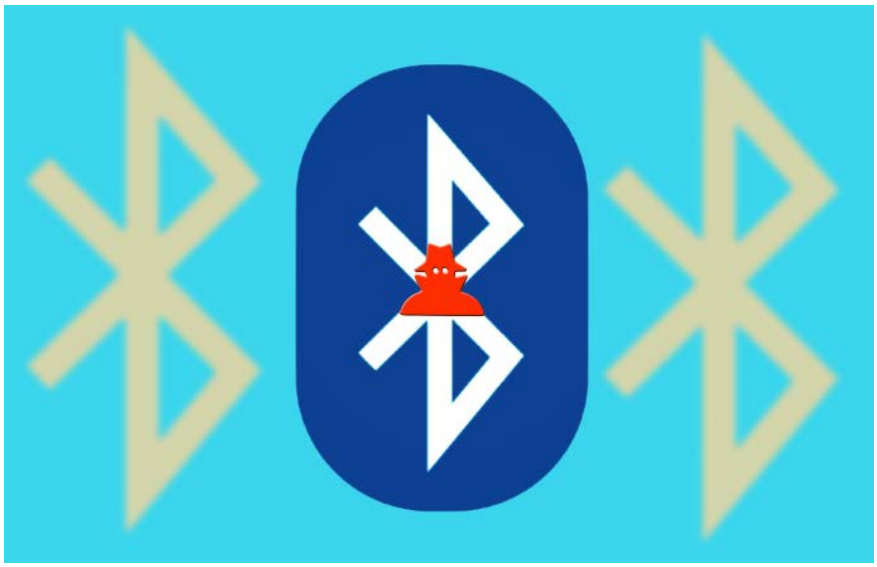


Angriffe auf Bluetooth Low Energy

Auf den Zahn gefühlt

Sarah Mader

BLE kennt einige Verschlüsselungs- und Privacy-Mechanismen. Doch wie wenig Verwendung sie finden, zeigen schon wenige Hacks.

Wie leicht Embedded-Geräte anzugreifen sind, haben bereits mehrere Artikel dieser IoT-Hacking-Serie gezeigt. Der letzte beschäftigte sich mit Replay-Angriffen über Funk [1]. Eine besondere Rolle unter den Drahtlostechniken für IoT-Geräte spielt BLE (Bluetooth Low Energy), auch Bluetooth Smart genannt. Ihr Einsatzfeld verteilt sich weit über alle Branchen.

Seit Bluetooth 4.0 ist BLE offiziell in den Standard integriert. Seitdem pflegt und entwickelt die Bluetooth SIG (Special Interest Group) BLE. Die aktuelle Bluetooth-Spezifikation Version 5.2 umfasst über 3000 Seiten. Deshalb kann der Artikel einleitend nur einen groben Überblick über BLE und seine Sicherheitsmechanismen geben, bevor er sich den Angriffen über BLE widmet.

Bluetooth Low Energy agiert wie Bluetooth oder Wireless LAN auf dem lizenzfreien ISM-Frequenzband (Industrial, Scientific and Medical) zwischen 2,4 und 2,48 GHz. Dabei reduziert die Frequenzsprungtechnik Wechselwirkungen mit anderen auf dem Band agierenden Geräten. Dafür wird das Frequenzband von Bluetooth und BLE in Kanäle aufgeteilt – im Fall von BLE sind es 40. Nach jedem Paket wechselt die Übertragung den Kanal nach einem zuvor ausgehandelten Verfahren. Das vermeidet die starke Auslastung einzelner Kanäle.

Im Bluetooth-Protokollstack verteilen sich die sechs Schichten unterhalb des Application Layer auf den Host und den Controller, die das Host-Controller-Interface miteinander verbindet (siehe Abbildung 1). Die unteren zwei Schichten, PHY und Baseband, sind unter anderem für den Transport der Daten, die Fehlerkorrektur, das Paketmanagement und die Flusskontrolle zuständig. Die Link-Layer-Schicht, die ebenfalls dem Controller zugeordnet ist, organisiert das Verbindungsmanagement.

Auf Hostebene wiederum stellt die L2CAP-Schicht verbindungslose und verbindungsorientierte Verbindungen für die darüberliegenden Protokolle zur Verfügung. Der Security-Manager ist für das Verwalten der kryptografischen Schlüssel und das Absichern der Verbindungen und des Datenaustausches zuständig. Das Generic Access Profile bietet die Basisfunktionen für alle Bluetooth-Geräte und interagiert dafür direkt mit der Anwendungsschicht. In dieser Schicht sind das Auffinden anderer Geräte, die Verbindungsmodi und die Sicherheit angesiedelt. Das Attribute Protocol propagiert die Geräteinformation und -attribute, während das Generic Attribute Protocol als Service-Framework dient.

Die Peripherie ist der Server

Bei BLE spricht man bei den Geräten häufig von Central und Peripheral respektive Master und Slave. Dabei nimmt meist ein Smartphone oder ein Computer die Rolle des Central oder Master ein. Ein Peripheral oder Slave ist meist ein Gerät, das Dienste bereitstellt, beispielsweise ein Pulssensor oder Peripheriegeräte zur Ein- und Ausgabe.

