A-SIT

Zentrum für sichere Informationstechnologie - Austria

Untersuchung der Fingerprintbarkeit von iOS-Geräten

A-SIT

Untersuchung der Fingerprintbarkeit von iOS-Geräten

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

Datum: 31.10.2022

Abstract/Zusammenfassung: In diesem Bericht wird untersucht, welche Informationsquellen die iOS-Programmierschnittstelle bietet um iOS-Geräte bzw. deren Nutzerinnen und Nutzer über verschieden Applikationen hinweg wiederzuerkennen. Dazu wird ein Framework verwendet, welche diese Daten automatisiert aus der iOS-Programmierschnittstelle ausliest. Zur Analyse werden die ausgelesenen Daten in ein strukturiertes Format umgewandelt. Dies ermöglicht den Vergleich der gesammelten Daten von unterschiedlichen Geräten. Für die Evaluierung wurden diese Daten auf verschiedenen iOS-Geräten gesammelt und analysiert. Im Bericht wird anhand der gesammelten Daten gezeigt, welche Arten von Informationen, die sich zur Erstellung eines Fingerabdrucks eignen können, über die Methoden und Properties der Programmierschnittstelle zugänglich sind.

Inhalt

1. Einleitung

Um Nutzerinnen und Nutzer über verschiedene Dienste hinweg wiederzuerkennen können verschiedene Eigenschaften des verwendeten Gerätes bzw. der vorgenommenen Einstellungen zu einem sogenannten Fingerabdruck kombiniert werden. Diese Technik wird bereits seit Jahren im Web verwendet. Doch auch auf Smartphones kann sie eingesetzt werden um Nutzerinnen und Nutzer über verschiedene Applikationen hinweg wiederzuerkennen. Dadurch kann die Privatsphäre der Nutzerin bzw. des Nutzers beeinträchtigt werden. Damit diese Technik jedoch überhaupt einsetzbar ist, muss eine ausreichende Anzahl an fingerprintbaren Informationsquellen vorhanden sein. Dies bedeutet auch, dass das Betriebssystem durch Maßnahmen verhindern bzw. erschweren kann einen solchen Fingerabdruck zu erstellen. Eine dieser Maßnahmen ist die Abschottung der einzelnen Applikationen durch die sogenannte Sandbox. Diese ist unter anderem dazu gedacht den direkten

Zugriff auf personenbezogene Daten ohne Einwilligung des Nutzers bzw. der Nutzerin als auch auf andere Daten des Systems zu verhindern. Viele nicht direkt personenbezogenen Daten eignen sich in der Summe jedoch dennoch um schlussendlich einen eindeutigen Fingerabdruck zu erstellen. Da diese Informationen auch von Applikationen abseits es Fingerprinting eingesetzt werden können muss eine gute Balance zwischen Reduzierung der Fingerprintbarkeit und Nutzen der Information gefunden werden. Um abschätzen zu können welche Informationsquellen sich zur Erstellung eines Fingerabdrucks unter iOS eignen, wurden für diesen Bericht in Experimenten möglichst viele Daten über die Programmierschnittstelle gesammelt. Als Grundidee dieser Experimente wurde festgelegt, dass sich Informationen aus Quellen, die sich über verschiedene iOS-Geräte unterscheiden potentiell fingerprintbar sind. Dabei wurde ein automatisiertes Framework verwendet um möglichst viele der für Applikationen zugänglichen Informationsquellen abzurufen. Diese Informationen wurden auf mehreren iOS-Geräten gesammelt. Nach der Aufbereitung der Informationen wurden diese analysiert. Informationsquellen, welche sich nicht unterschieden wurden entfernt, während Informationsquellen die sich unterschieden für die genauere Analyse behalten wurden. Diese wurden dann in Gruppen eingeordnet um so einen Überblick über die vorhandenen Informationsquellen geben zu können.

