Zentrum für sichere Informationstechnologie – Austria
Secure Information Technology Center – Austria

A-SIT

Seidlgasse 22 / 9, 1030 Wien
Tel.: (+43 1) 503 19 63–0
Fax: (+43 1) 503 19 63–66

Inffeldgasse 16a, 8010 Graz
Tel.: (+43 316) 873-5514
Fax: (+43 316) 873-5520

http://www.a-sit.at
E-Mail: office@a-sit.at

DVR: 1035461
ZVR: 948166612
UID: ATU60778947

MODELLIERUNG FINGERPRINTBARER INFORMATIONSQUELLEN

Version 1.0 vom 13.11.2020
Gerald Palfinger – gerald.palfinger@iaik.tugraz.at

Zusammenfassung: Der Bericht stellt ein Modell vor, welches es ermöglicht, fingerprintbare Informationsquellen einzuordnen. Als Grundlage wurden hierfür das Berechtigungssystem des Betriebssystems Android verwendet. Die Informationsquellen werden auf Basis der zur Abrufung benötigten Berechtigungen in das Modell eingeordnet. Das Modell spiegelt also die Anzahl der Methoden, Content Provider und Felder wieder, welche durch Erteilen der jeweiligen Berechtigungen für Applikationen zugänglich sind. Dadurch kann eingeschätzt werden, welche Berechtigungen zur Fingerprintbarkeit eines Geräts beitragen.

Inhaltsverzeichnis

1. Einführung

Viele Smartphoneapplikationen nutzen personalisierte Werbung als eine der Haupteinnahmequellen. Diese Werbung wird meistens von Werbenetzwerken bereitgestellt, welche in der Regel als Bibliothek in der jeweiligen Applikation integriert werden. Dadurch haben die eingebundenen Applikationen vollen Zugriff auf die Android API sowie alle Berechtigungen, welche der Applikation gewährt wurden. Dabei übernehmen diese Bibliotheken auch die Personalisierung der angezeigten Werbung. Um Werbung zu personalisieren, müssen die Nutzerinnen und Nutzer jedoch über verschiedene Applikationen hinaus wiedererkannt werden. Da der Zugriff auf viele eindeutige Identifikationsmerkmale wie die IMEI mit neueren Versionen von Android entfernt wurden, setzen Applikationen Fingerprinting ein, um diese Beschränkung zu umgehen. Durch Zusammenführen verschiedener Informationsquellen lassen sich so Geräte eindeutig wiedererkennen. Dieses Fingerprinting funktioniert vom Prinzip her ähnlich wie Browser-Fingerprinting. Dabei werden

möglichst viele Informationsquellen abgerufen, welche ein Gerät und/oder eine Nutzerin bzw. einen Nutzer identifizieren können. Durch Kombination der abgerufenen Elemente kann dann ein eindeutiger Fingerabdruck (Fingerprint) erstellt werden. So kann eine Wiedererkennung auch über verschiedene Applikationen hinweg stattfinden. Da Smartphones eine große Menge an personenbezogenen Daten speichern, ist die Anzahl an potentiell fingerprintbaren Informationen durchaus höher als im Browser. Während es bei Browsern mit Drittanbieter-Cookies eine Möglichkeit gibt, um Informationen wie selbstgenerierte Identifikationsnummern über Webseiten hinweg zu speichern, ist eine solche Funktion in der Android API nicht vorgesehen.

Dieser Bericht stellt ein Framework vor, um fingerprintbare Informationsquellen zu erkennen und diese in einem Modell einzuordnen. Dabei wird als Basis das Berechtigungssystem von Android verwendet. Zur Sammlung der potentiell fingerprintbaren Daten wird eine Smartphoneapplikation ausgeführt. Zur Einteilung der potentiell fingerprintbaren Informationen werden diese Daten mit verschiedenen erteilten Berechtigungen gesammelt. Der Prozess wird von einem PC gesteuert, da von diesem die Berechtigungen automatisch vergeben werden können. Im Anschluss wird ausgewertet, welchen Einfluss die verschiedenen unter Android verfügbaren Berechtigungsgruppen auf die Fingerprintbarkeit haben. So kann mit Hilfe des erstellten Modells abgeschätzt werden, welche zusätzlichen Informationsquellen durch Erteilen der verschiedenen Berechtigungen zugänglich sind.

