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

SYSTEMATISCHE ANALYSE VON ANDROID AUF FINGERPRINTBARKEIT

Version 1.1 vom 05.10.2020
Gerald Palfinger – gerald.palfinger@iaik.tugraz.at

Zusammenfassung: Der Bericht stellt ein Framework vor, welches fingerprintbare Informationsquellen auf Android-Geräten automatisch erkennen kann. Dazu ruft es automatisiert Methoden auf, fragt Felder ab und ruft Daten von Content Providern ab. Das Framework wurde in zwei verschiedenen Experimenten verwendet. So konnte eine Vielzahl an Informationsquellen erkannt werden, welche solch potentiell fingerprintbare Informationen liefern. Darüber hinaus wurden eindeutige Geräte-Identifikationsmerkmale gefunden, welche ein Resultat von Herstelleranpassungen sind.

Inhaltsverzeichnis

1. Einführung

Mit Android 10 hat Google den Zugang zu vielen fingerprintbaren Informationsquellen entfernt [1]. Dazu gehört der Zugriff auf Kameradaten, Details über gespeicherte WiFi-Netzwerke und der Zugriff auf eindeutige Identifikatoren wie die IMEI oder die Seriennummer des Geräts - einige davon konnten ohne explizite Einwilligung des Nutzers abgerufen werden. Dieser Bericht stellt ein Framework vor, welches Informationsquellen erkennen kann, die zur Erstellung von Fingerabdrücken einzelner Android-Geräte (und damit Nutzerinnen und Nutzer) verwendet werden können. Fingerprinting ist bereits bekannt und wird aktiv genutzt um Benutzerinnen und Benutzer im Web durch Browser-Fingerprinting zu verfolgen [2]. Im Wesentlichen können Eigenschaften wie

Auflösung oder installierte Schriftarten von Websites abgefragt werden. Wenn ein Skript (z.B. ein Werbebanner oder die Integration in soziale Netzwerke) auf vielen Webseiten vorhanden ist, können einzelne Benutzerinnen und Benutzer durch diesen Fingerabdruck über mehrere Webseiten hinweg verfolgt werden. Das Framework wendet diesen Ansatz auf die Android-Plattform an und ist in der Lage, automatisch Funktionen zu entdecken, die zur Erstellung gerätespezifischer Fingerabdrücke verwendet werden können. Einmal identifiziert, ist ein applikationsübergreifendes User-Tracking für jeden Anbieter von beliebten Bibliotheken, wie z.B. großer Werbenetzwerke, die in eine Vielzahl von Android-Apps eingebunden sind, leicht möglich, da die abgefragten Fingerabdrücke applikations-unabhängig sind. Im Gegensatz zur Abfrage des Adressbuches einer Benutzerin oder eines Benutzers erfordert der Aufruf der durch das Framework identifizierten API-Methoden keine Zustimmung der Benutzerin oder des Benutzers und geschieht somit unerkannt im Hintergrund.

Bei der Bewertung der Wirksamkeit unseres Ansatzes wurden Fingerabdruck-Merkmale entdeckt, die vollkommen stabil (d.h. ein Neustart des Gerätes hat keine Auswirkungen) und höchst eindeutig sind und somit die Identifizierung einzelner Benutzerinnen oder Benutzer mit hoher Wahrscheinlichkeit ermöglichen. So kann beispielsweise die vollständige Liste der Klingeltöne abgefragt werden. Da dies auch alle von der Benutzerin oder dem Benutzer hinzugefügten Klingeltöne einschließlich des genauen Zeitstempels des Ladens beliebiger benutzerdefinierter Klingeltöne auf das Gerät umfasst, kann die Identifizierung von Personen, die benutzerdefinierte Klingeltöne verwenden, mit hoher Sicherheit erfolgen. Darüber hinaus haben wir einen Fehler auf Samsung-Geräten aufgedeckt, der es jeder Anwendung ermöglicht, die E-Mail-Adresse, die mit dem auf dem Gerät verwendeten Samsung-Konto des Benutzers verknüpft ist, sowie eine eindeutige Hardware-ID abzurufen. Dies an sich hat bereits schwerwiegende Folgen für die Privatsphäre der Benutzerinnen und Benutzer, da diese nicht nur nicht zustimmen, sondern dies auch nicht verhindern können.

Die Ergebnisse wurden in zusätzlich in einer wissenschaftlichen Publikation veröffentlicht [3].