2. Verwandte Arbeiten

In [1] wird versucht Fingerabdrücke von iOS-Geräten zu erstellen. Dazu wurden von den Autoren 29 verschiedene Methoden bzw. Werte aus der iOS-Programmierschnittstelle manuell ausgewählt. Es wurde eine Applikation erstellt, um diese Werte auf verschiedenen iOS-Geräten zu sammeln um so einen Fingerabdruck zu erstellen. Bei der Auswertung wurde gezeigt, dass diese gesammelten Fingerabdrücke zu einem großen Teil eindeutig sind und so eine erneute Identifikation des Nutzers bzw. der Nutzerin über verschiedene Geräte möglich ist. Um zu zeigen, dass dies auch unter Android möglich ist, wurde in [2] ein ähnliches Experiment durchgeführt. Dabei wurden ebenfalls möglicherweise fingerprintbare Informationsquellen in der Android API manuell ausgewählt und gesammelt. Diese waren zum Zeitpunkt der Untersuchung ohne Berechtigung durch den Nutzer bzw. die Nutzerin zugänglich.

In [3] wurde untersucht, ob Android-Applikationen Daten sammeln um Fingerabdrücke zu erstellen. Im Rahmen der durchgeführten Experimente wurden acht Tracking-Bibliotheken erkannt, welche Informationen sammeln, die verwendet werden können um einen Fingerabdruck zu erstellen. In der Arbeit wurde ebenfalls festgestellt, dass diese Tracking-Bibliotheken zum Großteil andere bzw. weitere Informationsquellen einsetzen als in früheren Arbeiten gezeigt wurde.

Weiters wurde in [4] für Android ein automatisiertes Analysetool erstellt und verwendet um eine umfangreiche Übersicht über fingerprintbare Informationsquellen zu erhalten. Dazu wurden die Informationen von Methoden, Feldern und Content Providern abgerufen. Dabei wurde gezeigt, dass es viele verschiedene Informationsquellen gibt, die sich für das Fingerprinting eignen können. Ebenso wurde jedoch festgestellt, dass in der normalen Android API keine eindeutigen Wiedererkennungsmerkmale (ohne Berechtigung) abrufbar sind. Es wurde jedoch gezeigt, dass Herstelleranpassungen hier ein Risiko darstellen und möglicherweise eindeutige Geräte-Identifikationsmerkmale bzw. benutzerspezifische Informationen zugreifbar machen können.

3. Methodik

Zuerst werden alle Daten systematisch aus der iOS-Programmierschnittstelle abgerufen. Danach werden die Daten an das Backend übertragen, welche die gesammelten Daten bereinigt und in weiterer Folge auswertet. Dieser Vorgang wird auf verschiedenen iOS-Geräten wiederholt.

3.1. Ablauf

Zum Sammeln der Informationen aus der iOS-Programmierschnittstelle wird unser Automatisierungsframework für iOS verwendet. Dieses wurde für die Sammlung von Informationen aus der Programmierschnittstelle sowie für die Ausführung auf echten iOS-Geräten optimiert. Vor allem das Neukompilieren bei Problemen mit einzelnen Methoden bzw. Klassen entfällt nun. Dadurch beschleunigt sich die Ausführung des Automatisierungsframeworks. Im Folgenden wird kurz auf den generellen Aufbau des Automatisierungsframeworks eingegangen, bevor die Details der benötigten Änderungen genauer beleuchtet werden.

Das Automatisierungsframework besteht aus zwei Komponenten - der Kontrollapplikation und der mobilen Sammelapplikation. Die Kontrollapplikation wird auf einem macOS-Rechner ausgeführt, während die mobile Sammelapplikation auf einem iOS-Gerät läuft. Der generelle Aufbau sowie die Prozessschritte sind in Abbildung 1 ersichtlich. In den folgenden beiden Unterabschnitten wird auf die Aufgaben der beiden Komponenten und den Einsatz dieser zum Sammeln möglicher fingerprintbarer Informationsquellen eingegangen. Zur Auswertung der gesammelten Daten werden diese an die Analysekomponente weitergegeben. Details zu dieser Komponente gibt es im letzten Abschnitt dieses Kapitels.

