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

EINFLUSS VON HERSTELLERANPASSUNGEN AUF DIE GERÄTESICHERHEIT

Version 1.0 vom 14.04.2021
Gerald Palfinger – gerald.palfinger@iaik.tugraz.at

Zusammenfassung: Um sich von anderen Herstellern abzusetzen und den Benutzerinnen und Benutzern einen Mehrwert zu bieten passen Hersteller von Android-Smartphones ihre Geräte an. Dies umfasst sowohl vorinstallierte Programme als auch das Betriebssystem und seine Funktionen selbst. In diesem Bericht wird auf Sicherheitsprobleme eingegangen, die durch diese Anpassungen entstehen können. Ebenso wird eine Analyse durchgeführt, wie sich Herstelleranpassungen auf die Fingerprintbarkeit von Geräten auswirken. Dazu wurden Geräte verschiedener Hersteller verglichen. Dabei wurde festgestellt, dass durch Anpassungen an der Software als auch durch Hardwareunterschiede weitere potentiell fingerprintbare Informationsquellen zugänglich sind.

Inhaltsverzeichnis

1. Einleitung

Das offene Design von Android ermöglicht es den Herstellern von Smartphones, das Betriebssystem anzupassen und weitere Funktionalität hinzuzufügen. Während Smartphones abseits von Apple auf derselben Basis, also dem im Android Open Source Project (AOSP) bereitgestellten Code basieren, ermöglicht die Offenheit des Betriebssystems weitreichende Anpassungen seitens der einzelnen Hersteller. So können Geräte gegenüber anderen Herstellern differenziert und Alleinstellungsmerkmale geschaffen werden. Diese Anpassungen können vorinstallierte Programme und Services umfassen, die softwareseitige Unterstützung spezieller Hardwarefunktionen wie erweiterte Eingabemethoden aber auch Änderungen am Betriebssytem selbst bis hin zum Bootloader. Diese Anpassungen bergen jedoch auch die Gefahr, dass unbeabsichtigt neue Sicherheitslücken eingeführt werden. So wurde beispielsweise in [1] gezeigt, dass herstellerspezifische Anpassungen an der Fastboot-Schnittstelle zur Umgehung von Secure Boot oder der Gerätesperre missbraucht werden konnten. In diesem Bericht wird anhand einer Auswahl

an Schwachstellen gezeigt, welche Sicherheitslücken durch Herstelleranpassungen entstehen können.

Damit auch Drittapplikationen von spezifischen Hardwarefunktionen Gebrauch machen können, erweitern Gerätehersteller mitunter auch die Programmierschnittstelle von Android um weitere herstellerspezifische Funktionen. Diese bergen nicht nur die Gefahr neuer Sicherheitslücken, sondern können potentiell auch die Fingerprintbarkeit der Geräte erhöhen. Zusammen mit weiteren Diensten können dadurch Nutzerinnen und Nutzer über Applikationen hinweg besser wiedererkannt werden. In [2] wurde gezeigt, dass durch diese Herstelleranpassungen durchaus auch neu hinzugefügte Datenschutzvorkehrungen umgangen werden können. So wurde unter anderem gezeigt, dass auf Samsung-Geräten der Zugriff auf eine gerätespezifische ID möglich ist, nachdem mit Android 10 [3] eigentlich alle eindeutigen Identifikationsmerkmale entfernt wurden. Weitere Untersuchungen sind jedoch notwendig, um herauszufinden wie sich andere Herstelleranpassungen auf die Fingerprintbarkeit der Geräte auswirken. Deswegen wurde als Teil dieser Studie in Experimenten untersucht, welche dieser Herstelleranpassungen Auswirkungen auf die Fingerprintbarkeit von Geräten haben. Dazu werden Geräte von verschiedenen Herstellern untersucht und verglichen. Um auf möglichst viele verschiedene Geräte Zugriff zu haben werden verschiedene Remote-Geräteservices verwendet.