2. Hintergrund

Fingerprinting von Android-Geräten wurde im Wesentlichen in den folgenden zwei Werken diskutiert. Diese werden im Folgenden näher beleuchtet. In [3] werten Wu et al. manuell ausgewählte Informationen der Android API, welche ohne Berechtigung zugänglich sind, auf ihre Fingerprint-barkeit aus. Dazu wurde als Ausgangsbasis das Einstellungsmenü der Geräte verwendet. Anhand der dort vorhandenen Einstellungsmöglichkeiten wurde unter Zuhilfenahme der Android API Dokumentation nach Methoden gesucht, um die gewählten Einstellungen abzurufen. Diese wurden anhand drei verschiedener Algorithmen und einem Datensatz an gesammelten Fingerabdrücken ausgewertet. In [4] wurden Android-Applikationen auf ihr Fingerprinting-Verhalten untersucht. Dabei wurden acht verschiedene Tracking-Bibliotheken ausgemacht, welche Fingerprinting verwenden.

3. Methodik

Um Daten zu sammeln, wurde eine Android-App erstellt, welche die Android API systematisch nach möglicherweise fingerprintbaren Informationen absucht. In einem weiteren Schritt werden die Daten an das Backend übertragen, welches die gesammelten Daten auswertet.

3.1. Vorbereitungsphase

Um die Android API systematisch nach fingerprintbaren Informationen abzusuchen, werden zuerst einige Informationen benötigt. Diese werden vorab aus der Android API Dokumentation geparst und als Konfigurationsdatei aufbereitet. Diese Konfigurationsdateien werden mit der Android App mitgeliefert und von dieser verwendet, um Objekte zu erstellen, Methoden aufzurufen und Felder auszuwerten. Aus der API-Dokumentation werden alle dokumentierten Klassen, deren Methoden inklusive Parameternamen, und alle Konstruktoren sowie deren Parameternamen geparst. Das Parsen der Android-API-Dokumentation und die Smartphone-Anwendung, die die Daten sammelt, basieren in Teilen auf dem SCAnDroid-Framework [5]. Dieses wurde mit Unterstützung für

Feldauswertung, Content Provider und verbesserter API-Abdeckung durch die Berücksichtigung von Konstanten und Objekten abgeleiteter Typen erweitert.
Zusätzlich zum Aufbau der Klassen werden auch mögliche Konstanten aus der API-Dokumentation geparst. In der Regel werden diese in der Parametertabelle definiert, siehe Abbildung 1. Da diese kein einheitliches Format hat, werden sowohl alle Hyperlinks als auch Code-Blöcke auf potentielle Konstanten durchsucht. Zusätzlich wird der verbleibende Text nach möglichen Konstanten untersucht. Diese werden in einer Textdatei gespeichert, welche die Zuordnung zu Klasse, Methode und dem jeweiligen Parameternamen möglich macht. Durch diese breite Auswahl an möglichen Konstanten enthält die Liste auch Statements, bei denen es sich in Wirklichkeit nicht um Konstanten handelt oder welche aus anderen Gründen nicht interpretierbar sind. Diese werden in weiterer Folge auf dem Smartphone gefiltert. Ein vorheriges Entfernen möglicherweise nicht interpretierbarer Elemente durch Heuristiken erschien nicht zielführend, da das Interpretieren und Entfernen nicht interpretierbarer Elemente am Smartphone nur wenige Sekunden in Anspruch nehmen.

Parameters	
`archiveFilePath`	`String`: The path to the archive file This value must never be `null`.
`flags`	`int`: Additional option flags to modify the data returned. Value is either `0` or a combination of GET_ACTIVITIES, GET_CONFIGURATIONS, GET_GIDS, GET_INSTRUMENTATION, GET_INTENT_FILTERS, GET_META_DATA, GET_PERMISSIONS, GET_PROVIDERS, GET_RECEIVERS, GET_SERVICES, GET_SHARED_LIBRARY_FILES, GET_SIGNATURES, GET_SIGNING_CERTIFICATES, GET_URI_PERMISSION_PATTERNS, MATCH_UNINSTALLED_PACKAGES, MATCH_DISABLED_COMPONENTS, MATCH_DISABLED_UNTIL_USED_COMPONENTS, MATCH_SYSTEM_ONLY, android.content.pm.PackageManager.MATCH_FACTORY_ONLY, android.content.pm.PackageManager.MATCH_DEBUG_TRIAGED_MISSING, android.content.pm.PackageManager.MATCH_INSTANT, MATCH_APEX, GET_DISABLED_COMPONENTS, GET_DISABLED_UNTIL_USED_COMPONENTS, GET_UNINSTALLED_PACKAGES, and android.content.pm.PackageManager.MATCH_HIDDEN_UNTIL_INSTALLED_COMPONENTS

