

Bluetooth sniffing

Matter's Bluetooth sniffing tests included various tools designed to reveal information about the Matter device.

Hcitrust:

First, the tool hcitrust was used to probe the MAC addresses that could be found in the area. The commands hcitrust scan or hcitrust lescan can be used to differentiate between Bluetooth and Bluetooth Low Energy devices (Miller & Estrella, 2018).

```
michael@michael-testenv:~$ hcitrust scan
Scanning ...
1C:B7:96:38:8F:67      Smartphone
michael@michael-testenv:~$ hcitrust inq
Inquiring ...
1C:B7:96:38:8F:67      clock offset: 0x7718      class: 0x5a020c
michael@michael-testenv:~$ sudo hcitrust lescan
[sudo] password for michael:
LE Scan ...
00:C3:F4:F7:4A:8F (unknown)
EA:B7:87:B4:1F:76 (unknown)
79:91:C4:41:C9:C0 (unknown)
79:91:C4:41:C9:C0 (unknown)
BC:14:85:B8:E3:27 (unknown)
D3:5C:D4:6B:A3:71 (unknown)
D3:5C:D4:6B:A3:71 (unknown)
^Cmichael@michael-testenv:~$
```

Figure 1: hcitrust Bluetooth and BLE device scan

The distinction makes sense because the tools used later do not differentiate between Bluetooth and BLE. The test can also be carried out quickly and easily repeated, so that it was found that the Matter device, when unpaired, is assigned a new randomised MAC address every time it is switched on.

```
michael@michael-testenv:~$ sudo hcitrust lescan
LE Scan ...
6F:4A:B6:FE:C8:B8 (unknown)
6F:4A:B6:FE:C8:B8 (unknown)
00:C3:F4:F7:4A:8F (unknown)
F2:A4:82:14:D5:59 (unknown)
E5:F7:26:CB:F9:2F (unknown)
E5:F7:26:CB:F9:2F (unknown)
^Cmichael@michael-testenv:~$ sudo hcitrust lescan
LE Scan ...
00:C3:F4:F7:4A:8F (unknown)
6F:4A:B6:FE:C8:B8 (unknown)
6F:4A:B6:FE:C8:B8 (unknown)
F2:A4:82:14:D5:59 (unknown)
F9:25:23:3A:C2:2F (unknown)
F9:25:23:3A:C2:2F (unknown)
^Cmichael@michael-testenv:~$ sudo hcitrust lescan
LE Scan ...
00:C3:F4:F7:4A:8F (unknown)
6F:4A:B6:FE:C8:B8 (unknown)
6F:4A:B6:FE:C8:B8 (unknown)
F2:A4:82:14:D5:59 (unknown)
D5:62:7F:6B:83:12 (unknown)
D5:62:7F:6B:83:12 (unknown)
^Cmichael@michael-testenv:~$ sudo hcitrust lescan
LE Scan ...
00:C3:F4:F7:4A:8F (unknown)
6F:4A:B6:FE:C8:B8 (unknown)
6F:4A:B6:FE:C8:B8 (unknown)
F2:A4:82:14:D5:59 (unknown)
D1:2A:25:D1:71:46 (unknown)
D1:2A:25:D1:71:46 (unknown)
michael@michael-testenv:~$
```

Figure 2: Changing Matter MAC address

Sdptool:

A further examination of Bluetooth devices was carried out using the sdptool, which can determine information about the device and its functions (Krasnyansky, N.D.). For this, however, it is necessary that the MAC address of the device to be examined is known, so that the MAC addresses had to be probed beforehand using the hcitool. The application can be executed using the command ‘sdptool browse [MAC address]’.

An excerpt of the functions found for a smartphone as a test object can be seen in the following figure.