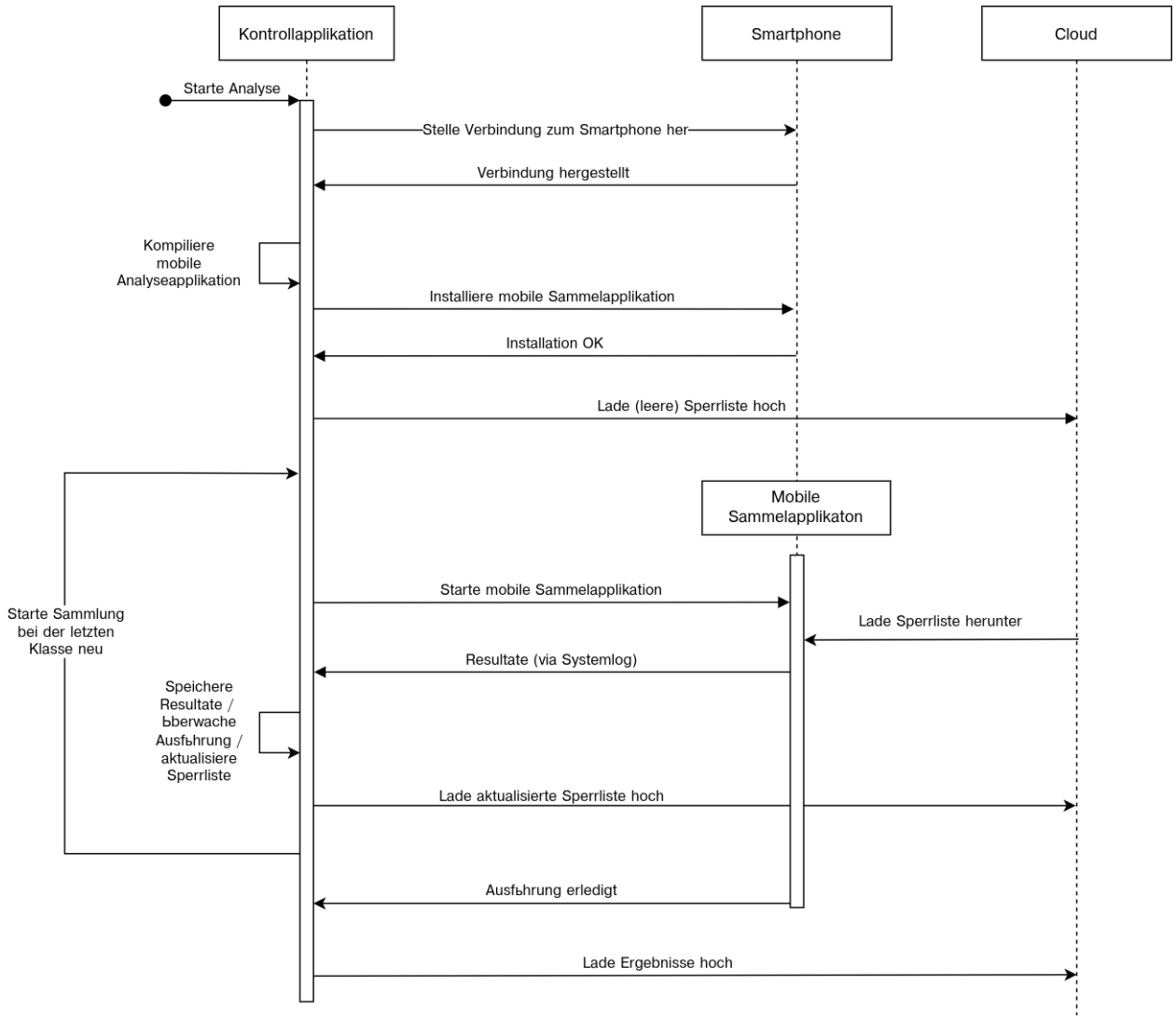

Abbildung 1 Übersicht über das Zusammenspiel der Kontrollapplikation und der mobilen Sammelapplikation beim Sammeln der Rohdaten.

3.2. Erhebung

Der Erhebungsprozess wird durch den Start der Kontrollapplikation gestartet. Nach dem Herstellen der Verbindung zum Smartphone wird dabei zuerst die aktuelle Version der mobilen Sammelapplikation installiert. Danach wird eine (anfangs noch leere) Sperrliste in die Cloud hochgeladen. In dieser werden von der Kontrollapplikation Methoden vermerkt, welche bei der Ausführung der mobilen Sammelapplikation Probleme bereiten. Im Detail handelt es sich hierbei um eine Liste mit Methoden, welche die mobile Sammelapplikation zum Absturz bringen und deswegen bei weiteren Ausführungen der Applikation ignoriert werden. Nach dem Abschluss dieser Vorbereitungsschritte wird die mobile Sammelapplikation gestartet.