2. Hintergrund

Die folgenden beiden Abschnitte behandeln das Applikationsberechtigungssystem, auf welchem das Modell aufbaut und verwandte Arbeiten im Bereich Smartphone-Fingerprinting.

2.1. Applikationsberechtigungen unter Android

Um Nutzerdaten vor unberechtigtem Zugriff zu schützen, verwendet Android verschiedene technische Maßnahmen. Im Zentrum dieser Maßnahmen steht das Berechtigungssystem des Betriebssystems. Dieses System verhindert, dass Applikationen direkt auf vertrauliche Daten wie Kontakte, Nachrichten oder Fotos zugreifen. Weiters beschränkt es auch den Zugriff auf Systemfunktionen, welche Rückschlüsse auf private Informationen erlauben, wie die Standortfunktionen oder Kamerafunktionalität. Um auf diese Informationen und Funktionen zuzugreifen, müssen Applikationen zuerst die dazugehörige(n) Berechtigungen anfordern. Je nach Art der Berechtigung wird diese entweder automatisch vom System vergeben oder benötigt zusätzliche Einwilligung der Benutzerin bzw. des Benutzers.

Android besitzt im Auslieferungszustand mehr als 150 Berechtigungen. Zur besseren Übersicht sind diese Berechtigungen in Berechtigungsgruppen eingeteilt. So kann eine Berechtigungsgruppe mehrere verwandte Berechtigungen umfassen, während eine Berechtigung den Zugriff auf eine oder mehrere verwandte API-Aufrufe regelt. So wird versucht, die große Anzahl an Berechtigungen übersichtlich darzustellen und es dem Nutzer bzw. der Nutzerin zu ermöglichen, eine informierte Entscheidung zu treffen ohne umfassende technische Details bzw. eine zu große Anzahl an Abfragen zu benötigen. Dies heißt im Umkehrschluss jedoch auch, dass eine Applikation, welcher eine Berechtigungsgruppe gewährt wurde, weitere Berechtigungen dieser Gruppe ohne Nutzerinteraktion anfordern kann.

2.2. Verwandte Arbeiten

Um das Fingerprinting von Nutzerinnen und Nutzern zu verhindern, hat Google mit Android 10 den Zugriff auf eindeutige Erkennungsmerkmale erschwert oder für Drittprogramme gänzlich entfernt [1]. Dennoch kann durch Kombination nutzer- sowie gerätespezifischer Informationen ein eindeutiger Fingerprint erstellt werden. So wird in [2] gezeigt, dass durch Kombinieren von Informationsquellen ein eindeutiger Fingerprint eines Android-Gerätes erstellt werden kann. Dazu wurden manuell nach potentiell fingerprintbaren Informationsquellen gesucht. Als Grundlage wurden hierbei vorrangig

Informationen verwendet, welche über das Einstellungsmenü verändert werden können sowie durch die API oder ähnliche Methoden abrufbar sind.

In [3] werden Applikationen auf mögliche Fingerprintaktivitäten untersucht. Dabei wurde festgestellt, dass vor allem die in vielen Applikationen eingebundenen Werbe- bzw. Trackingbibliotheken Informationen abfragen, welche die Erstellung eines Fingerprints ermöglichen. In [4] wurde systematisch untersucht, welche frei verfügbaren Informationsquellen verwendet werden können, um einen Fingerprint zu erstellen. Dazu wurde ein Framework erstellt, welches die Android API nach fingerprintbaren Informationsquellen durchsucht. Dabei werden Methoden aufgerufen, Felder ausgelesen und Content Provider gedumpt. In verschiedenen Experimenten wurde daraufhin verglichen, welche Werte sich auf unterschiedlichen Geräten unterscheiden. So wurden Rückschlusse auf die Fingerprintbarkeit verschiedener API-Aufrufe gezogen.

