A-SIT

Zentrum für sichere Informationstechnologie - Austria

Vergleich der Fingerprintbarkeit von Android und iOS

Vergleich der Fingerprintbarkeit von Android und iOS

Autor:
Gerald Palfinger
Tel: +43 316 873 -
Mail: gerald.palfinger@a-sit.at

Datum: 27.04.2023

Abstract/Zusammenfassung:
Das Tracking von Nutzerinnen und Nutzern über verschiedene Anwendungen hat es Anwendungsanbietern ermöglicht, die große Mehrheit ihrer Anwendungen kostenlos anzubieten, da es ihnen erlaubt Werbung gezielt einsetzen können. Das Tracking hat sich jedoch auch als Eingriff in die Privatsphäre der Nutzer erwiesen. Um dem entgegenzuwirken, haben die Entwickler von Betriebssystemen den Zugang zu eindeutigen Identifikatoren in ihren Programmierschnittstellen beschränkt. Dennoch können Anwendungen immer noch andere nicht eindeutige Informationen des Geräts kombinieren, um einen eindeutigen Fingerabdruck zu erstellen. In diesem Bericht wird untersucht, welche Informationsquellen dafür unter Android und iOS verfügbar sind, welche Arten von Informationen auf beiden Betriebssystemen vorhanden sind und welche sich unterscheiden.

Inhalt

1. Einleitung

Sowohl unter Android als auch unter iOS ist die umfassende Verfolgung von Nutzerinnen und Nutzern über verschiedene mobile Anwendungen hinweg üblich [1]. Dieses Tracking wird von Anwendungen und ihren Werbeanbietern genutzt, um mehr über ihre Nutzerinnen und Nutzer zu erfahren und gezieltere Werbung zu schalten. Da über 90 % der Anwendungen kostenlos angeboten werden, ist die weltweit führende Methode zur Monetarisierung von Anwendungen die Anzeige von Werbung in der Anwendung [2]. Dies ermöglicht es den Anwendungsanbietern zwar, ihre Anwendungen den Nutzerinnen und Nutzern kostenlos anzubieten, aber das Tracking, das zur Bereitstellung relevanter Werbung verwendet wird, beeinträchtigt in hohem Maße die Privatsphäre der Nutzer [3]. Um dem entgegenzuwirken, hat Apple in den letzten Versionen von iOS Maßnahmen ergriffen, um das Tracking durch Drittanbieteranwendungen und deren Tracking-Bibliotheken einzuschränken. Entwickler müssen nun die Zustimmung der Nutzerin beziehungsweise des Nutzers mit Hilfe des App-Tracking-Transparenzrahmens einholen [4]. Außerdem müssen sie angeben, welche Arten von Informationen sie zum Tracking der Nutzerinnen und Nutzer verwenden. Diese Änderungen verhindern zwar technisch gesehen die Erfassung

der eindeutigen Werbekennung, wenn keine Zustimmung erteilt wurde, doch hat eine kürzlich durchgeführte Studie [5] gezeigt, dass eine große Zahl von Anwendungen immer noch andere Daten vom Betriebssystem erfasst oder sogar zu erfassen beginnt, die zur Erstellung eines Fingerabdrucks des Geräts verwendet werden können. Eine ähnliche Einschränkung des Zugriffs auf eindeutige Identifikationsmerkmale wurde auch unter Android mit Version 10 eingeführt [6]. Mit der Beschränkung des Zugriffs auf eindeutige Kennungen wird die Technik des Fingerprinting von Geräten also immer relevanter. In Browsern beispielsweise, in denen es keine eindeutigen Identifikationsmerkmale gibt, die zur Identifizierung von Nutzerinnen und Nutzern verwendet werden können, ist das Fingerprinting bereits weit verbreitet [7]. Im Wesentlichen zielt das Browser-Fingerprinting darauf ab, so viele eindeutige Daten über die Umgebung wie möglich zu sammeln, um die Benutzerinnen und Benutzer zu unterscheiden. Eine ähnliche Methodik kann sowohl unter iOS [8] als auch unter Android [9] angewendet werden, um Geräte wiederzuerkennen. Welche Arten von Informationen dazu vorhanden sind und ob beziehungsweise, wie sich diese zwischen den beiden Betriebssystemen unterscheiden wurde jedoch noch nicht systematisch überprüft. Das wird in diesem Bericht angegangen.

