Seminar (Secure) Communication Networks (Sommersemester 2020)

Bettina Schnor
Institute of Computer Science
University of Potsdam

Berichte Institut für Informatik
Universität Potsdam

Summary

The seminar discusses communication protocols and security aspects of communication protocols. The first part, the wireless session, presents WLAN and WPA2, Bluetooth, and 5G and data over sound. In the second part, we had talks around IPv6 and its security challenges. Finally, RADIUS ist presented, the origin of our current eduroam.

Inhaltsverzeichnis

1. Wireless Communication: WLAN and WPA2 (IEEE 802.11 and IEEE 802.11i)
   Jan Philipp Behrens
2. Bluetooth - The architecture and protocol stack
   Tim Sauvageot
3. Aspects of 5G Security
   Daniel Nelle
4. Data over Sound - Übertragung per Ultraschall
   Florian Schmeller
5. (Secure) IPv6
   Elias Pichottka
6. Secure IPv6 with the IDSv6 Benchmark
   Richard Hegewald
7. Secure IPv6 with Hyhoneydv6
   Katharina Wolf
8. RADIUS - The protocol behind eduroam
   Fabian Lindner

Wireless Communication: WLAN and WPA2 (IEEE 802.11 and IEEE 802.11i)

Jan Philipp Behrens

Universität Potsdam
jabehrens@uni-potsdam.de

Zusammenfassung. Drahtlose lokale Netzwerke (WLAN) sind aus dem Alltag längst nicht mehr wegzudenken und gewinnen stetig an Bedeutung, da immer mehr Geräte von der Uhr bis zum Kühlschrank WLAN-fähig werden. Durch die Verwendung von WLAN wird lästige Verkabelung überflüssig und die Bewegung der Benutzer nicht mehr eingeschränkt. Im Folgenden soll zuerst eine Einführung in WLAN im Allgemeinen, mit dem Fokus auf die verschiedenen Betriebsmodi, Standards, das MAC-Protokoll und den Frame, gegeben werden. Anschließend sollen mit WEP und WPA die Vorgänger des WPA 2 Sicherheitsstandards betrachtet sowie deren Sicherheitslücken aufgezeigt werden. Zuletzt schauen wir uns das WPA 2 Protokoll und dessen Sicherheitslücken Hole196, KRACK und Kr00k an.

Schlüsselwörter: WLAN · WEP · WPA2 · IEEE 802.11 · IEEE 802.11i · Hole196 · KRACK · Kr00k · Wireless Communication

1 IEEE 802.11 / WLAN

1.1 Betriebsmodi

WLAN bietet eine Reihe verschiedener Betriebsmodi. Die beiden wichtigsten sind der Infrastruktur- (Abb. 1) und der Ad-hoc-Modus (Abb. 2). Der Gängigste ist der Infrastrukturmodus, bei dem Verbindungen stets über einen zentralen Access Point aufgebaut werden [8]. Der Ad-hoc-Modus hingegen verbindet die Teilnehmer über Punkt-zu-Punkt Verbindungen direkt miteinander [6].

1.2 Standards

Seit der Einführung des IEEE 802.11 Standards im Jahr 1997 kamen eine Vielzahl neuer Standards hinzu. Diese führten unter anderem neue Frequenzbänder (802.11a,b,g,n,...), Sicherheitsstandards (802.11i) und Quality of Service Verbesserungen (802.11e) ein. Zu den verwendeten Frequenzbereichen gehören der 0,9 GHz, 2,4 GHz, 5 GHz und 60 GHz Bereich. Da die Durchdringung von Objekten schlechter wird je höher die Frequenz ist, kommen die Frequenzbereiche in

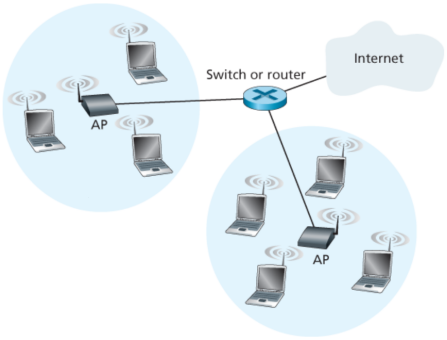


Abb. 1. Infrastrukturmodus (Quelle: [8]).

Abb. 2. Ad-hoc-Modus (Quelle: [8]).

unterschiedlichen Szenarien zum Einsatz. Der sub-GHz Bereich spielt beim Internet of Things (IoT) und der Machine to Machine (M2M) Kommunikation eine Rolle, während der 2,4 und 5 GHz Bereich für den alltäglichen Gebrauch zum Beispiel im Haus und auf der Arbeit zum Einsatz kommt. Der 60 GHz Bereich bleibt Großrechenanlagen vorbehalten und besitzt nur wenige Meter Reichweite. Die Datenrate hat seit 1997 stetig zugenommen, einerseits durch effizientere Kodierungen, jedoch auch durch Kanalbündelung (MIMO) und Richtfunktechnik (Beamforming). Die heute möglichen Datenraten liegen zwischen 5 und 176 GBit/s [1,7].

1.3 MAC Protokoll

Als Media Access Control (MAC) Protokoll kommt beim WLAN das sogenannte Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) zum Einsatz. Dieses soll, wie der Name schon sagt, Kollisionen vermeiden. Im Gegensatz dazu wird beim Ethernet das CSMA/CD (Collision Detection) verwendet. Um Kollisionen zu vermeiden horcht der Sender zuerst, ob der Empfänger beschäftigt ist. Falls dies nicht der Fall ist, sendet er und wartet auf die Empfangsbestätigung. Erhält der Sender keine Empfangsbestätigung wiederholt er diesen Vorgang. Ist der Empfänger jedoch beschäftigt, so wartet der Sender bis dies nicht mehr der Fall ist und startet dann das Herunterzählen eines zufälligen Backoff-Zählers, um gleichzeitiges Senden mehrerer wartender Clients zu verhindern. Dies führt jedoch zum Problem der Hidden Station [8].

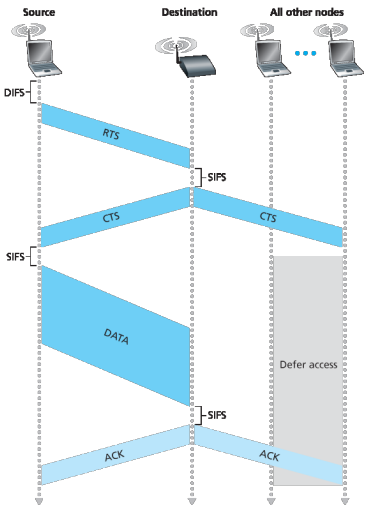


Abb. 3. Reservieren des Kanalzugriffs (Quelle: [8]).

Hidden Station Problem Das Hidden Station Problem tritt auf, wenn sich mehrere Sender nicht sehen und gleichzeitig senden. Um dies zu verhindern, werden Request to Send (RTS) und Clear to Send (CTS) Control Frames verwendet, um den Kanalzugriff zu reservieren (Abbildung 3) [8].

1.4 Frame

Der Frame besteht beim WLAN im Wesentlichen aus einer 4 Byte Prüfsumme, 2312 Byte Payload, vier Adressfeldern für je eine 6 Byte große MAC-Adresse und der Sequenznummer (Abbildung 4). Die Prüfsumme dient zum Korrigieren von Übertragungsfehlern, die beim WLAN wesentlich häufiger auftreten als beim Ethernet. Zwei der vier Adressfelder sind für den Sender und Empfänger, während die anderen beiden Adressfelder nur im Falle einer Weiterleitung verwendet werden. Weiterleitungen können beispielsweise im Ad-hoc- und im Infrastrukturmodus auftreten, wenn zum Beispiel nicht direkt sondern über einen AP mit dem Router kommuniziert wird. Die Sequenznummern dienen zur Unterscheidung neuer Frames von Retransmission-Frames [8].

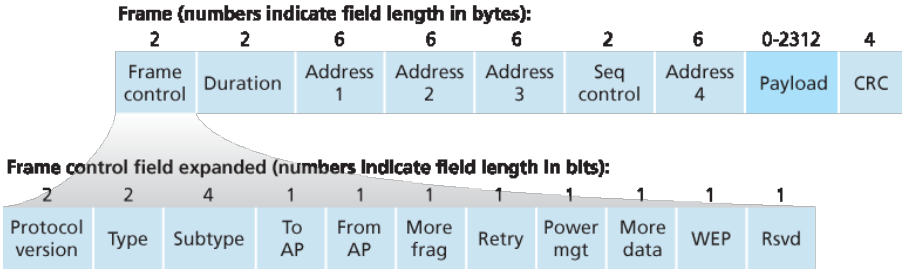


Abb. 4. WLAN Frame (Quelle: [8]).

1.5 Probleme / Nachteile

Privatssphäre Das Wi-Fi Positioning System erlaubt dezimetergenaue Ortung.

Gesundheitsschäden Es wird immernoch kontrovers diskutiert wie stark die Schäden, insbesondere durch oxidativen Stress, durch die Strahlung des WLANs sind.

Reichweite Die Reichweite bis zu der WLAN-Signale noch empfangbar sind liegt bei etlichen Kilometern, was dazu führt, dass man nicht mehr nur lokal angreifbar ist.

2 IEEE 802.11i / WPA 2

2.1 Vorgänger

Wired Equivalent Privacy Das 1999 eingeführte IEEE 802.11 Wired Equivalent Privacy (WEP) Protokoll war das erste Sicherheitsprotokoll für WLAN. Es basiert auf einer Stromchiffre, die den Klartext sowie die Prüfsumme bitweise mit dem Schlüsselstrom XOR-verknüpft, dem sogenannten RC4 Verfahren. Wobei die Sicherheit von Stromchiffren auf der Einmalverwendung der Schlüssel basiert. Die Schlüssellänge des WEP betrug zunächst 64, später 128 und zuletzt 256 Bit und setzt sich aus zwei Komponenten zusammen: dem Initialisierungsvektor und dem symmetrischen Schlüssel. Der 24 Bit große Initialisierungsvektor (IV) wird für jeden Frame neu generiert und steht unverschlüsselt im Header des Frames. Der symmetrische Schlüssel (auch WLAN-Passwort genannt) ist vorher festgelegt und muss einmalig über einen sicheren Kanal ausgetauscht werden. Da der symmetrische Schlüssel sich nicht ändert, reicht er alleine nicht aus, um das Sicherheitskriterium von Stromchiffren zu erfüllen und muss daher mit dem zufälligen IV kombiniert werden. Um, die bei funkübertragungen häufiger auftretenden, Übertragungsfehler finden und korrigieren zu können, beinhaltet das WEP Protokoll einen 32 Bit Cyclic Redundancy Check (CRC32). Da diese jedoch nur eine vergleichsweise einfache lineare Funktion ist, eignet sie sich nicht als sogenanntes Message Authentication Protocol. Das heißt Angreifer können unbemerkt Schlüsseltext und Prüfsumme verändern. Ein weiteres Problem des WEP Protokolls ist der IV der mit 24 Bit zu klein bemessen ist und sich, bei nur 2^{24} Möglichkeiten, zu schnell wiederholt. Das der IV außerdem unverschlüsselt im Header steht, macht es Angreifern zusätzlich einfach, da diese so direkt einen sich wiederholenden IV bemerken. In Abbildung 5 ist der schematische Ablauf der Verschlüsselung dargestellt [5,6,7,8,9].

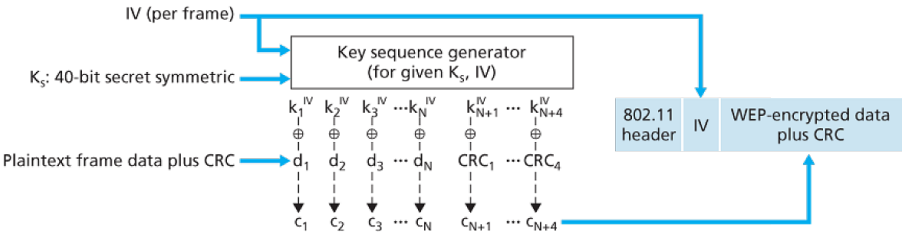


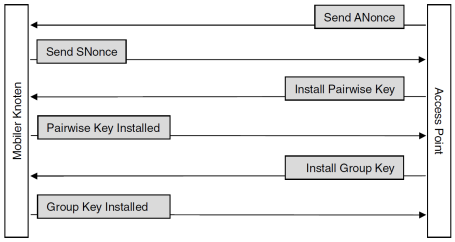
Abb. 5. WEP Protokoll (Quelle: [8]).

WPA Der Nachfolger des WEP war das Wi-Fi Protected Access (WPA) Protokoll, welches als Zwischenlösung diente bis Geräte für WPA 2 zur verfügung standen. Es benutzte weiterhin das RC4-Verfahren zur Verschlüsselung, da es auf der gleichen Hardware laufen musste wie WEP. Verbessert wurde das Protokoll um das sogenannte Temporal Key Integrity Protocol (TKIP), dass die Berechnung eines Message Integrity Codes (MIC), einen größeren 48 Bit IV, die Einführung von Sequenznummern sowie die Erzeugung unterschiedlicher Schlüssel je Frame vorsah. Das Berechnen eines MIC sollte die Schwächen des zuvor beim WEP verwendeten CRC32 beheben, während sich der größere IV im Vergleich zu dem des WEPs seltener wiederholte. Die Erzeugung unterschiedlicher Schlüssel für jeden Frame, lässt nicht direkt Rückschlüsse auf den symmetrischen Schlüssel zu, da dieser nur indirekt zum generieren der temporären Schlüssel verwendet wird. Und die Einführung von Sequenznummern sollte Replayattacken erschweren, in dem nun neue Frames von Retransmits unterschieden werden konnten [5,6,7,9].

2.2 WPA 2

Das IEEE 802.11i WPA 2 Protokoll wurde entwickelt, um die Schwachstellen des WEP zu beheben. Es verwendet den aufwändigeren Advanced Encryption Standard (AES) zum verschlüsseln, weshalb neue Hardware mit Coprozessoren nötig wurde. Das Protokoll unterscheidet zwischen einer Authentifizierung mittels Pre-Shared Key (PSK) dem sogenannten Personal Mode und einer Authentifizierung mittels des Extensible Authentication Protocols (EAP), bei dem der Access Point (AP) die Anfrage an einen Authentifizierungsserver weiterleitet, dem Enterprise Mode. Bei der Authentifizierung mittels Pre-Shared Key kommt ein vorher festgelegtes 8 bis 63 Zeichen langes Passwort zum Einsatz, aus dem sich dann durch 4096-fache Anwendung der SHA1 Hashfunktion der Pairwise Master Key (PMK) berechnet.

Auf die Authentifizierung folgt die Key Generation and Distribution (KGD) Phase. Diese sieht einen 4-Way sowie einen Group Handshake vor (Abbildung 6). Beim 4-Way Handshake tauschen sich Client und AP zunächst einen Nonce sowie einen Message Integrity Code (MIC) aus. Letzterer dient zur Erkennung von Übertragungsfehlern und dem Verhindern von Forging. Der Nonce (number only used once) ist eine Zahl die nur einmal verwendet wird und für den ersten Frame zufällig generiert und danach inkrementiert wird. Nach dem Austausch von Nonce und MIC berechnen Client und AP den Pairwise Transient Key (PTK) aus dem PMK, beiden Nonce und den MAC-Adressen. Bei dem Group Handshake kommt hingegen kein MIC zur Anwendung und der Group Master Key (GMK) wird zufällig generiert. Aus dem GMK leitet sich dann der Group Transient Key (GTK) ab. Eine Übersicht über die bei WPA 2 verwendeten Schlüssel ist in Abbildung 7 dargestellt [5,6,7,9].



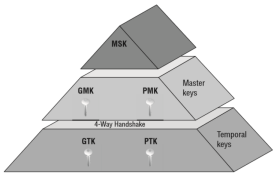


Abb. 6. Handshakes KGD Phase (Quelle: [5]). **Abb. 7.** Schlüsselhierarchie WPA 2 [1].

2.3 Schwachstellen

Hole 196 Die Hole 196 genannte Sicherheitslücke wurde so benannt, da sie auf Seite 196 der IEEE Spezifikation steht und bezeichnet das Fehlen eines MIC beim Group Handshake. Diese Lücke führt dazu, dass keine Möglichkeit besteht zu überprüfen, ob Adressen gefälscht oder Daten manipuliert wurden. Dadurch wird es jedem Netzwerkteilnehmer ermöglicht unbemerkt selbst gefälschte Broadcast-Nachrichten anstelle des AP zu versenden [4].

KRACK Die 2017 entdeckte Key Reinstallation Attack (KRACK) basiert auf der erzwungenen Wiederverwendung der Nonce. Dazu sendet der Angreifer zunächst die dritte Nachricht des 4-Way Handshakes erneut. Woraufhin das Opfer den bereits verwendeten Schlüssel reinstalliert und dabei den Nonce zurücksetzt. Der Angriff beinhaltet verschiedene Varianten, denn nicht jede Implementation erlaubt einen Retransmit im Klartext, sodass zusätzlich zum Beispiel Racing Conditions ausgenutzt werden müssen. KRACK ermöglicht letztlich das Entschlüsseln von Paketen und wenn nur WPA TKIP verwendet wird sogar Zugriff auf den MIC und somit Forging. Zu bemerken ist außerdem, dass sowohl iOS 10.3.1 als auch Windows 7 und 10 nicht anfällig für diesen Angriff sind, da diese die WPA 2 Spezifikation nicht komplett umsetzten in dem sie Retransmits der dritten Handshake-Nachricht verboten [10].

Kr00k KRACK führte zwei Jahre nach seiner Entdeckung zu der Entdeckung einer weiteren Schwachstelle, der Kr00k. Beim Testen der eigentlich bereits gepatchten zweiten Generation der Amazon Echo geräte bemerkte man, dass diese immernoch angreifbar waren, der Grund war diesmal jedoch Kr00k. Bei Kr00k handelt es sich im Vergleich zu KRACK nicht direkt um einen Angriff, sondern um einen Hardwarefehler der WLAN-Chips von Cypress und Broadcom. Dieser tritt nach einer Verbindungstrennung auf bei der der Session Key aus dem Speicher gelöscht (mit Nullen überschrieben) wird und führt dazu, dass restliche Pakete aus dem Transmit Buffer, mit diesem All-Zero-Key verschlüsselt,

[1] Quelle (abgerufen: 2020-05-24): https://www.wifi-professionals.com/wp-content/uploads/2019/01/Hierarchy-768x489.png

übertragen werden. Eine Verbindungstrennung und damit der Fehler, tritt auf wenn man sein WLAN abschaltet, die Verbindung manuell trennt, sich außerhalb der Reichweite des AP befindet, der AP oder sein WLAN abgeschaltet werden oder beim Roaming zwischen zwei APs. Außerdem kann eine Trennung von einem Angreifer, durch das Senden von Management Frames, auch absichtlich herbeigeführt werden, da diese unverschlüsselt verschickt werden. Dieser Schwachpunkt wurde mit IEEE 802.11w durch Protected Management Frames (PMF) behoben, welche beim Nachfolger WPA 3 pflicht sind. Der Fehler betraf Milliarden von Geräten verschiedenster Hersteller, ließ sich jedoch mit einem Softwareupdate beheben [2,3].

Konklusion

Die WLAN Spezifikation IEEE 802.11 zeigt beispielhaft auf, dass Standards im IT-Bereich einer häufigen Anpassung und Erweiterung bedürfen, um auf neue Begebenheiten zu reagieren und anfänglich nicht bedachte Szenarien abzudecken. Außerdem bieten Sicherheitsstandards nie absolute Sicherheit und sind manchmal sogar erst der Grund für eine potentielle Angreifbarkeit (Hole196 bzw. KRACK).

Literatur

1. IEEE 802.11 WLAN Standards im Vergleich, https://www.welotec.com/de/wlan-standards-vergleich, abgerufen: 2020-05-24
2. Kr00k: How kracking amazon echo exposed a billion+ vulnerable wifi devices, https://www.rsaconference.com/usa/agenda/kr00k-how-kracking-amazon-echo-exposed-a-billion-vulnerable-wifi-devices, abgerufen: 2020-05-27
3. Kr00k white paper, https://www.welivesecurity.com/wp-content/uploads/2020/02/ESET_Kr00k.pdf, abgerufen: 2020-05-24
4. WPA2 Hole196 Vulnerability: Exploits and Remediation Strategies. A Whitepaper by AirTight Networks, Inc., http://securedsolutions.com.my/pdf/WhitePapers/WPA2-Hole196-Vulnerability.pdf, abgerufen: 2020-05-24
5. Bless, R., Mink, S., Blaß, E.O., Conrad, M., Hof, H.J., Kutzner, K., Schöller, M.: Sichere Netzwerkkommunikation: Grundlagen, Protokolle und Architekturen. Springer-Verlag (2006)
6. Eckert, C.: IT-Sicherheit: Konzepte-Verfahren-Protokolle. Walter de Gruyter (2013)
7. Frankel, S., Eydt, B., Owens, L., Scarfone, K.: Establishing wireless robust security networks: a guide to IEEE 802.11 i. NIST Special Publication pp. 800–97 (2007)
8. Kurose, J.F., Ross, K.W.: Computer networking: a top-down approach. Pearson (2017)
9. Schmeh, K.: Kryptografie: Verfahren, Protokolle, Infrastrukturen. dpunkt.-Verlag (2016)
10. Vanhoef, M., Piessens, F.: Key reinstallation attacks: Forcing nonce reuse in wpa2. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1313–1328 (2017)

Bluetooth - The architecture and protocol stack

Tim Sauvageot

Universität Potsdam, Potsdam 14469, Deutschland
sauvageot@uni-potsdam.de

Zusammenfassung. Bluetooth ermöglicht nicht nur Musik hören ohne Kabelsalat, sondern unterstützt auch TCP/IP, Gruppenkommunikation und viele weitere Funktionen. Im Folgenden wird zuerst eine kurze Erläuterung zu der Entstehung von Bluetooth gegeben. Anschließend wird auf Bluetooth Protokolle und die Architektur eingegangen und gezeigt, weshalb Bluetooth eine Vielzahl bekannter Protokolle unterstützen kann. Besonderer Fokus wird dabei auf die Sicherheitsarchitektur gelegt. Abschließend werden neue Funktionen aus den Versionen 5.1 und 5.2 vorgestellt.

Schlüsselwörter: Bluetooth Architektur · Bluetooth Protokolle · IEEE 802.15 · LE · Bluetooth Sicherheit

1 Motivation

Bluetooth entstand aus einer Studie, die 1994 von Ericsson durchgeführt wurde. Ziel der Studie war es, eine günstige und energiesparsame Lösung zu finden, die drahtlose Kommunikation für kleine Entfernungen ermöglicht. Aus dieser Studie entstand 1998 die Bluetooth Special Interest Group (SIG). Diese ist für die Weiterentwicklung der Bluetooth Spezifikation verantwortlich. Prominente Mitglieder dieser Gruppe sind beispielsweise Nokia, Intel und Microsoft.

Bei der Namensgebung war der dänische König Haral Blauzahn (Bluetooth) das Vorbild. Dieser vereinigte zu seiner Zeit Dänemark, indem der Konflikte des Landes löste. Die Bluetooth SIG hatte ähnliche Ziele, denn sie wollten die drahtlose Kommunikation von Geräten vereinheitlichen. Sowohl die Kommunikation, als auch das Zusammenspiel von Anwendungen, konnte harmonisert und im Standard IEEE 802.15 definiert werden [2].

2 Architektur & Protokolle

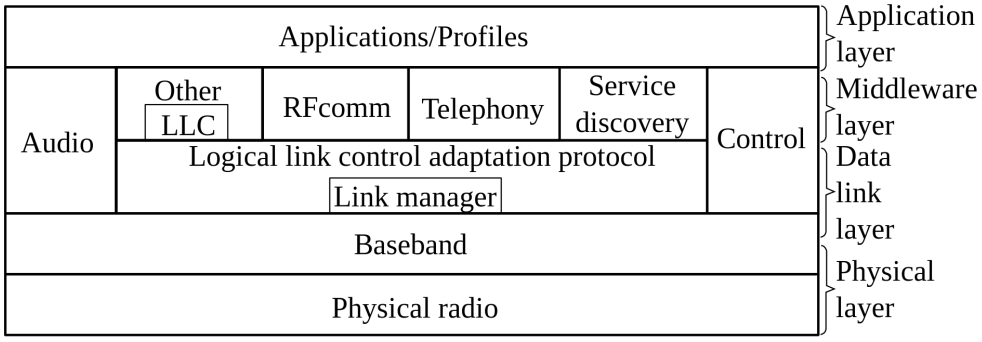


Abbildung 1. Bluetooth Architektur [1]

Bluetooth ist als Schichtenarchitektur entwickelt worden. An Hand von Abbildungen 1 und 2 erkennt man, dass es zu den Architekturkomponenten auch teilweise ein zugehöriges Protokoll gibt. Dies ist beispielsweise bei der Service Discovery und dem Service Discovery Protocol (SDP) als auch dem Link Manager und Link Manager Protocol (LMP) der Fall.

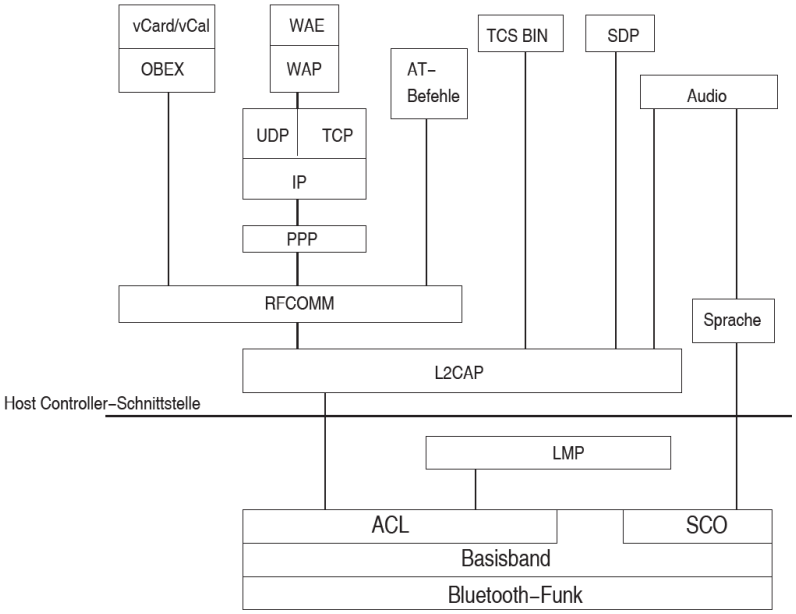


Abbildung 2. Bluetooth Protokollstack[2, p. 941]

2.1 MAC Control

Bluetooth war ursprünglich für Distanzen von maximal 10 Metern ausgelegt. Innerhalb dieser Distanz können sich bereits mehrere Bluetoothfähige Geräte wie Kopfhörer, Tastatur oder Maus befinden. Damit diese Geräte sich nicht gegenseitig stören, war es erforderlich, eine Lösung zu entwickeln, welche die Interferenz auf kleinen Räumen verhindert bzw. abschwächt. MAC Control bei Bluetooth wird mit Hilfe des Spread spektrums umgesetzt. Dabei wird das zu sendende Signal über den gesamten Frequenzbereich (2,401 - 2,479 Ghz) gespreizt. Der Frequenzbereich wird dafür in 79 Kanäle mit jeweils 1 MHz Bandbreite aufgeteilt. Der Master im jeweiligen Netzwerk bestimmt dann die Frequenz, auf der innerhalb des Netzwerks gesendet wird. Diese Frequenz wird dabei 1600 mal pro Sekunde gewechselt. Die Anzahl der Frequenzwechsel wird als Hop Rate bezeichnet. Die Einteilung der Bandbreite und das Wechseln der Kanäle wird als Frequence Hopping bezeichnet [2].

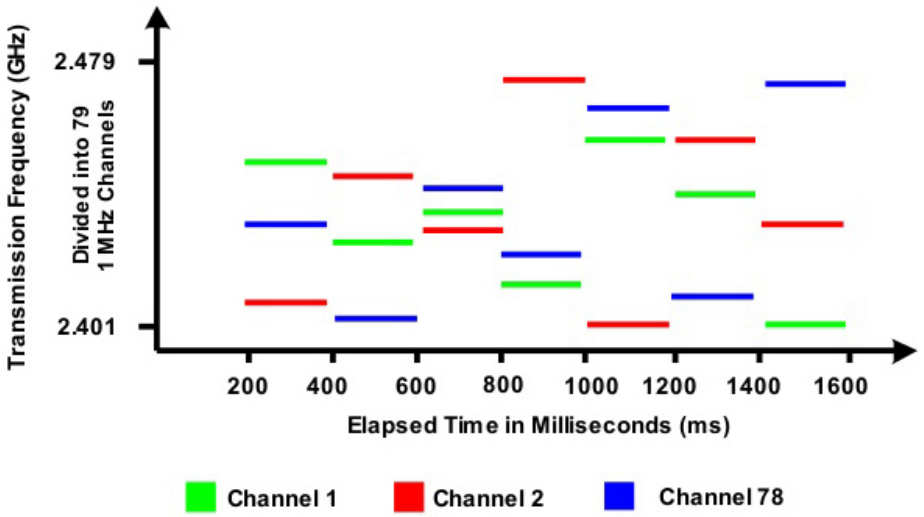


Abbildung 3. Spread spektrum[7]

Durch das schnelle Wechseln der Sendefrequenz wird nicht nur die Interferenz der Signale abgeschwächt, sondern auch eine grundlegende Abhörsicherheit gewährleistet. Angreifer, die nicht Teil des abzuhörenden Netzwerks sind, müssen die jeweils nächste Frequenz erraten, um dauerhaft die Kommunikation mitverfolgen zu können.

Aktuelle Bluetooth Geräte sind auch in der Lage über Distanzen von bis zu 100 Metern zu kommunizieren. Auf Grund der dafür benötigten Signalstärke verbraucht dies jedoch auch mehr Energie. Es ist daher wünschenswert, dass sich Kommunikationspartner in der Nähe befinden, um den Energieverbrauch zu minimieren. Mit Bluetooth 5.2 wurde eine Lösung gefunden, um den Energieverbrauch aktiv zu optimieren.

2.2 Pakete

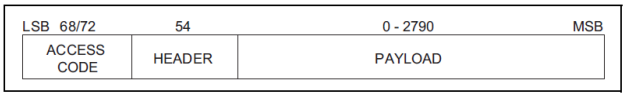


Abbildung 4. Paketformat[3, p. 461]

Der Datenaustausch bei Bluetooth findet über Pakete statt. Diese werden durch das Physical radio über die Luftschnittstelle übertragen. Der grundlegende Auf-

bau eines Pakets ist dabei in Abbildung 4 sichtbar. Der Zugangscode (access code) dient zur Synchronisierung von Kommunikationspartnern innerhalb eines Netzwerks. Die Größe des Zugangscodes beträgt, wenn kein anderer Inhalt des Pakets gesetzt ist, 68 Bit. Andernfalls ist der Zugangscode 72 Bit groß.
Der Header enthält Informationen zur Leitungssteuerung. Um Geräte innerhalb eines Netzwerks zu identifizieren, ist außerdem eine temporäre 3 Bit Addresse im Header gesetzt.
Der Payload dient zur Übertragung der eigentlichen Inhalte. Bei asynchroner Übertragung von Daten (ACL) ist außerdem eine 16 Bit Cyclic Redundancy Check Prüfsumme enthalten.
Die maximale Größe eines Pakets entspricht also 2912 Bit (364 Byte).

Paketvermittlung Bluetooth unterscheidet grundlegend zwischen zwei Paketmodi: Synchronous Connection-Oriented (SCO) und Asynchronous Connection-Less (ACL). Wie der Name schon erkennen lässt, ist SCO für synchrone Anwendungen mit geringer Latenz gedacht. Eine dieser Anwendungen ist beispielsweise die Sprachübertragung bei Kopfhörern. Jeder SCO Kanal hat eine Bandbreite von 64kBit/s.
Im Gegensatz dazu ist ACL dafür ausgelegt, "best effort"Verbindungen zu ermöglichen.