2. Hintergrund

Android ist ein Smartphone-Betriebssystem welches von der Open Handset Alliance unter der Leitung von Google entwickelt wird. Das Betriebssystem kann von jedem Gerätehersteller angepasst werden. Von dieser Möglichkeit wird auch umfassend Gebrauch gemacht. Dadurch versuchen sich die Hersteller von anderen abzugrenzen und so den Nutzern und Nutzerinnen einen Mehrwert zu bieten. Durch die Anpassungen können jedoch unbeabsichtigt auch neue Sicherheitslücken entstehen. Im folgenden Abschnitt werden wissenschaftliche Arbeiten beleuchtet, welche sich mit dem Problem befassen. In der darauffolgenden Sektion wird kurz auf Remote-Geräteservices eigegangen, welche es ermöglichen auf eine Vielzahl unterschiedlicher Smartphones zuzugreifen. Diese werden für die Evaluierung der Fingerprintbarkeit von Herstelleranpassungen verwendet.

2.1. Verwandte Arbeiten

In [4] werden potentielle Schwachstellen in durch Hersteller angepassten oder hinzugefügten Applikationen gesucht. Dazu wurden Abbilder des installierten Betriebssystems von verschiedenen Modellen verschiedener Hersteller verglichen. Dabei werden die vorinstallierten Applikationen eingeteilt in nicht angepasste, angepasste und vorinstallierte Drittsoftware. Weiters werden die angeforderten Berechtigungen der Applikationen untersucht um überprivilegierte Applikationen zu erkennen. Die Studie zeigt, dass mehr als 8 von 10 Applikationen Berechtigungen anforderten, die sie für die Ausführung ihrer Tätigkeit nicht zwingend benötigten. Zusätzlich wurden die Abbilder auch auf Schwachstellen untersucht. Ein Großteil der gefundenen Probleme konnte dabei auf Herstelleranpassungen zurückgeführt werden.

In [1] werden Schwachstellen im fastboot-Dienst aufgedeckt, welche durch Anpassungen dieses Dienstes hervorgerufen wurden. Fastboot ermöglicht die Kommunikation mit dem Bootloader via USB. Dazu muss das Smartphone im sogenannten Fastboot-Modus gestartet werden. Dieser ermöglicht es beispielsweise den Bootloader nach einem Reset zu entsperren oder das Gerät zu flashen. Der Modus wird jedoch oft von den Smartphone-Herstellern um weitere Funktionalität erweitert. Im Paper wird ein Tool vorgestellt, welche diese Erweiterungen erkennt und auflistet. In einem weiteren Schritt wurde dann nach Schwachstellen in den Erweiterungen gesucht. Die gefundenen Schwachstellen erlauben es Daten von Google-Geräten zu extrahieren, während auf den getesteten Motorola und OnePlus-Geräten sogar Secure Boot und die Gerätesperre umgangen werden kann. Manche der entdeckten Schwachstellen konnten auch über modifizierte Kopfhörer oder USB-Ladegeräte ausgenutzt werden. Dadurch musste der Nutzer nicht dazu gebracht werden, den Fastboot-Modus manuell zu starten.

Eine weitere über USB ausnutzbare Schwachstelle in Herstelleranpassungen wird in [5] diskutiert. Diese wird ermöglicht da Anpassungen durch die Hersteller HTC und Samsung es erlaubten Kommandos via USB-Anschluss an das Modem des Smartphones zu senden. Dabei werden sogenannte AT Befehle gesendet. Diese werden beispielsweise auch von GSM verwendet um Anrufe zu tätigen oder um auf Kontakte auf der SIM-Karte zuzugreifen. Im Fall der untersuchten Samsung-Smartphones wurde diese Kommandoschnittstelle um weitere Funktionalität erweitert. So konnten über AT Befehle auch Dateien vom Smartphone abgerufen oder die Firmware des Smartphones aktualisiert werden. Diese Funktionalität wurde eigentlich für die Steuerung des Smartphones über die von Samsung entwickelte dazugehörige Desktopapplikation eingebaut. Sie kann jedoch von jedem beliebigen angeschlossenem Ladegerät ausgenutzt werden. Dadurch können auf dem Smartphone gespeicherte Daten ausgelesen werden oder über die Aktualisierungsfunktion das Boot-Image ausgetauscht werden, wodurch das Smartphone durch den Angreifer übernommen werden kann.