Um die Dienste eines Peripheral zu nutzen, muss sich das Central zunächst mit

X-TRACT

- Bluetooth Low Energy ist bei IoT-Geräten weit verbreitet, kennt aber viele Pairing-Mechanismen, die die Verbindung unterschiedlich stark absichern.
- Zum Hacken von BLE-Verbindungen benötigt man keine Profiwerkzeuge. In den meisten Fällen genügen die in Smartphones und Computern vorhandenen BLE-Chips samt Open-Source-Software.
- Mit passiven Angriffen lassen sich vertrauliche Informationen erbeuten.
- Aktive Angriffe hingegen manipulieren Bluetooth-Geräte.

ihm verbinden. Hierfür stellt BLE drei der verfügbaren 40 Kanäle als Advertising-Kanäle bereit. Die restlichen 37 dienen der Datenübertragung. Um für andere Geräte sichtbar zu sein, sendet das Peripheral Advertisement-Nachrichten auf den entsprechenden Kanälen. Das Central scannt diese Kanäle und baut nach dem Empfangen eines Advertisement eine Verbindung auf. Danach tauschen die beiden Geräte Informationen über die verfügbaren Verbindungsoptionen aus.

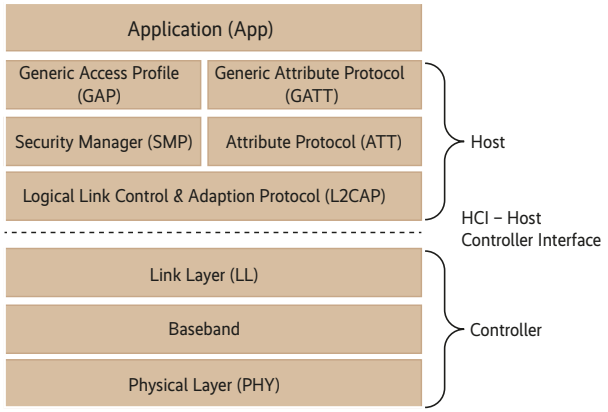
Bei BLE stellt ein Peripheral seine Dienste über das GATT-Profil nach dem Client-Server-Konzept zur Verfügung. Der GATT-Server ist dabei üblicherweise auf dem Peripheral implementiert. Er verfügt über eine Anzahl an Attributen, und das GATT-Profil beschreibt, wie der Client sie per ATT-Protokoll finden und verwenden kann. Er fragt in der Phase der GATT Service Discovery die Informationen über das GATT-Profil des Peripherals ab (siehe Abbildung 2). Optional können die Geräte in einem Pairing-Prozess kryptografische Parameter und Schlüssel generieren und austauschen.

Je nach Version kann der Pairing-Prozess mit zwei unterschiedlichen Verfahren arbeiten. Version 4.0 und 4.1 verwenden das BLE Legacy Pairing, ab Version 4.2 sind auch BLE Secure Connections möglich. Beide Varianten bieten für das Pairing je nach verfügbaren Ein- und Ausgabeschnittstellen der beiden Geräte – Displays oder Tastaturen etwa – unterschiedliche Verfahren für den Schlüsselaustausch an.

Viele Wege, die Schlüssel zu generieren

Für das Legacy Pairing sind drei Verfahren zum Austauschen eines Temporary Key (TK) spezifiziert. Beim Just Works setzen beide Geräte den TK auf 0. Komplexer ist das Passkey Entry: Die Geräte tauschen eine sechsstellige PIN aus und nutzen dazu ihre Ein- und Ausgabeschnittstellen. Beispielsweise generiert ein Gerät die PIN und zeigt sie an. Der Anwender muss sie dann auf dem anderen Gerät eingeben. Die PIN dient als TK und bietet so 20 Bit Entropie. Beim OOB (Out of Band) hingegen handeln die Geräte den TK über eine Nicht-Bluetooth-Verbindung wie NFC aus. Die Sicherheit hängt hier vom OOB-Protokoll ab.

Als Nächstes berechnen beide Geräte aus dem TK und einer eigenen Zufallszahl einen Bestätigungswert. Die Zufallszahlen und Bestätigungswerte tauschen beide Kommunikationspartner untereinander aus und verifizieren damit den Schlüsselaustausch. Anschließend leiten sie aus TK und Zufallszahlen einen 128 Bit langen Short Term Key (STK) ab, der der sicheren Schlüsselverteilung eines 128-Bit-LTK (Long Term Key), weiterer kryptografischer Schlüssel und Parameter dient.

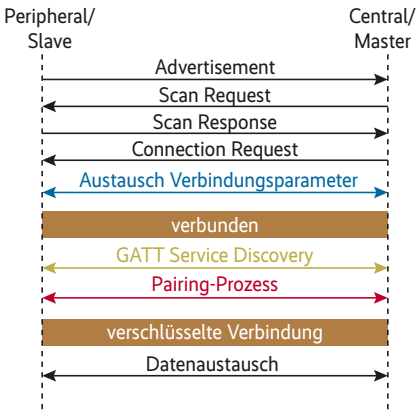


Der BLE-Protokollstack verteilt die Layer und damit die Zuständigkeiten auf Host und Controller (Abb. 1).

Solche Verfahren weisen selbstredend Schwachstellen auf. Kann ein Angreifer den Pairing-Prozess mitschneiden, bieten weder das Just-Works- noch das Passkey-Entry-Verfahren ausreichende Sicherheit, denn weder ein bekannter TK noch eine 20-Bit-Entropie bilden ein ernst zu nehmendes Hindernis.