The tool can therefore be used to determine the various functions of the device, with the information from which you can not only gain knowledge about the corresponding device, such as what device it is (e.g. camera, smartphone, smart TV, etc.). This information can also be used to carry out further hacking attacks tailored to the device. It should be noted that during the test it was not possible to examine BLE devices for their functions using this tool.

```
michael@michael-testenv:~$ sdptool browse 1C:B7:96:38:8F:67
Browsing 1C:B7:96:38:8F:67 ...
Service RecHandle: 0x10000
Service Class ID List:
  "Generic Attribute" (0x1801)
Protocol Descriptor List:
  "L2CAP" (0x0100)
    PSM: 31
    "ATT" (0x0007)
      uint16: 0x0001
      uint16: 0x0003
Service RecHandle: 0x10001
Service Class ID List:
  "Generic Access" (0x1800)
Protocol Descriptor List:
  "L2CAP" (0x0100)
    PSM: 31
    "ATT" (0x0007)
      uint16: 0x0014
      uint16: 0x001a
Service Name: BrManagerInsecure
Service RecHandle: 0x1000f
Service Class ID List:
  UUID 128: 8ce255c0-200a-11e9-ac64-0800200c9a56
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 4
Service Name: Headset Gateway
Service RecHandle: 0x10010
Service Class ID List:
  "Headset Audio Gateway" (0x1112)
  "Generic Audio" (0x1203)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 2
Profile Descriptor List:
  "Headset" (0x1108)
    Version: 0x0102
Service Name: Handsfree Gateway
Service RecHandle: 0x10011
Service Class ID List:
  "Handsfree Audio Gateway" (0x111f)
  "Generic Audio" (0x1203)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 3
Profile Descriptor List:
  "Handsfree" (0x111e)
    Version: 0x0106
Service Name: AV Remote Control Target
Service RecHandle: 0x10012
Service Class ID List:
  "AV Remote Target" (0x116c)
Protocol Descriptor List:
  "L2CAP" (0x0100)
    PSM: 25
    "AVCTP" (0x0017)
      uint16: 0x0103
Profile Descriptor List:
  "AV Remote" (0x110e)
    Version: 0x0104
Service Name: Advanced Audio Source
Service RecHandle: 0x10013
Service Class ID List:
  "Audio Source" (0x110a)
Protocol Descriptor List:
  "L2CAP" (0x0100)
    PSM: 25
    "AVDTP" (0x0019)
      uint16: 0x0103
Profile Descriptor List:
  "Advanced Audio" (0x110d)
    Version: 0x0103
```

Figure 3: Device information via sdptool (excerpt)

Bettercap

Another approach to determine information and potential vulnerabilities of BLE devices was pursued using the bettercap tool. The tool bettercap offers a variety of functions to sniff Wi-Fi and BLE networks (Margatelli et al., 2021).

As part of the tests carried out with bettercap, it was found that the most stable use of bettercap for BLE sniffing could be achieved with a Docker image. The image was then downloaded from the bettercap website and started with the command 'docker run -it -privileged -net=host bettercap/bettercap'. After bettercap was started, the tool automatically searched for discoverable BLE devices.

```

michael@michael-testenv: $ sudo docker run -it --privileged --net=host bettercap/bettercap
bettercap v2.32.0 (built for linux amd64 with go1.16.4) [type 'help' for a list of commands]

192.168.0.0/24 > 192.168.0.168 » [10:35:27] [sys.log] [tnf] gateway monitor started ...
192.168.0.0/24 > 192.168.0.168 » ble.recon on
192.168.0.0/24 > 192.168.0.168 » [10:35:31] [ble.device.new] new BLE device detected as F5:56:B8:90:30:22 -30 dBm.
192.168.0.0/24 > 192.168.0.168 » [10:35:31] [ble.device.new] new BLE device detected as 69:ED:28:4C:DA:9A (Apple, Inc.) -55 dBm.
192.168.0.0/24 > 192.168.0.168 » [10:35:31] [ble.device.new] new BLE device detected as EF:78:CE:8A:FD:C8 (Apple, Inc.) -66 dBm.
192.168.0.0/24 > 192.168.0.168 » [10:35:32] [ble.device.new] new BLE device detected as D0:89:38:47:2A:AD (Apple, Inc.) -70 dBm.
192.168.0.0/24 > 192.168.0.168 » ble.show


```

RSSI ▲	MAC	Vendor	Flags	Connect	Seen
-35 dBm	f5:56:b8:90:30:22		BR/EDR Not Supported	✓	10:35:36
-55 dBm	69:ed:28:4c:da:9a	Apple, Inc.	LE + BR/EDR (controller), LE + BR/EDR (host)	✓	10:35:36
-63 dBm	d0:89:38:47:2a:ad	Apple, Inc.		✗	10:35:36
-70 dBm	ef:78:ce:8a:fd:c8	Apple, Inc.		✗	10:35:35

```

192.168.0.0/24 > 192.168.0.168 » ble.enum 69:ed:28:4c:da:9a
[10:35:47] [sys.log] [tnf] ble.recon connecting to 69:ed:28:4c:da:9a ...
192.168.0.0/24 > 192.168.0.168 »