3. Methodik

Die erhobenen Daten zur Fingerprintbarkeit wurden mit einem dafür erstellten Framework gesammelt. Die Sammlung der möglicherweise fingerprintbaren Daten läuft dabei vollautomatisiert ab. Dazu wurde ein Framework erstellt, welches aus drei Komponenten besteht. Eine Übersicht über die Komponenten und deren Zusammenspiel ist in Abbildung 1 zu finden. Der Gerätekoordinator und die Analysekomponente werden auf einem PC ausgeführt, während die Smartphone-komponente auf einem an den PC per Android Debug Bridge (adb) angeschlossenem Smartphone ausgeführt wird.

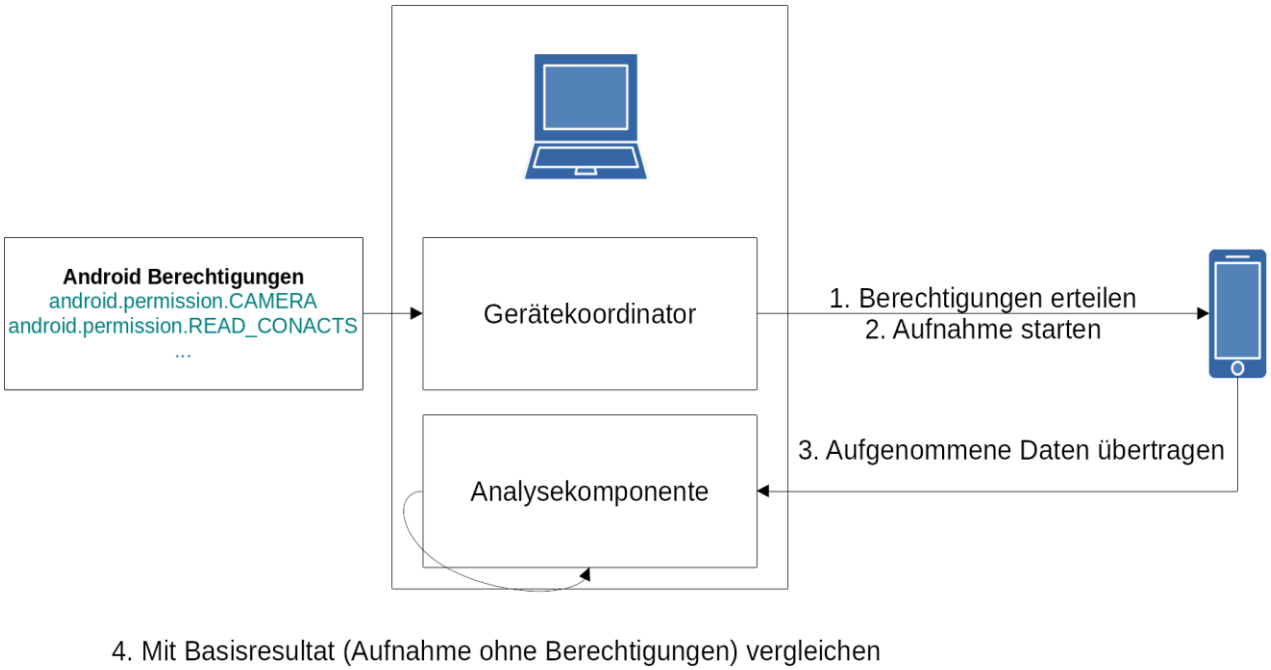

Abbildung 1 Übersicht über den Versuchsaufbau.

3.1. Gerätekoordinator

Der Gerätekoordinator übernimmt die Steuerung der Aufnahme. Dazu werden vor dem Start der Aufnahme alle Berechtigungen einer Berechtigungsgruppe erteilt. Da Berechtigungen am Gerät nur mit Interaktion des Benutzers bzw. der Benutzerin vergeben werden können, wird dieser Schritt vom PC per adb übernommen. Via adb können Berechtigungen auf angeschlossenen Geräten mit aktiviertem Entwicklermodus erteilt werden. Dazu wird der Befehl `adb shell pm grant <package_name> android.permission.<permission_name>` verwendet. Um eine Berechtigung zu erteilen, wird also der Name der jeweiligen Berechtigung benötigt. Deshalb wurden alle unter Android 11 verfügbaren Berechtigungen aus dem Quellcode von Android 11 gesammelt

und anhand der im Einstellungsmenü sichtbaren Berechtigungen gruppiert. Danach startet der Gerätekoordinator die Smartphonekomponente und damit die Aufnahme. Nach Abschluss der Aufnahme werden die gesammelten Daten für die Bereinigung und spätere Analyse auf den PC übertragen. Auf dem Smartphone wird der Ausgangszustand wiederhergestellt, indem die erteilten Berechtigungen widerrufen werden. Die gesammelten Daten werden durch die Analysekomponente mit dem Basisresultat, also einer Aufnahme ohne erteilte Berechtigungen, verglichen. Danach startet der Prozess mit der Erteilung der nächsten Berechtigungsgruppe erneut bis die Resultate aller Berechtigungsgruppen aufgenommen wurden.