Aus diesem Grund ersetzen die Secure Connections obige Algorithmen durch ein Elliptic-Curve-Diffie-Hellman-Verfahren. Dort generieren beide Parteien zunächst ein asymmetrisches Schlüsselpaar und tauschen die öffentlichen Schlüssel zu Beginn des Pairing-Prozesses aus. Secure Connections kennen vier Verfahren.

Just Works funktioniert hier anders: Hier generieren beide Geräte zunächst eine Nonce (Number used once) und tauschen sie aus. Das nicht initiierende Gerät berechnet dann aus den öffentlichen Schlüsseln und der eigenen Nonce einen Bestätigungswert und sendet diesen an den Kommunikationspartner. Den Bestätigungswert validiert der Initiator der Verbindung anhand der fremden Nonce.



Der Verbindungsaufbau zwischen Master und Slave respektive Central und Peripheral umfasst mehrere Phasen (Abb. 2).

Neu ist das Verfahren Numeric Comparison: Es arbeitet wie Just Works, zusätzlich berechnen beide Geräte aber aus Noncen und öffentlichen Schlüsseln eine sechsstellige Zahl und zeigen sie an. Der Nutzer schließt den Prozess ab, wenn er die Übereinstimmung beider Werte bestätigt.

Beim Passkey Entry hingegen muss der Anwender eine sechsstellige Zahl in beiden Geräten eingeben. Zusätzlich generieren diese eigene Noncen und tauschen sie aus. Aus Noncen und Nutzereingabe berechnen sie dann die Bestätigungswerte, tauschen sie aus und validieren sie. Lediglich das Out-of-Band-Verfahren ist vergleichbar mit dem des Legacy Pairing.

Wiedererkennen, aber nicht verfolgen

Anders als beim Legacy Pairing gewinnen die Geräte bei Secure Connections den LTK anschließend direkt aus den Noncen und den zusätzlich generierten Zufallszahlen. Den LTK verwenden sie im weiteren Verlauf des Pairing-Prozesses für den Austausch der weiteren kryptografischen Parameter und Schlüssel.

Der LTK dient, wenn angefordert, dem Bonding. Das bedeutet vereinfacht, dass die Geräte den ausgehandelten LTK speichern und beim nächsten Verbindungsaufbau zur gegenseitigen Authentifizierung nutzen.

Wie die Geräte die in der Pairing-Phase generierten Schlüssel verwenden, hängt außerdem von den Sicherheitsmodi ab. Die Verschlüsselung einer Verbindung legt Sicherheitsmodus 1 fest, der vier Level kennt:

- Level 1: keine Verschlüsselung und Authentifizierung;
- Level 2: Pairing ohne Authentifizierung mit Verschlüsselung;
- Level 3: Pairing mit Authentifizierung und Verschlüsselung;

Der BLE-Adapter Ubertooth One bildet die Basis für viele Projekte (Abb. 3).

Listing 1: Ubertooth One einrichten

```
wget https://github.com/greatscottgadgets/
ubertooth/releases/download/2018-12-R1/
ubertooth-2018-12-R1.tar.xz
tar xf ubertooth-2018-12-R1.tar.xz
cd ubertooth-2018-12-R1/host
mkdir build
cd build
cmake ..
make
sudo make install
```

No.	Time	Protocol	Length	Info
173	2020-06-16 15:03:46,234530747	LE LL	63	ADV_IND
174	2020-06-16 15:03:47,242108447	LE LL	63	ADV_IND
175	2020-06-16 15:03:48,250422547	LE LL	63	ADV_IND
176	2020-06-16 15:03:49,254352447	LE LL	63	ADV_IND
177	2020-06-16 15:03:50,256570247	LE LL	63	ADV_IND
178	2020-06-16 15:03:51,264436547	LE LL	63	ADV_IND
179	2020-06-16 15:03:51,264907047	LE LL	45	SCAN_REQ
180	2020-06-16 15:03:51,265235547	LE LL	39	SCAN_RSP
181	2020-06-16 15:03:52,270717847	LE LL	63	ADV_IND
182	2020-06-16 15:03:52,271188747	LE LL	67	CONNECT_REQ
183	2020-06-16 15:03:52,274981947	LE LL	33	Empty PDU
184	2020-06-16 15:03:52,275213447	LE LL	33	Empty PDU
185	2020-06-16 15:03:52,311232047	LE LL	42	Control Opcode: LL_FEATURE_REQ
186	2020-06-16 15:03:52,311535547	LE LL	33	Empty PDU
187	2020-06-16 15:03:52,347482447	LE LL	33	Empty PDU
188	2020-06-16 15:03:52,347713747	LE LL	42	Control Opcode: LL_FEATURE_RSP
189	2020-06-16 15:03:52,383732747	LE LL	39	Control Opcode: LL_VERSION_IND
190	2020-06-16 15:03:52,384013047	LE LL	33	Empty PDU
191	2020-06-16 15:03:52,419983147	LE LL	33	Empty PDU
192	2020-06-16 15:03:52,456233447	LE LL	33	Empty PDU
193	2020-06-16 15:03:52,456465547	LE LL	39	Control Opcode: LL_VERSION_IND

Die Paketanalyse in Wireshark offenbart die Details des BLE-Verbindungsaufbaus (Abb. 4).

- Level 4: Pairing mit LE Secure Connections als Schlüsselübertragungsmethode.

