5. Test execution and results

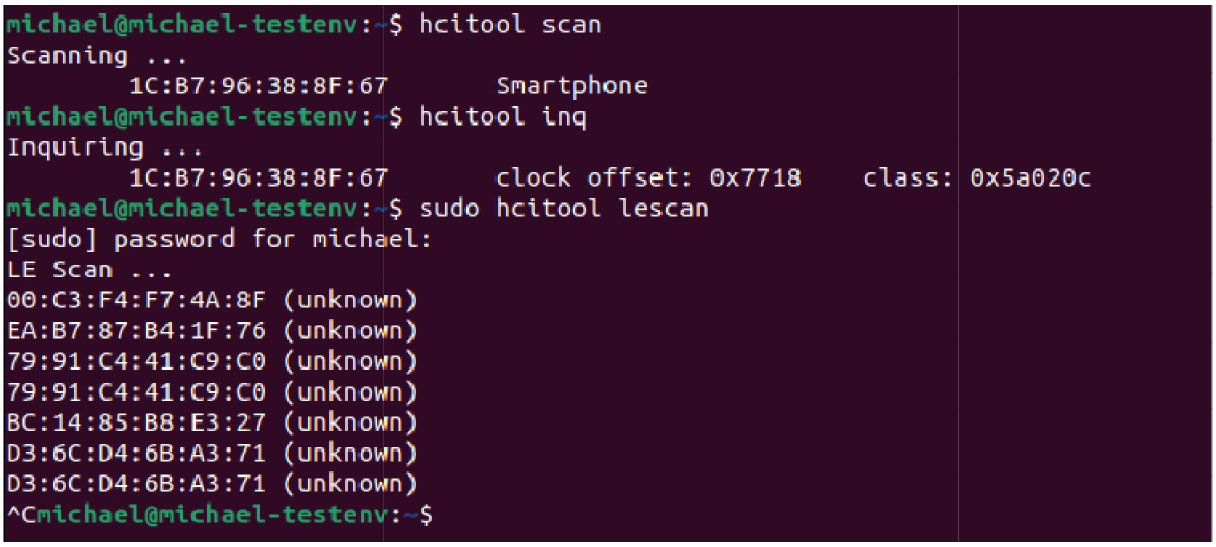
As part of this project, two of the central communication standards on which Matter is based were examined for dominant vulnerabilities and corresponding attacks were carried out to analyse the security of Matter. In the following, the Bluetooth and Wi-Fi standards are examined with regard to selected attack scenarios and exemplary tests are carried out against these standards as well as Matter.

5.1 Hacking attacks against Bluetooth

Bluetooth is one of the most widely used communication protocols in everyday use (Herrero, 2022). However, it should be noted that the standard has some vulnerabilities that affect all Bluetooth versions. A serious vulnerability is that when establishing a connection between two devices, only the slave device needs to be authenticated, but not the master device, which initiates the authentication. It follows that access to the control device means full control over the Bluetooth devices connected to the network. It also allows an unauthorised person to connect to the device before the owner and thus have control of the device until the owner physically intervenes to discard the connection. This is favoured by the fact that Bluetooth devices do not have a limited period for connection detection. This enables Bluetooth devices to be examined for their properties and possible vulnerabilities.

5.1.1 Bluetooth sniffing execution and results

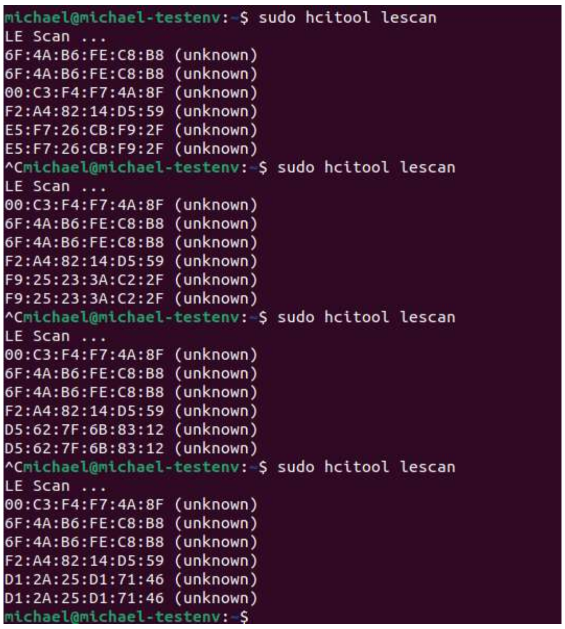
The tool hcitool was used to carry out an initial probing of the Bluetooth devices in the test environment (Miller & Estrella, 2018). Using hcitool, available Bluetooth devices can be detected and at the same time the devices can be subdivided with regard to the Bluetooth and BLE protocols. Figure 9 shows the found Bluetooth and BLE devices.



```
michael@michael-testenv:~$ hcitool scan
Scanning ...
        1C:B7:96:38:8F:67       Smartphone
michael@michael-testenv:~$ hcitool inq
Inquiring ...
        1C:B7:96:38:8F:67       clock offset: 0x7718    class: 0x5a020c
michael@michael-testenv:~$ sudo hcitool 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)
D3:6C:D4:6B:A3:71 (unknown)
D3:6C:D4:6B:A3:71 (unknown)
^Cmichael@michael-testenv:~$
```

Figure 9: hcitool Bluetooth and BLE device scan

The test shows that the Bluetooth device named for the test 'Smartphone' was found by the command ‘hcitool scan’ and the MAC address could be determined. Further information such as the clock offset and the class of the Bluetooth device could be determined using the inquire remote devices command ‘hcitool inq’. However, this command was unsuccessful with BLE devices. The ‘hcitool lescan’ command displays all BLE devices in the area, but the respective names cannot be determined. Repeated tests of the lescan command also revealed that the Matter device displayed a new MAC address after switching it off and on again. Figure 10 shows repeated lescans with multiple reboots of the Matter device.