Bei Chizpurfle [6] handelt es sich um einen Gray-Box Fuzzer, welcher durch Hersteller angepasst oder bereitgestellte Dienste auf Bugs und Sicherheitslücken untersucht. Dieser Fuzzer kann proprietäre Dienste erkennen und automatisch mit Hilfe von Fuzzing diese Dienste untersuchen. Dabei werden von Chizpurfle systematisch Eingabedaten erstellt und die Dienste mit diesen aufgerufen. Dabei wird überprüft, ob der Dienst sich nach Verarbeitung der gesendeten Daten abnormal verhält. Konkret überprüft Chizpurfle, ob der untersuchte Dienst als kritisch markierte Logausgaben ausgibt, nicht mehr reagiert oder mit einer Fehlermeldung abstürzt. Getestet wurde Chizpurfle auf einem Samsung Galaxy-Gerät. Dabei wurden bei der Ausführung von 34.645 Tests an 2.272 Methoden neun Fehler erkannt. Zwei davon konnten für Denial-of-Service Attacken missbraucht werden.

In [2] wurde die Fingerprintbarkeit von Android-Geräten untersucht. Dabei wurden Informationen aus verschiedenen Drittprogrammen zugänglichen Informationsquellen gesammelt, welche Nutzerinnen und Nutzer bzw. Geräte identifizieren können. Dabei wurde festgestellt, dass durch Herstelleranpassungen auf Samsung-Geräten im Vergleich zum konventionellen Android weitere Informationsquellen zur Verfügung stehen. Auf diesen Geräten wurde festgestellt, dass Drittanbierterapplikationen ohne Berechtigung auf eindeutige Geräteidentifikationsmerkmale und weitere private Daten zugreifen können.

2.2. Android Remote-Geräteservices

Durch die offene Natur von Android gibt es eine Vielzahl an Gerätemodellen von verschiedenen Herstellern. Um Applikationen auf verschiedenen Geräten testen zu können gibt es verschiedene Onlineservices, welche den Remote-Zugriff bzw. eine Testausführung auf einer Vielzahl verschiedener Modelle erlauben. Auf solche Services wird auch in den folgenden Experimenten zugegriffen. Im speziellen wurden dabei zwei Services verwendet. Ein Großteil der Untersuchungen wurde auf Geräten des Google Firebase Test Labs [7] durchgeführt. Weitere Geräte wurden über die AWS Device Farm [8] komplettiert.

Im Firebase Test Lab standen zum Zeitpunkt der Durchführung der Untersuchungen 105 physische Android-Geräte von Asus, Huawei, HTC, Google, LG Mobile, Motorola, Nokia, Samsung, Sony und Xiaomi zur Verfügung. In der AWS Deviec Farm konnte auf 52 verschiedene Android-Modelle von Asus, Dell, Samsung, Google, Motorola und Sony zugegriffen werden.

3. Mehodik

Um Daten von den Geräten zu sammeln, wird ein automatisiertes Framework verwendet. Dabei werden Rückgabewerte von Methodenaufrufen, Inhalte von Feldern und der Inhalt von Content Providern abefragt. Damit sollten alle wesentlichen einem Drittprogramm zugänglichen Informationsquellen abgerufen werden. Um die Informationen abzurufen wird das Framework aus [2] verwendet. Dieses wurde um ein Testsystem erweitert, sodass die Datensammlung auch über das Testframework gestartet werden kann. Weiters wurde die Performance verbessert, um die Daten schneller zu erheben und so innerhalb etwaiger Timeouts der Geräteanbieter zu bleiben.

Ebenso wurde eine Cloud-Uploadfunktion eingebaut, um die Ergebnisse direkt in die Cloud hochladen zu können. Schlussendlich wurde auch die Abdeckung an Methoden verbessert. Im Folgenden wird auf die Methodik genauer eingegangen.

Das Framework nutzt als Ausgangspunkt eine Liste aller bekannten Methoden und Konstruktoren in der Android API. Dazu wurde die API Dokumentation [9] geparst. Ausgehend von dieser geparsten Liste werden Objekte der Klassen erstellt und mit Hilfe dessen Methoden aufgerufen sowie Felder ausgewertet. Zusätzlich dazu werden mögliche Konstanten aus der Dokumentation geparst welche beim Aufruf von Methoden und Konstruktoren verwendet werden. Mit Hilfe von Java Reflection werden die Konstruktoren und Methoden aufgerufen. Für Parameter von Methoden und Konstruktoren für die keine Konstanten vorliegen werden entweder manuell definierte Werte verwendet oder Standardwerte verwendet. Die Zuteilung der manuell vordefinierten Werte erfolgt über den Parameternamen. Die Zuteilung der Standardwerte wird über den Parametertyp eruiert.