Der Sicherheitsmodus 2 entscheidet hingegen über den Einsatz von Signaturen:

- Level 1: Pairing ohne Authentifizierung mit Signatur;
- Level 2: Pairing mit Authentifizierung und Signatur.

Beide Sicherheitsmodi lassen sich kombinieren, um etwa eine verschlüsselte Verbindung mit Signatur einzusetzen. Zum Verschlüsseln und Authentifizieren spezifiziert BLE die als sicher geltende AES-128-Blockverschlüsselung und AES-CMAC.

Bluetooth und BLE nutzen zur eindeutigen Identifizierung von Geräten eine 48 Bit lange BD_ADDR (Bluetooth Device Address), ähnlich der MAC-Adresse von Netzwerkkarten. Da man die Geräte über diese BD_ADDR nachverfolgen und damit Bewegungsprofile ihrer Besitzer erstellen kann, führte der Standard BLE 4.2 die Funktion LE Privacy ein. Ist sie aktiviert, sendet ein Gerät nicht mehr die hart codierte Adresse, sondern eine zufällig generierte und verhindert dadurch ein Tracking. Damit die bereits verbundenen Geräte es identifizieren können, verwendet es den Identity Resolving Key (IRK), durch den sich aus der zufälligen Adresse die BD_ADDR ableiten lässt. Ihn tauschen die Devices in der letzten Phase des Pairing-Prozesses ebenfalls aus.

Eine große Werkzeugauswahl

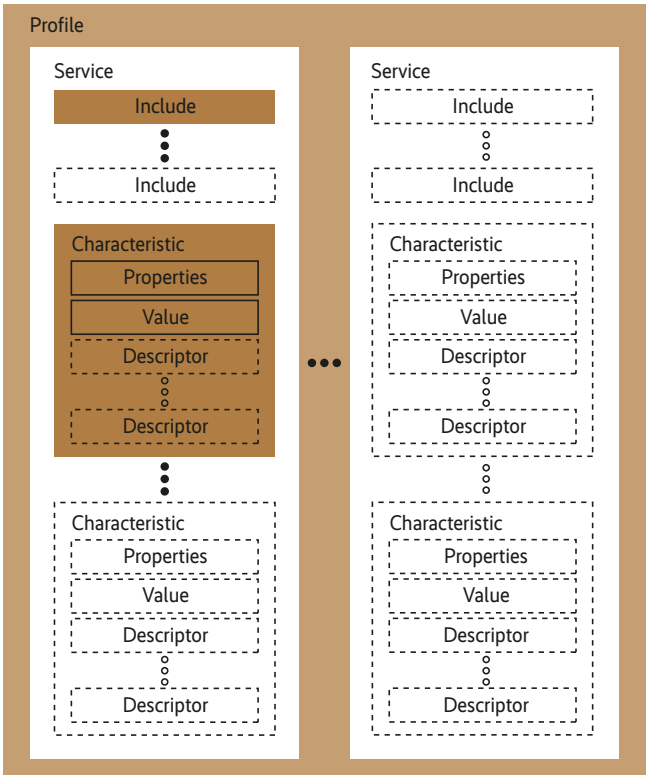
Wer drahtlose Verbindungen hacken will, kommt um das Mitschneiden der Daten nicht herum. Das Adaptive Frequency Hopping, das Wechseln des Datenkanals während der Übertragung, erschwert das zwar, doch mit den richtigen Werkzeugen ist diese Hürde leicht zu nehmen. Auch ohne Profi-Equipment, das alle Kanäle parallel aufzeichnet, erhält man bereits verwertbare Ergebnisse. Günstige Alternativen gibt es viele.

Erste Wahl ist der Onboard-Bluetooth-Chip eines Computers kombiniert mit geeigneter Software. Der Linux-Bluetooth-Stack bluez bietet bereits eine Vielzahl integrierter Werkzeuge wie hcitool, hciconfig und gatttool.

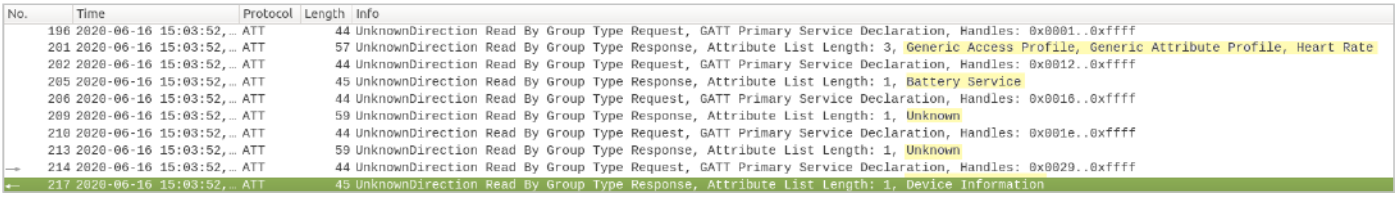
Zusätzlich oder alternativ kann sich ein externer Bluetooth-Adapter lohnen, der über mehr Funktionen oder eine stärkere Antenne verfügt. Einer der bekanntesten entstammt dem Projekt Ubertooth (alle Projekte siehe ix.de/z8zr). Es beinhaltet Hardware, Firmware und den Code für das Hostsystem. Die aktuelle Hardware ist Ubertooth One, die als Entwicklungsplattform dient (siehe Abbildung 3). Entstanden sind etwa der Spektrum-Analyser, Bluetooth Classic und Low Energy Sniffing. Die zugehörige Software ist für Linux- und macOS-Systeme verfügbar. Praktischerweise lassen sich die mitgeschnittenen Pakete direkt in Wireshark analysieren.

