— Masterthesis —

Analyse von Audio-Bypass-Bedrohungen auf Android-Smartphones und Konzeption von Abwehrmaßnahmen

Wolfgang Studier

Freie Universität Berlin

Fachbereich Mathematik und Informatik
Institut für Informatik

Betreuer:
Prof. Dr. Marian Margraf

24. Mai 2017

Abstract

Smartphones sind heutzutage ein alltäglicher Begleiter für die meisten Menschen im privaten als auch im beruflichen Umfeld. Dies sorgt dafür, dass Smartphones sich oft in unmittelbarer Nähe von Gesprächen befinden und damit zur Aufzeichnung von Gesprächen verwendet werden können, in denen eine große Vielfalt an Informationen mit verschiedenen Schutzbedürfnissen besprochen werden. In dieser Arbeit wird das Risiko für verschieden sensitive Informationen evaluiert, das von Audio-Bypass Angriffen auf Smartphones ausgeht.

Im Zuge der Analyse werden Audio-Bypass Bedrohungen basierend auf dem Baseband Subsystem sowie Android selbst behandelt. Hierzu werden die Systeme einzeln im Detail vorgestellt, um im späteren Verlauf die Bedrohung durch verschiedene Schwachstellen einschätzen zu können. Die anschließende Analyse der verschiedenen Angriffe, die zu einem Audio-Bypass führen könnten, wird mittels des Angriffspotentialmodells vorgenommen. Abschließend werden verschiedene Abwehrmaßnahmen gegen die möglichen Angriffe konzipiert und die aus den Angriffen entstehenden Risiken für verschieden sensitive Informationsklassen bewertet. Hierbei werden Empfehlungen zu Verhaltensweisen und zum Einsatz von Abwehrmaßnahmen, basierend auf dem Schutzbedürfnis der Informationen, ausgesprochen.

Inhaltsverzeichnis

Abkürzungsverzeichnis

OS Operating System

SP Smartphone

BB Baseband-Prozessor

AP Applikationsprozessor

3GPP 3rd Generation Partnership Project

GSM Global System for Mobile Communications

UMTS Universal Mobile Telecommunications System

LTE Long-Term Evolution

PDA Personal Digital Assistant

MMS Multimedia Messaging Service

SMS Short Message Service

MCU Memory Controller Unit

SoC System on Chip

ARM Advanced RISC Machine

GPU Graphics Processing Unit

MIPI Mobile Industry Processor Interface

DSP Digital Signal Processor

BSI Bundesamt für Sicherheit in der Informationstechnik

DFD Data Flow Diagrams

QC Qualcomm

AOSP Android Open Source Project

RTOS Real Time Operating System

AMMS Advanced Mode Subscriber Software

HLOS High Level Operation System

DMA Direct Memory Access

SMD Shared Memory Device

RCE Remote Code Execution

MITM Man-in-the-Middle

BTS Base Transceiver Station

RPM Resource Power Management

RCE Remote Code Execution

MPU Memory Protection Unit

DoS Denial of Service

NX Executable-Space Protection

ASLR Address Space Layout Randomization

IMEI International Mobile Equipment Identity

IMSI International Mobile Subscriber Identity

SLIMbus Serial Low-Power Inter-Chip Media Bus

QMI Qualcomm MSM Interface

RILD Radio Interface Layer Daemon

1 Einführung

Smartphones haben sich in den vergangenen Jahren zu einem integralen Teil unserer Kommunikationsinfrastruktur entwickelt. Sie begleiten uns im Alltag und bieten einen noch nie zuvor gesehenen mobilen Zugang zu Wissen und Kommunikation. Weltweit verwenden ca. 2,1 Milliarden Menschen jeden Tag Smartphones. Es ist davon auszugehen, dass diese Zahl auf 2,9 Milliarden im Jahr 2020 anwachsen wird [92].
Smartphones sind für viele Nutzer so sehr in das tägliche Geschehen eingebunden, dass ein Verzicht darauf wohl kaum in Frage kommt. Eine Studie des Branchenverbandes Bitkom im Jahre 2016 hat ergeben, dass 71% der befragten Deutschen nicht bereit wären, auf ein Smartphone zu verzichten [44].
In Hinsicht darauf, dass so viele Menschen Smartphones alltäglich einsetzen, ist es nicht verwunderlich, dass die Frage nach dem Schutz der Privatsphäre in den letzten Jahren, für die Anwender selbst aber auch die Hersteller, immer mehr in den Vordergrund getreten ist. Diese Entwicklung wurde durch viele datenschutz- und sicherheitsrelevante Vorfälle der letzten Jahre weiter vorangetrieben. Eine Studie der University of California in Berkeley zeigt [51], dass die meisten Smartphonenutzer mehr um den Schutz ihrer Privatsphäre und die Sicherheit ihrer Daten im Zusammenhang mit Smartphonenutzung besorgt sind, als bei der Nutzung eines Desktop-PCs. Die Studie zeigt auch, dass die meisten Smartphoneanwender aufgrund von fehlendem technischen Wissen und/oder dem fehlenden Verständnis der Zusammenhänge nicht in der Lage sind, die Sicherheit eines Smartphones selbst, oder einzelner auf dem Smartphone ausgeführter Applikationen einzuschätzen.

1.1 Audio-Bypass

1.1.1 Definition

Bei einem Audio-Bypass Angriff auf ein Smartphone (SP) wird versucht, ohne die Zustimmung des Benutzers auf die von dem Mikrofon des Smartphones aufgenommenen Geräusche zuzugreifen. Dabei wird angestrebt, dass der eigentliche Nutzer des Smartphones keine Anzeichen dieser Datenexfiltration wahrnimmt. Sollte der Nutzer eine Aktion durchführen müssen, um den Zugriff auf das Mikrofon zu ermöglichen, ist es für einen Angreifer erstrebenswert, diese Interaktion zu verschleiern.

Für die praktische Durchführung eines solchen Angriffs können verschiedene Wege gewählt werden, um Zugriff auf die vom Mikrofon aufgenommenen Audiodaten zu erhalten. Neben einem direkten Angriff auf die Infrastruktur, die diese Daten verarbeitet, ist auch die komplette Übernahme des Systems möglich. Daher ist ein Angriff auf das Smartphonesystem, der dem Angreifer die Kontrolle über das Smartphone ermöglicht, aus technischer Sicht einem reinen Audio-Bypass Angriff überlegen. Weiter noch ist es nahezu unmöglich, die sensitiven Audiodaten in der Umgebung des SP zu schützen, sollte der Angreifer bereits Kontrolle über das System haben.
Es ist darauf hinzuweisen, dass auch sogenannte "Side-Channel"-Angriffe existieren bei denen versucht wird, z.B. auf Basis von Vibrationen, die der gyroskopische Sensor des Smartphones aufgreifen kann, die Umgebungsgeräusche zu extrapolieren [86].

1.1.2 Anwendungsgebiete

Der Audio-Bypass Angriff ist in vielen Situationen von Interesse und kann auch von verschiedenen Aktoren durchgeführt werden. Die technischen Voraussetzungen für die verschiedenen Angriffe sind heutzutage nicht mehr nur durch staatliche Institutionen zu erreichen, sondern liegen durchaus auch im Möglichkeitsrahmen einer einzelnen technisch versierten Person [123].
Wie in der Einführung erörtert, hat sich das Smartphone für viele Anwender zu einem alltäglichen Begleiter entwickelt, der im privaten als auch professionellen Umfeld intensiv eingesetzt wird. Gerade durch die stetige Präsenz des Gerätes in der Nähe des Anwenders eignet es sich hervorragend für die Audioüberwachung des Anwenders selbst, aber auch aller Personen die sich in Aufnahmereichweite des Mikrofons des Smartphones befinden.
Besonders deutlich wird die Gefahr durch einen Audio-Bypass vor allem bei Personen, deren Beruf ein gewisses Maß an Verschwiegenheit und Datenschutz erfordert, wie z.B. Ärzten, Anwälten, Polizisten, im Management und vielen weiteren Berufsgruppen.
Noch bedrohlicher wird das Szenario wenn man bedenkt, wie viele Politiker, Staatsführer oder Militärs auf die Verwendung von Smartphones in ihrem alltäglichen Geschäft angewiesen sind. Es wird zwar versucht, das Risiko gerade bei diesen Personengruppen durch den Einsatz von speziellen Geräten zu minimieren, aber diese Maßnahmen werden unterminiert durch die Tatsache, dass schon ein privates Smartphone, das in der Tasche eines der Gesprächsteilnehmer vergessen wurde, zur kompletten Kompromittierung des Gesprächs führen kann [78]. Aber auch ein Angriff auf speziell gehärtete Systeme ist möglich, vor allem, wenn diese nicht ordnungsgemäß betrieben oder gewartet werden.

1.2 Smartphones

Seit der Einführung von Mobiltelefonen in den Endnutzermarkt, Anfang 1990, hat sich die Hardwarearchitektur von mobilen Endgeräten stark verändert. Neben der Einführung neuer Kommunikationsstandards für Mobilkommunikation durch die 3rd Generation Partnership Project (3GPP) Arbeitsgruppe haben auch die Hersteller von Mobiltelefonkomponenten ihre Produktionsmethoden verfeinert sowie die Hardware- als auch die Softwarearchitektur (Firmware und Treiber) weiterentwickelt und verbessert [125].
Trotz der Tatsache, dass die 3GPP Standards öffentlich zugänglich sind, ist das Wissen über die Architektur und Implementierung der in den Dokumenten beschriebenen Netze und Protokolle nur einem sehr kleinen Kreis von System- und Softwarearchitekten bekannt [125]. Dies geht Hand in Hand mit der Tatsache, dass die gesamte Industrie im Bereich Mobilkommunikationsgeräte (Hersteller von Mobiltelefonen sowie deren Zulieferer) ein sehr geschlossenes System bildet, das durch Geheimhaltung genauer technischer Spezifikationen (nur erhältlich im Rahmen eines Vertrags, mit non-disclosure agreement) versucht, die technischen Innovationen der einzelnen Firmen zu schützen. Besonders im Bereich der Baseband-Prozessoren (BB), auf die wir in Kapitel 4 genauer eingehen werden, ist diese Tendenz besonders auffällig.

1.2.1 Kommunikationsstandards

Im Bereich der Mobilkommunikation für Endnutzer sind die Standards 2G (GSM), 3G (UMTS) und 4G (LTE) der 3GPP die am meisten verbreiteten (1G steht für die früher verwendeten analogen Netze, wurde aber nicht von 3GPP standardisiert). Hierbei ist darauf hinzuweisen, dass neben der 3GPP auch noch die 3GPP2 existiert, die Standards für eine andere Trägertechnik entwickelt, die aber hauptsächlich in Nordamerika und Südkorea eingesetzt werden. Die Betrachtungen in dieser Arbeit beziehen sich auf Geräte, die den in Europa üblichen 3GPP Standard umsetzten.
Die 3GPP ist eine weltweite Kooperation von Organisationen, die sich aus verschiedenen Gruppen zusammensetzt. Die beiden Hauptgruppen sind hierbei die "Organizational Partners", verschiedene internationale Normungsinstitute (z.B. ETSI (European Telecommunication Standards Institute) und ATIS (Alliance for Telecommunications Industry Solutions, USA)) sowie die "Market Representation Partners", die sich aus Vertretern der Mobilkommunikationsindustrie zusammensetzen [1, 2]. Die Standards werden in einem iterativen Dreimonatszyklus von den verschiedenen Arbeitsgruppen der 3GPP erarbeitet und veröffentlicht.
Die Entwicklung der letzten Jahre zeigt eine eindeutige Tendenz hin zu 3G und 4G Technologien, die durch neue Techniken dem 2G Standard hinsichtlich des Datendurchsatzes und der Verbindungsqualität klar überlegen sind. Um dem

weltweit steigenden Bedarf an mobiler Datenkommunikation gerecht werden zu können, ziehen die Netzbetreiber in Erwägung, Sendefrequenzbereiche, die momentan noch mit 2G betrieben werden, in den nächsten Jahren auf 3G oder 4G umzustellen [117].
Beispielsweise wurde zum Ende des Jahres 2016 das 2G Netzwerk des australischen Netzbetreibers Telstra komplett abgeschaltet. Da Telstra der einzige große Infrastrukturbetreiber in Australien ist, bedeutet das effektiv das Ende des australischen 2G Netzwerks [116]. Genaue Pläne über die Zukunft des 2G-Netzwerks in Europa sind, abgesehen von dem Schweizer 2G Netz der Swisscom, das ab 2020 abgeschaltet werden soll, noch nicht genauer bekannt [70].
Aus dieser Tendenz ist abzusehen, dass Modems für Mobiltelefone, die heutzutage aus Kompatibilitätsgründen noch den 2G Standard unterstützen, in Zukunft ohne die Unterstützung des 2G Standards entwickelt werden könnten.

1.2.2 Feature- und Smartphones

Zu Beginn der Mobiltelefonentwicklung handelte es sich bei Mobiltelefonen um bloße Telefone mit wenig weiteren Funktionen. Neben der Fähigkeit, Anrufe abzusetzen und entgegenzunehmen sowie Kurznachrichten (SMS) zu versenden und zu empfangen, gab es nur wenige weitere Funktionen wie z.B. Taschenrechner, Wecker oder ein Telefonbuch. Diese Art der Mobiltelefone werden Featurephones genannt.
Im Zuge der Entwicklung der Neunzigerjahre entstand ein weiteres neuartiges Gerät, der Personal Digital Assistant (PDA). Diese Geräte mit ihren verhältnismäßig großen Touchdisplays und der Möglichkeit, die auf dem System installierte Software den persönlichen Bedürfnisse anzupassen, wurden schnell sehr beliebt und fanden eine weite Verbreitung unter Endanwendern. Im weiteren Entwicklungsprozess der mobilen Endgeräte wurden zur Jahrtausendwende die typischen PDA Funktionalitäten in Mobiltelefone integriert. Dies prägte den heute verwendeten Namen Smartphone für diese Geräte [125].
Neben den Unterschieden im Bereich des Formfaktors und des Funktionsumfangs, liegt einer der größten Unterschiede zwischen Feature- und Smartphones in der verwendeten Hardwarearchitektur. Featurephones beinhalten nur einen einzelnen Prozessor, der nicht nur das User-Interface (Tastatureingaben, Bildschirmausgaben) sondern auch einen 3GPP konformen Protokollstack (2G/3G/4G) zur Kommunikation mit dem Mobilfunknetz implementiert. Dieser Prozessor (sozusagen das Modem des Mobiltelefons) wird als Baseband-Prozessor (BB) bezeichnet. Da der Baseband-Prozessor mit einem getakteten Mobilnetz kommunizieren muss, das heißt, Anfragen müssen innerhalb eines strikten Zeitfensters beantwortet werden, gelten weiche Echtzeitanforderungen an die auf ihm laufenden Systeme, um mit den 3GPP Netzen kommunizieren zu können [89, 123].

Bedingt dadurch, dass Smartphones es dem Nutzer ermöglichen, eigenmächtig Apps zu installieren, musste eine neue Hardwarearchitektur verwendet werden. Dies soll sicherstellen, dass die Ausführung von, zum Teil hochkomplexen, Apps die Echtzeitfähigkeit des BBs nicht beeinträchtigt. Die Lösung hierfür ist die Auftrennung von nutzerseitigen und netzwerkseitigen Funktionen auf zwei verschiedene Prozessoren. Der sogenannte Applikationsprozessor (AP) übernimmt das User-Interface sowie die Ausführung des OS (Android, iOS, etc.), wobei der BB nur die Kommunikation mit dem Mobilfunknetz sowie, abhängig vom Modell des BB, auch die Kommunikation mit anderen drahtlosen Netzwerken (u.a. WLAN, Bluetooth) übernimmt [125].

1.2.3 Hardwarearchitektur

Durch die Anforderungen an den Smartphonemarkt, immer kleinere und leistungsfähigere Smartphones zu entwickeln, hat sich die sogenannte System on Chip (SoC) Architektur durchgesetzt. Hierbei handelt es sich um ein hochintegriertes System auf einem einzelnen Chip, das alle Komponenten, die für den Betrieb benötigt werden, beinhaltet.
Die hier dargestellte Hardwarearchitektur ist hauptsächlich in Android Smartphones zu finden. Apple setzt zwar ebenfalls auf einen integrierten SoC Ansatz, verwendet jedoch einen nicht integrierten BB.
Bei heutzutage typischerweise eingesetzten SoC handelt es sich um sogenannte Multi-Core-Systeme. Diese Systeme beinhalten in der Regel nicht nur mehrere Prozessoren (AP, GPU, BB), sondern auch mehrere Kerne pro Prozessor. In der Regel werden Advanced RISC Machine (ARM) Prozessoren für den AP und Digital Signal Processors (DSP) oder ARM Prozessoren für den BB verwendet [57]. Heutzutage setzten sich die APs meist aus verschieden schnell getakteten Kernen zusammen [18]. Dies ermöglicht es, Aufgaben mit geringer Komplexität besonders energieeffizient berechnen zu lassen, während das System ebenfalls in der Lage ist, komplexere Probleme in einer adäquaten Zeit zu lösen, aber, bedingt durch den Einsatz der höher getakteten Kerne, bei wesentlich höherem Energieverbrauch.
Die einzelnen Komponenten sind innerhalb des Chips direkt miteinander verbunden und greifen auch gemeinsam auf eine Speicherinfrastruktur zurück. Der RAM des Systems wird physikalisch oberhalb des SoC platziert und über eine gemeinsame Memory Controller Unit (MCU) angesprochen [57]. Diese Designentscheidung wird in der Risikoanalyse des BBs in Kapitel 4.2 genauer analysiert. Neben den bereits erwähnten Prozessoren beinhalten die meisten für Smartphones konzipierten SoCs weitere Komponenten. Es werden DSPs zur Verarbeitung von Audio- und Videosignalen integriert sowie verschiedene weitere Kommunikationsmodule wie Bluetooth, WLAN oder NFC (inklusive der dazugehörigen analogen Signalkonverter). Auf den Teil des SoCs der den BB bildet,

wird in Kapitel 4.1.1 detaillierter eingegangen.
Die Vorteile von SoCs liegen vor allem im Bereich der Energieeffizienz und Baugröße. Durch den hohen Grad an Integration der einzelnen Systemkomponenten kann sehr viel Platz gespart werden der es ermöglicht, größere Batterien oder zusätzliche Funktionsbausteine zu verbauen. Durch die extrem geringen physikalischen Abstände zwischen den einzelnen Komponenten innerhalb eines SoCs entstehen weniger Verzögerungen bei der Übertragung von Daten über das interne Bussystem.
Die gängigsten verwendeten Interfaces zur Befehls- und Datenübertragung innerhalb von Smartphones sind serielle Schnittstellen (vor allem in älteren Modellen), Serial Peripheral Interface Bus (SPI), USB 2.0 (oder der äquivalente Standard High-Speed Inter-Chip (HSIC), der die USB 2.0 Standards ohne steckbare Verbindungen für eingebettete Systeme darstellt) sowie die verschiedenen durch die Mobile Industry Processor Interface (MIPI) Allianz entwickelten Schnittstellen [57].
Diese durch die MIPI Allianz entwickelten Schnittstellen haben in den vergangenen Jahren immer mehr an Bedeutung gewonnen und sind heute der De-facto-Standard in vielen Bereichen. Zu den meist verwendeten Schnittstellen, die durch die MIPI Allianz standardisiert wurden, gehört das Camera Serial Interface (CSI), das Display Serial Interface (DSI), das Low Latency Interface (LLI), welches Anwendung bei der Anbindung verschiedener Prozessoren an den RAM innerhalb eines SoCs findet, sowie der Serial Low-Power Inter-Chip Media Bus (SLIMbus), der in Kapitel 4.1.3 ausführlich behandelt wird.

2 Methodik

Zur Bewertung von Bedrohungen und den daraus entstehenden Risiken werden in dieser Arbeit sogenannte Risikomodellierungssysteme verwendet. Diese versuchen durch die Verwendung verschiedener Metriken zur Abschätzung einer Bedrohung das Risiko zu bewerten, dem ein System ausgesetzt ist.
Eine klassische Risikoanalyse setzt sich mit der Identifikation gewisser Gefährdungen in Bezug auf Sicherheitsziele und deren Eintrittswahrscheinlichkeit auseinander. Dies setzt die Fähigkeit zur Abschätzung und Identifikation von Bedrohungen voraus. Da es sich bei der Audio-Bypass Bedrohung um eine einzelne spezifische Bedrohung handelt, werden wir eine Teilmenge der Risikomodellierungssysteme verwenden. Für die Betrachtungen in dieser Arbeit sind vor allem die Bedrohungs- und Schwachstellenanalyse innerhalb der vorgestellten Modelle von Interesse.

2.1 Terminologie

Im Bereich der Risiko- und Bedrohungsanalyse wird eine besondere Terminologie verwendet, die von dem ISO-Standard ISO27000 [76] definiert und hier in Kürze vorgestellt. Die Grafik 2.1 verdeutlicht den Zusammenhang zwischen den erklärten Begriffen.

Risiko (risk): Das Risiko wird oft als das Produkt aus der Eintrittswahrscheinlichkeit eines Vorfalls sowie dessen potentiellen Schadens definiert. Genauer drückt das Risiko die Wahrscheinlichkeit aus, dass eine Schwachstelle durch eine Bedrohung angegriffen wird und den daraus entstehenden Schaden.
Beispiel: Es wird eine Risikoanalyse des Audio-Bypasses durchgeführt, um das Risiko dieses Angriffs auf verschlüsselte Kommunikationen im Umfeld des Smartphones (SPs) zu bestimmen.

Asset (asset): Das zu schützende Objekt. Bei diesem Objekt kann es sich um ein System, einen Vorgang oder eine Eigenschaft handeln.
Beispiel: Audiodaten von Gesprächen, die in unmittelbarer Nähe des SP stattfinden.

Sicherheitsziel (security goal): Sicherheitsbedenken in Bezug auf ein Asset. Welche sicherheitskritischen Eigenschaften des Assets müssen geschützt wer-

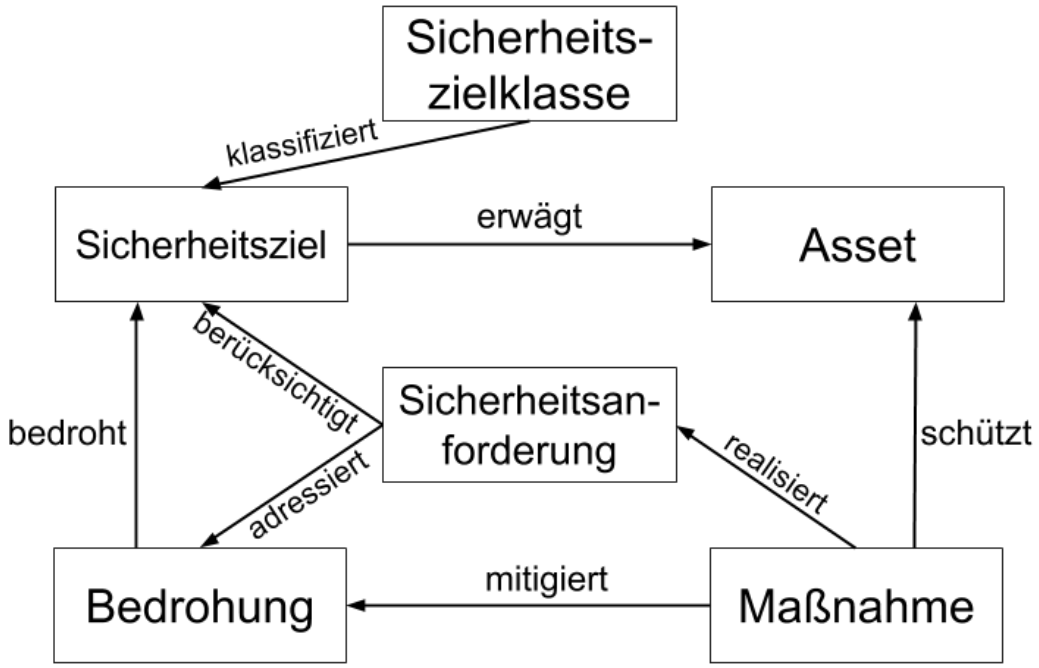

Abbildung 2.1: Zusammenhänge zwischen den verwendeten Begriffen

den?
Beispiel: Vertraulichkeit von Gesprächen, die in unmittelbarer nähe eines SP geführt werden.

Sicherheitszielklasse (security goal class): Übergeordnete Klassifikation von Sicherheitszielen.
Beispiel: Die klassischen Schutzziele der IT-Sicherheit: Vertraulichkeit, Verfügbarkeit und Integrität.

Bedrohung oder Gefahr (threat): Die mögliche Ursache eines ungewollten und sicherheitskritischen Vorgangs, der zu Schaden am System oder zur Kompromittierung der Sicherheitsziele führen kann.
Beispiel: Unautorisierter Zugriff des Baseband-Prozessors (BBs) auf das Mikrofon.

Schwachstelle (vulnerability): Schwachstelle eines Assets, die durch eine Bedrohung ausgenutzt werden kann.
Beispiel: Der BB kann den Mediabus routen, um auf das Mikrofon zuzugreifen.

Sicherheitsanforderungen (security requirement): Anforderungen an Schutzmaßnahmen basierend auf einem oder mehreren Sicherheitszielen.

Beispiel: Es darf dem BB nicht möglich sein den Mediabus zu routen, ohne dass der Nutzer davon in Kenntnis gesetzt wird.

Maßnahmen (control): Gezielte Maßnahmen, um eine Sicherheitsanforderung umzusetzen sowie die Kontrolle der Effektivität der Maßnahme.
Beispiel: Entfernen des Mikrofons aus dem SP. Entfernen der angreifbaren Komponente ist eine extreme, aber eindeutig effektive Mitigation der Schwachstelle.

2.2 Schwachstellen- und Bedrohungsanalysemodelle

Zur Bewertung der gefundenen Schwachstellen soll ein Modellierungssystem verwendet werden. Dies bietet nicht nur Vorteile im Bereich der Methodik durch schon bereits erarbeitete Prozesse und Vorgänge zur Analyse von Bedrohung und Schwachstellen, sondern auch im Bereich der qualitativen Analyse. Qualitative Modelle definieren ein numerisches Bewertungsverfahren zur Einordnung der Schwachstellen und Bedrohungen in unterschiedliche Risikogruppen (z.B. Niedrig (1-3), Mittel (4-6), Hoch (7-10)), wohingegen quantitative Modelle meist auf Basis von finanziellem Schaden bewerten. Beides ermöglicht, Gefährdungen über ihre numerischen Ordnung zu vergleichen und sie anhand dieser zu kategorisieren [87, 110].
Der numerische, qualitative Analyseansatz kann aber auch zu Problemen führen. Oft werden zwei unabhängige Personen, die die gleiche Bedrohung analysieren, sehr unterschiedliche Bewertungen für das entsprechende Risiko vergeben (z.B. Unterschied zwischen der Sicht eines Entwicklers und der eines Sicherheitsanalysten). Hierbei hilft der Einsatz einer weniger fein skalierten Messmethode, dies führt zu einer gröberen, aber dafür auch konsistenten, Einordnung von Bedrohungen in die entsprechenden Risikogruppen (z.B. Niedrig (1), Mittel (2), Hoch (3)) [87].
Im Folgenden werden drei verschiedene Modellierungssysteme vorgestellt und evaluiert, ob eine Anwendung auf die Analyse der Audio-Bypass Bedrohung in Android SP möglich und sinnvoll ist.

2.2.1 BSI - IT-Grundschutz Risikoanalyse

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) bietet den sogenannten IT-Grundschutz Standard. Der IT-Grundschutz ist in dem BSI-Standard für Informationssicherheit definiert und wird durch den IT-Grundschutz Katalog erweitert. Der BSI-Standard umfasst vier Teile (BSI-Standard 100-1 bis 100-4 [46]), die den gesamten Bereich des Informationssicherheitsmanagements

abdecken. Hierbei wird für Unternehmen und Behörden eine ganzheitliche Herangehensweise an den Themenkomplex Informationssicherheit geboten. Die verschiedenen Teile des Standards ermöglichen es, das Risiko eines gesamten Komplexes (z.B. eine Firmenniederlassung) zu ermitteln. Hierbei wird nicht nur auf die IT-Systeme geachtet, sondern auch auf Fragen wie räumliche Zugangskontrolle, Identitätsmanagement, Ausfälle durch höhere Gewalt und physikalische Sicherheit (z.B. Absicherung von Serverräumen oder Aktenschränken) sowie viele weitere Themengebiete.
Im IT-Grundschutz Katalog [49] findet sich der Gefährdungs- und Maßnahmenkatalog des BSI. Der Gefährdungskatalog definiert und beschreibt verschiedene generische Gefährdungen und teilt diese in sechs Gruppen (Go Elementare Gefährdungen bis G5 Vorsätzliche Handlungen) auf. Der Maßnahmenkatalog definiert und beschreibt Gegenmaßnahmen für die spezifischen Gefährdungen aus dem Gefährdungskatalog. Diese werden ebenfalls in sechs Gruppen aufgeteilt (M1 Infrastruktur bis M6 Notfallvorsorge).
Aus diesen Gefährdungen und Maßnahmen werden vom BSI fertig erarbeitete Bausteine für verschiedene Themenkomplexe angeboten. Diese Bausteine sind in fünf Gruppen sortiert (B1 Übergreifende Aspekte bis B5 Anwendungen) und bieten einer Organisation die Möglichkeit, das Risiko von weit verbreiteten Themenkomplexen (z.B. SPs, Büroräume, Druckerverwaltung, Microsoft Windows Sever 2003 und viele weitere) einschätzen zu können, ohne eine eigene Risikoanalyse durchführen zu müssen.
Der für diese Arbeit relevante Teil befindet sich in dem BSI-Standard 100-3: Risikoanalyse auf Basis von IT-Grundschutz [48]. Dieser Standard soll es ermöglichen, eine vollständige Risikoanalyse eines Komplexes durchzuführen für den Fall, dass das zu prüfende System nicht oder nicht ausreichend im IT-Grundschutz Katalog behandelt wurde oder es einen sehr hohen Schutzbedarf gibt, der nicht oder nur unzureichend in dem spezifischen Baustein des IT-Grundschutz Katalogs behandelt wird.
Das empfohlene Vorgehen zur Risikoanalyse beinhaltet die Identifizierung der zu schützenden Systeme sowie der Werte (Vertraulichkeit, Integrität und Verfügbarkeit), die es innerhalb dieses Systems zu schützen gilt. Hierbei wird die Bedrohung zusammen mit der Schwachstelle zu einer Gefährdung zusammengefasst. Dies ermöglicht es, den Prozess der Risikoerkennung einer Gefährdung zu beschleunigen und mehr auf die Bedürfnisse eines durchschnittlichen Unternehmens anzupassen. Auch wird kein numerisches Verfahren gewählt, um das Risiko auszudrücken, sondern es werden Schutzmaßnahmen mit Gefährdungen verknüpft und dokumentiert, ob diese die Gefährdung ausreichend mitigieren.

Bei der genauen Betrachtung der Audio-Bypass gibt es allerdings nur eine Bedrohung und mehrere mögliche Schwachstellen. Das vom BSI verwendete Modell erlaubt es aufgrund seiner Verallgemeinerung im Bereich der Begriffe

Schwachstelle und Bedrohung leider nicht, eine genaue Modellierung dieser Zusammenhänge durchzuführen.

2.2.2 Microsoft - STRIDE

Bei dem von Microsoft entwickelten Verfahren STRIDE (**s**poofing, **t**ampering, **r**epudiation, **i**nformation disclosure, **d**enial of service, **e**levation of privilege) handelt es sich um ein Bedrohungsmodellierungssystem. Es wurde als Nachfolger des DREAD Modellierungssystems [87] entwickelt und ist Teil des Microsoft Software Development Process (SDL) [110].
Das Vorgehen nach STRIDE ist in vier Phasen aufgeteilt [88]. Der erste Schritt ist die Erstellung eines Data Flow Diagrams (DFD) des zu prüfenden Systems (eine Legende befindet sich in Anhang 9.1.1 ein Beispiel findet sich in Kapitel 4.2). Hierbei werden alle am System beteiligten Komponenten sowie der Datenfluss zwischen ihnen dokumentiert. STRIDE erweitert DFDs um sogenannte Vertrauensgrenzen die verdeutlichen sollen, wann Daten zwischen sich nicht vertrauenden Prozessen ausgetauscht oder von außerhalb in das System eingebracht werden. Dies ist von besonderem Interesse, da Daten, denen nicht vertraut wird bis sie validiert wurden, als möglicherweise schädlich angesehen werden müssen. Als eine Alternative zu DFDs bietet sich die Modellierung in der Unified Modeling Language (UML) an, wobei auch hier die Einführung von Vertrauensgrenzen unerlässlich für die Verwendung in STRIDE ist.
Der zweite Schritt ist die Identifizierung von Bedrohungen basierend auf den im DFD aufgezeigten Datenverbindungen, die Vertrauensgrenzen überschreiten. Hierbei wird das sogenannte STRIDE-per-element verwendet, das bedeutet, jede Verbindung, die Vertrauensgrenzen überschreitet, wird gegen den Satz von STRIDE-Bedrohungen evaluiert [110].
Schritt drei ist die Konzeption von Schutz- sowie Abwehrmaßnahmen basierend auf der Bedrohung der einzelnen Datenverbindungen, aufgezeigt durch die STRIDE-per-element Methode. Anschließend werden diese Maßnahmen während Schritt vier auf ihre Wirksamkeit zur Abwehr der identifizierten Bedrohungen überprüft [110].

Wie auch bei dem IT-Grundschutz, so wird auch bei STRIDE eine Vielzahl verschiedener Bedrohungen betrachtet, um anschließend Abwehrmaßnahmen zu konzeptionieren und diese zu verifizieren. Hierbei wird besonders darauf geachtet, möglichst alle Bedrohungen die sich im System ergeben, zu erfassen. Leider gibt es auch bei STRIDE kein weiterführendes Modell, um auf den Komplex einer einzelnen Bedrohung und der dafür relevanten Schwachstellen genauer einzugehen. Allerdings bietet es mit DFDs (und deren Erweiterung durch Vertrauensgrenzen) ein geeignetes Werkzeug [88], um die möglichen Bedrohungen und Schwachstellen innerhalb eines Systems zu identifizieren.