Diese lädt zuerst die Sperrliste sowie den Index der aktuellen Klasse aus der Cloud herunter. Beim Start ist dieser Index auf 0, das heißt es wird mit der ersten Klasse begonnen. Die Liste der Klassen wird von der Applikation mit Hilfe von Reflection abgerufen. Anhand dieser wird nun mit dem Erstellen der Instanzobjekte begonnen. Sobald ein Objekt einer Klasse erstellt wurde werden zuerst die Properties abgerufen und danach die Methoden der Klasse aufgerufen. Die Werte der Properties sowie die Rückgabewerte der Methoden werden im Systemlog

ausgegeben und von der Kontrollapplikation aufgezeichnet. Sobald alle Properties ausgewertet und alle Methoden aufgerufen wurden wird mit der nächsten Klasse fortgefahren, bis alle Klassen ausgewertet wurden.

Beim Aufruf der Methoden bzw. beim Erstellen der Klassenobjekte kann es jedoch zu Speicherzugriffsfehlern kommen. Diese führen zu einer abrupten Beendigung der mobilen Sammelapplikation. Die Kontrollapplikation erkennt diesen Absturz und führt daraufhin einen Neustart der mobilen Sammelapplikation durch. Davor geht sie das Systemlog durch und erkennt die Methodenaufruf bzw. die Klassenerstellung, welche für den Absturz verantwortlich waren. Diese Methode bzw. Klasse wird dann in die Sperrliste aufgenommen. Die aktualisierte Sperrliste sowie der aktuelle Klassenindex werden dann in die Cloud hochgeladen. Daraufhin wird die mobile Sammelapplikation von der Kontrollapplikation neu gestartet und das Systemlog wieder auf Probleme hin überwacht. Die mobile Sammelapplikation lädt die aktualisierten Daten aus der Cloud herunter und beginnt beim hinterlegten Klassenindex erneut. Durch den neuen Eintrag in der Sperrliste wird die problembehaftete Methode nicht mehr aufgerufen bzw. die Klasse nicht mehr erneut erstellt. Sobald schlussendlich alle Klassen untersucht wurden werden die Ergebnisse in Form des Systemlogs von der Kontrollapplikation zur weiteren Auswertung in die Cloud hochgeladen.

3.2.1. Auswertung

Zur Auffindung von potentiell fingerprintbaren Informationsquellen werden Daten von verschiedenen iOS-Geräten gesammelt. Nach der Sammlung werden diese für die automatische Analyse aufbereitet und bereinigt. Daraufhin werden die Ergebnisse von den verschiedenen Geräten verglichen.

3.2.1.1. Aufbereitung

Bei der Sammlung der Daten werden alle Resultate von der mobilen Sammelapplikation in das Systemlog geschrieben. Dieses wird von der Kontrollapplikation unverändert gespeichert. In dem gespeicherten Log sind also alle Ausgaben der mobilen Sammelapplikation gespeichert, auch jene die beispielsweise durch aufgerufene Methoden getätigt werden. Um die gesammelten Daten zu analysieren und zu vergleichen müssen diese also zuerst in ein vordefiniertes Format umgewandelt werden. Dieses wurde definiert als KLASSE.METHODENNAME:::RESULTAT bzw. KLASSE.PROPERTY:::RESULTAT. Da das Resultat auch Zeilenumbrüche enthalten kann, werden diese durch die Zeichenfolge -NEWLINE- ersetzt. Dadurch kann das Resultat in weiterer Folge einfach automatisiert verglichen werden.