2. Verwandte Arbeiten

In [8] wurde gezeigt, dass es möglich ist, Fingerabdrücke von iOS-Geräten anhand von Werten zu erstellen, die von der Programmierschnittstelle stammen. Die Autoren wählten 29 Informationsquellen manuell aus und erstellten eine Anwendung, die Daten aus diesen Quellen von Geräten verschiedener Nutzerinnen und Nutzern sammelte. Obwohl die aus diesen Informationsquellen gewonnenen Werte für sich genommen nicht eindeutig waren, zeigten ihre Experimente, dass es durch die Kombination der Werte möglich war, die Geräte in ihrem Datensatz mit hoher Genauigkeit erneut zu identifizieren. Ähnlich wie in dieser Studie wurde auch in [9] gezeigt, dass die eindeutige Identifizierung eines Geräts durch eine Kombination von nicht eindeutigen Informationen auch unter Android möglich ist. Die Autoren wählten 38 verschiedene Informationsquellen aus, die sie wiederum von unterschiedlichen Geräten sammelten. Zur erneuten Identifizierung der Geräte aus den gewonnenen Datensätzen verwendeten sie verschiedene Algorithmen und verglichen anschließend deren Genauigkeit. Dabei zeigten sie, dass eine erneute Identifikation mit hoher Wahrscheinlichkeit möglich ist.

Im Gegensatz zu den beiden vorangegangenen Studien, bei denen eine Reihe von Informationsquellen von den Autoren ausgewählt wurde, wurde in [10] die Arten von Informationen untersucht, die von realen Tracking-Bibliotheken auf Android verwendet werden. Um diese Bibliotheken zu finden, stützten sie sich auf die Liste der Informationsquellen, die in [9] bereitgestellt wurde. Zusätzlich erweiterten sie diese um die Informationsquellen, die von zwei bereits bekannten Fingerprinting-Bibliotheken verwendet werden. Mit Hilfe dieser Informationsquellen fanden sie mithilfe einer statischen Analyse sechs ähnliche Bibliotheken, die Fingerabdrücke von Geräten erstellen. Darüber hinaus kamen sie zu dem Schluss, dass für das Fingerprinting von Smartphones andere Techniken verwendet werden als für das Fingerprinting von Browsern. Ein systematischerer Ansatz wurde in [11] gezeigt. Dazu wurde ein Framework namens AndroPRINT entwickelt, um Daten aus der Android-API zu sammeln. Das Framework bezieht Daten aus Feldern, Methoden und den Android-spezifischen Content Providern. Mithilfe von AndroPRINT wurde gezeigt, dass eine große Anzahl dieser Quellen fingerprintbare Informationen liefern. Darüber hinaus hat die Analyse gezeigt, dass Herstelleranpassungen persönliche Daten preisgeben und zusätzliche Informationsquellen für das Fingerprinting bereitstellen können, einschließlich einiger einzigartiger und benutzerspezifischer Identifikatoren. In [12] wurde dieses Prinzip auf das Betriebssystem iOS übertragen. Mit Hilfe eines automatisierten Frameworks wurde nach fingerprintbaren Informationsquellen in der Programmierschnittstelle des Betriebssystems gesucht. Es wurde ebenso gezeigt, dass es eine große Anzahl an Quellen gibt, die sich für das Fingerprinting eignen können. Auf Basis dieser beiden

Frameworks wird in diesem Bericht verglichen, welche Informationsquellen auf beiden Betriebssystemen existieren und welche nur auf einem der beiden Betriebssysteme gefunden werden konnte.