Abbildung 1 Parametertabelle mit möglichen Konstanten

3.2. Erhebungsphase

In der Erhebungsphase werden die Daten auf dem Smartphone gesammelt. Dazu werden sowohl die Rückgabewerte von Methoden als auch die Werte von Feldern und der Inhalt von Content Providern abgerufen. Um die Methoden aufzurufen bzw. Felder auszuwerten werden alle Klassen der Android API systematisch durchsucht. Als Ausgangsbasis dient hierzu die vorab geparste Liste der Klassen, welche in der Android API verfügbar sind. Anhand dieser Liste werden die Klassen mit Hilfe von Java Reflection geholt und darin deklarierte Methoden und Felder abgerufen. Statische Felder und statische Methoden ohne Parameter können dabei direkt ohne Erstellung von (Parameter-)Objekten aufgerufen werden. Für statische Methoden mit Parameter und alle nichtstatischen Methoden müssen vor dem Aufruf jedoch passende Objekte erstellt werden. Auf die Erstellung dieser Objekte wird im folgenden Abschnitt näher eingegangen.

3.2.1. Initialisierung

In der Initialisierungsphase werden die vorab geparsten Konstanten und vordefinierten Werte interpretiert sowie die benötigten Objekte erstellt.

Interpretation

Zur Interpretation der Konstanten und vordefinierten Werte wird der BeanShell Script Interpreter [6] verwendet. Dieser nimmt die Werte als String-Repräsentation und gibt das Auswertungsergebnis

zurück. Diese werden für die folgende Phase zwischengespeichert. Sollte ein Wert nicht interpretierbar sein, so wird dieser verworfen.

Objekterstellung

Für den Aufruf von Methoden mit Parametern kann es nötig sein, passende Objekte zu erstellen. Dazu werden die Typen der Parameter mit Hilfe von Reflection eruiert. Sollte es sich hierbei um primitive Datentypen wie `int` oder `boolean` handeln, so können diese mit Hilfe der Java-Wrapper Klassen wie Integer und Boolean direkt erstellt werden. Diese Werte können manuell definiert werden oder zufällig ausgewählt werden.

Die Namen der Parameter können nicht mit Reflection ausgelesen werden. Diese werden jedoch benötigt, um die Bedeutung der Parameter besser einschätzen und Werte vordefinieren zu können. Damit diese dennoch bekannt sind, wurde die Android API vorab geparst und die Parameternamen extrahiert. So können beispielsweise Werte wie `packageName` und `pid` vordefiniert werden. Ebenso werden die Parameternamen benötigt, um die vorab geparsten und interpretierten Konstanten den jeweiligen Parametern zuordnen zu können. Ist kein vordefinierter Wert vorhanden, so werden Parameter primitiven Typs zufällig erstellt.

Zur Erstellung von Objekten wird mit Reflection der Konstruktor/die Konstruktoren der Klasse abgerufen. In weiterer Folge werden die Konstruktoren aufgerufen und die erstellten Objekte gespeichert. Sollte ein Konstruktor auch Parameter benötigen, so werden diese analog zu den Methodenaufrufen erstellt.

3.2.2. Methodenauswahl

Zur Vermeidung von Seiteneffekten durch den Aufruf von Methoden wurden für die Datenbeschaffung nur Methoden ausgewählt, die in der Regel mit Informationsbeschaffung assoziiert werden. Dazu wurde das Set der aufzurufenden Informationen auf jene mit folgenden Präfixen eingeschränkt: `get`, `has`, `is`, `query` und `support`. Da diese Methoden nicht nur primitive Werte, sondern auch Objekte zurückgeben, werden alle Methoden vor der eigentlichen Datenbeschaffung gesondert aufgerufen und in zwei Gruppen geteilt. Wird ein primitiver Wert zurückgegeben, so wird der Rückgabewert für die Analyse gespeichert. Gibt eine Methode jedoch ein Objekt zurück, so wird dieses Objekt verwendet, um nach weiteren Methoden die dem beschriebenen Schema zu suchen.