```

Handles	Service > Characteristics	Properties	Data
0001 -> 0005 0003 0005	Generic Access (1800) Device Name (2a00) Appearance (2a01)	READ READ	iPad Generic Media Player
0006 -> 0009 0008	Generic Attribute (1801) Service Changed (2a05)	INDICATE	
000a -> 000e 000c 000e	Device Information (180a) Manufacturer Name String (2a29) Model Number String (2a24)	READ READ	Apple Inc. iPad8,9
000f -> 0013 0011	Apple Continuity Service (d0611e78bbb44591a5f8487910ae4366) 8667556c9a374c9184ed54ee27d90049	WRITE, NOTIFY, X	
0014 -> 0018 0016	9fa480e0496745429390d343dc5d04ae af0badb15b9943cd917aa77bc549e3cc	WRITE, NOTIFY, X	
0019 -> 001c 001b	Battery Service (180f) Battery Level (2a19)	READ, NOTIFY	insufficient authentication
001d -> 0022 001f 0022	Current Time Service (1805) Current Time (2a2b) Local Time Information (2a0f)	READ, NOTIFY READ	insufficient authentication insufficient authentication
0023 -> 002c 0025 0028 002b	Apple Notification Center Service (7905f431b5ce4e99a40f4b1e122d00d0) 69d1d8f345e149a898219bbdfdaad9d9 9fbf120d630142d98c5825e699a21dbd 22eac6e924d64bb5be44b36ace7c7bfb	WRITE, X NOTIFY NOTIFY	
002d -> 0038 002f 0033 0037	Apple Media Service (89d3502b0f36433a8ef4c502ad55f8dc) 9b3c81d857b14a8ab8df0e56f7ca51c2 2f7cabce808d411f9a0cbb92ba96c102 c6b2f38c23ab46d8a6aba3a870bbd5d7	WRITE, NOTIFY, X WRITE, NOTIFY, X READ, WRITE, X	insufficient authentication

```

192.168.0.0/24 > 192.168.0.168 »

```

Figure 4: iPad sniffing with bettercap

In the upper section of the figure it can be seen that the devices found are listed with the respective MAC address and, if identifiable, also the vendor. In addition, the dBm provides information about how strong the signal is. The signal strength can be used to estimate how far away the device is. However, further investigations into the signal strength have shown that this can only be assessed as an indication of the distance. Some devices inherently have a stronger or weaker signal strength. In addition, it can also be severely affected by sources of interference (electromagnetic devices, walls, water pipes, etc.). Using the 'ble.show' command, the devices found can be displayed in a table, whereby information can also be provided as to whether the respective device is connected to another device and whether it is a end-device (slave) or controller (master).

To get detailed information about a specific device, the command 'ble.enum [MAC]' can be used. The results of this research can be seen in the bottom table of the figure. Handles found, the functions of the device, as well as further information such as characteristics, properties and authorisations and other details are listed here. This is where things get particularly interesting, because possible vulnerabilities and attack vectors can be identified using the information about the handles' permissions. In addition, it can be determined whether a characteristic of the corresponding handle is encrypted or not. Finally, depending on the device, it is also possible to find out exactly which device it is in detail. The test with bettercap using the iPad as the test device revealed that it is such a device and also that it is the 8th or 9th generation of an iPad.

```
michael@michael-testenv:~$ sudo docker run -it --privileged --net=host bettercap/bettercap
bettercap v2.32.0 (built for linux amd64 with go1.16.4) [type 'help' for a list of commands]

192.168.0.0/24 > 192.168.0.168 » [10:31:35] [sys.log] [inf] gateway monitor started ...
192.168.0.0/24 > 192.168.0.168 » ble recon on
192.168.0.0/24 > 192.168.0.168 » [10:31:40] [ble.device.new] new BLE device detected as F5:56:B8:90:30:22 -31 dBm.
192.168.0.0/24 > 192.168.0.168 » [10:31:41] [ble.device.new] new BLE device detected as D0:89:38:47:2A:AD (Apple, Inc.) -68 dBm.
192.168.0.0/24 > 192.168.0.168 » [10:31:41] [ble.device.new] new BLE device detected as 40:B0:13:E0:EC:7D (Apple, Inc.) -68 dBm.
192.168.0.0/24 > 192.168.0.168 » [10:31:41] [ble.device.new] new BLE device detected as DB:82:35:DC:60:3F (Apple, Inc.) -68 dBm.
192.168.0.0/24 > 192.168.0.168 » ble.show
```