2.2.3 ISO/IEC 18045 - Angriffspotential

Angriffspotential (attack potential) ist ein Verfahren aus dem ISO/IEC 18045 Standard [74, siehe Annex B.4] und gehört zu dem gängigen Vorgehen während einer Common Criteria[1] (CC) Zertifizierung. Im Gegensatz zu den bisher vorgestellten Modellen handelt es sich bei dem Angriffspotential nicht um einen Asset-, sondern um einen Angreifer-zentrierten Ansatz.
Hierbei wird nicht das allgemeine Risiko einer Bedrohung bewertet, sondern der minimale Aufwand, der von einem Angreifer erbracht werden muss, um eine Schwachstelle auszunutzen, die zu einer Verletzung der definierten Sicherheitsziele führt. Das Risiko wird aus dem Schutzbedarf des zu evaluierenden Sicherheitsziels und dem Potential, das für einen erfolgreichen Angriff benötigt wird, anhand einer Risikomatrix ermittelt. Der Schutzbedarf eines Sicherheitsziels ergibt sich aus den zu erwartenden Konsequenzen, sollte es zu einer Verletzung der für ein Asset definierten Sicherheitsziele kommen.
Zur Berechnung des Angriffspotentials werden die benötigten Ressourcen für einen Angriff auf eine Schwachstelle in fünf Kategorien zerlegt und jeweils für die beiden Faktoren Identifikation und praktische Ausnutzung einer Schwachstelle bewertet. Die benötigten Ressourcen werden in folgende Kategorien zerlegt: Benötigte Zeit, Expertise des Angreifers, benötigtes Equipment, Wissen über das Angriffsziel und Zugang zum Angriffsziel. Die Bewertung dieser einzelnen Punkte findet anhand einer allgemeinen oder auf den Bereich der Bedohungssanalyse angepassten Tabelle (siehe Anhang 9.2) statt. Jeder Punkt wird numerisch für die beiden Faktoren bewertet, anschließend werden die Punkte der beiden Faktoren aufaddiert. Diese Summe ergibt das endgültige Angriffspotential. Die Tabelle 2.1 zeigt die Einordnung des Angriffspotentials in eine von vier Gruppen, die anschließend zur Einordnung des Risikos mit Tabelle 2.2 verglichen werden können.

Wert	Angriffspotentialgruppe
< 10	Keine Wertung
10 - 17	Niedrig
18 - 24	Mittel
> 24	Hoch

Tabelle 2.1: Angriffspotentialgruppen

Der Angreifer-zentrierte Ansatz ermöglicht dem Angriffspotentialmodell, die Beziehungen zwischen einzelnen Bedrohungen und die durch sie ausgenutzten Schwachstellen sehr genau zu modellieren. Durch die Berücksichtigung des

[1] Common Criteria ist ein internationaler ISO-Standard [75] zur Prüfung und Bewertung von sicherheitskritischen IT-Systemen.

Angriffspotential	Risiko		
Keine Wertung	Unerwünscht	Inakzeptabel	Inakzeptabel
Niedrig	Tolerierbar	Unerwünscht	Inakzeptabel
Mittel	Tolerierbar	Unerwünscht	Inakzeptabel
Hoch	Vernachlässigbar	Tolerierbar	Unerwünscht
Schutzbedürfnis	Normal	Hoch	sehr Hoch

Tabelle 2.2: Risikomatrix zur Einordnung der Bedrohung

benötigten Aufwands wird es möglich, eine realistische Einschätzung des Risikos, dem verschiedene Gruppen durch eine Bedrohung ausgesetzt sind, vorzunehmen. Die Kategorisierung der benötigten Ressourcen erlaubt einen Vergleich mit schon bekannten Angriffen und den für sie benötigten Aufwand. Durch die sehr ausgeprägte Modularisierung der ISO Standards, die innerhalb des CC angewendet werden, ist es ohne weiteren Aufwand möglich, das Verfahren aus seinem Umfeld zu lösen und ohne weitere CC Terminologie oder Verfahren zu verwenden.

2.3 Verwendete Methodik

Da es sich bei der Arbeit um die Betrachtung einer einzelnen Bedrohung handelt wird davon ausgegangen, dass die Modellierung mittels des Angriffspotentials die genaueste Einsicht in die Beziehung zwischen Audio-Bypass-Bedrohung und der Schwachstellen, die durch sie bedroht werden, bietet (siehe Kapitel 2.2.3). Zur Identifikation der möglichen Schwachstellen wird ein DFD mit der Erweiterung um Vertrauensgrenzen, nach dem Vorbild des STRIDE-Modells, verwendet. Für die Einschätzung des Angriffspotentials werden auf den Bereich passende Tabellen definiert, die mit einer Erklärung der Werte im Anhang 9.2 zu finden sind. Für die betrachteten Schwachstellen wird das Angriffspotential berechnet und mögliche Modifikatoren genannt. Diese Modifikatoren können sich auf bekannte Besonderheiten beziehen oder auf deutliche Unterschiede innerhalb des Angriffs, bedingt durch die Art des Angreifers.
Für die Einordnung der Sicherheitsbedürfnisse von Sicherheitszielen wird die Definition des BSI verwendet, die in Kapitel 7 vorgestellt wird.
Bei der Konzeption der Gegenmaßnahmen wird überprüft, inwieweit der Einsatz einer Maßnahme die Benutzbarkeit des SP beeinträchtigt und in welchem Umfang die Wirksamkeit von dem bewussten Einsatz der Abwehrmaßnahme durch den Benutzer abhängt (siehe Kapitel 6).

3 Verwendete Plattform

3.1 Testplattform: Nexus 5x

Die in dieser Arbeit beschriebenen Architekturen und Vorgänge wurden auf Basis eines Nexus 5x von Google (Produziert durch LG) recherchiert und, falls möglich, überprüft. Die Entscheidung für das Nexus 5x wurden aus mehreren Gründen getroffen:

- Das im Nexus 5x eingesetzte System on Chip (SoC) MSM8992 von Qualcomm (QC) stammt aus der Snapdragon 808 Reihe, der Aufbau der 800er Reihe ist aus Sicht der Hardwarearchitektur sehr ähnlich (dies beinhaltet Snapdragon 805, 808, 810, 820 und 821) und stellt als die neueste Modellreihe von QC die höchste Anzahl an verbauten Elementen in modernen Smartphones. Es ist darauf hinzuweisen, dass die neueste Generation der 800er Reihe, der Snapdragon 835, eine neue Hardwarearchitektur aufweist.

- Wie aus der Marktanteilübersicht in Kapitel 3.2 zu erkennen ist, beherrscht QC den Markt mit annähernd 50% und ist damit der am stärksten vertretene Produzent im SoC Markt für Android Smartphones (SPs).

- Das QC SoC und die dazugehörigen, meist proprietären, Techniken sind, auf Grund der weiten Verbreitung von QC SoCs, häufig gewählte Ziele von wissenschaftlichen Arbeiten oder Penetrationstests im Bereich der Smartphonesicherheit. Durch diese Forschung ist wesentlich mehr über die internen Vorgänge von QC SoCs bekannt, als über die Produkte ihrer Mitbewerber.

- Der von QC und dem Android Open Source Project (AOSP) entwickelte Android Bootloader ist Grundlage für die Bootloader vieler SP Produzenten (Samsung, HTC, LG uva).

- Das Nexus 5x ist ein verhältnismäßig modernes SP (veröffentlicht September 2015).
  - Aktuellste Version von Android (7.1.2) ist von Google erhältlich.
  - Es erhält aktuelle Security Patches (Patch-Stand: 05/2017).

- Es war ein Nexus 5x für Testzwecke verfügbar.

3.2 Marktanteile

Der Markt für SoCs, die bei Androidgeräten meistens den Baseband-Prozessor (BB) beinhalten, wird hauptsächlich von einigen großen SoC Herstellern dominiert, wie die Zahlen des Android-Marktanalyse und -Mobilbenchmark Unternehmens Antutu aufzeigen [42]. Diese Zahlen beziehen sich auf die erste Hälfte 2016 und zeigen daher einige der aktuelleren Vorgänge nicht auf. Besonders anzumerken ist, dass die marktbeherrschende Stellung von QC zunehmend angegriffen wird und einige große Abnehmer (Apple, Samsung) damit begonnen haben, Alternativen für ihre SP zu testen. Das Samsung Galaxy S7 z.B. setzt teilweise auf ein von Samsung selbst entwickeltes SoC.

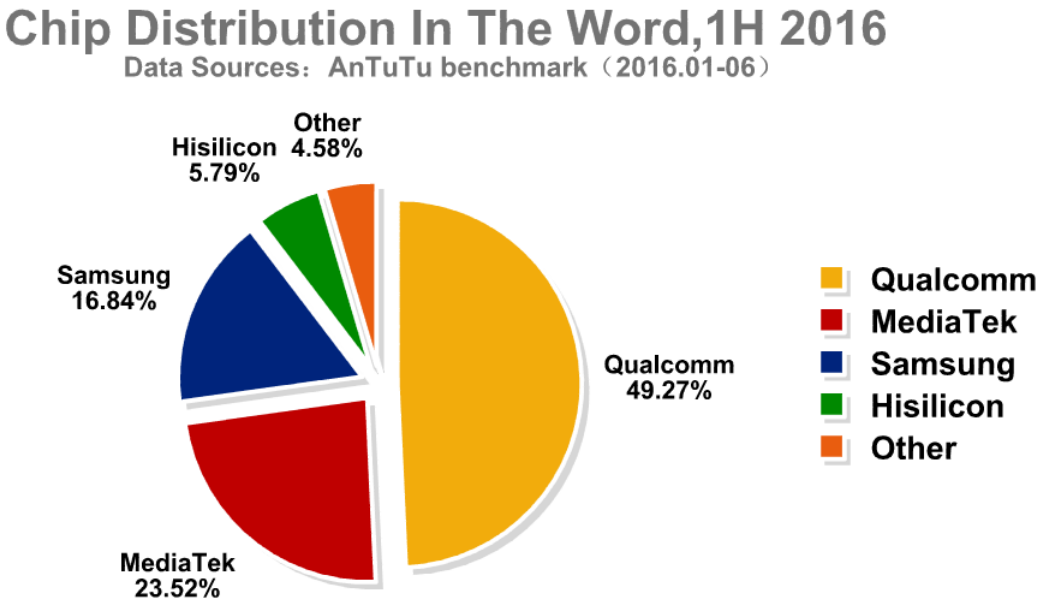

Abbildung 3.1: Marktanteile im Bereich System on Chip und Baseband-Prozessoren [42]

4 Baseband

In diesem Kapitel wird auf die Audio-Bypass Bedrohung, ausgehend von dem Baseband-Prozessor (BB), eingegangen. Da der BB das Modem eines Smartphones (SPs) darstellt, ist er auch zugleich eine der ersten möglichen Angriffsflächen, die sich an einem SP bieten. Der BB übernimmt sämtliche Kommunikation mit dem Mobilfunknetz (Daten- und Gesprächsverbindungen), in vielen Fällen wird auch das GPS Signal zur globalen Ortung innerhalb des BB verarbeitet. Damit stellt es einige der wichtigsten Dienste für heutige SP zur Verfügung und wird im Gegensatz zu der WLAN-Schnittstelle selten bis nie abgeschaltet (Flugmodus). Aus diesen Gründen scheint der BB für einen Angreifer, der einen Audio-Bypass Angriff (oder ein beliebiger anderer Angriff) auf ein SP durchführen will, ein lohnendes Ziel zu sein.
Im Verlauf dieses Kapitels wird aufgezeigt, dass ein direkter Angriff auf den BB eines SP, in Bezug auf den Audio-Bypass im BB Kontext, in der heute gängigen Architektur für SP nicht mehr so gravierend ist, wie es noch vor wenigen Jahren der Fall war. Dies ist Hauptsächlich bedingt durch die in modernen SPs verwendete Hardwarearchitektur die sich von dem omnipotenten BB als System on Chip Master abgewandt hat (siehe Kapitel 4.1.1). Das ein Angriff auf den BB auch heute noch effektiv ist liegt daran, dass er als Pivotingstation[2] angesehen werden kann, um das auf dem Applikationsprozessor (AP) laufende Betriebssystem zu attackieren.
Um die während der Analyse vorgestellten Schwachstellen (siehe Kapitel 4.2) verstehen und einordnen zu können, wird in diesem Kapitel zuerst über die allgemeine Architektur eines BB (Soft- als auch Hardware) und dessen Integration in das restliche System informiert. Anschließend wird die Schwachstellen- und Bedrohungsanalyse durchgeführt. Die Konzeption, Umsetzung und Prüfung auf Effektivität von Abwehrmaßnahmen wird in Kapitel 6 durchgeführt.

4.1 Baseband Architektur

In diesem Abschnitt wird eine kurze Übersicht über die in BB verwendete Hard- sowie Software gegeben. Des Weiteren wird aufgezeigt, welche Schnittstellen

[2] Der englische Begriff *pivoting* bezieht sich auf den Vorgang, ein System von einem bereits kompromittierten Subsystem innerhalb desselben Netzwerks oder derselben Anwendung anzugreifen. Mit dieser Technik lassen sich viele Abwehrmaßnahmen und Restriktionen umgehen, die sich hauptsächlich gegen Angriffe von außerhalb des Systems richten.

dem BB zur Verfügung stehen. Es wird ein kurzer Überblick über die allgemeine Implementation gegeben und anhand der von Qualcomm (QC) verwendeten Techniken genauer auf das Thema eingegangen. Es ist kaum öffentliche Dokumentation zu BB Softwarestacks vorhanden, die meisten Informationen sind entweder aus dem vorhandenen Kontext und Verhalten erschlossen, oder über die unerlaubte Veröffentlichung von internen Dokumenten im Internet bekannt geworden.
Über die im QC Umfeld eingesetzten Techniken ist aufgrund der hohen Marktanteile und dem daraus resultierenden Interesse für die verwendete Technologie mehr bekannt, als über die Techniken von anderen Herstellern (siehe Kapitel 3). Für den von QC auch in anderen Produkten eingesetzten Hexagon-Prozessoren ist ein Development Guide verfügbar [101]. Um diesen zu erhalten, muss man sich auf der QC Homepage registrieren und von einem Mitarbeiter freigeschaltet werden.

4.1.1 Hardware

Qualcomm BBs verwenden heutzutage sogenannte Hexagon Prozessoren. Die aktuelle Version 6 ist meistens in höherwertigen System on Chips (SoCs) zu finden und die Version 5 wird oft in günstigeren Modellen verbaut. Bei den anderen Herstellern im BB-Bereich wird im Allgemeinen noch auf eine ARM-Architektur gesetzt. Bei den Hexagon Prozessoren handelt es sich um eine Weiterentwicklung aus dem DSP Bereich, wo sie auch heute noch für Audiobearbeitung und zur Demodulation[3] von Signalen verwendet werden [43]. Der Prozessor ist hochgradig parallelisiert und bietet durch seine spezielle Architektur einen sehr geringen Energieverbrauch. Hierbei wird der Ansatz verfolgt, nicht die Taktrate zu erhöhen, sondern mehr Arbeit pro Zyklus zu verrichten, was durch die parallele Verarbeitung von bis zu 4 Instruktionen pro Zyklus ermöglicht wird [124].
Die von Weinmann [123] beschriebene Architektur ist typisch für ältere Geräte. Das in der Arbeit getestete Gerät wurde 2008 veröffentlicht, wird aber heute nicht mehr verwendet. Der Startvorgang eines auf einem modernen QC SoC basierendem SP beginnt mit einem Boot ROM auf dem AP des SoCs sowie einem weiteren ARM-Kern, der im späteren Betrieb des Systems das sogenannte Resource Power Management (RPM) ausführt [79].
Durch diese Architektur ist es möglich, einen Startprozess zu gestalten, in dem der BB nicht beteiligt ist und somit auch keinen Einfluss nehmen kann. Die vom BB benötigte Firmware wird daher erst während des Systemstarts von Android durch dessen Kernel in den BB geladen. Hierbei wird die Signatur der Firmware vom Modem Boot Authenticator (MBA) (der im internen, nichtflüchtigen (non-volatile memory, NVM) Speicher des BB abgelegt ist), vor dem Starten au-

[3] Demodulation ist ein Vorgang, der unter anderem zur Umwandlung von den im Mobilfunk eingesetzten Signalen in digitale Daten verwendet wird.

thentisiert [124]. Durch die Singnaturprüfung durch den MBA ist es nicht nötig, die BB-Firmware durch die im Android Kernel verwendeten Signaturmechanismen (siehe Kapitel 5.1.2) abzusichern.
Zum Laden von Firmwareimages wird der sogenannte Peripheral Image Loading (PIL) Treiber Pronto verwendet [102]. Dies spielt eng mit der Tatsache zusammen, dass der AP und BB sich einen Teil des, oberhalb des SoCs verbauten, Hauptspeichers (RAM) teilen. Zur Absicherung und Verwaltung dieses Speichers wird eine Memory Protection Unit (MPU) verwendet [41] (Qualcomm nennt diese XPU), die während des Startvorgangs konfiguriert wird. Hierbei soll sichergestellt werden, dass die Subsysteme, die Zugriff auf den Hauptspeicher des APs haben, nicht auf Speicherbereiche außerhalb ihrer designierten Speicherregion zugreifen (siehe Shared Memory Device in Kapitel 4.1.3).

4.1.2 Softwarestack

Bedingt durch die Taktung des Mobilfunknetzes müssen strikte Zeitfenster für die Kommunikation zwischen BB und Netz eingehalten werden. Zur Durchsetzung dieser Anforderungen wird ein Real Time Operating System (RTOS) eingesetzt. Dies bildet die Basis für den 3GPP konformen Softwarestack, der zur Kommunikation mit dem Mobilfunknetz eingesetzt wird. Im Folgenden wird das von QC verwendete BLAST/QuRT RTOS sowie der 3GPP konforme Softwarestack, Advanced Mode Subscriber Software (AMSS), beschrieben.

Das von Qualcomm selbst entwickelte RTOS BLAST/QuRT (es wird unter beiden Namen referenziert und vermarktet) ersetzt das seit 2012 nicht mehr weiterentwickelte REX RTOS, das auf Basis eines L4 Pistachio Microkernels aufbaut [124]. Es wird davon ausgegangen, dass in REX eine große Menge an Legacy-Code verwendet wurde und dieser Code ist auch zu Teilen, aus Gründen der Kompatibilität, in der QuRT Codebasis zu finden. Damit ist davon auszugehen, dass zumindest ein gewisser Teil der Sicherheitslücken, die in REX zu finden waren, auch in das neue RTOS migriert wurden.
Im Vergleich zu seinem Vorgänger bietet QuRT ein Mindestmaß an plattformbasierenden Sicherheitsmaßnahmen (Vergleich siehe Weinmann [124, 123]), die einen erfolgreichen Angriff erschweren können. Der Stack wird mittels Stack-Cookies (compilerseitig) vor einer ungewollten Manipulation geschützt (Over-/Underflow). Der Stack und Heap selbst können mittels Executable-Space Protection (NX) geschützt werden (das NX-Bit wird auch hardwareseitige respektiert) und QuRT bietet eine Kernel/User-Mode Separation, die es ermöglicht, AMSS in einem eigenen, isolierten, Speicherbereich auszuführen (kein Zugriff auf Kernelspeicher).
Der Schutz, der durch Address Space Layout Randomization (ASLR) geboten wird, ist zwar in der letzten Zeit durch mehrere Veröffentlichungen sehr stark

angegriffen worden [63, 66], stellt aber trotzdem immer noch eine ernst zu nehmende Hürde auf dem Weg zur ungewollten Codeausführung dar. Allerdings wird ASLR nur in sehr wenigen Embedded Systems bisher eingesetzt und wird auch von QuRT nicht unterstützt.

Bei AMSS handelt es sich um den von QC entwickelten, 3GPP konformen, Softwarestack zur Kommunikation mit den 2G/3G/4G Netzen. Der Stack teilt sich in drei Schichten auf [123]:

- L1 (Physikalische Schicht): Demodulation der Signale von den Antennen
- L2 (Data-Link Schicht): logisches Channel Managment, Vermittlung zwischen analogen und digitalen Signalen
- L3 (Management Schicht): größer und komplexer als Schicht eins und zwei, teilt sich in drei Unterschichten:
  - Radio Resource Managment (RR): Auf- und Abbau von Verbindungen
  - Mobility Management (MM): GPS und Lokalisierungsdienste
  - Connection Management (CM): Call Control (Auf- und Abbau von Gesprächsverbindungen), SMS sowie zusätzliche Dienste (z.B. USSD)

Als Ziel für einen Angriff ist besonders die Schicht L3 von Interesse, da unterhalb dieser Schicht die Nachrichten nur sehr kurz sind (kein Platz für Payload) oder nur auf Bit-Ebene betrachtet werden, ohne überhaupt geparsed zu werden. Für die Formatierung von Nachrichten auf Schicht drei wird in der Regel TLV[4] verwendet, in einigen Fällen aber auch TV[5] und LV[6]. Gerade diese uneinheitlichen Datenformate der Werte, teilweise ändert sich die Formatierung für eine identische Nachricht je nachdem ob sie über 2G, 3G oder 4G gesendet und verarbeitet wird, sorgt für eine große Angriffsfläche in der L3 Schicht.

4.1.3 Schnittstellen

Der BB hat eine Vielzahl an Schnittstellen. Für die spezifische Betrachtung in dieser Arbeit sind aber hauptsächlich drei Schnittstellen von Interesse: Das Shared Memory Device (SMD) (Schnittstelle zwischen BB und AP), der SLIMbus (Anbindung des BB an Mikrofon und Lautsprecher) sowie das RF-Frontend (Antennen zur Kommunikation mit dem Mobilfunknetz).

Im Bereich der SoCs mit integriertem BB hat sich die Anbindung mittels

[4]Type-Length-Value (TLV) ist ein Datenformat, bei dem jedes Datenpaket aus einer Angabe zum Typ (Type) sowie der Länge (Length) der gesendeten Daten (Value) besteht.
[5]Type-Value Datenformat
[6]Length-Value Datenformat

eines Shared Memory Device in den letzten Jahren durchgesetzt [53]. Zum einen bieten SMDs eine sehr hohe Datendurchsatzrate, die mit den aktuellen Entwicklungen im Mobilfunkbereich mithalten kann, sowie eine sehr geringe Latenz, die gerade im Bereich der mobilen Videotelefonie unentbehrlich ist. Diese Architektur ist sicherheitstechnisch unbedenklich, solange ein dedizierter, physikalisch vom restlichen Speicher des Systems abgetrennter, Speicher verwendet wird (in iPhones findet diese Architektur Verwendung). Allerdings wurde im Fall des Nexus 5X [81] sowie vieler anderer Android basierender SP auf einen dedizierten Speicher für dieses Interface verzichtet. Das bedeutet, dass die Speicherregion des SMD meistens innerhalb des auch vom AP verwendeten Arbeitsspeichers abgelegt wird und der BB somit Zugriff auf den Hauptspeicher des APs benötigt. Dieser Zugriff wird abgesichert durch die Verwendung einer MPU, die während des Startvorgangs des SP konfiguriert wird.
Im Fall von QC SoCs wird das proprietäre Qualcomm MSM Interface (QMI) [53] Protokoll für dieses SMD eingesetzt. QMI wird nicht nur für die Übertragung von Steuersignalen verwendet, sondern auch für die Übertragung von GPS und IP-Netzwerkverkehr. Über die genaue Implementation des SMDs auf BB Seite sind keine Details bekannt, der Einsatz des Protokolls auf Androidseite wird in Kapitel 5.1.4 genauer beschrieben.

Zur Gesprächsübertragung mittels des Mobilfunknetzes ist der BB an den SLIMbus des SP angeschlossen. Bei dem SLIMbus handelt es sich um einen digitalen Medienbus der es ermöglicht, den Audiopfad (und Videopfad) innerhalb des SP zu steuern. Dies ermöglicht die Verwendung verschiedener Ein- und Ausgabequellen, die über verschiedene Audiopfade geleitet werden können. Hierbei können diese Audiopfade verschiedene hardwareseitige Verstärker und Filter beinhalten, die dynamisch zur Bearbeitung des Audio- oder Videosignals zugeschaltet werden können (z.B. Rausch- oder Umgebungsgeräuschfilter). Im SLIMbus-Schema ist immer genau ein Produzent und mindestens ein Konsument vorgesehen sowie ein Busmaster. Dieser Busmaster ist für die Festlegung des Audiopfades über den SLIMbus verantwortlich (*routing*). Gerade in älteren SP-Modellen wurde aus historischen (der BB ist für das Audiorouting in Featurephones verantwortlich) und technischen (viele der Audioverstärker und Filter waren an das BB angebunden) Gründen der SLIMbus vom BB verwaltet [4, 123]. Diese Architektur wird aber aufgrund der geänderten Anforderungen an SP heutzutage nicht mehr verwendet. Mit dem Aufkommen mobiler Audiokommunikation abseits des klassischen Telefongesprächs (Messenger mit Audioübertragung und VoIP-Clients für SP) und der Verwendung von SP in verschiedensten Audioanwendungen ist es erforderlich geworden, die Audiocodecs sowie Verstärker und Filter in einen dedizierten Chip zu verlagern. Im Zuge dieser Architekturänderung wurde die Kontrolle des SLIMbus an den Audiochip übertragen und damit indirekt an den AP, der den Audiochip steuert.

Das RF-Frontend ist das physikalische Interface zur Kommunikation mit dem Mobilfunknetz. Es besteht aus einer oder mehreren Antennen und verschiedenen Filtern und Verstärkern. Die vom RF-Frontend kommenden Signale durchlaufen das sogenannte analoge Baseband, das die analogen Signale in digitale Informationen konvertiert. Sämtliche über das Mobilfunknetz empfangenen Daten sowie Steuersignale werden über diesen Pfad übertragen, um anschließend vom BB verarbeitet zu werden.

4.2 Schwachstellen- und Bedrohungsanalyse

In diesem Kapitel wird die Schwachstellen- und Bedrohungsanalyse des Baseband-Prozessor (BB) Komplexes vorgenommen. Hierbei wird versucht, die vielen verschiedenen möglichen Angriffe in übergeordnete Kategorien einzuteilen und diese Szenarien zu bewerten. Bei der Bewertung selbst wird versucht, auf Basis von konkreten Fällen und Schwachstellen zu arbeiten. In Fällen, wo dies nicht möglich ist, wird der Autor nach bestem Gewissen versuchen, die Situation entsprechend der vorhandenen Informationen zu bewerten.
Da die vollständige Betrachtung sämtlicher möglicher Angriffe die Grenzen dieser Arbeit überschreiten würde, wird sich die Schwachstellenanalyse des BB auf Angriffsvektoren, die auf dem Mobilfunknetz basieren, konzentrieren. Die Mobilfunknetzschnittstelle wurde ausgewählt, da sie sich einerseits bereits im Audiokontext des Mikrofons befindet, anderseits, weil der Zugang der zur Kommunikation verwendeten Hardware in den letzten Jahren immer einfacher wurde. Prinzipiell könnten Angriffe auf das BB jedoch auch über andere Schnittstellen erfolgen (z.B. USB [97], WLAN- oder GPS-Signale).
Die Analyse beginnt mit einer Übersicht über die möglichen Angriffsvektoren auf das BB und behandelt anschließend den für einen Angriff auf den BB meistens notwendigen Remote Code Execution (RCE) Angriff, ohne den kein weiterführender Angriff auf das restliche System möglich ist. Die weiterführenden Angriffe erfolgen auf den SLIMbus und das Shared Memory Device (respektive den Applikationsprozessor (AP)) oder sind ein Man-in-the-Middle (MITM) Angriff auf den Datenverkehr des APs, der im Fall von mobiler Datennutzung durch den BB geleitet wird.

Das in Abbildung 4.1 dargestellte Data Flow Diagrams (DFD) zeigt den BB und die verschiedenen in Kapitel 4.1.3 vorgestellten Schnittstellen mit ihren entsprechenden Vertrauensgrenzen. Es zeigt besonders die Abgrenzung des BB zum Rest des Systems und die Anbindung des SLIMbus an den BB ohne Kontrolle des BBs über den Audiopfad.

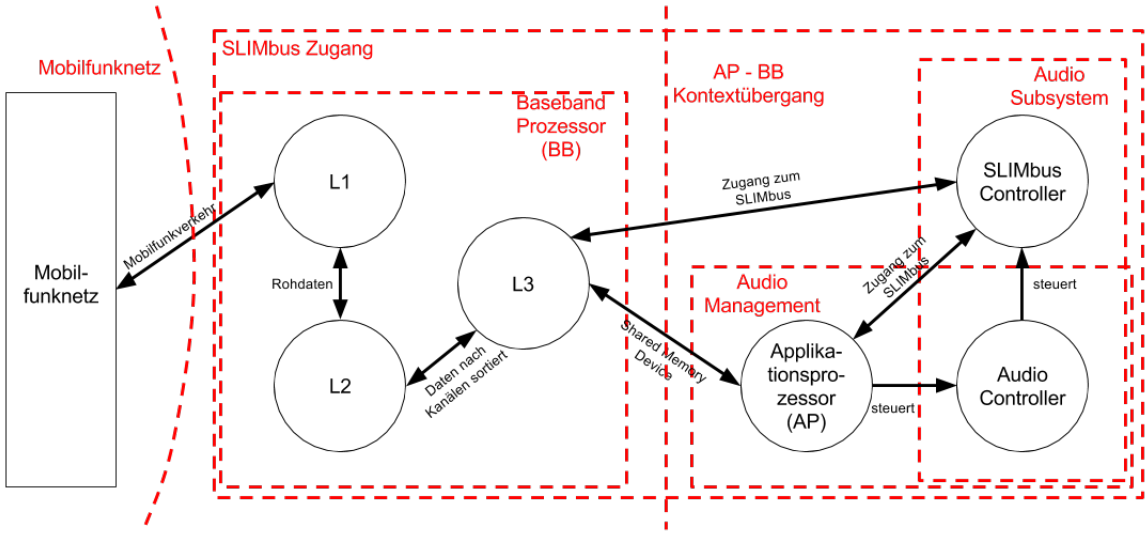

Abbildung 4.1: Data Flow Diagrams des Baseband-Prozessor Komplex, eine Legende ist in Anhang 9.1.1 zu finden.

4.2.1 Angriffsvektoren

Angriffsvektoren auf den BB, die in Bezug auf den Audio-Bypass Angriff von Interesse sind, werden hauptsächlich über die Mobilfunkschnittstelle ausgeführt. Diese teilen sich in 2 Kategorien: Die verschiedenen physikalischen Möglichkeiten mit dem BB zu kommunizieren und die Art der Nachrichten, die dabei gesendet werden.

In Hinsicht auf die verschiedenen Möglichkeiten, mit dem BB über ein Mobilfunknetz zu kommunizieren, gab es seit Einführung der 3GPP Standards einige Entwicklungen. Bis vor 15 Jahren war die Ausrüstung, die benötigt wird, um ein eigenes Mobilfunknetz zur Kommunikation mit dem BB über das RF-Frontend auszustrahlen, wenig verbreitet und meistens nur unter Restriktionen zu kaufen.
Allerdings hat die technische Entwicklung in diesem Bereich dafür gesorgt, dass schon 2012 eine leistungsfähige Base Transceiver Station (BTS) für ca. 1300€ ohne jegliche Restriktionen erhältlich war [123]. Dieser Trend hat sich bis heute fortgesetzt und es gibt heute verschiedenste BTS Anbieter, die unterschiedlich leistungsfähige Modelle für 2G Netzwerke ab einem Preis von 400€ [94] und Modelle für 3G und 4G ab 900€ [73] vertreiben. Diese preisgünstigen Modelle haben eine begrenzte Sendereichweite und können nur eine gewisse Zahl von Clients verwalten. Es sind aber auch größere Anlagen mit mehr Ressourcen zur Nutzerverwaltung und einer stärkeren Sendeleistung erhältlich. Dies alles wird zusätzlich dadurch begünstigt, dass es verschiedene Open Source Projekte gibt, die Softwarestacks zum Betrieb von vollständigen 2G, 3G und 4G Netzwerken

anbieten sowie die dazugehörigen Managementtools [52, 96].
Es muss darauf hingewiesen werden, dass viele der älteren Systeme im Bereich der Mobilfunktechnologie keinerlei Form der Authentisierung kennen und als Schutzmaßnahme einzig und allein darauf setzen, dass der Zugang zu dem entsprechenden Netzwerk historisch gesehen nur mit proprietärer Hardware, die nicht frei verfügbar war, möglich ist [69]. Dieses Problem wurde zu Teilen in den 3G und 4G Standards behoben, aber durch die immer noch verbleibende Kompatibilität der aktuellen SP Generation mit den 2G Standards mitigiert dies die Bedrohung nur zu Teilen.
Eine weitere Möglichkeit mit dem BB zu kommunizieren, ist der Kontakt über das von den Serviceprovidern zur Verfügung gestellte Mobilfunknetzwerk. Diese filtern zwar heutzutage die meisten der ungewollten oder nicht standardkonformen Nachrichten, aber es besteht die Möglichkeit, dass der Netzbetreiber die Bösartigkeit der versendeten Nachrichten nicht erkennt, durch einen staatlichen Akteur dazu bewegt wird, zu kooperieren oder schlicht selbst das Ziel eines Angriffs wird. Dieser Angriff könnte zu einer Infiltration der Netzwerkinfrastruktur führen und den Angreifern ermöglichen, ungewollte oder schädliche Nachrichten über das Netzwerk des Serviceproviders zu versenden.