Um die neu gefundenen Methoden des erhaltenen Objekts einordnen zu können, werden diese Methoden analog zu dem bereits beschriebenen Muster aufgerufen. Sollte einer der Methoden wieder ein Objekt zurückgeben, so wird auch dieses weiter erforscht. Da sich dieser Vorgang potentiell endlos fortsetzen kann, ist es möglich eine maximale Aufruftiefe zu setzen. Für die weiteren Experimente wurde diese auf 3 gesetzt.

Die Objekte die durch Methoden zurückgegeben werden müssen einer geparsten Klasse zugeordnet werden um vordefinierte Parameter und Konstanten zuordnen zu können. Dazu wird der Klassenname des Objekts mit Reflection eruiert und geprüft, ob dieser in der Liste der geparsten Klassen vorhanden ist. Das funktioniert jedoch nur, wenn es sich um ein Objekt exakt des geparsten Typs handelt. In der Praxis kann es sich jedoch auch um Objekte abgeleiteter Typen handeln. Deswegen wird in diesem Fall überprüft ob es sich um einen abgeleiteten Typ handelt und in diesem Fall die geparsten Informationen der Basisklasse verwendet.

3.2.3. Feldauswertung

Zusätzlich zum Aufruf von Methoden werden auch alle zugreifbaren Felder der API-Klassen ausgewertet. Um diese abzurufen, werden die für Methoden- und Konstruktorenaufrufe erstellten Objekte verwendet. Der Wert der Felder wird ebenso wie die Rückgabewerte in einer Liste mit Feldsignatur gespeichert. Die vollständige Liste wird dann in der Auswertungsphase analysiert.

3.2.4. Content Provider

Ein Content Provider auf Android bietet eine abstrakte Schnittstelle für Datenzugriff. Ähnlich einer Datenbank können die Inhalte in Form von Zeilen und Spalten über einen Cursor abgerufen werden. Die meisten der vom System bereitgestellten Content Provider, wie z.B. derjenige für Kontakte, benötigen eine Zugriffsberechtigung auf die vom Content Provider zur Verfügung gestellten Daten. Dies ist jedoch nicht bei allen Content Providern der Fall. Es ist jedoch immer eine gültige Inhalts-URI für den Zugriff erforderlich. Diese URIs beginnen mit dem Präfix *content://*. Um die Content-URIs der vom System bereitgestellten Content Provider zu erhalten, analysieren wir jeden zurückgegebenen String-Wert bei Methodenaufrufen und jedes abgefragte Feld. Wenn eine Zeichenfolge mit dem Präfix beginnt, wird die URI für den späteren Abruf gespeichert. Sobald die Methodenaufrufe und die Feldauswertung abgeschlossen sind, wird der Inhalt aller erkannten und zugänglichen Inhaltsanbieter durch Iteration über alle Zeilen und Spalten ausgegeben. Ähnlich wie bei Rückgabewerten und Feldern werden die ausgegebenen Daten für eine spätere Analyse gespeichert.

3.3. Auswertungsphase

Um fingerprintbare Informationen zu finden, werden die Daten von verschiedenen Geräten gesammelt. Diese gesammelten Informationen werden dann in der Auswertungsphase ausgewertet. Dazu werden die Daten der einzelnen Geräte bereinigt und daraufhin die verschiedenen Geräte verglichen.

3.3.1. Bereinigung

Durch die Vordefinition mehrerer Werte oder Konstanten für einen einzelnen Parameter können Methoden mehrfach aufgerufen werden. Ebenso kann durch den Aufruf von Methoden zurückgegebener Objekte eine Methode mehrmals auf verschiedenen Objekten aufgerufen werden. Daher kann eine einzelne Methode oder ein einzelnes Feld mehrfach in der Ergebnisliste vorkommen. Diese Vorkommnisse werden in der Bereinigungsphase zusammengefasst. Wenn sich die Rückgabewerte dieser Aufrufe nicht unterscheiden, werden die verschiedenen Vorkommen vereinheitlicht. Andernfalls werden die unterschiedlichen Werte für den Analyseschritt beibehalten.