3. Methodik

Es wurde eine Smartphone-Anwendung sowohl für Android als auch für iOS erstellt, um systematisch Daten zu sammeln, die für Anwendungen von Drittanbietern über die Programmierschnittstelle verfügbar sind. Mithilfe von Reflection instanziiert die Anwendung alle Klassen auf dem Gerät. Für jedes der erstellten Objekte werden alle Instanzmethoden der Klasse aufgerufen und die Rückgabewerte gespeichert. Außerdem werden alle Felder bzw. Properties der instanziierten Klassen abgerufen. Schließlich wird mithilfe des Klassen-Deskriptors jede Klassenmethode aufgerufen. Mit der Smartphone-Anwendung werden Daten von mehreren Geräten gesammelt. Um zu simulieren, dass zwei verschiedene Anwendungsentwickler die Daten sammeln, führen wir die Smartphone-Anwendung zweimal aus, wobei unter iOS unterschiedliche Entwicklerprofile und Bundle-Identifikatoren und unter Android unterschiedliche Signierschlüssel und Paketnamen verwendet werden. Anschließend bereitet das Framework die gesammelten Daten für die geräte- und plattformübergreifende Analyse auf. Dazu werden die Daten bereinigt und einheitlich formatiert. Schließlich werden die Daten analysiert, um fingerprintbare Informationsquellen in den Programmierschnittstellen zu erkennen. Zunächst werden Daten derselben Informationsquelle, die sich zwischen den beiden Durchläufen auf demselben Gerät unterscheiden, verworfen, da diese über verschiedene Anwendungshersteller beziehungsweise Applikationsneustarts hinweg nicht robust sind. Anschließend werden die verbleibenden Daten geräteübergreifend verglichen. Datenquellen, die sich zwischen den Geräten unterscheiden, werden als fingerprintbar betrachtet. Für den Vergleich über die beiden Betriebssysteme hinweg werden sie nach der Art der erhaltenen Information kategorisiert. In den folgenden beiden Sektionen wird darauf eingegangen, wie sich die Suche nach potentiell fingerprintbaren Informationen zwischen den beiden Betriebssystemen aufgrund von technischen Gegebenheiten unterscheidet. Danach wird auf die Analyse der gesammelten Daten eingegangen.

3.1. Android

Bevor Informationen aus der Programmierschnittstelle von Android extrahiert werden können, müssen gewisse Informationen aus der Dokumentation geparst werden, da diese Informationen nicht während der Ausführung der Smartphone-Applikation abgerufen werden können. Dabei handelt es sich um die Liste der Klassen der Programmierschnittstelle. Diese werden benötigt, um Objekte aller Klassen der Programmierschnittstelle erstellen zu können. Ebenso müssen die Namen von Methodenparametern aus der Dokumentation extrahiert werden, da diese während der Ausführung durch generische Namen wie *arg_0*, *arg_1*,... ersetzt werden.

Sind diese Informationen beschafft, so kann die eigentliche Sammlung der Daten durch die Smartphone-Applikation erfolgen. Dabei werden anhand der Liste der geparsten Klassen der Programmierschnittstelle Objekte dieser Klassen mit Hilfe von Reflection erstellt. Mit diesen Objekten werden daraufhin Methoden aufgerufen. Sollte eine Methode Parameter verlangen, so werden diese vorher von der Applikation erstellt. Nach dem Aufruf der Methode wird der Rückgabewert für die spätere Analyse gespeichert. Sollte ein nicht-primitives Objekt zurückgegeben werden, so wird dieses zusätzlich dafür verwendet, um weitere Methoden aufzurufen.

Neben dem Aufruf von Methoden werden auch alle zugänglichen Felder der Programmierschnittstelle abgefragt. Für die Abfrage werden ebenso die für den Methodenaufruf erzeugten Objekte und die nicht-

primitiven Rückgabewerte der Methoden verwendet. Der Wert dieser Felder wird ebenso für die spätere Analyse gespeichert.