Die Art der Nachrichten (Protokollebene) spielt auch eine erhebliche Rolle für den Angriff, wird aber zu Teilen durch den verwendeten Netzwerktyp bestimmt. In 2G und 3G Netzwerken ist es möglich SMS zuzustellen, ohne dass die BTS sich gegenüber dem SP authentifizieren muss. Dies ist in 4G Netzwerken nicht mehr möglich. Prinzipiell stehen für den Angriff zwei verschiedene Möglichkeiten zur Verfügung.
Zum einen kann der BB über die Steuersignale des Mobilfunknetzes (3GPP Standards) angegriffen werden. Dazu kann entweder über valide Steuersignale die Konfiguration des BB geändert werden (z.B. andere Einwahlparameter). Diese Art der Steuersignale werden natürlich auf Netzwerkebene von den Serviceprovidern gefiltert, die es ihren Kunden nicht erlauben, diese Art von Nachrichten über ihr Netzwerk zu versenden. Es kann aber auch das Parsing dieser Steuersignale angegriffen werden. Das Ziel wäre hierbei, einen Fehler in der Verarbeitung der empfangenen Nachrichten auszunutzen, um die Kontrolle über die Codeausführung innerhalb des BB zu übernehmen.
Zum anderen können aber auch SMS als Angriffsweg verwendet werden. Hierbei kann das Parsing von herkömmlichen SMS, im Sinne von kurzen Textnachrichten, oder von binären SMS, angegriffen werden. Dies erfolgt mit dem Ziel, wie bei dem Angriff über Steuersignale, die Codeausführung des BB unter die Kontrolle des Angreifers zu bringen. Binäre SMS werden zur Übertragung von Konfigurationsparametern vom Mobilfunknetz an den BB verwendet, können aber ebenfalls missbraucht werden, um die Konfiguration eines BB zu

manipulieren. Wie die Steuersignale wird auch der Versand von binären SMS auf Netzebene von den Serviceprovidern strikt kontrolliert und nur wenigen Großkunden (z.B. Banken) ist es gestattet, binäre SMS zu versenden.

4.2.2 Remote Code Execution Angriff

Der RCE Angriff ist unerlässlich für alle nachfolgenden Angriffe. Ein Angriff über die Manipulation der Konfigurationsparameter des BB kann zwar zu Schaden führen, ist jedoch vernachlässigbar in Bezug auf die Audio-Bypass Bedrohung. Das genaue Ziel des Angriffs ist es, eine Schwachstelle im anzugreifenden System auszunutzen, um den Kontrollfluss der Software so zu manipulieren, dass vom Angreifer kontrollierter Code ausgeführt wird.
Eine klassische Methode zum Erreichen dieses Ziels wäre die Ausnutzung eines Fehlers im Code, um den aktuellen Codefluss so zu beeinflussen, dass ein vom Angreifer zu beeinflussender Speicherbereich ausgeführt wird. Klassisch würde der Angreifer versuchen, die Daten einer Nachricht so zu konstruieren, dass sie als valide Befehle vom System interpretiert werden können und die Codeausführung auf den Speicherbereich lenken, der den weiteren Code des Angreifers enthält.

Kategorie	Identifizierung	Ausnutzung
Zeit	5	0
Expertise	5	2
Wissen über Ziel	3	2
Zugang zu Ziel	0	1
Equipment	4	4
Partielles Angriffspotential	17	9
Angriffspotential	26	
Benötigtes Potential	Hoch	

Tabelle 4.1: Angriffspotentialbewertung eines Remote Code Execution-Angriffs. Eine Aufschlüsselung der für die Einschätzung verwendeten Werte sowie deren weiterführende Beschreibung befindet sich in den entsprechenden Tabellen in Anhang 9.2.

Für den RCE-Angriff auf den Softwarestack des BB über das Mobilfunknetz gibt es mehrere Beispiele [123, 93], aber auch Angriffe über die USB- oder WLAN-Verbindung [59, 127] des SPs sind bekannt. Die Erkenntnisse aus diesen Angriffen wurden verwendet, um das Angriffspotential für diese Art von Angriff zu bewerten. Im Folgenden werden die einzelnen Kategorien aufgegriffen und die vergebenen Werte begründet.

Bei dem Angriff auf den BB ist zu beachten, dass bei modernen SP eine permanente Modifikation der Software durch die Signaturprüfung verhindert wird. Sollte die Möglichkeit, Code auszuführen dazu verwendet werden, das beim Start des Systems geladene Image zu modifizieren, wird beim nächsten Start die Signaturprüfung fehlschlagen und das SP wird nicht oder nur mit eingeschränkter Funktionalität starten. Eine Modifikation des laufenden Systems sowie der Konfigurationsparameter (die zum Teil persistent gespeichert werden) ist allerdings möglich, eine Persistenz im BB ist jedoch nicht auf diesem Weg zu erreichen.

Benötigte Zeit:
Wie aus Golde et al. [93] zu entnehmen ist, haben sich zwei Entwickler teilweise drei bis sechs Monate mit der Überprüfung von BB Code beschäftigt, um eine RCE Schwachstelle zu finden und den für die Ausnutzung nötigen Code zu entwickeln. Ein großer Anteil dieser Zeit musste auf das Reverse Engeneering der BB Firmware oder anderen Teilen des Softwarestacks verwendet werden, um eine angreifbare Schwachstelle zu identifizieren.
Ist ein passender Exploit[7] für das anzugreifende Ziel bereits vorhanden, wird kein weiterer Aufwand benötigt, um einen Angriff durchzuführen. Sollte jedoch der Code angepasst werden müssen (z.B. aufgrund eines Updates), kann dies variieren.

Benötigte Expertise:
Wie aus den bekannten Angriffen zu entnehmen ist, wird ein hohes Maß an Wissen über den anzugreifenden Hard- als auch Softwarestack benötigt, um einen erfolgreichen Angriff zu implementieren. Für die Identifikation einer solchen Schwachstelle und die Entwicklung eines entsprechenden Angriffs, wird Erfahrung mit dem proprietären RTOS und 3GPP Softwarestack sowie der verwendeten Hexagon Prozessorstruktur benötigt. Zusätzlich wird die Fähigkeit benötigt, die zwar schwachen, aber vorhandenen Schutzmaßnahmen des Systems zu überwinden sowie die für die Kommunikation mit dem BB verwendete BTS zu betreiben.
Für die aktive Durchführung eines solchen Angriffs muss der Angreifer in der Lage sein, die BTS zu bedienen sowie den eigentlichen Angriff zu starten. Dieser besteht meistens in der Ausführung eines Skripts in der Betriebssystemkonsole des für den Angriff verwendeten Computers, welcher in der Regel auch als Managementkonsole für die BTS verwendet wird.

Benötigtes Wissen über Ziel:
Für die erfolgreiche Identifikation und die Entwicklung eines Angriffs wird

[7] Exploit ist ein Englischer Fachbegriff für die technische Umsetzung eines spezifischen Angriffs (z.B. als Skript oder Programm).

ausführliches Wissen über die anzugreifende Plattform benötigt. Im Fall des hier als Basis verwendeten Angriffs [93] wurde die verwendete Schwachstelle in einer nicht öffentlichen Benachrichtigung des Herstellers gefunden. Obwohl schon eine Schwachstelle bekannt war, wurde eine ausführliche Recherche zum verwendeten Softwarestack und dessen Schutzmechanismen benötigt, um einen passenden Angriff zu entwickeln.
Zur Durchführung eines Angriffs muss die Marke, das Modell sowie die verwendete Softwareversion des Ziels bekannt sein. Auf dieser Basis kann entweder ein bereits vorhandener Exploit eingesetzt werden, oder die Informationen können genutzt werden, um einen entsprechenden Angriff zu identifizieren.
Neben diesen Informationen benötigt der Angreifer auch die Möglichkeit, das anzugreifende SP selbst zu identifizieren, hierfür benötigt er die International Mobile Equipment Identity (IMEI) Nummer. Diese ist eine einmalige Identifikationsnummer, die jedes SP weltweit identifiziert. Sie kann aber auch mittels der International Mobile Subscriber Identity (IMSI) des Ziels oder der dazu gehörigen Telefonnummer in Erfahrung gebracht werden.

Benötigter Zugang zum Ziel:
Da es sich bei SP um keine Einzelstücke handelt und die verschiedenen (auch veralteten Versionen) der Betriebssysteme heutzutage einfach zu erhalten sind, stellt es kein Problem dar, ein Testgerät für die Schwachstellensuche zu organisieren.
Zur eigentlichen Ausnutzung der Schwachstelle wird so gut wie keine Zeit benötigt. In dem von Weinmann beschriebenen Fall [123] wurde ein angreifbarer Fehler in der ersten Nachricht, die zur Anmeldung zwischen SP und Mobilfunknetz ausgetauscht wird, gefunden. Da der Angriff über das Mobilfunknetz stattfindet (auch wenn eine eigene BTS für den Angriff verwendet wird), und der BB keine direkte Rückmeldung an den Nutzer des SP gibt, läuft diese Art des Angriffs meistens ohne bemerkbare Anzeichen für den Anwender des SP ab.
Ein möglicher Faktor, der diesen Wert modifiziert, ist die Persistenz innerhalb des BB. Da das BB nach jedem Neustart erneut attackiert werden muss, ist unter Umständen ein wesentlich längerer Zugang zum Ziel nötig, um den BB dauerhaft unter Kontrolle zu halten.

Benötigtes Equipment:
Für die Identifikation eines Angriffs wird neben verschiedenen Tools auch eine Umgebung zum Decompilieren von Hexagon-Anwendungen (im Fall von Qualcomm (QC)) benötigt. Diese Anwendungen liegen in einem proprietären Format

vor, für das spezielle Routinen benötigt werden. Ein Plugin für IDA Pro[8] existiert [68] und es kann davon ausgegangen werden, dass weitere, nicht öffentliche, Anwendungen dieser Art existieren.
Für die praktische Durchführung dieses Angriffs, wie für die Identifikation, wird eine BTS benötigt. Eine Alternative hierzu ist die Nutzung des allgemeinen Mobilfunknetzes. Hierbei müsste entweder der Betreiber des Netzes kooperieren (z.B. im Fall bei der Anfrage durch den Staat) oder seine Schutzmaßnahmen umgangen werden (Angriff auf die Infrastruktur des Netzbetreibers). Hierbei wird auch das Problem der Persistenz innerhalb des BB beseitigt, da dieser jedes Mal, wenn das SP neu gestartet wird, erneut attackiert werden kann.

4.2.3 Angriff auf SLIMbus

Von Weinmann [123] wird ein Audio-Bypass Angriff auf Basis eines veralteten 3GPP Features sowie der Tatsache, dass der BB in älteren SP noch die Kontrolle über den SLIMbus hatte, vorgestellt. Diese Architektur wird, wie in Kapitel 4.1.3 beschrieben, heutzutage nicht mehr verwendet und somit ist ein direkter Angriff vom BB auf den SLIMbus nicht mehr möglich. Da der BB aber immer noch an den SLIMbus angeschlossen ist, besteht trotzdem die Bedrohung eines Angriffs. Dieser Angriff müsste sich aber gegen das Shared Memory Device (SMD) wenden und entweder versuchen, über direkten Speicherzugriff das Androidsystem zu übernehmen oder versuchen, durch einen Angriff auf die proprietäre Software auf Androidseite, die Steuersignale des BB ausliest und interpretiert, anzugreifen.
Der Angriff auf das SMD wird in Kapitel 4.2.4 beschrieben, der Angriff auf die Schnittstellensoftware in Kapitel 5.2.3.
Allerdings besteht immer noch eine passive Bedrohung, durch Angriffe auf den BB, für Telefongespräche. Der BB kann den SLIMbus nicht mehr steuern, trotzdem muss er in der Lage sein, während eines Telefongesprächs auf die Audiodaten des Mikrofons zuzugreifen. Dies könnte dazu genutzt werden, sollte keine Transportverschlüsselung verwendet werden, die Audiodaten eines Telefongesprächs zu kopieren und an den Angreifer zu senden. Android bietet keine native Transportverschlüsselung für Audiodaten während der Übertragung innerhalb des SPs.

4.2.4 Angriff auf das Shared Memory Device

Ein Angriff auf das SMD zwischen BB und AP stellt ein besonderes Risiko für das auf dem AP laufende High Level Operation System (HLOS) dar. Der BB hat

[8] Interactive Disassembler Pro (IDA Pro) ist ein von der Firma Hex-Rays [71] vertriebener Disassembler und Debugger der von vielen professionellen Reverse Engineers eingesetzt wird.

für das im Hauptspeicher des APs abgelegte SMD hardwarenahen Lese- und Schreibzugriff. Sollte es dem Angreifer gelingen, aus der für das SMD allozierten Speicherregion auszubrechen, wäre es möglich, den kompletten Speicherbereich des HLOS auszulesen und/oder zu manipulieren. Dies führt zum vollständigen Verlust der Integrität des Systems. Geheime Schlüssel können ausgelesen sowie der Speicherinhalt zugunsten des Angreifers manipuliert werden (z.B: deaktivieren von Schutzmaßnahmen wie SELinux). Besonders schwerwiegend an diesem Angriff auf das HLOS ist, dass keine Schutzmaßnahme auf der Seite des HLOS diesen Angriff mitigieren kann. Durch die Fähigkeit des Angreifers, Speicherinhalt direkt zu manipulieren ist es ihm möglich, sämtliche Abwehrmaßnahmen zu ignorieren.

Kategorie	Identifizierung	Ausnutzung
Zeit	4	0
Expertise	5	2
Wissen über Ziel	4	0
Zugang zu Ziel	0	1
Equipment	4	0
Partielles Angriffspotential	17	3
Angriffspotential	21	
Benötigtes Potential	Mittel	

Tabelle 4.2: Angriffspotentialbewertung eines Angriffs auf das Shared Memory Device. Eine Aufschlüsselung der für die Einschätzung verwendeten Werte sowie deren weiterführende Beschreibung befindet sich in den entsprechenden Tabellen in Anhang 9.2.

Ein Beispiel für einen sehr ähnlichen Angriff wird von Beniamini [128] beschrieben. Hierbei wird der AP eines SPs über die Speicheranbindung des Wireless LAN (WLAN) angegriffen. Der Angriff ermöglicht das Lesen und Schreiben des gesamten AP Hauptspeichers über die Direct Memory Access (DMA) Funktion des Peripheral Component Interconnect Express (PCIe) Bus, der für die Anbindung des WLAN-Subsystems an den AP verwendet wird.
Hierbei werden zwei verschiedene Herangehensweisen untersucht. Zum einen wird die DMA-Schnittstelle direkt attackiert, zum anderen wird die Schnittstellensoftware auf AP Seite attackiert. Ein Vergleich mit diesem Angriff und der für die im BB-Kontext verwendete Schnittstellensoftware wird in Kapitel 5.2.3 vorgenommen.
Der direkte Angriff auf die DMA Schnittstelle soll hier als Basis zur Bewertung des Angriffs auf die von QC verwendete SMD-Schnittstelle herangezogen werden. Ein Vergleich der beiden Systeme zeigt, dass sie mit direktem Zugriff

auf den Hauptspeicher des APs ausgestattet sind und dieser Zugriff über die Verwendung einer Memory Protection Unit abgesichert wird. In dem von Beniamini vorgestellten Fall [128] wurde festgestellt, dass diese MPU entweder nicht konfiguriert wurde, oder aber erst gar nicht vorhanden ist. So konnte über die Verwendung der DMA-Schnittstelle direkt auf arbiträre Speicherregionen des APs geschrieben werden.
Eine Schutzmaßnahme hemmt aber auch diesen Angriff. Da ein Android SP während des Startprozesses die Signaturen sämtlicher Systemkomponenten prüft, muss ein weiterer Angriff für Android existieren, um persistent Kontrolle über das SP zu erhalten. Dies schwächt diesen Angriff jedoch nur minimal ab, da es immer noch möglich wäre, das SP nach jedem Neustart erneut zu attackieren oder eine App zu installieren, die einen Audio-Bypass oder weitere Angriffe durchführen (siehe Kapitel 5.2.5) kann.
Bei der Bewertung des Angriffs ist zu beachten, dass der BB als eine Pivotingstation verwendet wird, hierfür wird ein RCE-Angriff benötigt. Der RCE-Angriff wird nicht mit in diese Bewertung einbezogen, was zu einer relativ niedrigen Bewertung führt. Es darf jedoch nicht vergessen werden, dass dieser Angriff nur für einen Angreifer möglich ist, der das Angriffspotential aufbringt, um einen RCE-Angriff zu identifizieren und auszuführen.

Benötigte Zeit:
Der Quelle [128] ist zu entnehmen, dass die Identifizierung der Schwachstellen nach Erreichen von RCE im Subsystem mit moderatem Zeitaufwand verbunden war.
Ist ein Exploit bereits vorhanden, wird kein weiterer Aufwand benötigt, um ein Ziel anzugreifen. Sollte jedoch der Code angepasst werden müssen (z.B. aufgrund eines Updates), kann dies variieren.

Benötigte Expertise:
Zur Identifikation einer solchen Schwachstelle bedarf es eines fundierten Wissens in mehreren Fachgebieten, zu denen SP Hardwarearchitektur, ARM Speicherarchitektur, SMD Architektur (QC proprietäres Wissen) und viele weitere Bereiche zählen.
Für die Verwendung eines weiterführenden Angriffs wird aber auch ein gewisser Fähigkeitsgrad vom Angreifer benötigt, da der Angriff in den bereits vorhandenen Quelltext eingebettet werden muss. Es besteht auch die Möglichkeit, dass der Angreifer ein Implantat im Code des BB zurücklässt, das es ermöglicht, Code nachzuladen ohne in direkter Nähe des anzugreifenden Ziels zu sein (z.B. über binäre SMS). Dies wird in Kapitel 4.2.5 betrachtet.

Benötigtes Wissen über Ziel:
Neben dem benötigten Fähigkeitsgrad ist auch spezifisches Wissen über die

anzugreifende Plattform erforderlich. Für die erfolgreiche Identifizierung eines Angriffs auf die SMD Schnittstelle wird detailliertes Wissen über dessen Funktion und die verwendete Hardware benötigt. Hierzu zählen die verwendete Art der MPU (QC proprietäre Architektur: XPU), der allgemeine Aufbau des verwendeten Speichers sowie das verwendete Protokoll (QMI).
Da wir uns bereits innerhalb des anzugreifenden Systems befinden, werden für die Durchführung dieses Angriffs nur Wissen über den verwendeten Softwarestack und dessen Version benötigt, um sicherzustellen, dass dieser für den verwendeten Angriff anfällig ist.

Benötigter Zugang zu Ziel:
Für die Identifikation als auch die Ausnutzung eines solchen Angriffs gelten dieselben Bedingungen wie für einen RCE-Angriff (siehe 4.2.2).

Benötigtes Equipment:
Für die Identifikation diesen Angriffs wird, wie auch bei dem RCE-Angriff, eine breite Auswahl an Tools benötigt. Es wird ebenfalls eine Umgebung zum Decompilieren von Hexagon Anwendungen benötigt. Es werden keine Hardwarekomponenten benötigt, ein hardwarenaher Zugriff auf das Speicherinterface (z.B. eine JTAG-Debugschnittstelle) könnte aber die Entwicklung beschleunigen [59].
Da der Angriff auf einem RCE-Angriff basiert wird davon ausgegangen, dass keine weitere Ausrüstung zur Verwendung dieses Angriffs benötigt wird.

4.2.5 Man-in-the-Middle Angriff

Ziel eines MitM Angriffs ist es, den Netzwerkverkehr des anzugreifenden Ziels abzuhören oder auch zu manipulieren. Dies kann weitreichende Folgen haben, da sämtlicher mobiler Datenverkehr durch den BB zum AP geleitet wird. Neben den Implikationen für die Privatsphäre des angegriffenen Ziels wäre auch eine Manipulation von Datenverbindungen möglich. Dies würde es z.B. ermöglichen, bösartigen Code in nicht verschlüsselte Verbindungen des Smartphones (SPs) einzuschleusen, um so einen Prozess auf dem AP zu attackieren [93]. Dieser Angriff kann komplett verhindert werden durch den standardmäßigen Einsatz von Transportverschlüsselung.
Eine weitere Möglichkeit wäre der Angriff auf den Updateprozess einer der Systemkomponenten oder einer App. Apps, die über den Google Play-Store installiert wurden sowie Androids eigene Komponenten, verwenden zum Laden von Updates (überprüft mittels eines Netzwerksniffers) eine verschlüsselte Verbindung. Selbst im Fall, dass diese Verbindung aufgebrochen und manipuliert wird, sind die App spezifischen Updates jeweils mit einem Zertifikat ihres Entwicklers signiert (siehe Kapitel 5.1.2). Sollte das Update während

des Transports manipuliert werden, wird dieses Zertifikat ungültig und die Installation des Updates schlägt fehl.
Allerdings treten immer wieder Fälle auf, bei denen sich Entwickler nicht an das gängige Vorgehen zum Aktualisieren von Apps halten und eine nicht verschlüsselte Verbindung nutzen, wie beispielsweise bei einem älteren Vorfall mit Samsungs Android Tastatur "SwiftKey", welche Updates ohne Authentisierung und Transportverschlüsselung durchführte [121]. Dies ermöglicht einen MitM Angriff auf den Updateprozess dieser App, daraus folgend kann der Angreifer die Kontrolle über sie übernehmen. Dies kann zu einem ungewollten Zugriff auf das System, mit den Rechten der initial attackierten App, führen.
Das Problem an dieser Art von Angriff ist, dass es erheblichen Mehraufwand für den BB bedeuten würde, als MitM-Station zu agieren. Die Aufgabe, den Netzwerkverkehr zu parsen und die entsprechenden Pakete zu manipulieren, könnte die Leistungskapazität des BB, vor allem im parallelen Betrieb mit seinen eigentlichen Aufgaben, schnell überlasten [59]. Der BB muss sich an strikte Zeitfenster zur Kommunikation mit dem Mobilfunknetz halten, ein Nichteinhalten dieser Zeitfenster führt zu einem Verbindungsabbruch [125].
Aufgrund dieser Einschränkungen wird davon abgesehen, eine Bewertung dieses Angriffs vorzunehmen, da die Modellierung (z.B. für die benötigte Zeit) nicht sinnvoll möglich ist.

5 Android

Nach der Darstellung der Bedrohung durch den Baseband-Prozessor (BB) im letzten Kapitel, wird in diesem Abschnitt auf die Audio-Bypass Bedrohung auf Seite des Applikationsprozessors und Android eingegangen. Als High Level Operation System (HLOS), mit vielen Schnittstellen und der Möglichkeit des Nutzers, das System zu modifizieren, bietet Android eine große Angriffsfläche. Im Gegensatz zum BB und dessen geschlossener und abseits des Nutzers ausgeführten Tätigkeit, ist Android direkt mit dem Anwender verbunden und muss es ihm ermöglichen, Apps zu installieren und weitere Dienste über sein SP aufzurufen. Angriffe auf Android sind nicht nur auf Hard- oder Software begrenzt, sondern können sich auch gegen den Nutzer selbst richten, um dessen Befugnisse innerhalb des Systems zu missbrauchen. Dies eröffnet neue Angriffswege im Vergleich zum BB.
In diesem Abschnitt werden, ähnlich dem Vorgehen zur Analyse des BB, erst der allgemeine Aufbau von Android, die verwendeten Schutzmaßnahmen und die vorhandenen Schnittstellen beschrieben. Da wir uns wesentlich weiter von der Hardware entfernt haben, wurde an dieser Stelle darauf verzichtet, weiter auf den ARM-Prozessor einzugehen, auf dem Android läuft und das Hauptaugenmerk auf die Software gerichtet. Das Kapitel wird abgeschlossen durch die Schwachstellen- und Bedrohungsanalyse der Audio-Bypass Bedrohung.

5.1 Android Architektur

Android ist ein auf dem Linux Kernel basierendes Open Source Betriebssystem (Operating System (OS)) für Mobilgeräte, das von Google/Alphabet vermarktet und weiterentwickelt wird. Die aktuelle Version 7.1 von Android basiert auf der Linux-Kernel Version 4.10. Es ist in vielerlei Hinsicht mit einer klassischen Linux Distribution (RedHat, Fedora, Ubuntu) zu vergleichen, einzig die fehlende Unterstützung der meisten GNU-Tools sowie die fehlende glibc-Bibliothek unterscheidet es von anderen Distributionen. Das Open Source Betriebssystem wird von verschiedenen Herstellern von Smartphones und Tablets modifiziert und auf die Plattform des Herstellers angepasst. Hierbei entsteht die in Kapitel 5.1.6 besprochene Fragmentierung der weltweit eingesetzten Androidversionen. In diesem Kapitel wird die allgemeine Architektur des Android Betriebssystems aufgezeigt, es wird eine Übersicht über die meistverbreiteten Versionen gegeben sowie die Modifizierung der Hersteller am Betriebssystem besprochen.

Einzelne Aspekte dieser Punkte werden in Kapitel 5.2 wieder aufgegriffen und genauer dargestellt.

5.1.1 Android Softwarestack

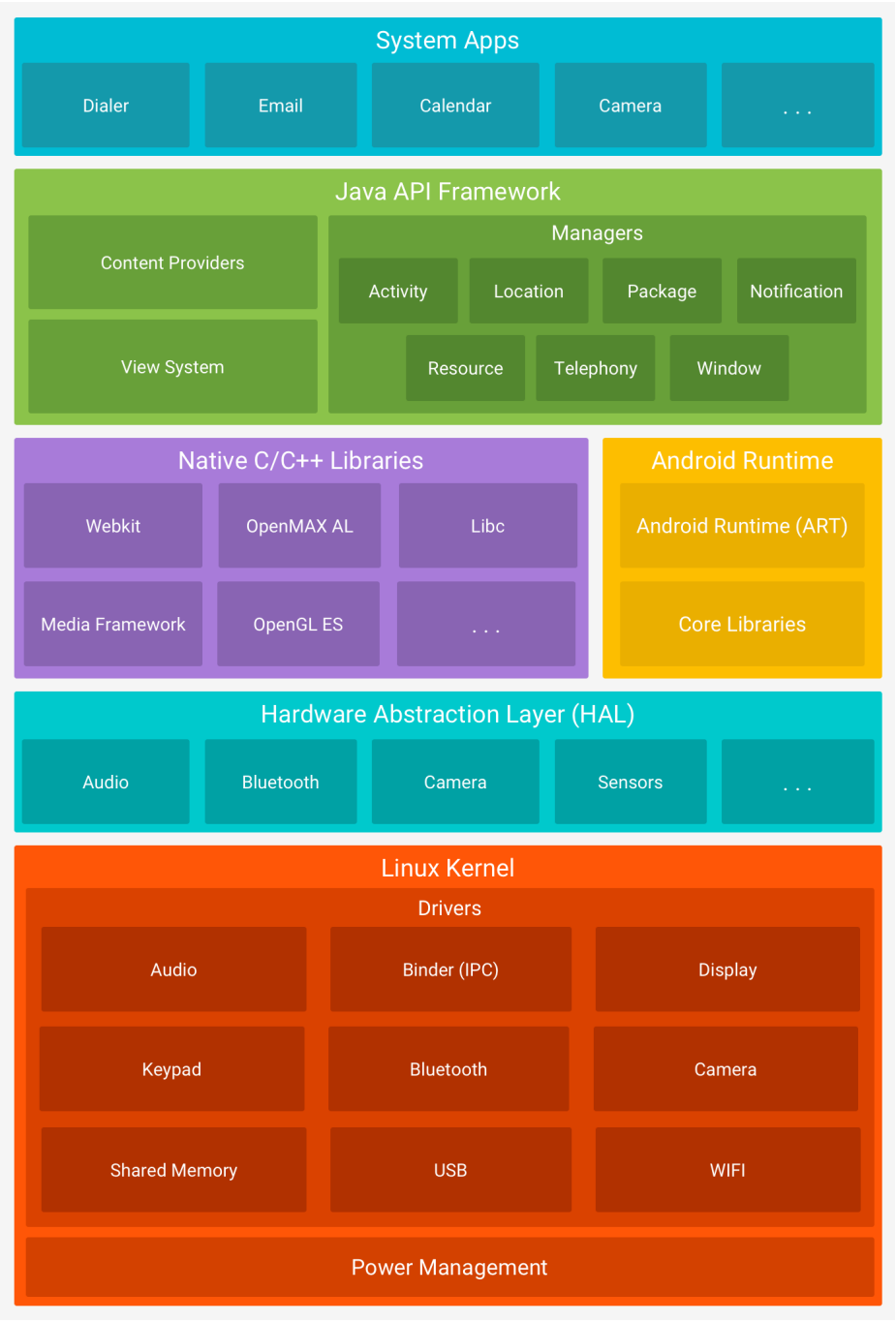

Abbildung 5.1: Aufbau des Android Betriebssystems [22]

Die Beschreibung der Android System Architektur basiert auf der Android Dokumentation zum Thema Plattformarchitektur [22] sowie Levin [80].

Kernel:
Ein Kernel oder Betriebssystemkern ist der zentrale Teil eines Betriebssystems, er legt die Prozess und Datenstruktur des Systems fest. Er übernimmt die Kommunikation mit der Hardware und ist deshalb auch für die Speicherverwaltung, Prozessverwaltung (Scheduling) und die Verwaltung von Geräten und des Dateisystems verantwortlich. Der Kernel stellt dem auf ihm aufbauenden Betriebssystem Funktionen zum Zugriff auf diese Ressourcen bereit.
Bei dem von Android verwendeten Linux-Kernel handelt es sich um einen sogenannten modularen monolithischen Kernel, das bedeutet, dass alle Gerätetreiber sowie das Prozessscheduling Teil des Kernel selbst sind, und nicht wie bei einem Micro-Kernel in sogenannte Server ausgelagert werden. Da es sich bei dem Linux-Kernel um einen modularen Kernel handelt, können neue Gerätetreiber, Dateisysteme oder weitere Funktionen während des Betriebs nachgeladen werden.
Der Linux-Kernel wird von den verschiedenen Herstellern entsprechend angepasst, um die für ihre Hardware benötigten Treiber zu integrieren. Hierauf wird in Kapitel 5.1.5 weiter eingegangen.

Hardware Abstraction Layer (HAL):
Der HAL wird von den Herstellern der verschiedenen Geräte entwickelt und ermöglicht, ohne genaues Wissen über die Kerneltreiber, einen Zugriff auf die peripheren Geräte (z.B. Sensoren, Lautsprecher, Modem) des Smartphones.
Den Herstellern wird bei der Entwicklung ihrer Gerätetreiber und deren Funktionsumfang freie Hand gelassen. Um aber die Kompatibilität mit Android sicherzustellen, muss das Interface des HAL, das von außerhalb des Kernels aufgerufen wird, nach festen Standards von den Geräteherstellern implementiert werden (hierfür werden Geräte in Klassen eingeteilt, jede dieser Klassen besitzt eine eigene Referenzimplementierung für sein HAL [17]).
Der HAL selbst ermöglicht es Androiddiensten und Apps, auf die Gerätetreiber des Kernel zuzugreifen ohne selbst mit dem Kernel kommunizieren zu müssen. Er dient damit als Brücke zwischen dem Java- und dem nativen Teil von Android. Der HAL übernimmt die Übersetzung der generischen Hardwareanfragen auf die von den Geräteherstellern definierten hardwarespezifischen Kernel-Aufrufe.

Android Runtime (ART):
Android Applikationen wurden, historisch gesehen, auf der Dalvik Virtual Machine (DVM) ausgeführt. Seit Android 5.0 wird allerdings die Android Runtime (ART) für diese Aufgabe verwendet. Die Funktionsweise ist vergleichbar. Soll ein

Programm (im Folgenden immer App genannt) auf der VM ausgeführt werden, wird zunächst der Javacode durch eine entsprechende Toolchain (z.B. Jack) zu Java-Bytecode kompiliert und anschließend zu DEX-Bytecode übersetzt. Dieser DEX-Bytecode wird während der Installation der App, im Betriebssystem von der ART zu nativem und plattformabhängigen Maschinencode im ELF-Format kompiliert. Dies sorgt zwar für mehr Aufwand während der Installation, erhöht aber die Performance und verringert den Overhead einer App zur Laufzeit.
Aus Sicherheitsgründen werden die einzelnen Apps immer in einer Sandbox gestartet. Dies bedeutet, dass jede App vom Rest des Systems isoliert wird und nur über vorher definierte Schnittstellen mit dem System kommunizieren kann (siehe Kapitel 5.1.2). Zu diesem Zweck wird für jede App eine eigene Instanz der ART, mit isolierten Speicherbereichen, gestartet, dies trägt zur Isolation der einzelnen gleichzeitig laufenden Apps gegen unerlaubten Zugriff auf die App selbst oder von der App auf andere Teile des Systems bei.

Native C/C++ Libraries:
Viele der Android Systemkomponenten sind in C oder C++ geschrieben (z.B. HAL, ART) und benötigen die Unterstützung verschiedener nativ laufender Programmbibliotheken. Hierzu gehören unter anderem die libc (C Standard Sprachmittel), libm (mathematische Operationen) und OpenGL (3D-Grafik).
Darüber hinaus ermöglicht das Java Native Interface (JNI) die Verwendung dieser und anderer nativer Bibliotheken durch Apps, die auf der ART laufen. Hierbei werden vor allem rechenintensive Operationen durch den schnelleren, nativen Code durchgeführt (z.B. Crypto, mathematische Berechnungen, 3D-Grafiken erzeugen) oder JNI wird genutzt, um auf tiefer im System liegende Schnittstellen zuzugreifen (z.B. Aufruf des nativen Audioflinger-Interface).

Java API Framework:
Das Android Java API Framework wird hauptsächlich von Applikationsentwicklern verwendet. Diese greifen darüber auf die verschiedenen von Android angebotenen Dienste zu, dies beinhaltet unter anderem GUI-Elemente, eingebettete Browser, den Resourcenmanager, den Notificationmanager (Darstellung von Nachrichten in der Status-Bar) und die Content Provider (ermöglichen Zugriff auf verschiedene Datenquellen wie z.B. Kalender, Adressbuch, Bilder).

System Apps:
Als System Apps werden einige dem Android Open Source Project (AOSP) entstammende Apps bezeichnet, die die Grundfunktionalitäten des Smartphones zur Verfügung stellen, dazu gehören z.B. eMail-, SMS-, Kalender-, Kontakt-, Systemeinstellungen-, Telefon- sowie Browser-App. Die meisten dieser Apps können von dem Anwender durch alternative Anwendungen ersetzt werden, hierbei gibt es jedoch Ausnahmen wie z.B. die Telefon- sowie

Systemeinstellungen-App.
Die System Apps können auch von anderen App Entwicklern in ihrer Anwendung verwendet werden, um z.B. eine Website zu öffnen oder eine SMS zu versenden.