Eine mit unterschiedlichen Bluetooth-Adaptern kompatible Alternative für unixoide Systeme bildet die Software btlejack. Entwickelt wurde sie ursprünglich für den Einsatz mit MICRO:BIT-Controllern, die Kindern den Umgang mit Technik und Programmierung näherbringen sollen. Richtig programmiert sind sie allerdings auch gut als BLE-Sniffer einsetzbar. btlejack ermöglicht das automatische Einrichten der Controller und das Belauschen, Blockieren und Hijacken von BLE-Verbindungen.

Viele Analysemöglichkeiten für Windows, Linux und macOS bieten die BLE-Produkte von Nordic Semiconductor. Angriffe wie Man-in-the-Middle und Replay lassen sich zudem mit den Node.js-



Ein GATT-Profil besteht aus einem oder mehreren Services, die wiederum Characteristics und andere Services enthalten (Abb. 5).



No.	Time	Protocol	Length	Info
196	2020-06-16 15:03:52,…	ATT	44	UnknownDirection Read By Group Type Request, GATT Primary Service Declaration, Handles: 0x0001..0xffff
201	2020-06-16 15:03:52,…	ATT	57	UnknownDirection Read By Group Type Response, Attribute List Length: 3, Generic Access Profile, Generic Attribute Profile, Heart Rate
202	2020-06-16 15:03:52,…	ATT	44	UnknownDirection Read By Group Type Request, GATT Primary Service Declaration, Handles: 0x0012..0xffff
205	2020-06-16 15:03:52,…	ATT	45	UnknownDirection Read By Group Type Response, Attribute List Length: 1, Battery Service
206	2020-06-16 15:03:52,…	ATT	44	UnknownDirection Read By Group Type Request, GATT Primary Service Declaration, Handles: 0x0016..0xffff
209	2020-06-16 15:03:52,…	ATT	59	UnknownDirection Read By Group Type Response, Attribute List Length: 1, Unknown
210	2020-06-16 15:03:52,…	ATT	44	UnknownDirection Read By Group Type Request, GATT Primary Service Declaration, Handles: 0x001e..0xffff
213	2020-06-16 15:03:52,…	ATT	59	UnknownDirection Read By Group Type Response, Attribute List Length: 1, Unknown
214	2020-06-16 15:03:52,…	ATT	44	UnknownDirection Read By Group Type Request, GATT Primary Service Declaration, Handles: 0x0029..0xffff
217	2020-06-16 15:03:52,…	ATT	45	UnknownDirection Read By Group Type Response, Attribute List Length: 1, Device Information

Wireshark legt die verwendeten Services offen (Abb. 6).

Programmen BTLEjuice und GATTacker durchführen.

Naheliegend ist auch der Einsatz eines Smartphones. Android- und iOS-Apps wie LightBlue oder nRF Connect erlauben das Analysieren von BLE-Verbindungen und die Interaktion mit Peripheral-Geräten, aber auch Replay-Angriffe und das Klonen eines Geräts.

Die Bluetooth-Kommunikation mitschneiden

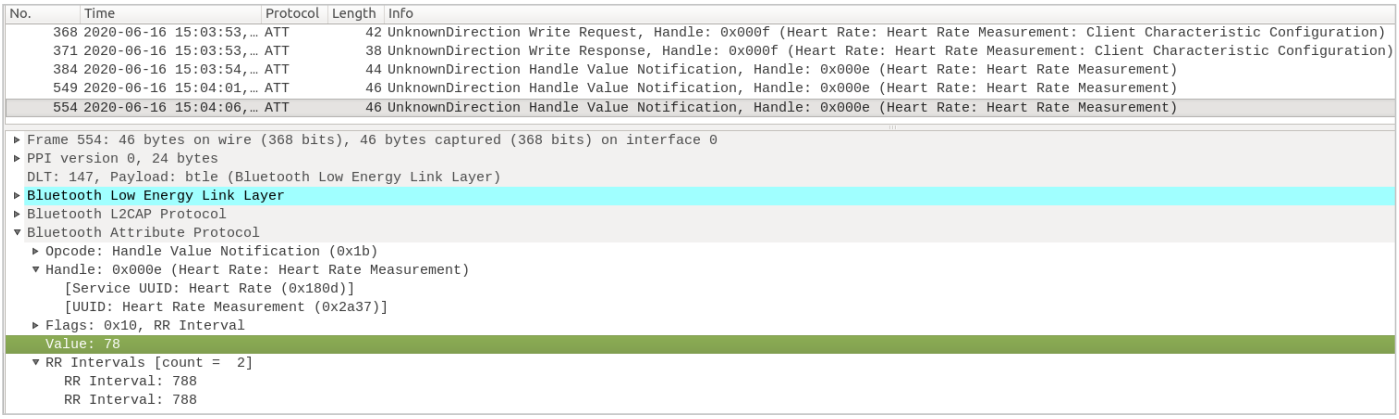
Grundsätzlich lässt sich das Belauschen drahtloser Kommunikation nicht verhindern. Ein solches Eavesdropping zählt zu den passiven Angriffen, da außer dem Mitschneiden der Nachrichten keine weitere Interaktion stattfindet. Als Beispiel dient eine mittelpreisige Pulsuhr. Solche kleinen, mehr oder weniger smarten Geräte analysieren meist die Herzfrequenz oder zählen die Schritte des Besitzers. Fortgeschrittenere Varianten bieten zusätzliche Funktionen an.

```
▸ Frame 252: 53 bytes on wire (424 bits), 53 bytes captured (424 bits) on interface 0
▸ PPI version 0, 24 bytes
  DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
▸ Bluetooth Low Energy Link Layer
▸ Bluetooth L2CAP Protocol
▾ Bluetooth Attribute Protocol
  ▸ Opcode: Read By Type Response (0x09)
    Length: 7
  ▾ Attribute Data, Handle: 0x000d, Characteristic Handle: 0x000e, UUID: Heart Rate Measurement
    ▸ Handle: 0x000d (Heart Rate: GATT Characteristic Declaration)
    ▸ Characteristic Properties: 0x10, Notify
    ▸ Characteristic Value Handle: 0x000e (Heart Rate: Heart Rate Measurement)
      UUID: Heart Rate Measurement (0x2a37)
  ▾ Attribute Data, Handle: 0x0010, Characteristic Handle: 0x0011, UUID: Body Sensor Location
    ▸ Handle: 0x0010 (Heart Rate: Heart Rate Measurement: GATT Characteristic Declaration)
    ▸ Characteristic Properties: 0x02, Read
    ▸ Characteristic Value Handle: 0x0011 (Heart Rate: Body Sensor Location)
      UUID: Body Sensor Location (0x2a38)
    [UUID: GATT Characteristic Declaration (0x2803)]
```