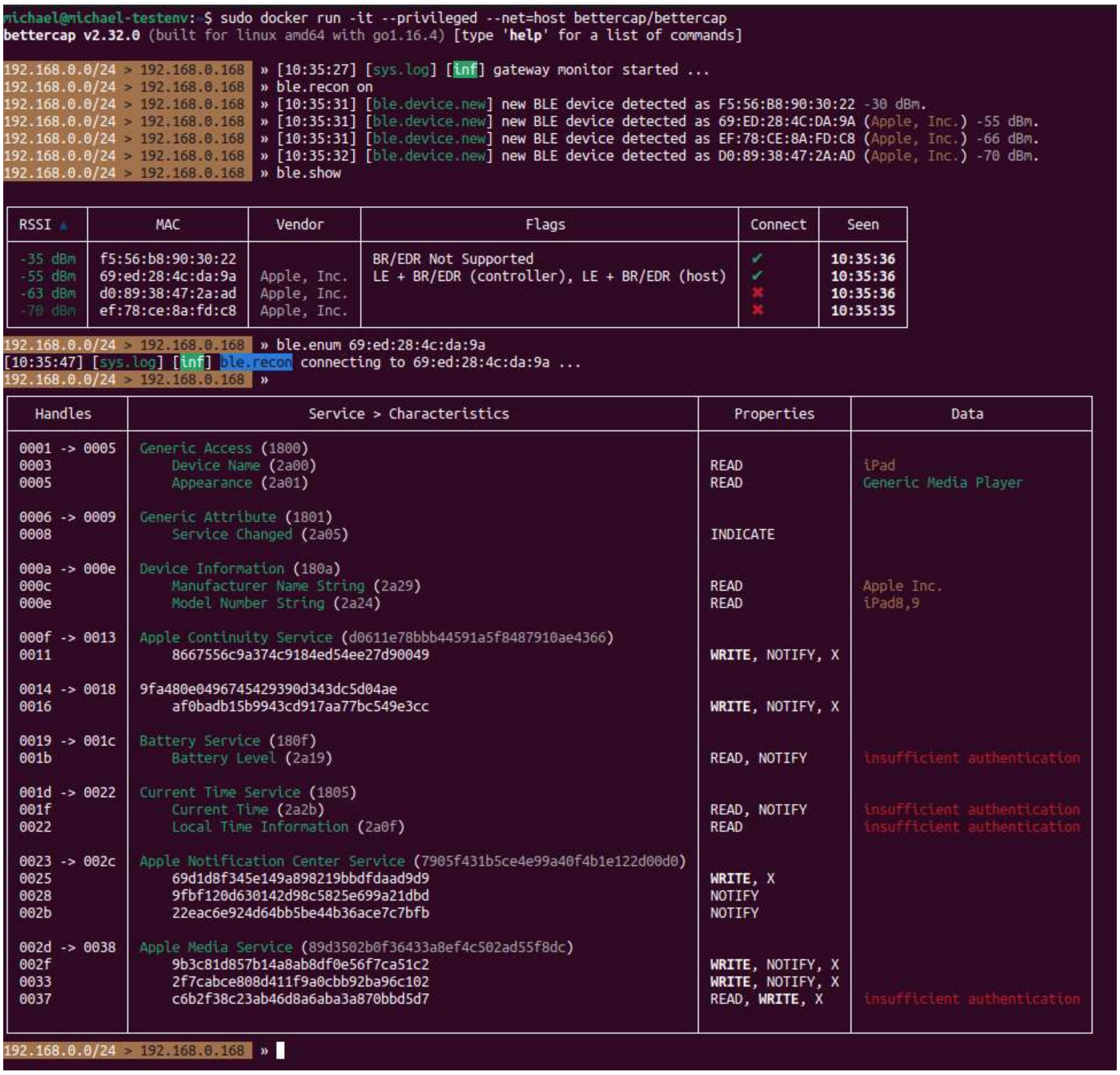
```
michael@michael-testenv:~$ sudo hcitool lescan
LE Scan ...
6F:4A:B6:FE:C8:B8 (unknown)
6F:4A:B6:FE:C8:B8 (unknown)
00:C3:F4:F7:4A:8F (unknown)
F2:A4:82:14:D5:59 (unknown)
E5:F7:26:CB:F9:2F (unknown)
E5:F7:26:CB:F9:2F (unknown)
^Cmichael@michael-testenv:~$ sudo hcitool lescan
LE Scan ...
00:C3:F4:F7:4A:8F (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 hcitool 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 hcitool 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 10: Changing Matter MAC address

To ensure that the MAC address and device are not wrongly assigned during this test, the respective scan was run for 20 seconds so that all BLE devices were detected before the Matter device was switched on. Therefore, the Matter device can be assigned to the lowest MAC address in each test.

Further tests were carried out with the tool `sdptool`. The sdptool enables further information about the device and its functions to be determined using the MAC address (Krasnyansky, N.D.). The results of this test can be found in the Appendix 2. The test revealed a variety of features and information about the Bluetooth device, but none about the BLE device.

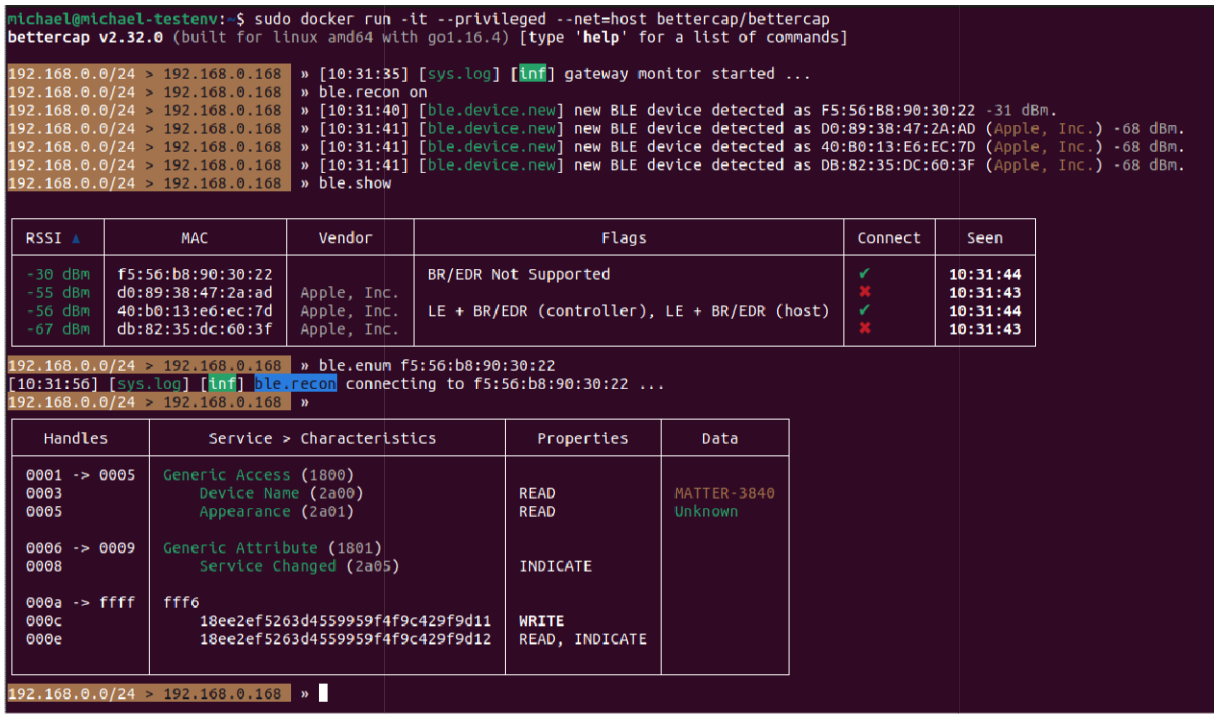
Another approach to determine information and potential vulnerabilities of BLE devices was pursued using the `bettercap` tool. The tool bettercap offers a variety of functions to sniff Wi-Fi and BLE networks (Margaritelli et al., 2021). The Ipad, which acted as a hub in the Matter network, and the Matter device were chosen as the sniffing targets for this test. For this purpose, bettercap was first used to search for BLE devices and then the characteristics available via bettercap, services and information about the devices were determined using the 'ble-enum [MAC]' command. Figure 11 shows the results of the BLE sniffing from the 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  » [10:35:27] [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  » [10:35:31] [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] [inf] 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 │ Generic Access (1800)                                            │                    │                             │
│ 0003         │     Device Name (2a00)                                           │ READ               │ iPad                        │
│ 0005         │     Appearance (2a01)                                            │ READ               │ Generic Media Player        │
│              │                                                                  │                    │                             │
│ 0006 -> 0009 │ Generic Attribute (1801)                                         │                    │                             │
│ 0008         │     Service Changed (2a05)                                       │ INDICATE           │                             │
│              │                                                                  │                    │                             │
│ 000a -> 000e │ Device Information (180a)                                        │                    │                             │
│ 000c         │     Manufacturer Name String (2a29)                              │ READ               │ Apple Inc.                  │
│ 000e         │     Model Number String (2a24)                                   │ READ               │ iPad8,9                     │