3.2. Smartphonekomponente

Die Sammlung der potentiell fingerprintbaren Daten wird auf einem Smartphone ausgeführt. Die Smartphonekomponente basiert im Wesentlichen auf einer aktualisierten Version der in [5] vorgestellten Applikation. Für das Verständnis wird im Folgenden die Funktionsweise der Smartphonekomponente zusammengefasst. Ausgehend von einer Liste mit Klassen, Methoden, Konstruktoren sowie deren Parametern ruft die Smartphonekomponente alle Methoden der Android API auf und wertet die Felder von Klassen aus. Dazu werden mittels Java Reflection alle Klassen geholt und mit den vorab eruierten Konstruktoren Objekte der jeweiligen Klasse erstellt. Wird für den Aufruf eines Konstruktors ein (oder mehrere) Parameter benötigt, so werden hierbei für diesen Parameter entweder vorab aus der API-Dokumentation geparste Konstanten oder manuell vordefinierte Werte verwendet. Ist keiner der beiden Typen vorhanden, so werden für den Datentyp übliche vordefinierte Werte verwendet. Dazu wird zuerst der Datentyp des Parameters von der Smartphonekomponente eruiert. Anhand des Parameterindizes werden die Parameter des Konstruktors oder der Methode den vorab geparsten Parameternamen zugeordnet. Dies ist notwendig, da die vorab definierten Parameter und Konstanten mit Hilfe des Parameternamen dem jeweiligen Parameter zugeordnet werden.

Der Aufruf der Methoden wird auf jene Methoden beschränkt, welche in der Regel Informationen zurückgeben. Dadurch sollen Seiteneffekte, die durch den Aufruf von Methoden ausgelöst werden können, minimiert werden. Die Auswahl der Methoden erfolgt anhand des Präfixes des Methodennamens – jene mit den Präfixen `get`, `has`, `is`, `query` und `support` werden aufgerufen. Der Rückgabewert der Methode wird nach dem Aufruf für die spätere Analyse gespeichert. Zusätzlich zu primitiven Werten können Methoden der Android API jedoch auch komplexere Objekte zurückgeben. Da diese nicht immer als Zeichenfolge repräsentiert werden können, werden diese Objekte zusätzlich auf weitere aufrufbare Methoden untersucht. Dazu werden die Objekte den vordefinierten Klasseninformationen zugeordnet und die darin enthaltenen Methoden anhand des oben beschriebenen Schemas aufgerufen. Da der Smartphonekomponente nur eine begrenzte Menge an Hauptspeicher zur Verfügung steht, wurde die Aufruftiefe auf maximal drei Ebenen begrenzt.

Zusätzlich zum Aufruf der Methoden einer Klasse werden auch die Felder abgerufen. Dazu werden die für den Methoden- bzw. Konstruktorenaufruf erstellten Objekte verwendet. Die abgerufenen Werte der Felder werden analog zu den Rückgabewerten der Methoden für die spätere Analyse gespeichert.

Als dritte und letzte Informationsquelle wertet die Smartphonekomponente die vom Betriebssystem zur Verfügung gestellten Content Provider aus. Dabei handelt es sich um ein Konzept von Android, welches den Zugriff auf verschiedene Datenspeicher vereinheitlicht. Diese Schnittstelle behandelt auch etwaige Berechtigungen, welche für den lesenden und/oder schreibenden Zugriff auf den Datenspeicher benötigt werden. Der Zugriff erfolgt über sogenannte Content URIs (`content://`). Diese werden von der Smartphonekomponente aus den vorab ausgewerteten Feldern und Rückgabewerten mit Hilfe des Präfixes extrahiert. Wird der Zugriff auf einen Content Provider vom System gewährt, so iteriert die Smartphonekomponente alle Zeilen und Spalten des Content Providers. Die dabei gesammelten Daten werden für die spätere Analyse gespeichert.