5.1.2 Android Sicherheitsmechanismen

In der Sicherheitsarchitektur von Android wurden seit der Einführung von Android viele Änderungen vorgenommen, daher wird hier nur die aktuelle Situation der Sicherheitsvorkehrungen besprochen. Die Android Sicherheitsmechanismen können grob in zwei Klassen aufgeteilt werden: Kernel- und App-Sicherheit.

Zu der Klasse der Kernelsicherheitsmechanismen [10] zählt das Sandboxing von Anwendungen auf dem Smartphone. Jede Anwendung wird mit einer eigenen Nutzer ID (uID) gestartet. Dies ermöglicht es dem Linux-Kernel, die Prozesse und ihre Speicherbereiche vollständig voneinander zu isolieren. Hierbei wird ausgenutzt, dass der Linux-Kernel aus dem Multiuser-OS Bereich stammt und ein seit vielen Jahren bewährtes System zur Isolierung der Ressourcen einzelner Nutzer implementiert. Dadurch, dass dieser Sicherheitsmechanismus auf Kernelebene ansetzt, werden von ihm nicht nur Apps, die in der ART laufen, erfasst, sondern auch sämtliche native Anwendungen (das beinhaltet u.a. auch Betriebssystemkomponenten).
Neben der kernelseitigen Isolierung der Apps über die uID, wird seit Android 4.3 die Sandbox ebenfalls durch SELinux (Security-Enhanced) weiter überwacht [11]. Der Ansatz hierbei ist Mandatory Access Control, wobei die einzelnen Teile des Systems in Domains aufgeteilt werden, die nur für diese Domain spezifische Operationen durchführen dürfen.
Seit Android 6.0 wird "Verified Boot" standardmäßig verwendet, hierbei verifiziert jede Stufe des Bootvorgangs die darauffolgende Stufe gegen eine vorhandene Signatur [12]. Solle bei diesem Check festgestellt werden, dass eine Stufe des Bootvorgangs kompromittiert wurde, wird der Bootvorgang abgebrochen und der Nutzer darüber in Kenntnis gesetzt.

Zu der Klasse der App-Sicherheitsmechanismen zählt das Permission Model (Berechtigungsmodell) für Apps. Android sichert manche seine sicherheits- und privatsphärenkritischen APIs mit diesen Permissions ab, wodurch es einem Nutzer ermöglicht wird zu entscheiden, ob eine App Zugang zu diesen Schnittstellen erhalten kann oder nicht [8]. Seit Android 6.0 werden Permissions nicht mehr während der Installation einer App vergeben, sondern zur Laufzeit. So kann der Nutzer auch nach der Installation einer App die ihr bereits eingeräumten Rechte wieder entziehen oder weitere Rechte, auch z.B. nur für begrenzte Zeit, verfüg-

bar machen.
Android unterscheidet zwischen normalen und gefährlichen Berechtigungen, wobei alle privatsphärenkritischen Berechtigungen (z.B. Zugriff auf GPS, Telefon, Kamera, Mikrofon, usw.) als gefährliche App Permissions angesehen werden [30]. Im Gegensatz dazu werden die App Permissions für den Zugang zu vielen anderen Schnittstellen wie z.B. den Zugang zu den Netzwerkschnittstellen als normale Berechtigungen eingestuft [29]. Der Unterschied zwischen diesen beiden Gruppen ist, dass der Nutzer explizit um Erlaubnis gefragt wird, wenn eine gefährliche Berechtigung zum ersten Mal genehmigt werden soll. Bei normalen Berechtigungen wird diese Erlaubnis ohne das Zutun des Nutzers vom System erteilt. Dies bedeutet auch, dass die Verwendung der mit normalen App Permission abgesicherten Schnittstellen durch Apps nicht vom Nutzer kontrolliert werden kann [31]. Dieser kann nur den Zugriff auf gefährliche Berechtigungen dynamisch festlegen.
Durch den Binder-IPC (Inter Process Communication), ein Android Kernel-Modul, wird den einzelnen Apps (die in einer Sandbox ausgeführt werden) ein sicherer Kommunikationskanal untereinander und mit dem Betriebssystem zur Verfügung gestellt [33]. Hierbei bietet der Binder-IPC Service eine eindeutige Attribuierung der Nachrichten sowie Mechanismen zur Authentisierung und Verifizierung der Nachricht.
Andoid Apps müssen zur Installation im Betriebssystem signiert sein [9]. Die Zertifikate, die hierfür verwendet werden, sind nicht an eine zentrale Zertifikatsautorität (Certificate Authority (CA)) gebunden. Das bedeutet, dass jeder App Entwickler seine App auch mit einem selbst erstellten Zertifikat signieren kann. Apps, die mit demselben Zertifikat signiert wurden, erhalten vom Android System die Möglichkeit, auf gemeinsam genutzten Speicher zuzugreifen. Die Signierung der App wird auch konsultiert, wenn ein Update für die entsprechende App in das System eingespielt wird. Das Update wird vom System nur als legitimes Update für die App akzeptiert, wenn es mit demselben Schlüssel wie die bereits installierte App signiert wurde. Seit Android 7.0 wird ein neues Signierungsschema verwendet, das, im Gegensatz zum Vorgänger, nicht nur den Code signiert, sondern sämtliche Ressourcen, die von einer App mitgeliefert werden. Das System ist abwärtskompatibel, da bei neu signierten Apps immer beide Typen von Signatur berechnet werden.
Seit Android 7.0 wird durch Google eine nicht veränderbare, systemweit verfügbare, CA in Android integriert [8, siehe Punkt: Certificate authorities]. Bevor dieses System implementiert wurde, konnten Gerätehersteller die Einträge dieser CA modifizieren oder weitere Einträge hinzufügen. Um in die CA eingetragen zu werden, muss der "Mozilla CA Inclusion Process" durchlaufen werden. Nach einer erfolgreichen Durchführung kann ein Antrag auf Aufnahme in die Standard Android-CA gestellt werden.

5.1.3 Audioaufnahme

In diesem Abschnitt sollen die von Android zur Verfügung gestellten Audioaufnahmefunktionalitäten sowie die allgemeinen Voraussetzungen für eine App, die diese zur Audioaufnahme verwenden möchten, besprochen werden.
Erste Voraussetzung für den Zugriff auf die vom Mikrofon stammenden Audiodaten ist die App Permission "RECORD_AUDIO" [30], die durch die App deklariert werden muss. Beim ersten Versuch, Gebrauch von dieser Berechtigung zu machen, wird dem Nutzer eine Aufforderung angezeigt, diesen Vorgang zu bestätigen. Wird dies bestätigt, kann die App auf das Mikrofon zugreifen und Audiodaten aufnehmen, bis diese Berechtigung vom Nutzer widerrufen wird.
Diese App Permission gilt nicht für die Aufzeichnung von Audiodaten, die während eines Telefongesprächs anfallen. Für den Zugriff auf den Audiostream des Mikrofons während eines Gesprächs über das Mobilfunknetz wird die spezielle "CAPTURE_AUDIO_OUTPUT" App Permission benötigt. Diese App Permission ist allerdings nicht für Drittanbietern verfügbar, das bedeutet im Endeffekt, dass nur Apps, die direkt aus dem Android Open Source Project stammen oder vom Hersteller des SP hinzugefügt wurden, diese App Permission nutzen können.
Für die Aufnahme von Audiodaten bietet Android die AudioRecorder-Klasse [32]. Diese bietet verschiedene Möglichkeiten zur Konfiguration der Aufnahmequalität, des Aufnahmeformats und der Audioquelle. Die Audioaufnahme kann auf Basis dieser Klasse auch als Service genutzt werden, der in der Lage ist, im Hintergrund, während andere Apps im Vordergrund sind, oder auch während das SP gesperrt und der Bildschirm ausgeschaltet ist, aufzunehmen.
Bei Versuchen im Zuge dieser Arbeit konnte keine öffentlich bekannte Methode gefunden werden, die es ermöglichen würde, Audioaufnahmen ohne die entsprechende App Permission anzufertigen. Neben der herkömmlichen Methode, Audioaufnahmen mittels des Android AudioRecorders anzufertigen, wurde auch die Verwendung verschiedener nativer Bibliotheken geprüft.
Weiter ist die Audioaufnahmefähigkeit über Android dadurch begrenzt, dass maximal eine App gleichzeitig auf das Mikrofon zugreifen kann. Versucht eine App dennoch auf das Mikrofon zuzugreifen, können verschiedene Effekte beobachtet werden. Gerade bei älteren SP Modellen kann oft eine Aufnahme gestartet werden, diese enthält dann aber keinerlei Audioinformation. Bei neueren Android Geräten wird hingegen öfters eine Fehlermeldung angezeigt, die indiziert, dass das Mikrofon gerade nicht verfügbar ist. Beide Verhalten konnten in einem Versuch beobachtet werden (Nexus S (2010) und Nexus 5X (2015)).
Für dieses Verhalten gibt es jedoch Ausnahmen. Diese beziehen sich aber auf die gesonderte Möglichkeit, den Audiostream eines Telefongesprächs aufzuzeichnen. Wie bereits erwähnt, wird für diese Audioquelle aber eine spezielle App Permission benötigt, die für Apps von Drittanbietern nicht verfügbar ist.
Neben dem Hauptmikrofon ("DEFAULT" oder "MIC" Audioquelle [25]) gibt es

an modernen SPs nocht weitere Mikrofone. Einige davon werden für Funktionen wie Rausch- oder Hallunterdrückung verwendet und sind über die Android Funktionalitäten nicht auf normalem Wege zu erreichen. Es gibt jedoch in vielen SP ein dediziertes Mikrofon, das für die Audioaufnahmen während laufender Videoaufnahmen verwendet wird. Dieses Mikrofon ist über die Audioquelle "CAMCORDER" zu erreichen und wird oft nicht verwendet, da das leistungsfähigere Hauptmikrofon bevorzugt wird. In den Versuchen zu diesem Kapitel wurde festgestellt, dass es auf dem Nexus S möglich ist, dieses Mikrofon auch während Telefongesprächen und weiteren Verwendungen des Hauptmikrofons zur Aufnahme von Audiodaten aus dem Umfeld des SP zu verwenden. Das ebenfalls getestete Nexus 5X hingegen erlaubte eine Audioaufnahme über dieses Mikrofon nur während einer Benutzung des Hauptmikrofons durch eine App, nicht jedoch während eines Telefongesprächs.
Im Audioaufnahmekontext genießen System-Apps besondere Rechte. So kann die für Telefonie zuständige System-App im Falle eines Anrufs die Kontrolle über das Mikrofon übernehmen. Sollte eine andere App das Mikrofon gerade verwenden, verliert diese den Zugang zu den Audioinformationen. Dieses Verhalten ist jedoch ausschließlich für System-Apps reserviert. Apps, die nicht zum System gehören, haben keine Möglichkeit, das Mikrofon ohne die Zustimmung der App, die das Mikrofon gerade nutzt, zu übernehmen.

5.1.4 Schnittstellen

Da Android ein HLOS ist, bietet es eine extrem große Anzahl an Schnittstellen. Im Vergleich zum BB muss Android mit einer wesentlich größeren Vielfalt an Diensten kommunizieren sowie sämtliche anwenderorientierte Schnittstellen des SP implementieren. Für die Betrachtung in Bezug auf die Audio-Bypass Bedrohung sind aber hauptsächlich die Shared Memory Device (SMD) Schnittstelle, über die mit dem BB kommuniziert wird, sowie die Fähigkeit, den SLIMbus zu routen, von Interesse. Die weiteren Schnittstellen, die während der Bedrohungsanalyse in Betracht gezogen werden, werden in Kapitel 5.2.1 während der Beschreibung der Angriffsvektoren oder bei der Analyse der für sie spezifischen Angriffe (z.B. Nutzerschnittstellen wie der Google Play Store oder die Installation von Apps aus alternativen Quellen) beschrieben.
Zur Kommunikation über das SMD mit dem BB wird der Android Radio Interface Layer (siehe Abbildung 5.2) verwendet. Dies beinhaltet das Übertragen von Steuersignalen (diese werden zur Kommunikation zwischen dem Applikationsprozessor (AP) und BB verwendet, um z.B. den Empfang eines Anrufs zu signalisieren, in diesem Fall muss das Handy die grafische Benutzeroberfläche entsprechend anpassen und den SLIMbus routen), die Übertragung der IP-Pakete (diese werden vom Linux Kernel Netzwerkstack empfangen und über das SMD an den BB weitergereicht) sowie Datenverbindungen für weitere vom

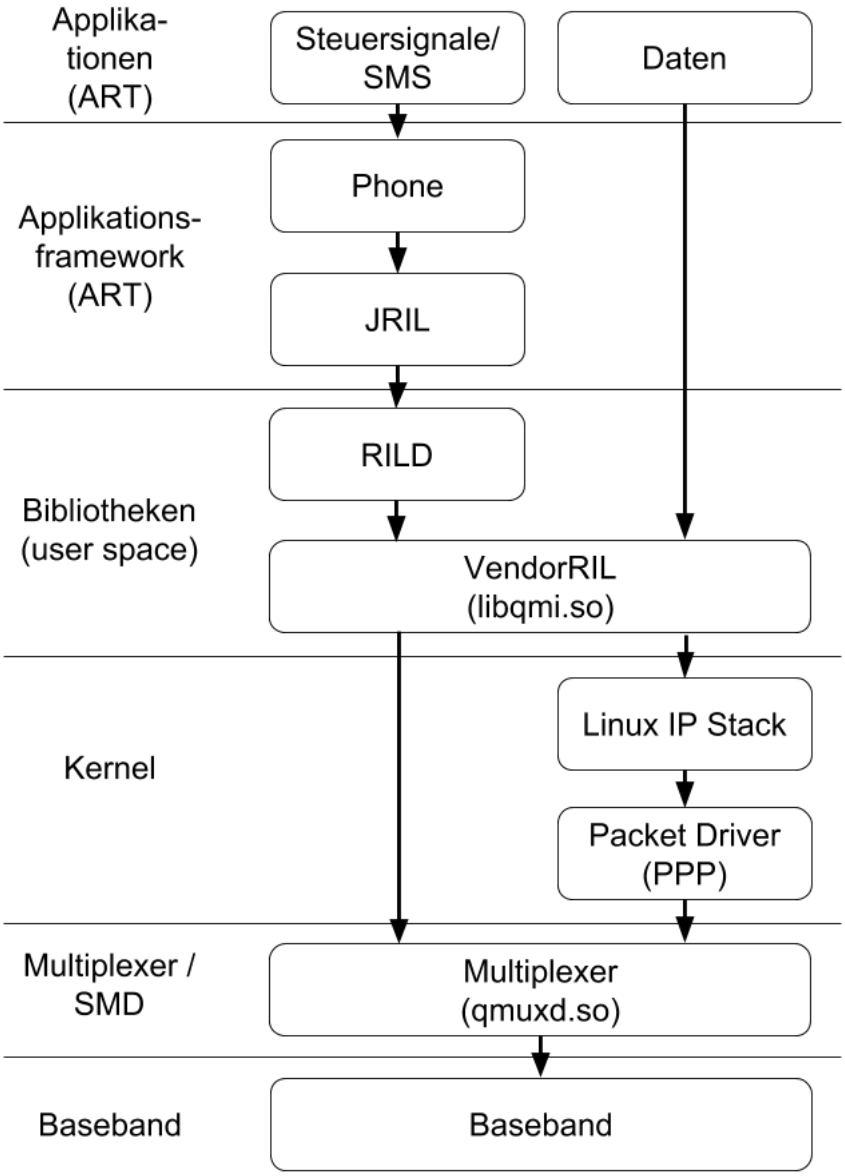

Abbildung 5.2: Android Radio Interface Layer

BB Komplex gebotene Dienste (z.B. GPS).
Im Folgenden wird der Aufbau des Android Telephony Stacks, der die API zur Kommunikation mit dem BB bietet, und sein Zusammenspiel mit dem SMD erklärt, dies wird auf Basis der von Qualcomm (QC) eingesetzten Techniken durchgeführt. Diese Architektur (Kommunikation über SMD) wird nicht nur von QC eingesetzt, sondern auch von anderen Produzenten, bei denen der BB in das SoC integriert ist (z.B. Samsung [93]).

Applikationen:
Die Apps in der Applikationsebene, die das RIL nutzt, sind hauptsächlich System-Apps (z.B. Sms-, Telefonie-, Simtoolkit-App), die mit dem BB kommunizieren müssen. Diese wenden sich mittels JNI an die nächste tiefere Schicht, welche die benötigten Funktionen zur Verfügung stellt.
Alle Anwendungen, die mobiles Internet benötigen, kommunizieren indirekt über ihre Socketverbindung ebenfalls mit dem RIL.

Applikationsframework:
Das Applikationsframework bietet mit der Phone-Klasse des Telephony-Stacks [38] eine vollständige Abbildung aller durch das BB an das SP zur Verfügung gestellten Funktionen. Diese verwenden die in der JRIL (siehe RIL.java in [38]) implementierten Methoden zum Versand und Empfang von RIL-Nachrichten über einen vom RIL-Daemon (RILD) kontrollierten Socket. Hierbei werden statische Werte [36] für die Kodierung von Steuersignalen verwendet.
An dieser Stelle ist darauf hinzuweisen, dass der Versand einer SMS über solche Steuersignale gehandhabt wird. Hierbei wird der Text sowie der Empfänger der SMS in ein Steuersignal eingebettet und über den auch für reine Steuersignale verwendeten Kanal versandt. Im Gegensatz dazu stehen Telefonanrufe. Diese werden durch Steuersignale (in denen die Informationen über den Anruf enthalten sind) angekündigt, die eigentliche Audioinformation des Anrufs wird jedoch über den SLIMbus geleitet, mit dem wir uns im späteren Verlauf dieses Kapitels befassen werden.

Bibliotheken:
Der Radio Interface Layer Daemon (RILD) ist für die Verbindung von androidseitigem Code mit den von den SoC Herstellern stammenden Bibliotheken verantwortlich. Diese stellen die Treiber für die Kommunikationskanäle zwischen dem AP und BB zur Verfügung [37]. Der RILD wird während des Android Systemstarts initialisiert und verbindet sich auf der einen Seite mit der VendorRIL (im Fall von QC ist das die libqmi.so in /vendor/lib64/), auf der anderen Seite öffnet er den Linux-Socket, der als Kommunikationskanal zur JRIL verwendet wird. Er übernimmt außerdem die Übersetzung der statischen Werte, die von der JRIL verwendet werden, um Steuersignale in die von der 3GPP definierten AT Kommandos zu kodieren [3]. Diese basieren auf dem schon seit den 80ern eingesetzten Hayes- oder AT-Befehlssatz zur Kommunikation mit Modems.
Die proprietäre VendorRIL ist für die Umsetzung der AT Kommandos in das herstellereigene, meist proprietäre, Protokoll verantwortlich (im Fall von QC ist dies QMI [53]). Steuersignale werden, über einen Socket, direkt an den Multiplexer weitergereicht, der sie auf das SMD schreibt.
Auch werden die für Datenverbindung notwendigen Sockets von der Ven-

dorRIL im System registriert und ermöglichen so eine Verbindung über diese Sockets zum Linux Netzwerkstack. Gleichzeitig alloziert die VendorRIL einen Speicherbereich im SMD für die Daten, die über diese Sockets gesendet werden. Der Vorgang, bei dem die Sockets im System registriert werden, kann in der von dem AOSP zur Verfügung gestellten Referenz VendorRIL (siehe reference-ril.c in [35]) nachvollzogen werden, allerdings nicht das Allozieren des SMD Speicherbereichs, da es sich hierbei um einen von der Hardwareplattform abhängigen Vorgang handelt.

Kernel:
Die über die von der VendorRIL geöffneten Sockets übertragenen Daten werden dem Kernel zugestellt und mittels des IP-Stacks für den Versand in das Internet vorbereitet. Anschließend werden sie von dem sogenannten Point to Point (PPP) Protokoll für den Transport über eine nicht IP-basierende Verbindung (SMD) verpackt. Anschließend werden sie, wieder mittels eines Sockets, an die in qmi_config.xml (in system/etc/data zu finden) konfigurierten Endpunkte übertragen. Im Fall von QC ist das der QMI Multiplexer Daemon (qmuxd.so in /system/bin/).

Multiplexer:
Der Multiplexer verwaltet die direkten Kommunikationskanäle zum BB, im Fall von QC verwaltet qmuxd.so das in QC SoCs verwendete SMD. Da über das SMD sämtliche Dienste des BB abgehandelt werden, wird ein Multiplexer verwendet, der es verschiedenen Subsystemen (GPS, SMS, Anrufinfo, Netzwerkstatus) und ihren proprietären Bibliotheken (vergleichbar mit libqmi.so) ermöglicht, gleichzeitig Steuersignale zu versenden [53]. Hierfür wird ein Queueing System verwendet, das die Steuersignale seriell auf das SMD schreibt. Netzwerkverkehr ist von diesem Queueing nicht betroffen, sondern besitzt seinen eigenen Kanal im SMD.

Die zweite Schnittstelle, die in Bezug auf den Audio-Bypass betrachtet wird, ist das Audiorouting über den SLIMbus. Wie schon in Kapitel 4.1.3 und 4.2.3 beschrieben, wird der SLIMbus heutzutage nicht mehr durch den BB verwaltet, sondern durch den AP. Das gesamte Audio-Framework von Android ist ausführlich dokumentiert [5].
Auf der Applikationsframework-Ebene bietet die Android Media API [21, 20] alle für Apps benötigten Funktionen (z.B. Aufnehmen und Abspielen von Audiodaten, Konfiguration der Audioqualität, Digital Rights Management). Apps können über die AudioRouting-Klasse (siehe AudioRouting.java in [20]) das aktuelle Audiorouting beeinflussen, aber nicht gegen den Wunsch des Nutzers ändern. Die Android Media API kommuniziert mittels JNI mit den nativen Audiobibliotheken, die wiederum direkt mit der von dem Hersteller stammenden

Audio-HAL verbunden sind.
Das native Android Audioframework, der sogenannte Audioflinger [23], ist für die Umsetzung der sogenannten Audio Policies verantwortlich. Dies sind die vom Hersteller vordefinierten Audiopfade für verschiedene Anwendungsfälle (z.B. Kopfhörer eingesteckt oder Videoaufnahme), die von dem System unterstützt werden. Die verschiedenen Werte sind hochgradig plattformabhängig und werden in der audio_policy_configuration.xml definiert (z.B. Nexus 5X audio_policy_configuration.xml [24]). Dies stellt die Basis der möglichen Audiokonfigurationen des Systems dar.
Meist basiert die Audio-HAL auf dem von AOSP empfohlenen [5] tinyALSA [19] sowie proprietären Audio-Codecs des Herstellers. TinyALSA ist eine Open Source Bibliothek, die mit der im Kernel verwendeten Advanced Linux Sound Architecture (ALSA) [99] kommuniziert. Dieses Modul ist letztendlich dafür verantwortlich, den SLIMbus zu routen und die entsprechenden Audiodaten auf den Bus zu schreiben.

5.1.5 Android Herstellermodifikationen

Wie bereits erwähnt, muss der Linux-Kernel von den verschiedenen Geräteherstellern auf die von ihnen verwendete Hardwareplattform angepasst werden. Hierbei werden vor allem entsprechende Treiber für das SoC des Geräts sowie weitere Peripheriegeräte in den Kernel integriert. Da der Linux Code unter der GPL-Lizenz verteilt wird, müssen die Hersteller diese veränderten Linux-Kernel wiederum als Quelltext veröffentlichen. Dies wird entweder zentral in einem von Google verwalteten Repository [34] durchgeführt oder auf von den Herstellern spezifisch zur Verfügung gestellten Websites [72, 82, 107].
Neben den Herstellern von Smartphones beteiligen sich auch die Hersteller der in SP verwendeten SoCs an der Kernelentwicklung. So stellt z.B. Qualcomm die Plattform CodeAurora.org zur projektübergreifenden Arbeit am Linux-Kernel für ihre SoCs zur Verfügung. Dort können Firmen oder Privatpersonen, die ein Qualcomm Produkt verwenden, einen Kernel beziehen (oder ihren eigenen Code veröffentlichen), der bereits die meisten Änderungen für die spezifische Hardware enthält [100].
Neben der Integration der Gerätetreiber in den Kernel muss auch die HAL-Schicht von den Herstellern entwickelt werden. Sie definiert den späteren Satz an Fähigkeiten, die jedes angeschlossene Gerät zur Verfügung stellt [14]. Diese Software ist nicht Teil des Linux-Kernel und unterliegt den entsprechenden Lizenzen der Hersteller. Dies bedeutet, dass der Code der HAL-Komponenten nicht im Zuge der GPL-Lizenz veröffentlicht werden muss. Es kann nicht garantiert werden, dass diese Softwarekomponenten ihr spezifiziertes Verhalten auch konsistent durchsetzten und nicht etwa über weitere, undokumentierte, Funktionalitäten verfügt.

Neben diesen für die Funktionalität des SP unerlässlichen Modifikationen, werden von den Herstellern aber auch weitere Änderungen vorgenommen. In der Regel werden SP von den Herstellern schon vor Auslieferung mit einem Satz von Anwendungen bespielt. Dazu gehören die von Google für den Zugang zum Google-Playstore-Ökosystem (Play-Store und Google Mobile Services) vorausgesetzten Google-eigenen Anwendungen (u.a. Chrome, GMail, Google Search und Maps) sowie Anwendungen, die der Hersteller selbst dem Nutzer zur Verfügung stellen möchte. In den letzten Jahren hat es sich bei vielen Herstellern durchgesetzt, dem Nutzer einen Satz an Anwendungen, die sich nicht deinstallieren lassen, mitzugeben. Diese Anwendungen werden allgemein als "Bloatware" [85] bezeichnet, da der Nutzer selbst keine vorgesehene Möglichkeit besitzt diese zu löschen, obwohl oft der Mehrwert für den Nutzer durch diese Apps nur minimal bis gar nicht vorhanden ist.

5.1.6 Android Versionen

Historisch gesehen wurde die erste Version, Alpha 1.0, von Android am 23.09.2008 veröffentlicht. Seitdem wird die Entwicklung von Google und anderen im Android Open Source Project (AOSP) vorangetrieben. Die aktuelle Versionsverteilung ist in Tabelle 5.3 zu sehen.
Google unterstützt Androidversionen so lange, wie sie das dazu korrespondierende Nexus-Gerät unterstützten. Hieraus lässt sich eine etwaige Supportdauer von 2 Jahren für Systemupdates und eine, angekündigte, 3-jährige Supportzeit für Sicherheitsupdates ableiten. Aktuell werden nur noch die Versionen 6.x und 7.x von Google offiziell unterstützt. Das bedeutet, dass alle Android-Geräte mit einer Betriebssystemversion kleiner als 6.0 offiziell keine Sicherheitsupdates von Google mehr erhalten [64]. Hier besteht allerdings auch eine Diskrepanz zwischen Google und dem AOSP. Im AOSP werden prinzipiell die letzten 3 Major-Releases von Android mit Sicherheitsupdates versorgt, das bedeutet, dass Android 5 derzeit noch mit Sicherheitsupdates vom AOSP versorgt wird. Firmen erhalten hierfür aber keine Integrationshilfe von Google [13, Siehe Punkt: Notifying partners].
Die Grafik 5.4 zeigt die Fragmentierung der weltweit eingesetzten Android Versionen auf. Die Gründe für die große Anzahl an eingesetzter unterschiedlicher Versionen und die große Anzahl an Geräten, die veraltete und nicht mehr unterstütze Versionen von Android einsetzten, sind vielzählig.
Die eben erwähnte Supportpolitik von Google sowie die Änderungen, die die Hersteller der Smartphones am Betriebssystem vornehmen, spielen hierbei eine große Rolle. Je mehr ein Hersteller den Code von Android auf seine Plattform anpasst, desto mehr müssen die Updates aus den Softwarequellen von Google für das entsprechende Gerät angepasst werden. Viele Hersteller sind anscheinend nicht in der Lage, mehr als nur die besten ihrer Geräte mit

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.8%
4.1.x	Jelly Bean	16	3.2%
4.2.x		17	4.6%
4.3		18	1.3%
4.4	KitKat	19	18.8%
5.0	Lollipop	21	8.7%
5.1		22	23.3%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	6.6%
7.1		25	0.5%

Data collected during a 7-day period ending on May 2, 2017.
Any versions with less than 0.1% distribution are not shown.

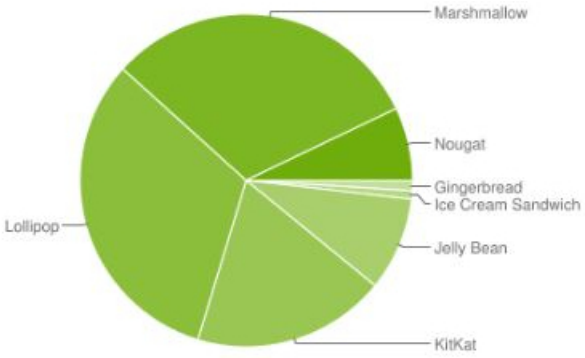

Abbildung 5.3: Übersicht über die Versionsverteilung [28]

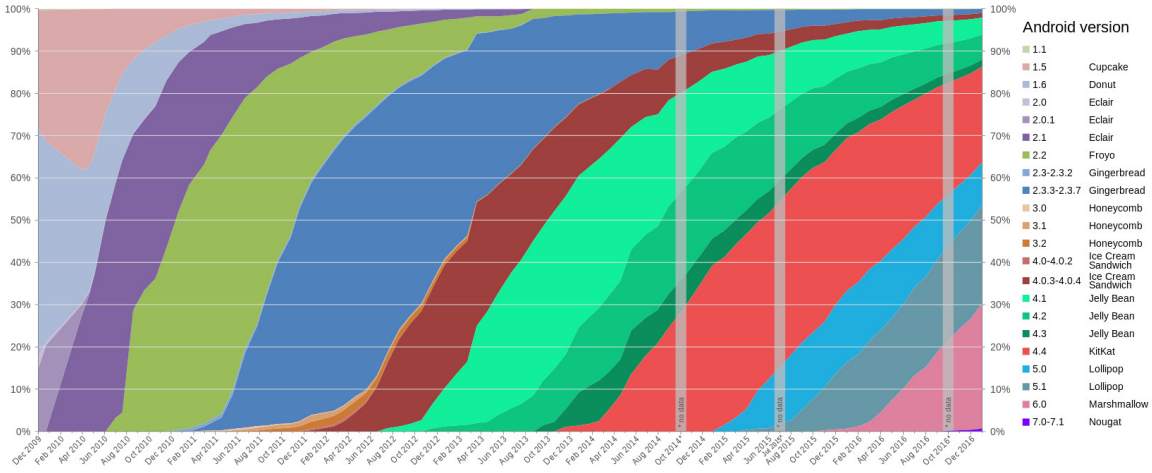

Abbildung 5.4: Übersicht über die Versionsfragmentierung [126]

diesen Updates zu versorgen. Dies wird weiter dadurch verkompliziert, dass Android im Laufe seiner Entwicklung verschiedene interne Strukturen gerade im Bereich der Hardwareanbindung verändert hat, was dazu führt, dass viele alte Hardwareplattformen schlicht nicht mehr kompatibel sind mit der neuen Betriebssystemstruktur von Android.
Android bietet seit Version 5.1 einen Sicherheitsupdatemechanismus. Dieser ermöglicht es, das System zu patchen ohne ein Versionsupdate des gesamten Betriebssystems durchzuführen. Diese Sicherheitsupdates werden von dem AOSP entwickelt und monatlich zur Verfügung gestellt [15]. Google passt diese Updates dann auf die von ihnen unterstützten Geräte (Nexus und Pixel) an. Anderen Herstellern von Android Smartphones werden dieselben Möglichkeiten geboten, diese monatlichen Updates auf ihre Geräte zu portieren. Leider ist die Akzeptanz als auch die Durchsetzung dieser monatlichen Updates bei Herstellern jenseits von Google nur sehr gering verbreitet [108]. Das bedeutet, dass viele Geräte Sicherheitsupdates nur sehr spät oder nie erhalten.

5.1.7 Google Now

Bei Google Now handelt es sich um einen von Google für die Android Plattform angebotenen Sprachassistenten, der seit Android 4.1 verfügbar ist. Dies ist eine Schnittstelle, die es ermöglicht, über Sprachbefehle das Smartphone zu kontrollieren, um z.B. eine Nachricht zu verfassen, den Wecker zu stellen oder Informationen aus dem Internet abzurufen.
Dieser Dienst kann in zwei verschiedenen Modi betrieben werden. Entweder wird er über die Verwendung eines Buttons aufgerufen oder aber er befindet sich in einem Sprachaktivierungsmodus, hierbei kann die Funktionalität über den Sprachbefehl "OK Google" oder "Hey Google" erreicht werden. Befindet sich der Dienst in diesem automatischen, sprachaktivierten Modus, nimmt das Smartphone die gesamte Zeit seine Umgebungsgeräusche auf und versucht, mittels lokaler Stimmerkennung die Aktivierungsbefehle zu erkennen. Hierbei werden einige Sekunden der aktuellen Audioaufnahmen vorgehalten, und, sollte der Aktivierungsbefehl gefunden werden, zusammen mit den folgenden Aufnahmen zur genaueren Analyse über das Internet an den Google-Spracherkennungsdienst übertragen. Dieses Verhalten kann nachempfunden werden, da Google die so aufgenommenen Audiodaten dem Anwender zur Verfügung stellt [65]. In diesen Aufnahmen lassen sich oft noch Geräusche, die kurz vor dem Nennen des Aktivierungsbefehls aufgenommen wurden, hören. Befindet sich der Dienst in dem durch Druck eines Buttons manuell aktivierten Modus, wird die Aufnahme anscheinend, basierend auf den von Google zur Verfügung gestellten Aufnahmen, erst nach Drücken des Buttons gestartet.
Alles in allem ist diese Funktionalität aus Sicht der Privatsphäre als auch der

des allgemeinen Datenschutzes äußerst problematisch und es kann Nutzern nur empfohlen werden, die Funktionalität vollständig zu deaktivieren.
Im weiteren Verlauf dieser Arbeit wird dieses Thema nicht weiter behandelt, da eine Risikoeinschätzung zu keinen weiteren Ergebnissen führen würde. Abschließend kann aber darauf hingewiesen werden, dass die Sicherheit sämtlicher Android SP relevanter Themen prinzipiell vom Vertrauen zu dem OS, das von dem AOSP (indirekt Google) und den Herstellern, die dieses auf ihre SP anpassen, abhängt.