│              │                                                                  │                    │                             │
│ 000f -> 0013 │ Apple Continuity Service (d0611e78bbb44591a5f8487910ae4366)      │                    │                             │
│ 0011         │     8667556c9a374c9184ed54ee27d90049                             │ WRITE, NOTIFY, X   │                             │
│              │                                                                  │                    │                             │
│ 0014 -> 0018 │ 9fa480e0496745429390d343dc5d04ae                                 │                    │                             │
│ 0016         │     af0badb15b9943cd917aa77bc549e3cc                             │ WRITE, NOTIFY, X   │                             │
│              │                                                                  │                    │                             │
│ 0019 -> 001c │ Battery Service (180f)                                           │                    │                             │
│ 001b         │     Battery Level (2a19)                                         │ READ, NOTIFY       │ insufficient authentication │
│              │                                                                  │                    │                             │
│ 001d -> 0022 │ Current Time Service (1805)                                      │                    │                             │
│ 001f         │     Current Time (2a2b)                                          │ READ, NOTIFY       │ insufficient authentication │
│ 0022         │     Local Time Information (2a0f)                                │ READ               │ insufficient authentication │
│              │                                                                  │                    │                             │
│ 0023 -> 002c │ Apple Notification Center Service (7905f431b5ce4e99a40f4b1e122d00d0) │                │                             │
│ 0025         │     69d1d8f345e149a898219bbdfdaad9d9                             │ WRITE, X           │                             │
│ 0028         │     9fbf120d630142d98c5825e699a21dbd                             │ NOTIFY             │                             │
│ 002b         │     22eac6e924d64bb5be44b36ace7c7bfb                             │ NOTIFY             │                             │
│              │                                                                  │                    │                             │
│ 002d -> 0038 │ Apple Media Service (89d3502b0f36433a8ef4c502ad55f8dc)           │                    │                             │
│ 002f         │     9b3c81d857b14a8ab8df0e56f7ca51c2                             │ WRITE, NOTIFY, X   │                             │
│ 0033         │     2f7cabce808d411f9a0cbb92ba96c102                             │ WRITE, NOTIFY, X   │                             │
│ 0037         │     c6b2f38c23ab46d8a6aba3a870bbd5d7                             │ READ, WRITE, X     │ insufficient authentication │
│              │                                                                  │                    │                             │
└──────────────┴──────────────────────────────────────────────────────────────────┴────────────────────┴─────────────────────────────┘

192.168.0.0/24 > 192.168.0.168  » █
```