2.3 Link Manager

Der Link Manager setzt auf dem Baseband an und ist dem Data link layer zuzuordnen. Die Funktionen des Link Manages werden durch das Link Manager Protocol umgesetzt,

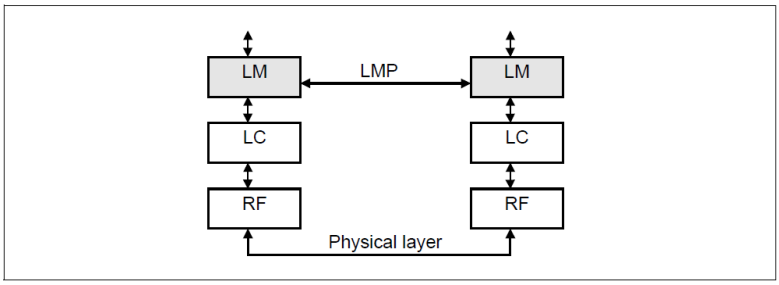


Abbildung 5. LMP Kommunikation[3, p. 572]

Link Manager Protocol Ein integraler Teil des Link Managers ist das Link Manager Protocol. Dieses ist für die Verbindungsverwaltung des Geräts verantwortlich. Dazu gehört beispielsweise die Authentifizierung von anderen Geräten oder die Verschlüsselung von Nachrichten.

Zwei Link Manager kommunizieren mittels Protocol Data Units (PDU). PDUs sind Nachrichtenformate, die standardmäßig durch Bluetooth spezifiziert werden. Allgemein werden die Nachrichten, die Link Manager austauschen, als Link Manager Protocol Nachricht bezeichnet. Diese Nachrichten werden im Payload der Pakete übermittelt. Weiterhin wird das L_CH Feld im Paketheader gesetzt. Über dieses können Link Manager die Nachrichten erkennen und abfangen. LMP Nachrichten werden anschließend nicht auf höhrere Schichten propagiert. Eine weitere Besonderheit vom LMP Nachrichten ist, dass diesen eine höhere Priorität als Benutzerdaten gegeben wird. Konkurrieren also ein Link Manager und eine Anwendung um Zugriff auf einen Kommunikationskanal, wird zuerst die LMP Nachricht gesendet.

Sicherheitsmanager Die zentrale Komponente der Bluetooth Sicherheitsarchitektur ist der Sicherheitsmanager. Dieser verwaltet Sicherheitsanforderungen von Diensten und Authentifizierungsinformationen von Geräten. Über die External Security Control Entity (ESCE) können externe Benutzereingaben, die beim Pairing erforderlich sein können, entgegengenommen werden. Weitere Funktionen sind die Durchführung der Authentifikation, Ver- und Entschlüsselung von Daten und die Initialisierung des Pairings [2].

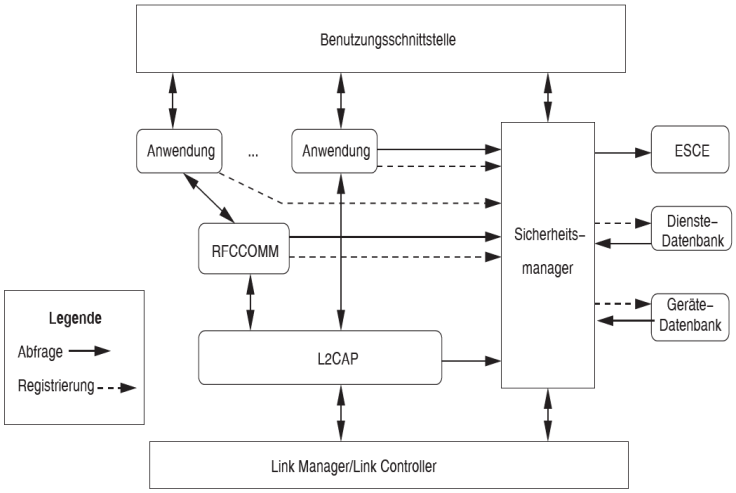


Abbildung 6. Sicherheitsmanager[2, p. 945]

Authentifizierung Um Gerätezugriff zu kontrollieren, ist es erforderlich alle Anfragen zu authentifizieren und authorisieren. Alle Anfrager werden dabei über eine 48 Bit Addresse (BD_ADDR) identifiziert. Dabei ist anzumerken, dass lediglich Geräte und keine einzelnen Benutzer authentifiziert werden. Eine An-

wendung mit benutzerbasierter Authentifizierung ist beispielsweise E-Mail.
Geräte in der Umbegung werden durch periodische inquiry Nachrichten gefunden. Empfängt ein Gerät eine solche Nachricht und hat den Discoverable mode aktiviert, so sendet es eine Nachricht mit dessen eigener BD_ADDR zurück. Durch das anschließende senden einer Page Nachricht wird die Kommunikation gestartet. Die Authentifizierung beider Parteien findet nun statt.
Bluetooth unterscheidet grundlegend zwischen vier unterschiedlichen Sicherheitmodi. Ein Gerät kann sich zu jeder Zeit in nur einem dieser Zustände befinden. Es ist wichtig zu beachten, dass die Namen der einzelnen Modi kein Indiz für die von ihnen bereitgestellte Sicherheit ist. In Abbildung 7 sind die einzelnen Pfade, die bei Authentifizierung und Authorisierung durchlaufen werden, gekennzeichnet.

Security Modus - 1 Im ersten Sicherheitsmodus werden keine Sicherheitsfunktionen bereitgestellt. Jede Anfrage wird sowohl authentifiziert als auch authorisiert. Anfragende Geräte haben also Zugriff auf alle Funktionen des Kommunikationspartners. Der Modus sollte deshalb nur für Geräte verwendet werden, die keine sicherheitsrelevanten Funktionen bereitstellen, da auch der Zugriff auf diese uneingeschränkt ist. Ein Anwendungsfall ist der Visitenkartenaustausch.

Security Modus - 2 und 4 Der zweite Sicherheitsmodus ermöglicht es anwendungsspezifische Sicherheitsmaßnahmen zu definieren. Diese werden im Service Layer umgesetzt. Wie in Abbildung 7 erkennbar ist, werden Anfragen in diesem Modus nicht authentifiziert sondern nur authorisiert. Um in diesem Modus zu kommunizieren muss außerdem eine L2CAP Verbindung aufgebaut werden.
Der Modus 4 unterscheidet sich von Modus 2 nur in einem Punkt. Im vierten Modus kann die Variante des Secure Simple Pairings definiert werden. Das Pairing spielt eine wichtige Rolle beim Generieren von Schlüsseln.
Eine weitere Funktion dieser Modi ist die Kategorisierung von Geräten an Hand von trust leveln. Die drei Kategorien sind: trustworty, not trustworthy und unknown. Wird ein Gerät als trustworthy eingestuft, so gilt es als authentifiziert und hat uneingeschränkten Zugriff auf alle Funktion des angefragten Geräts. Not trustworthy und unknown werden die gleichen Zugriffsrechte gewährt. Bei beiden wird der Zugriff eingeschränkt. Anwendungen und Dienste können definieren, mit welchem trust level auf sie zugegriffen werden darf. Diese Definitionen werden als Einträge in der Dienste-Datenbank des Sicherheitsmanagers gespeichert. Trust level, BD_ADDR, Gerätename und Link Key werden als Einträge in der Geräten-Datenbank des Sicherheitsmanagers gespeichert. Der Gerätename ist dabei ein friendly name wie beispielsweise Max Mustermanns Kopfhörer".

Modus 3 Im dritten Modus, welcher im Link Layers angesiedelt ist, wird ein grundlegender Schutz für alle Anwendungen gewährleistet. Alle Anfragen werden authentifiziert, jedoch nicht authorisiert. Um auf einen Dienst zugreifen zu können, muss vorher eine Link Manager Protocol Verbindung aufgebaut werden.

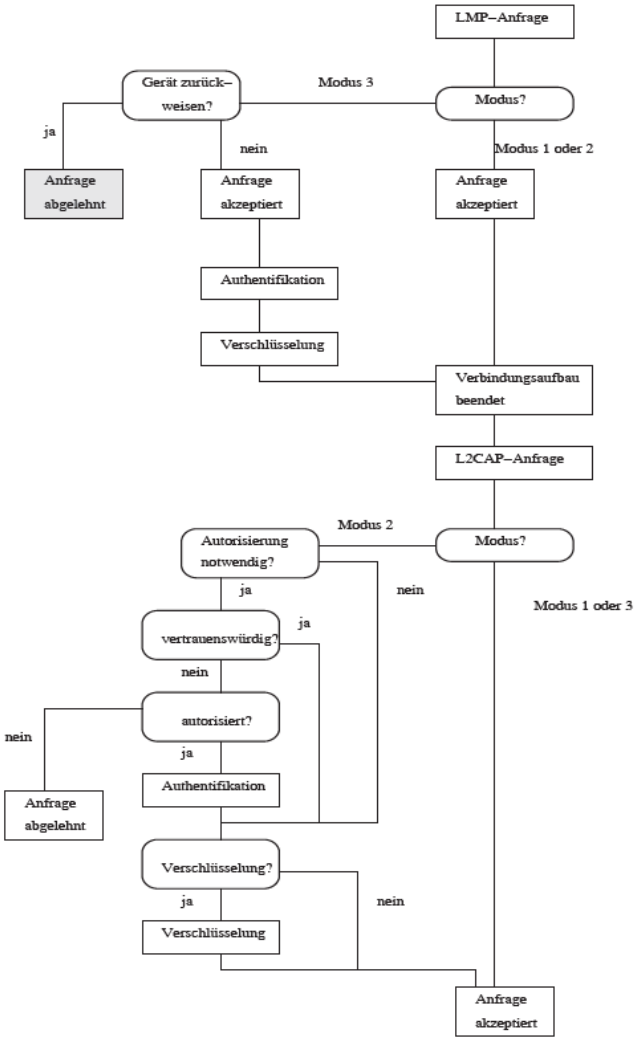


Abbildung 7. Sicherheitsmodi[2, p. 946]

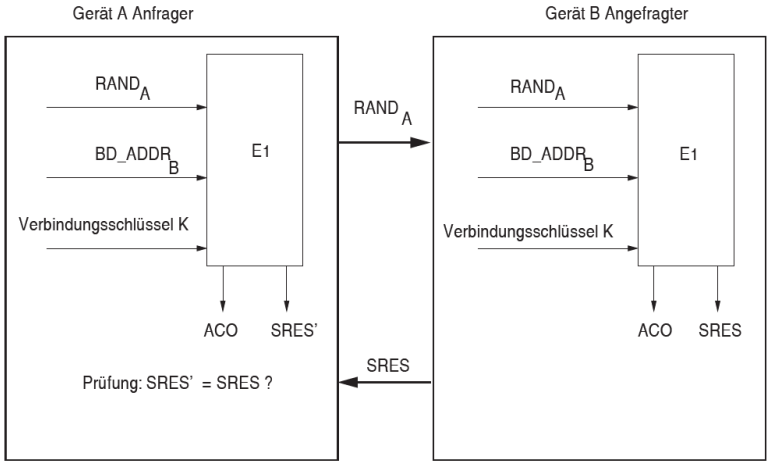


Abbildung 8. Ablauf Authentifizierungsprotokoll[2, p. 954]

Authentifizierungsprotokoll Bei der Authentifizierung sendet der Anfrager A zuerst eine 128 Bit Zufallszahl RAND an den Kommunikationspartner B. Beide berechnen anschließend eine Antwort. Diese wird mit dem E21 Algorithmus berechnet. Als Eingaben dienen die Zufallszahl RAND, die Geräteaddresse von B und ein Verbindungsschlüssel K. Haben sich die Geräte vorher bereits authentifiziert, so verfügen sie über einen gemeinsamen Verbindungsschlüssel, der beispielsweise in der Geräte-Datenbank des Sicherheitsmanagers abgelegt ist. Ist ein solcher Schlüssel nicht vorhanden, wird das Pairing initialisiert. Falls der Schlüssel vorhanden ist und die berechnete Antwort beider Geräte nicht übereinstimmt, schlägt die Authentifizierung fehl. Der Anfrager wird für eine bestimmte Zeit blockiert. Die Dauer der Sperrung steigt mit jedem weiteren fehlgeschlagenen Authentifizierungsversuch. Dadurch sollen Brute-Force Attacken verhindert werden.
Wie die beiden Geräte sich auf einen Verbindungsschlüssel einigen und diesen geheim halten, wird im folgenden Abschnitt erläutert.

Schlüsselmanagement Die Sicherheit bei Bluetooth basiert auf einer Reihe von Schlüsseln. Diese werden unterteilt in Verbindungs- und Kommunikationsschlüssel. Die Verbindungsschlüssel (link key) werden zur Berechnung des Kommunikationsschlüssels (communication key) verwendet. Mit dem Kommunikationsschlüssel wird dann der Payload der Pakete verschlüsselt.

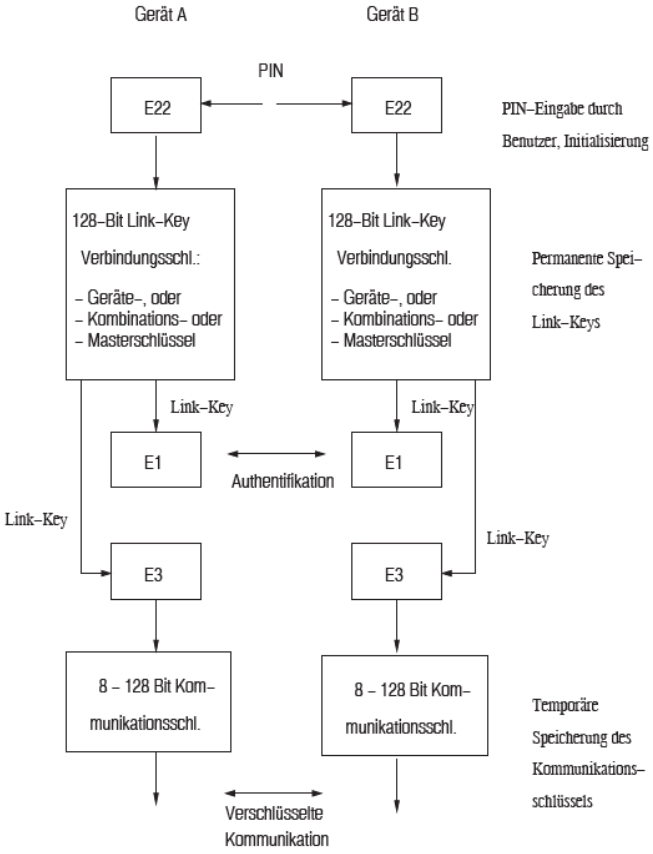


Abbildung 9. Übersicht Schlüsselmanagement[2, p. 949]

Pairing Wie bereits im Security Modus 4 erwähnt, spielt Pairing eine wichtige Rolle. Das Pairing wird beim Erstkontakt von zwei Geräten initialisiert. Dieses dient dem sicheren Austausch eines gemeinsamen Schlüssels. Um dieses durchzuführen ist es erforderlich, dass beide Geräte über ein gemeinsames Geheimnis, eine PIN, verfügen. Die Länge der PIN kann zwischen 1 und 16 Byte variieren. Folgende Möglichkeiten stehen dabei zur Verfügung:

- Auf beiden Geräten wird mittels der ESCE Komponente des Sicherheitsmanagers die gleiche PIN eingegeben.
- Eins der beiden Geräten besitzt eine feste, bekannte PIN. Diese wird auf dem zweiten Gerät eingegeben.
- Die Generierung der PIN wird auf Anwendungsebene durchgeführt. Ein Algorithmus der dafür verwendet wird ist beispielsweise Diffie-Hellman.

Initialisierungsschlüssel Nach Eingabe der PIN wird der 128 Bit Initialisierungschlüssel (IK) berechnet. Dieser wird während der weiteren Kommunikation zur Verschlüsselung der Nachrichten verwendet. Das anfragende Gerät A generiert zuerst eine Zufallszahl RAND und sendet diese an den Kommunikationspartner B. Beide Geräte berechnen anschließend mit Hilfe es E22 Algorithmus den IK. Die Eingaben bei E22 sind die PIN, PIN-Länge und die BD_ADDR von B. Eines der Geräte schickt anschließend eine Challenge in Form einer Zufallszahl an den anderen. Unter Einbeziehung der Zufallszahl, Geräteadresse und IK wird dann eine Antwort berechnet. Stimmen die Ergebnisse beider Parteien überein, wird der Vorgang fortgesetzt. Bei keiner Übereinstimmung wird der Vorgang abgebrochen und das Pairing muss erneut initialisiert werden.

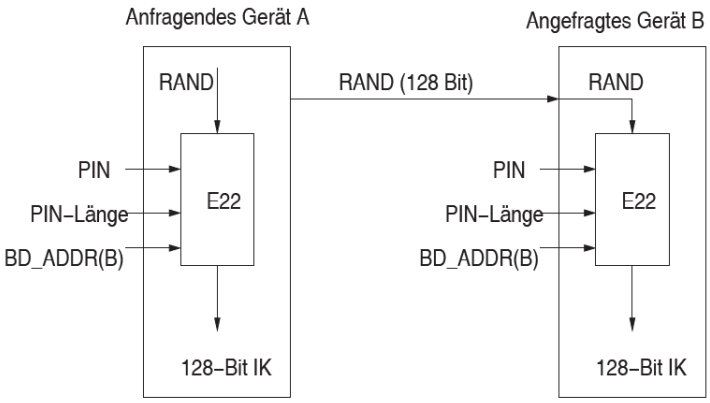


Abbildung 10. Berechnung IK[2, p. 951]

Verbindungsschlüssel Nach erfolgreicher Berechnung des IK muss der Verbindungsschlüssel (Link-key) gewählt werden. Dabei kann zwischen Geräte-, Kombinations- und Masterschlüssel gewählt werden. Jeder dieser Schlüssel ist 128 Bit lang.
Der Geräteschlüssel ist gerätespezifisch und wird bei der ersten Nutzung eines Geräts erstellt. Unter Einbeziehung der Geräteadresse und einer 128 Bit Zufallszahl berechnet der E21 Algorithmus den Schlüssel. Dieser wird anschließend im nicht flüchtigen Speicher des Geräts abgelegt.
Da Bluetooth Geräte die Verbindungsschlüssel von authentifizierten Geräten speichern, führt das bei geringem Speicher zu Problemen. Betroffene Geräte verwenden deshalb den Masterschlüssel als Verbindungsschlüssel. Dafür wird dieser exklusiv-oder mit dem IK verknüpft. Der Kommunikationspartner erhält durch erneutes exklusiv-oder verknüpfen mit dem IK den Masterschlüssel.

Masterschlüssel Der Masterschlüssel dient bei Punkt-zu-Multipunktverbindungen als Verbindungsschlüssel. Dieser ersetzt temporär den eigentlichen Verbindungsschlüssel. Der Master im Netzwerk berechnet dafür mittels des E22 Algorithmus und zwei Zufallszahlen den 128 Bit Masterschlüssel. Alle Slaves verwenden dann den Masterschlüssel zur Berechnung des Kommunikationsschlüssels. Der Master kann im weiteren Verlauf die Verwendung des ursprünglichen Verbindungsschlüssels auffordern.

Kombinationsschlüssel Bluetooth erfordert bei jeder Verbindung eine erneute Authentifizierung. Dazu gehört auch das Pairing, welches eventuell Benutzereingaben erfordert und dementsprechend Zeit kostet. Um dieses Problem zu umgehen, kann ein Kombinationsschlüssel berechnet und auf beiden Geräten persistent gespeichert werden. Der Ablauf der Schlüsselerzeugung ist in Abbildung 11 sichtbar.
Beide Geräte berechnen mittels des E21 Algorithmus, ihrer eigenen Geräteaddresse und einer Zufallszahl einen Schlüssel. Anschließend wird die Zufallszahl exklusiv-oder mit dem vorher berechneten IK verknüpft und an den jeweils anderen gesendet. Da beide Parteien über den gleichen IK verfügen, können sie durch erneutes eklusiv-oder verknüpfen der erhaltenen Nachricht die Zufallszahl des jeweils anderen erhalten. In der Abbildung 11 erkennt man nun, dass die Geräte in Schritt 3 jeweils die Berechnung des anderen in Schritt 1 durchführen. Beide erhalten damit den Schlüssel des Anderen. Durch exklusiv-oder Verknüpfung beider Schlüssel erhalten sie dann den Kombinationsschlüssel.

Nach der Wahl und Berechnung des Verbindungschlüssels kann nun das im vorherigen Abschnitt beschriebene Authentifizierungsprotokoll erfolgreich durchgeführt werden. Nach erfolgreicher Authentifizierung muss ein Kommunikationsschlüssel, der zur Verschlüsselung der Nachrichten dient, berechnet werden.

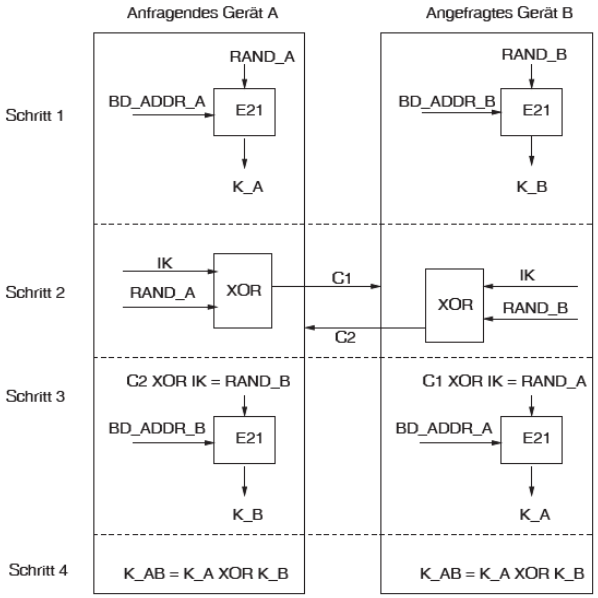


Abbildung 11. Berechnung Kombinationsschlüssel[2, p. 952]

Kommunikationsschlüssel Der Kommunikationsschlüssel wird mittels des E3 Algorithmus berechnet. Als Eingabe werden der vorher berechnete Verbindungsschlüssel, eine 128 Bit Zufallszahl und eine 96 Bit Ciphering Offset Number (COF) verwendet. Die COF wird aus dem ACO Wert des Authentifizierungsprotokolls abgeleitet. Die Länge des berechneten Schlüssels variiert zwischen 8 und 128 Bit. Dadurch kann Bluetooth auch in Ländern verwendet werden, welche die erlaubte Schlüssellänge einschränken. Jedes Gerät definiert deshalb eine maximale und minimale Länge des Kommunikationsschlüssels. Bei der Aushandlung der Schlüssellänge startet der Master immer mit der maximalen Länge. Die Länge wird dann runtergehandelt bis eine Einigung gefunden wurde. Im worst case bedeutet dies einen 8 Bit Schlüssel. Auf Anwendungsebene kann jedoch definiert werden, wie groß die minimale Schlüssellänge sein muss. Die Aushandlung wird abgebrochen, wenn diese Größe unterschritten wird.

Verschlüsselung Der Payload jeder Nachricht wird mit dem E0 Algorithmus, einer symmetrischen Stromchiffre, verschlüsselt. Als Schlüssel wird der zuvor berechnete Kommunikationsschlüssel verwendet. Durch Kryptoanalyse wurde gezeigt, dass Angriffe auf E0 mit einem Aufwand von mindestens $\mathcal{O}(2^{66})$ möglich sind. Auf Grund dieser Komplexität sind Angriffe nicht praxisrelevant.

Schwächen & Probleme

Geräteauthentifizierung Die gerätebasierte Authentifzierzung und Authorisierung ist problematisch, wenn ein Gerät entwendet wird. Da auf erneute Authentifizierung mit PIN Eingabe häufig verzichtet wird, kann der Angreifer das Gerät weiterhin verwenden, um auf alle bereits authentifizierten Geräte zuzugreifen.

Geräteschlüssel als Verbindungsschlüssel Die Verwendung des Geräteschlüssels als Verbindungsschlüssel bei Geräten mit wenig Speichern führt zu Problemen. Jedes Gerät, welches sich einmal mit einem speicherarmen Gerät authentifiziert hat, verfügt über dessen Geräteschlüssel. Da dieser als Grundlage für den Kommunikationsschlüssel dient, können alle Nachrichten entschlüsselt werden. Um diese abzuhören muss jedoch auch die richtige Frequenz gewählt werden.

Masterschlüssel Bei der Verwendung des Masterschlüssels als Verbindungsschlüssel verfügen alle Geräte in einer Multipunktverbindung über den selben Kommunikationsschlüssel. Dadurch ist es ihnen möglich, auch Nachrichten, die nicht für sie bestimmt sind, zu entschlüsseln.

PIN basiert Die Berechnung der Verbindungsschlüssel basiert auf dem Initialisierungsschlüssel, welcher wiederum auf der PIN basiert. Jegliche Schwächen der PIN beeinträchtigen dementsprechend die gesamte Sicherheit der Verschlüsselung. Da die PIN Eingabe Zeit kostet und lästig ist, wird häufig entweder eine kurze PIN vom Benutzer verwendet, oder die standardmäßig gesetzte PIN 0000 verwendet. Dies macht es einfach, die PIN zu knacken.
Eine weitere Schwäche liegt in der Bluetooth Spezifikation, welche die Übertragung der PIN über die Luftschnittstelle vorsieht. Dies macht es für Angreifer einfach, diese abzufangen. Erneut ist zu beachten, dass der Angreifer die jeweils richtige Frequenz abhören muss. Das BTCrackTool ermöglicht es aus abgefangenen Pairing-Daten und aus diesen den Verbindungsschlüssel herzuleiten.

Man-in-the-Middle Nachdem der Angreifer im Besitz des Verbindungsschlüssels ist, kann dieser ein Man-in-the-Middle Angriff durchführen. Dafür nimmt er Kontakt mit zwei Geräten auf und gibt sich als der jeweils andere aus. Damit dieser Angriff nicht erkannt wird, muss auf unterschiedlichen Frequenzen gesendet werden. Deshalb fordert der Angreifer die Ziele dazu auf, zum Netzwerkmaster zu werden. Diese können die Anfrage jedoch ablehnen, wodurch der Angriff abgewehrt wird. Geschieht dies nicht, kann der Angreifer die Geräte zur Aushandlung neuer Verbindungsschlüssel auffordern und sich daraufhin bei beiden Geräten als den jeweils anderen authentifizieren.

Secure Simple Pairing Das Secure Simple Pairing (SSP) ist eine Erweiterung des Pairings und soll Eavesdropping als auch MIM Angriffe während des Pairings verhindern. Dieses wurde bereits beim Sicherheitsmodus 4 kurz erwähnt. Das SSP ist in fünf Phasen eingeteilt:

1. Public-Key Exchange basierend auf Diffie-Hellman
2. Authentifizierung

3. Challenge
4. Berechnung Verbindungsschlüssel
5. Berechnung Kommunikationsschlüssel

Auf nährere Details der Phasen wird hier nicht weiter eingegangen. Der Hauptaspekt ist, dass die Authentifizierung des Key-Exchanges für zusätzliche Sicherheit sorgt. Die dafür zur Verfügung stehenden Optionen sind: numerischer Vergleich, Just Works, Out-of Band und Passkey Entry. Welcher dieser Varianten genutzt wird, kann im Sicherheits Modus 4 definiert werden [2].

2.4 Logical Link Control and Adaptation Protocol (L2CAP)

Genau wie das LMP befindet sich auch L2CAP im Data link layer. Dieses setzt, wie in Abbildung 2 sichtbar ist, auf ACL auf. Bluetooth ermöglicht durch das "KabelersatzprotokollRFCOMM auch die Nutzung von Protokollen, die einen seriellen Port erfodern. Dazu gehört unter anderem TCP/IP. Eine Übersicht der unterstützen Protkolle ist in der Abbildung Protokollstack sichtbar. Durch Paketfragmentierung ist es L2CAP möglich, höheren Protokollen das senden von bis zu 64KB großen Paketen zu erlauben. Das Basisband kann lediglich 341 Byte große Pakete versenden, weshalb die Fragmentierung und anschließende Zusammenfügung durch L2CAP erforderlich ist. Da das Basisbandprotokoll nicht in der Lage ist, Protokoll-Multiplexing durchzuführen, wird dies auch von L2CAP übernommen.
L2CAP stellt außerdem Kommunikationskanäle (ATT bearers) bereit, die vom Attribute Protocol verwendet werden [2].

2.5 Service Discovery Protocol

Im Abschnitt zu Sicherheitsmodi wurde bereits erwähnt, dass Sicherheitsfunktionen durch die Dienstebene bereitgestellt und umgesetzt werden. Was genau Dienste (services) sind und diese in Bluetooth funktionieren, wird in diesem Abschnitt erklärt.

Dienst Dienste sind dafür verantwortlich Informationen bereitzustellen oder Daten zu manipulieren. Sie werden in Primär- und Sekundärdienste aufgeteilt. Primärdienste stellen dabei die eigentliche Funktion bereit, während Sekundärdienste Hilfsfunktionalitäten bereitstellen. Am Beispiel eines Thermometerdienstes kann man sich das so vorstellen, dass der Primärdienst die Temperatur abliest und ein Sekundärdienst die Temperatur anschließend von Grad Celcius in Kelvin umwandelt.
Dienste können entweder als Software, Hardware oder als Kombination von beiden implementiert werden. Eine Einschränkung ist jedoch, dass backwards compability gewährleistet werden muss. Es können also zusätzliche Variablen (Characteristics) und Funktionalitäten hinzugefügt werden, die ursprüngliche

Funktion muss aber erhalten bleiben. Dies kann man sich ähnlich wie in der objektorientiertren Programmierung vorstellen. In diesem Fall wäre ein Dienst ein Interface und die Implementierung eine Klasse. Man kann zwar innerhalb der Klasse die Implementierungsdetails ändern, jedoch keine Änderungen an der Schnittstelle vornehmen.

Dienste definieren außerdem Characteristics, die man sich wie Klassenvariablen vorstellen kann. Diese werden durch 16 oder 128 Bit UUIDs identifiziert. Bei der Entwicklung eigener Dienste sollte man also darauf achten, dass man nicht bereits reservierte IDs für seinen Dienst verwendet. In der Abbildung Characteristics sieht man einen Ausschnitt von standardmäßig definierten Characteristics. Die Characteristics eines Dienstes kann man sich wie einen Key-Value Store vorstellen. Der Key ist dabei ein 16 Bit unsigned integer der innerhalb der Service Klasse eindeutig sein muss. Die Länge und Inhalte der Werte sind jeweils variabel.

Name	Uniform Type Identifier	Assigned Number	Specification
Aerobic Heart Rate Lower Limit	org.bluetooth.characteristic.aerobic_heart_rate_lower_limit	0x2A7E	GSS
Aerobic Heart Rate Upper Limit	org.bluetooth.characteristic.aerobic_heart_rate_upper_limit	0x2A84	GSS
Aerobic Threshold	org.bluetooth.characteristic.aerobic_threshold	0x2A7F	GSS
Age	org.bluetooth.characteristic.age	0x2A80	GSS
Aggregate	org.bluetooth.characteristic.aggregate	0x2A5A	GSS
Alert Category ID	org.bluetooth.characteristic.alert_category_id	0x2A43	GSS
Alert Category ID Bit Mask	org.bluetooth.characteristic.alert_category_id_bit_mask	0x2A42	GSS
Alert Level	org.bluetooth.characteristic.alert_level	0x2A06	GSS

Abbildung 12. Service Characteristics Beispiele [6]

Durch Service Klassen werden die Attribute eines Dienstes definiert. Jede Service Klassen muss eine eindeutige ID besitzen. Wie in der objectorientierten Programmierung, können Dienste auch von anderen Diensten erben. In diesem Fall werden alle Attribute der Superklasse geerbt. Ein Beispiel für Vererbung wäre ein Musikdienst, der das grundlegende Abspielen von Musik definiert. Eine weitere Service Klasse könnte dann von dieser erben und Musikstreaming über

TCP/IP ermöglichen.
Ein großer Vorteil von Diensten ist die Wiederverwendbarkeit. Dienste können andere Dienste einbinden. Der Grad-Celcius-Kelvin Dienst könnte also auch von anderen Diensten eingebunden und genutzt werden. Bei der beschriebenen Komposition gibt es keine Einschränkung bezüglich der maximal erlaubten Anzahl eingebundener Dienste.