Auf jedem Gerät werden alle erhältlichen Informationen zweimal gesammelt. Zwischen den beiden Erfassungsvorgängen wird die Anwendung vollständig entfernt. Danach wird sie mit einem anderen Signatur-Zertifikat wieder installiert, um die zweite Sammlung zu starten. Es wird ein anderes Signierzertifikat verwendet, um eine zweite Anwendung zu emulieren, die eine identische Tracking-Bibliothek verwendet, die den Benutzer wiedererkennen will. Dadurch können wir alle Informationsquellen eliminieren, die sich zwischen den einzelnen Anwendungsinstallationen unterscheiden, wie z.B. die Benutzer-ID oder das Datenverzeichnis der Anwendung. Darüber hinaus ermöglicht diese Maßnahme auch das Ausschließen aller Informationsquellen, die sich bei einem Neustart der Anwendung ändern, wie z.B. Thread- und Prozess-IDs, Prozesszeiten oder temporäre native Instanzbezeichnungen. Einige dieser Quellen könnten jedoch tatsächlich fingerabdruckfähige Informationen liefern. Beispielsweise könnte die verstrichene Prozesszeit verwendet werden, um auf die Geschwindigkeit des Geräts zu schließen. Die Interpretation aller sich ändernden Werte erfordert jedoch eine tiefere Kenntnis der Besonderheiten jedes dieser Werte. Dies ist jedoch außerhalb des Umfangs dieser Analyse. Die Daten werden daher aus der Ergebnisliste entfernt.

3.3.2. Analyse

Bei der Analyse werden die vorher bereinigten Daten verschiedener Geräte verglichen. Dazu werden zuerst Methoden, die nur auf einem Gerät existieren bzw. aufgerufen werden konnten gesammelt. Die Rückgabewerte der verbleibenden Methoden, also jene, die auf beiden Geräten aufgerufen werden konnten, werden im darauffolgenden Schritt verglichen. Dabei werden Methoden, bei denen sich der Rückgabewert nicht unterscheidet entfernt. Jene deren Rückgabewert sich unterscheidet werden beibehalten und als potentiell fingerprintbar markiert.

4. Ergebnisse

In den folgenden Paragraphen wird auf das Test Setup und die Ergebnisse näher eingegangen.

4.1. Test Setup

Die folgenden Experimente wurden auf mehreren Geräten durchgeführt um möglicherweise fingerprintbare Informationsquellen zu erkennen. Die Hauptgeräte waren zwei identische Google Pixel 4-Smartphones mit 64 GB Speicher. Beide wurden auf die neueste zum Zeitpunkt des Projekts verfügbare Softwareversion aktualisiert. D.h. auf den Geräten läuft Android 10 mit Sicherheitspatches vom 5. März 2020. Im ersten Experiment wurden die zwei Geräte auf bereits vorhandene, möglicherweise identifizierbare Informationen untersucht. Die Geräte wurden identisch eingerichtet. Daher stellt dieses Setup die minimale Menge an identifizierbaren Informationen dar. Nach Abschluss der Setup-Assistenten könnte ein Gerät in Wirklichkeit beispielsweise bereits unterschiedliche Netzwerke oder Anzeigegrößen konfiguriert und zusätzliche Anwendungen installiert haben.

In unserem zweiten Experiment wurden an einem der Geräte Änderungen der Einstellungen vorgenommen. Dieses Gerät wurde dann erneut auf fingerprintbare Informationen untersucht. Durch dieses Setup eines gleichen Gerätes aber unterschiedlicher Benutzerkonfiguration können Methoden, Felder und Content Provider gefunden werden, welche einen Benutzer identifizieren können. Darüber hinaus vergleichen wir auch zwei Samsung Galaxy S10e-Geräte mit Android 10, um zu eruieren, ob durch Herstelleranpassungen weitere fingerprintbare Informationsquellen entstehen.

Die App sammelt alle Daten, die entweder keine oder nur eine normale Berechtigung benötigen. Laut der API-Dokumentation schützen normale Berechtigungen Ressourcen, die nur ein sehr geringes Risiko für die Privatsphäre der Benutzerin oder des Benutzers darstellen [7]. Daher prüfen wir auch, ob dieses Statement in Bezug auf Fingerprinting zutrifft. Wo immer eine normale Permission erforderlich ist, wird die Anzahl der Informationsquellen die eine solche benötigen in Klammern angegeben. Die überwiegende Mehrheit der gefundenen Informationsquellen erfordert jedoch keine Permission.