Figure 11: iPad sniffing with bettercap

The table below in Figure 11 lists the respective handles, which correspond to the functions of the device with Bluetooth, the characteristics and the access authorisations, as well as further information under the Data column. By doing this, some information related to permissions and possible insufficient settings and thus vulnerabilities could be determined. The results of testing the Matter device with bettercap are shown in Figure 12.



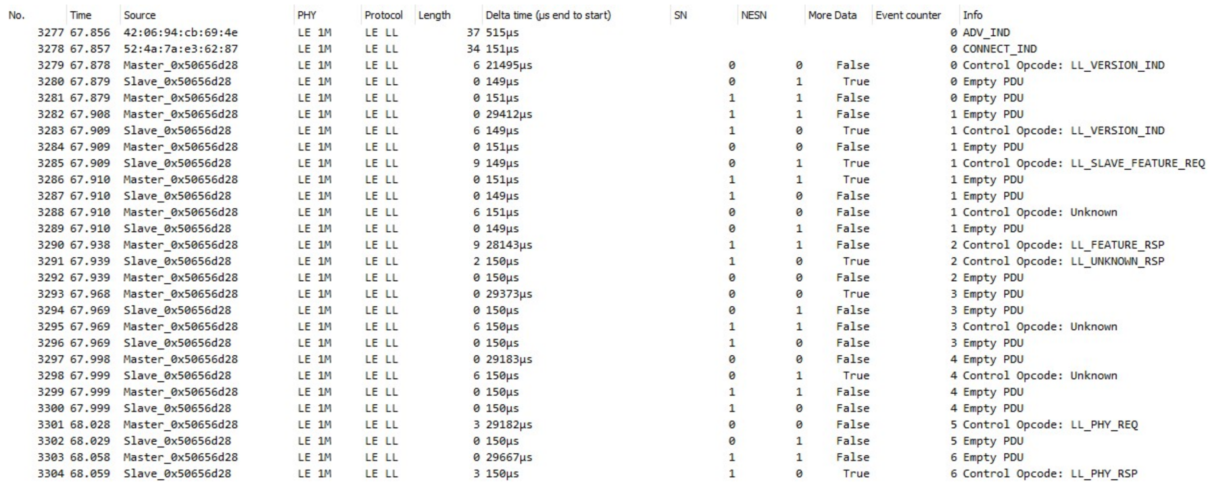
```
michael@michael-testenv:~$ sudo docker run -it --privileged --net=host bettercap/bettercap
bettercap v2.32.0 (built for linux amd64 with go1.16.4) [type 'help' for a list of commands]

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:E6: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                            │ ✔       │ 10:31:44 │
│ -55 dBm │ d0:89:38:47:2a:ad │ Apple, Inc.  │                                                 │ ✖       │ 10:31:43 │
│ -56 dBm │ 40:b0:13:e6:ec:7d │ Apple, Inc.  │ LE + BR/EDR (controller), LE + BR/EDR (host)    │ ✔       │ 10:31:44 │
│ -67 dBm │ db:82:35:dc:60:3f │ Apple, Inc.  │                                                 │ ✖       │ 10:31:43 │
└─────────┴───────────────────┴──────────────┴─────────────────────────────────────────────────┴─────────┴──────────┘

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            │ MATTER-3840 │
│ 0005         │     Appearance (2a01)                      │ READ            │ 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 12: Matter device sniffing with bettercap

While it can be seen that the handles 000c and 000e are encrypted, only the name of the Matter device 'MATTER-3840' could be determined by the handle 0003 by reading the Bluetooth characteristics. The number in the device name is the discriminator of the device, which was already assigned to the device during the flash process and is required for commissioning in a network (CSA, 2022c).

Finally, as part of the Bluetooth sniffing experiments, an attempt was made to read the communication between two devices during the pairing process. For this purpose, the Wireshark tool was used, which enables the capturing of different communication standards (Combs, N.D.). A suitable device capable of capturing BLE packets was required so that the sniffing of the Bluetooth communication could be carried out. As part of these tests, the Nordic nRF52840 dongle was used (Nordic, 2022). The process of commissioning BLE devices was chosen because Matter devices automatically set off the BLE functions after commissioning the device in a network. Furthermore, sensitive data such as the encryption and the password of the Wi-Fi network are sent during this process, so that the commissioning represents a vulnerable process. In order to minimise the probability of incorrect measurements, this test was carried out 10 times for the comparison device and the Matter device.

The commissioning of an Apple Watch was examined as a comparative test. A section of intercepted packets during commissioning can be seen in Figure 13, the complete protocol in Appendix 3.

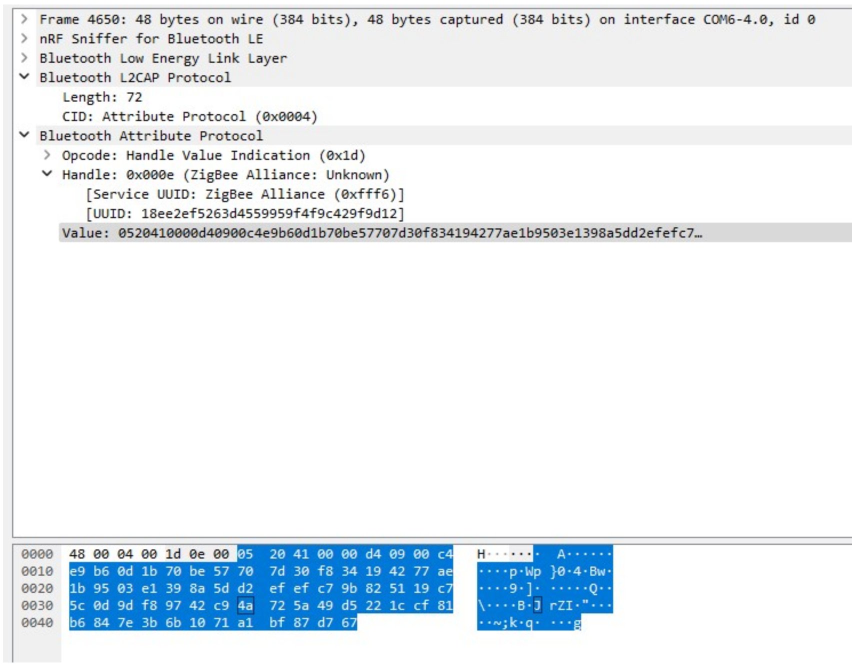


No.	Time	Source	PHY	Protocol	Length	Delta time (µs end to start)	SN	NESN	More Data	Event counter	Info
3277	67.856	42:06:94:cb:69:4e	LE 1M	LE LL	37	515µs				0	ADV_IND
3278	67.857	52:4a:7a:e3:62:87	LE 1M	LE LL	34	151µs				0	CONNECT_IND
3279	67.878	Master_0x50656d28	LE 1M	LE LL	6	21495µs	0	0	False	0	Control Opcode: LL_VERSION_IND
3280	67.879	Slave_0x50656d28	LE 1M	LE LL	0	149µs	0	1	True	0	Empty PDU
3281	67.879	Master_0x50656d28	LE 1M	LE LL	0	151µs	1	1	False	0	Empty PDU
3282	67.908	Master_0x50656d28	LE 1M	LE LL	0	29412µs	1	1	False	1	Empty PDU
3283	67.909	Slave_0x50656d28	LE 1M	LE LL	6	149µs	1	0	True	1	Control Opcode: LL_VERSION_IND
3284	67.909	Master_0x50656d28	LE 1M	LE LL	0	151µs	0	0	False	1	Empty PDU
3285	67.909	Slave_0x50656d28	LE 1M	LE LL	9	149µs	0	1	True	1	Control Opcode: LL_SLAVE_FEATURE_REQ
3286	67.910	Master_0x50656d28	LE 1M	LE LL	0	151µs	1	1	True	1	Empty PDU
3287	67.910	Slave_0x50656d28	LE 1M	LE LL	0	149µs	1	0	False	1	Empty PDU
3288	67.910	Master_0x50656d28	LE 1M	LE LL	6	151µs	0	0	False	1	Control Opcode: Unknown
3289	67.910	Slave_0x50656d28	LE 1M	LE LL	0	149µs	0	1	False	1	Empty PDU
3290	67.938	Master_0x50656d28	LE 1M	LE LL	9	28143µs	1	1	False	2	Control Opcode: LL_FEATURE_RSP
3291	67.939	Slave_0x50656d28	LE 1M	LE LL	2	150µs	1	0	True	2	Control Opcode: LL_UNKNOWN_RSP
3292	67.939	Master_0x50656d28	LE 1M	LE LL	0	150µs	0	0	False	2	Empty PDU
3293	67.968	Master_0x50656d28	LE 1M	LE LL	0	29373µs	0	0	True	3	Empty PDU
3294	67.969	Slave_0x50656d28	LE 1M	LE LL	0	150µs	0	1	False	3	Empty PDU
3295	67.969	Master_0x50656d28	LE 1M	LE LL	6	150µs	1	1	False	3	Control Opcode: Unknown
3296	67.969	Slave_0x50656d28	LE 1M	LE LL	0	150µs	1	0	False	3	Empty PDU
3297	67.998	Master_0x50656d28	LE 1M	LE LL	0	29183µs	0	0	False	4	Empty PDU
3298	67.999	Slave_0x50656d28	LE 1M	LE LL	6	150µs	0	1	True	4	Control Opcode: Unknown
3299	67.999	Master_0x50656d28	LE 1M	LE LL	0	150µs	1	1	False	4	Empty PDU
3300	67.999	Slave_0x50656d28	LE 1M	LE LL	0	150µs	1	0	False	4	Empty PDU
3301	68.028	Master_0x50656d28	LE 1M	LE LL	3	29182µs	0	0	False	5	Control Opcode: LL_PHY_REQ
3302	68.029	Slave_0x50656d28	LE 1M	LE LL	0	150µs	0	1	False	5	Empty PDU
3303	68.058	Master_0x50656d28	LE 1M	LE LL	0	29667µs	1	1	False	6	Empty PDU
3304	68.059	Slave_0x50656d28	LE 1M	LE LL	3	150µs	1	0	True	6	Control Opcode: LL_PHY_RSP

Figure 13: Commissioning Apple Watch package sniffing

The test shows that the first communication could be recorded. However, it had to be determined that all captured packets were encrypted and the exchange of keys could not be captured. After the first information had been exchanged, eavesdropping was no longer possible and the further course of commissioning was secret.

For the test of sniffing of the commissioning process of the Matter device, an ESP32 controller freshly flashed with the lighting app was integrated into a Matter network with an iPad as a hub. For this purpose, the Matter device was searched for on the iPad and commissioning process was initialised. The complete protocol of the collected packages of a commissioning can be found in Appendix 4. As in the comparison test, the communication could be monitored via Wireshark. However, compared to the previous tests, it was possible in some scans to listen to the complete communication of the commissioning up to the end. A sample package from this process is shown in Figure 14.