Die Rückgabewerte der Methoden, Felder und Content Provider werden zweimal ausgewertet. Zwischen den beiden Durchläufen wird die Smartphonekomponente deinstalliert. Beim zweiten Durchgang wird die Smartphonekomponente mit einem anderen Signatur-Zertifikat signiert. Dadurch wird simuliert, dass es sich bei den beiden Durchläufen um Sammlungen mit verschiedenen Applikationen unterschiedlicher Entwickler handelt. Dadurch kann simuliert werden, dass es sich bei der Smartphonekomponente um eine Tracking-Bibliothek handelt, welche in zwei unabhängige Applikationen integriert ist. So kann ausgeschlossen werden, dass Informationsquellen als fingerprintbar markiert werden, welche sich zwischen Neuinstallationen oder unterschiedlichen Applikationsentwicklern unterscheiden. Darüber hinaus können auch temporäre Informationsquellen entfernt werden, welche sich nach einem Neustart der Applikation ändern und so auch nicht zum Fingerprinten von Geräten geeignet sind.

3.3. Analysekomponente

Die Analysekomponente wertet die gesammelten Daten von verschiedenen Geräten aus. Dazu werden die Daten bereinigt und verglichen.

Bei der Bereinigung werden etwaige Duplikate in den Ergebnissen zusammengefasst. Dazu kann es kommen, wenn mehrere Werte für einen Parameter definiert werden oder mehrere Methoden Objekte derselben Klasse zurückgeben. Sollte der Rückgabewert trotz unterschiedlicher Parameter oder Objekte gleichbleiben, so werden die Duplikate entfernt. Unterscheidet sich der aufgezeichnete Wert, so werden beide Instanzen behalten.

Nach der Bereinigung der gesammelten Daten werden die verbleibenden Ergebnisse analysiert. Dabei wird vor allem Augenmerk auf jene Methoden, Felder und Content Provider gelegt, welche auf mehreren Smartphones aufgerufen bzw. ausgewertet werden können und deren Wert sich unterscheidet. Diese Informationsquellen werden als potentiell fingerprintbar markiert.

4. Auswertung

In den folgenden Abschnitten werden der Experimentaufbau und die Ergebnisse inklusive des erstellten Modells näher betrachtet.

4.1. Experimentaufbau

Als Testgeräte wurde ein Google Pixel 4 sowie ein Samsung Galaxy S9 verwendet. Beide verwendeten die aktuellsten Softwareversionen, welche zur Zeit der Analyse verfügbar waren. Dadurch liefen auf dem Google Pixel 4 zum Testzeitpunkt Android mit Sicherheitspatches vom 5. Oktober 2020, während auf dem Samsung Galaxy S9 der Sicherheitspatchlevel 1. September 2020 installiert war.

4.2. Ergebnisse

Zur Einteilung der fingerprintbaren Informationsquellen wird das Berechtigungssystem von Android verwendet. Als Grundlage dienen hierbei die zustimmungspflichtigen Berechtigungen. Diese schützen besonders heikle Informationsquellen. Ähnlich gelagerte Berechtigungen sind in Gruppen eingeordnet. Da eine Zustimmung zur Verwendung einer Berechtigung die ganze Berechtigungsgruppe umfasst, orientiert sich das Modell auch an den Berechtigungsgruppen. Die Zuordnung einzelner Berechtigungen zu den jeweiligen Gruppen ist in Abbildung 2 ersichtlich. Bei der Berechtigungsgruppe der Sensoren handelt es sich um einen Sonderfall. Da diese Nutzerinteraktion erfordern werden diese vom Framework nicht unterstützt. Deshalb werden Sensoren im Zuge dieser Studie nicht ausgewertet. Zur Fingerprintbarkeit von Sensoren wird deshalb auf [6] verwiesen.