Um Dienste auf einem Gerät ausfindig zu machen wird das Service Discovery Protocol (SDP) verwendet. Eine vereinfachte Kommunikation ist dabei in der Abbildung 13 sichtbar. Der SDP Client schickt eine Anfrage an den SDP Server und erhält anschließend Informationen über den angegebenen Dienst. Dienstinformationen werden auf dem SDP Server in einer SDP Datenbank aufbewahrt. Zu jedem Dienst gibt es einen Eintrag, der als Service Record bezeichnet wird.

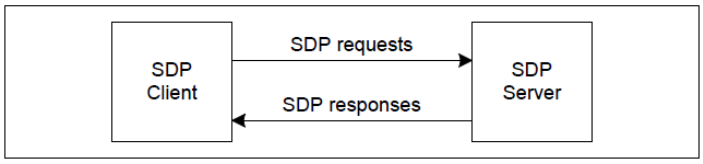


Abbildung 13. SDP Kommunikation [3, p. 1213]

Ein Service Record wird eindeutig durch eine 32 Bit ID identifiziert. Dabei darf die ID auf dem SDP Server nur einmal vergeben werden. Es ist nicht festgelegt, welcher Dienst eine jeweilige ID erhält. Der gleiche Dienst kann also auf zwei Servern unterschiedliche IDs besitzen. Die ID 0x00000000 ist teilweise für den SDP Server selbst reserviert und enthält Informationen über unterstützte Protkolle.
Eine weitere Möglichkeit Dienste aufzufinden ist über die Search Service Transaction. Dabei sendet der SDP Client eine List von Attributwerten. Der SDP Server guckt dann in den Service Records nach Einträgen in denen die gesendeten Werte als Attributwerte gesetzt sind. Sind alle Werte in dem Service Record enthalten, wird die ID des Records in der Antwort des Servers zurückgegeben. Anschließend kann der Client eine SDP Request mit einer Record ID stellen um so die vollständigen Informationen des Dienstes zu erhalten [3].

2.6 Bluetooth Profile

Ziel von Bluetooth Profilen ist die Interoperabilität von Anwendungen zu gewährleisten. Aus diesem Grund beschreibt ein Profil die Interaktion von mehreren Komponenten. Dazu gehört die Interaktion der einzelnen Schichten, das Verhalten von Anwendungen, die Schichteninteraktion von zwei Geräten, die ausgetauschten Da-

tenformate als auch die Service Discovery Anforderungen. Stimmen zwei Anwendungen in den genannten Punkten überein, so können diese Zusammenarbeiten. Beispiele für Profile sind das Advanced Audio Distribution Profile (A2DP) für audio streaming und das Video Distribution Profile (VDP) für Videoübertragung [8].

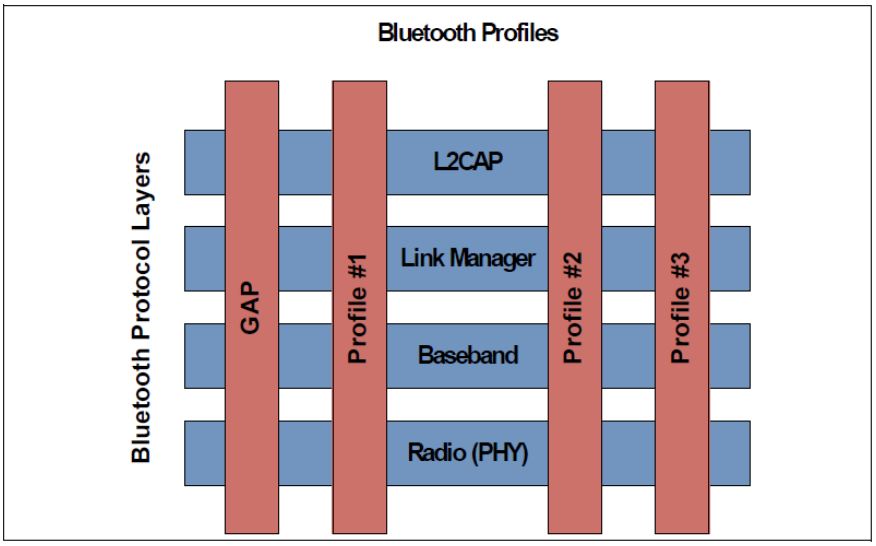


Abbildung 14. Profil [3, p. 281]

Generic Access Profile Das Generic Access Profile (GAP) wird von jedem Bluetooth Gerät implementiert und definiert grundlegende Anforderungen. Abhängig von der grundlegenden Technologie unterscheidet sich das GAP. Es ist an dieser Stelle deshalb wichtig, diese kurz zu erwähnen.

Bluetooth Technologien Bluetooth unterscheidet sich grundlegend in zwei Technologien: Basic Rate / Enhanced Data Rate (BR/EDR) und Low Enery (LE). BR/EDR ist vorallem für Anwendungsfälle mit hoher Datenrate und kurzen Entfernungen ausgelegt. Darunter fällt beispielsweise Musikstreaming. Bluetooth LE kennzeichnet sich, wie der Name schon sagt, durch einen geringeren Energieverbrauch aus. Zeitgleich verfügt es jedoch über den gleichen Funktionsumfang wie BR/EDR. Bluetooth LE wurde mit Version 4.0 eingeführt und ist nicht backwards kompatibel.
Bei BR/EDR umfasst GAP Definitionen zum Radio, Baseband, Link Manager, L2CAP und dem SDP. Bei Bluetooth LE hingegen zum Physical und Link Layer, L2CAP, dem Security Manager, Attribute Protocol und GAT.
Basierend auf dem Generic Access Profile können weitere Profile angelegt wer-

den. In diese können dann weitere Anforderungen definiert werden. Diese Profile werden als Generic Profile bezeichnet. Profile, die Anwendungsinteroperabilität beschreiben heißen Application Profile. In der Abbildung 16 ist die Hierarchie der Profile nochmal dargestellt. Man erkennt in diesem auch deutlich, dass Profile alle Spezifikation der unteren Profilschicht umfassen und entsprechend implementieren müssen.

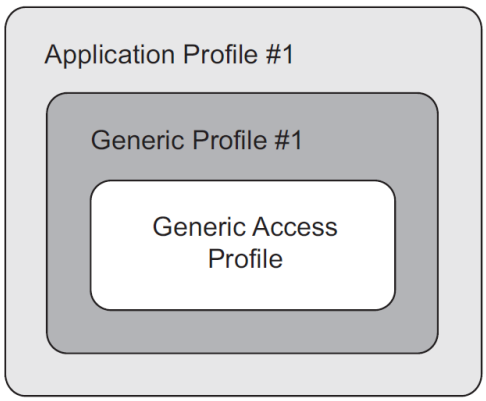


Abbildung 15. Profilhierarchie [3, p. 283]

Attribute Protocol Das Attribute Protocol (ATT) ermöglicht Clients das auslesen und manipulieren von Serverwerten. Jeder Wert ist dabei ein Attribut, dessen Datentyp durch eine UUID identifiziert wird. Das Protokoll verwendet L2CAP Kanäle zur Datenübertragung.

Generic Attribute Architecture Aufbauend auf ATT gibt es auch die Generic Attribute Architecture (GATT). Diese ist Teil der Service Discovery in Bluetooth LE. Die Implementierung in BR/EDR ist optional. Das GATT Profil definiert jeweils wie Anwendunge Daten austauschen [4].

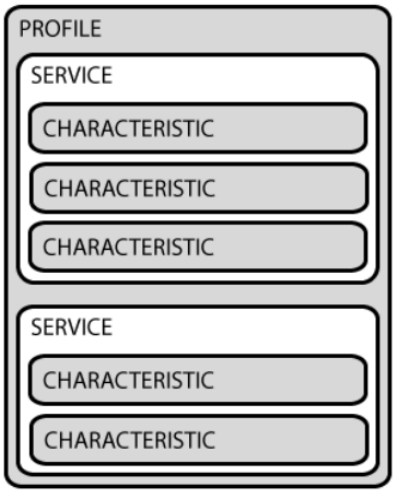


Abbildung 16. GATT Profil [2]

3 Neue Änderungen

3.1 Version 5.1

GATT Caching Bisher hat der Server den Client informiert, wenn sich seine Attribute Table geändert hat. Der Client hat dann mit einer Bestätigung geantwortet und per Service Discovery die aktuellen Werte heruntergeladen. Dafür musste der Server Informationen über jeden verbundenen Client und deren zuletzt geladenen Daten pflegen. Mit dem neuen GATT Caching ist dies nicht mehr nötig.
Der GATT Server verwaltet nun eine Liste mit Clients und dem Daten-Hash zum Zeitpunkt an dem diesen zuletzt die Daten geladen haben. Clients können dann anfragen, ob sich die Daten geändert haben. Der Server vergleicht dann den letzten Clienthash mit dem aktuellen Datenhash. Falls die Daten sich geändert haben stimmen die Hashwerte nicht mehr überein und der Client kann die neuen Daten per Service Discovery herunterladen.

Direction finding Bisherige Bluetooth Ortungssysteme benutzen die Signalstärke (RSSI) um Entfernungen abzuschätzen. Dieser Lösungsansatz wurde auch in der deutschen Corona Warn App [9] gewählt. Die neue Direction finding Funktionen ermöglichen die Bestimmung des Angle of arrival (AoA) und Angle of departure(AoD). Ein sendendes Gerät kann jeweils den AoD berechnen während der Empfänger den AoA berechnet. Um diese Werte zu berechnen sind jedoch mehrere Antennen erforderlich. In Abbildung 17 ist die Funktionsweise dargestellt.

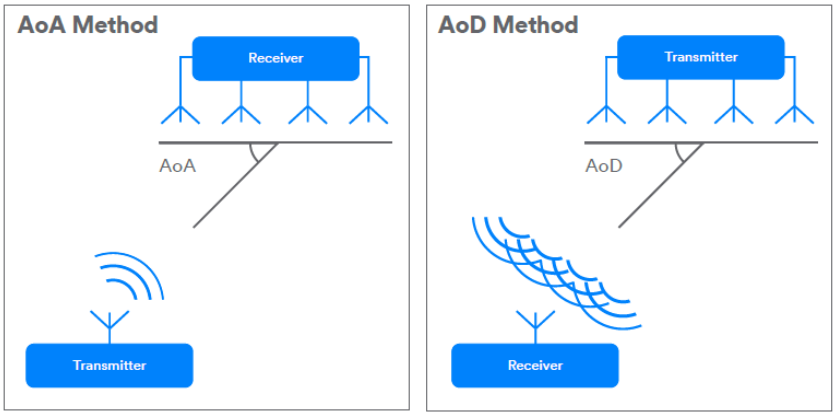


Abbildung 17. Direction finding [5]

Advertising Verbesserungen Advertising wird verwendet, um Informationen von sich selbst an Geräte in der Umgebung zu senden. Ein Problem dabei war, dass Geräte den Adversiting Channel in einer festgelegten Reihenfolge gewählt haben. Durch die zufällige Wahl eines Channels wird nun die Wahrscheinlichkeit, dass Advertising Pakete kollidieren, verringert.
Durch Periodic Advertising Sync Transfer (PAST) ist es Geräten nun möglich, dass ein Gerät Synchronisierungsdetail aus Advertisements mit anderen Geräten teilt. Dadurch können Geräte, die momentan selbst beschäftigt sind, von einem anderen Gerät aktuelle Informationen erhalten und müssen keinen eigenen Aufwand betreiben, um an diese Daten zu gelangen. Ein Beispiel dafür ist in der Abbildung 18 sichtbar [5].

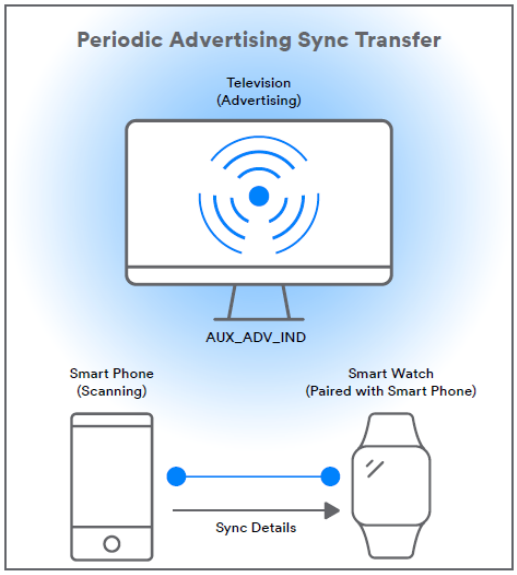


Abbildung 18. Anwendungsfall PAST [5]

3.2 Version 5.2

LE Isochronous Channel Die neuen LE Isochronous Channel ermöglichen die synchronisierte Übertragung von Daten an mehrere Kommunikationspartner. Eine Anwendung der Channel ist Musikstreaming von einem zentralen Bluetooth Gerät an mehrere Geräte in der Nähe. Die Kanäle dienen außerdem als Grundlage für LE Audio, die neue Generation der Bluetooth Audioübertragung.

Enhanced Attribute Protocol Das Enhanced Attribute Protocol (EATT) ist eine Erweiterung des Attribute Protocol. EATT erlaubt im Gegensatz zu ATT die Durchführung paralleler Transaktionen. Dadurch wird die Latenz bei Geräten, auf denen mehrere Bluetooth LE Anwendungen laufen, reduziert. EATT führte außerdem einen neuen L2CAP Kanal mit Flow control ein. Datentransporte über diese Kanäle sind also deutlich zuverlässiger. Ein Sicherheitsvorteil von EATT ist, dass dieses nur über eine verschlüsselte Verbindung genutzt werden kann.

LE Power Control LE Power Control ermöglicht es Kommunikationspartnern die Signalstärke, die zur Übermittlung verwendet wird, dynamisch anzupassen. Dabei wird an Hand der von Empfängern wahrgenommenen Signalstärke eine Verstärkung oder Abschwächung der Signalstärke angefordert. Durch die Anpassung der Signalstärke ist es möglich Energie zu sparen. Die Anpassung der Signalstärke verbessert außerdem die Zuverlässigkeit, da sichergestellt wird,

dass diese in einem optimalen Bereich liegt. Bei geringerer Signalstärke werden Geräte, die sich in der Nähe befinden weniger beeinflusst, da die Reichweite, in der Interferenzeffekte auftreten können, reduziert wird. Dies hat Vorteile für alle Geräte, welche den 2,4 GHz Frequenzbereich verwenden. Besonders auf kleinen Räumen mit vielen Bluetoothgeräten wird dies für Verbesserungen der Übertragungsqualität sorgen [4].

4 Fazit

Die ursprünglichen Ziele eine günstige und stromsparende Lösung für kurze Distanzen zu finden wurden erreicht. Durch die Bluetooth LE Technologie und neue Funktionen wie LE Power Control wird der Energieverbrauch weiter reduziert. Durch die Verwendung von Profilen und Diensten wurde es nicht nur geschafft die drahtlose Kommunikation zu vereinheitlichen sondern auch die Anwendungen, die auf dieser aufbauen. Die Wiederverwendbarkeit und Komposition von Diensten macht es außerdem für Entwickler einfacher neuer Anwendung zu entwickeln.
Die Sicherheitsprobleme bei Bluetooth entstehen vorallem durch die gerätebasierte Authentifizierung und die Abhängigkeit vom PIN. Mit Secure Simple Pairing wurde jedoch eine Lösung gefunden, um den PIN Austausch sicherer zu gestalten und das Risiko von Eavesdropping als auch dadurch möglichen Man-in-the-middle Angriffe zu reduzieren.
Ein weiterer Negativpunkt ist, dass Bluetooth standardmäßig keine Ende-zu-Ende Sicherheit sondern nur Punkt-zu-Punkt Sicherheit bietet. Ist E2E Sicherheit gewünscht, muss diese durch Protokolle wie SSL/TLS zusätzlich hinzugefügt werden.
Zwar bietet Bluetooth Sicherheitsmodi an, die einen grundlegenden Schutz und Verschlüsselung anbieten, standardmäßig sind Modus 2 und 3 jedoch deaktiviert, wodurch der Endbenutzer keinen Vorteil daraus zieht.
Auf Grund dieser Bedenken sollte Bluetooth nur für sicherheitskritische Anwendungen verwendet werden, wenn die Anwendung selbst zusätzliche Sicherheitskonzepte umsetzt oder diese durch Einbindung anderer Dienste hinzufügt.

Literatur

1. Bluetooth Architektur, Wikipedia https://upload.wikimedia.org/wikipedia/commons/thumb/9/9f/Bluetooth_protokoly.svg/1920px-Bluetooth_protokoly.svg.png
2. Eckert, C.: IT-Sicherheit: Bluetooth. Walter de Gruyter(2013)
3. Bluetooth Core Specification Version 5.2, Version 1, https://www.bluetooth.com/de/specifications/bluetooth-core-specification/
4. Bluetooth Core Specification Version 5.2 Feature Overview, Version 1, https://www.bluetooth.com/wp-content/uploads/2020/01/Bluetooth_5.2_Feature_Overview.pdf

5. Bluetooth Core Specification Version 5.1 Feature Overview, Version 1, https://www.bluetooth.com/bluetooth-resources/bluetooth-core-specification-v5-1-feature-overview/
6. https://www.bluetooth.com/de/specifications/gatt/characteristics/, abgerufen: 22.06.2020
7. https://image.slidesharecdn.com/lecture3spreadspectrumtechnologies-151115073210-lva1-app6892/95/spread-spectrum-technologies-14-638.jpg?cb=1447572747, abgerufen: 22.06.2020
8. https://de.wikipedia.org/wiki/Bluetooth-Profile, abgerufen: 25.05.2020
9. https://github.com/corona-warn-app/cwa-documentation, abgerufen: 22.06.2020

Aspects of 5G Security

Daniel Nelle
University of Potsdam,
Potsdam, Germany
dnelle@uni-potsdam.de

ABSTRACT

With the 5G rollout to the general public underway, it is insightful to examine the improvements and changes the new standard brings regarding security, not only in comparison with its predecessor, 4G, but also in the context of new functions and use-cases. This paper will take a brief look at the new service-based security architecture of 5G, explain how the new authentication prodedure increases security through identity protection and home network control and finally analyze how the much anticipated feature of network slicing might be protected in a 5G environment.

1 INTRODUCTION TO 5G

Mobile carriers across the globe are increasingly rolling out 5G to end users while device manufacturers are announcing more and more devices capable of using the new standard. This happens against a background of political noise generated by governments trying to limit China's influence in the markets it sells its mobile equipment to. A significant part of the population has thus heard of this technology, claimed by some to be poised to change our lives throughout the next decade. Most of those familiar with the term 5G from advertisements and evening news coverage, however, seem to associate it with higher data rates and see it as the final blow to loading bars on high-definition content and choppy images of relatives/ colleagues in video calls.

While increased data rates certainly are one of 5G's outstanding features, it is unlikely that it was this improvement that caused opinion leaders to declare it to have such profound impact on the 2020's. We thus begin this short treatise on aspects of 5G security with a brief introduction to the standard and its features and only then will turn to the architecture and some of the mechanisms intended to make it safer and more flexible than its predecessor, 4G. Afterwards we will examine one of the other features of 5G, network slicing, more closely and look at the implications this technology has from a security perspective. Finally, we will summarize our findings in a brief conclusion.

1.1 What is 5G?

5G is a mobile standard specified by the 3rd Generation Partnership Project (3GPP), an umbrella term for a number of standards organizations developing mobile communication protocols. Its chief features are composed of four use cases which will allow an improvement to existing or enable completely new applications, often related, but not limited to, the Internet of Things (IoT), or, to employ a broader term, the *Internet of Everything*. The most prominent feature is enhanced Mobile Broadband (eMBB), which will allow peak data rates of 10 GB/s [10] and will be what most people will have direct contact with through mobile devices such as smartphones. Massive Machine-Type Communications (MMTC) is supposed to allow an excess of one million connections per square kilometer [10], thus enabling interconnected, sensor-rich environments such as "smart" factories and cities. Ultra-Reliable Low-Latency Communications (URRLC) aims at latencies lower than one millisecond and 99,9999% reliability [10], paving the way for applications such as Vehicle-to-vehicle (V2V) communications. Compared to 4G it is conceived as a Service Based Architecture (SBA) [6], where control plane functionality and data repositories are provided by means of interconnected Network Functions (NFs). Those expose their services through well-defined Representational State Transfer (REST) Application Programming Interfaces (APIs) and thus enable the respective components to be virtualized and distributed [6]. This in turn permits *network slicing*, where resources on the mobile network can be offered in an Infrastructure-as-a-service (IaaS) manner, much like popular cloud operators already do with compute and regular network services.

1.2 5G Architecture

Figure 1 gives a rough overview of the 5G architecture. Pictured in the middle is the 5G Core (5GC), which houses the aforementioned NFs, such as the Authentication Mobility Function (AMF)/Session Management Function (SMF), responsible for authentication in a roaming context, User Plane Function (UPF), managing connections to Data Networks (DNs), the Non-3GPP Interworking Function (N3IWF), serving as an endpoint to untrusted non-3GPP access, and the Authentication Server Function (AUSF), another component involved in authentication. The node captioned *5G NF* is a placeholder for all other possible NFs, such as functions related to V2V. On the far left of Figure 1 some of the devices expected to be connecting to the 5G core are pictured, together with two possible access methods: The wireless variant via a Radio Access Network (RAN), a mode most

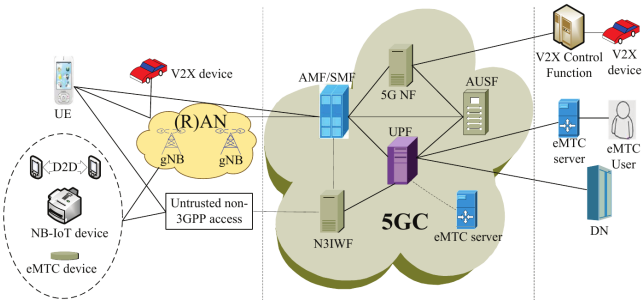


Figure 1: 5G Architecture [4]

users will associate with how their mobile phone gets a network signal, where the device establishes a radio connection with a base station, called Next generation NodeB (gNB) in a 5G context [2]. Additionally, 5G allows so-called *untrusted non-3GPP access* via Wi-Fi and cable connections, where the corresponding NF serves as an intermediary. A range of devices other than regular mobile handsets (called called User Equipment (UE) here), is expected to be part of the network. Those include, but are not limited to, Device-to-device (D2D) communications enabled devices, Narrow Band IoT (NB-IoT) devices, enhanced Machine Type Communication (eMTC) devices and Vehicle-to-everything (V2X) enabled smart cars.

2 5G SECURITY ARCHITECTURE

Figure 2 shows 5G schematically from a security perspective and adds roaming to the picture, where a device connects to its Home Public Land Mobile Network (PLMN) through a Visited PLMN. End users know this scenario from when they are traveling abroad and using their mobile devices or, in some countries, when their phone uses a different carrier's network because the own carrier does not provide coverage in a given area. The AUSF is located in the home network and plays a central role in the authentication process of a UE wanting to join a network. It relies on the User Data Management (UDM) for keys and other services, which this provides through two functions: The Authentication Credential Repository and Processing Function (ARPF), responsible for selecting an appropriate authentication method as well as computing keys [5] and the Subscriber Identity De-concealing Function (SIDF), taking care of encrypting and decrypting the UE's unique identifier. The SMF is located in the visited network and provides session management as well as DHCP and IP allocation services. The AMF is located in the visited network, too, and plays the most important function here, since it acts as a middleman between UE and the home network [1]. It is co-located with the Security Anchor Function (SEAF) holding the anchor key for the visited network, from which all other keys are derived [1]. The UPF

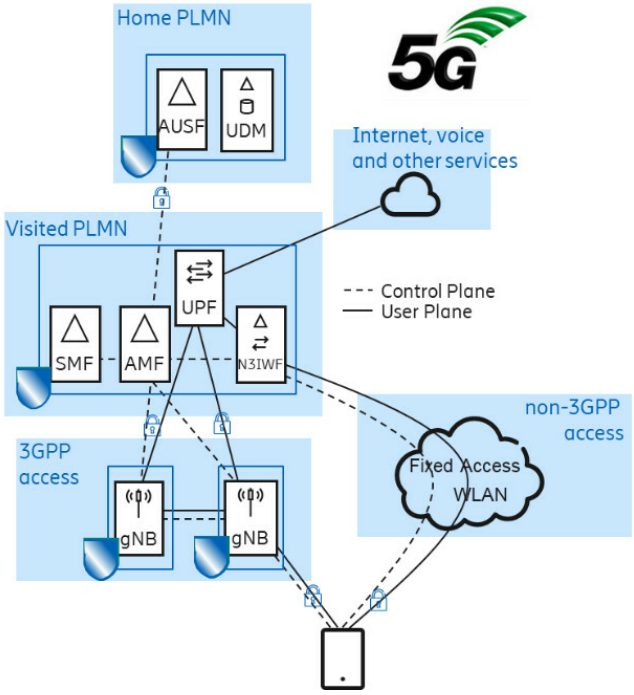


Figure 2: 5G Security Architecture [7]

manages packet routing and forwarding and is thus responsible for connecting the user to services such as the Internet. The N3IWF acts as a VPN endpoint for connections not established via a 5G radio connection. A gNB, usually depicted as one component, in fact consists of a Distributed Unit (DU) and a Central Unit (CU). The former is a "dumb" component unable to access any data it forwards and is intended to be deployed to remote sites vulnerable to illegal access, while the latter is where *access stratum security* is terminated and which is intended for locations where access can be more strictly controlled. This means that data encrypted for wireless transmission over the RAN is decrypted here [1]. As the padlocks on the links indicate traffic and control messages are encrypted when being transferred between access stratum, visited network and home network while data exchanged between the UPF and the Internet is not.

3 5G ACCESS

The 5G access procedure is improved in several ways compared to its predecessor. Support for multiple authentication methods allows a wider variety of use cases and enables equipment without a Universal subscriber identity module (USIM)(aka SIM-card) to participate in the network. A major improvement is better identity protection, because the SUbscription Permanent Identifier (SUPI), an identifier hard-coded onto the SIM-card, now is always encrypted using

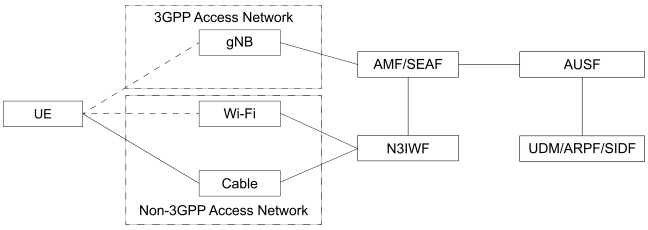


Figure 3: Security Contexts [3]

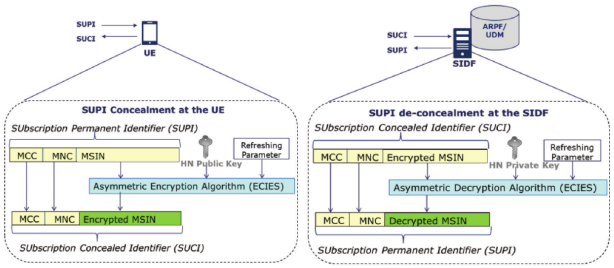


Figure 4: SUPI Encryption and Decryption [8]

the home network public key. Another big advancement is enhanced home network control through the authentication procedure. While in 4G the visited network had to be trusted, 5G puts the final responsibility on the home network and gives it the power to verify an authentication attempt by requiring proof of the UEs participation in the exchange. Before that, an attacker could feign the presence of an UE in the network and thus carry out a man-in-the-middle attack. However, with all the improvements in place, tracking of a UE might still be possible [3]. In the following sections we will examine how increased security and privacy is achieved in 5G more closely.

3.1 5G Authentication Framework

In order to allow 5G authentication to be open as well as access-network agnostic, a unified authentication framework depicted in Figure 3 has been defined for 5G. It makes it mandatory to implement at least two authentication options: 5G Authentication and Key Agreement (AKA) as well as (Extensible Authentication Protocol (EAP)-AKA'). EAP support ensures the openness requirement is met, while the introduction of the N3IWF allows access to the 5G core over untrusted alternatives such as Wi-Fi and cable by putting the traffic through IPsec tunnels [3]. The framework also supports the establishment of several security contexts with only one authentication pass, thus allowing a device to move between 3GPP and non-3GPP seamlessly without the overhead of another authentication [3]. Finally, support for EAP-Transport Layer Security (TLS) is possible, which allows equipment without a USIM to participate. We will examine 5G-AKA more closely in a later section.

3.2 Increased Privacy

Figure 4 shows how a subscriber's identity is protected through encryption in 5G. The SUPI, a unique identifier, consist of Mobile Country Code (MCC), Mobile Network Code (MNC) and Mobile Subscriber Idendification Number (MSIN) [8]. Since the former two components have to be readable and are not considered sensitive information, only the MSIN is protected. On the UE the home network public key is used to encrypt the MSIN via an asymmetric encryption algorithm to produce the SUbscription Concealed Identifier (SUCI) [1]. This protected identifier is used for transmissions over unsafe channels. Only after a successful authentication the visited network gains access to the decrypted form. On the other side, in the home network, the SIDF, which is hosted by the UDM, is able to decrypt the MSIN by using the home network private key. This scheme prevents i.e. an attack via a fake base station, where the attacker could insert herself between subscriber and home network and thus carry out a man-in-the-middle attack [8].

3.3 5G Authentication and Key Agreement

Let us now take a closer look at the 5G AKA message exchange depicted in Figure 5. We will see where the aforementioned improvements, namely identity protection and home network control, come into effect [3].

(1) The SEAF starts the authentication procedure after receiving any signalling message from the UE. This message should include a SUCI or Globally Unique Temporary Identifier (GUTI). For the sake of brevity, we will focus on the former case.
(2) The SEAF starts the procedure by sending a request to the AUSF.
(3) The SEAF verifies the request is authorized and forwards the request to the UDM.
(4) The UDM decrypts the SUCI in the SIDF and lets the ARPF select the appropriate authentication method, 5G-AKA in our case.
(5) The UDM initiates AKA by sending an authentication vector consisting of an AUTH token, a XRES (expected response) token, a key K_{AUSF} and the SUPI, if it was included earlier, to the AUSF.
(6) The AUSF stores the key K_{AUSF} and computes a hash of XRES, called HXRES.
(7) The AUSF passes the AUTH token and HXRES to SEAF. **Note how the SUPI is not sent here**.
(8) The SEAF forwards the AUTH token to the UE.

[1] Spelled *Sushi*

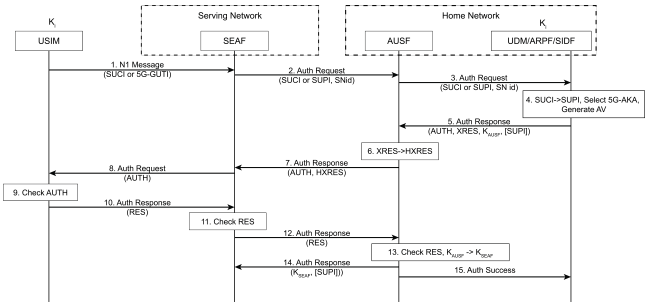


Figure 5: 5G AKA Message Flow [3]

(9) The UE validates the AUTH token with the secret key it shares with the home network. If this validation succeeds, from the UEs perspective, the authentication has succeeded.
(10) The UE sends a RES token to the SEAF.
(11) The SEAF validates the RES token.
(12) The SEAF sends the RES token to the AUSF.
(13) The AUSF validates the RES token. **The final decision about whether the procedure succeeded is made here, in the home network!** If the token is valid, a key K_{SEAF} is computed from the stored K_{AUSF}.
(14) The AUSF sends K_{SEAF} and SUPI to the SEAF. **Note how the SUPI is only sent here, after the procedure succeeded.**
(15) The UDM is informed of the outcome for logging.