Um etwaige Seiteneffekte wie Änderungen an der Applikation oder am System oder auch Applikationsabstürze beim Aufruf von Methoden zu minimieren werden nur Methoden aufgerufen, welche normalerweise Informationen bereitstellen. Diese werden anhand des Präfixes des Methodennamen ausgewählt. Für die vorliegende Studie werden die untersuchten Methoden auf die Präfixe `get`, `has`, `is`, `query`, `check`, `support` und `scan` eingeschränkt. Wird ein primitiver Wert (also beispielsweise Integer oder String) zurückgegeben, so wird dieser für die weitere Analyse gespeichert. Wird jedoch das Objekt einer Klasse zurückgegeben, so werden darin enthaltene Methoden, sofern sie einen der oben erwähnten Präfixe im Methodennamen haben, ebenso aufgerufen. Ebenso wird das Resultat der `toString()` Methode für die Analyse abgerufen. Die Tiefe der Aufrufe wird auf zwei Ebenen eingeschränkt. Zusätzlich zu den Rückgabewerten der Methoden werden auch alle Inhalte von Feldern der erstellten bzw. erhaltenen Objekte abgerufen. Auch diese werden für die weitere Analyse gespeichert.

Zusätzlich zu den Rückgabewerten von Methoden und Feldern werden auch die Inhalte von Content Providern abgerufen. Mit Hilfe der Content Provider Architektur kann auf Daten in Systemdiensten zugegriffen werden [10]. Ein Content Provider kann auch von Applikationen implementiert werden um Daten über eine einheitliche Schnittstelle anderen Applikationen zur Verfügung zu stellen. Dabei ermöglicht das System es auch Zugriffsrechte über das Berechtigungssystem von Android zu verwalten. Für die Auswertungen in diesem Bericht konzentrieren wir uns jedoch auf die Content Provider welche vom System bzw. von vorinstallierten Diensten bereitgestellt werden. Dazu wurden im vorherigen Schritt alle Content Provider URIs (also alle Strings welche mit content:// starten) gesammelt. Diese Content Provider werden in diesem Schritt nun ausgelesen. Dabei können natürlich nur jene Content Provider ausgelesen werden, welche auch ohne erhöhte Berechtigung auslesbar sind. Die zugreifbaren Daten werden für die weitere Analyse gespeichert.

Im letzten Schritt werden alle gesammelten Daten ausgewertet. Dazu werden die Daten zuerst bereinigt. Konkret bedeutet dies, dass Duplikate entfernt werden. Das heißt, dass also beispielsweise Aufrufe von Methoden, welche trotz unterschiedlicher Parameter oder Aufrufobjekte den gleichen Wert zurückgeben, entfernt werden. Bei der Auswertung der gesammelten Daten wird konkretes Augenmerk auf die Methoden, Felder und Content Provider gelegt, welche nur auf Geräten einzelner Hersteller verfügbar bzw. aufrufbar sind. Ebenso wird nach potentiell persönlichen Daten gesucht sowie (eindeutige) Geräteidentifizierungsmerkmale gesucht.

4. Auswertung

In diesem Abschnitt werden die Ergebnisse der durchgeführten Analyse dargestellt. Die Analyse wurde auf Geräten verschiedener Hersteller durchgeführt, um die Einflüsse verschiedener Herstelleranpassungen auf die Fingerprintbarkeit von Mobilgeräten zu eruieren. Für die Analyse wurde das Framework auf Geräten der Hersteller Samsung, Nokia, Motorola, LG Mobile, Huawei und Asus ausgeführt. Zur Ausführung wurden Testgeräte des Firebase Test Labs [7] und der AWS Device Farm [8] verwendet. Die Resultate der Auswertung sind in Tabelle 1 zusammengefasst. Die gefundenen Ergebnisse sind nach der Anzahl der Methoden, Content Provider und Felder sortiert

und in Gruppen zusammengefasst. Im nun folgenden Abschnitt wird auf den Inhalt der einzelnen Gruppen genauer eingegangen.