```
> 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
v Bluetooth L2CAP Protocol
    Length: 72
    CID: Attribute Protocol (0x0004)
v Bluetooth Attribute Protocol
  > Opcode: Handle Value Indication (0x1d)
  v Handle: 0x000e (ZigBee Alliance: Unknown)
      [Service UUID: ZigBee Alliance (0xfff6)]
      [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 ae   ····p·Wp }0·4·Bw·
0020  1b 95 03 e1 39 8a 5d d2  ef ef c7 9b 82 51 19 c7   ····9·]· ·····Q··
0030  5c 0d 9d f8 97 42 c9 4a  72 5a 49 d5 22 1c cf 81   \····B·J rZI·"···
0040  b6 84 7e 3b 6b 10 71 a1  bf 87 d7 67               ··~;k·q· ···g
```

Figure 14: Sniffed Matter commissioning package

This packet shows that various information such as the UUID can be determined by capturing the BLE packets. However, the actual packet content was encrypted as in the comparison test and the initial exchange of the key could not be identified. After the Matter device was successfully integrated into the network, all communication via BLE was stopped by the Matter device.

5.2 Hacking attacks against Wi-Fi

Wi-Fi is the most widely used standard for realising home networks. The standard has security mechanisms such as WPA3, which are used for encryption. In addition to the assumption that WPA3 no longer has to be classified as completely secure, other methods can be found to exploit vulnerabilities in the standard and carry out attacks against Wi-Fi networks (Baray & Ojha, 2021). Two important approaches for attacks against Wi-Fi networks are de-authentication and replay attacks, which were carried out on Matter networks as part of the project to investigate Matter's resilience to these types of attacks.

5.2.1 De-authentication attack

A de-authentication attack aims to interrupt a client's connection to an AP and therefore falls into the category of a denial of service attack. De-authentication frames are packets based on the IEEE 802.11 standard, which are sent to terminate an existing connection (IEEE Standards Association, 2021). After the target has received these packets, all further packets from the affected devices are rejected until re-establishing of the connection occurs via a handshake.

In a de-authentication attack, the attacker pretends to be either the AP or the target device. A schematic representation can be seen in Figure 15.

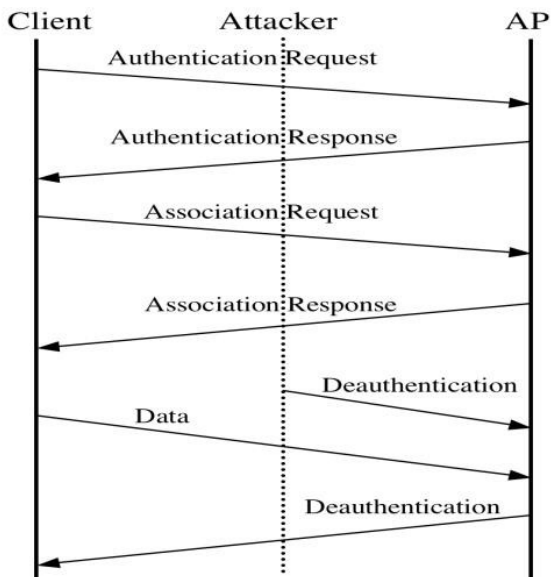


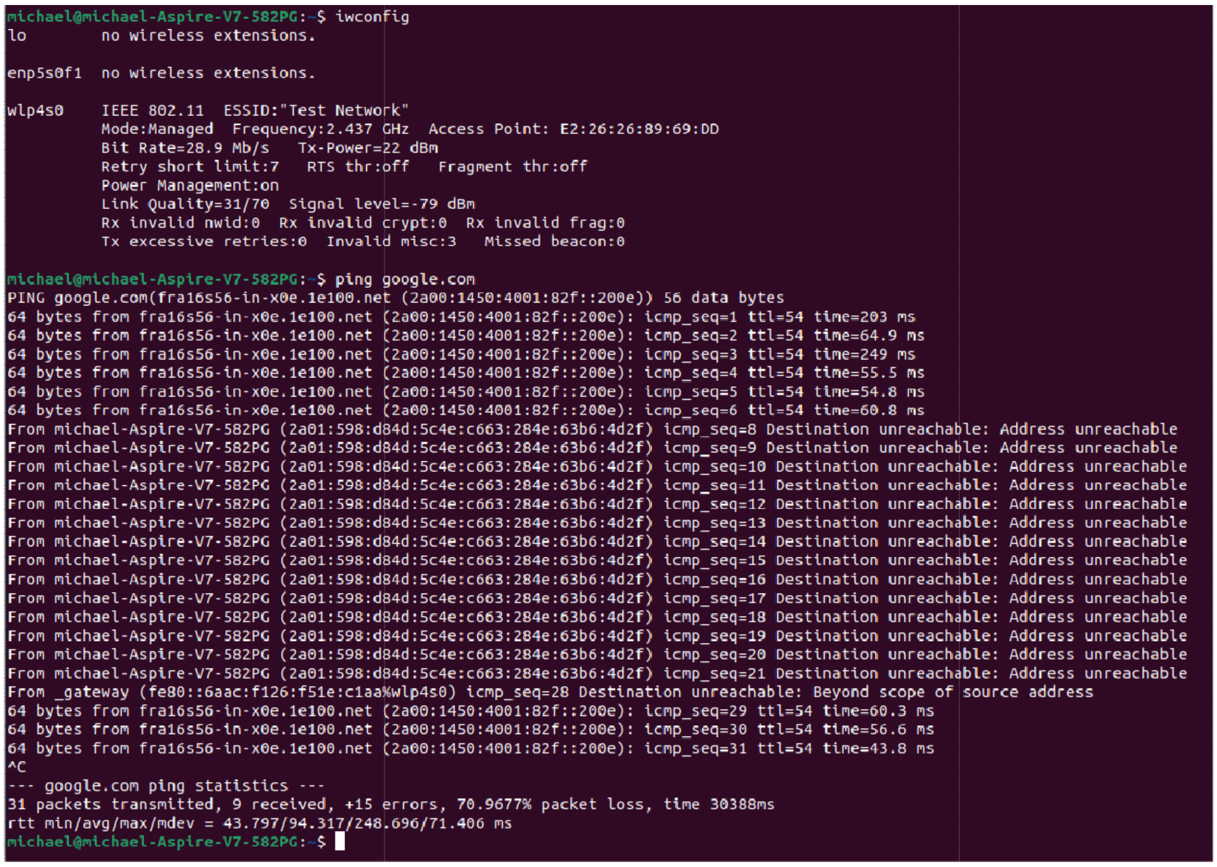
Figure 15: De-authentication attack schematic (Noman, et al., 2015)

The attacker sends spoofed de-authentication packets to the target device or AP that appear to be trying to disconnect. Once the target device or AP receives the spoofed de-authentication packets, they interpret them as legitimate disconnect requests (Kristiyanto & Ernastuti, 2020). As a result of the attack, communication within the network is disrupted and the client no longer has access to the network. The target device must re-authenticate and reconnect to the network to re-establish the communication, but this can be prevented by the attacker by repeated de-authentication frames.

In order to test and compare the resilience of Matter devices to de-authentication attacks, attacks were carried out on the AP and the device of the Matter network. The comparative study was carried out using a test network consisting of an AP provided by an Android smartphone (HUAWEI P30) and a computer, which acted as a device. A test was also executed in which the same computer was connected to the Matter network router and this AP was attacked. The attack was carried out by the WiFi Deauther Mini V3 development board. This is based on the ESP8266 microcontroller, which is able to scan for Wi-Fi devices and APs and then carry out targeted attacks on the devices found (Kristiyanto & Ernastuti, 2020).

5.2.1.1 De-authentication attack execution and results

First, the de-authentication attacks were carried out on the test network of the smartphone and the computer. The attack on the computer can be seen in Figure 16.