In den Serviceattributen finden sich die Characteristics Heart Rate Measurement und Body Sensor Location (Abb. 7).

Die verwendete Pulsuhr misst lediglich den Puls und sendet ihn an die zugehörige App, die so zur Analyse der eigenen sportlichen Leistung dient. Zusätzlich zeigt die Pulsuhr per LED-Farbcodierung an, ob der Anwender am Rande seiner Leistungsfähigkeit ist oder noch einen Zahn zulegen sollte.

Zur Analyse dient ein Ubertooth One sowie die zugehörige Software von Great-ScottGadgets. Die Einrichtung unter Linux zeigt Listing 1. Es ist darauf zu



Vertrauliche Inhalte wie hier der Puls des Anwenders finden sich in den unverschlüsselten Paketen (Abb. 8).

achten, den Ubertooth One nicht ohne Antenne zu betreiben, da der Bluetooth-Dongle sonst Schaden nehmen kann.

Mit dem Kommando `ubertooth-btle -f` hört man die BLE-Verbindungen in der Umgebung ab. Da die Software die Pakete mehrerer Verbindungen finden kann, empfiehlt es sich, explizit nach denen des eigenen Geräts zu suchen und ihre Ausgabe an Wireshark umzuleiten:

```
mkfifo /tmp/pipe
ubertooth-btle -f -c /tmp/pipe -t
XX:XX:XX:XX:XX:XX
```

Der Befehl scannt nach Paketen mit der BD_ADDR `XX:XX:XX:XX:XX:XX` und übergibt diese der zuvor generierten FIFO-Pipe (First In, First Out), die man in Wireshark als Datenquelle angibt. Die BD_ADDR des eigenen Geräts lässt sich meist in der zugehörigen App auf dem Smartphone einsehen. Alternativ helfen Bluetooth-Scanner-Apps wie nRFConnect oder LightBlue weiter.

Nützliche Verbindungsinformationen finden

Für die Analyse sind zunächst die Pakete des ersten Verbindungsaufbaus von Interesse, da sie Informationen über die verwendete Verbindung enthalten, etwa die BLE-Version der Uhr (siehe Abbildung 4).

Zu Beginn sieht man die Advertisement-Nachrichten `ADV_IND` des Peripherals, in diesem Fall der Pulsuhr, gesendet auf den drei Kanälen. Die Pakete des weiteren Verbindungsaufbaus verraten, dass die Pulsuhr BLE-Version 4.0 verwendet und LE-Privacy deshalb nicht verfügbar ist. Das heißt, die Uhr sendet die fest im Controller codierte BD_ADDR.

Die Empty-PDU-Pakete können als eine Art Acknowledge-Nachrichten verstanden werden und sind für die weitere Analyse nicht relevant. Es fällt aber auf, dass ein für die Verschlüsselung notwendiger Pairing-Request fehlt. Da aber ohne Pairing keine kryptografischen Parameter für den LTK ausgetauscht werden, ist eine Verschlüsselung mit den erwähnten Mechanismen nicht möglich. Es liegt also nahe, dass das Gerät die Daten ungesichert überträgt.

Um die Daten interpretieren zu können, muss man das grundlegende Prinzip hinter den GATT-Profilen verstehen. Jedes GATT-Profil enthält einen oder mehrere Services, die für die Funktion des Geräts verwendet werden, wie das Auslesen des Batterieladestatus oder verfügbarer Sensorwerte (siehe Abbildung 5). Jeder Service besteht wiederum aus einer oder mehreren Characteristics und anderen zugehörigen GATT-Services. Eine Characteristic beschreibt ein Attribut und seine Verwendung. Mögliche Zugriffsarten sind etwa Read, Write oder Notify.