Wie in der Tabelle 1 ersichtlich ist, macht den größten Teil der Unterschiede zwischen den Geräten sich unterscheidende Indizes aus. In dieser Gruppe sind vorrangig Unterschiede bei den Indizes von Content Providern enthalten. Im Speziellen handelt es sich um Content Provider, welche Einstellungen des Systems bereitstellen. Diese Unterschiede sind darauf zurückzuführen, dass unterschiedliche Herstelleranpassungen verschiedene Einstellungen anbieten um so das Smartphone besser anpassbar an die Nutzerin bzw. den Nutzer zu machen. Ebenso kann es durch zusätzliche Hardware zu weiteren Einstellungsmöglichkeiten kommen. Durch die unterschiedliche Anzahl an Einstellungsmöglichkeiten unterscheiden sich jedoch auch die Indizes der auf mehreren oder allen Geräten verfügbaren Einstellungen. Dadurch kommt es zur großen Anzahl der erkannten Unterschiede. Zwischen den Geräten der verschiedenen Hersteller unterscheiden sich jedoch nicht nur die Indizes der Einstellungsoptionen, sondern auch der Inhalt. Dadurch dass diese Einstellungsmöglichkeiten auch jedem App zugreifbar sind steigt zusätzlich auch die Fingerprintbarkeit des Geräts.

An zweiter Stelle der erkannten Unterschiede stehen Unterschiede bei den Ressourcen IDs. Durch die aufsteigende Nummerierung der vom System bereitgestellten Ressourcen unterscheiden sich diese auf den Geräten der verschiedenen Hersteller. Auch im Bereich der Telefon-APIs gibt es Unterschiede bei den Rückgabewerten. So unterscheiden sich beispielsweise Interfacenamen, Softwareversionen oder IDs.

Große Unterschiede gibt es auch beim Aufbau der durch Content Provider bereitgestellten Informationen. Im Vergleich zur Android API gibt es bei den Content Providern weniger Vorgaben zum Aufbau der dahinterstehenden Datenstrukturen. Dadurch können von den Herstellern weitere Spalten oder auch ganze Content Provider hinzugefügt werden um zusätzliche Funktionen zu unterstützen. So enthalten beispielsweise auf den untersuchten Huawei-Geräten die Kontaktdatenbanken eine weitere Spalte names notification_tone, welche es erlaubt einen kontaktspezifischen Benachrichtigungston zu setzen.

Viele der erkannten Unterschiede sind auf Hardwareunterschiede zurückzuführen. So reagieren viele Methoden auf unterschiedliche Bildschirmgrößen oder Differenzen bei der Auflösung und Pixeldichte. Ebenso unterscheideen sich die unterstützen Funktionen der Bildschirme. So bieten manche Geräte HDR, breitere Farbunterstützung oder Wifi Display an. Auch bei den Funktionen im Bereich Audio gibt es große Unterschiede. So unterstützen die untersuchten Geräte verschiedene Effekte, eine unterschiedliche Anzahl an Kanälen, Codecs, Abtastraten und unterschiedliche Lautstärkeabstufungen. Ebenso wurden Unterschiede bei der Anzahl an unterstützten SIM-Karten, verschiedene Speichergrößen, unterschiedliche RAM-Austattung, IR-Support, und GPS-Hardware erkannt.

Durch die Anpassungen kommt es auch zu Implementationsunterschieden zwischen den verschiedenen Herstellern. So unterscheiden sich beispielsweise Buffergrößen, Umgebungsvariablen, Scrollgeschwindigkeiten oder Flags. Weitere Informationsquellen wurden im Bereich der WLAN-, Bluetooh- und Mobilunterstützung gefunden. So unterstützen die Geräte unterschiedliche Frequenzen und Funktionen. Ebenso unterscheiden sich Timeouts in diesem Bereich.

Ein weiterer Bereich der erkannten Unterschiede umfassen Gebietsschemen, Sprachen und Zeitzonen. So unterstüzten bzw. verwenden die untersuchten Geräte unterschiedliche Gebietsschemen, Sprachen und Zeitzonen. Ebenso unterscheiden sich die Buildinformationen, welche hauptsächlich über die Klasse android.os.Build abgerufen werden kann. Auch durch die API abrufbare Softwareversionen unterscheiden sich zwischen den Geräten.