3.2.1.2. Bereinigung

Der Sammelvorgang wird auf jedem Gerät zweimal durchgeführt. Dazwischen wird die mobile Sammelapplikation entfernt und das Gerät neu gestartet. Nach dem Entfernen der Applikation wird sie mit einem anderen Signatur-Zertifikat neu installiert. Daraufhin wird die zweite Sammelphase gestartet. Diese aufwendige Prozedur wird durchgeführt, um die Sammlung der Daten durch Applikationen verschiedener Entwickler zu simulieren. Dadurch können jene Informationsquellen ausgeschlossen werden, die sich zwischen zwei verschiedenen Applikationen unterscheiden und sich deswegen auch nicht eignen um einen applikationsübergreifenden Fingerabdruck zu erstellen. Dadurch werden False Positives ausgeschlossen wie beispielsweise das Installationsverzeichnis der Applikation, welches einen zufälligen, für jede Applikation unterschiedlichen Zufallswert beinhaltet. Auch der über LSApplicationWorkspace.deviceIdentifierForVendor() abrufbare entwicklerspezifische Identifier wird so ausgeschlossen. Ebenso schließt diese Maßnahme alle temporären Werte aus, welche sich durch einen Applikationsneustart bzw. einen Geräteneustart ändern könnten. Alle Werte, die sich zwischen den beiden Läufen

auf demselben Gerät nicht unterscheiden werden dann als Resultat abgespeichert. Diese werden für alle untersuchten Geräte gesammelt und im nächsten Schritt, der Analyse, verglichen.

3.2.1.3. Analyse

Die gesammelten, bereinigten und geordneten Werte eines Gerätes werden bei der Analyse mit allen anderen untersuchten Geräten verglichen. Methoden, die nur auf einem Gerät aufgerufen werden konnten bzw. Felder, die nur auf einem Gerät abgerufen werden konnten werden entfernt da kein Vergleichswert vorhanden ist. Alle verbleibenden Werte, also jene die auf mindestens zwei Geräten abgerufen werden konnten werden dann automatisiert verglichen. All jene Werte, die auf allen untersuchten Geräten ident sind werden entfernt. Alle Werte, die sich zumindest auf einem Gerät unterscheiden werden behalten und ihre Quelle als möglicherweise fingerprintbar markiert. Diese werden dann manuell begutachtet und zur besseren Evaluierung in Kategorien unterteilt. Die Resultate dieser Analyse sind im nächsten Kapitel zu finden.

4. Evaluierung

In den folgenden Abschnitten wird auf das verwendete Testsetup und die erhaltenen Ergebnisse näher eingegangen.

4.1. Testaufbau

Zur Durchführung der Experimente wurde das Automatisierungsframework auf verschiedenen iOS-Geräten ausgeführt. Im Detail waren dies ein iPhone SE und ein iPhone 11. Auf den Testgeräten lief iOS in der zum Zeitpunkt der Testung aktuellen Version 16.1. Die mobile Analyseapplikation sammelte Daten von Properties und Methoden, welche beim Aufruf bzw. Abruf keine gesonderte Berechtigung durch den Benutzer bzw. die Benutzerin benötigten. Bei der Sammlung wurden alle Methoden aufgerufen die technisch aufrufbar waren. Dabei können auch ggf. private Programmierschnittstellen aufgerufen worden sein, die von iOS-Apps im App Store in der Regel nicht aufgerufen werden dürfen. Dies ist bei der Interpretation der Ergebnisse zu beachten.

4.2. Ergebnisse

Eine Übersicht über die Ergebnisse ist in Tabelle 1 ersichtlich. Die Ergebnisse wurden in instabile (also sich mittelfristig ändernde) und stabil erscheinende fingerprintbare Informationsquellen unterteilt. Zu den instabilen Informationsquellen zählen die Netzwerkdetails und der Bootzeitpunkt. Zu den Netzwerkdetails zählen beispielsweise die IP des Gateways, IP Adresse und Domainnamen. Diese Informationen wurden als instabil markiert da sich diese durch die Verbindung zu einem anderen WLAN ändern werden. Dennoch können sie beispielsweise Informationen über das Heimnetz oder oft benutzte Netzwerke verraten. Weiters wird auch der Bootzeitpunkt zu den instabilen Quellen gezählt, da dieser sich mit jedem Geräteneustart ändert.

Tabelle 1 Übersicht über die gefundenen Informationsquellen auf den untersuchten Geräten.