RSSI ▲	MAC	Vendor	Flags	Connect	Seen
-30 dBm	f5:56:b8:90:30:22		BR/EDR Not Supported	✓	19:31:44
-55 dBm	d0:89:38:47:2a:ad	Apple, Inc.	LE + BR/EDR (controller), LE + BR/EDR (host)	✗	19:31:43
-55 dBm	40:b0:13:e0:ec:7d	Apple, Inc.		✓	19:31:44
-67 dBm	db:82:35:dc:60:3f	Apple, Inc.		✗	19:31:43

```
192.168.0.0/24 > 192.168.0.168 » ble.enum f5:56:b8:90:30:22
[10:31:56] [sys.log] [inf] ble.recon connecting to f5:56:b8:90:30:22 ...
192.168.0.0/24 > 192.168.0.168 »
```

Handles	Service > Characteristics	Properties	Data
0001 -> 0005	Generic Access (1800)		
0003	Device Name (2a00)	READ	
0005	Appearance (2a01)	READ	MATTER-3840 Unknown
0006 -> 0009	Generic Attribute (1801)		
0008	Service Changed (2a05)	INDICATE	
000a -> ffff	fff6		
000c	18ee2ef5263d4559959f4f9c429f9d11	WRITE	
000e	18ee2ef5263d4559959f4f9c429f9d12	READ, INDICATE	

```
192.168.0.0/24 > 192.168.0.168 »
```

Figure 5: Matter device sniffing with bettercap

A similar test using the Matter device as a target revealed basically the same possible information available. Since the Matter device used is a device that was self flashed using a test application, the information received regarding the device name is rather cryptic. However, it can be clearly stated that the device is a Matter device (see name). Furthermore, the number 3840 can be found behind the name of the device, which is the discriminator of the device, which was assigned to the device during the flash process. While all tests carried out with bettercap did not lead to any detectable vulnerabilities, it can still be noted that possible vulnerabilities can be easily identified with bettercap, so that a responsible and thoughtful implementation of handles/functions of Bluetooth devices is of central importance.

Wireshark:

With Matter devices, the Bluetooth communication standard is used exclusively for commissioning and switches off automatically after the device has been successfully implemented into a Matter network. However, when putting a Matter device into operation, extremely sensitive information such as the access point (router) name and its password is exchanged. Therefore, a final attempt was made to record the packets exchanged when the Matter device was put into operation. The wireshark tool was used for this, which can listen to a variety of other standards in addition to Bluetooth (Wireshark Foundation, N.D.). In order to record Bluetooth packets with Wireshark, a Bluetooth sniffing device is required. As part of the project, the nRF52840 dongle from Nordic was used (Nordic, 2022). After the dongle was flashed with software also provided by Nordic and the necessary addons for Wireshark were added, Bluetooth packets sent in the area could be recorded.