Speziell unter Android gibt es das Konzept der Content Provider. Dabei handelt es sich um eine abstrakte Schnittstelle für den Zugriff auf Daten. Ähnlich wie bei einer Datenbank können die Inhalte in Form von Zeilen und Spalten mit einem Cursor abgefragt werden. Dazu ist eine Content-URI notwendig. Um die Content-URIs der vom System bereitgestellten Content-Provider zu erhalten, analysieren wir jedes abgefragte Feld und jeden zurückgegebenen String-Wert während der Methodenaufrufe. Wenn eine Zeichenkette mit dem Präfix *content://* beginnt, wird die URI für den späteren Abruf gespeichert. Sobald die Methodenaufrufe und Feldauswertungen abgeschlossen sind, wird der Inhalt aller erkannten und zugänglichen Content Provider durch Iteration über alle Zeilen und Spalten ausgelesen. Ähnlich wie die Rückgabewerte und Felder werden die ausgelesenen Daten für die spätere Analyse gespeichert.

3.2. iOS

Das Framework zur Sammlung der Rohdaten unter iOS ist in zwei Hauptkomponenten unterteilt: das Backend und die Smartphone-Anwendung. Im Wesentlichen ruft die Smartphone-Anwendung Daten aus der Programmierschnittstelle ähnlich wie unter Android über Reflection ab. Dabei werden wie unter Android automatisch Methoden aufgerufen und Properties (ähnlich der Felder unter Android) ausgelesen. Aufgrund des Low-Level-Charakters von Objective-C können fehlgeschlagene Methodenaufrufe oder Objektinstanziierungen jedoch zu Speicherproblemen führen, die den Abbruch der Smartphone-Anwendung zur Folge haben. Um dieses Problem zu beheben, überwacht das Backend die Ausführung der Smartphone-Anwendung und startet sie bei Bedarf neu. Methoden, Klassen und Properties, die zu dem Problem führten werden vor dem Neustart vom erneuten Aufruf, von der erneuten Instanziierung beziehungsweise vom erneuten Auslesen ausgeschlossen. Ähnlich wie unter Android ruft das Backend im Voraus einige spezifische Informationen ab, die während der Laufzeit nicht verfügbar sind. Im Fall von iOS handelt es sich hierbei um den konkreten Typ von Methodenparametern, die nicht zur Laufzeit eruiert werden können. Schließlich überträgt das Backend die gesammelten Daten von der Smartphone-Anwendung zur Analyse. Unter iOS werden diese aus dem Systemprotokoll ausgelesen, was eine weitere Strukturierung der gesammelten Daten notwendig macht.

Das Backend speichert die Ausgabe der Smartphone-Applikation in das Systemprotokoll vorerst unverändert. Im gespeicherten Protokoll werden alle Ausgaben der Smartphone-Anwendung beibehalten, zum Beispiel auch die Ausgaben von aufgerufenen Methoden. Daher müssen die gesammelten Daten vor der Analyse bereinigt werden. Dies wird von unserer Analyseanwendung durchgeführt. Sie sammelt die Rückgabewerte aus dem Systemprotokoll und verwirft überflüssige Ausgaben. Außerdem konvertiert sie die Ergebnisse in ein vordefiniertes Format. Dieses wurde als CLASS.METHOD:::RESULT oder CLASS.PROPERTY:::RESULT definiert und stimmt damit der Ausgabe der Android-Anwendung überein. Da das Ergebnis Zeilenumbrüche enthalten kann, werden diese wie unter Android durch die Zeichenkette -NEWLINE- ersetzt. Mit diesen Schritten können die Ergebnisse über verschieden Geräte verglichen werden.