```
michael@michael-Aspire-V7-582PG:~$ iwconfig
lo        no wireless extensions.

enp5s0f1  no wireless extensions.

wlp4s0    IEEE 802.11  ESSID:"Test Network"
          Mode:Managed  Frequency:2.437 GHz  Access Point: E2:26:26:89:69:DD
          Bit Rate=28.9 Mb/s   Tx-Power=22 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
          Link Quality=31/70  Signal level=-79 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:3   Missed beacon:0

michael@michael-Aspire-V7-582PG:~$ ping google.com
PING google.com(fra16s56-in-x0e.1e100.net (2a00:1450:4001:82f::200e)) 56 data bytes
64 bytes from fra16s56-in-x0e.1e100.net (2a00:1450:4001:82f::200e): icmp_seq=1 ttl=54 time=203 ms
64 bytes from fra16s56-in-x0e.1e100.net (2a00:1450:4001:82f::200e): icmp_seq=2 ttl=54 time=64.9 ms
64 bytes from fra16s56-in-x0e.1e100.net (2a00:1450:4001:82f::200e): icmp_seq=3 ttl=54 time=249 ms
64 bytes from fra16s56-in-x0e.1e100.net (2a00:1450:4001:82f::200e): icmp_seq=4 ttl=54 time=55.5 ms
64 bytes from fra16s56-in-x0e.1e100.net (2a00:1450:4001:82f::200e): icmp_seq=5 ttl=54 time=54.8 ms
64 bytes from fra16s56-in-x0e.1e100.net (2a00:1450:4001:82f::200e): icmp_seq=6 ttl=54 time=60.8 ms
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=8 Destination unreachable: Address unreachable
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=9 Destination unreachable: Address unreachable
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=10 Destination unreachable: Address unreachable
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=11 Destination unreachable: Address unreachable
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=12 Destination unreachable: Address unreachable
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=13 Destination unreachable: Address unreachable
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=14 Destination unreachable: Address unreachable
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=15 Destination unreachable: Address unreachable
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=16 Destination unreachable: Address unreachable
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=17 Destination unreachable: Address unreachable
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=18 Destination unreachable: Address unreachable
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=19 Destination unreachable: Address unreachable
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=20 Destination unreachable: Address unreachable
From michael-Aspire-V7-582PG (2a01:598:d84d:5c4e:c663:284e:63b6:4d2f) icmp_seq=21 Destination unreachable: Address unreachable
From _gateway (fe80::6aac:f126:f51e:c1aa%wlp4s0) icmp_seq=28 Destination unreachable: Beyond scope of source address
64 bytes from fra16s56-in-x0e.1e100.net (2a00:1450:4001:82f::200e): icmp_seq=29 ttl=54 time=60.3 ms
64 bytes from fra16s56-in-x0e.1e100.net (2a00:1450:4001:82f::200e): icmp_seq=30 ttl=54 time=56.6 ms
64 bytes from fra16s56-in-x0e.1e100.net (2a00:1450:4001:82f::200e): icmp_seq=31 ttl=54 time=43.8 ms
^C
--- google.com ping statistics ---
31 packets transmitted, 9 received, +15 errors, 70.9677% packet loss, time 30388ms
rtt min/avg/max/mdev = 43.797/94.317/248.696/71.406 ms
michael@michael-Aspire-V7-582PG:~$
```

Figure 16: De-authentication of test network, computer attacked

Based on the observations, it can be stated that as soon as the de-authentication attack took place, the Wi-Fi communication with the AP was disrupted. The computer was still connected to the AP, but the ping request was reported as unreachable during the attack. After the end of the attack, the connection was restored, which can be confirmed by the response to the ping request.

The attack on the AP connected to the computer produced similar test results. The test can be seen in Appendix 5. A difference to the previous test with the computer as the target of the attack was that the connection between the computer and the AP was lost during the attack on the AP. Only after the attack ended could the computer find the Wi-Fi network again and establish a new connection. Both of the previous tests were also carried out with the AP via which the Matter network also communicates. The same computer was used here as for the network test. The results of the test can be seen in Appendix 6 and 7.

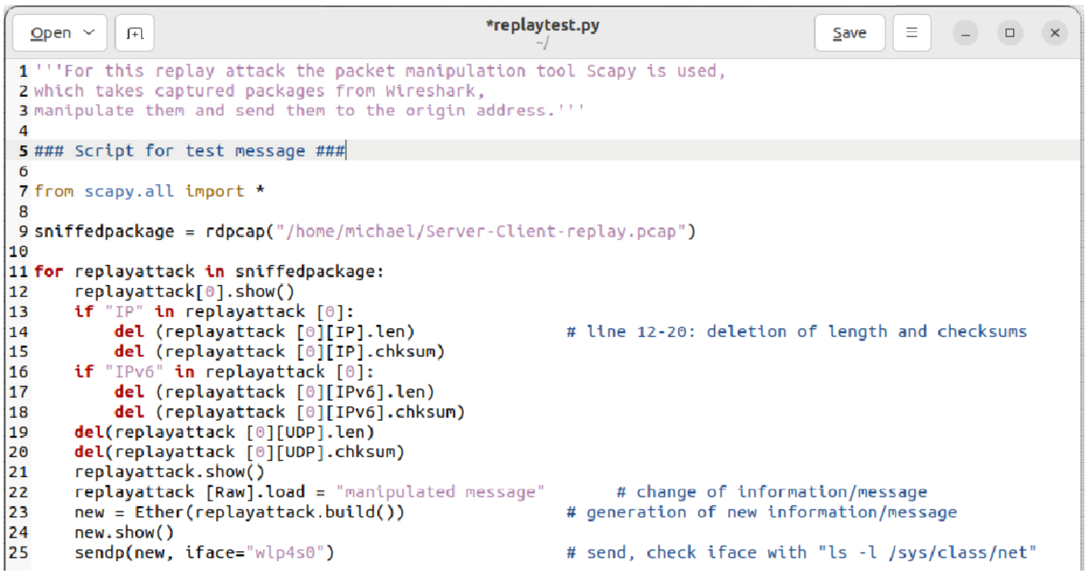
Based on these tests, it could be determined that an attack on the AP as well as the computer does not lead to any impairment of communication. The same attacks were also carried out against the Matter network. The test result against the AP of the Matter network can be seen in Appendix 8 and against the Matter device in Appendix 9. The same result was observed in both tests. The attack against the AP as well as against the device leads to a termination of communication. This can be seen from the response 'app-devicecallbacks: Lost IPv4 (IPv6) connectivity...'. During the attacks, the device repeatedly tries to reconnect, but without success. A new connection could only be restored after the attacks had ended.

5.2.2 Replay attack

A replay attack is a cryptanalytic form of attack on the authenticity of data in a network. Here, an attacker sends previously recorded data packets to impersonate a legitimate identity and execute a command. Thus, the replay attack falls into the spectrum of man-in-the-middle attacks. Since the successful execution of a replay attack can result in unauthorised commands being resent as often as required, this type of attack poses a serious risk, as it can have a significant impact on the function of devices in a network.

5.2.2.1 Replay attack execution and results

A Python script was created to test the data integrity of Matter devices using a replay attack. The scapy tool was used for this, with which recorded packets can be read in, individual components manipulated and the changed packets finally sent again (Scapy, N.D.). Wireshark was used to intercept the originally sent packets. In order to confirm the functionality of the created Python script, a test network was created by creating a master-client network between two Ubuntu 22.04 TLS computers and the communication via the freely chosen port 9999 was recorded. Figure 17 shows the Python script used.

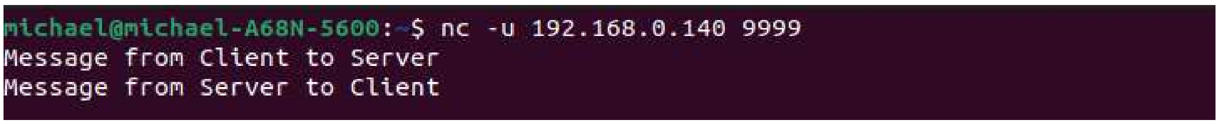


```
 1 '''For this replay attack the packet manipulation tool Scapy is used,
 2 which takes captured packages from Wireshark,
 3 manipulate them and send them to the origin address.'''
 4
 5 ### Script for test message ###
 6
 7 from scapy.all import *
 8
 9 sniffedpackage = rdpcap("/home/michael/Server-Client-replay.pcap")
10
11 for replayattack in sniffedpackage:
12     replayattack[0].show()
13     if "IP" in replayattack [0]:
14         del (replayattack [0][IP].len)             # line 12-20: deletion of length and checksums
15         del (replayattack [0][IP].chksum)
16     if "IPv6" in replayattack [0]:
17         del (replayattack [0][IPv6].len)
18         del (replayattack [0][IPv6].chksum)
19     del(replayattack [0][UDP].len)
20     del(replayattack [0][UDP].chksum)
21     replayattack.show()
22     replayattack [Raw].load = "manipulated message"      # change of information/message
23     new = Ether(replayattack.build())              # generation of new information/message
24     new.show()
25     sendp(new, iface="wlp4s0")                     # send, check iface with "ls -l /sys/class/net"
```

Figure 17: Replay attack python script for comparison test

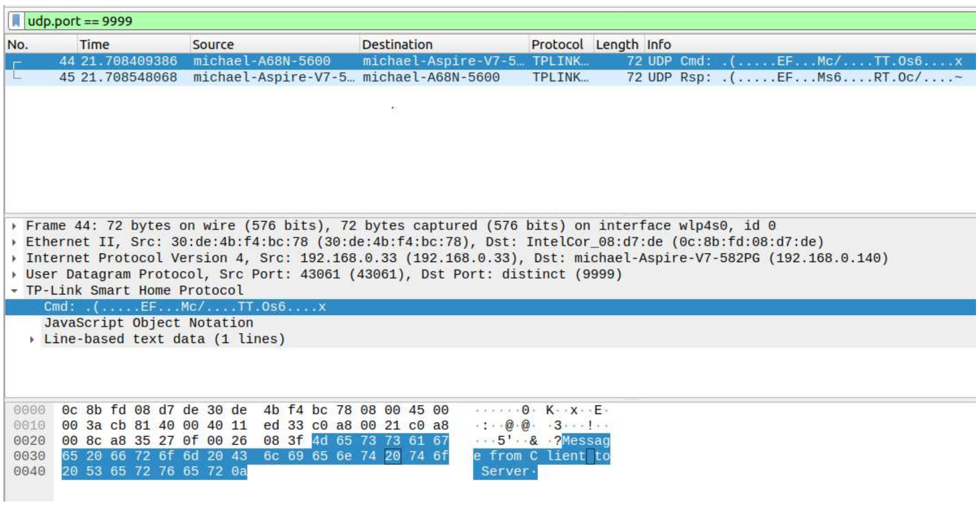
In the created script, previously collected packets are read using a pcap file, which can be seen in line 10. Then, in lines 12 to 20, the length specifications and checksums are deleted at the IP and UDP level. In line 22, the information content is changed, in this case the message, and then in line 23 the deleted information is regenerated. Finally, in line 25, the manipulated packets are sent via a selected network connection.

Figure 18 shows the messages sent on the client side and Figure 19 shows the packets captured using Wireshark on the server side.



