um die abrufbaren fingerprintbaren Informationen zu verringern. Im Verlauf dieser Untersuchung wird das Patch-Paket auf die Demo-Applikationen der Fingerprinting-Bibliotheken angewendet und die gepatchte Applikation dann auf dem Testgerät installiert. Die Ergebnisse der Untersuchung sind im Folgenden dargelegt.

Die FingerprintJS Android Demo-Applikation berechnet eine lokale Gerätekennung auf Basis der auf dem Gerät gesammelten Informationen. In der Demo-Applikation können verschiedene Stabilitätsstufen bei der Erstellung des Fingerabdrucks ausgewählt werden. Abhängig von der gewählten Stabilitätsstufe wählt die Applikation unterschiedliche Informationsquellen für die Berechnung des Fingerabdrucks aus. Auf dem Testgerät berechnete die gepatchte Demo-Anwendung auf allen Stabilitätsstufen denselben Fingerabdruck

wie auf dem Referenzgerät. Dadurch zeigt sich, dass AndroPROTECT die Erstellung des Fingerabdrucks wirkungsvoll verhindern kann. Alle abgerufenen Informationsquellen, die sich von Gerät zu Gerät unterschieden, wurden von AndroPROTECT gepatcht.

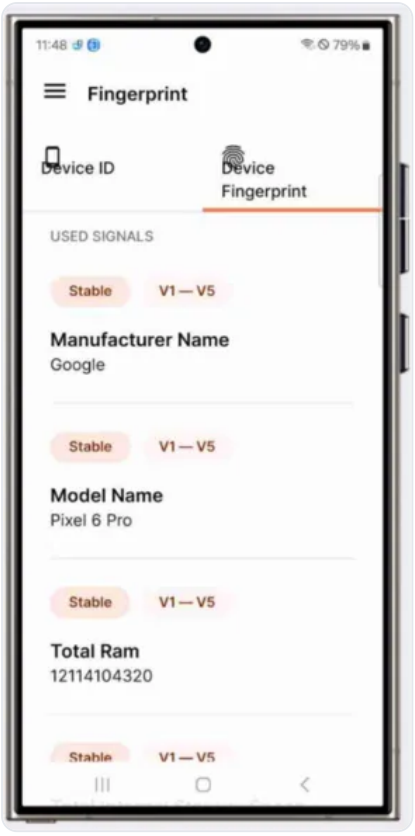


Abbildung 1: Gepatchte Demo-Applikation von Fingerprintjs-Android auf einem Samsung-Gerät. Die Abbildung zeigt, dass verwendete Datenquellen durch den Patch der Applikation erfolgreich geändert wurden.

Die TrustDevice-Demo-Applikation sammelt Geräteinformationen in einer JSON-Struktur, welche zur Berechnung des Fingerabdrucks an ein Backend übermittelt werden kann. Im Folgenden werden die JSON-Daten des Testgeräts mit denen des Referenzgeräts verglichen. Es konnte festgestellt werden, dass die gesammelten Daten auf beiden untersuchten Geräten identisch sind und somit keine Verwendung zur Erstellung eines unterscheidbaren Fingerabdrucks finden können. Zusätzlich zu den aus der Android API gesammelten Daten nutzt TrustDevice eine native Bibliothek, um den Root-Status des Geräts zu verifizieren. Derzeit werden native Binärdateien nicht von AndroPROTECT unterstützt. Da unsere Testgeräte jedoch nicht gerootet waren, unterschieden sich die von diesen Prüfungen zurückgegebenen Informationen nicht zwischen den Geräten. Zudem war das Verstecken des Root-Status nicht Ziel von AndroPROTECT, da es andere Lösungen wie beispielsweise MagiskHide [10] gibt.