3.3. Analyse der Rohdaten

Die Datenerfassung wird auf jedem Smartphone zweimal durchgeführt. Dazwischen wird die Smartphone-Anwendung entfernt und das Gerät neu gestartet. Nach dem Entfernen der Anwendung wird diese mit einem anderen Signierzertifikat und einer anderen Bundlekennung neu installiert. Anschließend wird die zweite Erfassungsphase gestartet. Dieser aufwendige Prozess wird durchgeführt, um das Sammeln von Daten durch Anwendungen verschiedener Entwickler zu simulieren. Auf diese Weise werden die Informationsquellen eliminiert, die sich zwischen zwei verschiedenen Anwendungen unterscheiden und daher nicht für die Erstellung eines anwendungsübergreifenden Fingerabdrucks geeignet sind. So werden zum Beispiel falsch-positive Informationen wie das Installationsverzeichnis der

Anwendung eliminiert, das einen zufälligen Wert enthält, der für jede Anwendung unterschiedlich ist. Außerdem werden entwicklerspezifische Bezeichner wie die das UIDevice.identifierForVendor Property unter iOS oder der Settings.Secure.ANDROID_ID Wert auf Android mit diesem Ansatz ebenfalls ausgeschlossen. Schließlich werden bei dieser Maßnahme auch alle temporären Werte eliminiert, die sich aufgrund eines Anwendungsneustarts oder eines Geräteneustarts ändern könnten. Alle verbleibenden Werte, das heißt diejenigen, die sich zwischen den beiden Durchläufen auf demselben Gerät nicht unterscheiden, werden dann gespeichert. Diese werden für alle untersuchten Geräte gesammelt und im nächsten Schritt, der Analyse, verglichen.

Die gesammelten, bereinigten und geordneten Werte eines Geräts werden bei der Analyse mit allen anderen untersuchten Geräten verglichen. Methoden, die nur auf einem Gerät aufgerufen werden konnten, oder Properties bzw. Felder, die nur auf einem Gerät abgerufen werden konnten, werden entfernt, da kein Vergleichswert vorhanden ist. Alle verbleibenden Werte, das heißt diejenigen, die auf mindestens zwei Geräten abgerufen werden konnten, werden dann automatisch verglichen. Alle Werte, die auf allen analysierten Geräten identisch sind, werden entfernt. Alle Werte, die sich auf mindestens einem Gerät unterscheiden, werden beibehalten und ihre Quelle wird als potenziell fingerprintfähig gekennzeichnet. Diese Werte werden dann manuell überprüft und kategorisiert, um sie zwischen iOS und Android vergleichbar zu machen. Die Ergebnisse dieser Analyse sind im nächsten Kapitel zu finden.

4. Ergebnisse

Zur Sammlung der fingerprintbaren Informationsquellen wurden die erstellten Smartphone-Anwendungen auf Android- und iOS-Geräten ausgeführt. Unter Android wurden dabei Pixel 4-Geräte verwendet, während unter iOS Geräte des Typs iPhone 11 verwendet wurden. In Tabelle 1 sind die Ergebnisse der Auswertung ersichtlich. Die gefundenen Quellen wurden in stabile und unstabile Quellen eingeteilt. Die unstabilen Quellen können sich beispielsweise mit einem Neustart des Geräts ändern. Da Smartphones tagelang oder sogar wochenlang in Betrieb sein können, könnten solche Informationen immer noch zur Erstellung von Fingerabdrücken eines Nutzers verwendet werden.

Viele der gefundenen Kategorien an Informationsquellen sind sowohl unter Android als auch unter iOS verfügbar. Zu den unstabilen Quellen, die auf beiden Betriebssystemen verfügbar sind, gehören Netzwerkinformationen und Informationen zum Bootvorgang. Unter Android sind jedoch wesentlich mehr Informationen über verbundene Netzwerke verfügbar. Unter iOS ist der Bootzeitpunkt abrufbar, während unter Android die Anzahl der Bootvorgänge eruiert werden kann.