Steps 7, 13 and 14 show how increased privacy and home control are achieved.

3.4 Key Hierarchy

Figure 6 shows the key hierarchy in 5G. After concluding the AKA message exchange, the SEAF derives the *anchor key* K_{AMF} and deletes K_{SEAF} immediately. The new key is then passed to the AMF, which is located together with the SEAF. The AMF can now derive all the other keys needed for access stratum (K_{gNB}) and non-access stratum (K_{NASint}, K_{NASenc}) security as well as a key for non-3GPP access (K_{N3IWF}). Since the root key K is shared between the home network and the UE, because it houses the USIM, onto which the key is hard-coded, the UE can derive any key in the hierarchy and thus possesses a complete set of keys [2].

4 SLICING

4.1 What is Slicing?

Network slicing is understood as the creation of independent logical networks on shared infrastructure by means of Software Defined Networking (SDN) and Network Function Virtualization (NFV). While the former separates the control

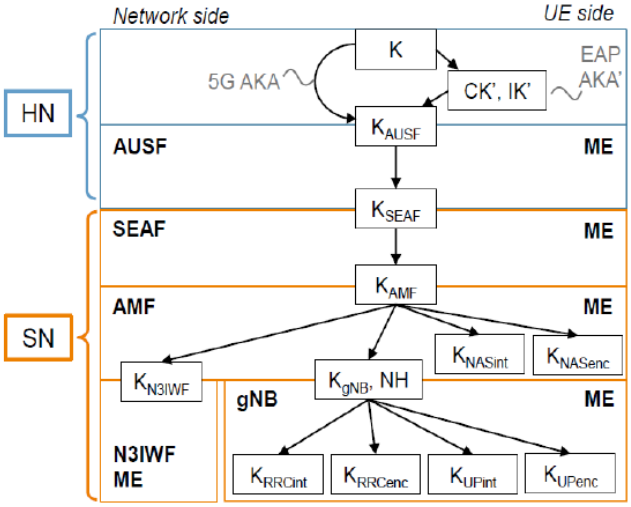


Figure 6: 5G Key Hierarchy [8]

from the forwarding plane and thus allows centralized administration of network resources via a protocol such as OpenFlow, the latter decouples network functions i.e. firewalls, from specialized hardware by emulating the corresponding functions on regular servers through software. Slicing is intended to do to 5G what IaaS providers such as Amazon EC2 or Microsoft Azure do to regular network and compute resources [9]. Naturally, this involves a broad spectrum of actors, from hardware manufacturers to mobile network operators. Protecting sensitive data sent over a network slice from third parties thus becomes an almost impossible feat, since the mobile network operator renting out a slice to a customer in any case will have the ultimate authority over infrastructure and access to at least metadata. In the following a couple of possible approaches to ensure varying degrees of isolation from [9] will be briefly outlined.

4.2 Slice Isolation and Security

4.2.1 Over-the-top Isolation. One possible means of protecting data sent through an untrusted network is Over-the-top (OTT) security, meaning isolation by through technologies such as a Virtual Private Network (VPN). Authentication can be performed in the user plane, which, however, implies a prior admission of the user trying to authenticate herself into the network slice without knowing whether she actually has the credentials needed for a secure connection. To avoid this problem, the control plane can be used [9]. This type of isolation requires two sets of credentials, namely one for OTT security and one for the mobile network, thus putting additional overhead on the entity renting a network slice from a mobile operator, as credentials have to be issued and

maintained in a database. OTT security offers confidentiality and integrity protection against the mobile operator, but comes with drawbacks attached: The mobile operator can still siphon off significant amounts of metadata such as a user's location, identity, connection time and duration, etc. Additionally, no resource isolation can be guaranteed, meaning that the tenant renting a slice can not know whether he is actually allocated the resources he is paying for or whether he might be sharing excess resources with another client of the mobile operator. This is relevant in a scenario where a tenant pays for a specific amount of resources to be available on short notice, i.e. in an emergency/ peak load scenario.

4.2.2 Private 5G Network. As the authors of [9] point out, full isolation from the operator without own infrastructure is not possible. A straightforward solution to this problem is to deploy a private network, including base stations, a 5G core and a data network, where the whole network and all devices are managed by the owner without participation of a mobile operator. This approach is feasible for huge industrial actors owning large sites or factories. In order to ensure wireless coverage over the area of the site, unlicensed spectrum would have to be used or frequencies would have to be leased from a mobile operator for geographically constrained on-site operations, since the spectrum on which mobile networks operate is heavily regulated and auctioned off to mobile operators for high prices. Such a solution would grant the highest degree of isolation, leaving only direct attacks on radio interfaces as well as deliberate backdoors as possible vectors for security breach. It comes, however, at the cost of significant overhead and is thus only suitable for organizations with the corresponding resources [9].

4.2.3 Private-Public Network. Figure 7 shows the schematics of a private/ public network mix which corresponds to a home/ visited network scenario in a regular public mobile network context. The tenant operates his own private 5G network as described above, but enables devices that leave the area covered by this private network to connect to the home network by means of roaming. A possible use case might be employees' mobile devices or vehicles, which spend time on-site as well as off-site but need constant access to some kind of Industry 4.0 data network. The improvements to 5G authentication described earlier allow for authentication between the home network's AUSF and the off-site device. Owing to the fact that the 5G security framework allows the establishment of several security contexts with only on authentication pass, the AUSF can use one key for OTT security between the home network 5G core and the device, while the visited network's AMF has another key for control plane communication with the roaming device. This approach makes the maintenance of an additional credentials database on the tenants side obsolete and hides metadata

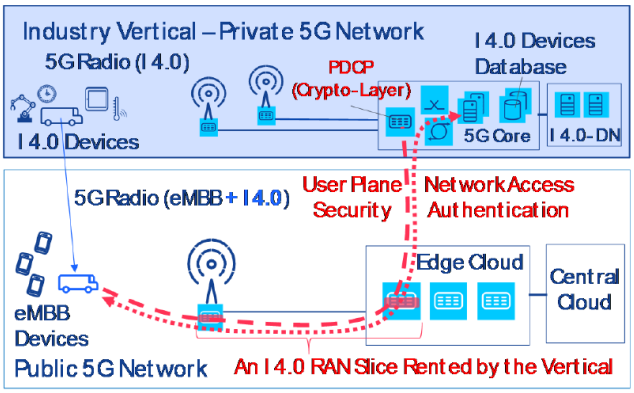


Figure 7: Private-Public Network Split [9]

from third parties, as long as the mobile equipment is kept on-site. In the roaming scenario, the metadata is still exposed to the mobile operator [9].

4.2.4 Private Network with Public RAN Slice. Another approach can be seen in Figure 8. Here, a tenant only rents a slice of a public RAN for off site operations, while still having a full private network on-site. We can see that more functions are delegated to the private network, as there is no mobile carrier operated AMF present anymore. This function, too, now is part of the private network's 5G core. Isolation is achieved by encrypting the data until it reaches the private network, thus preventing the mobile operator from listening. This is achieved by implementing the Packet Data Convergence Protocol (PDCP), which usually would be located at the gNBs CU (we recall that access stratum security is terminated there) in the private network, a solution allowed by the 5G standard [9]. The PDCP is responsible, among others, for data encryption and decryption. With such an approach, all credentials would be stored exclusively in the private network and therefore well protected. The private network, however, would have to manage mobility for its devices, as there is no mobile carrier operated AMF anymore. Additionally, devices would still have to somehow identify themselves to the RAN for it to decide which slice they should be attached to.

4.2.5 Gateway Core Network. The last approach, pictured in Figure 9, is suitable for organizations without a large enough site to warrant the establishment of a private 5G network, but with possibly a large network of critical IoT devices whose communications are to be isolated on an own slice, while ensuring connectivity to the Internet via eMBB. Here, RAN and AMF are provided by the mobile operator. The critical IoT slice is managed by a private 5G core, thus keeping sensitive data, including credentials and user plane keys, on private infrastructure. Another slice is responsible

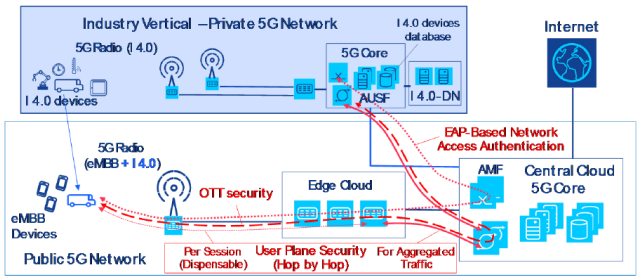


Figure 8: Private Network with Public RAN Slice [9]

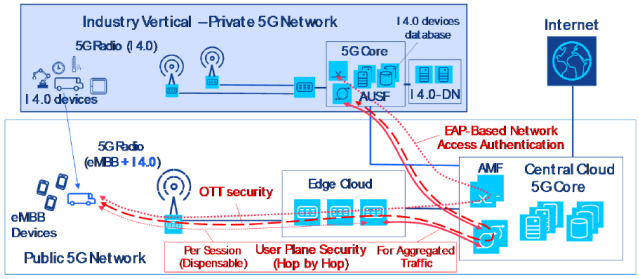


Figure 9: Gateway Core Network Approach [9]

for internet connectivity via eMBB and managed by a 5G gateway core operated by the mobile operator. This way critical communication is isolated against third parties, while the devices still have access to public parts of the network, such as Internet. As with the other approaches, the issue of metadata being visible to the operator persists.

4.2.6 Summary. In summary it is evident that flexibility in the form of greater coverage/ the possibility to roam outside of privately owned sites has to be bought at the expense of isolation. Conversely, an increase in privacy usually entails a significant increase in overhead, as more functions usually offered by a mobile network operator have to be taken care of. In all cases, except for the completely private network, metadata can be easily acquired by the mobile network operator.

5 CONCLUSION

5G offers several improvements over its predecessor 4G. The SBA allows for flexibility and can be leveraged to virtualize and decentralize the 5G core, while leaving open the possibility of adding new functionality. SUPI encryption and increased home network control achieved through the home network's AUSF having the final say in authentication significantly increase privacy and security, among others preventing man-in-the-middle attacks with false base stations. The new security framework enables participants without USIM powered devices to connect to the network as well as access through Wi-Fi/ cable through the newly introduced N3IWF. Seamless switching of access methods is enabled through the establishment of several security contexts with only one authentication pass. Slicing as a revenue model for mobile operators may proliferate, however, it comes with concerns regarding isolation of the respective tenants renting resources on the network.

REFERENCES

[1] 3GPP. Security architecture and procedures for 5G System. Technical Specification (TS) 33.501, 3rd Generation Partnership Project (3GPP), 03 2020. Version 15.8.0.

[2] 3GPP. System Architecture for the 5G System. Technical Specification (TS) 23.501, 3rd Generation Partnership Project (3GPP), 03 2020. Version 15.9.0.

[3] Cablelabs. A Comparative Introduction to 4G and 5G Authentication. https://www.cablelabs.com/insights/a-comparative-introduction-to-4g-and-5g-authentication. Accessed: 2020-06-04.

[4] J. Cao, M. Ma, H. Li, R. Ma, Y. Sun, P. Yu, and L. Xiong. A survey on security aspects for 3gpp 5g networks. *IEEE Communications Surveys Tutorials*, 22(1):170–195, 2020.

[5] Ericsson. Overview: Security architecture in 5G and LTE/4G systems. https://www.ericsson.com/en/blog/2019/7/3gpp-5g-security-overview. Accessed: 2020-06-04.

[6] Metaswitch. What is the 5G Service-Based Architecture (SBA)? https://www.metaswitch.com/knowledge-center/reference/what-is-the-5g-service-based-architecture-sba. Accessed: 2020-06-20.

[7] Anand R. Prasad, Sivabalan Arumugam, Sheeba B, and Alf Zugenmaier. 3GPP 5G Security. https://www.3gpp.org/news-events/1975-sec_5g. Accessed: 2020-06-04.

[8] Anand R. Prasad, Sivabalan Arumugam, Sheeba B, and Alf Zugenmaier. 3gpp 5g security. *Journal of ICT Standardization*, 6(1):137–158, 2018.

[9] P. Schneider, C. Mannweiler, and S. Kerboeuf. Providing strong 5g mobile network slice isolation for highly sensitive third-party services. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2018.

[10] IEEE Spectrum. 3GPP Release 15 Overview. https://spectrum.ieee.org/telecom/wireless/3gpp-release-15-overview. Accessed: 2020-06-20.

Data over Sound - Übertragung per Ultraschall

Seminar (Secure) Communication Networks

Florian Schmeller
schmeller@uni-potsdam.de

09. Juli 2020

Abstract. Daten über den Ultraschallbereich zu kommunizieren verspricht neben dem Auskommen von zusätzlicher Konfiguration auch energieeffiziente Datenübertragung in Geräten, die nichts weiter benötigen, als ein Mikrofon und ein Lautsprecher. Dennoch birgt es ebenfalls Gefahren für den Verbraucher, die Tür und Tor für Datenspionage von Firmen und Regierungen öffnen. Erste Gegenmaßnahmen bestehen in Aufklärung und Firewalls, die auf Ultraschallsignale hören.

1. Einführung

Als Schall versteht man im physikalischen Sinne die mechanische Schwingung in einem Medium. Dabei kann das Übertragungsmedium zum Beispiel Luft oder Wasser sein. Es existieren drei wesentliche Frequenzbereiche des Schalls: Infraschall, Hörschall oder Ultraschall [1]. Nur der Hörschall kann von dem Menschen wahrgenommen werden. Der Frequenzbereich des Hörschalls liegt zwischen 20 Hz und 20 kHz. Infraschall- und Ultraschallfrequenzen liegen unter bzw. über dem Frequenzbereich des Hörschall.

Ein Verfahren zur digitalen Kodierung von Frequenzen ist die Frequenzumtastung [1]. Dieses legt für ein Alphabet von M Symbolen genau M verschiedene Frequenzen in einem vorher festgelegten Frequenzband an. Dadurch entsteht eine duale Transformation zwischen Wörter eines Alphabets und Schallsignalen.

Als Kommunikationstechnologie ist die Übertragung von Daten über Frequenzbereiche außerhalb des Hörschalls eine Lösung für Problemstellungen der Datenübertragung zwischen mehreren Geräten. Diese Technologie bezeichnet man als *Data over Sound* Übertragung. Dafür benötigt es alleine ein Mikrofon auf dem einen Gerät und einen Lautsprecher auf dem anderen [2].

Ein Sektor der Informationsverarbeitung, welcher sich besonders über diese Technologie erfreuen kann, ist das Internet of Things (IoT). Oft schon sind diese Geräte mit der nötigen Hardware ausgerüstet und brauchen keine Erweiterungen, um Daten über

Schall zu senden. Zudem ist für das Starten einer Übertragung keine Konfiguration und Pairing notwendig, wie zum Beispiel bei Kommunikation über WLAN [2].

Wie in Abbildung 1 zu sehen ist, wird das prognostizierte Interesse am IoT stetig steigen. Genau für diesen Anwendungsfall ist es für die Geräte wichtig, dass sie in ihren Aufgaben energiesparend arbeiten können. Somit ist es außer Frage, dass die Kommunikation im IoT kosteneffizient sein muss und wenig Strom verbraucht. Die sogenannten 'wake-on-sound' Mikrophone erfüllen genau diese Anforderung [2], woran man sehen kann, dass Data over Sound Technologien hierfür attraktiv sind. Worauf die Hersteller dennoch achten müssen ist, wie hoch die Ansprüche des Kunden für die IoT Geräte sind. Algorithmen der digitalen Signalverarbeitung, wie zum Beispiel die schnelle Fourier-Transformation (englisch Fast Fourier-Transform, FFT), können ein Argument gegen die massenhafte Verbreitung der Data over Sound Technologien sein, wenn für die Industrie ersichtlich wird, dass der Preis für das Gerät nicht der erwarteten Performanz des Geräts entspricht.

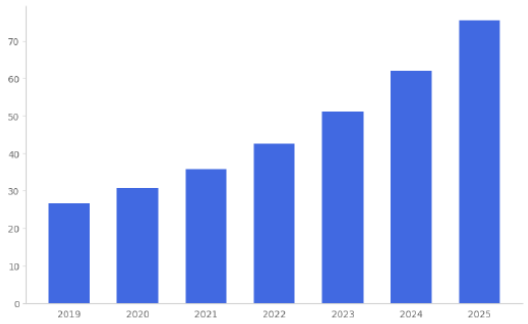


Abbildung 1: Anzahl an verbundenen IoT Geräten von 2019 bis 2025 (in Milliarden). Quelle [2]

Ein wesentliches Problem der Adaption von Data over Sound Technologien in Alltagsgeräten ist der Umstand, dass die Audiokomponenten (Mikrofon und Lautsprecher) nicht dafür entwickelt worden sind, Frequenzen zu senden, die nicht für den Menschen wahrnehmbar sind [2]. Aus diesem Grund hat sich man dafür entschieden, auf dem Frequenzbereich zwischen 18 und 20 kHz die Daten zu übertragen. Diese Entscheidung beruht auf der Tatsache, dass der Mensch im zunehmenden Alter Frequenzen von 20 kHz nicht mehr wahrnimmt. Ein weiterer Vorteil dieses Frequenzbandes ist, das Infraschall auf integrierter Hardware deutlich schwerer zu generieren und übertragen ist, als Ultraschall, da Ultraschall sehr kurze Wellenlängen besitzt [1].

Zusammen mit dem arm Befehlssatz hat die Firma Chirp eine C Programmbibliothek für die Cortex M Prozessorreihe entwickelt [2], welche darauf optimiert ist, Daten über Ultraschall zu senden und empfangen. Das SDK implementiert die Bitübertragungs- und Sicherungsschicht des OSI Modells und erlaubt das Stapeln weiterer Schichten, wie zum Beispiel Verschlüsselung oder Ende-zu-Ende Übertragung.

Die Cortex M Serie ist eine Lösung für kosteneffiziente sowie energiewirksame digitale Signalverarbeitung. Unter anderem umfasst der Befehlssatz Multiply-Accumulate und

SIMD Operationen. Gerade SIMD ist interessant für das Dekodieren von Signalen, da es erlaubt, auf mehreren Eingabedaten gleichzeitig zu arbeiten.

In Abbildung 2 ist ein typisches Data over Sound Paket zu erkennen, wie Chirp es generiert. Für die Kodierung eines Chirp Signals erfordet es die Generierung von Fehlerkorrektursymbolen und eine Sinusoszillatorsynthese [2].

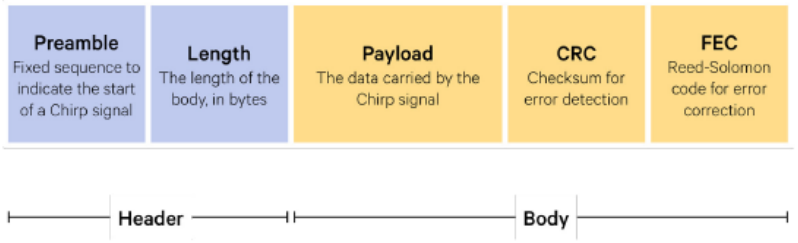


Abbildung 2: Ein typisches *data-over-sound* Paket. Source [2]

2. Datenschutzgefahren

Für die nachfolgenden Szenarien, in denen die Data over Sound Technologie Schwachstellen offenbart, welche von einem Angreifer einfach ausgenutzt werden können, wird sich auf ein Benutzer beschränkt, welcher auf seinem Mobilgerät eine Anwendung installiert hat, die einen Seitenkanal für Ultraschallsignale etabliert. Dadurch ist es für Dritte möglich, von einem Individuum ein umfangreiches Benutzerprofil erstellen.

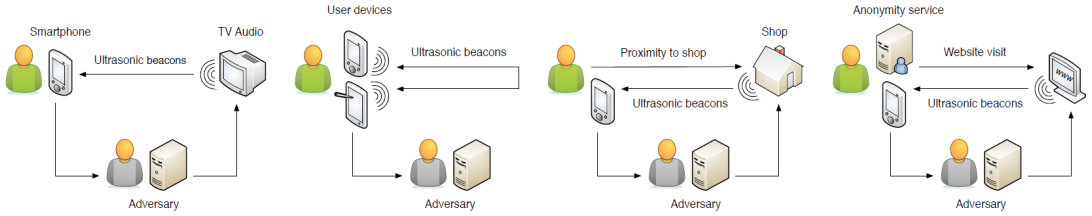
In Abbildung 3 sind vier Szenarien dargestellt, welche zeigen, wie ein Angreifer Informationen über ein Individuum durch Ultraschallseitenkanäle erlangen kann.

Die Medienüberwachung in Abbildung 3a zeigt, wie ein Angreifer (Adversary) Informationen über die Mediennutzung eines Individuum erhalten kann. Dafür schickt das Medium Ultraschallsignale an das Mobilgerät des Individuum, welches dafür durch einen Seitenkanal einer installierten Anwendung empfänglich ist. Diese Anwendung kann dann Daten an einen externen Dienst senden, welche zum Beispiel die Mediennutzung enthalten können.

Das auch Mobilgeräte untereinander mit Hilfe von Seitenkanälen kommunizieren können, ist in Abbildung 3b zu sehen und wird als Cross-Device-Tracking bezeichnet. Hierbei kann ein Angreifer erfahren, welche Geräte ein Individuum besitzt.

Durch Seitenkanäle kann auch ermittelt werden, wo sich ein Individuum genau befindet. Das Szenario wird in Abbildung 3c dargestellt. Dabei funktioniert ein Kaufgeschäft als Signalquelle, welche mit den Seitenkanälen der Mobilgeräte der Besucher kommuniziert. Der Angreifer erhält dann die Information von den Mobilgeräten.

Internetdienste, die sich hauptsächlich durch Pseudonymisierung profilieren, können so ebenfalls zu kurz kommen, wie in Abbildung 3d gezeigt. Dabei verschickt das Medium, mit denen die Dienste genutzt werden, die Ultraschallsignale an das Mobilgerät des Individuum und schlussendlich zum Angreifer, welcher dann den Benutzer entpseudonymisiert.



(a) Media Tracking (b) Cross-Device Tracking (c) Location Tracking (d) Deanonymization

Abbildung 3: Datenschutzgefahren durch Seitenkanäle. Quelle [1]

Wie verbreitet ultraschallempfängliche Seitenkanäle sind, hat die Fallstudie "Privacy Threats through Ultrasonic Side Channels"[1] von Daniel Arp et al. der TU Braunschweig dokumentiert. Sie haben untersucht, wie zuverlässig die Erkennung von Frequenzen zwischen 18 und 20 kHz ist und welche Anwendungen auf einen Android Mobilgerät einen Seitenkanal implementiert haben. Drei kommerzielle Implementierung der Ultraschallüberwachung sind SilverPush, Lisnr und Shopkick, welche in Anwendungen ermittelt werden sollten. Dafür bedienten sie sich Radare2 und Androguard, um entweder einen gegebenen Java bzw. nativen, durch das Android Native Development Kit entwickelten, Code zu dekompilieren und analysieren.

SilverPush ist spezialisiert auf Medien- und Cross-Device-Tracking [1]. Die Datenübertragung wird durch ein M-FSK im Frequenzbereich zwischen 18 und 20 kHz verschlüsselt. Das Alphabet besteht hierbei aus fünf Buchstaben des englischen Alphabets. Für die Fehlererkennung bedient sich SilverPush zwei Mechanismen: (1) Kein Buchstabe darf zweimal in einer Übertragung auftreten und (2) der Buchstabe A muss in jeder Übertragung vorhanden sein. Dekodiert wird ein Signal durch den Goertzel Algorithmus.

Lisnr und Shopkick sind in ihrem Kommunikationsprotokoll und ihrer Signalverarbeitung ähnlich aufgestellt und ihr Anwendungsgebiet ist die Ortsüberwachung [1]. Beide kodieren Frequenzen im Bereich zwischen 18.5 kHz und 19.5 kHz durch ein FSK. Ebenfalls implementieren beide für die Dekodierung ein FFT, jedoch implementiert Lisnr ebenfalls dazu den Goertzel Algorithmus.

Die eingangs formulierten Szenarien, in denen der Datenschutz durch einen Seitenkanal gefährdet ist, wurde von Arp et al. anhand einer empirischen Untersuchung genauer beleuchet. Dafür haben sie drei Experimente veranlasst: (1) Die Überprüfung der technischen Leistungsfähigkeit und Evaluierung der Limitierungen von Ultraschallsignalen unter realistischen Bedingungen (*kontrolliertes Experiment*); (2) die Analyse von Medien für Hinweise auf Ultraschallsignale (*Schallsignale im Alltag*); und (3) die Analyse von 1.3 Millionen Android-Anwendungen (*Anwendungen im Alltag*).

Kontrolliertes Experiment. Die Ausgangsfragestellung war, ob es denn überhaupt möglich sei, dass die eingebaute Hardware der Mobilgeräte die hohen Frequenzen zuverlässig erfassen kann, wenn ebenfalls Umgebungsgeräusche dazu kommen. Dafür wurden in alltäglichen Videodaten Ultraschallsignale im Frequenzbereich zwischen 18 und 20 kHz mit variierender Signallänge und Schallstärke eingebettet und durch einen Standardlautsprecher eines TV mit 60 dBA gesendet. Als Mobilgeräte wurden hier fünf Android-Geräte untersucht: das LG-P880, das Motorola Moto G 2, das Fairphone 1 und das Asus

Nexus 7a und 7b.

In Abbildung 4 präsentieren sie ihren Versuchsaufbau und ihre Ergebnisse. Wie in Abbildung 4b ersichtlich wird, haben alle fünf Mobilgeräte eine Erfolgsquote von mindestens 60% die Frequenzen zu erkennen. Was aber auch durch Abbildung 4c ersichtlich wird, ist dass die Geräte untereinander teilweise sehr abweichen, die Frequenzen alle zu erkennen. So kann das Nexus 7b Frequenzen von 20 kHz nur mit etwa 75% und das Fairphone Frequenzen von 18 kHz nur mit etwa 70% erkennen.

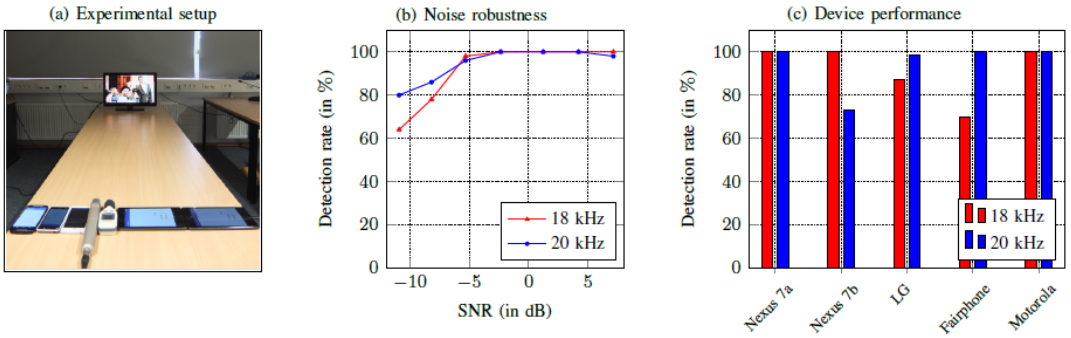


Abbildung 4: Ergebnisse des Geräteexperiments. Quelle [1]

Weiterhin haben sie ein Experiment veranlasst, in dem 20 Menschen zwischen 20 und 54 Jahren 10 Minuten Videos anschauen sollten, welche manchmal Schallsignale mit einer Frequenz von 18 kHz in verschiedenen Lautstärken zwischen 0 bis 18 dB enthielten. Sie kamen zu dem Ergebnis, dass niemand der Teilnehmer die Signale zuverlässig erkennen konnte.

Schallsignale im Alltag. Aufzeichnungen, die Ultraschallsignale eingebettet haben, welche auf Veranstaltungen abgespielt werden, können von Lisnr erkannt werden. Dies sind aber ausschließlich Veranstaltungen, an denen Lisnr ebenfalls teilnahm. Die Erkennung von der Verbreitung von Shopkick wurde in 35 Kaufgeschäften zweier europäischen Städten vorgenommen. Davon konnten in fünf Läden Ultraschallsignale erkannt werden. Was Shopkick hier von den passiven Seitenkanälen unterscheidet, ist die Tatsache, dass der Benutzer eigenwillig die Anwendung dafür starten muss. Somit sollte einem Anwender bewusst sein, dass zurzeit eine Datenübertragung stattfindet. Zur Untersuchung von der Verbreitung von SilverPush haben Arp et al. Audiodaten verschiedener Länder und TV Kanälen über das Internet gestreamt. Der Fokus war hier auf Kanälen, die viele Werbeanzeigen schalten. Sie konnten in den Daten jedoch keinen Hinweis auf Ultraschallfrequenzen vorfinden, aber schließen die Existenz dieser in den Audioquellen auch nicht aus, da durch das Streaming die Daten komprimiert empfangen worden sind. Abbildung 5 zeigt das Ursprungsland, die Anzahl der TV Kanäle und die Datengröße.

Country	# TV channels	Size
United States	7	25h
Germany	5	24h
Spain	6	23h
Austria	3	21h
United Kingdom	2	16h
Philippines	5	16h
India	10	15h

Abbildung 5: Datensatz der TV Streamanalyse. Quelle [1]

Anwendungen im Alltag. Der letzte Teil ihrer empirischen Untersuchung bildet das Ermitteln der Verbreitung von Lisnr-, Shopkick- und SilverPush-Implementierungen in Android-Anwendungen. Dafür haben Arp et al. 1,320,822 Anwendungen der VirusTotal-Datenbank der dritten Dezemberwoche 2015 analysiert und sind zu dem Schluss gekommen, dass zwei und ein Sample mit Lisnr- bzw. Shopkick-Funktionalität ausgerüstet waren und 39 weitere Sample Instanzen von SilverPush waren. Hier sei anzumerken, dass die Anwendungen mit Lisnr- bzw. Shopkick-Implementierung von den Firmen selber oder in Kooperation mit Drittfirmen entwickelt worden sind. In Abbildung 6 sind Drittanbieteranwendungen mit SilverPush-Funktionalität zu sehen, die aufgrund ihrer Anzahl an Downloads nicht unbeachtet bleiben sollten. Im Januar 2017 haben sie 234 Sample mit SilverPush-Funktionalität ermitteln können.

Application Name	Developer	Version	Downloads
100000+ SMS Messages	Moziberg	2.4	1,000,000 – 5,000,000
McDo Philippines	Golden Arches Dev. Corp.	1.4.27	100,000 – 500,000
Krispy Kreme Philippines	Mobext	1.9	100,000 – 500,000
Pinoy Henyo	Jayson Tamayo	4.0	1,000,000 – 5,000,000
Civil Service Reviewer Free	Jayson Tamayo	1.1	50,000 – 100,000

Abbildung 6: Drittanbieteranwendungen mit SilverPush-Funktionalität. Quelle [1]

3. Gegenmaßnahmen

Gegenmaßnahmen können auf der Seite des Empfängers als auch auf der Seite des Senders getroffen werden. Zudem gibt es Herausforderungen, die das Verwenden von Data over Sound Technologien deutlich schwerer machen.

Eine im vorherigen Abschnitt angerissene Herausforderungen ist die Bandbreitenbeschränkung. Streaming und Dateiformate wie MP3 komprimieren ihre Daten basierend auf der Annahme, dass der Mensch Frequenzen höher als 18 kHz oftmals gar nicht wahrnehmen kann [1]. Das hat zur Folge, dass die eingebetteten Ultraschallsignale entfernt werden und so ihre Verwendung ausbleibt.