4.2. Neugerätevergleich

Bei neuen Geräten bieten nur wenige Methoden potenziell fingerprintbare Informationen. Unser Framework hat 14 Methoden und einen Content Provider entdeckt, die Informationen zurückgaben, die zwischen den Geräten unterschiedlich waren. Ein Überblick über die gefundenen Quellen ist in Tabelle 1 aufgelistet. Diese liefern Informationen über die installierten Apps, freien Speicherplatz und Zugänglichkeitsdienste. Außerdem gibt es einige weniger stabile Identifikatoren, die eine Neuinstallation der Anwendung überstehen, sich jedoch beim Neustart des Geräts ändern. Da Smartphones tage- oder sogar wochenlang laufen können, könnten diese Informationen jedoch immer noch zum Fingerprinting verwendet werden. Acht Methoden und drei Felder liefern solche Informationen. Diese enthalten Informationen aus dem Media Player, wie zum Beispiel die aktuelle Position oder Dauer, die ID des aufrufenden Elternprozesses, und native Instanz-IDs der verwendeten Schriftart und des Threads Klasse. Außerdem stellen der Deskriptor des Eingabegerätes, die Version des Medienspeichers und die Unlock- und Startzeit des Nutzers erkannte instabile Identifikatoren dar. Während neue Geräte keine Fülle an potentiell fingerprintbaren Informationen bieten, werden Smartphones in der Regel von der Benutzerin oder vom Benutzer modifiziert und auf die Bedürfnisse eingestellt. Deshalb werden im nächsten Schritt angepasste Geräte verglichen.

Tabelle 1 Übersicht über die gefundenen Informationsquellen auf Neugeräten. (M = Methoden, F = Felder, CP = Content Provider)

Typ	**#M**	**#F**	**#CP**
Instabil			
Medien Zeitstempel	3	-	-
Native Instanz-Ds	-	2	-
Nutzer Start- & Freigabezeitpunkt	2	-	-
Eingabegerät	1	-	-
Unkategorisiert	-	1	-
PPID	1	-	-
Applikationen	1	-	-
Summe	**8**	**3**	**0**
Stabil			
Applikationen	7	-	-
Verfügbarer Speicher	5	-	-
Services zur Barrierefreiheit	2	-	-
Klingeltöne (Zeitstempel des Einlesens)	-	-	1
Summe	**14**	**0**	**1**

4.2.1. Angepasste Geräte

Bei diesem Aufbau wurde eines der Testgeräte durch Änderung von Einstellungen und Installation von Anwendungen angepasst. 530 Methoden, 101 Felder und 72 Zeilen in den Content Providern lieferten unterschiedliche Werte. Die Klassifizierung dieser Informationsquellen ist in Tabelle 2 zu finden. Zusätzlich zu den Methoden und Content Provider-Einträgen, die im vorherigen Experiment gefunden wurden, lassen sich daraus verschiedene Einstellungen und Anpassungen am Gerät ableiten. Die auffälligste Veränderung der Ergebnisse wurde durch die Änderung der Anzeige- und Schriftgröße erreicht. Die ermittelten Informationsquellen geben Werte wie Anzeigeeigenschaften, Standardgrößen verschiedener UI-Komponenten oder Größen von Bitmaps an.

Eine weitere auffällige Änderung, die sich auf viele Methoden auswirkte, war die Änderung der Systemsprache. Während einige Methoden und Felder die Systemsprache direkt melden, kann aus anderen Methoden und Feldern auf die Sprache geschlossen werden. Dazu gehören zum Beispiel Methoden, die Namen von Speichergeräten wie *Internal Storage* vs. *Interner Speicher* oder Schaltflächenbeschriftungen wie *On* vs. *An* melden. Darüber hinaus wirkte sich die Änderung des Systemgebietsschemas auch auf viele Methoden, Felder und Content Provider aus.