Darüber hinaus wurde eine wesentlich größere Anzahl an stabileren Informationsquellen auf beiden Betriebssystemen gefunden. Zu den auf beiden Betriebssystemen vorhandenen Quellen gehören Informationen über die Größe von verschiedenen Elementen der Benutzeroberfläche sowie Informationen über die Bildschirmauflösung. Ebenso kann über eine Vielzahl an Quellen die Sprache sowie verschiedene Daten über das Gebietsschema eruiert werden. Das Datum- & Zeitformat kann auch auf beiden Systemen abgerufen werden. Ebenso verfügbar sind Informationen über die eingestellte Währung. Auf beiden Systemen vorhanden sind verschiedene Einstellungen zur Barrierefreiheit, die entweder die Benutzeroberfläche beeinflussen oder direkt abgerufen werden können. Einstellungen die sich auf die Farben der Benutzeroberfläche beziehungsweise das Aussehen dieser auswirken, können ebenso überall erkannt werden. Auch der verwendete Netzwerkbetreiber ist auf beiden Betriebssystemen abrufbar. Informationen über die Eingabemethode können sowohl auf Android als auch auf iOS abgerufen werden. Darüber hinaus können gewisse Details über Systemapplikationen eruiert werden. Manche Einstellungen zum Systemsound und Do-not-Disturb können ebenso abgerufen werden. Der

durch den Benutzer beziehungsweise die Benutzerin definierte Gerätename konnte auf beiden Systemen erkannt werden.

Details über die verwendete Tastatur sowie Tastatureinstellungen konnten nur auf iOS abgerufen werden. Die Modellnummer sowie die Farbe des Geräts konnten ebenso nur unter iOS erkannt werden. Das Nutzerpersona sowie weitere Identifikationsmerkmale gab es nur auf iOS. Unter Android wurden keine direkten Identifikationsmerkmale erkannt. Der Aktivierungsstatus von personalisierter Werbung sowie UI Effekte konnten auch nur unter iOS erkannt werden.

Auch unter Android gab es Informationsquellen, welche so nicht unter iOS erkannt werden konnten. Dazu zählen verschiedene Bluetooth-Einstellungen, aktivierte Netzwerke, Klingeltöne und Benachrichtigungseinstellungen. Verschiedene Anzeigeeinstellungen, Soundeffekte und Informationen über die verwendete Gerätesperre sowie die Passwortkomplexität konnten nur auf Android gefunden werden. Darüber hinaus konnten spezifische Androidfunktionen erkannt werden, wie Widgets, Bildschirmschoner oder der aktivierte persönliche Assistent. Darüber hinaus konnten konkrete Timestamps sowie der Anpassungsstatus erkannt werden.

Tabelle 1 Gefundene Informationsquellen unter Android und iOS beim Vergleich von Geräten desselben Modells (CP = Content Provider).

Kategorie	Android			iOS	
	# Methoden	# Felder	# CP	# Methoden	# Properties
Unstabil					
Netzwerkinformationen	44	10	4	4	-
Bootzeitpunkt	-	-	-	2	-
Bootanzahl	-	-	2	-	-
Weiteres	-	1	-	-	-
TOTAL	**44**	**11**	**6**	**6**	**0**
Stabil					
UI Größe & Bildschirmauflösung	90	40	2	114	139
Sprache und Gebietsschema	180	8	3	89	31
Tastatureinstellungen	-	-	-	54	33
Einstellungen zur Barrierefreiheit	9	3	21	39	22
Datum & Zeitformat	2	1	-	16	21
UI Farben & Style	21	3	-	7	4
Modellnummer	-	-	-	5	5
Nutzerpersona und andere Identifikationsmerkmale	-	-	-	6	4
Währungsinformationen	22	-	-	5	5
Netzwerkbetreiber und MMS Einstellung	29	2	6	5	2
Eingabemethode	16	1	3	6	-
(Gelöschte) Systemapplikationen	6	-	1	6	-
Verschiedene Einstellungen	-	-	109	4	-

Farbe des Geräts	-	-	-	1	3
Applikationsinformationen	-	-	-	-	3
Weiteres	11	3	1	2	1
Personalisierte Werbung	-	-	-	1	-
UI Effekte	-	-	-	1	-
Systemsound	5	-	-	1	-
Nutzerdefinierter Gerätename	2		2	-	1
Bluetootheinstellungen	17	-	1	-	-
Aktivierte Netzwerke	23	-	2	-	-
Benachrichtigungseinstellungen	4	-	6	-	-
Anzeigeeinstellungen	5	1	8	-	-
Klingeltöne	5	1	-	-	-
Soundeffekte	-	-	6	-	-
Gerätesperre	3	-	-	-	-
Widgets	2	-	-	-	-
Timestamps	-	-	2	-	-
Passwortkomplexität	1	-	-	-	-
Anpassungsstatus	-	-	1	-	-
Bildschirmschoner	-	-	1	-	-
Aktivierter Assistent	-	-	1	-	-
TOTAL	**453**	**63**	**176**	**362**	**274**