5.2 Schwachstellen- und Bedrohungsanalyse

In diesem Kapitel wird die Schwachstellen- und Bedrohungsanalyse des Android OS in Bezug auf die Audio-Bypass Bedrohung vorgenommen. Im Vergleich zum Baseband-Prozessor (BB) Komplex existiert eine wesentlich größere Angriffsfläche. Dies liegt darin begründet, dass es sich bei Android um ein HLOS handelt, das mit dem Nutzer interagieren können muss. Es muss in der Lage sein, Anwendungen zu installieren und es muss die Ausführung dieser Anwendungen unterstützten sowie Schnittstellen zur Erweiterung der vom OS gebotenen Funktionalität anbieten. Diese Vielseitigkeit sorgt zwar nicht unbedingt für eine höhere Komplexität im Vergleich zu dem auf dem BB laufenden Softwarestack, vergrößert aber die Menge an Funktionalitäten, Schnittstellen und damit Quellcode, der potentiell angreifbar ist.
Es wird versucht, wie bereits in Kapitel 4.2 bei der Schwachstellen- und Bedrohungsanalyse des BB Komplexes, die vielen verschiedenen möglichen Angriffe in übergeordnete Kategorien einzuteilen und diese Szenarien zu bewerten. Bei der Bewertung selbst wird versucht, auf Basis von konkreten Fällen und Schwachstellen zu arbeiten. In Fällen, wo dies nicht möglich ist, wird der Autor versuchen, nach bestem Gewissen die Situation entsprechend der vorhandenen Informationen zu bewerten. Im Vergleich zum BB Softwarestack ist der größte Teil von Android Open Source, was die Analyse wesentlich vereinfacht und vor allem viele unabhängige Sicherheitsforscher anzieht, die den Quellcode nach Fehlern durchsuchen [6].
Es ist darauf hinzuweisen, dass diese Schwachstellenanalyse bei weitem nicht alle möglichen Schwachstellen und Angriffe in Bezug auf die Audio-Bypass Bedrohung betrachten kann. Eine erschöpfende Betrachtung aller möglichen Angriffsvektoren und Schwachstellen würde die Grenzen dieser Arbeit überschreiten. Es wird daher versucht, eine Auswahl von besonders relevanten Angriffsvektoren und möglichen Schwachstellen vorzustellen und zu analysieren.
Die Analyse beginnt mit der Übersicht über die verschiedenen Angriffsvektoren und behandelt anschließend den Remote Code Execution (RCE) Angriff auf Android. Weiterführend werden ein Angriff auf den RIL sowie ein Angriff

zur Rechteausweitung (Privilege Escalation, PE) behandelt. Abgeschlossen wird die Analyse durch die Betrachtung von Apps und ihrem Verhalten im System bezüglich der vorhandenen Audioquellen (Umfang und Einhaltung von Permissions).

Das Data Flow Diagrams (DFD) für Android und die in Kapitel 5.1.4 sowie in Kapitel 5.2.1 vorgestellten Schnittstellen, ist auf Grund der Größe im Anhang 9.1.2 zu finden. Es verdeutlicht die hohe Komplexität der Schnittstellen, die von Android verwaltet werden müssen und gibt einen Eindruck von der vorhandenen Angriffsfläche.

5.2.1 Angriffsvektoren

Aufgrund der unüberschaubar großen Menge an Angriffsvektoren muss darauf hingewiesen werden, dass diese Betrachtung nicht erschöpfend ist. Daher werden hier die verschiedenen im Zuge der Schwachstellenanalyse für diese Arbeit recherchierten Angriffsvektoren mit der größten Relevanz für den Audio-Bypass Angriff wiedergegeben. Dies dient als eine Übersicht über die im Rahmen der Schwachstellenanalyse von Android geleisteten Recherche. Gerade im Bereich der Hardwareschnittstellen wurde nur die Verbindung mit dem BB betrachtet, weitere Schnittstellen wie z.B. USB, WLAN, JTAG, NFC oder Bluetooth wurden nicht in die Betrachtung mit einbezogen.
Um die Angriffsvektoren zu gruppieren, werden sie über den verwendeten Übertragungsweg eingeteilt: Steuersignale, die vom BB und dem Applikationsprozessor (AP) gesandt werden (hierzu zählen auch Kurznachrichtendienste), Datenverkehr über die IP-basierenden Netzwerke sowie Nutzerinteraktionen (z.B. Installation einer App).

Angriffsvektoren, die auf den Steuersignalen, die zwischen BB und AP ausgetauscht werden, basieren, richten sich in der Regel gegen den RIL. Hierbei können sie in zwei Gruppen aufgeteilt werden: Angriffe auf die Kurznachrichtenprotokolle (SMS, MMS) sowie auf die VendorRIL. Die Ergebnisse von Mulliner et al. [91, 90] zeigen, dass ein Angriff über simple Textnachrichten (SMS), auf Grund der geringen Datenmenge und des sehr einfach aufgebauten Protokolls, wahrscheinlich maximal zu einem Denial of Service (DoS) Angriff innerhalb des RIL führen könnte.
Allerdings existieren inzwischen komplexere Kurznachrichtenprotokolle als textbasierende SMS. So können binäre SMS zur Übertragung von Konfigurationsparametern oder Nachrichten über den Multimedia Messaging Service (MMS) zur Übertragung von Medieninhalten verwendet werden. Dies eröffnet mehrere Angriffswege. Zum einen können die Systeme attackiert werden, die für die Verarbeitung solcher Nachrichten verantwortlich sind, zum anderen ist

es aber auch möglich, Systeme, die zur Verarbeitung des Nachrichteninhalts benötigt werden, anzugreifen.
So können binären SMS zur Codierung und Übertragung weiterer Protokolle verwendet werden. Ein Beispiel für einen Angriff über diesen Weg findet sich in Court et al. [119]. Bei diesem Angriff werden die Konfigurationsparameter für die Netzeinwahl mittels einer bösartig modifizierten Nachricht, ohne weitere Nutzerzustimmung, geändert.
Als Beispiel für einen Angriff, der MMS zur Übertragung verwendet, kann der unter dem Namen Stagefright bekannt gewordene Angriff [62] genannt werden. Hierbei wird eigentlich eine Schwachstelle im Mediaframework von Android ausgenutzt. Dabei kommt es den Angreifern zugunsten, dass die Standardeinstellung von Android einen über MMS empfangen Medieninhalt direkt, ohne weiteres Zutun des Nutzers, zum Mediaframework überträgt und somit den schädlichen Code ausführt.

Hat der Angreifer bereits Kontrolle über den BB erhalten und ist in der Lage, arbiträre Pakete auf das Shared Memory Device zu schreiben, kann dies für einen Angriff auf die VendorRIL verwendet werden. Hierbei kann entweder versucht werden, das eigentliche Verhalten der VendorRIL zu nutzen oder diese zu manipulieren [105], um das System weiter zu attackieren oder es kann versucht werden, einen Fehler in der VendorRIL zu nutzen, um Kontrolle über die Codeausführung zu erhalten [122].

Die zweite Gruppe von Angriffsvektoren bezieht sich auf den von Android oder den installierten Apps empfangenen IP-basierten Datenverkehr. Hierbei handelt es sich meistens um Angriffe auf verschiedene Apps, die vom Nutzer benutzt werden, um Internetinhalte abzurufen (z.B. Browser oder Messaging-Dienste) sowie die vom System gebotenen Funktionen zur Verarbeitung verschiedener Daten- und Medienformate.
Angriffe auf Apps selbst, jenseits des Browsers, sind selten. Der Browser wird jedoch oft das Ziel verschiedenster Angriffe, da er eine vergleichbar einfach zu erreichende Schnittstelle ist (Aufruf eines Links) und ein großes Maß an Unterstützung für die verschiedensten Protokolle und Medieninhalte bietet. Bei einem solchen Angriff wird entweder der Browser selbst attackiert [114] oder der Browser wird verwendet, um Daten- oder Medieninhalte auf das SP herunterzuladen, die eine Schwachstelle in den weiterverarbeitenden Systemkomponenten ausnutzen [62].

Im Zusammenhang mit der Audio-Bypass Bedrohung ist hier aufzuführen, dass der Zugriff auf die Audiodaten sowie das Internet über das Android App Permission Modell geregelt wird. Dies bedeutet, eine App kann in der Lage sein, Audiodaten zu arbiträren Zeitpunkten aufzunehmen und über das Internet zu

versenden, sollte der Nutzer ihr diese Rechte eingeräumt haben. Somit spielen die vom Nutzer installierten Apps und die ihnen gestatten Rechte eine große Rolle als Angriffsvektor auf die Audiodaten eines SP. Die Granulierung des App Permission Modells erlaubt keine gezielte Berechtigung für Apps basierend auf dem Kontext, in dem auf das Mikrofon zugegriffen werden soll.

5.2.2 Remote Code Execution Angriff

Im Gegensatz zu dem BB Komplex ist der Remote Code Execution Angriff für weiterführende Angriffe auf das Android OS nicht unbedingt zwingend erforderlich. Durch die Möglichkeit des Nutzers, Apps zu installieren, bietet sich ein weiterer Weg, in das System einzudringen (siehe Kapitel 5.2.5). Zudem ermöglicht Android den verschiedenen Apps auch einen Zugang zum Internet, dies ermöglicht es einem Angreifer nicht nur Systemkomponenten direkt zu attackieren, sondern auch vom Nutzer installierte Apps.
Es ist darauf hinzuweisen, dass das auf dem Linuxkernel basierende Rechtesystem über Benutzergruppen mit spezifischen Berechtigungen Schutz gegen solch eine Art von Angriffen bietet. Dieser Schutz ist durch die Erweiterung des von Android verwendeten Linuxkernels durch SELinux besonders gestärkt worden. So wird nicht nur Privilege Escalation Angriff zur Ausweitung der Rechte (z.B. zur Umgehung des Android App Permission Modells) benötigt, sondern es bedarf auch eines Angriffs zur Umgehung des domainbasierenden SELinux Schutzes [109].

Kategorie	Identifizierung	Ausnutzung
Zeit	5	0
Expertise	5	1
Wissen über Ziel	2	2
Zugang zu Ziel	0	0
Equipment	1	2
Partielles Angriffspotential	13	5
Angriffspotential	18	
Benötigtes Potential	Mittel	

Tabelle 5.1: Angriffspotentialbewertung eines Remote Code Execution-Angriffs auf Android. Eine Aufschlüsselung der für die Einschätzung verwendeten Werte sowie deren weiterführende Beschreibung befindet sich in den entsprechenden Tabellen in Anhang 9.2.

Für RCE-Angriffe auf das Android OS, den verwendeten Kernel und verschiedenste Apps, gibt es eine Vielzahl an Beispielen. Das bekannteste sollte der Stagefright [62] Angriff auf den Android Mediaserver sein. Dabei wird über eine

modifizierte Videodatei RCE innerhalb des Mediaservers erreicht, besonders gravierend hierbei ist, dass Videodateien in Android teilweise automatisch dem Mediaframework zugesandt werden, um diese für ein Abspielen ohne Verzögerung für den Anwender vorzubereiten.
Der Stagefright Angriff wurde bereits mit der Version 5.1 von Android behoben, der Vorfall führte zu weitreichenden Veränderungen der Android Systemarchitektur. So wurde vor allem der Zugang der nativen Frameworks zum Kernel extrem beschnitten und die einzelnen Teile des Frameworks in mehrere kleinere Einheiten aufgeteilt. Die Berechtigungen dieser Einheiten wurden zusätzlich durch die Verwendung von Benutzergruppen (auf denen die Android Sandbox basiert) sowie speziellen SELinux Domainen auf ein absolutes Minimum eingeschränkt. Nach diesen Architekturänderungen wird dieser Angriff durch die Verwendung von SELinux mitigiert, nicht jedoch die Möglichkeit eines Audio-Bypass Angriffs, basierend auf den Rechten des Mediaframeworks. Da dieses sowohl Zugriff auf das Internet als auch auf die Audiostreams hat, könnten auf Basis des Mediaframeworks, ohne weitere Komponenten zu verwenden, Audiodaten in der Umgebung des SP aufgenommen, und über die Internetverbindung an den Angreifer versendet werden.
Trotz aller Abwehrmaßnahmen werden immer wieder neue RCE-Schwachstellen im Android System und verschiedener am Markt erhältlicher Apps gefunden. So wurden 2017 bereits innerhalb der ersten fünf Monate 107 RCE-Angriffe auf das Android (OS und Apps) öffentlich gemeldet [60].
Als Beispiel für einen RCE-Angriff auf einen Teil des Kernels wird der von Brand [129] vorgestellte Angriff verwendet. Dieser greift die Binder-Infrastruktur, die vom Kernel angeboten wird, an und ermöglicht die Übernahme von sehr hoch privilegierten Prozessen. Bei der Ausnutzung dieser Schwachstelle werden speziell präparierte Medieninhalte dem SP zugesandt. Bei der Übertragung dieser Medieninhalte über den Binder-IPC wird ein spezieller Teil des Mediaframeworks verwendet, der einen Fehler enthält, der es endgültig ermöglicht, Kontrolle über den für die Binder-Infrastruktur zuständigen Prozess zu erlangen.
Dieser RCE-Angriff ist besonders schwerwiegend, da er auch als ein Privilege Escalation Angriff angesehen werden kann, da ein Prozess mit sehr weitreichenden Berechtigungen übernommen wird. Allerdings muss auch hier, trotz der weitreichenden Rechte des übernommenen Prozesses, für einen weiterführenden Angriff auf das System der von SELinux gebotene Schutz ausgehebelt werden. Ein Patch für die Schwachstelle wurde veröffentlicht, allerdings unterbinden auch die mit der Veröffentlichung von Android 7 weiter ausgebauten SELinux Richtlinien den für diesen Angriff gewählten Weg vollständig.

Benötigte Zeit:
Die Schwachstelle ist zwar über einen Fuzzingtest[9] initial gefunden worden, was für eine niedrige Entdeckungshürde spricht, aber es wird ein sehr komplexer, weiterführender Angriffsweg zur Umgehung von Schutzmaßnahmen wie Address Space Layout Randomization (ASLR) und Executable-Space Protection (NX) benötigt.
Für die Durchführung eines RCE-Angriffs wird an sich keine Zeit benötigt, allerdings ist hierbei nicht berücksichtigt, dass es, je nachdem wie der Angriff übertragen und ausgelöst wird, zu Verzögerungen kommen kann. Diese liegen in der technischen Funktion eines Exploits begründet. Die Verzögerungen können nur wenige Sekunden (Angriff wird automatisch auf dem angegriffenen SP ausgeführt z.B. ein zugesandtes Bild wird automatisch geladen) betragen, aber auch wesentlich mehr Zeit beanspruchen (Anwender muss mit dem Smartphone interagieren, damit der Angriff ausgelöst wird z.B. der Nutzer muss ein modifiziertes Bild betrachten).

Benötigte Expertise:
Aufgrund der vielen in Android verwendeten systemseitigen Schutzmaßnahmen gegen ungewollte Codeausführung, müssen verschiedenste Techniken zur Umgehung dieser Maßnahmen verwendet werden. Dies erfordert nicht nur ein großes Maß an Erfahrung, sondern auch sehr spezialisiertes Wissen.
Für die Anwendung eines solchen Angriffs wird ein Mindestmaß an Verständnis für das anzugreifende Systeme sowie für den verwendeten Angriff benötigt.

Benötigtes Wissen über Ziel:
Nahezu der gesamte Android Quelltext ist Open Source und es gibt ein breites Spektrum an Sekundärliteratur zu den verschiedensten Teilen des Systems. Das für einen solchen Angriff benötigte Wissen kann aus diesen öffentlichen Dokumenten bezogen werden.
Zur Auswahl des richtigen Angriffs wird ein Grundmaß an Wissen (SP Marke und Modell, sowie die verwendete Softwareversion) über das Ziel benötigt. Teile dieser Informationen werden ebenfalls zur Identifikation eines relevanten Angriffs benötigt. Zusätzlich wird entweder die Netzwerkadresse des Opfers benötigt oder es muss ein anderer Weg bekannt sein, dem Ziel die für den Angriff verwendeten Datenpakete zukommen zu lassen.

Benötigter Zugang zum Ziel:
Da es sich bei Android um ein weitverbreitetes System handelt und die verschiedenen (auch veralteten Versionen) der Betriebssysteme heutzutage

[9]Bei sogenannten Fuzzingtests werden große Mengen an zufälligen Daten an die zu testende Schnittstelle übertragen, um die Resistenz der Datenverarbeitung gegen Fehler innerhalb der übertragenen Daten zu prüfen.

einfach zu erhalten sind, stellt es kein Problem dar, eine Testumgebung für die Schwachstellensuche zu organisieren.
Da es sich um einen RCE-Angriff handelt, der über den IP-basierenden Netzwerkverkehr des Smartphones ausgeführt werden kann, bedarf es keinen Zugang zum Ziel, um diesen Angriff durchzuführen.

Benötigtes Equipment:
Für die Entwicklung eines solchen Angriffs wird keine sonderlich spezialisierte Ausrüstung benötigt, er kann mit dem gängigen Toolset eines mit Kernel Entwicklung erfahrenen Entwicklers identifiziert sowie implementiert werden.
Für die praktische Durchführung dieses Angriffs wird ein Computer mit Internetverbindung benötigt sowie die Möglichkeit mit dem Ziel Daten auszutauschen (z.B. Mail, HTTP, MMS).

5.2.3 Angriff auf den Radio Interface Layer

Durch die Einführung von SELinux und die Neustrukturierung des RILs in Android 7 [7] ist ein Angriff auf die verschiedenen Teile des RILs erheblich erschwert worden. Derzeit sind keine aktuellen Angriffe auf den vom Android Open Source Project stammenden Teil des RIL selbst öffentlich bekannt. Jedoch werden immer wieder Schwachstellen in der VendorRIL Komponente des Softwarestacks gefunden [122]. Diese Schwachstellen sind von besonderem Interessse, sollte der Angreifer bereits den BB kontrollieren. Es ermöglicht einen, im Vergleich zu dem Angriff auf das SMD, einfachen Weg, die Kontrolle über einen Prozess auf dem AP zu übernehmen. Zwar ist auch der RIL inzwischen durch entsprechend eingegrenzte Berechtigungen und, von SELinux geprüfte, Zugriffsrechte vom System weitestgehend isoliert, dennoch wäre es durch die Verwendung eines weiteren Angriffs möglich, das System zu übernehmen oder zumindest das Audiorouting zu beeinflussen.
Im Zusammenhang mit Angriffen auf die VendorRIL ist die, von dem Replicant Projekt gefundene, Schwachstelle in der VendorRIL von älteren Samsung SP zu erwähnen. Die am Projekt beteiligten Entwickler haben eine Open Source Alternative für diese VendorRIL entwickelt und sind dabei auf eine Funktionalität gestoßen, die es ermöglicht, auf das Dateisystem von Android zuzugreifen [105]. Dieser Angriff bezieht sich aber auf ältere SP Modelle und wurde vollständig durch die Einführung von SELinux mitigiert. Dieser Fall zeigt jedoch auf, dass die proprietären Anteile des Android OS, die durch die Hersteller hinzugefügt werden, ein erhebliches Sicherheitsrisiko darstellen können. Es ist jedoch auch anzuzweifeln, dass eine Offenlegung des gesamten herstellerseitigen Codes eine Verbesserung bringen würde, da es einen erheblichen Aufwand darstellt, einen Nachweis darüber zu führen, ob eine entsprechende binäre Datei wirklich mit dem für sie vorliegenden Quelltext korrespondiert.

Da es zur Analyse dieser Art von Angriff kein direktes Beispiel öffentlich bekannt ist, wird, wie in der Analyse des Angriffs auf das SMD (siehe Kapitel 4.2.4), der sehr ähnliche Angriff auf die Schnittstelle des WLAN-Subsystems [128] als Basis zur Bewertung herangezogen.

Kategorie	Identifizierung	Ausnutzung
Zeit	5	0
Expertise	5	1
Wissen über Ziel	3	2
Zugang zu Ziel	0	1
Equipment	4	2
Partielles Angriffspotential	17	6
Angriffspotential	23	
Benötigtes Potential	Mittel	

Tabelle 5.2: Angriffspotentialbewertung eines Angriffs auf die VendorRIL. Eine Aufschlüsselung der für die Einschätzung verwendeten Werte sowie deren weiterführende Beschreibung befindet sich in den entsprechenden Tabellen in Anhang 9.2.

Der Angriff ermöglicht die Übernahme eines Moduls mit der höchst möglichen Berechtigungsstufe innerhalb des Kernels, von hieraus kann der Angreifer den Kernelkontext unter seine Kontrolle bringen und arbiträren Code im System ausführen. Dies führt zu einem Totalverlust sämtlicher Integrität des Systems, da jegliche Schutzmaßnahmen durch den Kernel überbrückt oder schlicht deaktiviert werden können. Eine Ausnahme hierzu bildet die Singnaturprüfung der Systempartition des Bootloaders während des Systemstarts, dies bedeutet, dass persistente Veränderungen am System, selbst bei dem nächsten Neustart des SP zu einem Fehler während der Singnaturprüfung führen würden. In diesem Fall würde der Systemstart abgebrochen und der Nutzer über diesen Vorfall informiert.
Im Gegensatz zu einem Angriff zur Rechteausweitung (siehe Kapitel 5.2.4) ist die Persistenz auf dem SP für diese Art von Angriff ein Problem. Der Angreifer ist zwar in die Firmware des entsprechenden Subsystems eingedrungen, kann aber auch dort, auf Grund der Codesignierung, keine Persistenz erreichen. Dies bedeutet, dass der Angreifer nach jedem Neustart des Systems wieder einen Angriff durchführen muss, um die Kontrolle über das SP zu erhalten. Es könnte jedoch eine App mit entsprechenden App Permissions im System installiert werden, diese Art von App könnte unerkannt im Hintergrund laufen und Audiodaten sammeln sowie versenden (siehe Kapitel 5.2.5).

Benötigte Zeit:
Die Schwachstelle befindet sich in proprietärem Code, für den kein Quelltext vorliegt. Das bedeutet, für die Identifizierung einer solchen Schwachstelle muss das System ausgiebig mittels eines Decompilers analysiert werden, um einen ausnutzbaren Fehler im Code zu finden. Neben dieser zeitaufwendigen Arbeit müssen im Weiteren auch die Android Schutzmaßnahmen ausgehebelt werden (z.B. ASLR oder NX).
Ist ein Exploit bereits vorhanden, wird kein weiterer Aufwand benötigt, um ein Ziel anzugreifen. Sollte jedoch der Code angepasst werden müssen (z.B. aufgrund eines Updates), kann dies variieren.

Benötigte Expertise:
Da bei einem Angriff auf die VendorRIL nicht nur der proprietäre Code des Herstellers analysiert werden muss, sondern auch der Android Code des AOSP, benötigt der Angreifer zur Identifikation nicht nur Reverse Engineering Fähigkeiten, sondern auch fundiertes Wissen über den Code des AOSP.
Bei der Durchführung eines solchen Angriffs, basierend darauf, dass schon der BB erfolgreich angegriffen wurde, ist kein weiteres Wissen jenseits der Bedienung des verwendeten Equipments erforderlich (Betriebssystemkonsole, Skripte, Base Transceiver Station).

Benötigtes Wissen über Ziel:
Da es sich bei der VendorRIL um proprietären Code handelt und in den meisten Fällen ein nicht öffentliches Protokoll zur Kommunikation über diese Schnittstelle verwendet wird, ist weitreichendes Wissen über die Architektur des Systems und Protokolls nötig, um solch eine Schwachstelle zu identifizieren. Auch für die Ausnutzung eines bekannten Exploits muss ein Mindestmaß an Wissen über das Ziel vorhanden sein. So müssen Marke und das Modell des anzugreifenden SP bekannt sein sowie die Softwareversion, die auf dem Gerät vorliegt. All diese Informationen sind unerlässlich für einen erfolgreichen Angriff.

Benötigter Zugang zum Ziel:
Da es sich bei Android um ein weitverbreitetes System handelt und die verschiedenen (auch veralteten Versionen) der Betriebssysteme heutzutage einfach zu erhalten sind, stellt es kein Problem dar, eine Testumgebung für die Schwachstellensuche zu organisieren.
Da dieser Angriff auf der Tatsache beruht, dass der BB schon mittels eines RCE-Angriffs übernommen wurde, muss der Angreifer sich zumindest in der Nähe des SP aufhalten, um den entsprechenden RCE-Angriff durchführen zu können.

Benötigtes Equipment:
Da es sich bei der angegriffenen VendorRIL um proprietären Code handelt, wird ein ARM-Architektur fähiger Decompiler für die Analyse des Systems benötigt. Da der Angreifer bereits Kontrolle über den BB hat, wird keine weitere Ausrüstung für diesen Angriff benötigt.

5.2.4 Privilege Escalation Angriff

Bei einem Privilege Escalation (PE, Deutsch: Rechteausweitung) Angriff handelt es sich um den Versuch, aus den möglicherweise eingeschränkten Berechtigungen eines Prozesses auszubrechen, um im idealen Fall die höchst mögliche Berechtigung ohne SELinux Beschränkungen zu erreichen. Das Rechtesystem, das Android für Prozesse verwendet, darf nicht mit dem App Permission Modell verwechselt werden. Bei den Berechtigungen handelt es sich um die auf Kernelebene vergebenen Lese- und Schreibrechte, über die bestimmt wird, mit welchen Teilen des Systems ein Prozess interagieren darf und mit welchen nicht. Abgesehen davon werden die einzelnen Rechte im System zusätzlich durch die SELinux Domains abgesichert, die die unerlaubte Ausweitung von Berechtigungen weiter erschweren.
Dieser Vorgang ist eng mit dem im SP Bereich bekannten Rooting verbunden. Dies bezieht sich auf das englische Wort root (Wurzel), das das bei der Installation eines Betriebssystems angelegte Benutzerkonto bezeichnet, das mit den weitreichendsten Zugriffsrechten ausgestattet ist. Bei dem sogenannten Rooting wird dem Nutzer des Telefons ermöglicht, Befehle mit diesem Level an Systemrechten auszuführen. Die höchstmögliche Berechtigung in Android ist die sogenannte "System_Server" Gruppe, die mit der root-Nutzer Gruppe unter herkömmlichen Linux Systemen verglichen werden kann. Das von dem Nutzer des Smartphones durchgeführte Rooting basiert meistens auf der Veränderung der Systempartition, dies ist nicht möglich, solange die Signatur des System vor jedem Start geprüft wird. Der Nutzer muss die Prüfung dieser Signatur deaktivieren, da das SP ansonsten nicht mehr startet. Allerdings stellt diese Singnaturprüfung eine der wichtigsten Integritätsprüfungen des Android Systems dar, ohne diese ist es nicht möglich, Aussagen über die vom OS gebotene Sicherheit zu tätigen.
Von besonderer Bedeutung ist der PE Angriff auch für auf dem SP installierte Apps. Diese können mittels eines PE Angriffs ihre Rechte ausweiten und z.B. auf das Mikrofon des SP zugreifen und Audioaufnahmen durchführen, ohne die entsprechende App Permission zu haben. Jegliche Schutzmaßnahmen gegen einen Angreifer mit einem vollständigen PE Angriff sind ineffektiv, da sie auf Kernelebene umgangen werden können. Dies beinhaltet auch die Möglichkeit, den SLIMbus neu zu rooten, was einen Audio-Bypass Angriff über die GSM Verbindung des SP, wie er in Kapitel 4.2.3 für ältere Architekturen vorgestellt wurde, wieder ermöglichen würde.

Kategorie	Identifizierung	Ausnutzung
Zeit	5	0
Expertise	5	1
Wissen über Ziel	2	2
Zugang zu Ziel	0	2
Equipment	1	2
Partielles Angriffspotential	13	7
Angriffspotential	20	
Benötigtes Potential	Mittel	

Tabelle 5.3: Angriffspotentialbewertung eines Privilege Escalation Angriffs. Eine Aufschlüsselung der für die Einschätzung verwendeten Werte sowie deren weiterführende Beschreibung befindet sich in den entsprechenden Tabellen in Anhang 9.2.

Als Basis für die Bewertung dieser Art von Angriff wird der sogenannte Quadrooter Angriff [61, 113] verwendet. Dabei handelt es sich um einen lokalen (Angriff aus einer App ohne jegliche App Permissions) PE Angriff mit weitreichenden Folgen. Er ermöglicht das arbiträre Schreiben von Speicherinhalt innerhalb das Kernelkontexts. Dies wird dazu verwendet, SELinux zu deaktivieren und anschließend den eigenen Prozess in die höchst mögliche Berechtigungsgruppe zu verschieben. Damit ist effektiv ein vollständiger Zugriff auf das gesamte System möglich.

Benötigte Zeit:
Die Schwachstelle ist bei der Überprüfung mehrerer von Qualcomm stammender Android Kernel Module gefunden worden. Da es sich bei dem Angriff um eine Aneinanderreihung sehr spezifischer Angriffe auf verschiedene Teile des Systems handelt, ist davon auszugehen, dass ein erheblicher Zeitaufwand in die Analyse des gesamten Softwarestacks geflossen ist, bevor diese außergewöhnliche Angriffsreihe gefunden wurde.
Besonders bei der praktischen Durchführung dieses Angriffs ist anzumerken, dass er nur wenig von der spezifischen Hardwareplattform des SP abhängig ist, da die attackierten Infrastrukturen in allen QC basierenden SP gefunden werden konnten. Dadurch ist die Anpassung dieses Angriffs auf andere Hardwareplattform meistens nicht notwendig.

Benötigte Expertise:
Für die Identifikation der Schwachstelle sowie die Umsetzung des explizit gewählten Weges der zu dem PE Angriff führt, wird weitgehendes Wissen über verschiedene komplexe Kernelprozesse benötigt. Zusätzlich dazu muss

die Fähigkeit vorhanden sein, eine Implementation des Angriffs, inklusive der Umgehung für verschiedene Schutzmaßnahmen, durchzuführen.
Aufgrund der hohen Komplexität dieses Exploits und der Voraussetzung, ihn in einen Prozess auf dem SP einzubinden, ist die Anwendung nicht trivial. Es wird hierbei davon ausgegangen, dass kein Werkzeug zur Verfügung steht, das sämtliche Aufgaben in diesem Zusammenhang automatisiert.

Benötigtes Wissen über Ziel:
Da es sich bei allen angegriffenen Systemen um Open Source Teile des Android OS handelt, ist der Zugang zu ihnen nicht eingeschränkt, des Weiteren wird über diese Systeme, gerade im Bereich der Kernelentwicklung für Android, aktiv diskutiert.
Für die Ausführung dieses Angriffs muss beachtet werden, dass es sich bei dem Ziel um eine QC Plattform handeln muss. Jenseits des für die Identifikation erforderlichen Wissens wird noch eine Möglichkeit benötigt, das angegriffene Ziel zu identifizieren (z.B. Nutzername des für Android verwendeten Benutzerkontos oder andere Informationen, die zur Identifikation des Ziels geeignet sind).

Benötigter Zugang zum Ziel:
Da es sich bei Android um ein weit verbreitetes System handelt und die verschiedenen (auch veralteten Versionen) der Betriebssysteme heutzutage einfach zu erhalten sind, stellt es kein Problem dar, eine Testumgebung für die Schwachstellensuche zu organisieren.
Da schon eine App auf dem System installiert sein muss, wird davon ausgegangen, dass eine Bewertung von 2 Punkten in der Kategorie zwar nicht die tatsächliche Zeit repräsentiert, während der Zugang zum Ziel vorhanden sein muss, aber die Tatsache, dass ein Startpunkt auf dem SP für diese Art von Angriff notwendig ist.

Benötigtes Equipment:
Für die Entwicklung eines solchen Angriffs wird keine sonderlich spezialisierte Ausrüstung benötigt, er kann mit dem gängigen Toolset eines mit Kernel Entwicklung erfahrenen Entwicklers identifiziert sowie implementiert werden.
Für die praktische Durchführung dieses Angriffs wird ein Computer mit Internetverbindung benötigt, um den Angriff vorzubereiten.

5.2.5 Angriff über eine App

Im Zuge der Analyse der Audio-Bypass Bedrohung durch Angriffe, die auf Apps basieren ist festgestellt worden, dass die verschiedenen Vorgänge, die zu solch einem Angriff führen könnten, sich nicht direkt im Angriffspoten-

tialmodell abbilden lassen. Die hohe Abhängigkeit dieses Angriffs in Bezug auf Nutzerinteraktionen kann nicht im Angriffspotentialmodell abgebildet werden und würde keine repräsentative Bewertung dieses Angriffs ermöglichen. Daher wird an dieser Stelle darauf verzichtet, eine formelle Angriffspotentialbewertung durchzuführen. Dennoch wird auf die Gefahren in Bezug auf diesen Angriff eingegangen und die verschiedenen beteiligten Mechanismen werden vorgestellt.