Ebenfalls liegen Beschränkungen auf der Seite der Empfangsgeräte vor. So erlaubt das in Android 6 eingeführte Genehmigungsmodell das Setzen von potenziell gefährlichen Berechtigugen nur zur Laufzeit [1]. Das verschiedene Geräte auch teilweise stark in ihrer Zuverlässigkeit der Ultraschallfrequenzerkennung sich voneinander unterscheiden, spielt gleichfalls eine nicht zu unterschätzende Rolle. All dies sind Beschränkungen seitens der Hardware und Software, die einer Verbreitung von Data over Sound im Wege stehen.

Sollten Data over Sound Technologien die selbe Verbreitung wie zum Beispiel Bluetooth erfahren, besteht eine weitere Überlegung seitens der Betriebssystementwickler, ob man denn dem Nutzer benachrichtigt und über Statusinformationen mitteilt, wenn eine Ultraschallübertragung stattfindet.

Eine weitere Gegenmaßnahme bildet eine Anwendung auf dem Mobilgerät, welche erkennen kann, wenn eine Ultraschallübertragung angefangen wird und diese dann blockiert. Die SoniControl-Firewall [3] verspricht genau das. In ihrer Funktionsweise ist sie wie folgt aufgebaut: Zunächst wird von einer Eingabefrequenz das wahrnehmbare Spektrum entfernt und das Restspektrum normalisiert, um breitbandigem Rauschen entgegenzuwirken. Das Restspektrum (oder auch *Frame*) wird in einem zyklischen Buffer gespeichert und sobald dieser Buffer voll ist, wird von der Spektralfrequenzverteilung der Median berechnet, welcher Aufschluss über die grobe Verteilung der Hintergrundgeräuschkulisse ergibt. Es wird dann damit fortgefahren, die Kullback-Leibler-Divergenz D_{KL}, $0 \leq D_{KL} \leq 1$ zwischen der Spektralverteilung des Hintergrundmodells und der Spektralverteilung des eintreffenden Eingabeframes zu berechnen. Die Kullback-Leibler-Divergenz zwischen zwei Wahrscheinlichkeitsverteilungen gibt an, inwiefern sich diese voneinander unterscheiden. Wenn man D_{KL} nun berechnet hat, kann man noch einen Schwellwert t auf diese anwenden, um ein genaueres Bild davon zu bekommen, ob in den Umgebungsgeräuschen eine Veränderung vorliegt und gegebenenfalls dann entsprechend reagieren.

4. Fazit

Data over Sound Technologien sind Fluch und Segen zugleich. Zum einen bieten sie einen effizienten und ohne Konfiguration auskommenden Weg in Near-Field Communication (NFC) Umgebungen zu kommunizieren. Auf der anderen Seite können sie Firmen und Regierungen simpel in die Hände spielen, um entweder Datenspionage zu betreiben oder authoritär ihre Staatsbürger zu überwachen. In diesem Anwendungsgebiet versuchen erste Ansätze wie SoniControl die Entscheidungsmacht über ihre Daten den Benutzern zurückzugeben.

Literatur

[1] Daniel Arp, Erwin Quiring, Christian Wressnegger, and Konrad Rieck. Privacy threats through ultrasonic side channels on mobile devices. pages 35–47, 04 2017. In Proceedings of the 2nd IEEE European Symposium on Security and Privacy (EuroS&P).

[2] K. Marneweck, J. Nesfield, Dr. A. Mehrabi, and Dr. D. Jones. White paper: Why data-over-sound is an integral part of any iot engineer's toolbox: Chirp + arm = frictionless low power connectivity. Technical report, 2019.

[3] Matthias Zeppelzauer, Alexis Ringot, and Florian Taurer. Sonicontrol - a mobile ultrasonic firewall. *2018 ACM Multimedia Conference on Multimedia Conference - MM '18*, 2018.

(Secure) IPv6

Elias Pichottka
Universität Potsdam

8. Juli 2020

Zusammenfassung

In dieser Arbeit wird das Internet Protokoll in der Version 6 (IPv6) vorgestellt, unter Betrachtung von sicherheitsrelevanten Aspekten.

1 Einleitung

Diese Arbeit soll einen generellen Überblick zum Internet Protokoll in der Version 6 (IPv6) bieten und zusätzlich auf sicherheitsrelevante Erweiterungen und Sicherheitslücken eingehen. Im Abschnitt 2 wird die Nutzung des IPv6 durch das Internet Protokoll Version 4 (IPv4) Adressproblem motiviert und eine alternative Lösung, Network Adress Translation (NAT), aufgezeigt. Der Abschnitt 3 beschäftigt sich sich mit dem IPv6 Header und dessen mögliche Erweiterungen, durch Extension Header. Ein Vergleich zum Vorgänger Protokoll, das Internet Protokoll in der Version 4 (IPv4), wird im Abschnitt 4 behandelt. Auf die Protokoll Suite IP Security (IPsec) wird im Abschnitt 5 eingegangen. Im Abschnitt 6 wird das Internet Control Message Protocol (ICMPv6) beleuchtet. Der Abschnitt 7 geht auf zwei Angriffe ein, welche über das IPv6 ausgeführt werden können. Zum Ende der Arbeit, in Abschnitt 8, wird eine Client/Server Anwendung unter Verwendung des IPv6 vorgestellt, gefolgt von einem persönlichen Fazit, in Abschnitt 9.

2 Motivation

Die Anzahl der Haushalte mit einer Internetverbindung steigt stetig, dies lässt durch Abbildung 1 [1] leicht nachvollziehen. Abbildung 1 stellt die relative Anzahl an Haushalten dar, welche eine Internet Verbindung (helles Blau) haben und welche einen PC besitzen (dunkles Blau). Hierbei repräsentiert die Y-Achse den relativen Anteil der Haushalte in Prozent und die X-Achse das Jahr der Messung, von 2005 bis 2019. Die jetzige Welbevölkerung beträgt ungefähr $7.8 * 10^9$ Menschen [3] und oft gibt es mehrere internetfähige Geräte pro Haushalt und sogar pro Person. Wenn man nun die Größe des IPv4 Adressraums mit $2^{32} \approx 4.3 * 10^9$ Adressen betrachtet, wird es klar das es nicht genügend

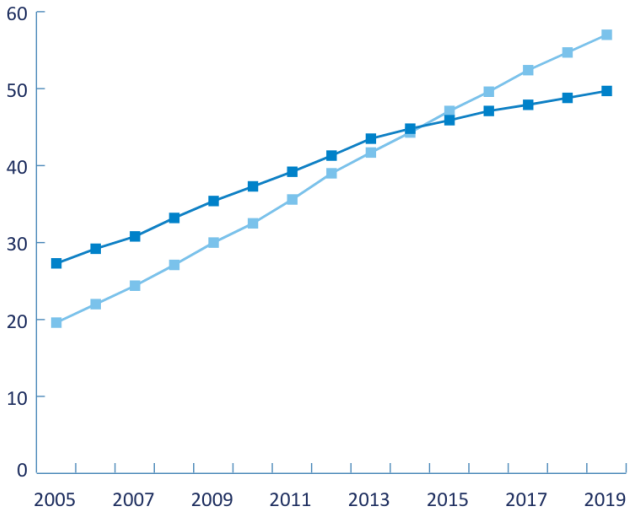


Abbildung 1: Haushalte mit Internetverbindung (helles Blau) und PC (dunkles Blau)

IPv4 Adressen für weltweit alle Geräte gibt, dieses Problem wird auch Adressproblem genannt. Ein Lösungsansatz für das Adressproblem ist die Network Adress Translation (NAT). Durch NAT werden weniger öffentliche beziehungsweise globale IPv4 Adressen genutzt, in dem mehrere Geräte in einem privaten Netzwerk, private unterschiedliche IP Adressen erhalten, aber insgesamt nur eine öffentlich IP Adresse zugeteilt wird. Dieses Konzept lässt sich auch zusätzlich auf die Ebene des Internet Service Providers (ISP) erweitern, in dem dieser ebenfalls ein privates Netzwerk für all seine Kunden führt. In Abbildung 2 ist diese mehrstufige NAT dargestellt, wobei die roten Boxen für NAT auf der Ebene des Kunden und die Blaue Box für NAT auf der Ebene des ISP steht. Hierbei ist es jedoch nachteilig, dass die einzelnen Router einen erhöhten Arbeitsaufwand durch die Übersetzung der Adressen haben, der Netzwerkaufbau komplizierter wird und in der Regel auch der Stromverbrauch ansteigt. Eine andere Alternative, zur Lösung des Adressproblems, die deutich besser skaliert ist IPv6. Durch die Verwendung von 128-bit Adressen ist das IPv6 in der Lage $2^{128} \approx 3.4 * 10^{38}$ Adressen darzustellen. In den letzten Jahren ist die Verwendung des IPv6 stark gestiegen, dies kann man Anhand der Nutzungsstatistiken für die Suchmaschine Google, in Abbildung 3 [6], nachvollziehen. Die X-Achse der Abbildung 3 steht hierbei für das Datum der Messung und die Y-Achse für den relativen Anteil der IPv6 Google Nutzung, im Vergleich zur gesamten Google Nutzung. Seit Anfang 2015 steigt die IPv6 Google Nutzung um circa 5% pro Jahr, heutzutage werden ungefähr ein Drittel aller Google Zugriffe über das IPv6 getätigt.

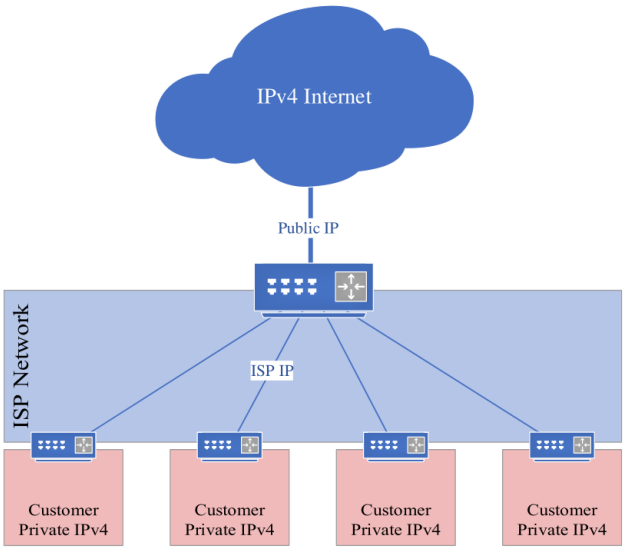


Abbildung 2: Network Adress Translation (NAT)

3 IPv6 Standard

Der IPv6 Standard wurde durch die Internet Engineering Task Force (IETF) erarbeitet. Die IETF ist eine internationale Vereinigung von Freiwilligen zur Standardisierung des Internets, welche in Arbeitsgruppen organisiert ist. Die Früchte der Arbeit der IETF sind technische Dokumente, sogenannte Requests for Comments (RFC's). RFC's werden in folgende Typen eingeteilt:

- Informational
- Experimental
- Standards Track
- Historic

In dieser Arbeit werden hauptsächlich RFC's die einen Internet Standard bilden (Typ: Standard Track) und vereinzelt auch überholte Standards (Typ: Historic) betrachtet. Das IPv6 wurde hauptsächlich durch die Arbeitsgruppe IP next generation (IPng) erarbeitet. Der erste Standard wurde im Jahre 1995 als RFC 1883 veröffentlicht, im Jahr 1998 durch RFC 2460 ersetzt, und später, 2017, durch den aktuellen Standard, RFC 8200.

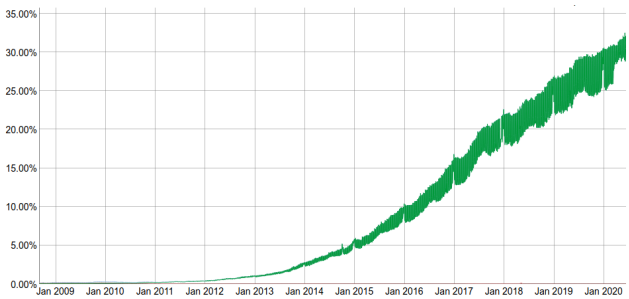


Abbildung 3: Google Nutzung über IPv6

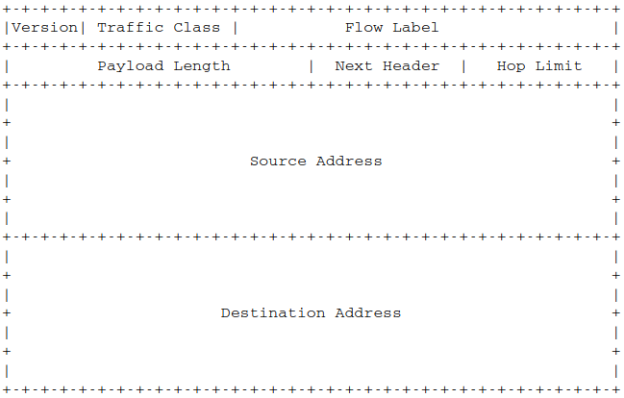


Abbildung 4: IPv6 Header nach RFC 8200

In Abbildung 4 ist der Aufbau des IPv6 Headers, nach RFC 8200 [8], sichtbar. Nachfolgend wird auf die Größe und Verwendung der einzelnen Header-Felder eingegangen.

Version (4-bit)	belegt durch Internet Protokoll Version, sprich 6
Traffic Class (8-bit)	wird für Packet-Priorität genutzt
Flow Label (20-bit)	kennzeichnet zusammenhängende Pakete
Payload Length (16-bit)	Länge der folgendend Daten inkl. Extension Header in Byte
Next Header (8-bit)	gibt Typ des nächsten Extension Headers vor
Hop Limit (8-bit)	gibt maximale Anzahl an Weiterleitungen an, wird an jedem Knoten dekrementiert
Source Adress (128-bit)	Adresse des Paket-Absenders
Destination Adress (128-bit)	Ziel Adresse für Paket

Wie bereits erwähnt wird im Next Header Feld der Typ des nächsten Extension Headers angegeben. Ein Extension Header ist hierbei eine Header Erweiterung die am Ende des vorherigen Headers angefügt werden kann und dessen Funktionaltität vergrößern kann.

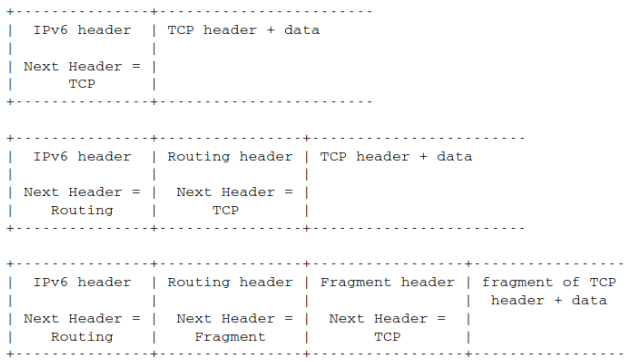


Abbildung 5: IPv6 Header Extensions nach RFC 8200

In Abbildung 5 [8] ist die Verkettung von IPv6 Header und Extension Header verdeutlicht. Im oberen Teil der Abbildung 5 ist ein minimales Beispiel ohne Extension Header aufgezeigt, hierbei enthält das Next Header Feld des IPv6 Header TCP, ausgeschrieben Transmission Control Protocol, dadurch wird signalisiert das ein TCP-Header sowie Nutzdaten folgen. In der Mitte von Abbildung 5 ist die Erweiterung des minimalen Beispiels durch einen Routing Header als Extension Header zu sehen. Im unteren Teil der Abbildung 5 wird das vorhergehende Beispiel noch zusätzlich durch einen Fragment Header erweitert und es handelt sich dementsprechend um ein Fragment/Teil von Nutzdaten das versendet wird.

4 Vergleich zum IPv4

Das Internet Protocol Version 4 (IPv4) besitzt eine komplexere Header Struktur als der IPv6 Header, da ein großer Teil optionaler Funktionaltiät auf Extension Header ausgelagert wurde. Der IPv4 Header, nach RFC 791 [12], ist in Abbildung 6 zu sehen. Im Folgenden werden IPv4 Header Felder die in den IPv6 Header:

- übernommen wurden durch '=',
- welche unter anderem Namen übernommen wurden durch 'r',
- und verschwundene Felder durch 'x'

gekennzeichnet. Falls es sich um eine Umbennung handelt wird der neue Name des IPv6 Feldes erwähnt, andernfalls wird Auskunft über das IPv4 Feld gegeben.

Version (4-bit) = nur andere Belegung

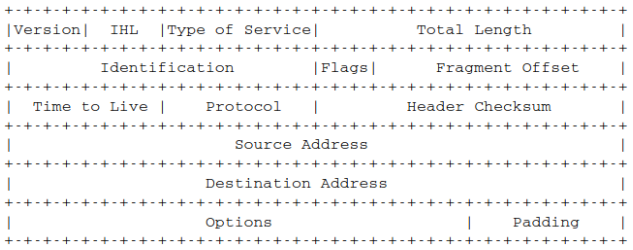


Abbildung 6: IPv4 Header nach RFC 791

IHL (4-bit)	x	Internet Header Length, obsolet durch IPv6 Header mit fester Größe
Type of Service (8-bit)	r	IPv6 Traffic Class
Total Length (16-bit)	r	IPv6 Payload Length
Identification (16-bit)	x	Nummerierung von Paketen
Flags (3-bit)	x	Information zu Fragmentierung
Fragment Offset (13-bit)	x	gibt Position des Fragments in Dateneinheit an
Time to Live (8-bit)	r	IPv6 Hop Limit
Protocol (8-bit)	r	gibt Protokoll für Datensektion an, IPv6 Next Header
Header Checksum (16-bit)	x	sichert Korrektheit des IPv4 Headers
Source Adress (32-bit)	=	lediglich 4-fache Größe
Destination Adress (32-bit)	=	lediglich 4-fache Größe
Options + Padding	x	(variabel) Routing, Debugging oder Statistische Information aufgefüllt mit Nullen

Neben den genannten Änderungen, ist auch die Schreibweise von IPv4 Adressen unterschiedlich zur Schreibweise von IPv6 Adressen. IPv4 Adressen sind 32-bit lang und können daher relativ kompakt durch 4 Dezimalzahlen dargestellt werden. Als Trennzeichen zwischen den 4 Dezimalzahlen werden jeweils einfache Punkte verwendet. So kann durch zum Beispiel: 127.0.0.1 die lokale IPv4 Loopback-Adresse beschreiben werden. Im Gegensatz dazu sind IPv6 Adressen 128-bit lang, dies führt zu einer im Regelfall längeren Schreibweise. Es hat sich etabliert eine Hexadezimaldarstellung aus 8 Zahlen zu verwenden, wobei das Trennzeichen ein Doppelpunkt ist. Um trotz größerer Datenmenge die Darstellung ohne Datenverlust weiter zu kürzen, ist es erlaubt führende Nullen und Blöcke von Nullen zusammenzufassen. Zusätzlich können die letzten 32 bits der Adresse IPv4 konform dargestellt werden. Diese syntaktische Vielfalt für semantisch gleiche Adressen macht es jedoch schwieriger Adressen zu vergleichen. Im nachfolgenden Beispiel ist eine IPv6 Adresse auf 4 verschiedene Arten niedergeschrieben:

- 0000 : 0000 : 0000 : 0000 : 0000 : $ffff$: 7f00 : 0001
- :: $ffff$: 7f00 : 0001
- :: $ffff$: 7f00 : 1
- :: $ffff$: 127.0.0.1

5 IPsec

Internet Protocol Security (IPsec) ist eine Protokoll Suite/Sammlung bestehend aus:

- Authentication Header (AH)
- Encapsulating Securtiy Payload (ESP)

IPsec wird zur gesicherten Kommunikation durch Verschlüsselung genutzt und findet zum Beispiel bei Virtual Private Networks (VPN's) Anwendung. Der jetzige IPsec Standard ist im RFC 4301 [11] verfasst und ersetzt damit RFC 2401. In Abbildung 7 ist der Authentication Header gezeigt, er wird genutzt um

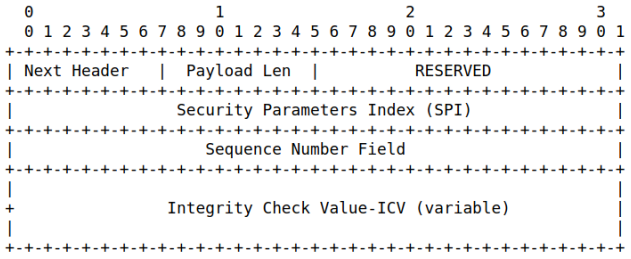


Abbildung 7: Authentication Header nach RFC 4301

die Quelle des Pakets (den Absender) zu authentifizieren beziehungsweise zu bestätigen. Durch den AH kann die Datenintergrität von Header und Payload gewährleistet werden. Um ein AH zu nutzen muss der Next Header Wert 51 verwendet werden. Im Folgenden sind die Felder des AH kurz beschrieben.

Next Header (8-bit)	Typ des nächsten Payloads
Payload Length (8-bit)	Länge des AH (in 4-byte Wörtern)
RESERVED (16-bit)	reserviert für Erweiterte Funktionalität
Security Parameter Index (32-bit)	Tag zu Unterscheidung verschiedener Datenströme
Sequence Number Field (32-bit)	Counter wird Paketweise Erhöht für eine Datenstrom
ty Check Value (multiple of 32-bit)	aus Header und Payload bestimmter Wert (Datenintegrität)

Im Gegensatz zum AH kann der Encapsulating Security Payload (ESP) nicht nur zur Authentifizierung eines Datenpakets genutzt werden, sondern auch zur Verschlüsselung eines Datenpakets. Abbildung 8 zeigt den Aufbau des ESP Hea-

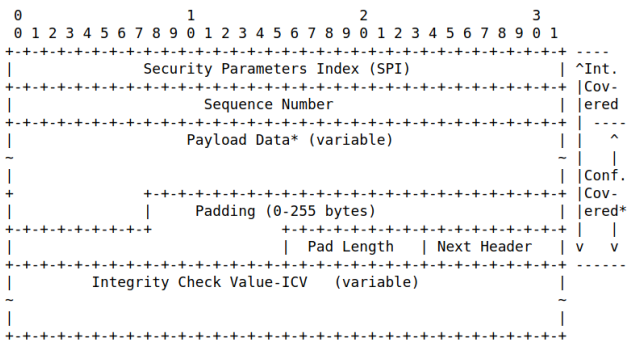


Abbildung 8: Encapsulating Security Payload Header nach RFC 4301

der. Um den ESP Header zu verwenden ist es notwendig, im vorherigen Header, den Next Header Wert 50 zu setzen. Im Folgenden werden die Header Felder des ESP kurz beschreiben:

Security Parameter Index (32-bit)	Tag zu Unterscheidung verschiedener Datenströme
Sequence Number (32-bit)	Counter wird Paketweise Erhöht für eine Datenstrom
Payload Data (multiple 1-byte)	variable Länge enthält verschlüsselte Daten
Padding (0-255 bytes)	sinnvoll für Verschlüsselungsalgorithmen die gewisse Datenlänge erwarten
Pad Length (1-byte)	Länge vom Padding
Next Header (8-bit)	Typ vom folgenden Extension Header
Integrity Check Value (variable)	berrechnet aus ESP Header und Payload

6 ICMPv6

Das Internet Control Message Protocol Version 6 (ICMPv6) wird für Fehler- und Statusmeldungen genutzt. Der aktuelle ICMPv6 Standard ist im RFC 4443 [13] gefasst und um das ICMPv6 zu nutzen ist der Next Header Wert 58 notwendig.

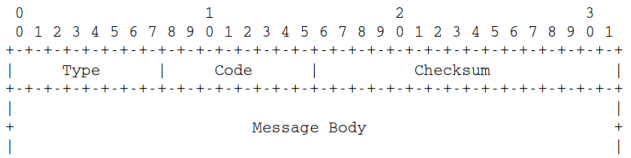


Abbildung 9: ICMPv6 Header nach RFC 4443

Das generelle ICMPv6 Header Format ist in Abbildung 9 gezeigt. Die einzelnen Felder des Headers werden im Folgenden kurz beschrieben:

Type der Typ der Nachricht

Code unterschiedliche Nutzung je nach Nachrichtentyp

Checksum benutzt zum Erkennen von Fehlern im Header sowie Message-Body

Message Body enthält Nachricht entsprechend Typ

Typ 1 bis 127 wird für Fehlermeldungen genutzt, die folgenden Typen sind im RFC 4443 definiert:

1 Destination Unreachable

2 Packet Too Big

3 Time Exceeded

4 Parameter Problem

100 Private experimentation

101 Private experimentation

127 Reserved for expansion of ICMPv6 error messages

Typ 128 bis 255 wird für Stausmeldungen genutzt, die folgenden Typen sind im RFC 4443 definiert:

128 Echo Request

129 Echo Reply

200 Private experimentation

201 Private experimentation

255 Reserved for expansion of ICMPv6 informational messages

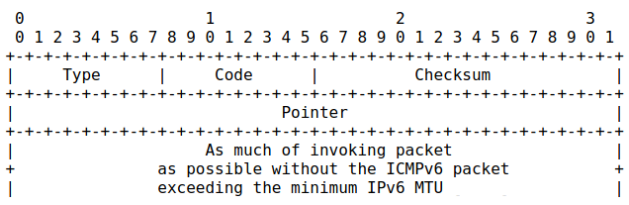


Abbildung 10: ICMPv6 Typ 4 Header

In Abbildung 10 ist der Header für eine ICMPv6 Meldung vom Typ 4 (Parameter Problem) gezeigt. Im Folgenden werden die für den ICMPv6 Typ 4 Header spezifischen Felder beschrieben:

Code 0 Error im Header

1 unbekannter next Header Typ

2 unbekannte IPv6 Option

Pointer zeigt auf Stelle an dem Fehler aufgetreten ist

Message Body möglichst viel des ursprünglichen Pakets ohne größer als MTU zu sein

Der Begriff MTU steht in der obigen Beschreibung für Maximal Transmission Unit (deutsch: maximale Übertragungseinheit).

7 IPv6 Angriffe

Dieser Abschnitt wird sich mit dem

- Routing Header Typ 0 Angriff und
- dem Smurf Amplifier Angriff

befassen. Der Routing Header von Typ 0 kann eine beliebige Anzahl an Wei-

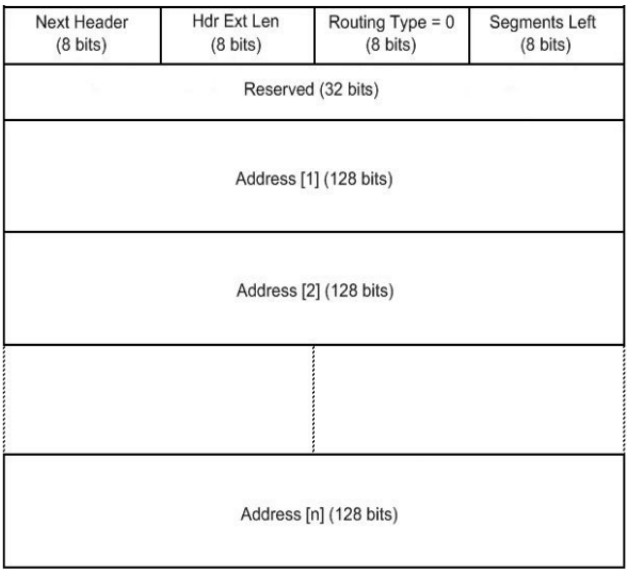


Abbildung 11: Routing Header von Typ 0

terleitungsadressen enthalten, der generelle Aufbau ist in Abbildung 11 [14] gezeigt. Um den Routing Header von Typ 0 zu nutzen muss der Next Header Wert 43 spezifiert sein. Da die Möglichkeit besteht Weiterleitungsadressen mehrfach anzugeben, kann man durch die Wiederholung von 2 Adressen erreichen das Pakete zwischen 2 Routern pendeln. Dies kann dazur führen das die Router

keine weiteren Pakete mehr verarbeiten können und somit in einem Denial of Service (DoS) enden. Eine Gegenmaßnahme für diesen Angriff bietet der Ingress Filter. Ein Ingressfilter blockiert bestimmte Pakete (e.g. Routing Header Typ 0 Pakete). Die Smurf Amplifier Attacke ist bereits durch das IPv4 bekannt. Bei dieser Attacke sendet der Angreifer modifizierte Pakete mit der Quelladresse des Opfers und als Ziel Adresse eine Multicast-Adresse, welche mehrere Zieladressen zusammenfasst. Im gefälschten Paket muss eine von den Zieladressen nicht unterstützte Option gewählt sein und die höchsten 2 bits des Options-Typ müssen 10 gesetzt sein. Erhalten die Empfänger nun die gefälschten Pakete, so sendet jeder Empfänger jeweils eine ICMPv6 Typ 4 (Parameter Problem) Fehlermeldung, da der Options-Typ nicht unterstützt wird. Das kann in einem DoS auf Seite des Opfers führen, wenn zu viele Fehlermeldungen ankommen. Das

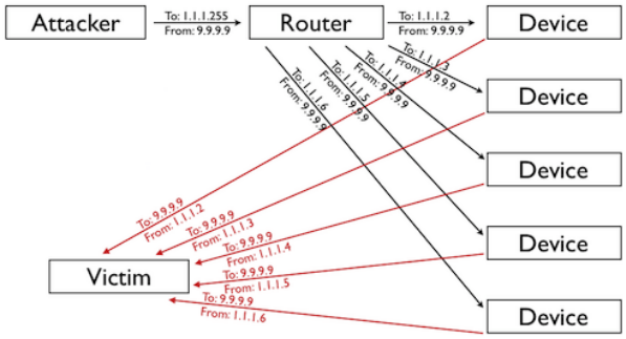


Abbildung 12: Visualisierung Smurf Amplifier Attacke

beschriebene Konzept des Smurf Amplifier Angriffs kann auch Anhand von Abbildung 12 [15] nachvollzogen werden, wobei die genutzten Adressen dem IPv4 entsprechen, das Prinzip ist jedoch analog.

8 Client Server Anwendung

In diesem Abschnitt wird eine minimale Client/Server Anwendung, in Python, unter Verwendung des socket Moduls beschrieben.

```
1  import socket
2
3  reciever_ip = "::1"
4  port = 1337
5  message = "Hello, World!"
6
7  client_socket = socket.socket(socket.AF_INET6,socket.SOCK_STREAM,0)
8  client_socket.connect((reciever_ip,port,0,0))
9  client_socket.send(message)
10
11 print "Client: \"{}\" has been sent to {}".format(message,reciever_ip)
```

Listing 1: client.py

Listing 1 zeigt hierbei den Quelltext der Client Anwendung. Das Client Programm eröffnet zuerst einen socket, verbindet sich dann mit dem Empfänger, dem Server und sendet im Anschluss eine 13 Byte lange Nachricht, “Hello World!”. Die Eingaben der genutzten Funktion sind im Folgenden aufgelistet:

socket(3) Internet Protokoll, Protokoll Klasse für Datenübertragung, Index Datenübertragungsprotokoll

connect(1) IP Adresse, Port, Flow Label, Scope Id

send(1) Daten die gesendet werden

```
1  import socket
2
3  server_ip = "::1"
4  port = 1337
5
6  server_socket = socket.socket(socket.AF_INET6,socket.SOCK_STREAM,0)
7  server_socket.bind((server_ip,port,0,0))
8  server_socket.listen(1)
9  connection, sender = server_socket.accept()
10 message = connection.recv(13)
11
12 print "Server: received \"{}\" from {}".format(message,sender[0],sender
       [1])
```

Listing 2: server.py

Listing 2 zeigt den Quelltext der Server Anwendung. Das Server Programm eröffnet ebenfalls ein socket, lauscht dann auf diesem beziehungsweise wartet auf eine Verbindung, wurde diese Verbindung hergestellt so nimmt der Server 13 Byte vom Client entgegen, was in diesem Beispiel der “Hello World!” Nachricht entspricht. Durch die Client Anwendung noch nicht verwendete Funktionen des Servers werden im Folgenden beschrieben:

bind(1) definiert Adresse, Port, Flow Label, Scope Id mit denen gelauscht werden soll

listen(1) max. Anzahl der Verbindungen

accept() stellt Verbindung mit Client her

recv(1) gibt an wieviel Bytes entgegenommen werden

Als sinnvolle Erweiterung der Server Anwendung könnte nun noch ein Dual-Stack Server implementiert werden. Ein Dual-Stack Server wartet in einer Schleife auf entweder IPv4 oder IPv6 Verbindungen und ist über IPv4 sowie IPv6 adressierbar.