```
michael@michael-A68N-5600:~$ nc -u 192.168.0.140 9999
Message from Client to Server
Message from Server to Client
```

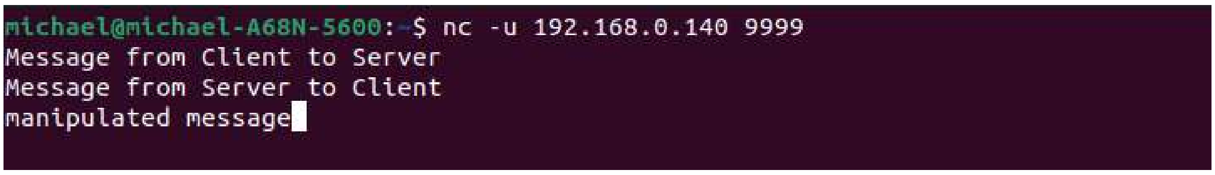
Figure 18: Client-Server terminal communication



```
udp.port == 9999
No.  Time          Source                 Destination            Protocol  Length Info
  44 21.708409386  michael-A68N-5600      michael-Aspire-V7-5…   TPLINK…       72 UDP Cmd: .(.....EF...Mc/....TT.Os6....x
  45 21.708548068  michael-Aspire-V7-5…   michael-A68N-5600      TPLINK…       72 UDP Rsp: .(.....EF...Ms6....RT.Oc/....~

▸ Frame 44: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface wlp4s0, id 0
▸ Ethernet II, Src: 30:de:4b:f4:bc:78 (30:de:4b:f4:bc:78), Dst: IntelCor_08:d7:de (0c:8b:fd:08:d7:de)
▸ Internet Protocol Version 4, Src: 192.168.0.33 (192.168.0.33), Dst: michael-Aspire-V7-582PG (192.168.0.140)
▸ User Datagram Protocol, Src Port: 43061 (43061), Dst Port: distinct (9999)
▾ TP-Link Smart Home Protocol
    Cmd: .(.....EF...Mc/....TT.Os6....x
    JavaScript Object Notation
  ▸ Line-based text data (1 lines)

0000  0c 8b fd 08 d7 de 30 de  4b f4 bc 78 08 00 45 00   ······0· K··x··E·
0010  00 3a cb 81 40 00 40 11  ed 33 c0 a8 00 21 c0 a8   ·:··@·@· ·3···!··
0020  00 8c a8 35 27 0f 00 26  08 3f 4d 65 73 73 61 67   ···5'··& ·?Messag
0030  65 20 66 72 6f 6d 20 43  6c 69 65 6e 74 20 74 6f   e from C lient to
0040  20 53 65 72 76 65 72 0a                             Server·
```

Figure 19: Captured Client-Server communication via Wireshark

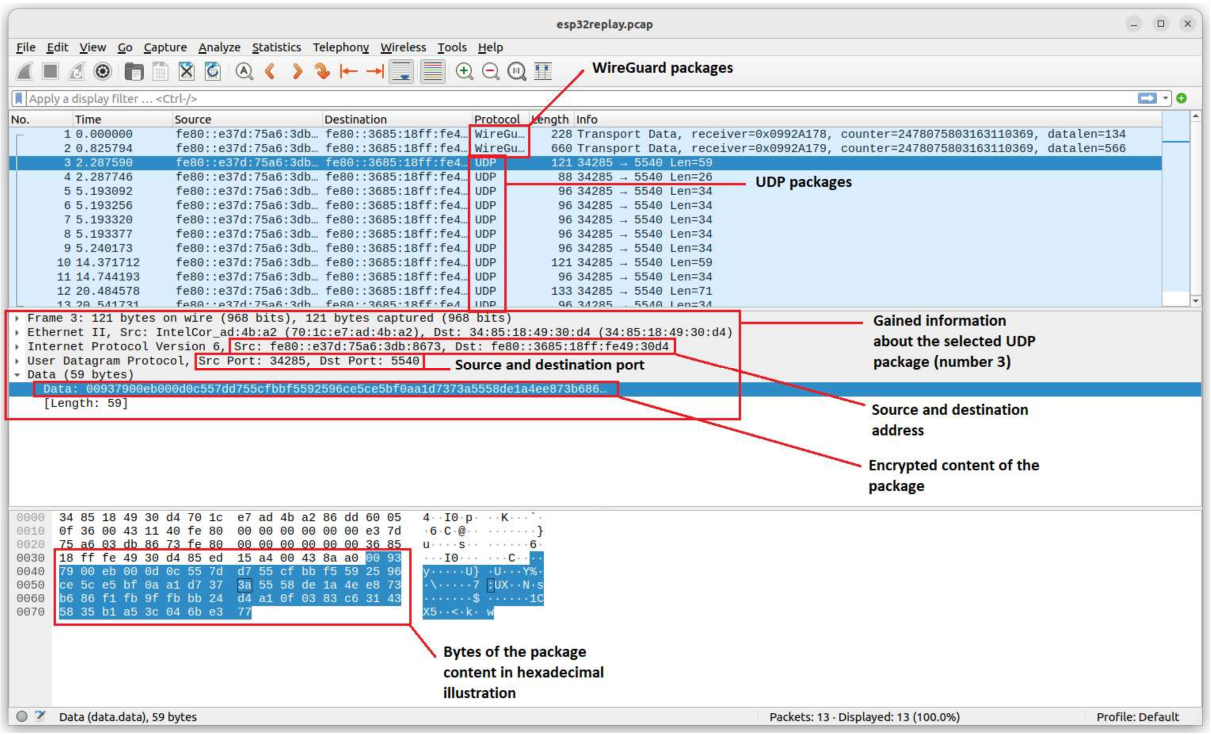
The packets recorded by Wireshark can be used to find out some information about the data exchange between the devices involved. In this way, the names and IP addresses of the sender and recipient can be seen. Furthermore, the previously selected port, via which the communication takes place, and finally also the content, in this case the message that was transmitted by the packet, could be displayed. The UDP packet from server to client was then saved in isolation and the Python script executed, see Appendix 10. After the replay attack was carried out, the manipulated content of the message 'manipulated message' was displayed on the client side, as can be seen in Figure 20.