KALENDAR

- android.permission.READ_CALENDAR
- android.permission.WRITE_CALENDAR

ANRUFHISTORIE

- android.permission.READ_CALL_LOG
- android.permission.WRITE_CALL_LOG
- android.permission.PROCESS_OUTGOING_CALLS

KAMERA

- android.permission.CAMERA

KONTAKTE

- android.permission.READ_CONTACTS
- android.permission.WRITE_CONTACTS
- android.permission.GET_ACCOUNTS

STANDORT

- android.permission.ACCESS_COARSE_LOCATION
- android.permission.ACCESS_FINE_LOCATION
- android.permission.ACCESS_BACKGROUND_LOCATION

MIKROFON

- android.permission.RECORD_AUDIO

TELEFON

- android.permission.CALL_PHONE
- android.permission.READ_PHONE_NUMBERS
- android.permission.ACCEPPT_HANDOVER
- android.permission.ADD_VOICEMAIL
- android.permission.USE_SIP
- android.permission.ANSWER_PHONE_CALLS
- android.permission.READ_PHONE_STATE

SENSOREN

- android.permission.BODY_SENSORS

SMS

- android.permission.READ_SMS
- android.permission.SEND_SMS
- android.permission.RECEIVE_SMS
- android.permission.RECEIVE_MMS
- android.permission.READ_CELL_BROADCASTS
- android.permission.RECEIVE_WAP_PUSH

SPEICHER

- android.permission.READ_EXTERNAL_STORAGE
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.ACCESS_MEDIA_LOCATION

Abbildung 2 Übersicht der zustimmungspflichtigen Berechtigungen sowie deren Zuordnung zu Berechtigungsgruppen.

Das erstellte Modell ist in Abbildung 3 ersichtlich. Das Modell zeigt, dass durch jede Berechtigungsgruppe eine unterschiedliche Anzahl an weiteren fingerprintbaren Informationsquellen dazukommen. Das Modell enthält die Anzahl der hinzugekommenen Informationsquellen, gruppiert nach dem Typ der Quelle. Ein Großteil der Informationen ist entweder über Content Provider oder Methoden abrufbar, nur wenige Informationen über Felder.

Die SMS-Berechtigung ermöglicht Zugriff auf SMS- sowie MMS-Nachrichten über verschiedene Content Provider. Dabei kann nicht nur der Inhalt dieser Nachrichten abgerufen werden, sondern auch detaillierte Metadaten. Diese beinhalten unter anderem die Telefonnummer bzw. den Namen des Absenders, Zeitpunkt des Erhalts sowie des Sendens einer Nachricht, Lesestatus, sowie eine eindeutige ID. Zusätzlich ermöglicht diese Berechtigungsgruppe den Aufruf von Methoden, welche

die Telefonnummern der eingelegten SIM-Karten zurückgeben. Insgesamt sind 23 zusätzliche Content Provider sowie 5 Methoden durch Erteilen dieser Gruppe zugänglich.

Die Storage-Berechtigungsgruppe ermöglicht den Zugriff auf Dateien des internen Speichers. Die Dateien werden von Android nach Dateityp indiziert. Der Zugriff auf diese Datenbanken ist über 13 verschiedene Content Provider möglich. Die Content Provider erlauben direkten Zugriff auf Audiodateien, Videos, Bilder und andere Medien. Dabei kann auch direkt auf Metadaten, die in den Dateien gespeichert sind, zugegriffen werden. Dies umfasst beispielsweise EXIF-Dateien von Bildern, welche auch Standortinformationen enthalten können, als auch ID3-Tags von Audiodateien, welche detaillierte Informationen über die gespeicherten Musikdateien enthalten. Die Storage-Berechtigung wird auch benötigt, um auf dem internen Speicher gespeicherte Bildschirm-hintergrundbilder zuzugreifen. Verschiedene Methoden und Felder erlauben dabei den Zugriff auf Details wie Auflösung und Größe. Insgesamt handelt es sich hierbei um 25 Methoden und 4 Felder.

Die Telefon-Berechtigungsgruppe erlaubt es Applikationen auf die Telefonnummer der eingelegten SIM-Karten zuzugreifen. Darüber hinaus erlaubt das Erteilen dieser Gruppe Netzbetreiber-informationen auszulesen. Weiters können Notrufnummern und der Typ des Daten- sowie Telefon-netzwerks abgerufen werden. Ebenso können Hardware-Informationen wie Multi-SIM Unterstützung sowie die Softwareversion des Telefonstacks und die verwendete Voicemail-App ausgelesen werden. Das Framework hat 34 weitere Methoden und einen Content Provider als Teil dieser Gruppe gefunden.