Type	# Informationsquellen
Instabil	
Netzwerkdetails	5
Bootzeitpunkt	2
Stabil	
Anzeigegröße von Elementen	1
Sprache	5
Systemgebietsschema	5
Einstellungen zur Barrierefreiheit	8
Eingabeapplikationen	3
Eingabemethode	7
Zeitzone	2
UI Farben / Farbraum	3
Toolbargröße	2
Verschiedene UI Einstellungen	8
Verschiedene Einstellungen	4
Laustärkeeinstellungen	1
Schriftart/-größe	2
Benutzerdefinierter Gerätename	2
Entfernte Systemapplikationen	2
Mögliche Geräteidentifikationsmerkmale	8
Modellspezifische Unterschiede	75

Für die Erzeugung eines Fingerabdrucks interessanter sind die stabilen Informationsquellen. Dazu zählt die durch den Benutzer gewählte Anzeigegröße von Elementen sowie die Größe der Toolbar. Darüber hinaus sind Informationen zum Farbraum und verschiedene weitere Einstellungen zur Benutzeroberfläche abrufbar. Dies inkludiert beispielsweise den Style der Statusbar oder die Verwendung von Weichzeichnung. Ebenso über verschiedene Methoden abrufbar ist die eingestellte Sprache sowie das Systemgebietsschema. Die Programmierschnittstelle erlaubt auch das Abrufen der Zeitzone. Weiters sind viele Einstellungen zur

Barrierefreiheit für Applikationen abrufbar. Darunter fällt beispielsweise die Einstellung Bewegungen zu minimieren, bestimmte Exklusionsbereiche in der Benutzeroberfläche, das Präferieren von skalierten Inhalten, die Deaktivierung von Slidern oder die Verwendung der Diktierfunktion des Geräts. Darüber hinaus ist auch die Eingabemethode sowie die Eingabeapplikation inklusive deren Paketnamen abrufbar. Je nach verwendeter Eingabemethode unterscheiden sich auch die unterstützten Eingabesprachen. Die verwendete Schriftart sowie die Schriftgröße sind ebenfalls abrufbar. Ebenso konnten Informationen über entfernte in iOS mitgelieferte Systemapplikationen in den Ergebnissen gefunden werden. Darüber hinaus konnte der durch den Benutzer bzw. die Benutzerin definierte Gerätename über zwei verschiedene Methoden eruiert werden. Schlussendlich gab es auch noch einige Methoden die Informationen über die verwendeten verschiedenen Gerätemodelle ausgaben. Dies inkludiert beispielsweise unterstützte Funktionen, Bildschirm- sowie Speichergröße, Hardware Modellnamen, User Agent, die unterstützten biometrischen Anmeldemethoden und die verwendete Prozessorfamilie.

5. Fazit

In diesem Bericht wurde ein automatisiertes Framework verwendet um möglicherweise fingerprintbare Informationsquellen unter iOS zu erkennen. Dies ermöglicht es ein umfangreicheres Bild über mögliche fingerprintbare Informationsquellen zu erhalten und vereinfacht die Suche nach diesen. Im Vergleich zu Android, wo ein Abruf aller Systemeinstellungen über zwei sogenannte Content Provider möglich ist, wurde unter iOS keine solche direkte Abrufmöglichkeit gefunden. Einige Einstellungen sind jedoch über verschiedene Methoden der Programmierschnittstelle abrufbar. Dabei handelt es sich jedoch im Großen und Ganzen um Einstellungen, die für eine Vielzahl an Applikationen wichtig sind, wie beispielsweise Spracheinstellungen, Größe der Benutzeroberfläche und verschiedene Anpassungen zur Barrierefreiheit. Um die Vergleichbarkeit zwischen Android und iOS zu verbessern wird es jedoch noch nötig sein, die Anzahl der nicht aufrufbaren Methoden unter iOS zu minimieren, beispielsweise durch das Vordefinieren von Methodenparametern.

Referenzen

[1] A. Kurtz, H. Gascon, T. Becker, K. Rieck und F. C. Freiling, „Fingerprinting Mobile Devices Using Personalized Configurations," in *PoPETs*, 2016.

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

[3] C. F. Torres und H. Jonker, „Investigating Fingerprinters and Fingerprinting-Alike Behaviour of Android Applications," 2018.

[4] G. Palfinger und B. Prünster, „Systematic Analysis of the Fingerprintability of Android," 2020.