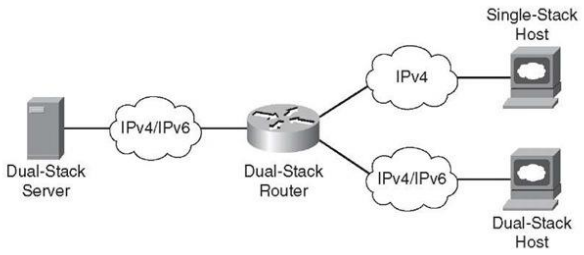


Abbildung 13: Visualisierung Dual-Stack Server

In Abbildung 13 [2] ist dieses Konzept veranschaulicht.

9 Fazit

Diese Arbeit sollte gezeigt haben, das IPv6 hat durch Extension Header viel zu bieten, auch hinsichtlich der Übertragungssicherheit. Weiterhin lässt es sich auch relativ bequem in Anwendungen nutzen, jedoch könnte der Umstieg, von IPv4 auf IPv6, schneller von statten gehen, wenn Bremsen wie zum Beispiel die NAT der ISP gelöst werden würden.

Literatur

[1] "Measuring digital development - itu," 2019.

[2] K. Mohbey, "Future internet plan using ipv6 protocol," *International Journal of Scientific and Engineering Research*, vol. 3, 2012.

[3] "Real time world statistics - worldometer," 2020.

[4] F. Seele, "Neue Möglichkeiten für Lastverteilung in Netzwerken," Master's thesis, University of Potsdam, 2015.

[5] C. Frieben, "IPv6 im Wandel - Analyse der IPv6-Standardisierung unter Sicherheitsaspekten," Master's thesis, University of Potsdam, 2014.

[6] "Ipv6 statistics - google," 2020.

[7] J. D. Sangam Racherla, *IPv6 Introduction and Configuration.* IBM Redbooks, 2012.

[8] R. H. S. Deering, "Internet protocol, version 6 (ipv6) specification," RFC 8200, RFC Editor, 2017.

[9] S. Kent, "Ip encapsulating security payload (esp)," RFC 4303, RFC Editor, 2005.

[10] S. Kent, "Ip authentication header," RFC 4302, RFC Editor, 2005.

[11] S. Kent, "Security architecture for the internet protocol," RFC 4301, RFC Editor, 2005.

[12] J. Postel, "Darpa internet program protocol specification," RFC 791, RFC Editor, 1981.

[13] S. D. A. Conta, "Internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification," RFC 4443, RFC Editor, 2006.

[14] "Cisco security threat and vulnerability intelligence," 2014.

[15] "Visualization for smurf amplifier attack - cloudflare," 2019.

Secure IPv6 with the IDSv6 Benchmark

Richard Hegewald
Seminar „Secure Communication Networks“
Universität Potsdam
Sommersemester 2020

1 Einleitung

Die Umstellung des Internets auf die Version 6 des Internet Protocols (IPv6) ist seit über 10 Jahren ein aktuelles Thema, in Forschung und Industrie. Die Notwendigkeit der Umstellung von IPv4 auf IPv6 ist gemeinhin bekannt und wird besonders dadurch verschärft, dass seit November 2019 keine freien IPv4-Adressen mehr in Europa existieren [9]. Google berichtet von einer deutschlandweiten IPv6 Adoption von ca. 48,8%, was deutlich über dem weltweiten Schnitt von ca. 32,2% liegt. [10] Trotzdem ist demnach klar, dass die IPv6 Adaption weiter zunehmen wird. Vor diesem Hintergrund stellt sich unter anderem die Frage, wie sich IPv6 Netze gegen Angreifer verteidigen lassen und wie effektiv die eingesetzten Verteidigungsmechanismen sind.

In dieser Arbeit soll ein IPv6 Benchmark für Intrusion Detection Systeme vorgestellt werden. Dafür werden kurz relevante IPv6 Thematiken eingeführt, Angriffsmöglichkeiten anhand von zweier konkreter Angriffstypen exemplarisch dargestellt, grundlegendes Wissen über Intrusion Detection Systeme aufgebaut bevor eine Ergänzung zum IDSv6 Benchmark besprochen wird.

2 IPv6 Basics

Viele der Neuerungen in IPv6 im Vergleich zu IPv4 finden sich im Header der Netzwerkpakete. So sind Felder wie *Fragmentation Offset, Identification* und alle weiteren optionalen Felder nicht mehr im IPv6 Header vorzufinden. Stattdessen wurden diese in *Extension Header* ausgelagert. Indem im *Next Header* Feld des IPv6 Headers ein weiterer Header referenziert wird, lässt sich eine Kette von Headern am Beginn eines Pakets aufbauen, welche Zusatzinformationen enthalten. Am Ende so einer Kette steht dann beispielsweise ein TCP Header mit den zugehörigen Daten, die versendet werden.

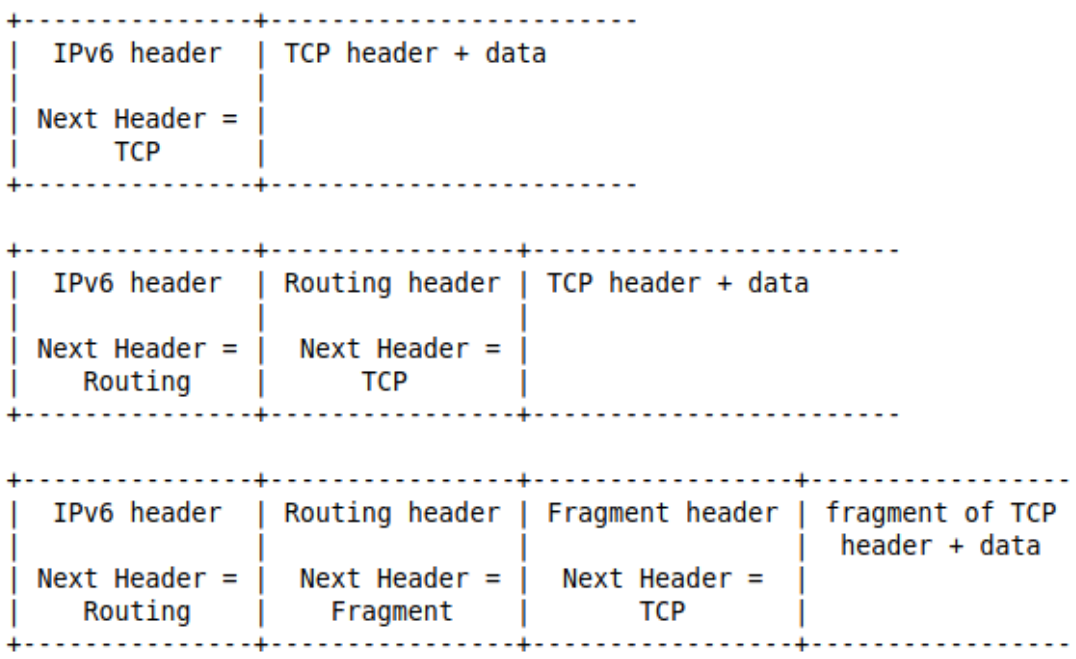


Abbildung 1: IPv6 Extension Header Structure [7]

3 How to attack an IPv6 network?

Um zu verstehen, wie ein IPv6 Netzwerk effektiv zu verteidigen ist, muss zuerst geklärt werden wie Angriffe in diesem funktionieren.´

3.1 The Hackers Choice IPv6 Attack Toolkit

Das 2005 von Mark Heuse veröffentlichte 'The Hackers Choice IPv6 Attack Toolkit' stellt eine Paketfactory-Bibliothek zur Verfügung, die dazu benutzt wird IPv6 Pakete zu erstellen, die so konfiguriert sind, dass gezielt Angriffe auf Ziele in IPv6 Netzen erzeugt werden.

3.2 Smurfing

Smurf-Angriffe basieren auf ICMPv6 Echo Requests. Laut RFC4890 ist vorgesehen, dass alle Systeme auf ein Echo Request mit einer Echo Response antworten: "[...] connectivity checks should be allowed by default." [8, S.26] Wird dieses Verhalten wie spezifiziert im System umgesetzt, können Angreifer Echo Requests durch Verwenden einer Multicast Adresse an alle Geräte im Netzwerk senden. Wird anstelle der eigenen Netzwerkadresse die Adresse des Angriffziels in den Paketheader eingetragen, werden alle Echo Responses an das Ziel bzw. den vermeintlichen Sender antworten und diesen somit mit einer Nachrichtenflut blockieren. Aufgrund dieser Angriffsmög-

lichkeit weichen aktuelle Betriebssysteme von der RFC Vorgabe ab und antworten standardmäßig nicht auf Echo Requests, was Smurf-Angriffe effektiv verhindert.

3.3 Fragmentationsangriffe

Einer der in Abbildung 1 sichtbaren Extension Header ist ein sogenannter Fragmentation Header. Um den Zweck dieses Header zu verstehen, müssen zuerst wichtige Begriffe geklärt werden. Jede Netzwerktechnologie hat eine Maximalgröße für Pakete, die sogenannte *Maximal Transmission Unit (MTU)*. In der IPv6 Spezifikation [7] ist dabei festgelegt, dass die MTU mindestens 1280 Bytes umfassen muss. Eine häufig gesehene MTU ist die im Ethernet verwendete Größe von 1500 Bytes. Wird im Netzwerk ein Paket versendet, das größer ist als die spezifizierte MTU des Netwerks, muss das Paket in kleinere Pakete zerlegt werden. Man spricht dabei von *Fragmentation*. In IPv4 Netzen kann das Fragmentieren der Pakete vom Router übernommen werden, nachdem ein zu großes Paket empfangen wurde. Dies wurde aus Sicherheitsgründen in IPv6 Netzen dahingehend verändert, dass ein Router bei Empfang eines Pakets, das größer ist als die MTU, eine "Paket Too Big"Nachricht an den Sender zurückschickt und das Paket ablehnt. Der Sender muss dann nach Empfang der Nachricht das Paket selbst zerstückeln und die Paketfragmente nochmals versenden. Die Paketfragmente enthalten dann zusätzlich zum IPv6 Header einen Fragmentation Header, der die ID des Fragments und die Position (Offset) des Fragments enthält. Mithilfe dieser Informationen können die einzelnen Fragmente durch den Empfänger wieder zusammengesetzt werden. Dieser Mechanismus erzeugt einige Angriffsvektoren, welche auch im THC IPv6 Toolkit in Form von Fragmentationsangriffen ausgenutzt werden. Der grundlegende Aufbau und Zweck dieser Angriffe weicht nicht von den aus IPv4 bekannten Fragmentationsangriffen ab. In diesen werden beispielsweise Fragmente von einem Angreifer versendet, welche nicht zusammensetzbar sind. Marc Heuse berichtete auf der HITB 2012 von einem Angriff, bei dem viele unvollständige Fragmentströme an eine Firewall gesendet werden, um vorzugeben, dass sehr große Datenmengen gesendet werden. Um die vermeintlichen Fragmente zu verarbeiten wurden in der Firewall große Mengen an Speicher reserviert, was dazu führte, dass keine Pakete mehr analysiert werden konnten. [6, 24:53min]

4 How to defend an IPv6 network?

Wie in Punkt 3.3 aufgeführt, bietet IPv6 einige Angriffsvektoren, welche sich nicht ohne eine fundamentale Umstrukturierung des Protokolls vermeiden lassen. In Folge dessen setzen Verteidigungsstrategien unter anderem an Anomalieerkennung in Netzen an. Tools welche Netzwerkpakete analysieren, um Angriffe zu erkennen, nennen sich *Intrusion Detection Systeme (IDS)*. Diese lassen sich in zwei weitere Arten einteilen, hostbasierte IDS und netzwerkbasierte IDS. Hostbasierte IDS überwachen dabei lediglich einen einzigen Rechner, wohingegen netzbasierte IDS alle Pakete analysieren, die im Netzwerk versendet werden. Letzteres ist die Art von IDS, welche im Folgenden

vorgestellt werden.

4.1 Snort 2 Architecture

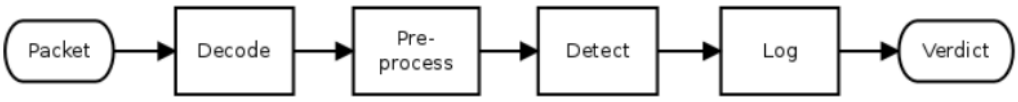


Abbildung 2: Snort 2 Architecture [3]

Eines der bekanntesten Intrusion Detection Systeme ist Snort. Snort wird von Cisco entwickelt und ist momentan in der Version 3.0 Beta verfügbar. Da der Aufbau des Programms jedoch historisch geprägt ist, lohnt es sich zuerst zu umreißen wie die Vorgängerversion Snort 2 funktioniert, bevor auf die Neuerungen in Snort 3 eingegangen wird.

Die Verarbeitung eines ankommenden Pakets ist in Abbildung 2 dargestellt. Das Paket durchläuft demnach 4 Phasen. [3, 2]

1. In der Dekodierungsphase wird das empfangene Paket 'zerlegt' und beispielsweise Informationen im Protokoll-Header, wie Sender und Empfängeradresse analysiert.

2. In der Präprozessorphase wird versucht, Inhalt und Zweck des Pakets zu erhalten. Dafür werden IP Fragmente zusammengesetzt und Kontext zu den einzelnen Paketen gesammelt. Die Daten werden dann in *Protocol Data Units (PDU)* zusammengefasst.

3. Aus Effizienzgründen werden in der Detectionphase alle Regeln mit identischen Signaturen zusammengefasst. Diese Gruppen an Regeln werden dann auf PDUs angewandt.

4. Die Loggingphase umfasst das Abspeichern (loggen) der analysierten Informationen und ggf. das Blockieren analysierter Pakete.

Die in Punkt 3. erwähnten Regeln folgen dabei dem Muster:
`action protocol source direction destination (body)`
Konkret könnte eine Regel so aussehen:
`alert tcp any any -> 192.168.1.1 80 (msg:"A ha!"; content:"attack"; sid:1;)`
Diese Regel erzeugt einen *alert* (action), wenn über das TCP Protokoll (protocol) von irgendeiner Adresse über irgendeinen Port (source) an die Adresse 192.168.1.1:80 (direction, destination) ein Paket gesendet wird. Die von Snort erzeugte Nachricht ist im body festgelegt und wäre in diesem Beispiel 'A ha!'.

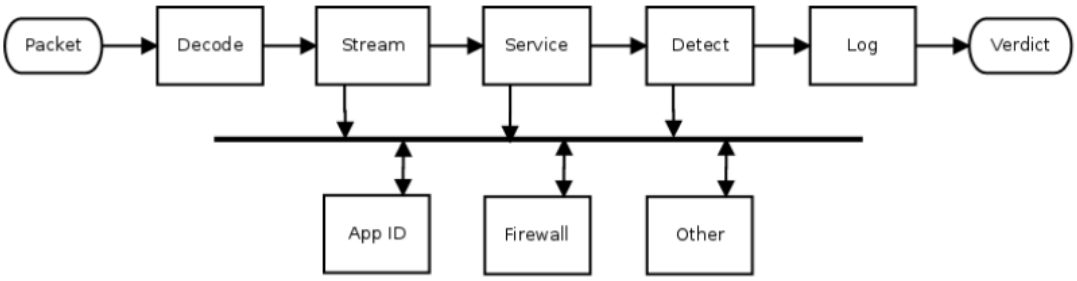


Abbildung 3: Snort 3 Architecture [3]

4.2 Snort 3 Architecture

Die 2013 angekündigte Version 3 von Snort ist nicht nur ein Update, sondern eine komplette Neuimplementation in C++. Unter anderem wurde die Paketverarbeitung derart verändert, dass die Präprozessorphase weiter aufgetrennt wurde (Abbildung 3), um Snort einfacher und effektiver erweitern zu können. Service und Netzwerkinspektoren übernehmen in Tandem nun das Sammeln von Kontext zu Paketen. Das Zusammensetzen von IP-Fragmenten wurde in eine eigenständige Komponente ausgelagert, den Streamprozessoren. Des Weiteren wurde Unterstützung für mehrere *Packet Processing Threads* hinzugefügt. [2] Diese ist besonders relevant, da die erste Version von Snort zu einer Zeit erschien, zu der noch keine kommerziellen Multicoreprozessoren verfügbar waren. Ausnutzung von Hardwareressourcen die Parallelität ermöglichen, ist jedoch oft ein wesentlicher Schritt zur besseren Skalierbarkeit von Applikationen.

5 How effective are my defenses?

Administratoren welche ein IDS im Netzwerk aktivieren wollen, stehen vor der Frage welches der verfügbaren Optionen vermeintlich die meisten Angriffe registriert. Der Vergleich mehrerer IDS ist jedoch problematisch, da Tests in einem Live-Netz abhängig von Netzwerk-Setup, Software Version etc. sind und deswegen keine vergleichbaren Resultate erzeugen. Der IDSv6 Benchmark [1] setzt an dieser Problemstellung an. Dafür wurden in einem Test-Netzwerk Angriffe des THC IPv6 Attack Toolkits und eines weiteren Toolkits ausgeführt und per `tcpdump` aufgezeichnet. Die entstandenen `.pcap` Dateien lassen sich von allen gängigen IDS analysieren und die erzeugten Logdateien durch das stabile Testbed (Abbildung 4) vergleichen.

Der ursprüngliche Benchmark vergleicht dabei 4 unterschiedliche IDS. [1]

1. Snort, welches in Punkt 4.2 vorgestellt wurde
2. ein von Max Schrötter (Universität Potsdam, 2018) im Rahmen seiner Bachelorarbeit entwickeltes IPv6 Plugin für Snort 3

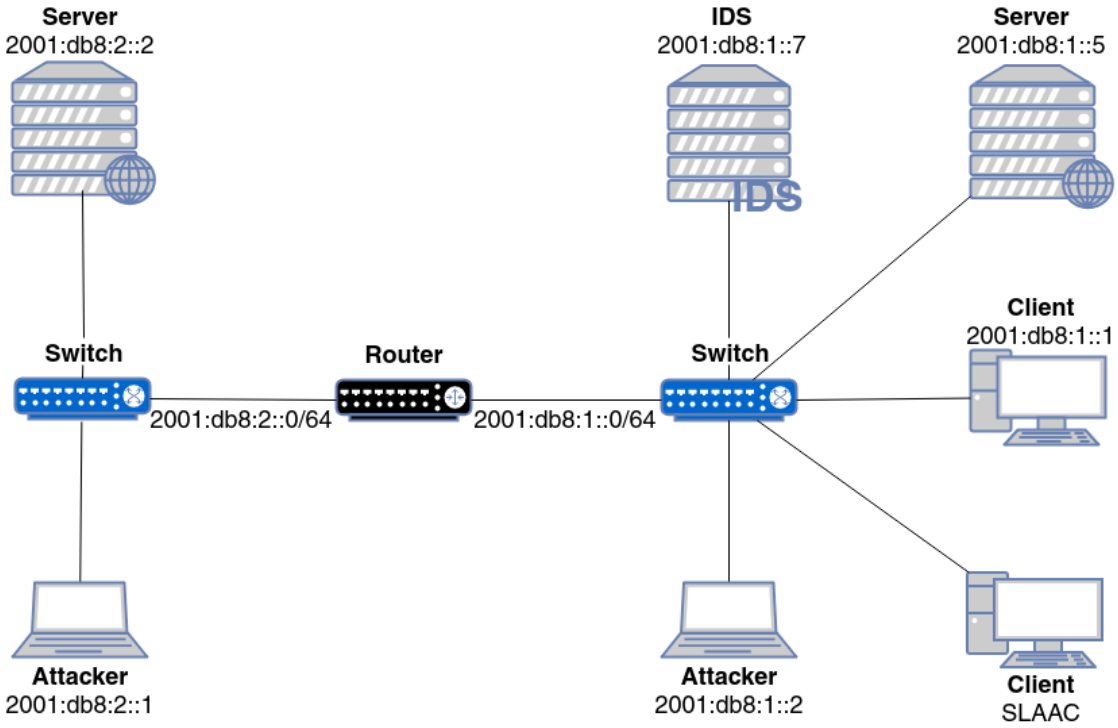


Abbildung 4: Test-Setup im IDSv6 Benchmark [1]

3. Zeek, das ursprünglich von Vern Paxson in Berkeley entwickelt wurde und einen Fokus darauf legt Flexibilität durch Trennung von Dekodierung und Paketanalyse zu erreichen

4. Suricata, das Neuste der vorgestellten IDS, welches von der Open Internet Security Foundation (OISF) entwickelt wird und einen starken Fokus auf Nutzung von Multithreading legt

5.1 Benchmark Results

Die originalen Benchmarkergebnisse sind unter Abbildung 6 zu finden. Es wird deutlich, dass Snort mit dem IPv6 Plugin mit 47 Angriffen die höchste Erkennnungsrate hat, gefolgt von der Snort 3 Beta (ohne Plugin) mit 33 erkannten Angriffen. Suricata landet mit 29 Angriffen auf Platz 3. Die geringste Erkennungsrate ist bei Zeek erkennbar, mit 22 erkannten Angriffen. [1]

5.2 Suricata Architecture

Suricata als jüngster Vertreter der vorgestellten IDS unterstützt mehrere Runmodi, welche den Grad der Parallelität der Analyse bestimmen. Prinzipiell unterscheidet Su-

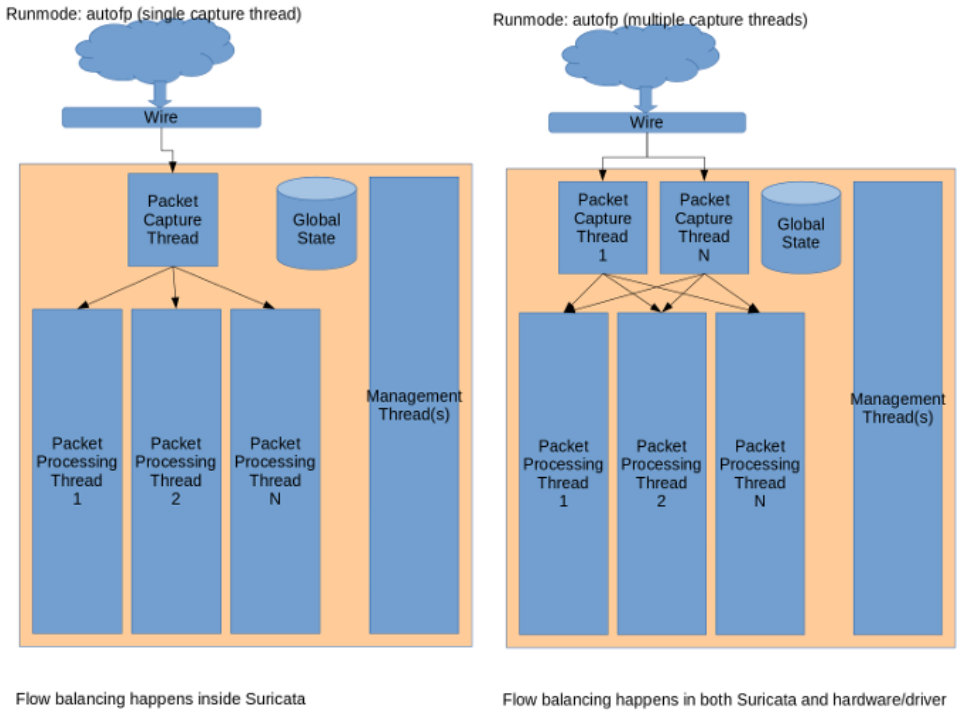


Abbildung 5: Multithreading in Suricata [4]

ricata zwischen *Packet Capture Threads*, welche das Einsammeln der Pakete vom Netzwerkinterface übernehmen und *Packet Processing Threads* die wiederum für die eigentliche Paketanalyse zuständig sind. Der Benchmark wurde im Standardmodus `autofp` ausgeführt. Dieser unterstützt sowohl mehrere Capture Threads als auch mehrere Processing Threads, wie in Abbildung 5 dargestellt. Es existiert zusätzlich der Modus `single`, welcher die Lastverteilung der NIC überlässt und lediglich einen Processing Thread startet und hauptsächlich für Debugging verwendet wird. [4]

5.3 Fazit

Der Benchmark wurde auf einer virtuellen Maschine (Oracle VMBox) mit Ubuntu 20.04, Suricata in der Version 5.0.3 und dem Regelsatz den `suricata-update` zur Verfügung stellt (ca. 20.000 Regeln) durchgeführt. Beim Einlesen der `.pcap` Dateien wurde anhand der von Suricata erstellten Alerts (oder der Abwesenheit dieser) analysiert, ob ein Angriff erkannt wurde oder nicht. Dabei wurde festgestellt, dass der Suricata Output für Amateure teilweise sehr kryptisch und nur schwer zu deuten ist, was die Einarbeitung in Suricata deutlich zeitintensiver gestaltet. Die Ergebnisse des Benchmarks für Suricata 3.2.1 [1] und Suricata 5.0.3 sind in Tabelle 1 dargestellt. Die Benchmarkergebnisse von Suricata 5.0.3 weichen dabei nur an einer Stelle von den originalen Ergebnissen ab. Der rsmurf6 Angriff welcher ursprünglich erkannt wurde, wurde in der neusten Version nicht mehr erkannt. Dies lässt sich durch den bereits in

3.2 erwähnten Umstand erklären, dass Echo Responses in allen gängigen Betriebssystemen nicht mehr standardmäßig aktiviert sind. Es ergibt demnach Sinn, Regeln zur Erkennung von Smurf-Angriffen nicht standardmäßig zu aktivieren, da diese nur in seltenen Fällen auftreten und zusätzliche Rechenressourcen beanspruchen. Das wird durch einen Versuch der Arbeitsgruppe von Prof. Schnor bestätigt, in dem durch das Aktivieren standardmäßig deaktivierter Regeln der rsmurf6 Angriff wieder erkannt wird, bestätigt.

Des Weiteren wurden Verbesserungen in folgenden Punkten beobachtet:

> „ For the parasite6 attack, Suricata warns "HTTP Host header invalid" for a correct HTTP request [...] “

[1]

> „ In almost all attacks it also falsely detects ’zero length padN option’ which is a valid padding option[...] “

[1]

Beim erneuten Benchmark wurde weder die Warnung ’HTTP Host hear invalid’ erzeugt, noch die Padding Option als Anomalie erkannt. Die in [1] beschriebene sehr hohe Anzahl an false positives konnte ebenfalls nicht beobachtet werden.

6 Zusammenfassung

Die Umstellung von IPv4 auf IPv6 bleibt weiterhin ein hochrelevantes Thema. Besonders Aspekte der Netzwerksicherheit müssen dabei betrachtet werden. IPv6 bringt wichtige Neuerungen, wie das Auslagern nicht standardmäßig verwendeter Headeroptionen in Extension Header, welche flexibel in Paketen konfiguriert werden können. Einige dieser Änderungen sind besonders sicherheitskritisch, wie die im RFC4890 festgelegte Vorgabe, dass Rechner auf ICMPv6 Echo Requests mit einer Echo Response antworten sollen. Dieses Verhalten erlaubt jedoch Smurfing-Angriffe, weswegen moderne Betriebssysteme von dieser Vorgabe abweichen. Fragmentation wurde in IPv6 dahingehend verbessert, dass Router, Pakete die größer sind als die Maximal Transmission Unit (MTU) des Netzwerks nicht mehr fragmentieren, sondern eine "Paket Too Big"Nachricht an den Sender zurücksenden der das Paket dann selbst zerstückeln muss. Viele dieser Angriffe sind im THC IPv6 Attack Toolkit, einer Paketfactory-Bibliothek verfügbar.

Zur Verteidigung von (IPv6) Netzen bieten netzwerkbasierte Intrusion Detection Systeme (IDS) wie Snort oder Suricata, eine Möglichkeit der Angriffserkennung, indem sie Pakete im Netzwerk sammeln und gegen vordefinierte Regeln prüfen. Mit Hilfe des IDSv6 Benchmarks kann die Effektivität verschiedener IDS in der Erkennungsrate von IPv6 spezifischen Angriffen vergleichbar gemacht werden. Dafür wurden in einem Testnetzwerk mit dem THC IPv6 Toolkit durchgeführte Angriffe so aufgezeichnet, dass sie von gängigen IDS eingelesen werden können. Im Vergleich zwischen Snort 3, einem IPv6 Plugin für Snort 3, Zeek und Suricata zeigt Snort 3 mit

Attack	Suricata v3.2.1	Suricata v5.0.3
covert_send6	✓	✓
denial-1,2,3,4,7	✓	✓
denial6-5,6	✗	✗
dos_mld_chiron	✗	✗
dos_new_ipv6	✗	✗
fake_mldrouter6_advertise, terminate	✗	✗
flood_advertise6, mld26, mld6, router26, rs6, solicitate6	✗	✗
frag-1,2,3,4,6,7,9,10,11,12,15, 16,17,18,25,26,27,28,29 ,31,32,33	✓	✓
frag-5,8,13,14,19,20,21, 22,23,24,30,34,35,36	✗	✗
kill_router6	✗	✗
ndpexhaust6	✗	✗
parasite_fake, real	✗	✗
redir6	✗	✗
rsmurf6	✓	✗
smurf6	✗	✗
toobig6	✗	✗

Tabelle 1: IDSv6 Benchmark Results for Suricata 3.2.1 and 5.0.3

IPv6 Plugin die höchste Erkennungsrate, wohingegen Zeek die Geringste zeigt. Suricata erkannte in der Version 3.2.1 29 der 62 Angriffe im Benchmark, in der Version 5.0.3 wurden nur 28 Angriffe erkannt. Bei hinzuschalten deaktivierter Regeln wurden jedoch wieder alle 29 Regeln erkannt. In der neuen Version hat sich außerdem die Rate der false-positives verbessert, jedoch nicht die Lesbarkeit der Logdateien.

Der Betrieb eines IDS stellt somit zwar einen nicht zu vernachlässigenden Zeitaufwand an den Administrator, kann aber dazu beitragen Angriffe im Netzwerk zu erkennen und zu verhindern.

Literatur

[1] Schrötter, M., Scheffler, T., Schnor, B. *Evaluation of Intrusion Detection Systems in IPV6 Networks.* 16th International Conference on Security and Cryptography, 2019.

[2] Schrötter, M *Entwurf und Implementierung einer IPv6 Erweiterung für Snort 3.0* Bachelorarbeit an der Universität Potsdam, 2018.

[3] Snort 3 User Manual. available from: *http://manual-snort-org.s3-website-us-east-1.amazonaws.com/*

[4] Suricata Documentation. available from: *https://suricata.readthedocs.io/*

[5] Heuse, M., The Hackers Choice *THC-IPV6 Attack Toolkit.* 2005

[6] Heuse, M. *IPv6 Insecurity Revolutions.* available from : *https://youtu.be/3cLIVygS_Fc* Vortrag auf den Hashdays, 2012.