Die Anrufhistorie-Berechtigungsgruppe ermöglicht Zugriff auf getätigte Anrufe. Insgesamt ist ein weiterer Content Provider und eine Methode aufrufbar. Dabei ermöglicht eine Methode das Abrufen des letzten Anrufes, während der zugehörige Content Provider Zugriff auf alle getätigten Anrufe ermöglicht. Dieser Content Provider ermöglicht auch den Zugriff auf weitere Details, wie Datum, Dauer oder der ungefähre Standort des Gesprächspartners.

Die Standort-Berechtigungsgruppe erlaubt es, Apps zwei weitere fingerprintbare Methoden aufzurufen. Über eine dieser Methode ist es möglich auf den aktuellen sowie letzten bekannten Standort zuzugreifen. Weiters können die installierten Standort-Provider abgerufen werden.

Wird einer App die Mikrofon-Berechtigungsgruppe zugeteilt, so kann auf verschiedene Audio-Charakteristika des Systems zugegriffen werden. So kann über verschiedene Methoden abgerufen werden, ob AudioFX aktiviert wurde, die verwendete sowie verfügbaren Abtastfrequenzen, verfügbare Buffergrößen und Kanalanzahl, sowie der gerätespezifische Produktname des Audiogerätes. Insgesamt wurden 21 zusätzliche Methoden identifiziert, jedoch keine weiteren Content Provider oder Felder.

Über die Kalender-Berechtigungsgruppe kann auf 16 zusätzliche Content Provider zugegriffen werden. Es wurden keine weiteren Methoden oder Felder gefunden. Die Content Provider erlauben es, auf allgemeine Informationen wie die Version der installierten Zeitzonendatenbank sowie eingestellte Zeitzone zuzugreifen. Weiters kann auf aktivierte Reminder zugegriffen werden. Über die Kalender-Content Provider kann direkt auf alle auf dem Gerät vorhandenen Kalender sowie die darin gespeicherten Events sowie deren Details zugegriffen werden. Ebenso kann über weitere Content Provider auf eingerichtete Kalenderfarben sowie weitere Details zugegriffen werden.

Durch die Berechtigungsgruppe der Kontakte kann auf 53 weitere Content Provider zugegriffen werden. Neben dem Zugriff auf die Kontakte sowie deren Details wie Adresse oder Geburtsdatum kann das Erstellungsdatum der Kontakte sowie auf Kontaktgruppen zugegriffen werden. Weiters können erstellte Organisationen und häufig benutzte Kontakte abgerufen werden. Auf Samsung-Geräten kann zusätzlich auf die E-Mail-Adresse, die zur Anmeldung beim Samsung-Account verwendet wird, zugegriffen werden.