Auch verschiedenste Einstellungen zur Barrierefreiheit können über die Android-API abgefragt werden. Es ist zum Beispiel möglich, die eingestellte Zeit zum Reagieren, deaktivierte Animationen oder Tastenwiederholungseinstellungen zu erkennen. Mit Hilfe der Captions-API ist es möglich, die Schriftgröße, die Farben und das Gebietsschema abzufragen, die speziell für Untertitel konfiguriert wurden. Darüber hinaus könnte der Gerätename, falls er angepasst wurde, eine Benutzerin oder einen Benutzer eindeutig identifizieren. Interessanterweise erfordert der Zugriff auf den Gerätenamen über die Methode *android.bluetooth.BluetoothAdapter.getName()* eine normale Permission, während der Zugriff auf den Gerätenamen ohne Permission über Content Provider erfolgen kann. Die Android-API erlaubt auch die Abfrage von UUIDs von verbundenen Bluetooth-Geräten, wenn die dazugehörende normale Permission im Manifest deklariert wurde. Die AudioManager-API

ermöglicht den Zugriff auf die von der Benutzerin oder vom Benutzer eingestellte Lautstärke. Während die Hauptlautstärke häufiger geändert werden kann, ermöglicht die API auch den Zugriff auf andere Streams, wie z.B. die Headset-, Bluetooth- oder HDMI-Lautstärke. Zusätzlich ermöglichen es Sicherheits-APIs wie KeyStore, FingerprintManager oder BiometricManager abzufragen, ob eine Benutzerin oder ein Benutzer eine Gerätesperre eingerichtet oder einen Fingerabdruck oder andere biometrische Daten registriert hat.

Weitere Informationsquellen, die erkannt wurden, sind die Liste der installierten Widgets und Anwendungen. Dazu gehören detaillierte Informationen über die Anwendungen, wie z.B. Benutzer-ID, Ressourcenverzeichnisse und letzte Aktualisierungszeit. Änderungen an Standardanwendungen für bestimmte Aufgaben, wie z.B. Nachrichten oder der Telefonapp können ebenfalls abgefragt werden. Der aktivierte Startbildschirm kann z.B. nach unterstützten Funktionen oder Symbolgrößen abgefragt werden. Das Framework erkannte auch einige instabile Informationsquellen, die sich zwischen den Neustarts der Geräte ändern. Dazu gehören Methoden, die den aktuellen Inhalt der Zwischenablage melden oder die von einem Content Provider gemeldete Anzahl von Bootvorgängen.

Zusätzlich zu den zwei Google Pixel-Geräten haben wir auch zwei Samsung Galaxy S10e mit Hilfe des Samsung Remote Test Lab analysiert. Auf beiden Geräten läuft Android 10. Die Geräte wurden von uns nicht speziell angepasst, aber sie lieferten dennoch wertvolle Identifikatoren. Die zusätzlich identifizierten Informationsquellen sind in Tabelle 3 dargestellt. Im Gegensatz zu den Pixel-Geräten liefern diese beiden Samsung-Geräte über den Settings-Content Provider eindeutige Identifikatoren. Daher hat jede Anwendung Zugriff auf eine Geräte-ID, einen Zugriffstoken und eine Smart Tethering GUID. Wenn die Funktion "Call & Text auf anderen Geräten" aktiviert ist, kann eine SIP-Adresse abgefragt werden, die aus der Geräte-ID und dem Access-Token besteht. Darüber hinaus kann auf die E-Mail-Adresse zugegriffen werden, die zur Anmeldung für das Samsung-Konto verwendet wurde. Zusätzlich zu diesen Kennungen hat das Framework mehrere Samsung-spezifische Einstellungen erkannt, die sich zwischen den beiden Geräten unterscheiden. Weitere unterschiedliche Werte sind unterschiedliche Zeitstempel, der Zustimmungsstatus zu verschiedenen Nutzungsbedingungen und Softwareversionen von proprietären Diensten.

Tabelle 2 Übersicht über die gefundenen Informationsquellen auf eingerichteten Geräten. Die Zahlen in Klammern stehen für Methoden, welche eine normale Permission benötigen. (M = Methoden, F = Felder, CP = Content Provider)