Die vorinstallierten Applikationen unterscheiden sich je nach Hersteller ebenso substantiell. Dies betrifft auch den Bereich der Eingabemethode, bei denen oft eine herstellerspezifische Tastatur

verwendet wird. Auch die Unterstützung weiterer Eingabehardware, wie Stifte, kann über die API abegfragt werden.

Tabelle 1 Übersicht über die identifizierten Informationsquellen auf den untersuchten Geräten. (M = Methoden, F = Felder, CP = Content Provider)

Typ	**#M**	**#CP**	**#F**	**Σ**
Indizes	2	117	0	119
Ressourcen (IDs)	14	0	98	112
Telefon/SIP (Features, Unterstützung, Netzwerktypen,...)	73	1	3	77
Aufbau Content Provider Datenbanken	66	0	1	67
Einstellungen	18	39	1	58
Implementationsdetails	39	2	15	56
Bildschirmgröße (physische bzw. Pixelgröße/-anzahl)	49	0	2	51
Bluetooth/WLAN/Mobil	36	1	5	42
Gebietsschema	40	0	0	40
Audiounterstützung (Funktionen, Hardwaresupport)	35	0	2	37
Build Informationen (Versionen, Gerätename,...)	2	2	22	26
Applikationsmanagement/Berechtigungen	21	3	2	26
Zeitzone (Einstellung, Unterstützung)	16	1	4	21
SIM Unterstützung	17	0	0	17
Content Provider Einträge	0	13	0	13
Eingabegeräte/Eingabemethode	8	4	0	12
Speichergröße (freier/gesamter interner Speicher)	12	0	0	12
Benachrichtigungstöne	6	6	0	12
Sprache (Einstellung, Unterstützung)	12	0	0	12
Softwareversionen	5	0	2	7
Aufbau Einstellungsspeicher	0	0	6	6
Bildschirm (unterstütze Funktionen, Helligkeit)	6	0	0	6
Farben	6	0	0	6
Verschlüsselungsalgorithmen	5	0	0	5
User Agent (Java, MMS)	3	0	0	3
Standardapplikationen (Wählprogramm, SMS)	3	0	0	3
Gerätename	1	1	0	2
Nutzerunterstützung	2	0	0	2
Speichergröße (Arbeitsspeicher)	2	0	0	2
Wallpaper/Screensaver	1	1	0	2
Verschlüsselungstyp	2	0	0	2
Benachrichtigungen	2	0	0	2
GPS Hardware (Modell, Herstellungsjahr)	2	0	0	2
Gerätefunktionen	2	0	0	2
Kameras	2	0	0	2
IR Support	1	0	0	1
Datenträger	1	0	0	1
DRM Engines	1	0	0	1
SUMME	513	191	163	867

Während sich Herstelleranpassungen durch Soft- und Hardwareunterschiede vor allem auf Einstellungsmöglichkeiten und das Content Provider-System auswirken, wurden im Gegensatz zu [2] keine weiteren eindeutigen Identifikationsmerkmale oder andere Informationslecks auf den untersuchten Geräten gefunden. Es wurde jedoch eine Vielzahl an herstellerspezifischen Feldern auf den Geräten aller Hersteller erkannt. Diese beinhalten beispielsweise herstellerspezifische Berechtigungsstrings (z.B. com.huawei.permission.HUAWEI_KERNEL_HOTFIX) oder Konstanten für herstellerspezifische Funktion (z.B. moto_show_brazil_settings). Da diese jedoch konstant sind haben sie auf die Fingerprintbarkeit nur eine geringe Auswirkung. So können sie zwar zur Bestimmung des Herstellers und gegebenenfalls auch des Gerätemodells verwendet werden, jedoch nicht zur Wiedererkennung einzelner Nutzer. Zur Wiedererkennung des Herstellers bzw. des Gerätemodells können zwar auch die Informationen verwendet werden, welche über android.os.Build zugänglich sind. Als einfache Gegenmaßnahme gegen das Fingerprinting des Gerätemodells könnten die über android.os.Build verfügbaren Informationen jedoch verschleiert werden. Wie diese Studie jedoch zeigt reicht eine solche einfache Gegenmaßnahme – ähnlich wie beim Fingerprinting des Browsers [11] – jedoch nicht aus, da der Hersteller und das Gerätemodell durch die vielen gefundenen Eigenheiten trotzdem bestimmt werden kann.