[7] Internet Engineering Task Force. *Internet Protocol, Version 6 Specification (RFC 8200).* 2017

[8] Internet Engineering Task Force. *Recommendations for Filtering ICMPv6 Messages in Firewalls (RFC 4890).* 2007

[9] heise online. *Das war's mit IPv4-Adressen in Europa.* available from: *https://www.heise.de/newsticker/meldung/Das-war-s-mit-IPv4-Adressen-in-Europa-4596197.html* 2019

[10] Google IPv6 Statistics. available from: *https://www.google.com/intl/en/ipv6/statistics.html*

Anhang

Attack	Snort3-beta	Snort3-ipv6-suite	Zeek	Suricata
covert_send6	✓	✓	✗	✓
denial6-1, 2, 3, 4, 7	✓	✓	✗	✓
denial6-5	✗	✗	✗	✗
denial6-6	✓	✓	✗	✗
dos_mld_chiron	✗	✓	✗	✗
dos_new_ipv6	✗	✓	✗	✗
fake_mldrouter6_advertise	✗	✓	✗	✗
fake_mldrouter6_terminate	✗	✗	✗	✗
flood_advertise6, mld26, mld6, router26, rs6, solicitate6	✗	✓	✗	✗
frag-1, 2, 3, 4, 15, 16, 17, 18, 26, 28, 29, 31, 32, 33	✓	✓	✓	✓
frag-5, 19, 20, 21, 34	✗	✗	✗	✗
frag-6, 7	✗	✗	✓	✓
frag-8, 13, 14, 30, 35, 36	✗	✗	✓	✗
frag-9, 10, 11, 12, 27	✓	✓	✗	✓
frag-22, 23, 24	✓	✓	✗	✗
frag-25	✓	✓	✗	✓
kill_router6	✗	✓	✗	✗
ndpexhaust26	✗	✓	✗	✗
parasite6_fake, real	✗	✓	✗	✗
redir6	✗	✓	✗	✗
rsmurf6	✓	✓	✗	✓
smurf6	✓	✓	✗	✗
toobig	✓	✓	✗	✗
Total = 62	33	47	22	29

Abbildung 6: Original Benchmarkergebnisse [1]

Secure IPv6 with Hyhoneydv6

Katharina Wolf

Universität Potsdam
Sommersemester 2020

Abstract

Im vorliegenden Vortrag werden Entwicklung, Aufbau und Funktionsweise des hybriden Honeypots Hyhoneydv6 erläutert, der im Rahmen der Doktorarbeit von Sven Schindler an der Universität Potsdam entwickelt wurde, um komplexe Angriffe, auch in komplexen IPv6-Netzwerkstrukturen, zu studieren.

1 Einleitung

1.1 Das Konzept Honeypot

Was verbirgt sich hinter Hyhoneydv6? Das soll im Folgenden in mehreren Schritten geklärt werden.

Gehen wir einmal ganz zurück. Bei Hyhoneydv6 handelt es sich um einen sogenannten Honeypot, einer Sicherheitslölsung für Computernetzwerke. [3]

Der Begriff "Honeypot" entstand durch die Analogie Bär-Honigtopf: Der Honig soll den Bär in eine Falle locken oder ihn vom eigentlichen Ziel ablenken. Dieses "Anlocken" ist hier besonders wichtig. Auch der Angreifer wird mit einer Attrappe angelockt, um ihn dort in Ruhe studieren zu können. Speziell sollen so auch neue Angriffe kennengelernt werden.

Neben vielen Methoden der Sicherheit ist der Honeypot somit eine, die sich deutlich von den anderen unterscheidet.

Dies wird auch an der Definition von Spitzner deutlich:

"A honeypot is very different from most traditional security mechanisms. It's a security resource whose values lies in being probed, attacked or compromised" [2]

Es handelt sich hier nicht um ein traditionelles Sicherheitssystem, sondern es gibt eine zusätzliche Schicht. Cyberkriminelle werden nicht abgewehrt, sondern ihr Verhalten wird protokolliert. Programme analysieren Angriffsmethoden und decken Schwächen im System auf.

Es gibt verschiedene Arten von Honeypots. Man unterscheidet zwischen High-Interaction Honeypots, Low-Interaction Honeypots, Hybriden Honeypots und Honeyfarms.

1.1.1 High-Interaction Honeypots

High-Interaction Honeypots sind reale oder virtuelle Maschinen, die echte Dienste anbieten. Dadurch wird eine umfassende Analyse von potentiellen Angriffen, auch in komplizierten Netzwerk-Strukturen, ermöglicht. Das große Problem dabei ist, das der Honeypot selbst angreifbar wird.

1.1.2 Low-Interaction Honeypots

Low-Interaction Honeypots funktionieren anders. Sie emulieren/simulieren Dienste. Dadurch wird der Honeypot nicht selbst angreifbar, was einen großen Vorteil darstellt. Andererseits sind die gewonnenen Infomationen weniger genau und nicht so detailreich wie bei High-Interaction Honeypots.

1.1.3 Hybrid Honeypots

Hybride Honeypots sind eine Kombination von High- und Low-Interaction Honeypots.

Hierbei wird versucht, die Vorteile von beiden zu verbinden und die Nachteile abzuschwächen. Der Honeypot sollte also möglichst nicht angreifbar sein, aber trotzdem möglichst genaue Informationen liefern.

1.1.4 Honeyfarms

Unter einer Honeyfarm versteht man eine Sammlung von mehreren Honeypots. Dabei können bis zu mehreren 100 oder 1000 Honeypots zusammengeschlossen werden, was aber heutzutage aufgrund des großen Aufwandes kaum mehr gemacht wird.

1.2 Hyhoneydv6

Mit dem voherigen Wissen können wir uns nun anschauen, was sich hinter Hyhoneydv6 verbirgt. Es handelt sich um eine hybride Honeypot-Architektur, die in IPv6–Netzwerken eingesetzt werden kann (alles bis dahin war fast ausschließlich nur für IPv4-Netzwerke konzipiert). Entwickelt wurde Hyhoneydv6 im Rahmen der Doktorarbeit von Sven Schindler an der Universität Potsdam. Die Entwicklung erfolgete in zwei Schritten durch Erweiterung des bestehenden Low-Interaction Honeypots Honeyd, was im Folgenden weiter ausgeführt werden soll.

2 Erweiterung von Honeyd zu Honeydv6

Die Erweiterung von Honeyd zu Honeydv6 kann als ersten Schritt in der Entwicklung des hybriden Honeypots Hyhoneydv6 verstanden werden.

Die Grundlage hierzu bildet der bekannte Low-Interaction Honeypot Honeyd, der von Niels Provos entwickelt wurde. Er ist ein Open-Source-Computerprogramm, das dem Benutzer erlaubt, mehrere virtuelle Hosts in einem Computernetzwerk zu installieren und laufen zu lassen. Man kann sogar ganze Netzwerktopologien auf einer Maschine simulieren.

Alle Hosts, die simuliert werden sollen, werden in einem configuration file gespeichert.

Das Verhalten eines simulierten Hosts kann durch ein sogenanntes System-Template spezifiziert werden. Zu jedem Host existiert also ein jeweiliges Template. Die Configuration Database verwaltet die verschiedenen Templates.

Wenn nun ein IPv4-Paket eintrifft, sucht Honeyd nach einem korrespondierendem Template, basierend auf der Zieladresse. Wenn kein korrespondierendes Template besteht wird das Paket verworfen, ansonsten wird es zum Dispatcher weitergeleitet, der das Paket wiederum, abhängig vom IP-Payload zu einem TCP-, UDP- oder ICMP-Prozessor weiterleitet.

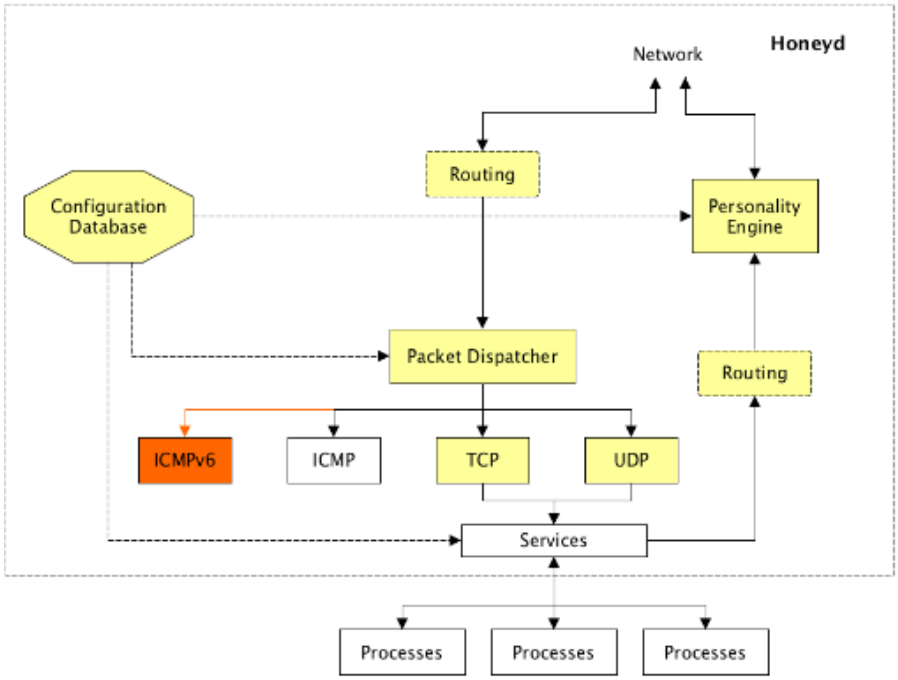


Figure 1: Interne Honeydv6-Architektur [1]

In Figure 1 ist dieser Weg auf der linken Seite sehr gut zu erkennen.

Außerdem wird hier verdeutlicht, welche Veränderungen nötig waren, um Honeyd zu Honeydv6 zu erweitern.

Zu sehen ist nämlich die interne Honeydv6-Architektur, basierend auf der Honeyd-Architektur. Alle gelb markierten Komponenten wurden modifiziert, orange markierte neu hinzugefügt. Die einzige komplett neue Komponente ist der ICMPv6-Prozessor, nötig zur Verarbeitung von IPv6-Paketen.

Das ICMP (Internet Control Message Protocol) reicht für ein IPv6-Netzwerk nicht mehr aus. Unter anderem benutzt ICMP das ARP (Adress Resolution Protocol) zur Auflösung von IPv4-Adressen in MAC-Adressen, ICMPv6 (Internet Control Message Protocol Version 6) benutzt das ND (Neighbour Discovery) - Protocol (zur Auflösung von IPv6-Adressen in MAC-Adressen), das ND-Protokoll muss also mitinstalliert werden.
Auch die Konfigurierung weist einige Unterschiede auf.

Die Menge der IPv6-Adressen ist so groß, dass die Configuration Database nur im Vergleich wenige enthalten kann und so viele Pakete verworfen werden würden.

Eine Lösung hierfür ist die "Konfigurierung on demand":

Wenn ein Paket eintrifft, versucht Honeydv6 einen Low-Interaction Honeypot-Host, der der Zieladresse entspricht, zu finden. Wenn dieser keinen findet, wird das Paket nicht, wie bei Honeyd, verworfen, sondern ein neues Template wird dynamisch erstellt.

Ein Problem könnte die Zeit sein, die wir hierfür benötigen, was auch zu einem späteren Zeitpunkt an ähnlicher Stelle nochmal aufgegriffen wird.

2.1 "Austricksen" von Nmap-Fingerprinting

Um zu verstehen, wie Honeydv6 arbeitet, sollte man noch einmal zu Honeyd zurückschauen. Wie wir in der Einleitung erörtert haben, soll ein Honeypot einen Angreifer anlocken. Dazu hilft es, sich einmal in die Rolle des Angreifers hineinzuversetzen und zu schauen, wie dieser arbeitet.

Eine viel verwendete Methode ist der Netzwerk-Scan und eine viel verwendete Software hierfür ist Nmap. Schauen wir uns Nmap etwas genauer an, zunächst für IPv4-Netzwerke.

2.1.1 TCP/IP-Fingerprinting mit Nmap

Nmap sendet 16 spezielle (TCP-, UDP- und ICMP-) Testpakete (hier erkären sich auch einige Komponenten aus Figure 1!) an die Zielmaschine und lauscht auf die Antworten. Die Attribute der Nachrichten/Antworten werden analysiert und kombiniert, um daraus einen sogenannten Fingerprint zu generieren. Dieser wird dann mit der Fingerprint Database abgeglichen und der Angreifer kann so das Betriebssystem identifizieren.

In IPv6-Netzwerken funktioniert das TCP/IP-Fingerprinting mit Nmap im Prinzip genauso, es werden jetzt nur 18 statt 16 Testpakete verschickt und der Abgleich geschieht nicht mit Hilfe iener Datenbank, sondern mit (in der Zeit der Entwicklung beliebten) Machine-Learning-Methoden (insbesondere logistic regression). Dies soll an dieser Stelle aber nocht weiter ausgeführt werden.

2.1.2 Implementierung einer IPv6 Personality Engine für Honeydv6

Wie kann man dieses Wissen nun nutzen?
Dafür haben sich die Entwickler von Honeyd/Honeydv6 etwas sehr Schlaues ausgedacht, nämlich die Implementierung einer Personality Engine.

Diese erlaubt die Imitation von willkürlichen Betriebssystemen aus dem Nmap-Fingerprinting-Modell.

Die Tatsache, dass es Durchschnittswerte gibt, die zu einer Klassifizierung führen, war die Grundlage für die Idee. Schickt man nun nämlich genau die Antworten zurück, mit denen der Angreifer ein bestimmtes Betriebssystem identifiziert, können wir dem Angreifer vorspielen, ein bestimmtes Betriebssystem zu sein. Und hätten "den Spieß damit umgedreht":

Der Angreifer, der eigentlich Informationen über uns sammeln wollte, über den können wir jetzt Informationen sammeln.

Schaut man nun noch einmal auf Figure 1 zurück, so kann man nun auch diesen zweiten Weg (Rückweg) auf der rechten Seite der Abbildung nachvollziehen.

Ob dies in der Praxis auch tatsächlich funktioniert, wurde u.a. an einem Studenten getestet, der im Rahmen seiner Bachelorarbeit die Betriebssysteme von drei Maschinen bestimmen sollte. Zwei vin diesen waren ein Honeydv6. Der Student hat drei Betriebssysteme erfolgreich identifiziert, hat hierbei aber nicht erkannt, dass zwei von ihnen Honeypots waren.

3 Erweiterung von Honeydv6 zu Hyhoneydv6

Der zweite Schritt in der Entwicklung des Hyhoneydv6 ist die Erweiterung von Honeydv6, wie wir ihn im vorigen Abschnitt kennengelernt haben, zu Hyhoneydv6.

Was kann Hyhoneydv6 jetzt noch zusätzlich?

Das "Hy" steht wir hybrid. Erinnern wir uns noch einmal an die Einleitung und die Nachteile des Low-Interaction Honeypot (wie Honeydv6 einer ist) , nämlich, dass die gesammelte Information nicht so genau und detailreich ist.

Hyhoneydv6 verfolgt nun das Ziel, auch komplexere Angriffe studieren zu können.

Die hybride Architektur beruht auf zwei Layern, wie man in Figure 2 sehr gut erkennen kann.

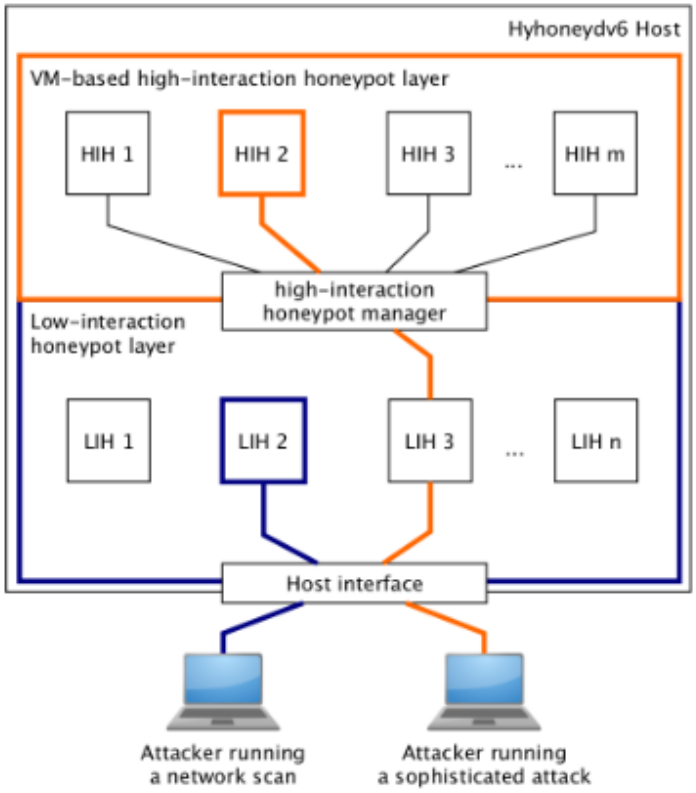


Figure 2: Hybride Hyhoneydv6-Architektur: Überblick [1]

In dunkelblau wird ein "einfacher" Angriff dargestellt, der vom Low-Interaction Honeypot bearbeitet wird. Ein komplizierterer Angriff, hier in orange dargestellt, wird vom High-

Interaction Honeypot-Manager (der noch genauer beschrieben wird) zum High-Interaction Honeypot weitergeleitet.

4 Die Architektur von Hyhoneydv6

In Figure 3 sind die internen Komponenten von Hyhoneydv6 zu sehen.

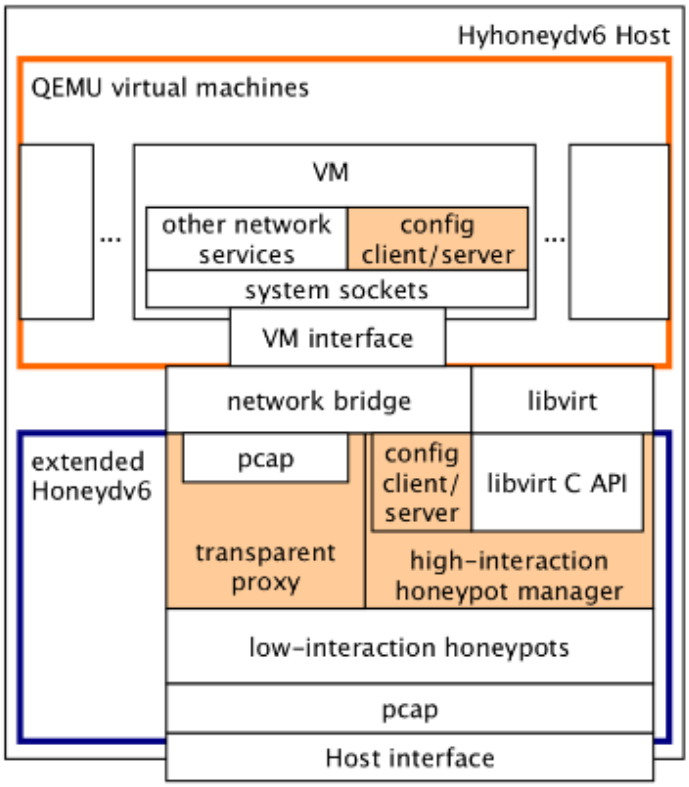


Figure 3: Interne Komponenten von Hyhoneydv6 [1]

Die orange markierten Komponenten sind komplett neue Komponenten und sollen in den folgenden Abschnitten noch genauer erläutert werden.

Wie auch in Figure 2 sind die beiden Layer gut zu erkennen.

Im unteren Teil (in dunkelblau) erkennt man den Low-Interaction-Honeypot-Layer mit dem im vorigen Abschnitt beschriebenen Honeydv6. Wichtige Komponenten (neben den neuen, die noch beschrieben werden), sind pcap (Packet Capture), eine freie Programmierstelle, um Netzwerkverkehr mitzuschneiden und libvirt, eine Sammlung quelloffener Werkzeuge zur Verwaltung von virtuellen Maschinen auf einem Hostsystem.

Im oberen Teil (in orange) erkennt man den High-Interaction-Honeypot-Layer. Dieser besteht aus virtuellen Maschinen (VMs), die durch die Virtualisierungssoftware QEMU (Quick Emulator) realisiert werden.

Verbunden werden beide Layer durch die Network Bridge.

4.1 Der High-Interaction-Honeypot-Manager

Die Hauptfunktion des High-Interaction-Honeypot-Managers ist der Aufbau, Abbau und Backup von High-Interaction-Honeypot-Instanzen (QEMU-basierte virtuelle Maschinen) mit Hilfe von libvirt.

So wie bei Honeydv6 die Erstellung der einzelnen Templates dynamisch erfolgte, gibt es hier eine dynamische Instanziierung. Das Problem hierbei ist, dass durch den Zeitaufwand, der hierdurch entsteht, der Honeypot aufgedckt werden könnte.

Eine Lösung hierfür bietet ein (Maschinen-)Pool, in dem alle laufenden High-Interaction-Honeypots registriert sind. Verschiedene Konfigurationen können hierbei gleichzeitig laufen. Für den High-Interaction-Honeypot-Manager ergibt sich hieraus eine neue Aufgabe: die Verwaltung einer Liste mit allen High-Interaction Honeypots mit Stati und zugeordneten Angreifern.

Intern wird dieser Pool als splaytree (Spreizbaum, spezieller Typ eines Binärbaums, "sich selbst balancierender" Binärbaum) implementiert. Einzelne High-Interaction-Honeypots sind die Knoten. Diese werden im struct hih realisiert (Unterer Teil Listing 1). Dieses besteht aus einer einigartigen ID, einem Configuration Filename (Zeiger auf das original libvirt XML file) und einem ganz neu implementierten Datentyp hih_pool_entry_t (Oberer Teil Listing 1), der Statusinformationen liefert.

```
1  typedef struct hih_pool_entry
2  {
3          virDomainPtr domain;
4          char *xml_configuration;
5          char *path_to_hd_image;
6          int vnc_port;
7          struct addr mac_address;
8          struct addr real_addr;
9          struct addr addr_expected_by_attacker;
10         enum hih_status hih_status;
11         struct event backup_and_remove_timeout;
12         SPLAY_HEAD(attackers, attacker) assigned_attackers;
13 } hih_pool_entry_t;
14
15
16 struct hih {
17         SPLAY_ENTRY(hih) node;
18         char *id;
19         char *configuration_filename;
20         hih_pool_entry_t honeypot_pool[HIH_POOL_SIZE];
21 };
```

Figure 4: Listing 1 [1]

Ein wichtiges Element des Datentyps hih_pool_entry_t ist der Domain Pointer, welcher von libvirt nach Instanziierung einer neuen Maschine erstellt wird.

Weiterhin enthält er einen Zeiger auf das Configuration File, sowie auf den dynamisch erstellten "path to harddisk image" der virtuellen Maschine, so dass ein backup erstellt werden kann, wenn die Maschine nicht mehr in Gebrauch ist, außerdem den VNC (Virtual Network Computing) -Port für die Bilschirmübertragung eines entfernten Rechners.

Auf Einzelheiten zur Hardware wird im Rahmen dieses Vortrags aber verzichtet.

Es folgen einige Adresseinträge, die für die Netzwerkkonfiguration benötigt werden, sowie das Statuselement hih_status mit drei verschiedenen Stati: HIH_AVAILABLE, HIH_IN_USE

und HIH_BOOTING. Nach dem Start von Hyhoneydv6 beginnt der Booting-Prozess und alle virtuellen High-Interaction Honeypots befinden sich im Status HIH_BOOTING. Nach Beendigung des Booting-Prozesses wechselt der Status zu HIH_AVAILABLE, was anzeigt, dass der jeweilige Honeypot vollständig geladen ist und noch keinem Angreifer zugeteilt wurde. Bei der Ankunft eines Angriffs durchsucht der High-Interaction-Honeypot-Manager den Maschinenpool auf Maschinen im Status HIH_AVAILABLE. Wird ein Angreifer zugeteilt, wechselt der Status zu HIH_IN_USE.

Der Eintrag backup_and_remove_timeout speichert benötigte Backup-Informationen.

Eine weitere Aufgabe des High-Interaction-Honeypot-Manager ist die Zuweisung von Angreifern.

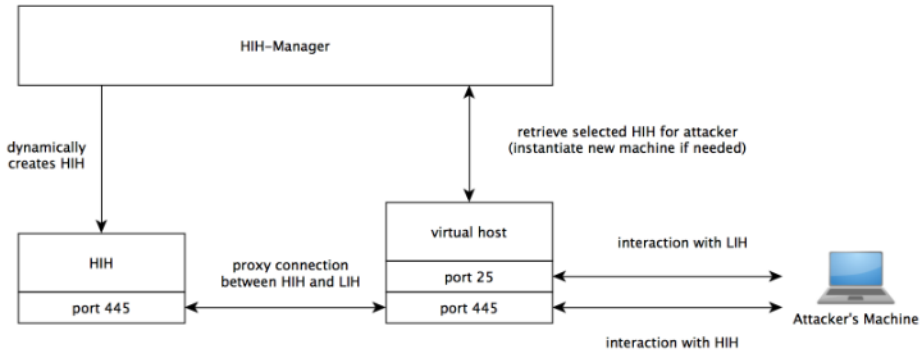


Figure 5: Interaktion zwischen Angreifer und Low- und High-Interaction Honeypot [1]

In Figure 5 ist dies gut zu erkennen. Die Verbindung zu Port 25 wird vom Low-Interaction Honeypot behandelt. Die Verbindung zu Port 445 wird weitergeleitet und anschließend vom High-Interaction Honeypot behandelt.

Für die Interaktion ist der High-Interaction-Honeypot-Manager zuständig. Welcher Angriff weitergeleitet wird, ist vorinstalliert.

In Figure 5 wird außerdem auch nochmal die dynamische Instanziierung verdeutlicht und man erkennt auch schon, wo die dritte neue Komponente, der Transparent Proxy, eingesetzt wird. Dies soll in einem späteren Abschnitt noch genauer beschrieben werden.

4.2 Der Address Reconfiguration Server

Die zweite neue Komponente ist der Address Reconfiguration Server. Dieser hat die eine wichtige Funktion, IPv6-Adressen upzudaten, um die erwartete Zieladresse des Angreifers zu matchen. Dies ist ein weiterer Trick, der dem Angreifer gespielt wird.

4.3 Der Transparent Proxy

Der Transparent Proxy, die dritte neue Komponente, ist verantwortlich für das Weiterleiten vom Low- zum High-Interaction Honeypot. Dabei entscheidet der High-Interaction-

Honeypot-Manager, welcher Angriff weitergeleitet wird. Einem High-Interaction Honeypot wird ein Angreifer außerdem nur zugeteilt, wenn ein Verbindungsaufbau (TCP Handshake) erfolgreich beendet wurde.

Einen allgemeinen Mechanismus, eine bestehende TCP-Verbindung einem anderen Knoten im Netzwerk zuzuteilen nennt man **connection handoff**.

Die Frage ist nun, wie man einen solchen connection handoff zwischen Low- und High-Interaction Honeypot realisieren kann. Dafür gibt es zwei Vorchläge:

Proxied Handoff

In dieser Variante beendet der Low-Interaction Honeypot zuerst den initialen TCP-Handshake mit dem Angreifer und öffnet danach eine Verbindung mit dem High-Interaction Honeypot.

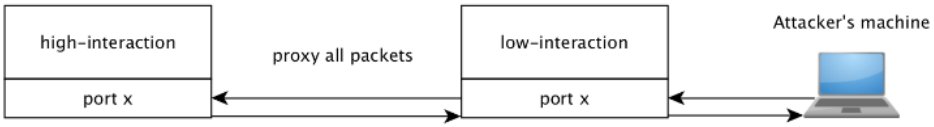


Figure 6: Proxied handoff [1]

Wie man in Figure 6 sehen kann, werden alle Pakete über den Low-Interaction Honeypot geleitet, die ganze Kommunikation passiert den Low-Interaction Honeypot. Alle Nachrichten des Angreifers werden an den High-Interaction Honeypot weitergeleitet und laufen auch auf dem Rückweg über den Low-Interaction Honeypot.

Controlled Bypass

Dies ist die zweite mögliche Variante. Hier wird nur das erste Paket des Angreifers weitergeleitet, danach wird eine direkte Verbindung zum High-Interaction Honeypot aufgebaut.

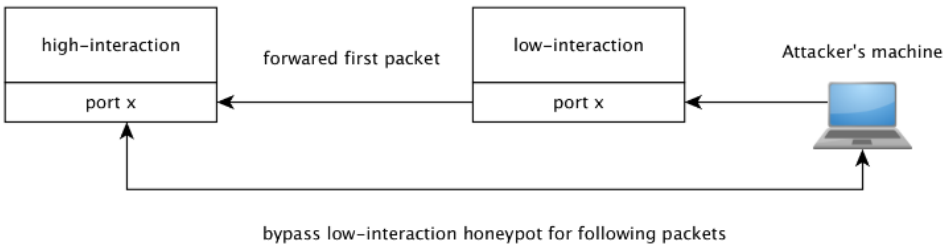


Figure 7: Controlled Bypass [1]

Bevorzugt wird im Allgemeinen der Proxied Handoff. Der einfache Grund dafür ist zunächst, dass bestehende Funktionen von Honeyd dafür genutzt werden können. Außerdem hat man so eine bessere Kontrolle über alle High-Interaction-Honeypot-Verbindungen.

In Figure 8 ist eine Beispielkommunikation dargestellt.

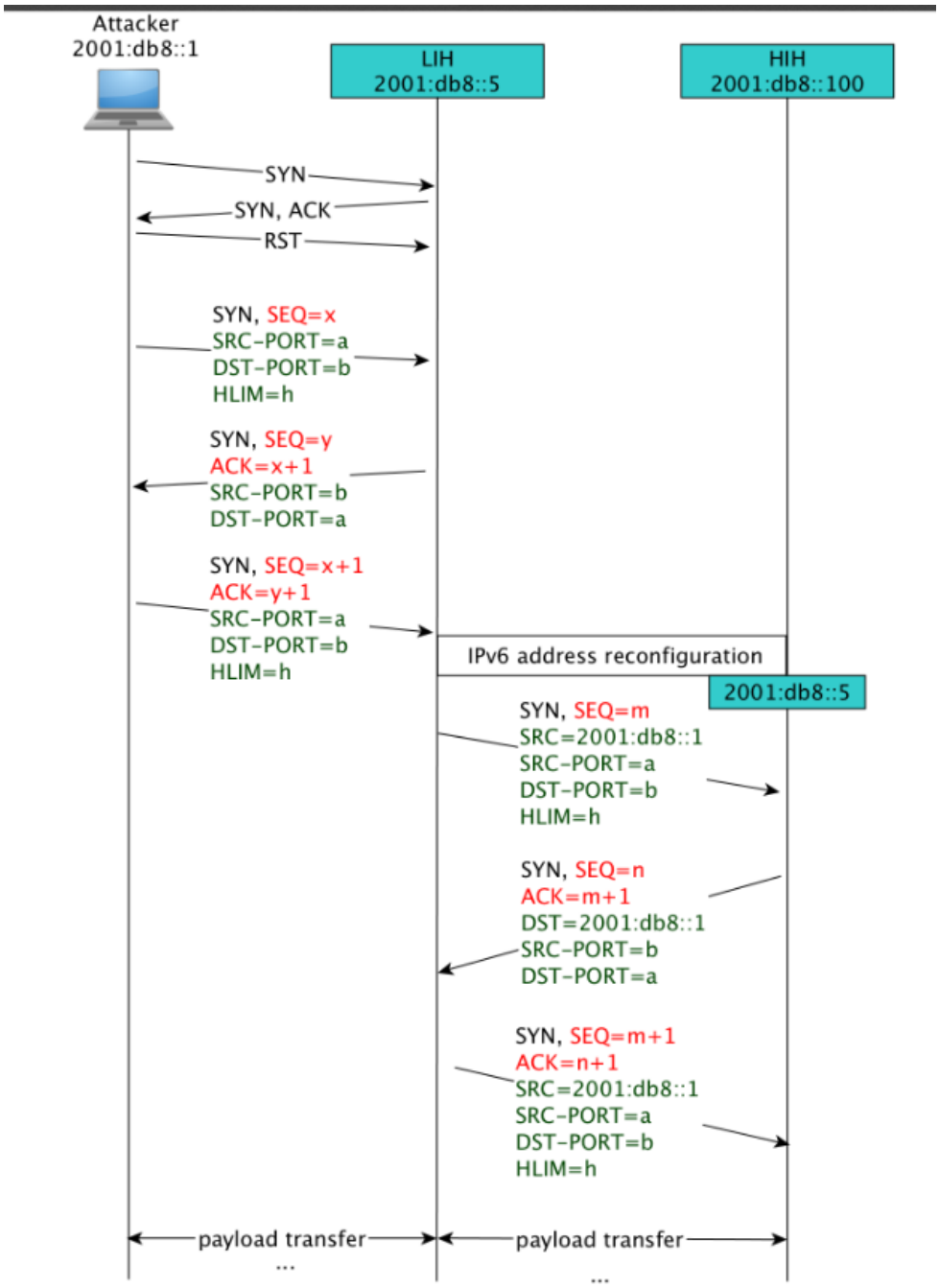


Figure 8: Beispielkommunikation [1]

Wir können hier einen dreimaligen (Versuch zum) TCP-Three-Way-Handshake erkennen.