5. Fazit

In diesem Bericht wurden Fingerprinting-Bibliotheken für Android Applikationen untersucht. Dazu wurden zwei Bibliotheken ausgewählt, die näher beleuchtet wurden. Dabei wurde festgestellt, dass beide Bibliotheken Informationen aus einer ähnlichen Anzahl an Quellen sammeln, um daraus einen Fingerabdruck des Geräts zu erstellen. Dabei gibt es eine gewisse Überschneidung der benutzten Informationsquellen, vor allem was grundlegende Informationen wie Software- und Hardwaredetails sowie

gewisse durch den Nutzer beziehungsweise die Nutzerin beeinflussbare Einstellungen betrifft. Während TrustDevice Android die gesammelten Daten zur Erstellung des Fingerabdrucks an ein Backend sendet, berechnet Fingerprintjs-Android den Fingerabdruck lokal am Gerät durch Hashing der konkatenierten Werte. Abschließend wurde gezeigt, dass durch das Patchen der Applikationen die Erstellung eines unterscheidbaren Fingerabdrucks verhindert werden konnte.

Referenzen

[1] W. Wu, J. Wu, Y. Wang, Z. Ling und M. Yang, „Efficient Fingerprinting-Based Android Device Identification With Zero-Permission Identifiers," IEEE Access, pp. 8073 - 8083, 2016.

[2] C. F. Torres und H. Jonker, „Investigating Fingerprinters and Fingerprinting-Alike Behaviour of Android Applications," 23rd European Symposium on Research in ComputerSecurity - ESORICS 2018, pp. 60 - 80, 2018.

[3] G. Palfinger und B. Prünster, „AndroPRINT: Analysing the Fingerprintability of the Android API," ARES 2020: The 15th International Conference on Availability, Reliability and Security, pp. 94:1 94:10, 2020.

[4] G. Palfinger, „AndroPROTECT: Hardening the Android API against Fingerprinting," NSS 2024: 18th International Conference on Network and System Security, 20 11 2024.

[5] TrustDecision, „GitHub - trustdecision/trustdevice-android: Leading open source version of android device fingerprint, accurate deviceID and risk identification," 2022. [Online]. Available: https://github.com/trustdecision/trustdevice-android. [Zugriff am 26 09 2024].

[6] „GitHub - fingerprintjs/fingerprintjs-android: Swiss army knife for identifying and fingerprinting Android devices," 2020-2023. [Online]. Available: https://github.com/fingerprintjs/fingerprintjsandroid. [Zugriff am 26 09 2024].

[7] Android Developers, „Settings," 2024. [Online]. Available: https://developer.android.com/reference/android/provider/Settings. [Zugriff am 26 09 2024].

[8] Google Inc, „The VBMeta Digest - Android Verified Boot 2.0," [Online]. Available: https://android.googlesource.com/platform/external/avb/+/master/README.md#the-vbmetadigest. [Zugriff am 26 09 2024].

[9] A. Appleby, „MurmurHash3 Quellcode - Github," 09 01 2016. [Online]. Available: https://github.com/aappleby/smhasher/blob/master/src/MurmurHash3.cpp. [Zugriff am 25 09 2024].

[10] J. Wu, „GitHub - topjohnwu/Magisk: The Magic Mask for Android," [Online]. Available: https://github.com/topjohnwu/Magisk. [Zugriff am 23 09 2024].

[11] Google Inc., „Android API reference | Android Developers," 02 03 2024. [Online]. Available: https://developer.android.com/reference. [Zugriff am 14 06 2024].

[12] Google Inc., „Content Providers | Android Developers," 23 01 2024. [Online]. Available: https://developer.android.com/guide/topics/providers/content-providers. [Zugriff am 14 06 2024].

[13] Google Inc., „SDK Runtime Overview - Process Isolation,“ 10 06 2024. [Online]. Available: https://developers.google.com/privacy-sandbox/relevance/sdk-runtime#process-isolation. [Zugriff am 12 06 2024].

[14] „Best practices for unique identifiers | Android Developers,“ 23 04 2024. [Online]. Available: https://developer.android.com/identity/user-data-ids. [Zugriff am 14 06 2024].