5. Fazit

Für diesen Bericht wurden potenziell fingerprintbare Informationsquellen von Android- und iOS Geräten gesammelt. Die gesammelten Quellen wurden daraufhin anhand des Typs der erhaltenen Information kategorisiert. Dabei stellte sich heraus, dass viele Kategorien von Informationsquellen sowohl unter Android als auch unter iOS verfügbar sind. Viele der Informationsquellen, welche nur auf einem der beiden Betriebssysteme gefunden werden konnte, stammen von Funktionen, welche so nicht direkt auf dem anderen Betriebssystem vorhanden sind. Durch die Content Provider gibt es jedoch unter Android eine zusätzliche Quelle an fingerprintbarem Material, welche in dieser Form nicht unter iOS vorhanden ist. Da jede Methode nur mit einer begrenzten Anzahl an unterschiedlichen Parametern aufgerufen werden kann und da nicht jeder Methodenaufruf beziehungsweise jede Objektinstanziierung erfolgreich abgeschlossen werden konnte, ist es jedoch nicht möglich abschließend zu sagen auf welchem der beiden Betriebssysteme mehr fingerprintbare Informationsquellen vorhanden sind. Es gibt jedoch auf beiden Betriebssystemen eine große Anzahl an fingerprintbaren Informationsquellen, die es ermöglichen eine Nutzerin beziehungsweise einen Nutzer über verschiedene Applikationen hinweg wiederzuerkennen.

Referenzen

[1] K. Kollnig, A. Shuba, R. Binns, M. V. Kleek und N. Shadbolt, „Are iPhones Really Better for Privacy? A Comparative Study of iOS and Android Apps," *Proc. Priv. Enhancing Technol.,* pp. 6--24, 2022.

[2] Statista, „Mobile app monetization - Statistics & Facts," Statista, 4 8 2022. [Online]. Available: https://www.statista.com/topics/983/mobile-app-monetization/#topicOverview. [Zugriff am 24 04 2023].

[3] I. Shklovski, S. D. Mainwaring, H. H. Skúladóttir und H. Borgthorsson, „Leakiness and creepiness in app space: perceptions of privacy and mobile app use," *Conference on Human Factors in Computing Systems,* pp. 2347-2356, 2014.

[4] „App Tracking Transparency," Apple Inc., [Online]. Available: https://developer.apple.com/documentation/apptrackingtransparency/. [Zugriff am 27 04 2023].

[5] K. Kollnig, A. Shuba, M. V. Kleek, R. Binns und N. Shadbolt, „Goodbye Tracking? Impact of iOS App Tracking Transparency and Privacy Labels," *Conference on Fairness, Accountability, and Transparency,* pp. 508-520, 2022.

[6] Google, „ Privacy in Android 10," [Online]. Available: https://developer.android.com/about/versions/10/privacy/. [Zugriff am 25 04 2023].

[7] U. Iqbal, S. Englehardt und Z. Shafiq, „Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors," *Symposium on Security and Privacy,* pp. 1143-1161, 2021.

[8] A. Kurtz, H. Gascon, T. Becker, K. Rieck und F. C. Freiling, „Fingerprinting Mobile Devices Using Personalized Configurations," *PoPETs,* pp. 4--19, 2016.

[9] 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.

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

[11] G. Palfinger und B. Prünster, „AndroPRINT: Analysing the Fingerprintability of the Android API," *Availability, Reliability and Security - ARES,* pp. 94:1 - 94:10, 2020.

[12] G. Palfinger, „OCScraper: Automated Analysis of the Fingerprintability of the iOS API," *SECRYPT,* 2023.