Type	**#M**	**#F**	**#CP**
Instabil			
Inhalt der Zwischenablage	13	-	-
Anzahl der Boot-Vorgänge	-	-	2
Unkategorisiert	-	1	-
Sum	**13**	**1**	**2**
Stabil			
Anzeigegröße von Elementen	88	39	2
Sprache	122	3	-
Systemgebietsschema	62	2	1
Netzwerkdetails	44 (15)	10	4
Apps	13	27	-
Netzbetreiber, SIM	29	2	6
Aktivierte Funkschnittstellen (Mobile Daten, WiFi, NFC, GPS)	23 (19)	-	2
Einstellungen zur Barrierefreiheit	6	1	17
Währung	24	-	-
Eingabemethoden	16	1	3
Bluetooth Einstellungen und Geräte	18 (17)	-	1
Zeitzone	13	3	2
Unkategorisiert	11	3	1
Bildschirmeinstellungen (Nachtmodus, Automatische Rotation, …)	5	2	8
Farben	9	3	-
Hintergrundeinstellungen und -farben	12	-	-
Untertiteleinstellungen	3	2	4
Nicht-Stören Modus, Benachrichtigungslautstärke	4	-	4
Standardapps (Nachrichten, Wählapp, Launcher)	6	-	1
Klingelton (Derzeit gesetzter Klingelton, Liste der hinzugefügten, inkludiert Zeitstempel der Hinzufügung)	5	1	-
Soundeffekte	-	-	6
Laustärkeeinstellungen	5	-	-
Schriftgröße	2	1	-
Gerätename	1 (1)	-	2
Zeit- und Datumsformat	2	1	-
Versch. Zeitstempel	-	-	2
Widgets	2	-	-
Gerätesperre	2	-	-
Benachrichtigungseinstellungen	-	-	2
Komplexität der gewählten Sperrmethode	1 (1)	-	-
Anpassungsstatus	-	-	1
Bildschirmschoner	-	-	1
Aktivierter Assistant	-	-	1
Nutzerbeschränkungen	1	-	-
Einstellungskacheln (Aktiviert, Anordnung)	-	-	1
Sicherheit	1	-	-
Summe	**530**	**101**	**72**

Tabelle 3 Übersicht über die identifizierten Informationsquellen auf zwei Samsung Galaxy S10e. (M = Methoden, F = Felder, CP = Content Provider)

Type	**#M**	**#F**	**#CP**
Samsung-spezifische Einstellungen	-	-	12
Verschiedene Zeitstempel (Sicherheitsrichtlinienüberprüfungszeitpunkt, Änderungszeitpunkt des Sperrbildschirmhintergrund, …)	-	-	9
Systemgebietsschema	1	-	7
Zustimmung zu EULA (Status, Zeitstempel)	-	-	7
Geräte-ID, Tethering-GUID	-	-	5
Klingelton	1	-	3
Soundeffekte	-	-	4
WiFi MWIPS Special SSIDs	-	-	4
Software Versionen	-	-	3
WiFi Learning Score	-	-	2
Einstellungen zur Barrierefreiheit	-	-	2
Logginganzahl	-	-	2
Samsung Account E-Mail-Adresse	-	-	1
WiFi P2P Gerätename	-	-	1
Access Token	-	-	1
Unkategorisiert	-	-	1
Farben	-	-	1
Ressourcen-IDs	-	-	1
Theme	-	-	1
Sum	**2**	**0**	**67**

5. Fazit

In diesem Bericht wurde ein Framework vorgestellt, das dazu dient, fingerprintbare Methoden, Felder und native Content Provider auf Android zu finden. Das Framework automatisiert dabei die mühsame Suche nach fingerprintbaren Informationsquellen. Während einige Informationsquellen im Laufe der Zeit entfernt wurden oder nun Zugriffsberechtigungen erfordern, gibt es immer noch viele Methoden, Felder und Content-Provider, die es erlauben, auf Informationen über die Benutzerin oder den Benutzer zuzugreifen oder auf diese zu schließen. Dazu gehören anbieter- und geräte-spezifische Informationen, betriebssystemspezifische Informationen und Informationen, die es erlauben, auf benutzerdefinierte Einstellungen zu schließen. Dieser automatisierte Ansatz erlaubt es, über das hinauszugehen, was durch manuelle Analyse erkannt werden kann. Durch das Auslesen aller verfügbaren Daten entdeckten wir undokumentierte Anbieteranpassungen, die eindeutige Gerätekennungen und sogar Zugriff auf die Mailadresse der Benutzerin oder des Benutzers bieten.

Das hier vorgestellte Framework ermöglicht es, mögliche fingerprintbare Informationsquellen zu erkennen, bevor diese zur Verfolgung von Benutzerinnen und Benutzer verwendet werden können. Bei der Verwendung im Entwicklungsprozess neuer oder modifizierter Android-Versionen könnte das Framework Informationsquellen aufdecken, bevor diese zur Benutzerverfolgung missbraucht 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] E. P., „How unique is your web browser?,“ in *PET*, 2010.

[3] 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.

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

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

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

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