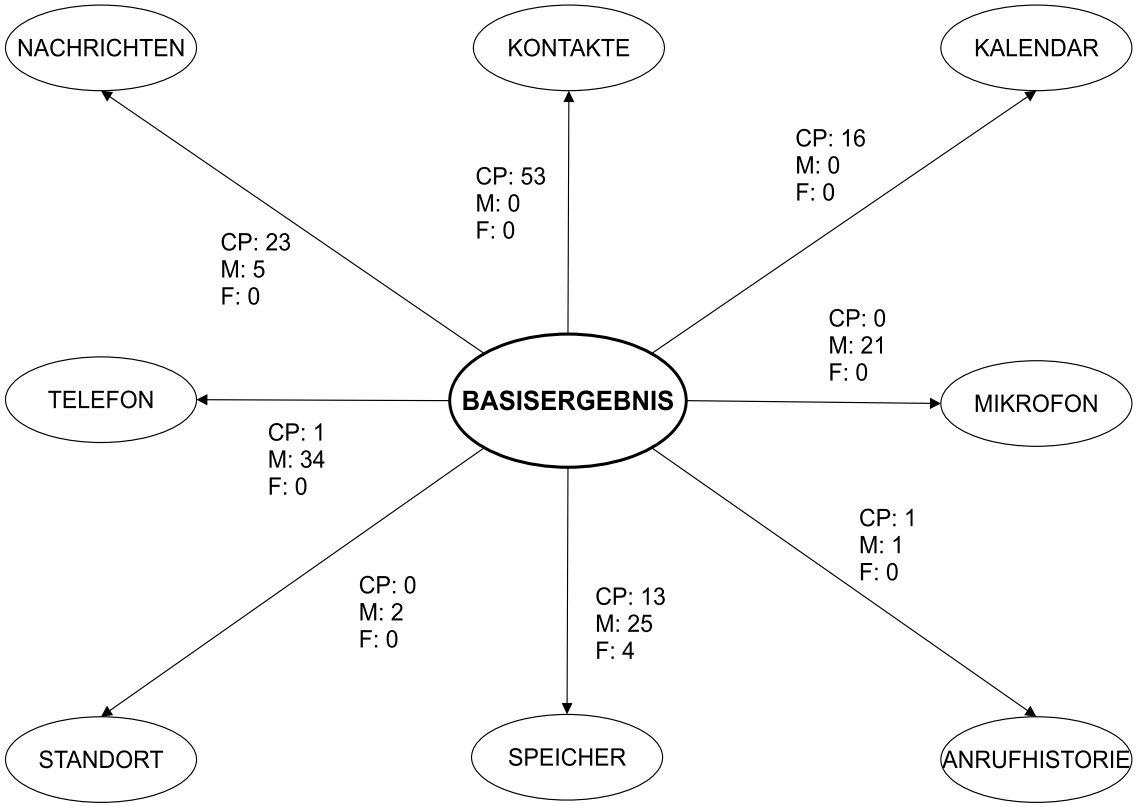

Abbildung 3 Erstelltes Modell auf Basis der Berechtigungsgruppen (CP = Content Provider, M = Methoden, F = Felder).

5. Fazit

Der Bericht stellt ein Framework vor, welches es erlaubt, ein Modell fingerprintbarer Informationsquellen auf Android zu erstellen. Dazu wurden Methoden aufgerufen, Content Provider gedumpt und Felder ausgewertet. Dieser Vorgang wurde mit unterschiedlichen Berechtigungen wiederholt. Aus den gewonnenen Rohdaten wurde daraufhin ein Modell erstellt. Dieses gibt an, wie viele zusätzliche Informationsquellen durch Erteilen einer Berechtigung verfügbar sind. So kann eingeschätzt werden, wie stark sich die Methoden, Felder sowie Content Provider einer Berechtigungsgruppe auf die Fingerprintbarkeit auswirken. Die Studie zeigt auch, dass durch Erteilen der jeweiligen Berechtigungsgruppen Informationsquellen zugänglich gemacht werden, die durchaus anhand des Namens der Gruppe antizipiert werden können.

Referenzen

[1] Google, „Privacy changes in Android 10,“ 2019. [Online]. Available: https://developer.android.com/about/versions/10/privacy/changes. [Zugriff am 18 02 2020].

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

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

[4] G. Palfinger und B. Prünster, „AndroPRINT: analysing the fingerprintability of the Android API,“ in *ARES '20: Proceedings of the 15th International Conference on Availability, Reliability and Security*, Virtual, Ireland, 2020.

[5] G. Palfinger, B. Prünster und D. J. Ziegler, „AndroTIME: Identifying Timing Side Channels in the Android API,“ in *TrustCom*, Guangzhou, China, 2020.

[6] H. Bojinov, Y. Michalevsky, G. Nakibly und D. Boneh, „Mobile Device Identification via Sensor Fingerprinting,“ 2014.

[7] BeanShell, „BeanShell scripting language,“ [Online]. Available: https://github.com/beanshell/beanshell. [Zugriff am 19 02 2020].

[8] Google, „Permissions Overview - Normal permissions - Android Developers,“ [Online]. Available: https://developer.android.com/training/permissions/requesting#normal-dangerous. [Zugriff am 30 03 2020].

[9] G. P. S. M. R. Spreitzer, „Scandroid: Automated side-channel analysis of android APIs,“ in *WiSec*, Stockholm, 2018.

[10] E. P., „How unique is your web browser?,“ in *PET*, 2010.