4.1. Einschränkungen

Für die Analyse wurden großteils Geräte von Online-Testsystemen verwendet, um eine hohe Abdeckung von Herstellern zu erreichen. Dadurch konnten die Daten vollautomatisch erhoben werden. Diese Systeme bieten jedoch kaum Kontrolle über die verwendeten Geräte an. Dadurch konnten diese weder zurückgesetzt noch angepasst werden. Eine Differenzierung zwischen der Fingerprintbarkeit von Neugeräten und angepassten Geräten war damit nicht möglich. Ebenso boten die verwendeten Testsysteme nicht von jedem Hersteller Geräte mit der neuesten Androidversion (Version 11 zum Zeitpunkt der Analyse) an. Dadurch wurde teilweise auf Geräte mit älteren Androidversionen zurückgegriffen.

5. Fazit

In der vorliegenden Studie wurde der Einfluss von Herstelleranpassungen auf die Gerätesicherheit von Smartphones analysiert. Dabei wurden mehrere wissenschaftliche Studien ausgewertet die Sicherheitslücken in den angepassten Android-Versionen und den vorinstallierten Diensten und Applikationen zeigten. Vor allem durch Anpassungen in sicherheitskritischen Diensten können Schwachstellen entstehen, welche Angreifern erlauben Daten auszulesen oder die Kontrolle über das Gerät zu übernehmen. Weiters wurde in der Studie die Auswirkung von Herstelleranpassungen auf die Fingerprintbarkeit von Smartphones untersucht. Dazu wurde ein Framework verwendet, welches automatisch Daten aus verschiedenen den Applikationen zur Verfügung stehenden Quellen ausliest. Dabei wurde festgestellt, dass vor allem zusätzlich zur Verfügung stehende Einstellungen und sich dadurch ergebende Implementationsunterschiede die Fingerprintbarkeit erhöhen können. Es wurden jedoch keine weiteren eindeutigen Identifikationsmerkmale oder anderwertige Informationslecks erkannt.

Referenzen

[1] R. Hay, „fastboot oem vuln: Android Bootloader Vulnerabilities in Vendor Customizations,“ *11th USENIX Workshop on Offensive Technologies,* p. 17, 2017.

[2] G. Palfinger, „Systematic Analysis of the Fingerprintability of Android,“ 6 4 2020. [Online]. Available: https://technology.a-sit.at/en/systematic-analysis-of-the-fingerprintability-of-android/.

[3] Google Inc, „Privacy in Android 10 | Android Developers,“ Google Inc, [Online]. Available: https://developer.android.com/about/versions/10/privacy. [Zugriff am 8 4 2021].

[4] L. Wu, M. Grace, Y. Zhou, C. Wu und X. Jiang, „The impact of vendor customizations on android security,“ *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications,* p. 623–634, 2013.

[5] A. Pereira, M. Correia und P. Brandão, „USB Connection Vulnerabilities on Android Smartphones: Default and Vendors' Customizations," *IFIP International Conference on Communications and Multimedia Security,* pp. 19-32, 2014.

[6] A. K. Iannillo, R. Natella, D. Cotroneo und C. Nita-Rotaru, „Chizpurfle: A Gray-Box Android Fuzzer for Vendor Service Customizations," *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE),* p. 11, 2017.

[7] „Firebase Test Lab," Google, [Online]. Available: https://firebase.google.com/docs/test-lab/. [Zugriff am 8 4 2021].

[8] „AWS Device Farm | Mobile & Web App Testing | Amazon Web Services," Amazon, [Online]. Available: https://aws.amazon.com/device-farm/. [Zugriff am 8 4 2021].

[9] „Android API Reference | Android Developers," Google, [Online]. Available: https://developer.android.com/reference. [Zugriff am 9 4 2021].

[10] „Content Providers | Android Developers," Google, [Online]. Available: https://developer.android.com/guide/topics/providers/content-providers. [Zugriff am 9 4 2021].

[11] P. Eckersley, „How unique is your web browser?," *International Symposium on Privacy Enhancing Technologies Symposium,* pp. 1-18, 2010.

[12] S. Kumar, L. J. Kittur und A. R. Pais, „Attacks on Android-Based Smartphones and Impact of Vendor Customization on Android OS Security," *International Conference on Information Systems Security,* p. 12, 2020.