Für die Durchführung eines Audio-Bypass Angriffs braucht eine App neben den in Kapitel 5.1.3 vorgestellten App Permissions zur Audioaufnahme auch eine Möglichkeit, die hierbei erhobenen Daten an den Angreifer zu übermitteln. Die zur Übertragung von Daten benötigten Berechtigungen gehören zu den normalen App Permissions. Der Angreifer kann daher frei zwischen allen verfügbaren Schnittstellen wählen, da der Anwender weder um Erlaubnis gefragt noch direkt darauf hingewiesen wird, dass die entsprechende App diese Schnittstellen nutzen kann.
Für die Funktionalität einer App, die einen Audio-Bypass Angriff durchführen soll ist die Möglichkeit, im Hintergrund Audioaufnahmen anzufertigen, integral. Dies sollte auch während andere Apps im Vordergrund sind oder das SP gesperrt und der Bildschirm ausgeschaltet ist möglich sein. Dies kann über einen Android Background Service [26] erreicht werden. Dieser kann, wenn er die zusätzliche normale App Permission "WAKE_LOCK" deklariert hat, weiter ausgeführt werden, auch wenn das Handy gesperrt wurde.
Um sicherzustellen, dass die App auch ausgeführt wird, kann der Autostartmechanismus von Android eingesetzt werden. Für diesen wird die ebenfalls als normal eingestufte App Permission "RECEIVE_BOOT_COMPLETED" [30] benötigt. Android ermöglicht einer einzelnen App mehrere Background Services zu starten, was dafür genutzt werden kann, einen weiteren Background Service zu starten, der die Ausführung des zur Audioaufnahme verwendeten Services überprüft und diesen gegebenenfalls neu startet.

Für die Übertragung der aufgezeichneten Daten muss jedoch bedacht werden, dass die meisten mobilen Datenverbindungen über das Datenvolumen abgerechnet werden und nach der Übertragung einer gewissen Datenmenge die Verbindungsgeschwindigkeit gedrosselt wird. Daher können nicht beliebig viele Daten über diese Verbindung übertragen werden ohne dass der Nutzer Auswirkungen davon mitbekommt. Diesem Problem kann aber mit verschiedenen Techniken entgegengetreten werden. Daten können nach der Aufnahme zuerst gespeichert werden und erst wenn eine nicht volumenbegrenzte Datenverbindung (WLAN) zur Verfügung steht übertragen werden. Apps ist es erlaubt, Daten auch auf den internen Speicher des Telefons zu schreiben ohne dafür eine App Permission zu benötigen. Android bietet zur Audioaufnahme

unter anderem den sehr stark komprimierenden AMR-NB Codec, der auf Sprachübertragung spezialisiert ist [27]. Auf dem zum Testen verwendeten Nexus 5x konnte mit diesem eine 30 minütige Audioaufnahme eines Gesprächs angefertigt werden, die nur 3 Megabyte Speicherplatz benötigt. Dies rückt es in den Bereich des Möglichen, Audioaufnahmen auch über einen längeren Zeitraum zu speichern und erst später über WLAN zu versenden.
Neben dem Datenvolumen ist die Audioaufnahmekapazität eines SP auch durch den Energieverbrauch beschränkt. Aber auch diese Begrenzung lässt sich umgehen. So könnte die Aufnahmefunktionalität zeitgesteuert ablaufen, vom aktuellen Ort des SP abhängen oder nur während mit dem SP telefoniert wird aktiv sein. Zusätzlich konnte bei Tests im Rahmen dieser Arbeit erst nach mehrstündigen Aufnahmen ein deutlicher Unterschied zu der normalen Laufzeit des SP (ohne Audioaufnahme) festgestellt werden.

Oft werden Angriffe mittels einer App über das sogenannte Social Engineering durchgeführt. Hierbei handelt es sich um den Versuch, über zwischenmenschliche Beeinflussung das Ziel dazu zu bewegen, eine sicherheitskritische Aktion durchzuführen, die einen anschließenden Angriff ermöglicht [77]. Hierfür kann die Funktionalität für einen Audio-Bypass Angriff in eine App eingebettet werden, die dem Nutzer weitere Funktionen bietet, die von dem eigentlichen Angriff ablenken sollen. Ein Beispiel für einen Angriff dieser Art wurde von Flossman [83] dokumentiert. Bei diesem Angriff wurde mittels Social Engineering verschiedenen Mitgliedern einer militärischen Einheit eine manipulierte Version einer Messenger App untergeschoben. Diese ermöglichte den Angreifern neben dem Zugriff auf verschiedene persönliche Daten der SP Nutzer auch einen Audio-Bypass Angriff.
Neben solchen gezielten Angriffen sind aber auch wesentlich breitflächigere Angriffe bekannt. In Arp et al. [40] wird ein Audio-Bypass Angriff beschrieben, der dazu verwendet wird, Bewegungsprofile des SP Nutzers für Marketingzwecke anzulegen. Hierbei werden örtlich begrenzte Ultraschallsignale ausgesendet, die von einer App auf dem SP aufgenommen und erkannt werden. Hierdurch kann die genaue Position des Nutzers, z.B. innerhalb eines Ladengeschäfts festgestellt werden. Aus diesen Daten lässt sich ein Bewegungsprofil erstellen, das mögliche Rückschlüsse auf die Interessen des entsprechenden Kunden geben kann. Bei diesem Angriff werden die eigentlichen Audiodaten zwar lokal auf dem Smartphone verarbeitet und keine tatsächlichen Audiodaten in das Internet übertragen. Dieses Verhalten könnte sich jedoch durch ein Update ändern.
Neben direkten Audio-Bypass Angriffen können Apps aber auch als Pivotingstation für weitere Angriffe auf das System verwendet werden. Eine App, die durch den Nutzer installiert wird, könnte durch die Ausnutzung eines PE Angriffs Kontrolle über das gesamte SP erhalten und diese Rechte für weitere Angriffe nutzen.

Es ist darauf hinzuweisen, dass eine solche App auch als Teil eines weiteren Angriffs ohne eine Interaktion des Nutzers installiert werden könnte (siehe Kapitel 5.2.3 und 5.2.2). Hierbei verwenden Angreifer oft verschiedene Techniken, um vom Nutzer unerkannt zu bleiben, indem sie sich z.B. als ein Systemdienst tarnen oder als eine andere legitime App.

6 Konzeption von Abwehrmaßnahmen

In diesem Kapitel sollen verschiedene Abwehrmaßnahmen gegen die Audio-Bypass Bedrohung selbst sowie für die verschiedenen Angriffe, die zu einem Audio-Bypass führen könnten, konzipiert werden. Sollte es für die vorgeschlagene Abwehrmaßnahme schon eine Implementation geben (kommerziell oder Open Source), wird diese in der entsprechenden Sektion ebenfalls erwähnt.
Die Abwehrmaßnahmen können entweder auf physikalischen Maßnahmen basieren, wie z.B. die Veränderung von Hardwareaspekten des Smartphones (SPs), oder aber auf Software-Basis greifen, wie z.B. eine App, die versucht, das Mikrofon dauerhaft zu belegen, sodass kein anderer Prozess darauf zugreifen kann. Es ist anzumerken, dass es sich gezeigt hat, dass die effektivsten Maßnahmen auch meistens die stärksten Einschränkungen für die Benutzbarkeit des SP mit sich bringen.
Im Folgenden werden erst zwei allgemeine, aber besonders wichtige Abwehrmaßnahmen besprochen: Softwareupdates und Nutzerschulung. Im Anschluss daran werden die physikalischen Abwehrmaßnahmen vorgestellt, diese sind: Entfernung des Mikrofons, Isolierung des SP sowie eine Hardware Firewall. Den Abschluss bilden die auf Software basierenden Abwehrmaßnahmen: allgemeine Härtung von Android, Ausweitung des App Permission Modells sowie die Verwendung eines Open Source Softwarestacks.

6.1 Softwareupdates

Für Android SP gibt es zwei unterschiedliche Arten von Updates. Dies ist zum einen ein Systemupdate, zum anderen die sogenannten monatlichen Sicherheitsupdates. Die Sicherheitsupdates können wohl als eine der wichtigsten Abwehrmaßnahmen angesehen werden, da sie zeitnahe Mitigation für bekannt gewordene Schwachstellen bietet. Schwachstellen werden heutzutage auf verschiedenen Wegen entdeckt und an die Hersteller der entsprechenden Software gemeldet. Idealerweise werden sie bereits von dem Testteam des Herstellers selbst noch vor der Veröffentlichung entdeckt und beseitigt. In diesem Fall dringt meistens keinerlei Information über die Schwachstelle an die Öffentlichkeit. Doch nach der Veröffentlichung einer Software ist dies leider

nicht mehr möglich, Kunden müssen über eine Schwachstelle informiert werden und ein entsprechendes Update in ihr System einspielen.
Unabhängige Sicherheitsforscher entdecken regelmäßig Schwachstellen in schon sehr lange veröffentlichtem Code [62]. Nachdem eine Schwachstelle von unabhängigen Sicherheitsforschern an den Hersteller gemeldet wurde, wird heutzutage in der Regel eine Frist für die Veröffentlichung eines Patches gesetzt [115]. Nach Ablauf dieser Frist werden Informationen über die Schwachstelle veröffentlicht (meist mit einem bereits implementierten Angriff). Dieses Vorgehen hat sich bewährt, da es maßgeblich dazu beigetragen hat, die durchschnittliche Zeit zu verkürzen, die für die Beseitigung einer Schwachstelle von Seiten des Produzenten benötigt wird [39]. Dies ist von Interesse, da es nicht unüblich ist, dass die gleiche Schwachstelle von verschiedenen Sicherheitsanalysten unabhängig voneinander entdeckt wird.
Nach der Veröffentlichung eines Patches oder der Informationen über eine Schwachstelle kann davon ausgegangen werden, dass diese Informationen zeitnah dazu verwendet werden, einen Angriff gegen die entsprechende Schwachstelle zu implementieren. In der Regel ist es nur eine Frage der Zeit, bis die Quelltexte für solche Angriffe öffentlich zur Verfügung gestellt werden. Es gibt gleich mehrere Websites, die solche Angriffe archivieren und kostenlos zur Verfügung stellen [58, 104].

Sogenannte Systemupdates beziehen sich nicht nur auf Schwachstellen, sondern erweitern das System um neue Funktionalitäten sowie, meist auf architekturellen Veränderungen basierende, verbesserte Schutzmaßnahmen auf Systemebene. Das Update auf Android Version 7.0 erweiterte die kernelseitigen Schutzmaßnahmen sowie den internen Updatemechanismus. Zusätzlich wurde das gesamte Mediaframework überarbeitet, um eine stärkere Modularisierung und Isolierung der einzelnen Komponenten zu erreichen [16].
Mit Android 7.1.1 ist die Installation von Sicherheitsupdates, zumindest auf den von Google geförderten Pixel SPs, vollständig automatisiert worden. Dies bedeutet, Sicherheitsupdates werden im Hintergrund heruntergeladen und installiert, ohne dass das System neu gestartet werden muss.

Sollte ein Nutzer die Durchführung von Updates vernachlässigen oder sollten keine Updates für sein SP zur Verfügung gestellt werden, mindert dies das Angriffspotential, das benötigt wird, um einen erfolgreichen Angriff auf das Ziel durchzuführen drastisch.
Leider ist, wie in Kapitel 5.1.6 beschrieben, die Verbreitung der monatlichen Sicherheitsupdates unter Modellen jenseits der von Google vermarkteten Pixel und Nexus SPs eher schlecht [108]. Nicht nur werden die monatlichen Updates von vielen Herstellern nicht durchgesetzt, sondern auch die Systemupdates. Die Begründung für dieses Verhalten liegt wohl in dem benötigten Aufwand, die

generischen Sicherheitsupdates auf die von den Herstellern angepassten Androidsysteme zu portieren.
Als Abwehrmaßnahme kann daher hier nur empfohlen werden, vor dem Kauf eines SP zu recherchieren, wie die Updatepolitik des Herstellers aussieht. Allerdings ist es in der Regel jedoch so, dass nur die Spitzenmodelle der verschiedenen Hersteller über längere Zeit hinweg mit Updates versorgt werden.

6.2 Nutzerschulung

Ein zentraler Punkt jeder Abwehrmaßnahme ist die benötigte Akzeptanz durch den Nutzer, besonders bei Abwehrmaßnahmen, deren Effektivität von der korrekten und aktiven Anwendung durch den Nutzer abhängt. Sollte es einem Angreifer durch geschickte Manipulation seines Ziels auf sozialer Ebene (sogenannten Social Engineering) gelingen, dieses dazu zu bewegen, eine App zu installieren, reicht dies für einen erfolgreichen Audio-Bypass Angriff vollständig aus. Immer wieder werden Anwender sogar dazu bewegt, Sicherheitsmaßnahmen zu deaktivieren, was einen anschließenden Angriff erst ermöglicht [83].
Teilweise ist es möglich und sinnvoll, die Rechte der Nutzer zur Konfiguration des SP soweit einzugrenzen, dass sie nicht in der Lage sind, die Konfiguration des Gerätes bezüglich sicherheitskritischer Optionen zu verändern.
Dennoch spielt das Sicherheitsbewusstsein des Anwenders eine große Rolle für die Effektivität vieler Maßnahmen. Daher sollte darauf geachtet werden, gerade bei besonders gefährdeten Nutzergruppen, mittels regelmäßiger Schulungen die Fähigkeiten der Anwender aufzufrischen und die Möglichkeit einer Bedrohung wieder in das Gedächtnis zu rufen.
Nutzer müssen weiter über die Gefahren durch Apps aufgeklärt werden. Es sollte besonders darauf geachtet werden, den Nutzer vor Gefahren durch Apps aus unbekannten Quellen zu warnen. Da der Google App Markt überwacht wird, ist hier ein gewisser Grundschutz vor ungewollten Funktionalitäten in Apps geboten. Dieser Grundschutz wird von alternativen Quellen in der Regel nicht geboten [16].

6.3 Physikalische Abwehrmaßnahmen

6.3.1 Entfernung des Mikrofons

Eine sehr wirksame Abwehrmaßnahme ist die vollständige Entfernung aller in das SP integrierter Mikrofone. Dies schränkt zwar die Nutzbarkeit des SP stark ein, bietet jedoch einen vollständigen Schutz vor einem Audio-Bypass Angriff mittels des SP.
Die integrierten Mikrofone können für die Verwendung des SP zum Telefonie-

ren mittels eines externen Mikrofons ersetzt werden (kabelgebunden oder Bluetooth). Dies erhält die Funktionsfähigkeit des SP, sorgt allerdings für einen Mehraufwand für den Nutzer. Dieser muss vor jedem Telefonat ein externes Mikrofon verbinden, welches nach Abschluss des Telefonats wieder entfernt werden muss, um die Wirksamkeit dieser Maßnahme zu garantieren.
Diese Maßnahme wurde im Rahmen dieser Thesis praktisch evaluiert, für Testzwecke wurde das Mikrofon aus einem älteren SP, einem Nexus S (2010), entfernt. Das Mikrofonmodul war mit dem Gehäuse verklebt und mittels einer Steckerverbindung mit der Leiterplatte des SP verbunden. Nach der Entfernung des Mikrofonmoduls funktionierte das SP weiterhin und es konnte ein externes kabelgebundenes Mikrofon für ein Telefongespräch verwendet werden.
Es ist darauf hinzuweisen, dass bei moderneren SP mehrere Mikrofone verbaut werden. Es muss darauf geachtet werden, dass sämtliche Mikrofone entfernt werden, da die Maßnahme ansonsten wirkungslos bleibt.
Eine elegante Alternative wäre die Verwendung von physikalischen Schaltern, die es ermöglichen, die Mikrofone des SP physikalisch vom Schaltkreis zu trennen. Allerdings wäre dies nur schwer bei einem modernen SP nachzurüsten, sondern müsste aus Platzgründen von Anfang an in das Design integriert werden.
Diese Lösung bietet einen nahezu vollständigen Schutz gegen Audio-Bypass Angriffe ohne die Anschaffung weiterer Werkzeuge und erhält dabei die Möglichkeit, auch während sicherheitskritischer Gespräche die weiteren Funktionen des SP zu nutzen. Allerdings existieren auch sogenannte "Side-Channel"-Angriffe, die durch diese Schutzmaßnahme nicht abgedeckt werden. Es ist möglich, Audiosignale in der Umgebung des SP durch Daten des gyroskopischen Sensors zu extrapolieren, dies kann aber ebenfalls durch die Entfernung des entsprechenden Sensors unterbunden werden.

6.3.2 Isolation des Smartphones

Die Isolation des SP vom gefährdeten Audiokontext ist eine weitere Möglichkeit. Hierbei wird das SP physikalisch aus dem Umfeld entfernt (z.B. in einen anderen Raum), was jedoch einige Nachteile mit sich führt. Es muss sichergestellt werden, dass sich das SP weit genug von der Audioquelle entfernt befindet, um unter keinen Umständen Aufnahmen davon anfertigen zu können. Dies wiederum entfernt das SP aus der direkte Nähe des Anwenders, was in einigen Fällen nicht akzeptabel sein kann (z.B. Erreichbarkeit oder Sicherheit des Gerätes).
Als Alternative bietet sich hierfür die Verwendung einer sogenannten Rauschbox [84]. Hierbei handelt es sich um eine mit geräuschdämpfenden Materialien ausgestattete Box oder einen Koffer, in dessen Inneren ein Rauschgeräusch abgespielt wird. Die Kombination aus Abschirmung von Geräuschen außerhalb der Box und dem Rauschen, das innerhalb der Box abgespielt wird, macht eine

Audioaufnahme des Audiokontextes außerhalb der Rauschbox unmöglich. Professionelle Rauschboxen sind in der Lage zu erkennen, wenn ein Gerät, das in ihnen aufbewahrt wird, über das Mobilfunknetz angerufen wird und signalisieren dies mittels eines eigenen Audiosignals.
Eine weitere Möglichkeit ist die Verwendung eines Kühlschranks [118] als Alternative zu einer professionellen Rauschbox. Dies wurde mit einem Nexus 5X ausprobiert. Ein Gespräch, das in direkter Nähe des Kühlschrankes geführt wurde, konnte nicht in der Audioaufnahme des SP, das währenddessen im Kühlschrank lag, gefunden werden.
Eine weitere Möglichkeit der Isolation ist es, das SP abzuschalten. Aufgrund der in modernen SPs eingesetzten Architektur ist sichergestellt, dass ein Angriff auf den Baseband-Prozessor (BB) diese Abwehrmaßnahme nicht überwinden kann, da dieser keine Kontrolle über die weiteren Hardwarekomponenten des SPs mehr besitzt. Es ist jedoch nicht auszuschließen, dass ein Angreifer, der über vollständige Kontrolle des Android OS verfügt, in der Lage ist, ein Abschalten nur zu simulieren. Für den Anwender wirkt es, als wäre das SP von ihm abgeschaltet worden, in Wahrheit jedoch befindet es sich immer noch in einem eingeschalteten Zustand und reagiert nur nicht auf die Eingaben des Nutzers. Daher ist eine Isolation über das Abschalten des Gerätes nicht vollständig verlässlich und andere Isolationsmethoden sollten, falls möglich, vorgezogen werden. Die Batterie nach dem Ausschalten aus dem Gerät zu entfernen, bietet in diesem Szenario einen guten Schutz. Allerdings ist es bei den meisten modernen SP nicht mehr vorgesehen, dass die Batterie aus dem Gerät entfernt werden kann.
Ein klarer Nachteil dieser Lösung (auch in Form einer Rauschbox) ist, dass die weiteren Funktionalitäten des SP (z.B: Messenger, Kalender, Notizen), während dieses isoliert wird, nicht verwendet werden können.

6.3.3 Hardware Firewall

Eine auf Hardware basierende Firewall zwischen dem Baseband-Prozessor und dem Applikationsprozessor (AP) könnte die Bedrohung durch den BB stark eingrenzen. Eine zusätzlich Überwachung des Shared Memory Device könnte einen Angriff über diesen, wie sie in Kapitel 5.2.3 und 4.2.4 beschrieben werden, frühzeitig erkennen und den Nutzer warnen oder aber die Angriffe auch komplett unterbinden. Wie aus Qualcomm (QC) internen Dokumenten hervorgeht, [124] wird diese Schnittstelle bereits besonders überwacht, aber weder über diese Abwehrmechanismen noch über die für den Speicherschutz verantwortlichen, QC proprietären, XPU Einheiten [79, 41] ist öffentlich viel bekannt.
Neben den durch den Hersteller gebotenen Schutzmaßnahmen bietet die deutsche Firma GSMK für ihre Cryptophone 500i [67] eine selbst entwickelte Hardware Firewall an. Leider ist bis auf ein nicht sehr spezifisches Patent [120] nichts über die Fähigkeiten dieser Firewall bekannt, sie verspricht jedoch, den Daten-

verkehr zwischen BB und AP auf Anomalien zu überwachen. In diesem Zusammenhang kann auf den von GSMK veröffentlichten Open Source Decompiler, für das von QC auf Hexagonprozessoren für den BB Softwarestack verwendete Format, hingewiesen werden [68].
Die Effektivität dieser Lösung hängt stark von der gewählten Umsetzung ab. Es ist unmöglich, eine Garantie dafür zu bieten, dass solch eine Art von Schutz in jedem Fall einen Angriff abwehren kann. Jedoch bietet sie eine Stärkung des Schutzes, ohne die Nutzbarkeit des SP für den Anwender zu verschlechtern. Allerdings ist auch in diesem Fall die Schutzmaßnahme nur wirksam, wenn der Anwender die ihm gebotenen Möglichkeiten wahrnimmt und auf etwaige Warnungen der Firewall auch eingeht.

6.3.4 Detektion auf Netzebene

Angriffe auf den BB erfordern in der Regel die Verwendung eines eigenen Mobilfunknetzwerkes zur Kommunikation mit dem BB. Für den Aufbau solcher bösartigen Netze werden Base Transceiver Stations oder auch zum Teil IMSI-Catcher verwendet. Da sich die Netze dieser Geräte doch in verschiedenen kleineren Details von den statischen Netzen der Mobilfunkprovider unterscheiden, können sie mittels spezieller Geräte, wie in Dabrowski et al. [56] vorgestellt, aufgespürt werden. Darüber hinaus kann bei einer dauerhaften Überwachung eines Gebietes mit solchen Detektionsgeräten eine Datenbank mit zu erwartenden Netzen angelegt werden. Sollte nun ein neues, noch unbekanntes Netz auftauchen, kann der Nutzer über eine mögliche Bedrohung durch dieses Netzwerk informiert werden.
Diese Systeme existieren auch in Form verschiedener Android Apps (AIMSICD [50], SnoopSnitch [111]), sie haben jedoch mit der großen Anzahl unterschiedlicher SP Modelle und der daraus resultierenden Hardwarevielfalt zu kämpfen und funktionieren daher leider nur auf den wenigstens SP Modellen zuverlässig. Desweiteren bieten sie keine Möglichkeit einer statischen und längerfristigen Überwachung eines Gebietes.
Diese Systeme können zwar nicht direkt einen Angriff auf den BB verhindern, können einen Nutzer jedoch auf die Anwesenheit eines möglicherweise bösartigen, unbekannten Mobilfunknetzwerks hinweisen. Dies kann als Indiz für einen möglichen Angriff angesehen werden und die Warnung über diesen Zustand ermöglicht es dem Anwender, auf die Situation entsprechend zu reagieren.

6.4 Software Abwehrmaßnahmen

6.4.1 Android Härtung

Unter Härtung einer Software wird im Allgemeinen verstanden, die Angriffsflächen zu minimieren sowie weitere Schutzmaßnahmen zu integrieren oder den Zugang zu dem zu härtenden System durch weitere Schutzmechanismen abzusichern.
Im Fall von Android werden die Schutzmaßnahmen durch das Android Open Source Project (AOSP) schon regelmäßig ausgeweitet (siehe Kapitel 6.1). Hierbei geht es darum, die allgemeine Sicherheit und Resistenz des Systems gegen Angriffe zu verbessern, hierdurch wird auch die Resistenz gegen eine Audio-Bypass Bedrohung im Allgemeinen verbessert.
Neben der durch das AOSP durchgeführten Härtung gibt es verschiedene weitere Gruppen, die versuchen, Android mit noch weiterführenden Schutzmechanismen auszustatten. Eines der bekanntesten Android Härtungs Projekte ist das CopperheadOS Projekt [54], das viele Sicherheitserweiterungen sowie striktere SELinux Regelungen implementiert [55]. Da besonderer Wert auf Datenschutz gelegt wird, sind die Google Playservices nicht Teil dieser Installation. Das bedeutet, dass keine der gängigen Google Anwendungen (z.B. Maps, Playstore, GMail) verfügbar sind, es werden jedoch Alternativen für diese geboten. Diese alternativen Standardanwendungen bieten zwar dieselbe Grundfunktionalität sowie einen besseren Datenschutz als ihren Google-Gegenstücke, sind diesen aber in Hinsicht auf Bedienungskomfort und oft auch Geschwindigkeit (besonders bei der Ortung innerhalb des Kartendienstes) unterlegen. Von besonderem Interesse für die Audio-Bypass Bedrohung ist die Tatsache, dass sämtliche Unterstützung für Google Now aus dem System entfernt wurde.
Der Einsatz eines solchen Systems bringt zwar gewisse Einschränkungen für den Nutzer, erhöht die Resistenz gegen verschiedene Angriffe aber enorm. Neben der allgemeinen Stärkung der verwendeten Sicherheitsmechanismen wurden zentrale Teile des Kernels durch besonders gehärtete Komponenten ersetzt (z.B. Prozess-Scheduler und Speicherallokation), was dafür sorgt, dass Angriffe, die für die AOSP Version von Android entwickelt wurden, in der Regel keinen Effekt auf CopperheadOS haben.

6.4.2 AuDroid

Das Android App Permission Modell ermöglicht keine kontextsensitive Vergabe von Rechten. Zum Beispiel ist es nicht möglich, den Zugriff für das Mikrofon nur unter gewissen Umständen oder Zuständen zu gestatten. Die Gefahr eines Audio-Bypasses durch eine App im Hintergrund könnte durch eben solch ein System verhindert werden, selbst wenn diese App eigentlich die Befugnis besitzt, Audioaufnahmen anzufertigen. Es kann für eine Audiorekorder App ein

vollkommen akzeptables Verhalten sein, eine Audioaufnahme zu starten, während sie im Fokus ist. Ein auffälliges Verhalten liegt vor, wenn sie versuchen sollte, eine Aufnahme zu starten, während sie nicht im Fokus ist oder der Bildschirm des SP abgeschaltet ist.
Ein System für kontextsensitive App Permissions wird von AuDroid [98] implementiert. Forscher schlagen ein auf SELinux basierendes Rechtemanagmentsystem vor, bei dem jedem geöffneten Audiokanal ein Label zugewiesen wird. Anhand dieses Labels (abhängig von den Endpunkten des Kanals) und einem fest definierten Regelwerk werden die Audiokanäle von SELinux überwacht und entweder zugelassen, blockiert oder eine Nachfrage an den Nutzer gestellt. Hierauf basierend könnten kontextsensitive Regeln für Audiokanäle definiert werden, die verhindern, dass Applikationen jenseits ihres eigentlichen Kontexts Audioaufnahmen anfertigen, ohne dass der Nutzer diese Interaktion gestattet. In diesem Szenario würde der Nutzer bei der ersten Audioaufnahme mit der Rekorder-App um Erlaubnis gefragt werden, ob die App Audioaufnahmen (im aktuellen Kontext) anfertigen darf. Hierbei kann der Nutzer wählen, ob dies nicht, einmal oder immer gestattet werden soll. Sollte die App nun außerhalb des Kontextes, in dem ihr der Zugriff auf das Mikrofon gewährt wurde, versuchen, eine Audioaufnahme zu starten, wird der Nutzer erneut angefragt, ob die App auch in diesem Kontext Audioaufnahmen anfertigen darf.
Ein solches System schränkt die Nutzung des SP nicht ein, ermöglicht aber einen App-basierten Angriff auf die Audiodaten effektiv abzuwehren. Sollte jedoch das gesamte System unter der Kontrolle des Angreifers stehen, kann auch diese Abwehrmaßnahme verhältnismäßig einfach umgangen werden.

6.4.3 Open Source

Eines der Probleme mit proprietären Anwendungen ist die Tatsache, dass nicht gewiss ist, welche Funktionalitäten letztendlich in der Anwendung implementiert wurden. Im Fall der vom Replicant Project gefundenen Schwachstelle (siehe Kapitel 5.2.3) stand zur Diskussion, ob es sich bei dem vorgefundene Verhalten um einen Fehler in der Anwendung handelte, oder ob die Funktionalität in ihrem Umfang gezielt in der Anwendung belassen wurde. Gerade im Zuge der staatlichen Überwachung werden geheimen Zugängen zu Kommunikationsgeräten verwendet, die von den Herstellern z.B. auf Geheiß eines Geheimdienstes in ihren Produkten versteckt wurden, und vor allem Zugriff auf die Daten sowie Audioaufnahmen des SP bieten sollen [95, 47].
Aber nicht nur die Ungewissheit über die vorhandene Funktionalität wäre ein Argument für Open Source, auch die Tatsache, dass der Quellcode Sicherheitsforschern eine wesentlich bessere Basis für ihre Analysen bieten würde, spricht dafür. Die Veröffentlichung des Quellcodes würde weiter ein vollständiges und unabhängiges Code-Audit der Software ermöglichen, wie dies z.B. mit dem

sicherheitskritischen Open Source Projekt VeraCrypt [103] durchgeführt wurde. Es existieren zwar mit dem Replicant Projekt [106] eine Open Source Version der VendorRIL für verschiedene ältere Samsung SP und mit Osmocom [52] ein Open Source Softwarestack für den BB einiger alter Featurephones, aber wie aus der Beschreibung zu entnehmen ist, sind diese nicht für moderne SPs verfügbar. Abgesehen davon würden diese Softwarestacks, wenn sie für aktuelle Geräte verfügbar währen, einen enormen Wartungsaufwand mit sich bringen, da sie für jedes Update des OS vom Nutzer oder der Open Source Gemeinschaft angepasst werden müssten.

6.4.4 App basierende Abwehrmaßnahmen

App basierende Abwehrmaßnahmen versuchen, über das Standardverhalten von Android in Bezug auf Audioaufnahmen (siehe Kapitel 5.1.3), einen Schutz vor der Audio-Bypass Bedrohung zu bieten. Da Android derzeit die simultanen Zugriffe mehrerer Anwendungen auf das Mikrofon nicht unterstützt, wird mittels einer speziellen App das Mikrofon belegt. Diese App verhindert durch die dauerhafte Verwendung der für das Mikrofon zuständigen Systemkomponente effektiv, dass eine weitere App Zugriff auf diese erhält (sehr ähnlich zu einem Denial of Service Angriff).
Dieser Ansatz bietet eine für den Nutzer sehr transparente Lösung, hat jedoch mehrere Nachteile. Diese App läuft mit geringeren Rechten als Systemkomponenten und kann daher das Mikrofon auch nur gegen andere Apps mit demselben Grad an Berechtigungen schützen. Sollte eine Systemkomponente mit höheren Rechten das Mikrofon anfragen, kann das System entscheiden, den Zugriff der Block-App zu beenden, um das Mikrofon der Systemkomponente zur Verfügung zu stellen. Dieses Verhalten ist hochgradig von dem verwendeten SP Modell sowie der vom Hersteller durchgeführten Modifikationen abhängig.
Dieses Verhalten konnte in verschiedenen Tests mit Apps aus dem Google Playstore nachempfunden werden. Es wurde keine App gefunden, die auf einem Nexus 5X mit Android 7.1.2 das Mikrofon vor dem Zugriff durch Systemkomponenten beschützen konnte. Für den Test wurden die Schutzmaßnahmen der verschiedenen Apps aktiviert und es wurde ein Anruf aus der systemeigenen Telefon-App gestartet. Da der für ein Telefongespräch benötigte Softwarestack eine Systemkomponente ist, die mehr Rechte als eine vom Nutzer installierte App besitzt, konnte erfolgreich auf die Audiodaten des Mikrofons zugegriffen werden.
Auf Apps basierende Schutzmaßnahmen können gerade für Angriffe, die selbst auf Apps basieren, einen gewissen Schutzfaktor bieten. Sollten jedoch Teile oder das gesamte System unter die Kontrolle des Angreifers gelangen, können sie nur einen geringen Schutz gegen einen Audio-Bypass Angriff bieten.

7 Bewertung

In diesem Kapitel wird die Bewertung der Bedrohungs- und Schwachstellenanalyse der Audio-Bypass Bedrohung vorgenommen. Hierbei wird auf die Bedrohung durch die verschiedenen vorgestellten Angriffe eingegangen und ihr Risiko in Bezug auf einen Audio-Bypass Angriff evaluiert. Abgeschlossen wird das Kapitel durch Empfehlungen zu Gegenmaßnahmen und Verhaltensweisen, die einen Anwender vor der Audio-Bypass Bedrohung schützen können.
Zur Bewertung dieser Angriffe muss zwischen gezielten und breitflächigen Angriffen unterschieden werden. Breitflächige Angriffe richten sich gegen eine große Gruppe von Zielen und laufen meist zu großen Teilen automatisch ab. Eine Audio-Bypass Bedrohung ist hier zwar gegeben, allerdings ist fragwürdig, ob diese gezielt zum Abhören von Gesprächsinformationen im Umfeld des Smartphone (SP) eingesetzt wird. Angriffe dieser Art werden oft zum Tracking größerer Personengruppen für Werbe- und Marketingzwecke verwendet. Hierbei werden die angefertigten Audioaufnahmen vollautomatisch auf gewisse Audiosignale durchsucht, die es ermöglichen, das Ziel des Angriffs anhand des Standortes der Quelle des entsprechenden Audiosignals örtlich zu lokalisieren [40].
Im Kontrast hierzu stehen gezielte Angriffe auf einzelne Personen oder kleinere Personengruppen. Hierbei werden die Ziele aufgrund ihres Informationskontextes explizit ausgewählt und gezielt angegriffen [83]. Da bei dieser Art von Angriff die Audio-Bypass Bedrohung genutzt wird, um an Gesprächsinformationen im Umfeld des Anwenders zu gelangen, müssen die angefertigten Audioaufnahmen von einem Menschen auf interessante Informationen überprüft werden.
Für die Bewertung dieser Angriffe muss das Schutzbedürfnis der Informationen im Umfeld des Anwenders betrachtet werden. Da die Schwachstellenanalyse die Audio-Bypass Bedrohung behandelt, ist das Schutzziel die Vertraulichkeit der Audioinformationen im Umfeld des SPs. Das Schutzbedürfnis wird in drei Kategorien aufgeteilt: Normal, Hoch und sehr Hoch [45].
In die Kategorie "Normal" fallen alle Informationen, bei denen der Verlust der Vertraulichkeit begrenzte Schadensauswirkungen hat. In diese Kategorie fallen z.B. die persönlichen Informationen der meisten Privatanwender (Datenschutz) sowie die Informationen aus den meisten beruflichen Umfeldern.
In die Kategorie "Hoch" fallen Informationen, bei denen der Verlust der Vertraulichkeit eine beträchtliche Schadensauswirkung hat. In diese Kategorie fallen z.B. Informationen von Berufsgruppen, die auf eine gewisse Geheimhaltung

angewiesen sind (Juristen, Journalisten, Aktivisten, usw.), aber auch Informationen aus dem polizeilichen Ermittlungsbereich können in diese Kategorie fallen. Die letzte Kategorie ist "sehr Hoch". Bei dem Verlust der Vertraulichkeit von Informationen aus dieser Kategorie ist mit katastrophalen Schadensauswirkungen zu rechnen, die weitreichende Folgen nach sich ziehen können. In diese Kategorie fallen z.B. Informationen von nationalen und internationalen Politikern oder Managern aus dem internationalen Industriebereich. Aber auch Informationen aus dem privaten Umfeld dieser Personen können in diese Kategorie fallen.