Unterschieden wird zwischen herstellerspezifischen und allgemeinen Services, die die Bluetooth SIG in ihren Assigned Numbers definiert hat (siehe ix.de/z8zr). Zu erwarten ist, dass die Pulsuhr den Heart Rate GATT Service mit der ihm zugehörigen UUID (Universal Unique Identifier) 0x180d implementiert, um den Puls für den GATT-Client zur Verfügung zu stellen. Es ist aber auch möglich, dass weitere Services implementiert sind oder der Hersteller einen eigenen Service für die Übertragung verwendet. Ein genauer Blick lohnt sich also.

Der Client fragt die Services mit Read-by-Group-Response-Paketen ab (siehe Abbildung 6). Die genauere Analyse bestätigt die Vermutung. Neben dem Heart Rate Service enthält das GATT-Profil der Pulsuhr sechs weitere Services:

- Generic Access Profile (UUID 0x1800)
- Generic Attribute Profile (UUID 0x1801)
- Battery Service (UUID 0x180f)
- Unknown – herstellerspezifisch
- Unknown – herstellerspezifisch
- Device Information (UUID 0x180a)

Generic Access Profile, Generic Attribute Profile, Battery Service und Device Information Service sind, wie der Heart Rate Service, in den Assigned Numbers definiert und deshalb „well-known". Über den Zweck der beiden herstellerspezifischen Services lässt sich ohne weitere Analyse keine Aussage treffen. Für die Untersuchung sind sie aber nicht von Bedeutung.

Vertrauliche Informationen öffentlich hinausposaunt

Die nähere Betrachtung des Heart Rate Service bringt zwei definierte Characteristics zum Vorschein: Heart Rate Measurement und Body Sensor Location (siehe Abbildung 7). Der GATT-Server propagiert das Puls-Attribut also per Notify. Eine solche Benachrichtigung bedeutet, dass der Server den Wert des Attributs, in diesem Fall den Puls, bei Veränderung an den registrierten Empfänger sendet. Der Client, das Smartphone, registriert sich für den Heart Rate Measurement Service, daraufhin beginnt der GATT-Server der Pulsuhr den Puls an das Smartphone zu übertragen – wie zuvor vermutet unverschlüsselt und für jeden abhörbar (siehe Abbildung 8).

Der Puls eines Menschen lässt sich für weit mehr als die Analyse des persönlichen Trainings nutzen. In Kombination

Listing 2: Interaktive Konsole von gatttool

```
[XX:XX:XX:XX:XX:XX][LE]> connect
Attempting to connect to XX:XX:XX:XX:XX:XX
Connection successful
[XX:XX:XX:XX:XX:XX][LE]> char-write-cmd 0x0007 56ff000000f0aa
[XX:XX:XX:XX:XX:XX][LE]> char-write-cmd 0x0007 5600ff0000f0aa
[XX:XX:XX:XX:XX:XX][LE]> char-write-cmd 0x0007 560000ff00f0aa
```

Zu finden ist in den Paketinformationen diesmal ein einzelner herstellerspezifischer Service, über den sich Werte an den Server übergeben lassen (Abb. 9).

```
▸ Frame 2133: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface 1
▸ PPI version 0, 24 bytes
  DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
▸ Bluetooth Low Energy Link Layer
▸ Bluetooth L2CAP Protocol
▾ Bluetooth Attribute Protocol
  ▸ Opcode: Read By Type Response (0x09)
    Length: 7
  ▾ Attribute Data, Handle: 0x0006, Characteristic Handle: 0x0007, UUID: Unknown
    ▸ Handle: 0x0006 (Unknown: GATT Characteristic Declaration)
    ▸ Characteristic Properties: 0x04, Write without Response
    ▸ Characteristic Value Handle: 0x0007 (Unknown: Unknown)
      UUID: Unknown (0xffd9)
    [UUID: GATT Characteristic Declaration (0x2803)]
    [Request in Frame: 2130]
```

Im Wert des Write Request versteckt sich ein RGB-Farbcode, den der Farbwechselbefehl der Lampe übergibt (Abb. 10).

```
No.   Time                       Protocol Length Info
 3937 2020-08-03 16:48:57,… ATT        47 UnknownDirection Write Request, Handle: 0x0007 (Unknown: Unknown)
 3980 2020-08-03 16:48:57,… ATT        47 UnknownDirection Write Request, Handle: 0x0007 (Unknown: Unknown)
 3983 2020-08-03 16:48:57,… ATT        38 UnknownDirection Write Response, Handle: 0x0007 (Unknown: Unknown)
 4117 2020-08-03 16:48:58,… ATT        47 UnknownDirection Write Request, Handle: 0x0007 (Unknown: Unknown)
 4118 2020-08-03 16:48:58,… ATT        47 UnknownDirection Write Request, Handle: 0x0007 (Unknown: Unknown)
 4119 2020-08-03 16:48:58,… ATT        38 UnknownDirection Write Response, Handle: 0x0007 (Unknown: Unknown)
 4131 2020-08-03 16:48:58,… ATT        47 UnknownDirection Write Request, Handle: 0x0007 (Unknown: Unknown)
 4134 2020-08-03 16:48:58,… ATT        38 UnknownDirection Write Response, Handle: 0x0007 (Unknown: Unknown)
 4135 2020-08-03 16:48:58,… ATT        47 UnknownDirection Write Request, Handle: 0x0007 (Unknown: Unknown)

▸ Frame 3937: 47 bytes on wire (376 bits), 47 bytes captured (376 bits) on interface 1
▸ PPI version 0, 24 bytes
  DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
▸ Bluetooth Low Energy Link Layer
▸ Bluetooth L2CAP Protocol
▾ Bluetooth Attribute Protocol
  ▸ Opcode: Write Request (0x12)
  ▸ Handle: 0x0007 (Unknown: Unknown)
    Value: 560026ff00f0aa
```