Zuerst führt der Angreifer einen einfachen TCP SYN scan durch und beendet den Three-Way-Handshake mit einem RST (Reset)-Paket, d.h. der Vorgang wird abgebrochen, es kommt keine Verbindung zustande.

Der zweite Anlauf führt zu einem Verbindungsaufbau mit dem Low-Interaction Honeypot mit der Adresse 2001:db8::5.

In diesem Moment startet nun Hyhoneydv6 die IPv6 Reconfiguration eines verfügbaren High-Interaction Honeypot, sehr gut zu erkennen in der rechten Hälfte von Figure 8.

Die beiden grün marierten Felder zeigen an, dass die Adressen von Low- und High-Interaction Honeypot nun gleich sind. Dadurch bleibt der Honeypot verborgen. Auch die Eigenschaften source(SRC)- und destination(DST)- Adresse und Port sowie das Hoplimit (HLIM) sind gleich, nur die Sequence-Number (SEQ) kann nicht angeglichen werden, was neben der Zeit, die für Address Reconfiguration und evtl. Instanziierung und Übertragung verbraucht wird, zu einer Entdeckung des Honeypots führen könnte.

Im Allgemeinen geschieht das "Weiterreichen" vom Low- zum High-Interaction Honeypot aber durchsichtig für den Angreifer, daher auch der Name Transparent Proxy.

5 Performance

Es wurden zahlreiche Performance-Messungen durchgeführt.

Zunächst wurde Honeydv6 bezüglich Durchsatz und Skalierbarkeit mit Honeyd verglichen. Dabei wurden nur geringe Unterschiede festgestellt. Die Performance war nur minimal schlechter bei Honeydv6 trotz des viel höheren Verkehrs.

Für die Messungen zu Hyhoneydv6 wurde sogar ein eigenes Test Setup erstellt, Details sind zu finden in der Doktorarbeit von Sven Schindler. [1]

Figure 9: Performance measurement test setup [1]

Die Performance-Messungen lieferten im Allgemeinen sehr gute Ergebnisse, vor allem auch mit einfachen Standardgeräten. Das bedeutet, dass man in den meisten Fällen keine teure Hardware und Extrageräte braucht, um Hyhoneydv6 laufen zu lassen.

Beispielsweise konnten vier High-Interaction Honeypots in weniger als zwei Minuten gestartet werden, ein Maschinen-Pool mit 16 verschiedenen High-Interaction-Honeypot-Instanzen innerhalb von 10 Minuten.

Besonders interessant sind die Messungen, die die im vorigen Abschnitt angesprochenen Problemstellen betreffen, vor allem die Zeit, die beim Weiterreichen des Angreifers vom Low- zum High-Interaction Honeypot verloren geht.

In Figure 9 ist das Ergebnis (der Median) einer fünffachen internen Zeitmessung dargestellt.

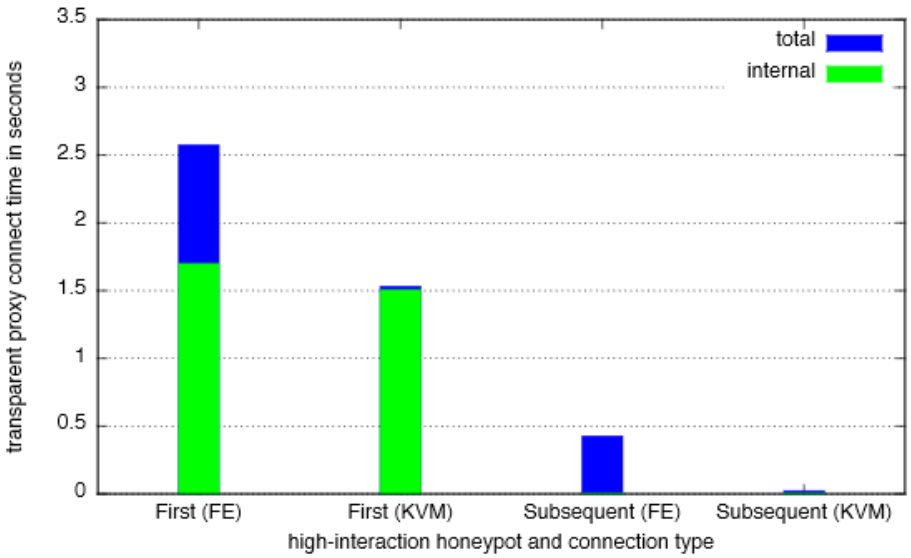


Figure 10: Benötigte Zeit zum Aufbau einer SSH-Verbindung zu einem Debian-basierten High-Interaction Honeypot [1]

Man kann erkennen, dass eine SSH-Verbindung zu einem vollständig emulierten Debian-Honeypot in ungefähr 2,5 Sekunden aufgebaut werden konnte, bei Benutzung einer KVM-basierten virtuellen Maschine sogar in nur 1,5 Sekunden.

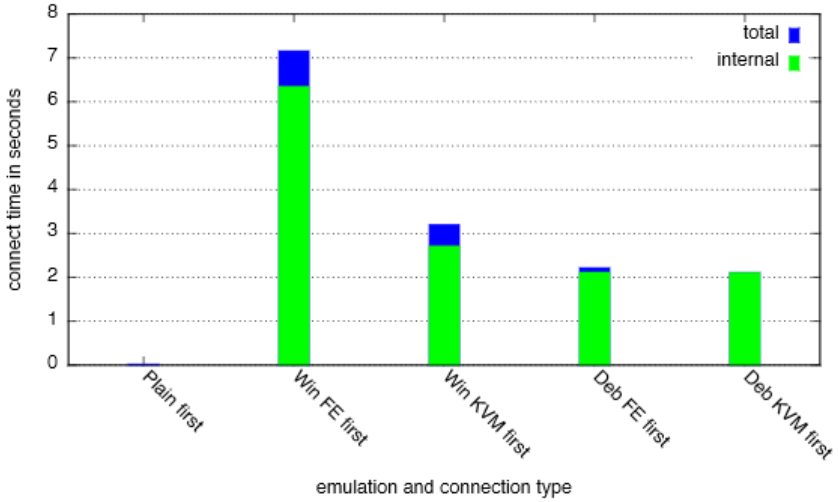


Figure 11: Benötigte Zeit zum Aufbau der ersten HTTP-Verbindung [1]

Figure 10 zeigt, dass es Unterschiede bei der Benutzung verschiedener Maschinen gibt. Braucht die Debian-basierte virtuelle Maschine nur 2,5 Sekunden für den Verbindungsaufbau, so braucht die Windows XP-basierte virtuelle Maschine bis zu 8 Sekunden, was eventuell schon zu lange sein könnte, so dass der Honeypot entdeckt werden könnte.

Weitere Messungen zeigen, dass die Proxy-Implementierung mehr als 200 Anfragen behandeln kann, vollkommen ausreichend für die meisten IPv6-Honeypot-Anwendungsfälle.

6 Zusammenfassung

In den vorigen Kapiteln wurde schrittweise die Entwicklung von Hyhoneydv6, sowie dessen Aufbau und trickreiche Funktionsweise erörtert.

Ganz Allgemein ist Hyhoneydv6 ein Honeypot, also eine Sicherheitsmethode für Computer-Netzwerke, um Angreifer anzulocken.

Dem Angreifer wird vorgespielt, ein bestimmtes Betriebssystem zu sein und im Gegenzug kann sein Verhalten beobachtet und protokolliert werden.

Der bekannte Low-Interaction-Honeypot (Dienste werden simuliert) Honeyd wurde als Grundlage genommen und erweitert, zunächst zu Honeydv6, einem Low-Interaction-Honeypot, der in IPv6- Netzwerken eingesetzt werden kann.

Die Erweiterung zur hybriden Hyhoneydv6-Architektur, bei der bestimmte komplizierte Angriffe (für den Angreifer quasi transparent) zu einem High-Interaction Honeypot (virtuelle Maschine, die echte Dienste anbietet) weitergeleitet werden, ermöglicht sogar die Beobachtung komplexer Angriffe.

References

[1] Sven Schindler, *Honeypot Architectures for IPv6 Networks* (PhD Thesis, University of Potsdam, 2016)

[2] L. Spitzner, *Honeypots: Tracking Hackers* (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002)

[3] `https://it-security-wissen.de/honeypot.html`.

RADIUS

The protocoll behind eduroam

Fabian Lindner

22nd Sept, 2020

Abstract

This documentation aims to analyze the security of RADIUS. We will first give a summary on the packet structure and the types of packets. This also includes how the encryption and authentication of packets work. Afterwards we will shortly discuss what Realms are and how they are used in the eduroam system. At the end, we will show some generell security shortcommings of RADIUS and outline some ideas for possible attacks.

Contents

1 Introduction **2**

2 Packets in RADIUS **3**
2.1 Structure and format of packets 3
2.2 Access packets 4
2.3 Accounting packets 5

3 Use of the Authenticator **5**
3.1 Authentication of the Response packets 5
3.2 Encryption of the password 6

4 Realms in RADIUS **7**

5 Practical applications: eduroam **7**

6 Security concerns **8**
6.1 Generell weaknesses 8
6.2 Reconstructing the original password 9
6.3 Faking of Response packets 10
6.4 Faking of Access Request packets 10

7 Conclusion **11**

8 Further reading **11**

1 Introduction

RADIUS (Remote Authentication Dial-In User Service) is a networking protocoll, which was developed 1992 by Livingston Enterprises. There goal was to create a protocoll, which makes it possible to save the login information of different users at only one central place, but still be able to access these information from different locations in a secure manner. It was first added to the Internet Engineering Task Force in 1997. Since then it has recieved various updates and expansions in order to keep the protocoll up to date with modern technologie and standards. The RADIUS documentation itself only defines a standard and there exist many different implementations of this standard.

More precisely: the protocoll want's to provide a solution to have a centralized Authentication, Authorization and Accounting. (Also known as Triple A or AAA.) For that matter it uses a client and server modell. The generell idea is, that many different RADIUS clients want to verify the login credentials of their users and the RADIUS server, being the only part in the system that has access to the correct information, checks these credentials. A RADIUS server is generally just a deamon process on a Linux server. When it was first documentet it used UDP as a transport layer, because it was considered the easiest to implement, but newer versions also allow TCP to be used to get a better encryption of the packets.

A typical RADIUS system can be seen in figure 1. Here a users tries to

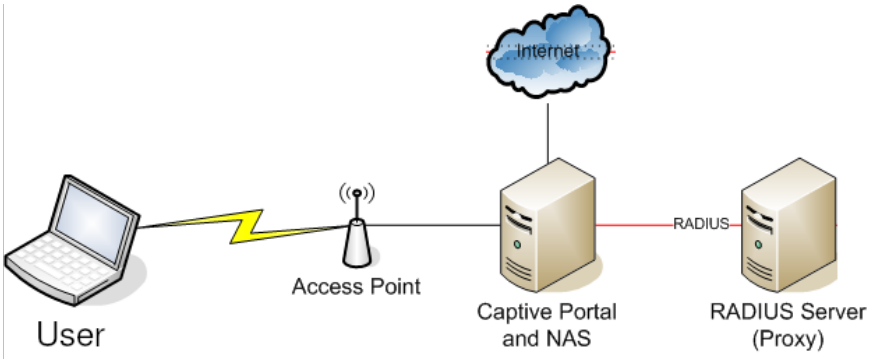


Figure 1: A typical RADIUS system

Source: `https://en.wikipedia.org/wiki/File:Drawing_Roaming_RADIUS.png` (Last accessed: 22nd Sept, 2020, modified)

connect to an access point, which itself is connected to a Network Access Server (NAS). The NAS works as a RADIUS client and can send RADIUS packets to the RADIUS server. The NAS creates a RADIUS packet, which contains

the credentials the users entered to connect to the access point, and sends this RADIUS packet to the RADIUS server. The RADIUS server then verifies the recieved credentials and either confirms the credentials or rejects them. Depending on the result the NAS recieves from the server, it will then allow the user to connect to the internet.

2 Packets in RADIUS

RADIUS has two main types of packets: Access packets and Accounting packets. Additionally RADIUS seperates these types into Requests and Responses. A successfull communication consist of one Request, which is sent by one client to the server, and one corresponding Response, which is sent by the server back to the client. The main structure of each packet is the same.

2.1 Structure and format of packets

Each packet must have the following fields in the order they are listed below:

Code: The Code is a one byte long field, which indicates the type of packet. Each packet type has it's own unique code. The codes are defined in the protocoll and remain equivalent accross different implementations.

Identifier: The Identifiert aids in matching responses to the correct request. For that purpose the client generates a one byte long number and places it in the Identifier field. When creating the response packet, the server copies the Identifier from the request to the response. The client can then use the Identifier from the request to figure out, to which request the incomming response belongs. This is especially needed, if one client sends multiple request out in a short amount of time.

The only constraint the protocoll has on the Identifier, is that it should be unique for every unresponded request. For the sake of simplicity, most implementations just use a simple counter to generate the Identifier.

Length: The Length field is two bytes long and contains the total length of the entire packet.

Authenticator: The Authenticator is 16 Bytes long. It is used in the encryption of the user password and the authentication of the response packets. The Authenticator is further discussed in chapter 3.

Attributes: A packet can have any number of attributes. The amount of attributes a packet has is only indirectly specified by the Length field. It is even possible, that the same attributes appear multiple times in the same packet. The effect this has is different for each attribute. Some attributes however are only allowed to be used in some packet types and some packet types also specifiy attributes that need to be present.

Attributes have their own format. Each attribute has a one byte long Type field, indicating the type of the following attribute. This is followed by a one byte long Length field, which holds the length of the entire attribute. Lastly they have the Value field, which holds the value of the attribute. The length of the Value field is determined only by the Length field of the corresponding attribute.

2.2 Access packets

Access packets are used to validate user credentials. They fulfill the Authentication and Authorization of RADIUS.

A client starts by sending a Access-Request. This Access-Request needs to have at least a Username and Password Attribute, but it can also have more Attributes for different information, such as the MAC-Adress of the user or the location from where he logs in. After the Access-Request is send to the server, the server validiates the credentials. The way the server does this is not further specified and can change between implementations. One common approach is to validate the Username and Password with the Extensible Authentication Protocoll (EAP).

Depinding on the result of the validation the server can answer with one of three possible Responses:

Access-Accept: When the user credentials are correct the server answers with an Access-Accept Response. If the client recieves an Access-Accept the user is allowed to be logged in. The Access-Accept packet can also contain some attributes, which define the services the user is allowed to use and other possible configuration data. This is, for example, usefull to give administrators access to different internal sites, but not to normal users.

Access-Reject: When the user credentials are incorrect the server answers with an Access-Reject Response. If the client recieves an Access-Reject the user is not allowed to be logged in and needs to be rejected by the server.

Access-Challange: This Response is sent by the server, if more information from the user is necessary. To indicate which type of information is needed, the server places a `State` Attribute in the Access-Challange packet. The interpretation of the `State` Attribute is not defined by the protocoll and left to the implementation.

If the client recieves an Access-Challange it asks the users for the additional information. When the users answers the challange, the client creates a new Access-Request packet and places the answer into the Password Attribute. To indicate that this Access-Request is an answer to a challange and not a new Request, the client copies the `State` Attribute from the Access-Challange Response into this new Access-Request. The server can afterwards respond with an Access-Accept, an Access-Reject or another Access-Challange.

2.3 Accounting packets

Accounting packets fulfill the Accounting part of RADIUS protocoll. They are mainly used to log the network access of a user, but can also be used to log other activities of the users.

There are three important types of Accounting-Requests. Every Accounting-Requests needs to have a way to identify a users. This can be the username, but in most implementations it is just a user id. The type of an Accounting-Request is specified with a special Attribute called `acct_status_type`.

One important type is Accounting-Start. This packet is sent, when the users starts to use a specific service. It contains the type of service the users uses.

When the user stops the use of one service, the client sends an Accounting-Stop packet. This type marks the end and contains all relevant statistics network access, these may include the time the user was connected to the internet or the ammount of data transfered from and to the user.

Inbetween an Accounting-Start and an Accounting-Stop the client can send Interim-Updates, which hold the statistics for one user up to the time the client sent out an Interim-Update. This can be used, so that the client doesn't have to keep all the information in memory.

Every Accounting-Request is confirmed by the server with a Accounting-Response. This Accounting-Response holds no further information and is just used to let the client know, that the information was delivered successfully.

3 Use of the Authenticator

RADIUS has built in features for encrypting the Password Attribute and authenticating the Response Packets. Both of these features use the Authenticator field.

Additionally both features also use a shared secret. A RADIUS server and client have a common shared secret. This shared secret is entered manually via an administrator, when setting up a new RADIUS client. A RADIUS server has a unique shared key for every client and accepts only Request packets from these clients.

The Authenticator in the Response packets, called the Response Authenticator, is generated by the client. It is a random 16 byte long sequence, which serves no real purpose on it's own.

3.1 Authentication of the Response packets

The Authenticator for the Response packets, also called the Response Authenticator, is calculated by the server, before transmitting the Response packet. It is calculated with the following formula:

$$\begin{aligned}\text{Authenticator}_{\text{Response}} = \text{MD5}(&\text{Code} \mathbin{+\!\!+} \text{Identifier} \mathbin{+\!\!+} \text{Length} \\ &\mathbin{+\!\!+} \text{Authenticator}_{\text{Request}} \mathbin{+\!\!+} \text{Attributes} \mathbin{+\!\!+} \text{shared\_secret})\end{aligned}$$

where Code, Identifier, Length and Attributes are the corresponding fields of the Response packet, ++ denotes string concatination and MD5 is the MD5-Hash function. The MD5-Hash function always outputs 16 byte long sequences.

When the client recieves the Response packet, it has all the information available to do the same calculation again and therefor is able to confirm that the packet was transmitted correctly and sent by the RADIUS server and not an unknown third party. This confirmation relies mainly on the shared secret and the Request Authenticator. The Request Authenticator specifically is there to ensure, that every Response Authenticator is unique and not only dependent on the Response itself. If the Request Authenticator wouldn't be part of this calculation, the Response Authenticator would always be identical for two identical Response packets, which would make it very easy to fake Response packets.

3.2 Encryption of the password

The Password Attribute is the only Attribute which is encrypted by the RADIUS protocoll. For that, the password is split into 16 byte long segments and these segments are then encrypted iterativly. If needed, the last segment is padded to reach the desired length of 16 bytes. The first segment is encrypted with the following formula:

$$\text{Password}_{\text{new}_1} = \text{Password}_{\text{old}_1} \oplus \text{MD5}(\text{shared\_secret} \mathbin{+\!\!+} \text{Authenticator}_{\text{Request}})$$

where $\oplus$ is a bitwise XOR operation. After that, we can use the following formula to encrypt the other segments of the password one after another:

$$\text{Password}_{\text{new}_i} = \text{Password}_{\text{old}_i} \oplus \text{MD5}(\text{shared\_secret} \mathbin{+\!\!+} \text{Password}_{\text{new}_{i-1}})$$

The server can repeat the same calculation to retrieve the original Password. This is possible, beacuse the following rules always holds for XOR: $x \oplus x = 0$ and $x \oplus 0 = x$. Therefor we have:

$$\begin{aligned} &\text{Password}_{\text{new}_i} \oplus \text{MD5}(\text{shared\_secret} \mathbin{+\!\!+} A) \\ =\ &\text{Password}_{\text{old}_i} \oplus \text{MD5}(\text{shared\_secret} \mathbin{+\!\!+} A) \oplus \text{MD5}(\text{shared\_secret} \mathbin{+\!\!+} A) \\ =\ &\text{Password}_{\text{old}_i} \end{aligned}$$

where A is either the Request Authenticator or the encrypted previous segment of the password, depinding on the value of i.

So we have an encryption of the password, which is different each time, since the Request Authenticator is different each Request, and secure, since it relies on the shared secret, which only the RADIUS client and server should know. Furthermore the password can easily be decrypted, if one knows the shared secret.

4 Realms in RADIUS

With RADIUS multiple clients can connect to the same RADIUS server to verify user credentials at many different locations. But RADIUS has another feature, called Realms, which make it possible to have also multiple RADIUS servers, so that clients don't have to connect to the RADIUS server that stores the specific user credentials, but can instead connect to any RADIUS server in the system.

Every RADIUS server in a RADIUS system has a unique Realm. A Realm is just a string, which indicates a specific RADIUS server. It can be thought of as a name for the server. When a client sends a packet to a RADIUS server, it can specify, that the Request should be forwarded to a specific Realm. If a RADIUS server recieves a Request, that is intended for a different Realm, it will act as a proxy and transmitt the Request to the RADIUS server with the correct Realm. To do this, every RADIUS server keeps a list, with Realms and the specific RADIUS server, that Realm belongs to. This list has to be manually created by an administrator.

If a Realm is present, it is transmitted as part of the Username Attribute. There are two common ways to do this: First, and most commonly, is to transmitt the Realm in postfix notation, meaning, that the Realm is transmitted after the real Username. The Username and the Realm are commenly seperated by an '`@`'. The resulting final Username resembles an E-Mail, but the Realm does not need to be a real Mail-Domain. Secondly, the Realm can be transmitted in prefix notation, meaning, that the Realm is transmitted before the real Username. The common seperator here is '\' and the final Username resembles a file path.

It is also possible to have more than one Realm for a Request. This leads to a proxy chain, where a Request is transmitted over multiple different RADIUS servers, until it reaches it's final destination. A good example of this can be found by looking at eduroam, as explained in chapter 5.

5 Practical applications: eduroam

Eduroam (education roaming) is a roaming service for research institutions. It aims at supplying a roaming service to many different organizations, that can be accessed with credentials from any participating institution.

To achieve this goal, the eduroam system uses multiple different protocolls, one of which being RADIUS. Eduroam makes extensive use of Realms, in order to redirect the Requsts to the institution that the user belongs to.

The Realms in eduroam are organized in three different layers, which follow a strict hierarchy. The eduroam servers have only Realms registered for servers of a lower or higher layer, but not for servers on the same layer.

On the most bottom layer, we have the eduroam server of the different institutions. Each institution has exactly one eduroam server.

On the second layer, we have one eduroam server for each participating nation. These servers have a Realm configured for all the eduroam servers from

the institutions in their country.

On the top layer, we have one worldwide server. This server has a Realm configured for all the nationwide eduroam servers.

Using this layer based approach, every server only needs to know a few Realms, which makes setting up a new eduroam server signifficantly easier.

Eduroam uses two Realms to pass all packets arround correctly. The first Realm specifies the country the institution of the user is located in, using the two letter country code domain. And the second Realm specifies the specific institution of the user. Both Realms are given in the postfix notation, so that the final results resembles a full E-Mail Adress: `USERNAME@INSTITUTION.COUNTRY`. When a users want's to log in, he needs to enter the full Username, including the institution realm and country code.

When an eduroam server recieves a packet for a different Realm than it's own, it will forward the packet to the nationwide eduroam server. The nationwide server will look at the Realm and forward the packet again to the eduroam server from the institution with that Realm or to the worldwide server, if the country code doesn't match. The world-wide server then forwards the packet to the nationwide server with the corresponding country Realm, from where it is again forwarded to the final eduroam server of the institution.

6 Security concerns

Now that we have discussed how RADIUS works, we will look at some generell security weaknesses that the protocoll has and outline some generell ideas for attacks against the protocoll.

We will only look at the protocoll itself and assume that there are no additional security systems in place. For that matter, we will assume, that a hacker is able to intercept and read the sent packets, as well as send fake packets to the RADIUS server and client. Since a hacker is able to intercept the packets, it is also possible, that he stores previously sent packets in a database, for future access. Having access to previously sent packets is one key feature, which makes the attacks described below possible. We will also assume, that a hacker is able to make login attempts at the access point, that is connected to the RADIUS client.

6.1 Generell weaknesses

The first generell problem of RADIUS is that only the password is encrypted and no other information. This is a problem, since a hacker is still able to infer other private or sensible information from the packets, for example when a specific user tries to log in and from which location.

Another problem is that Request packets are not really authenticated by the server. The server only accepts packets, which come from clients which have a shared secret configured. Therefor this is only a problem, if a hacker can change their own IP and MAC adress.

One of the big problems of RADIUS is the use of MD5 Hashs. MD5-Hashs are long since considered insecure and it is really easy, even with just normal modern home computers, to find Hash collisions, meaning we can easily find inputs to the Hash function, which create the same output hash. This on itself doesn't break security, but it makes brute force attacks considerably easier, since it limits the search space for the shared secret immensely.

The biggest problem however is the use of XOR. A XOR operation still leaves a lot of information, that we can deduce. The attacks described below make use of the weak encryption of XOR.

6.2 Reconstructing the original password

It is possible to reconstruct parts of the original password of a user. To do this, some conditions need to be met. First of, we need to have two different Access-Request packets, that have the same Request Authenticator. Usually the Request Authenticator should not repeat itself for a very long time, since it is, with 16 bytes, quite long. But there were cases, where the random number generator was implemented poorly, so that the Request Authenticator repeated itself. Secondly, the packets also need to have a different password and both passwords need to be equal or shorter than 16 bytes. We can than XOR both encrypted passwords together and get the XOR of the unencrypted (decrypted) passwords:

$$
\begin{aligned}
&\text{Password1}_{\text{enc}} \oplus \text{Password2}_{\text{enc}} \\
=\ &\text{Password1}_{\text{dec}} \oplus \text{MD5}(\text{shared\_secret} \mathbin{+\!\!+} \text{Authenticator}_{\text{Request}}) \\
\oplus\ &\text{Password2}_{\text{dec}} \oplus \text{MD5}(\text{shared\_secret} \mathbin{+\!\!+} \text{Authenticator}_{\text{Request}}) \\
=\ &\text{Password1}_{\text{dec}} \oplus \text{Password2}_{\text{dec}}
\end{aligned}
$$

Since a password has normally just ascii printable symbols, we can use this equation to make brute force attacks on the passwords easier. This is done by limiting the search scope only to the values for Password1, that leave Password2 ascii printable. When doing this, we effectivly brute force two passwords at the same, making our attack double as efficient.

Under the assumption, that one password is shorter than the other password, we can even find out the higher bytes of the longer password. This is possible, since the passwords are padded with zero-bytes, to reach the desired length of 16 bytes. By knowing the higher bytes of one password, we can limit our search scope for a brute force attack even more.

If we can take advantage of a bad random number generator and predict the next Request Authenticators, we can carefully make a login attempt ourselfs, so that the resulting Request Authenticator is identical to the Request Authenticator of our target password. Since we can choose the password to login ourselfs, we already know one of the passwords, which means we can find out the other password, by doing a simple XOR operation.

6.3 Faking of Response packets

When we intercept a Request packet with a Request Authenticator, that was already used in a previous Request, we can retransmit the corresponding Response of that old Request.

As a reminder, here is the formula for generating the Response Authenticator, as already seen in chapter 3:

$$\begin{aligned}\text{Authenticator}_{\text{Response}} = \text{MD5}(&\text{Code} \mathbin{+\!\!+} \text{Identifier} \mathbin{+\!\!+} \text{Length} \\ &\mathbin{+\!\!+} \text{Authenticator}_{\text{Request}} \mathbin{+\!\!+} \text{Attributes} \mathbin{+\!\!+} \text{shared\_secret})\end{aligned}$$

We see that the Response Authenticator is dependent on the contents of the Response and the Request Authenticator. Since the Request Authenticator is identical, we can leave the contents of the Response identical, and the calculation for the Response Authenticator will yield the same result. Therefor the RADIUS client will accept the packet, as a Response from a server.

If our database of previously transmitted packets is big enough, we can do this procedure for nearly every Request, that the client sends. If every fake Response packet, that we send, is a Access-Reject, we effectively disable the functionality for users to log in on this RADIUS client, resulting in a denial-of-service attack.

6.4 Faking of Access Request packets

We are also able to fake Access Request packets very easily. As discussed previously, a hacker needs to mask his own IP and MAC adress, so that the server accepts the Request packets. This is usefull, when we want to run a brute force attack on a user password against the RADIUS server directly. There is however one constraint: the password can not be longer than 16 bytes.

To do this, we first need to make one manuall login attempt at the access point, with a password shorter than 16 bytes. We can than intercept the Request packet and use this packet as a blueprint to create new Request packets. We have to, however, leave the Request Authenticator the same, since it is used in the password encryption.

As a reminder, here is the formula for encrypting the password, as already seen in chapter 3:

$$\text{Password}_{\text{new}_1} = \text{Password}_{\text{old}_1} \oplus \text{MD5}(\text{shared\_secret} \mathbin{+\!\!+} \text{Authenticator}_{\text{Request}})$$

Since the password is equal or shorter than 16 bytes, it is not split into multiple segments. Because we leave the Request Authenticator identical, the entire input to the MD5 Hash funtion will also stay identical, meaning the resulting hash value will always be the same. Since we chose the original password ourselfs, we can use this original password to do an XOR operation with the encrypted

password, leaving only the resulting MD5 Hash:

$$\begin{aligned}
&\text{Password}_{\text{new}_1} \oplus \text{Password}_{\text{old}_1} \\
&= \text{Password}_{\text{old}_1} \oplus \text{MD5}(\text{shared_secret} \mathbin{+\!\!+} \text{Authenticator}_{\text{Request}}) \oplus \text{Password}_{\text{old}_1} \\
&= \text{MD5}(\text{shared_secret} \mathbin{+\!\!+} \text{Authenticator}_{\text{Request}})
\end{aligned}$$

We can then use this MD5 Hash to encrypt every possible password, as long as we leave the Request Authenticator untouched. All other fields in a Request packet are not encrypted, so we don't have to do any additional calculations, before creating and transmitting our fake Request packet to the server. Many implementations of RADIUS are highly vulnerable against this attack, since they don't expect a Request Authenticator to be reused and therefor don't check against that.

7 Conclusion

RADIUS is not a secure protocoll anymore. Any form of external encryption is strongly advised. Additionally the packets should be transmitted securly, so that no interception is possible.

These problems don't exist when using RADIUS over TCP (RADIUS/TCP), since there the protocoll mandates the use of an external encryption, like TLS or IPSec. The same is true for the successor of RADIUS: Diameter. Eduroam also has none of these problems, since it uses a secure End-to-End encryption and other protocolls to secure the users credentials and other data.

8 Further reading

The official documentation for the basic RADIUS protocoll can be found here: `https://tools.ietf.org/html/rfc2865` and here `https://tools.ietf.org/html/rfc2866`.

Updates and changes, for using RADIUS with TCP and TLS, can be found here: `https://tools.ietf.org/html/rfc6613` and here `https://tools.ietf.org/html/rfc6614`. And the documentation for Diameter can be found here: `https://tools.ietf.org/html/rfc6733`.

The official documentation for eduroam can be found here: `https://tools.ietf.org/html/rfc7593`.

All of these are only the respective latest base documentation. Since then updates and extensions have been released, so all of these are not complete.

For a good overview of more possible attacks on the RADIUS protocoll and some suggestions on how to improve the protocoll I recommend "An Analysis of the RADIUS Authentication Protocol" by Joshua Hill, 2001: `https://www.untruth.org/~josh/security/radius/radius-auth.html`.

All sites have been last accessed at 22$^{\text{nd}}$ Sept, 2020.