Im Folgenden werden die Auswirkungen der vorgestellten Angriffe und das daraus resultierende Risiko für diese Gruppen beschrieben.

7.1 Baseband

Angriffe, die sich gegen das Baseband-Prozessor (BB) eines SP richten, erfordern ein hohes Maß an Wissen, Equipment und Vorbereitung. Das Angriffspotential, das von einem Angreifer benötigt wird, um einen Angriff in diesem Bereich zu identifizieren und durchzuführen, ist jedoch in den vergangen Jahren enorm gesunken. Zwar sind auch die Abwehrmaßnahmen verstärkt worden, aber schon die Tatsache, dass mit relativ geringen Kosten (ca. 500€) und einem Open Source Softwarestack eine Base Transceiver Station (BTS) in Betrieb genommen werden kann, hat die Möglichkeit eines Angriffs auf diesen Teil des SPs erheblich vergrößert. Trotzdem ist der Aufwand, der benötigt wird, ein SP über den BB anzugreifen, nicht zu unterschätzen. Die frei verfügbare Hardware hat in den meisten Fällen eine sehr schwache Sendeleistung und Modelle mit mehr Kapazität kosten wesentlich mehr Geld und sind auch heute noch teilweise nicht frei verfügbar.
Weiter hat die Analyse gezeigt, dass ein Angriff auf das BB für einen Audio-Bypass alleine nicht ausreicht. Der Remote Code Execution (RCE) Angriff auf den BB benötigt mindestens einen weiteren Angriff, um die Kontrolle über das Mikrofonsignal des SP übernehmen zu können, da der BB in modernen Architekturen keine direkte Kontrolle über den SLIMbus mehr besitzt. Trotzdem besteht weiterhin eine passive Bedrohung durch den Angriff auf den BB, da dieser während eines Telefongesprächs Zugriff auf die Audiodaten des Mikrofons hat. Dies könnte dazu genutzt werden, die Audiodaten von Telefongesprächen an den Angreifer zu übermitteln.
Auch ist ohne einen weiterführenden Angriff keine Persistenz auf dem SP möglich, da der Softwarestack des BB bei jedem Start auf seine Integrität geprüft wird. Ohne einen weiterführenden Angriff bedeutet das, dass der Angriff nach jedem Neustart des SP erneut durchgeführt werden muss damit der Angreifer in Kontrolle des BB bleibt. Der Aufwand hierfür ist aufgrund der für den Angriff

benötigten Hardware und der Nähe zum Ziel erheblich.
Für die Bewertung des Risikos in Tabelle 7.1 ist anzumerken, dass der RCE-Angriff eine Voraussetzung für den Angriff auf das Shared Memory Device (SMD) ist, da dieser ohne die vorherige Kontrolle des BB nicht möglich ist.

Angriff	Angriffsp.	Risiko		
RCE-Angriff	Hoch (26)	Vernachlässigbar	Unerwünscht	Unerwünscht
SMD-Angriff	Mittel (21)	Tolerierbar	Tolerierbar	Inakzeptabel
Schutzbedürfnis		Normal	Hoch	sehr Hoch

Tabelle 7.1: Risiko durch Angriffe auf den Baseband-Prozessor

7.1.1 Betrachtung nach Schutzbedürfnis

normales Schutzbedürfnis:
Für Benutzer mit einem normalen Schutzbedürfnis sind Angriffe auf den BB im Kontext der Audio-Bypass Bedrohung zu vernachlässigen. Neben dem hohen technischen Aufwand sind diese Angriffe auch sehr Hardwareplattform spezifisch, dies bedeutet, dass ein Angriff für jede separate Hardwareplattform einzeln identifiziert und entwickelt werden muss. Durch diese technischen Einschränkungen gibt es immer noch, im Vergleich zu Android, wenige Sicherheitsexperten, die sich mit diesem Themengebiet beschäftigen. Dies führt dazu, dass nur sehr wenige Angriffe auf das BB öffentlich bekannt sind. Sie werden meistens auch zuerst an die Hersteller gemeldet, bevor sie der allgemeinen Öffentlichkeit zugänglich gemacht werden.
Für einen Angreifer, der das benötigte Angriffspotential für diese Art von Angriffen aufweist, ist meistens der Aufwand nicht verhältnismäßig, um an nicht sicherheitskritische Informationen zu gelangen. Für die meistens privaten Nutzer, die in diese Schutzbedürfnisgruppe fallen, ist eine Bedrohung über einen Angriff auf den BB in Bezug auf die Audio-Bypass Bedrohung zu vernachlässigen.
Es existieren aber auch breitflächige Angriffe auf den BB. Diese basieren jedoch nicht auf der Ausnutzung einer Softwareschwachstelle, sondern bedienen sich der verschiedenen durch den 3GPP Standard gebotenen Funktionalitäten. Ein Beispiel ist der Aufbau sogenannter Bewegungsprofile. Hierbei wird durch den Angreifer mittels einer BTS ein Mobilfunknetzwerk aufgebaut, das dem SP des angegriffenen Anwenders suggeriert, das Netz mit der besten Signalqualität zu sein. Nachdem sich das SP in dieses Netz eingewählt hat (SP wählen in der Regel das Netz mit der besten Signalqualität), erhält der Angreifer Zugriff auf die verschiedenen, eindeutig identifizierenden Informationen des SP. Hiermit

ist er in der Lage, dieses SP auch längerfristig eindeutig zu identifizieren und er kann eine Datenbank über die SP in seinem Netzwerk anlegen. Aufgrund der örtlichen Begrenzung eines solchen Netzwerks gibt diese Datenbank Aufschluss über die Dauer und die Regelmäßigkeit, mit der sich ein bestimmtes SP in dem überwachten Gebiet aufhält. Diese Art von Angriff wurde nicht in dieser Arbeit besprochen, stellt aber eine der größten Bedrohungen für durchschnittliche Anwender in Bezug auf den BB dar.

hohes Schutzbedürfnis:
Für Nutzer, deren Informationen ein hohes Schutzbedürfnis haben, stellt der Audio-Bypass Angriff über den BB eine größere Gefährdung dar. In diesem Umfeld werden Informationen besprochen, bei denen der Verlust der Vertraulichkeit über einen Audio-Bypass Angriff weitreichende Folgen haben kann.
In diesem Bereich beginnt ein Angriff über den BB an Bedeutung zu gewinnen, da er schwer zu entdecken ist, wenige Spuren hinterlässt und es keiner Nutzerinteraktion für einen erfolgreichen Angriff bedarf. Diese Art von Angriff ist besonders attraktiv, wenn das Ziel bereits herkömmliche Schutzmaßnahmen zur Absicherung des Androidsystems verwendet. Ein Angreifer, der in der Lage ist, über den BB einen Angriff auf das SMD durchzuführen, erhält dadurch hardwarenahen Lese- und Schreibzugriff auf den Hauptspeicher des Applikationsprozessors und kann dies nutzen um sämtliche auf Android basierenden Schutzmaßnahmen zu umgehen.
Bei Informationen, die in diese Schutzklasse fallen, ist es schwer, eine genaue Beurteilung zu treffen, ob der Aufwand im Vergleich zu dem Nutzen, der aus den abgegriffenen Informationen gezogen werden kann, gerechtfertigt ist. Es ist klar, dass Informationen, die in diese Schutzklasse fallen, auch Angreifer mit höherem Angriffspotential anziehen können. Es ist jedoch hochgradig von den angegriffenen Nutzern und ihren Informationen abhängig, ob sich der Aufwand für solch einen Angriff rentiert.
Es ist davon auszugehen, dass ein hoch technologisierter Angreifer bereits einen entsprechenden Angriff besitzt und eine opportunistische Möglichkeit nutzen würde, um ihn einzusetzen. Jedoch ist es fragwürdig, ob ein spezifischer Angriff explizit entwickelt werden würde, um ein Ziel in dieser Schutzklasse zu attackieren.

sehr hohes Schutzbedürfnis:
Der Audio-Bypass Angriff über den BB stellt eine besondere Bedrohung dar für Nutzer, deren Informationen ein sehr hohes Schutzbedürfnis besitzen. Es ist davon auszugehen, dass diese Nutzer ein speziell gehärtetes SP einsetzten und die zusätzlich Unterstützung von Fachpersonal haben, das sicherstellen sollen, dass die verwendeten Schutzmaßnahmen vollständig gewartet und funktionstüchtig sind. Hier entfaltet ein Angriff über den BB sein volles Potential.

Ein Angriff über das BB ist von Seiten des Applikationsprozessors (der Android treibt) nahezu nicht feststellbar, da einfach keine entsprechende Schnittstelle besteht, um den Zustand des BB zu überprüfen. Dieses Problem wird weiter durch die Art verschärft, in der BB angegriffen werden. Aufgrund der Singnaturprüfung während des Starts müssen alle Codemanipulationen im flüchtigen Speicher des BB vorgenommen werden, was dafür sorgt, dass nach einem Neustart keinerlei Spuren eines erfolgreichen Angriffs zurückbleiben. Dies verhindert effektiv die forensische Analyse des Angriffs, nachdem er bemerkt wurde.
Es kann davon ausgegangen werden, dass Nutzer, deren Informationen in diese Schutzklasse fallen, eine entsprechende Schulung in der Verwendung ihrer SPs erhalten haben. Aber auch dies bietet keinerlei Schutz gegen einen Angriff über den BB, da keine Nutzerinteraktion benötigt wird. Auch die Abschirmung dieser Person von der allgemeinen Öffentlichkeit stellt kein Hindernis dar, da einerseits die Mobilfunkschnittstelle eine sehr große Reichweite besitzt, und andererseits viele Informationen über das SP mittels einer BTS gesammelt werden können (z.B. Marke und Modell des verwendeten SP).
Offensichtlich ziehen Informationen dieser Art Angreifer an, die ein hinreichend großes Angriffspotential besitzen, um solch einen Angriff durchzuführen. Darüber hinaus sind sie wohl auch in der Lage, nicht nur ein bösartiges Mobilfunknetz aufzuspannen, sondern gleich eine ganze Reihe solcher Netzwerke zu etablieren. Dies würde das Problem der Persistenz bei Angriffen über den BB beheben und eine rasche Neuinfektion nach dem Neustart des Gerätes ermöglichen.

7.2 Android

Android ist in den vergangenen Jahren durch kontinuierliche Verbesserungen am System wesentlich resistenter gegen Angriffe geworden. Gerade die Architekturänderungen, die mit Android 7.0 eingeführt wurden sowie die stark ausgeweitete Nutzung von SELinux Domains, haben die Abwehr des Systems wesentlich verstärkt. Dennoch ist Android, wie jedes andere Betriebssystem, nicht immun gegen Angriffe und es werden immer wieder neue Angriffe gefunden, die sämtliche Schutzmaßnahmen umgehen können.
Aufgrund der sehr variablen Anwendungsfälle für SPs und der von den Herstellern vorangetriebenen Datenaggregation auf den Geräten (Mail, Messenger, Kalender, Kontakte, Bilder, usw) sind sie ein sehr attraktives Ziel für Angreifer. Inzwischen existiert eine große Anzahl an verschiedenen Schadprogrammen, wie sie aus dem Desktop-PC Bereich bekannt sind, auch für Android (z.B. Onlinebanking-Betrug oder erpresserische Datenverschlüsselung). Durch diese Entwicklung gibt es heutzutage sehr viele verschiedene Angriffe auf SPs und

Softwareentwickler, die sich auf diesem Bereich spezialisiert haben. Dies sorgt dafür, dass, im Vergleich zu Angriffen auf den BB, es möglich ist, schneller das benötigte Angriffspotential zu erreichen, um Angriffe auf Android durchführen zu können.
Für die Sicherheit von Android sind aber nicht nur Angriffe auf die Software ein Risiko, sondern auch Angriffe auf den Nutzer selbst. Bei einem Social Engineering Angriff wird der Nutzer auf sozialer Ebene dazu manipuliert, schadhafte Software auf seinem SP zu installieren. Dieser Vorgang kann ohne große technische Mittel sehr viel Kontrolle über das SP ermöglichen (abhängig von den App Permissions der installierten App) und kann dazu genutzt werden, einen Audio-Bypass Angriff durchzuführen.
Um Persistenz auf dem SP zu erreichen, bieten sich dem Angreifer gleich mehrere Möglichkeiten. Neben dem Social Engineering Angriff kann auch mittels RCE-Angriff eine nicht dem System zugehörige App (da diese auf der Systempartition liegt und damit durch die Codesignierung geschützt wird) übernommen werden. Besitzt diese App bereits die Berechtigungen, Audioaufnahmen anzufertigen sowie das Internet zu nutzen, sind alle Voraussetzungen für einen Audio-Bypass Angriff erfüllt und es bedarf keiner weiteren Schritte durch den Angreifer. Sollte die App nicht benötigte App Permissions besitzen, oder es wird mehr Kontrolle über das System benötigt, als über die App Permissions zu erreichen ist, kann ein Privilege Escalation Angriff verwendet werden, um vollständige Kontrolle über das System zu erhalten.

Angriff	Angriffsp.	Risiko		
RCE-Angriff	Mittel (18)	Tolerierbar	Unerwünscht	Inakzeptabel
RIL-Angriff	Mittel (23)	Tolerierbar	Unerwünscht	Inakzeptabel
PE-Angriff	Mittel (20)	Tolerierbar	Unerwünscht	Inakzeptabel
Schutzbedürfnis		Normal	Hoch	sehr Hoch

Tabelle 7.2: Risiko durch Angriffe auf Android

7.2.1 Betrachtung nach Schutzbedürfnis

normales Schutzbedürfnis:
Für Nutzer, deren Informationen ein normales Schutzbedürfnis haben, ist der Audio-Bypass Angriff über Android eine ernstzunehmende Bedrohung. Gerade die niedrige technische Schwelle für den Einstieg in diese Art von Angriffen sorgt dafür, dass die hierfür benötigten Techniken verhältnismäßig weitverbreitet sind. Es existieren Techniken aus dem Marketing Bereich [40], die auf einen Audio-Bypass Angriff setzten, um ein Bewegungsprofil über den

Nutzer anzulegen (siehe Kapitel 5.2.5). Dies stellt einen starken Eingriff in die Privatsphäre des Nutzers dar und beleuchtet zugleich ein weiteres Problem in Bezug auf den Audio-Bypass Angriff.
Da ein eher geringes Angriffspotential für diesen Angriff benötigt wird, stellt er eine ernstzunehmende Gefahr für die Privatsphäre und den Datenschutz privater Nutzer dar. Ohne eine entsprechende Einweisung kann einem Anwender nicht bewusst sein, dass die Erlaubnis einer App, Audioaufnahmen anzufertigen und das Internet zu benutzen, im Hintergrund dazu verwendet werden könnte, den Anwender selbst abzuhören. Hier besteht ein besonderes Problem, da für die alltägliche Nutzung des SP oft Apps verwendet werden, die eine Vielzahl von Funktionalitäten aufweisen und in der Lage wären, ihre Möglichkeiten auf dem SP auch zu missbrauchen. Es liegt im Ermessen des Anwenders, entweder auf diese Funktionalität zu verzichten oder dem Hersteller der entsprechenden App zu vertrauen, dass diese keine ungewollte Funktionalität enthält.
Es zeigt sich, dass selbst für Informationen von normalem Schutzbedürfnis eine ernstzunehmende Bedrohung durch den Audio-Bypass Angriff auf Android besteht. Die Einschätzung dieser Angriffe als akzeptables Risiko (siehe Tabelle 7.2) nach dem Vokabular des Angriffspotentialmodells (siehe Kapitel 2.2.3) ist leicht missverständlich. Hierbei wird es nicht als akzeptabel angesehen, dass die Privatspähre von Nutzern angegriffen wird, sondern das Risiko bewertet, das aus dem Verlust der Vertraulichkeit von Informationen mit normalem Schutzbedürfnis entsteht.

hohes Schutzbedürfnis:
Bei Informationen, die ein hohes Schutzbedürfnis haben, stellt der Audio-Bypass Angriff mittels des Android OS offensichtlich ebenfalls ein ernsthaftes Problem dar. Nutzer, deren Informationen in diese Schutzbedürfnisklasse fallen, sind sich zwar oft des Schutzbedürfnisses der besprochenen Informationen bewusst, aber nicht der Möglichkeit, dass ihr SP ein Sicherheitsrisiko darstellen könnte.
Besonders in diesem Bereich werden jedoch Ziele inzwischen auch individuell ausgewählt und attackiert [83]. Hierbei wird eine Mischung aus Angriffen auf die Software (Privilege Esclation) sowie auf den Nutzer (Social Engineering) verwendet. Die niedrige technische Schwelle für die Durchführung eines solchen Angriffs und die Tatsache, dass viele Angriffe öffentlich verfügbar sind, kombiniert mit der Updateproblematik vieler Hersteller, führt zu einer schwerwiegenden Bedrohung im Bereich von gezielten Audio-Bypass Angriffen im Android Kontext auf Informationen mit einem hohen Schutzbedürfnis. Wie in Kapitel 5.1.6 gesehen, erhält ein Großteil der aktuell im Einsatz befindlichen Android SP keine oder nur sehr veraltete monatliche Sicherheitsupdates. Auf der Basis eines veralteten SP ist die Bedrohung durch einen Audio-Bypass desaströs.

Aufgrund des verhältnismäßig geringen Aufwandes für eine solche Art von Angriff, vergrößert sich die Gruppe von potentiellen Angreifern enorm. Im Gegensatz zu Angriffen auf den BB mittels einer BTS braucht der Angreifer weder spezielle Hardware, noch muss er sich in der Nähe des Ziels aufhalten. Dies senkt nicht nur das für den Angriff benötigte Angriffspotential, sondern auch die für einen Angriff benötigten finanziellen Mittel.
Wie sich in der Bewertung in Tabelle 7.2 widerspiegelt, ist die Höhe des Risikos durch einen Audio-Bypass im Android Kontext für Informationen mit einem hohen Schutzbedürfnis unerwünscht groß. Dies spiegelt wider, dass ein erheblicher Aufwand betrieben werden muss, um entsprechende Angriffe abzuwenden; dies beinhaltet grundlegende Maßnahmen wie regelmäßige Updates und Nutzerschulungen, sollte aber auch weiterführende Maßnahmen (siehe Kapitel 6) beinhalten, um das Risiko weiter zu verringern.

sehr hohes Schutzbedürfnis:
Für SP von Anwendern, deren Informationen ein sehr hohes Schutzbedürfnis aufweisen, wird davon ausgegangen, dass ein Mindestmaß an Sicherheitsvorkehrungen in Bezug auf das SP getroffen wurden. Hierzu zählt, dass ein SP Modell gewählt wurde, das die aktuellsten monatlichen Sicherheitsupdates zeitnah erhält und möglichst frei von unnötiger Bloatware gehalten wird (siehe Kapitel 5.1.5). Der Anwender des SP ist sich der Risiken in Bezug auf die Installation von Apps auf dem SP bewusst und hält sich an strikte Regeln, die den unerlaubten Zugriff einer App auf das Mikrofon verhindern sollen.
Doch selbst in diesem idealen Szenario ist Android nicht in der Lage, einen adäquaten Schutz für Informationen zu bieten, die ein sehr hohes Schutzbedürfnis aufweisen.
Es ist darauf hinzuweisen, dass dies nicht unbedingt ein Versagen von Android darstellt. Es ist vielmehr fragwürdig, ob es überhaupt möglich ist, ein System zu entwickeln, das den Ansprüchen eines modernen SPs genügt und das zugleich in der Lage ist, die Sicherheitsbedürfnisse von Informationen mit einem sehr hohen Schutzbedürfnis zu befriedigen.
Die Bewertung in Tabelle 7.2 zeigt eine inakzeptabel hohe Risikoeinschätzung für den gesamten Android Kontext. Diese Einschätzung bedeutet nicht, dass in solch einem Umfeld keine Android SP eingesetzt werden können. Es zeigt jedoch auf, dass in Bezug auf die Audio-Bypass Bedrohung klare Regeln aufgestellt werden müssen, wie mit den SP umzugehen ist. Diese müssen strikt von sensitiven Audioumgebungen isoliert werden.

7.3 Empfehlungen

In diesem Abschnitt werden die verschiedenen möglichen Schritte besprochen, die geeignet sind, um die Resistenz gegen eine Audio-Bypass Bedrohung

zu erhöhen. Zuerst werden ganz allgemeine, für alle Schutzbedürfnisklassen zutreffende Empfehlungen ausgesprochen, anschließend werden weitere mögliche Maßnahmen für die verschiedenen Schutzbedürfnisse individuell hervorgehoben.

7.3.1 Allgemeine Empfehlungen

Diese Empfehlungen beziehen sich nicht ausschließlich auf eine Audio-Bypass Bedrohung, sondern erhöhen die Resistenz des SP gegen jegliche Art von Angriffen.
Wie schon in Kapitel 6.1 herausgestellt wurde, sind Softwareupdates die wahrscheinlich wichtigste Abwehrmaßnahme. Daher kann einem SP Nutzer nur empfohlen werden, bei der Anschaffung eines SP sich vorher über die Updatepolitik der verschiedenen Hersteller zu informieren und ein Modell zu wählen, bei dem auch eine längerfristige Versorgung mit Sicherheitsupdates gewährleistet ist. Das Risiko, das durch nicht aktuelle Software entsteht, besonders bei Personen, die mit gezielten Angriffen auf ihre Informationen rechnen müssen, ist als extrem hoch einzustufen. Insbesondere die Erfahrungen der letzten Jahre haben gezeigt, dass öffentlich bekannt gewordene Angriffe oft noch lange nach dem Erscheinen eines diesen Angriff mitigierenden Sicherheitsupdates dazu verwendet werden, breitflächige Angriffe auf alle immer noch verwundbaren Systeme durchzuführen [112].
Es müssen aber nicht nur Updates für das Gerät angeboten werden, diese müssen auch durch den Nutzer regelmäßig installiert werden. Ohne ein entsprechendes Nutzerverhalten verlieren viele Schutzmaßnahmen Teile oder den gesamten Schutz, den sie dem System bieten können. Daher ist eine Schulung der Anwender unerlässlich, die nicht nur den korrekten Umgang mit SP vermittelt, sondern den Nutzer auch für sicherheitskritische Vorgänge sensibilisiert. Warnungen, die dem Nutzer vom System angezeigt werden, können nur dann Erfolg haben, wenn der Nutzer die dargestellte Information auch verstehen kann.
Abschließend muss den meisten Nutzern davon abgeraten werden, das SP zu rooten. Dieser Vorgang ermöglicht dem Nutzer zwar eine wesentlich größere Kontrolle über das OS, aber er verliert im Gegenzug einige sehr wichtige von Android gebotene Schutzmaßnahmen.

7.3.2 Empfehlungen basierend auf Schutzbedürfnisklassen

normales Schutzbedürfnis:
Neben den allgemeinen Empfehlungen kann der durchschnittliche Anwender,

der in der Regel in diese Schutzbedürfnisklasse fällt, einige weitere Schritte beachten, um sich vor einem Audio-Bypass Angriff zu schützen.
Gerade in diesem Bereich ist eine Sensibilisierung des Anwenders für sicherheitstechnische Aspekte verschiedener Vorgänge besonders wirksam. Schon eine aufmerksame Auswahl der installierten Apps kann das Risiko eines Audio-Bypass Angriffs stark senken. Dieser Schutz kann weiter ausgebaut werden, wenn sich der Anwender über die verschiedenen Schutzmaßnahmen bewusst ist und versteht, vor welche Art von Angriffen sie ihn schützen.
Nutzer sollten von dem neuen App Permission Modell (seit Android 6) Gebrauch machen und die Erlaubnis zur Verwendung des Mikrofons nur an Apps vergeben, bei denen sie die Funktionalität, die auf das Mikrofon angewiesen ist, auch wirklich nutzen.

hohes Schutzbedürfnis:
Für Nutzer, deren Informationen ein hohes Schutzbedürfnis aufweisen, gibt es mehrere Möglichkeiten, das Risiko eines gezielten Audio-Bypass Angriffs zu senken.
Neben der Einhaltung der allgemeinen Empfehlungen kann in diesem Bereich der Einsatz einer besonders gehärteten Android Version empfohlen werden. Aufgrund der Komplexität dieses Gebietes sollte in Betracht gezogen werden, diese Leistung als Service bei einem entsprechenden Unternehmen einzukaufen [54]. Diese Maßnahme alleine reicht jedoch leider nicht aus, das Risiko vollständig zu mitigieren.
Es kann in Erwägung gezogen werden, eine entsprechende Schutzmaßnahme gegen einen Angriff auf den BB mittels einer BTS zu installieren. Dies kann zwar einen Angriff nicht verhindern, würde jedoch ein Indiz liefern, dass gerade ein Angriff stattfinden könnte.
Schon in dieser Schutzklasse ist zu empfehlen, dass während Gesprächen, in denen ein SP nicht unbedingt benötigt wird (z.B. Kalender oder Notizen), das SP für die Zeit des Gesprächs außerhalb des Raums gelagert wird.

sehr hohes Schutzbedürfnis:
In dieser Schutzbedürfnisklasse können nur Empfehlungen zum allgemeinen Umgang mit SP gegeben werden. Eine Senkung des Risikos auf ein akzeptables Level ist nur durch sehr einschneidende Maßnahmen zu erreichen. Konsequenterweise muss entweder ein Gerät verwendet werden, das keine Audioaufnahmen anfertigen kann (Mikrofone entfernt) oder das Gerät muss von sämtlichen sensitiven Audiokontexten isoliert werden.
Aufgrund des hier dargestellten Risikos, zusammen mit der Tatsache, dass keine Gegenmaßnahme (außer der Isolierung) einen vollständigen Schutz bieten kann, muss empfohlen werden, den Einsatz von SP in dieser Umgebung insgesamt zu überdenken.

Der Einsatz einer ständigen Überwachung des Mobilfunknetzes zur Identifikation neuer, möglicherweise bösartiger Netze, besonders an Orten, an denen viele sensitive Informationen besprochen werden, kann empfohlen werden. Dies verhindert zwar keine Angriffe, kann aber als Frühwarnsystem für mögliche Angriffe eingesetzt werden.

8 Fazit

Im Verlauf dieser Arbeit hat sich gezeigt, dass ein gewisses Grundrisiko durch die Audio-Bypass Bedrohung für Informationen in allen Schutzbedürfnisklassen besteht. Die Schwachstellenanalyse verdeutlicht, dass für die praktische Durchführung eines Audio-Bypass Angriffs schon ein mittleres Angriffspotential ausreicht und Audio-Bypass Angriffe auch bereits für breitflächige Angriffe auf die Privatsphäre von Smartphone (SP) Nutzern verwendet werden [40].
Android als SP Betriebssystem hat in den vergangenen Jahren in Bezug auf Sicherheit starke Fortschritte gemacht [16], hat jedoch extreme Altlasten durch die hohe Zahl an Geräten mit veralteten Versionen von Android, die keine Updates mehr erhalten. Diese Updateproblematik betrifft aber nicht nur Systemupdates, sondern auch die monatlichen Sicherheitupdates, da viele Hersteller keine oder nur sehr veraltete Sicherheitsupdates für ihre Geräte bieten. Die Sicherheit eines Systems ist allerdings stark von der Aktualität der Software abhängig. Daher bietet diese Situation denkbar schlechte Voraussetzungen für den Umgang mit sensitiven Informationen.
Die durch die Hersteller der SPs durchgeführten proprietären Modifikationen an Android sind nicht nur mitverantwortlich für die Updateproblematik, sondern stellen an sich ein schwer einzuschätzendes Sicherheitsrisiko dar. Es ist schwer festzustellen, ob die von den Herstellern in das System integrierten Softwarekomponenten weitere ungewollte Funktionalitäten beinhalten. Darüber hinaus werden weder Quellcode noch Details der verwendeten Hardwarearchitektur von den Herstellern öffentlich zur Verfügung gestellt, was eine Analyse durch unabhängige Sicherheitsforscher weiter erschwert. Die Vergangenheit hat gezeigt, dass gerade in den von den Hersteller stammenden Softwarekomponenten besonders viele Fehler zu finden waren [62, 122]. Open Source ist keine Lösung für alle sicherheitsrelevanten Probleme. Es kann auch in Open Source Software bösartiger Quelltext versteckt werden. Aber die Wahrscheinlichkeit, dass diese Funktionalität entdeckt wird, steigt mit der Anzahl der Menschen, die Zugang zum Quelltext haben, um eine ausführliche Analyse ohne den ansonsten benötigten Reverse Engineering Aufwand durchzuführen.

Durchschnittliche Nutzer, deren Informationen ein normales Schutzbedürfnis aufweisen, können mit einigen mit geringem Aufwand verbundenen Schritten die Resistenz gegen die meisten Angriffe auf ihr SP stark verbessern. Wie in den Empfehlungen in Kapitel 7.3 beschrieben, sind die Verwendung eines mit regelmäßigen Sicherheitsupdates versorgten SP und der bewusste Umgang

mit dem SP selbst (vor allem bei der Entscheidung, welche Apps installiert werden) die wichtigsten Schutzmaßnahmen für alle Anwender und sollten für den privaten Gebrauch von SP vollkommen ausreichen. Sollten doch einmal Informationen besprochen werden, die ein höheres Schutzbedürfnis aufweisen, so kann empfohlen werden, das SP entsprechend vom sensitiven Audiokontext zu isolieren.

Ein Umfeld mit Informationen, die ein hohes Schutzbedürfnis haben, ist für Android eine Grauzone. Android ist ein hoch komplexes System, das auf eine möglichst vollständige Vernetzung des Gerätes mit verschiedenen Kommunikationsschnittstellen hin optimiert wurde. Die Vielfalt der Schnittstellen sowie die Komplexität des Systems erschweren nicht nur die Konzeption von Abwehrmaßnahmen, sondern erhöhen auch den Aufwand, der für eine Evaluation dieser Abwehrmaßnahmen benötigt wird. Dies wird durch die Verwendung von proprietärem Komponenten noch weiter erschwert.
Für einen Einsatz von Android in diesem Umfeld muss auf jeden Fall ein beträchtlicher Aufwand betrieben werden, um das Risiko eines Angriffs zu vermindern. Der Einsatz eines gehärteten Android zusammen mit weiteren Abwehrmaßnahmen ist stark zu empfehlen.

Im Bereich der Anwender, deren Informationen ein sehr hohes Schutzbedürfnis aufweisen, ergeben sich jedoch wesentlich größere Risiken. Da in diesem Bereich auch mit gezielten Angriffen durch Angreifer mit sehr hohem Angriffspotential gerechnet werden muss, reichen die Android-Schutzvorkehrungen hier nicht aus. Die Erfahrungen der letzten Jahre haben gezeigt, dass beispielsweise verschiedene Geheimdienste über eine ganze Bibliothek an Angriffen für noch nicht veröffentlichte Schwachstellen verfügen. Sollte der Angreifer über eine entsprechende Kombination aus Angriffen verfügen, die auf nur ihm bekannten Schwachstellen basieren, kann weder die Einhaltung aller Sicherheitsauflagen durch den Nutzer noch ein aktuelles System den Angriff verhindern.
Es bleibt fragwürdig, ob es überhaupt möglich ist, Android so abzusichern, dass das Risiko für einen Audio-Bypass Angriff im Kontext von Informationen mit sehr hohem Schutzbedürfnis in einen tolerablen Rahmen fällt. Allgemein muss davon abgeraten werden, ein Android SP in solch einer Umgebung einzusetzen.