mit der eigenen Erscheinung, dem geschätzten Alter und Gewicht kann er Auskunft geben über den Gesundheitszustand oder die aktuelle Gemütslage. Dabei handelt es sich bei dem Demonstrationsobjekt um einen sehr einfachen Fitnesstracker. Spätestens wenn man weitere Funktionen wie die Analyse des Schlafverhaltens in Betracht zieht, wird die Notwendigkeit entsprechender Sicherheitsmaßnahmen für die BLE-Kommunikation deutlich.

Statt nur passiv abzuhören, kann der Angreifer auch aktiv in die Verbindung eingreifen. Dazu gehört etwa das Wiedereinspielen zuvor aufgenommener Nachrichten, das Senden modifizierter oder eigener Daten oder Man-in-the-Middle-Angriffe.

Bluetooth-Geräte manipulieren

Opfer eines aktiven Angriffs soll eine „smarte“ Glühbirne im unteren Preissegment sein. Farbe, Helligkeit und Farbabfolge lassen sich über eine App steuern. Wie zuvor werden zunächst, wieder mit dem Ubertooth One, Verbindungseigenschaften und die verfügbaren GATT-Services analysiert.

Aus den Paketen des Verbindungsaufbaus geht hervor, dass die Glühbirne BLE-Version 4.0 verwendet, also ebenfalls kein LE-Privacy-Feature beherrscht. Auch einen Pairing-Prozess sucht man vergeblich. Ob das Tracking über die BD_ADDR einer üblicherweise fest installierten Glühbirne unbedingt zu unterbinden ist, sei dahingestellt. Fehlende Sicherheitsmaßnahmen wie die Verschlüsselung sind jedoch kritisch zu betrachten.

Wireshark offenbart diesmal, dass die Glühbirne auf der GATT-Schicht genau einen Service anbietet, und zwar einen herstellerspezifischen Service mit der Eigenschaft Write without Response (siehe Abbildung 9). Über solch einen modifizierbaren Wert werden sehr wahrscheinlich die Farb- und Helligkeitseigenschaften der Lampe gesetzt.

Weiter kommt man, wenn man die Pakete analysiert, die die App während der Konfiguration der Lampe sendet. Bei der Beobachtung mehrerer Farbwechsel stellt sich heraus, dass es sich bei den Bytes 2 bis 4 des Value-Feldes um einen RGB-Farbcode handelt (siehe Abbildung 10).

Mithilfe dieser Information kann nun ein Angriff stattfinden. Um zu demonstrieren, wie unkompliziert das Hard- und Software-Set-up für einen solchen Angriff sein kann, soll das Linux-Kommandozeilenprogramm gatttool in Kombination mit dem Bluetooth-Chip des PCs zum Einsatz kommen. Das Kommando

```
gatttool -b XX:XX:XX:XX:XX:XX -I
```

startet gatttool im interaktiven Modus, während `XX:XX:XX:XX:XX:XX` die BD_ADDR der Glühbirne ist. Zunächst ist mit `connect` eine Verbindung zur Glühbirne aufzubauen, anschließend setzt der Befehl `char-write-cmd` den Wert des Attributs (siehe Listing 2).

Die drei Kommandos setzen die Farbe der Lampe nacheinander auf Rot (0x ff0000), Grün (0x00ff00) und Blau (0x0000ff). Die Glühbirne lässt sich also auch ohne App über die Kommandozeile steuern – mit einer entsprechenden Antenne kann ein Angreifer das außerhalb der Wohnung bewerkstelligen.

Das Steuern der eigenen Hauselektronik durch unautorisierte Personen ist schon inakzeptabel. Bei über BLE gesteuerten Türschlössern oder Herzschrittmachern liegt die dringende Notwendigkeit geeigneter Sicherheitsmechanismen aber auf der Hand.

Fazit

Obwohl Bluetooth Low Energy prinzipiell den Einsatz von Sicherheitsmaßnahmen spezifiziert, fehlen sie in der Praxis oft. Die Analyse der beiden BLE-Geräte bestätigte das. Zudem ließ sich zeigen, dass man keine teuren Profigeräte benötigt, um die BLE-Übertragung mitzuschneiden und zu analysieren. Mit gängigen Linux-Tools ließ sich zudem ein erfolgreicher Angriff auf eine „smarte“ Glühbirne demonstrieren. (sun@ix.de)

Quellen

[1] Alexander Poth; Replay-Angriffe über Funk; Wieder eingespielt; *iX* 12/2020, S. 142

[2] Alle Projekte: ix.de/z8zr

Sarah Mader

ist IT-Security-Analystin bei NSIDE Attack Logic. Zu ihren Schwerpunkten zählen Hardwareschnittstellen, besonders im Embedded-Bereich.