3147	4525.7..	Master_0x5065451e	LE 1M	LE LL	6	17267µs	0	0	False	0	Control Opcode: LL_VERSION_IND
3148	4525.7..	Slave_0x5065451e	LE 1M	LE LL	9	151µs	0	1	True	0	Control Opcode: LL_SLAVE_FEATURE_REQ
3149	4525.7..	Master_0x5065451e	LE 1M	LE LL	0	150µs	1	1	False	0	Empty PDU
3150	4525.7..	Slave_0x5065451e	LE 1M	LE LL	0	151µs	1	0	False	0	Empty PDU
3151	4525.7..	Master_0x5065451e	LE 1M	LE LL	9	21609µs	0	0	False	1	Control Opcode: LL_FEATURE_RSP
3152	4525.7..	Slave_0x5065451e	LE 1M	LE LL	6	150µs	0	1	True	1	Control Opcode: LL_VERSION_IND
3153	4525.7..	Master_0x5065451e	LE 1M	LE LL	0	151µs	1	1	False	1	Empty PDU
3154	4525.7..	Slave_0x5065451e	LE 1M	LE LL	0	150µs	1	0	False	1	Empty PDU
3155	4525.8..	Master_0x5065451e	LE 1M	LE LL	3	21609µs	0	0	False	2	Control Opcode: LL_PHY_REQ
3156	4525.8..	Slave_0x5065451e	LE 1M	LE LL	0	151µs	0	1	False	2	Empty PDU
3157	4525.8..	Master_0x5065451e	LE 1M	LE LL	0	22166µs	1	1	False	3	Empty PDU
3158	4525.8..	Slave_0x5065451e	LE 1M	LE LL	3	151µs	1	0	True	3	Control Opcode: LL_PHY_RSP
3159	4525.8..	Master_0x5065451e	LE 1M	LE LL	0	150µs	0	0	False	3	Empty PDU
3160	4525.8..	Slave_0x5065451e	LE 1M	LE LL	0	151µs	0	1	False	3	Empty PDU
3161	4525.8..	Master_0x5065451e	LE 1M	LE LL	5	21704µs	1	1	False	4	Control Opcode: LL_PHY_UPDATE_IND

Figure 6: Sniffed initial Bluetooth packages

The figure shows the first exchanged packages when putting a Matter device into operation. The respective transmitter (master or slave) can be seen here, as well as the communication standard used for the packets, the time of transmission, the sequence number, the next sequence number to be expected, whether further information is available, the event counter and finally further information about the packets. A complete record of the captured packets when a Matter device is put into operation can be found separately. However, Wireshark or another program that can read .pcap files is required to read them. By selecting individual packages, their contents and other information can be read.


```

> Frame 4650: 48 bytes on wire (384 bits), 48 bytes captured (384 bits) on interface COM6-4.0, id 0
> nRF Sniffer for Bluetooth LE
> Bluetooth Low Energy Link Layer
> Bluetooth L2CAP Protocol
  Length: 72
  CID: Attribute Protocol (0x0004)
> Bluetooth Attribute Protocol
  Opcode: Handle Value Indication (0x1d)
  Handle: 0x000e (ZigBee Alliance: Unknown)
    [Service UUID: ZigBee Alliance (0xffff6)]
    [UUID: 18ee2ef5263d4559959f4f9c429f9d12]
    Value: 0520410000d40900c4e9b60d1b70be57707d30f834194277ae1b9503e1398a5dd2efefc7...

```

0000	48 00 04 00 1d 0e 00 05	20 41 00 00 d4 09 00 c4	H..... A.....
0010	e9 b6 0d 1b 70 be 57 70	7d 30 f8 34 19 42 77 aep-wp }0.4.Bw.
0020	1b 95 03 e1 39 8a 5d d2	ef ef c7 9b 82 51 19 c79.}.....Q....
0030	5c 0d 9d f8 97 42 c9 4a	72 5a 49 d5 22 1c cf 81B.} rZI.....
0040	b6 84 7e 3b 6b 10 71 a1	bf 87 d7 67	...~;k-q.....g

Figure 7: Sniffed Matter commissioning package

These include the packet size, the protocol used and the handle involved. The UUID of the sender and receiver is also specified. Finally, the contents of the package can also be seen. In all packets examined, the content was encrypted, so that the greatest risk of the router's password being captured did not apply. However, the ease with which these packages could be recorded shows how delicate the commissioning process is for Matter devices.

References

Krasnyansky, M. (N.D.) sdptool (1) – Linux man page. Available from: <https://linux.die.net/man/1/sdptool> [Accessed 14 July 2023].

Margaritelli, S., Braga, G., Trotta, G. & Gruber, K. (2021) bettercap. Available from: <https://github.com/bettercap/bettercap> [Accessed 14 July 2023].

Miller, A. C. & Estrella, M. (2018) hcitool. Available from: <https://github.com/MillerTechnologyPeru/hcitool> [Accessed 14 July 2023].

Nordic (2022) nRF Sniffer for Bluetooth LE v4.1.x. Available from: https://infocenter.nordicsemi.com/pdf/nRF_Sniffer_BLE_UG_v4.1.x.pdf [Accessed 17 July 2023].

Wireshark Foundation (N.D) The world's most popular network protocol analyzer. Available from: <https://www.wireshark.org/> [Accessed 31 July 2023].