9 Anhang

9.1 Data Flow Diagram

9.1.1 Legende

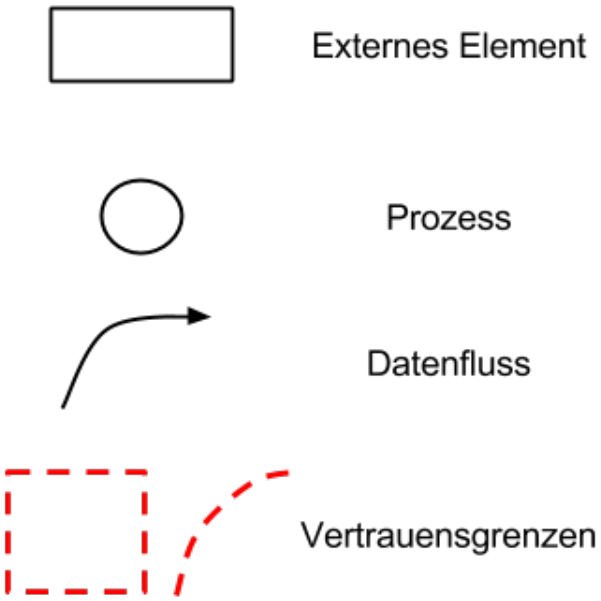

Abbildung 9.1: Data Flow Diagrams Legende

9.1.2 Android Data Flow Diagram

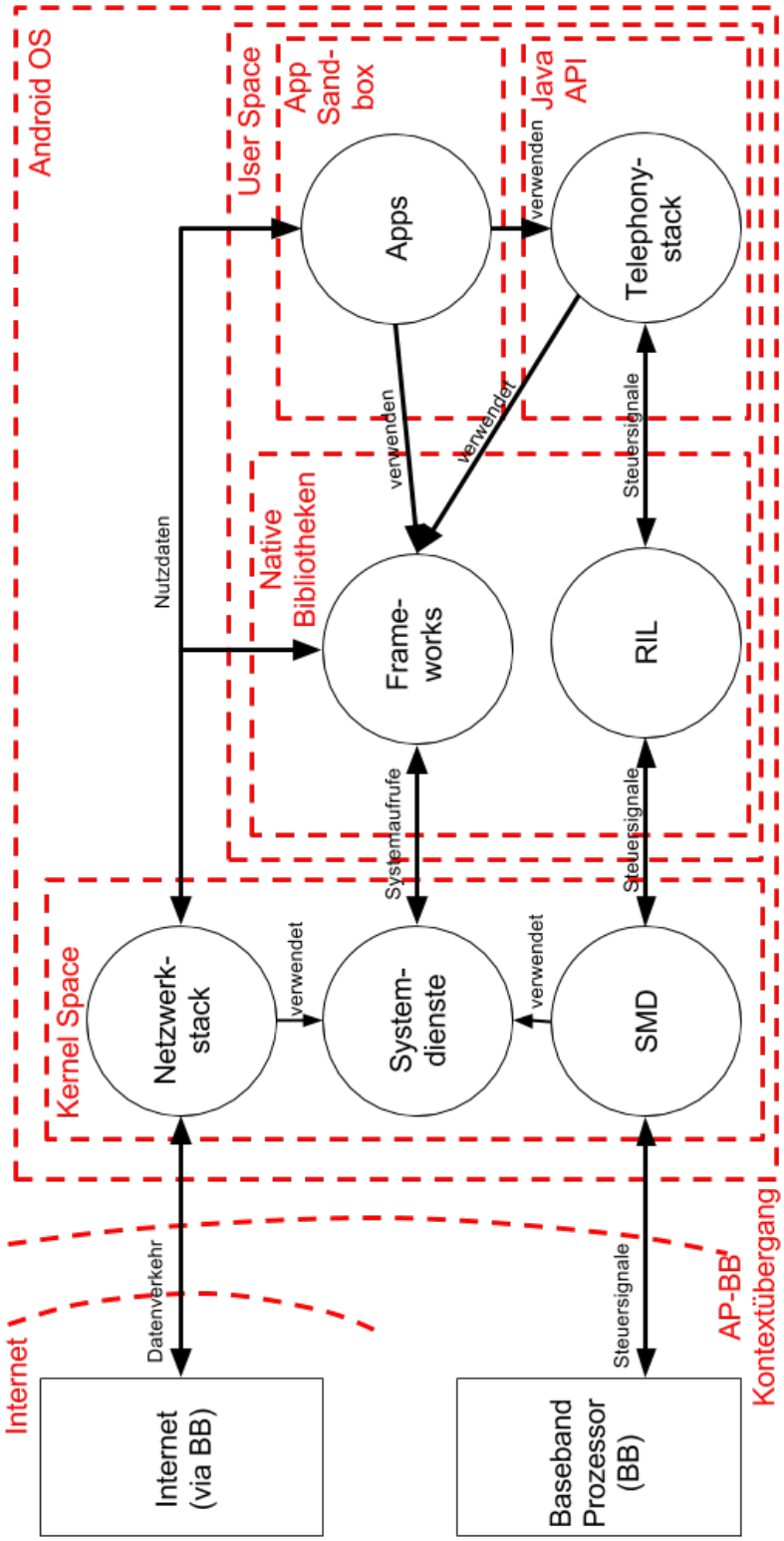

Abbildung 9.2: Data Flow Diagrams des Android Komplex

9.2 Angriffspotential

Benötigte Zeit			
Level	Beschreibung	Identifikation	Ausnutzung
< 10 Minuten	Angriff kann innerhalb von 10 Minuten identifiziert oder ausgenutzt werden	0	1
< 1 Stunde	Angriff kann innerhalb von einer Stunde identifiziert oder ausgenutzt werden	1	2
< 1 Tag	Angriff kann innerhalb von einem Tag identifiziert oder ausgenutzt werden	2	3
< 1 Woche	Angriff kann innerhalb von einer Woche identifiziert oder ausgenutzt werden	3	4
< 1 Monat	Angriff kann innerhalb von einem Monat identifiziert oder ausgenutzt werden	4	5
> 1 Monat	Angriff benötigt mehr als einen Monat, um identifiziert oder ausgenutzt zu werden	5	8
Nicht möglich		*	*

Tabelle 9.1: Benötigte Zeit für die Identifikation und Ausnutzung einer Schwachstelle

Benötigte Fähigkeiten			
Level	Beschreibung	Identifikation	Ausnutzung
Layman	Unausgebildete Person, die kein weiteres fachspezifisches Wissen besitzt (z.B. Anwender einer "Click and Play" Lösung).	0	0
Inter-mediate	Eingewiesene Person, die das System anwenden kann (z.B. Verwendung eines Terminals, um bekannten Angriff zu starten).	1	1
Proficient	Geschulter Anwender eines Systems, der das Verhalten des Systems versteht und die einzelnen Komponenten bedienen kann (z.B. Aufsetzen eines BTS und die Anpassung und Verwendung eines bekannten Angriffs).	2	2
Expert	Experte des Fachgebiets, der Kenntnis des gesamten Systems besitzt und die technischen Beziehungen zwischen den Komponenten versteht (z.B.: Modifikationen am System vornehmen oder Schwachstellen identifizieren und Code zur Ausnutzung dieser Schwachstellen entwickeln).	5	4

Tabelle 9.2: Benötigte Fähigkeiten für die Identifikation und Ausnutzung einer Schwachstelle

Benötigtes Wissen über das Ziel			
Level	Beschreibung	Identifikation	Ausnutzung
None	Keine Informationen über das Ziel jenseits von Allgemeinwissen	0	0
Public	Öffentlich verfügbare Informationen über das Ziel (z.B. wissenschaftliche Veröffentlichungen oder öffentliche Dokumentationen)	2	2
Restricted	Öffentlich verfügbare, aber restriktierte Informationen über das Ziel (z.B. Bugreports und Schwachstellenbeschreibungen oder Service-Dokumente)	3	3
non Public	Nicht öffentlich verfügbare Informationen (z.B. System oder Chipset Dokumentationen), die aber weiter verbreitet sind (z.B. bei Sub-Kontraktoren, Kunden und Service Partnern)	4	4
Sensitive	Informationen, die nur intern vorhanden sind und nicht oder nur mit den engsten Partnern geteilt werden (z.B. Schlüssel zur Signierung von Bootrom).	5	5

Tabelle 9.3: Benötigtes Wissen über das Ziel für die Identifikation und Ausnutzung einer Schwachstelle

Benötigte Zeit mit Zugang zum Ziel			
Level	Beschreibung	Identifikation	Ausnutzung
Kein Zugang	Angriff kann ohne Zugang zum Ziel identifiziert oder ausgenutzt werden	0	0
< 10 Minuten	Angriff kann innerhalb von 10 Minuten, mit Zugang zum Ziel, identifiziert oder ausgenutzt werden	1	1
< 1 Stunde	Angriff kann innerhalb von einer Stunde, mit Zugang zum Ziel, identifiziert oder ausgenutzt werden	2	2
< 1 Tag	Angriff kann innerhalb von einem Tag, mit Zugang zum Ziel, identifiziert oder ausgenutzt werden	3	4
< 1 Woche	Angriff kann innerhalb von einer Woche, mit Zugang zum Ziel identifiziert oder ausgenutzt werden	4	5
< 1 Monat	Angriff kann innerhalb von einem Monat, mit Zugang zum Ziel, identifiziert oder ausgenutzt werden	5	6
> 1 Monat	Angriff benötigt mehr als einen Monat, mit Zugang zum Ziel, um dentifiziert oder ausgenutzt zu werden	6	8
Nicht möglich		*	*

Tabelle 9.4: Benötigte Zeit, mit Zugang zum Ziel, für die Identifikation und Ausnutzung einer Schwachstelle

Benötigte Ausrüstung			
Level	Beschreibung	Identifikation	Ausnutzung
None	Keine Ausrüstung benötigt	0	0
Standard	Weit verbreitete Tools (z.B. Hexagon SDK), Teile des Angriffsziels (z.B. Debugschnittstellen) sowie simple Skripte (z.B. Automatisierung von Prozessen) und öffentlich verfügbare Exploits	1	2
Specialized	Wenig verbreitete Tools (z.B. Qualcomm Emergency USB Downloader), Spezialanwendungen (IDA Pro für ARM) und Gerätschaften (z.B. leistungsstarke BTS), die ohne Restriktionen verfügbar sind. Dies beinhaltet auch größere finanzielle Ausgaben (bis 5000€) sowie aufwändigere Eigenentwicklungen von Skripten und Exploits (Software zum Angriff auf eine Schwachstelle)	4	4
Bespoke	Nicht öffentliche Tools (z.B. Qualcomm QFuse Burner) oder sehr spezialisierte Anwendungen (z.B. Hexagon Decompiler) und Gerätschaften (z.B. IMSI-Catcher oder Kontrolle über einen Mobilfunksendemast eines Netzanbieters), deren Verfügbarkeit meistens restriktiert ist (NDAs oder nur herstellerintern verfügbar). In diese Kategorie fallen auch finanzielle Ausgaben über 5000€	5	6

Tabelle 9.5: Benötigte Ausrüstung für die Identifikation und Ausnutzung einer Schwachstelle

Literaturverzeichnis

Der Inhalt der hier angegebenen URLs wurden zuletzt am 22.05.2017 überprüft..

[1] 3GPP. 3gpp members, . URL http://webapp.etsi.org/3gppmembership/QueryForm.asp. (URL geprüft am: 22.05.17).

[2] 3GPP. 3gpp membership, . URL http://www.3gpp.org/about-3gpp/membership.

[3] 3GPP. At command set for user equipment (ue), March 2017. URL https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1515.

[4] Bushra Aloraini, Daryl Johnson, Bill Stackpole, and Sumita Mishra. A new covert channel over cellular voice channel in smartphones. *arXiv preprint arXiv:1504.05647*, 2015. URL https://arxiv.org/ftp/arxiv/papers/1504/1504.05647.pdf.

[5] Andoird. Audio framwork overview, . URL https://source.android.com/devices/audio/.

[6] Andoird. Android security acknowledgements, . URL https://source.android.com/security/overview/acknowledgements.

[7] Andoird. Ril refactoring, . URL https://source.android.com/devices/tech/connect/ril.

[8] Andoird. Application security, . URL https://source.android.com/security/overview/app-security.html.

[9] Andoird. Application signing, . URL https://source.android.com/security/apksigning/index.html.

[10] Andoird. System and kernel security, . URL https://source.android.com/security/overview/kernel-security.html.

[11] Andoird. Security-enhanced linux in android, . URL https://source.android.com/security/selinux/index.html.

[12] Andoird. Verifying boot, . URL https://source.android.com/security/verifiedboot/verified-boot.html.

[13] Andoird. Update resources, . URL https://source.android.com/security/overview/updates-resources.html.

[14] Andoird. Hal overview, . URL https://source.android.com/devices/#Hardware-Abstraction-Layer.

[15] Andoird. Android security bulletins, . URL https://source.android.com/security/bulletin/index.html.

[16] Andoird. Android security 2016 year in review, March 2017. URL https://source.android.com/security/reports/Google_Android_Security_2016_Report_Final.pdf.

[17] Android. Hal reference. URL https://source.android.com/devices/halref/files.html.

[18] Darcy LaCouvee (androidauthority.com). Fact or fiction: Android apps only use one cpu core. Blogartikel, May 2015. URL http://www.androidauthority.com/fact-or-fiction-android-apps-only-use-one-cpu-core-610352/.

[19] AOSP. Tinyalsa, . URL https://android.googlesource.com/platform/external/tinyalsa/+/master.

[20] AOSP. Android media api code, . URL https://android.googlesource.com/platform/frameworks/base/+/master/media/java/android/media/.

[21] AOSP. Android media api dokumentation, . URL https://developer.android.com/reference/android/media/package-summary.html.

[22] AOSP. Android platform architecture, . URL https://developer.android.com/guide/platform/index.html.

[23] AOSP. Audioflinger code, . URL https://android.googlesource.com/platform/frameworks/av/+/master/services/audioflinger/AudioFlinger.cpp.

[24] AOSP. Audio policy configuration, . URL https://android.googlesource.com/device/lge/bullhead/+/master/audio_policy_configuration.xml.

[25] AOSP. Android audio source dokumentation. Dokumentation, . URL https://developer.android.com/reference/android/media/MediaRecorder.AudioSource.html.

[26] AOSP. Creating a background service. Dokumentation, . URL https://developer.android.com/training/run-background-service/create-service.html.

[27] AOSP. Android supported media formats. Dokumentation, . URL https://developer.android.com/guide/topics/media/media-formats.html.

[28] AOSP. Platform version overview, . URL https://developer.android.com/about/dashboards/index.html.

[29] AOSP. Android normal app permissions. Dokumentation, . URL https://developer.android.com/guide/topics/permissions/normal-permissions.html.

[30] AOSP. Android app permission dokumentation. Dokumentation, . URL https://developer.android.com/reference/android/Manifest.permission.html.

[31] AOSP. Requesting permissions. Dokumentation, . URL https://developer.android.com/guide/topics/permissions/requesting.html.

[32] AOSP. Android audiorecorder dokumentation. Dokumentation, . URL https://developer.android.com/reference/android/media/AudioRecord.html.

[33] AOSP. Binder reference, . URL https://developer.android.com/reference/android/os/Binder.html.

[34] AOSP. Git repositories on android, . URL https://android.googlesource.com/?format=HTML.

[35] AOSP. Vendorril referenz, . URL https://android.googlesource.com/platform/hardware/ril/+/master/reference-ril/.

[36] AOSP. Rilconstants.java, . URL https://android.googlesource.com/platform/frameworks/base/+/master/telephony/java/com/android/internal/telephony/RILConstants.java.

[37] AOSP. Radio interface layer daemon, . URL https://android.googlesource.com/platform/hardware/ril/+/master/rild/rild.c.

[38] AOSP. Android telephony-stack, . URL https://android.googlesource.com/platform/frameworks/opt/telephony/+/master/src/java/com/android/internal/telephony.

[39] Ashish Arora, Rahul Telang, and Hao Xu. Optimal policy for software vulnerability disclosure. *Management Science*, 54(4):642–656, 2008. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.5029&rep=rep1&type=pdf.

[40] Daniel Arp, Erwin Quiring, Christian Wressnegger, and Konrad Rieck. Privacy threats through ultrasonic side channels on mobile devices. *IEEE European Symposium on Security and Privacy*, 2017. URL http://christian.wressnegger.info/content/projects/sidechannels/2017-eurosp.pdf.

[41] Sean Beaupre. Trustnone. -, November 2015. URL http://theroot.ninja/disclosures/TRUSTNONE_1.0-11282015.pdf.

[42] Antutu Benchmark. Top performance chips & market shares, 1h 2016, July 2016. URL http://www.antutu.com/en/doc/107008.htm.

[43] Inc Berkeley Design Technology. Qualcomm's qdsp6 v6: Imaging and vision enhancements via vector extensions, September 2015. URL https://www.bdti.com/InsideDSP/2015/09/30/Qualcomm.

[44] Bitkom. Smartphone-markt: Konjunktur und trends, February 2017. URL https://www.bitkom.org/Presse/Anhaenge-an-PIs/2017/02-Februar/Bitkom-Pressekonferenz-Smartphone-Markt-Konjunktur-und-Trends-22-02-2017-Praesentation.pdf.

[45] BSI. It-grundschutzkatalog schutzbedürfnisklassen, . URL https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzSchulung/WebkursITGrundschutz/Schutzbedarfsfeststellung/Schutzbedarfskategorien/schutzbedarfskategorien_node.html.

[46] BSI. Bsi-standards, . URL https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzStandards/ITGrundschutzStandards_node.html.

[47] BSI. Gefährdungskatalog g 5.96 manipulation von mobiltelefonen, . URL https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/g/g05/g05096.html?nn=6605036.

[48] BSI. Bsi-standard 100-3: Risikoanalyse auf der basis von it-grundschutz, June 2008. URL https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/ITGrundschutzstandards/BSI-Standard_1003.pdf?__blob=publicationFile&v=1.

[49] BSI. It-grundschutz-kataloge (15. ergänzungslieferung), 2016. URL https://download.gsb.bund.de/BSI/ITGSK/IT-Grundschutz-Kataloge_2016_EL15_DE.pdf.

[50] CellularPrivacy. Android-imsi-catcher-detector. Homepage. URL https://github.com/CellularPrivacy/Android-IMSI-Catcher-Detector/wiki.

[51] Erika Chin, Adrienne Porter Felt, Vyas Sekar, and David Wagner. Measuring user confidence in smartphone security and privacy. In *Proceedings of the Eighth Symposium on Usable Privacy and Security - SOUPS '12*. Association for Computing Machinery (ACM), 2012. doi: 10.1145/2335356.2335358.

[52] Open Source Mobile Communications. Open source 3gpp stack. URL http://osmocom.org/.

[53] Open Source Mobile Communications. Qmi (qualcomm msm interface), December 2016. URL https://osmocom.org/projects/quectel-modems/wiki/QMI.

[54] Copperhead. Copperheados, . URL https://copperhead.co/android/.

[55] Copperhead. Technical overview of copperheados, . URL https://copperhead.co/android/docs/technical_overview.

[56] Adrian Dabrowski, Nicola Pianta, Thomas Klepp, Martin Mulazzani, and Edgar Weippl. Imsi-catch me if you can: Imsi-catcher-catchers. In *Proceedings of the 30th annual computer security applications Conference*, pages 246–255. ACM, 2014. URL https://www.sba-research.org/wp-content/uploads/publications/DabrowskiEtAl-IMSI-Catcher-Catcher-ACSAC2014.pdf.

[57] Sajal Kumar Das. *Mobile Terminal Receiver Design*. Wiley, 2016. URL http://www.ebook.de/de/product/27475048/sajal_kumar_das_mobile_terminal_receiver_design.html.

[58] Exploit DB. Offensive security's exploit database archive. URL https://www.exploit-db.com/.

[59] Guillaume Delugre. Reverse engineering a qualcomm baseband. Slides: https://events.ccc.de/congress/2011/Fahrplan/attachments/2022_11-ccc-qcombbdbg.pdf, December 2011. URL https://www.youtube.com/watch?v=IWSCdpAeONA.

[60] CVE Details. Übersicht der gemeldeten rce-angriffe auf android für das jahr 2017, 2017. URL http://www.cvedetails.com/vulnerability-list/vendor_id-1224/product_id-19997/year-2017/opec-1/Google-Android.html.

[61] Adam Donenfeld. Stumping the mobile chipset. Slides: https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEFCON-24-Adam-Donenfeld-Stumping-The-Mobile-Chipset.pdf, 2016. URL https://www.youtube.com/watch?v=vAo76EjL_m4.

[62] Joshua Drake. Stagefright: Scary code in the heart of android. Slides: https://www.blackhat.com/docs/us-15/materials/us-15-Drake-Stagefright-Scary-Code-In-The-Heart-Of-Android.pdf. URL https://www.youtube.com/watch?v=71YP65UANPo.

[63] Dmitry Evtyushkin, Dmitry Ponomarev, and Nael Abu-Ghazaleh. Jump over aslr: Attacking branch predictors to bypass aslr. In *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*, pages 1–13. IEEE, 2016. URL http://www.cs.ucr.edu/~nael/pubs/micro16.pdf.

[64] Google. Android version updates, . URL https://support.google.com/nexus/answer/4457705?hl=en#nexus_devices.

[65] Google. Google now sprachbefehl history. Homepage & Service, . URL https://myactivity.google.com/myactivity?hl=en&utm_source=udc&utm_medium=r&restrict=vaa.

[66] Ben Gras, Kaveh Razavi, Erik Bosman, Herbert Bos, and Christiano Giuffrida. Aslr on the line: Practical cache attacks on the mmu. *NDSS (Feb. 2017)*, 2017. URL http://www.cs.vu.nl/~herbertb/download/papers/anc_ndss17.pdf.

[67] GSMK. Cryptophone 500i, . URL http://www.cryptophone.de/upload/files/46/original/CP500i-Brochure.pdf.

[68] GSMK. Ida processor module for the hexagon (qdsp6v55) processor, . URL https://github.com/gsmk/hexagon.

[69] Mesud Hadžialić, Mirko Škrbić, Kemal Huseinović, Irvin Kočan, Jasmin Mušović, Alisa Hebibović, and Lamija Kasumagić. An approach to analyze security of gsm network. In *Telecommunications Forum Telfor (TELFOR), 2014 22nd*, pages 99–102. IEEE, 2014. URL https://www.researchgate.net/profile/Jasmin_Musovic3/publication/269105419_An_Approach_to_Analyze_Security_of_GSM_Network/links/5480d96e0cf22525dcb6051d.pdf.

[70] heise online. Gsm-zukunft in deutschland, Österreich und der schweiz ist offen, December 2016. URL https://www.heise.de/newsticker/meldung/GSM-Zukunft-in-Deutschland-Oesterreich-und-der-Schweiz-ist-offen-3582914.html.

[71] Hex-Rays. Ida pro (disassembler/debugger). URL https://www.hex-rays.com/products/ida/.

[72] HTC. Htcdev file search. URL http://www.htcdev.com/devcenter/downloads.

[73] IP.access. nanobts. URL http://www.ipaccess.com/en/viper/enterprise-small-cells.

[74] ISO/IEC. Iso/iec 18045:2008: Information technology – security techniques – methodology for it security evaluation, August 2008. URL http://standards.iso.org/ittf/PubliclyAvailableStandards/c046412_ISO_IEC_18045_2008(E).zip.

[75] ISO/IEC. Iso/iec 15408-1:2009: Information technology – security techniques – evaluation criteria for it security, 2009. URL https://www.iso.org/standard/50341.html.

[76] ISO/IEC. Iso/iec 27000:2016: Information technology — security techniques — information security management systems — overview and vocabulary, February 2016. URL http://standards.iso.org/ittf/PubliclyAvailableStandards/c066435_ISO_IEC_27000_2016(E).zip.

[77] Katharina Krombholz, Heidelinde Hobel, Markus Huber, and Edgar Weippl. Advanced social engineering attacks. *Journal of Information Security and applications*, 22:113–122, 2015.

[78] Eike Kühl. Die zeit - trump will sich nicht trennen. Zeitungsartikel, January 2017. URL http://www.zeit.de/digital/datenschutz/2017-01/us-praesident-donald-trump-smartphone-android-sicherheit.

[79] Huaqian Lee. Qualcomm android device startup process analysis. Blogpost, March 2017. URL http://huaqianlee.github.io/2015/08/23/Android/%E9%AB%98%E9%80%9AAndroid%E8%AE%BE%E5%A4%87%E5%90%AF%E5%8A%A8%E6%B5%81%E7%A8%8B%E5%88%86%E6%9E%90-%E4%BB%8Epower-on%E4%B8%8A%E7%94%B5%E5%88%B0Home-Lanucher%E5%90%AF%E5%8A%A8/.

[80] Jonathan Levin. *Android Internals::Power User's View*. Jonathan Levin, 2015. ISBN 0991055527. URL http://newandroidbook.com.

[81] LG. Service manual lg-d821, November 2013.

[82] LGE. Open source code distribution. URL http://opensource.lge.com/index.

[83] Michael Flossman (Lookout). Viperrat: The mobile apt targeting the israeli defense force that should be on your radar, February 2017. URL https://blog.lookout.com/viperrat-mobile-apt.

[84] Marenius. Noisebox. URL http://www.marenius.se/audiodesign/noisebox.htm.

[85] Patrick McDaniel. Bloatware comes to the smartphone. *IEEE Security & Privacy Magazine*, 10(4):85–87, jul 2012. doi: 10.1109/msp.2012.92. URL http://www.patrickmcdaniel.org/pubs/mcd12.pdf.

[86] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing speech from gyroscope signals. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 1053–1067, San Diego, CA, 2014. USENIX Association. ISBN 978-1-931971-15-7. URL https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/michalevsky.

[87] Microsoft. Bedrohungsanalyse mit dread, May 2004. URL https://msdn.microsoft.com/de-de/library/cc431335.aspx.

[88] Caroline Mockel and Ali E. Abdallah. Threat modeling approaches and tools for securing architectural designs of an e-banking application. In *2010 Sixth International Conference on Information Assurance and Security*. Institute of Electrical and Electronics Engineers (IEEE), aug 2010. doi: 10.1109/isias.2010.5604049. URL http://ieeexplore.ieee.org/abstract/document/5604049/.

[89] Abdullah Gani Muhammad Shiraz, Md Whaiduzzaman. A study on anatomy of smartphone. In *Computer Communication & Collaboration*. BAPress, 2013.

[90] Collin Mulliner and Charlie Miller. Injecting sms messages into smart phones for security analysis. In *USENIX Workshop on Offensive Technologies (WOOT)*, volume 29, 2009. URL https://www.usenix.org/legacy/event/woot09/tech/full_papers/mulliner.pdf.

[91] Collin Mulliner, Nico Golde, and Jean-Pierre Seifert. Sms of death: From analyzing to attacking mobile phones on a large scale. In *USENIX Security Symposium*, 2011. URL http://static.usenix.org/events/sec11/tech/full_papers/Mulliner.pdf.

[92] Newzoo. Global mobile market report quarterly update(q1). Market Report, 2017. URL https://www.emarketer.com/Article/Slowing-Growth-Ahead-Worldwide-Internet-Audience/1014045?SOC1001.

[93] Daniel Komaromy Nico Golde. Breaking band. Slides: https://comsecuris.com/slides/recon2016-breaking_band.pdf, June 2016. URL https://www.youtube.com/watch?v=o280NiZjNu8.

[94] Nuand. Nuand bladerf. URL http://www.nuand.com/.

[95] Schneier on Security. Remotely eavesdropping on cell phone microphones, December 2006. URL https://www.schneier.com/blog/archives/2006/12/remotely_eavesd_1.html.

[96] RangeNetworks openBTS. Open source 3gpp stack. URL http://openbts.org/.

[97] André Pereira, Manuel Correia, and Pedro Brandão. Usb connection vulnerabilities on android smartphones: Default and vendors' customizations. In *IFIP International Conference on Communications and Multimedia Security*, pages 19–32. Springer, 2014. URL http://cracs.fc.up.pt/sites/default/files/USB%20connection%20vulnerabilities%20on%20Android%20smartphones.pdf.

[98] Giuseppe Petracca, Yuqiong Sun, Trent Jaeger, and Ahmad Atamli. Audroid: Preventing attacks on audio channels in mobile devices. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 181–190. ACM, 2015. URL https://arxiv.org/pdf/1604.00320.pdf.

[99] ALSA Project. Alsa system on chip (asoc). URL http://www.alsa-project.org/main/index.php/ASoC.

[100] Qualcomm. Codeaurora, . URL https://wiki.codeaurora.org/xwiki/bin/QAEP/.

[101] Qualcomm. Hexagon dsp sdk tools and resources, . URL https://developer.qualcomm.com/software/hexagon-dsp-sdk/tools.

[102] Qualcomm. Qualcomm mss qdsp6v5 peripheral image loader. Dokumentation, January 2016. URL https://android.googlesource.com/kernel/msm/+/android-lego-7.1.1_r0.2/Documentation/devicetree/bindings/pil/pil-q6v5-mss.txt.

[103] QuarksLab. The quarkslab audit of veracrypt has been completed, and this is the public release of the results., October 2016. URL https://ostif.org/the-veracrypt-audit-results/.

[104] Rapid7. Exploit database. URL https://www.rapid7.com/db/modules/.

[105] Replicant. Samsung galaxy back-door, . URL http://redmine.replicant.us/projects/replicant/wiki/SamsungGalaxyBackdoor.

[106] Replicant. Open source samsung vendorril, . URL https://www.replicant.us/.

[107] Samsung. Open source release center. URL http://opensource.samsung.com/reception.do.

[108] Duo Security. Thirty percent of android devices susceptible to 24 critical vulnerabilities, June 2016. URL https://duo.com/blog/thirty-percent-of-android-devices-susceptible-to-24-critical-vulnerabilities.

[109] Himanshu Shewale, Sameer Patil, Vaibhav Deshmukh, and Pragya Singh. Analysis of android vulnerabilities and modern exploitation techniques. *ICTACT Journal on Communication Technology*, 5(1), 2014. URL http://ictactjournals.in/paper/IJCT_Paper_1_863_to_867.pdf.

[110] Adam Shostack. Experiences threat modeling at microsoft. In *Modeling Security Workshop. Dept. of Computing, Lancaster University, UK*, 2008. URL https://adam.shostack.org/modseco8/Shostack-ModSeco8-Experiences-Threat-Modeling-At-Microsoft.pdf.

[111] SRLabs. Snoopsnitch. Homepage. URL https://opensource.srlabs.de/projects/snoopsnitch.

[112] Matt Suiche. Wannacry the largest ransom-ware infection in history. Blogartikel, May 2017. URL https://blog.comae.io/wannacry-the-largest-ransom-ware-infection-in-history-f37da8e30a58.

[113] Check Point Research Team. Quadrooter analyse, 2016. URL https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEFCON-24-Adam-Donenfeld-Stumping-The-Mobile-Chipset-WP-UPDATED.pdf.

[114] Chromium Dev Team. Chrome exploit: V8 properties + p2phostmsgsend. URL https://bugs.chromium.org/p/chromium/issues/detail?id=416449.

[115] Project Zero Team. Feedback and data-driven updates to google's disclosure policy, February 2015. URL https://googleprojectzero.blogspot.de/2015/02/feedback-and-data-driven-updates-to.html.

[116] Telstra. Our 2g network has closed, April 2015. URL https://crowdsupport.telstra.com.au/t5/Telstra-Product-Exits/Our-2G-Network-has-closed/ba-p/440992.

[117] David Chambers (thinksmallcell.com). Overview: Mobile network statistics for 2016. Blogartikel, February 2016. URL https://www.thinksmallcell.com/Opinion/mobile-network-statistics-for-2016.html.

[118] Keith Bradsher (NY Times). Hasty exit started with pizza inside a hong kong hideout, June 2013. URL http://www.nytimes.com/2013/06/25/world/asia/snowden-departure-from-hong-kong.html.

[119] Neil Biggs (Context Information Security) Tom Court. Wap just happened to my samsung galaxy?, January 2017. URL https://www.contextis.com/resources/blog/wap-just-happened-my-samsung-galaxy/.

[120] GSMK (via Google Patents). Mobile device and method to monitor a baseband processor in relation to the actions on an applicaton processor, 2013. URL https://www.google.com/patents/US20140004829.

[121] Common Vulnerabilities and Exposures. Cve-2015-4640 (mitm rce), June 2015. URL https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-8411. http://www.cvedetails.com/cve-details.php?t=1&cve_id=CVE-2015-4640.

[122] Common Vulnerabilities and Exposures. Cve-2016-8411 (qmi buffer overflow), December 2016. URL https://source.android.com/security/bulletin/2016-12-01#vulnerabilities-in-qc-components. https://source.android.com/security/bulletin/2016-12-01#vulnerabilities-in-qc-components.

[123] Ralf-Philipp Weinmann. Baseband attacks: Remote exploitation of memory corruptions in cellular protocol stacks. In *WOOT*, pages 12–21, 2012. URL https://www.usenix.org/system/files/conference/woot12/woot12-final24.pdf.

[124] Ralf-Philipp Weinmann. Talk: Baseband exploitation in 2013: Hexagon challenges. Slides: https://rpw.sh/slides/rpw-30c3-hexagon.pdf, December 2013. URL https://media.ccc.de/v/30C3_-_5618_-_en_-_saal_1_-_201312272145_-_baseband_exploitation_in_2013_-_rpw_esizkur.

[125] Harald Welte. Anatomy of contemporary gsm cellphone hardware. -, April 2010. URL http://ondoc.logand.com/d/373/pdf.

[126] Wikimedia. Android historical version distribution graphic. URL https://commons.wikimedia.org/wiki/File:Android_historical_version_distribution_-_vector.svg.

[127] Gal Beniamini (Project Zero). Over the air: Exploiting broadcom's wi-fi stack (part 1), April 2017. URL https://googleprojectzero.blogspot.de/2017/04/over-air-exploiting-broadcoms-wi-fi_4.html.

[128] Gal Beniamini (Project Zero). Over the air: Exploiting broadcom's wi-fi stack (part 2), April 2017. URL https://googleprojectzero.blogspot.de/2017/04/over-air-exploiting-broadcoms-wi-fi_11.html.

[129] Mark Brand (Project Zero). Return to libstagefright: exploiting libutils on android, September 2016. URL https://googleprojectzero.blogspot.de/2016/09/return-to-libstagefright-exploiting.html.

Selbständigkeitserklärung

Ich erkläre ausdrücklich, dass es sich bei der von mir eingereichten schriftlichen Arbeit mit dem Titel

Analyse von Audio-Bypass-Bedrohungen auf Android-Smartphones und Konzeption von Abwehrmaßnahmen

um eine von mir selbst und ohne unerlaubte Beihilfe verfasste Originalarbeit handelt.

Ich bestätige überdies, dass die Arbeit als Ganzes oder in Teilen nicht zur Abgeltung anderer Studienleistungen eingereicht worden ist.

Ich erkläre ausdrücklich, dass ich sämtliche in der oben genannten Arbeit enthaltenen Bezüge auf fremde Quellen (einschließlich Tabellen, Grafiken u. Ä.) als solche kenntlich gemacht habe. Insbesondere bestätige ich, dass ich nach bestem Wissen sowohl bei wörtlich übernommenen Aussagen als auch bei in eigenen Worten wiedergegebenen Aussagen anderer Autorinnen oder Autoren die Urheberschaft angegeben habe.

Ich bestätige mit meiner Unterschrift die Richtigkeit dieser Angaben.

Berlin, 24. Mai 2017

Wolfgang Studier