```
michael@michael-A68N-5600:~$ nc -u 192.168.0.140 9999
Message from Client to Server
Message from Server to Client
manipulated message
```

Figure 20: Client-Server replay attack result

The same procedure was now used to test Matter devices for replay attacks. For this purpose, an ESP32 controller with the Matter lighting app application was integrated into a network with a computer as a hub. With the Matter controller implementation chip-tool it is possible to integrate Matter devices into a network by using a computer as a hub to perform tests on the Matter device (Kajor et al., 2023). After the Matter device was integrated into the network, the LED of the Matter device was switched on via chip-tool. The packets sent were now recorded again using Wireshark, see Figure 21.


esp32replay.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
WireGuard packages
UDP packages
Gained information about the selected UDP package (number 3)
Source and destination port
Source and destination address
Encrypted content of the package
Bytes of the package content in hexadecimal illustration

```
No.  Time        Source                Destination           Protocol Length Info
   1 0.000000    fe80::e37d:75a6:3db… fe80::3685:18ff:fe4… WireGu…    228 Transport Data, receiver=0x0992A178, counter=2478075803163110369, datalen=134
   2 0.825794    fe80::e37d:75a6:3db… fe80::3685:18ff:fe4… WireGu…    660 Transport Data, receiver=0x0992A179, counter=2478075803163110369, datalen=566
   3 2.287590    fe80::e37d:75a6:3db… fe80::3685:18ff:fe4… UDP        121 34285 → 5540 Len=59
   4 2.287746    fe80::e37d:75a6:3db… fe80::3685:18ff:fe4… UDP         88 34285 → 5540 Len=26
   5 5.193092    fe80::e37d:75a6:3db… fe80::3685:18ff:fe4… UDP         96 34285 → 5540 Len=34
   6 5.193256    fe80::e37d:75a6:3db… fe80::3685:18ff:fe4… UDP         96 34285 → 5540 Len=34
   7 5.193320    fe80::e37d:75a6:3db… fe80::3685:18ff:fe4… UDP         96 34285 → 5540 Len=34
   8 5.193377    fe80::e37d:75a6:3db… fe80::3685:18ff:fe4… UDP         96 34285 → 5540 Len=34
   9 5.240173    fe80::e37d:75a6:3db… fe80::3685:18ff:fe4… UDP         96 34285 → 5540 Len=34
  10 14.371712   fe80::e37d:75a6:3db… fe80::3685:18ff:fe4… UDP        121 34285 → 5540 Len=59
  11 14.744193   fe80::e37d:75a6:3db… fe80::3685:18ff:fe4… UDP         96 34285 → 5540 Len=34
  12 20.484578   fe80::e37d:75a6:3db… fe80::3685:18ff:fe4… UDP        133 34285 → 5540 Len=71
  13 20.541731   fe80::e37d:75a6:3db  fe80::3685:18ff:fe4  UDP         96 34285 → 5540 Len=34
```

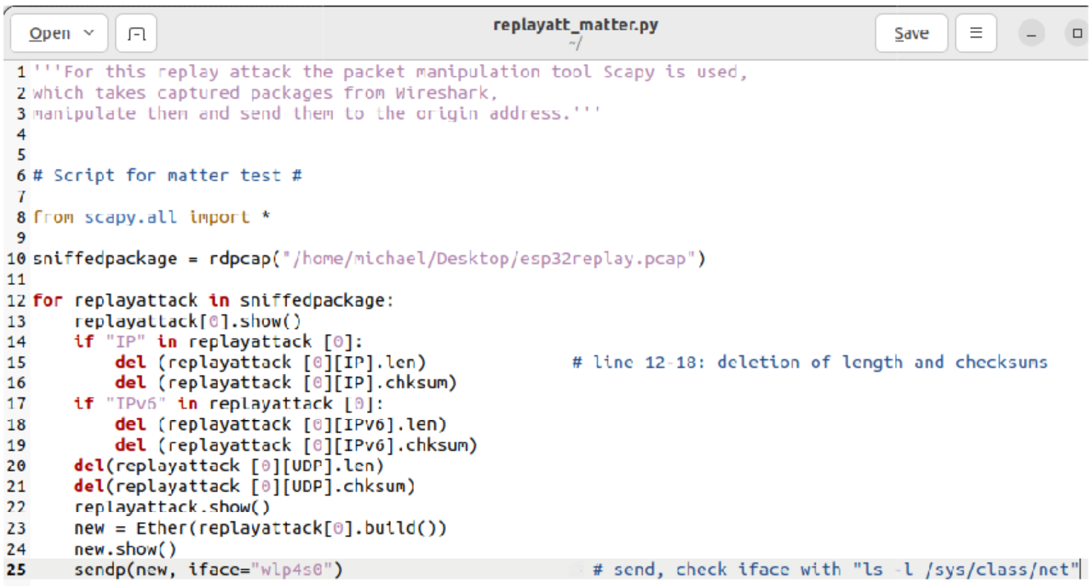
```
› Frame 3: 121 bytes on wire (968 bits), 121 bytes captured (968 bits)
› Ethernet II, Src: IntelCor_ad:4b:a2 (70:1c:e7:ad:4b:a2), Dst: 34:85:18:49:30:d4 (34:85:18:49:30:d4)
› Internet Protocol Version 6, Src: fe80::e37d:75a6:3db:8673, Dst: fe80::3685:18ff:fe49:30d4
› User Datagram Protocol, Src Port: 34285, Dst Port: 5540
⌄ Data (59 bytes)
    Data: 00937900eb000d0c557dd755cfbbf5592596ce5ce5bf0aa1d7373a5558de1a4ee873b686…
    [Length: 59]
```

```
0000  34 85 18 49 30 d4 70 1c  e7 ad 4b a2 86 dd 60 05   4··I0·p· ··K···`·
0010  0f 36 00 43 11 40 fe 80  00 00 00 00 00 00 e3 7d   ·6·C·@·· ·······}
0020  75 a6 03 db 86 73 fe 80  00 00 00 00 00 00 36 85   u····s·· ······6·
0030  18 ff fe 49 30 d4 85 ed  15 a4 00 43 8a a0 00 93   ···I0··· ···C····
0040  79 00 eb 00 0d 0c 55 7d  d7 55 cf bb f5 59 25 96   y·····U} ·U···Y%·
0050  ce 5c e5 bf 0a a1 d7 37  3a 55 58 de 1a 4e e8 73   ·\·····7 :UX··N·s
0060  b6 86 f1 fb 9f fb bb 24  d4 a1 0f 03 83 c6 31 43   ·······$ ······1C
0070  58 35 b1 a5 3c 04 6b e3  77                        X5··<·k· w
```

```
Data (data.data), 59 bytes          Packets: 13 · Displayed: 13 (100.0%)          Profile: Default
```

Figure 21: Captured Matter communication via Wireshark

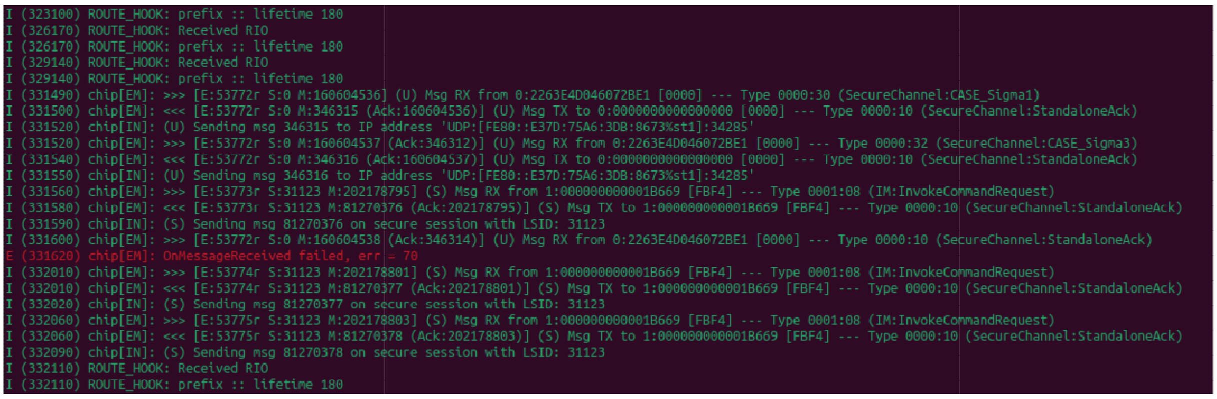
The packets show that initially WireGuard protocol packets are sent and the subsequent communication takes place via UDP packets. It should be noted that the content of the WireGuard packets is specified as 'Encrypted Packet', while the content of the UDP packets is readable but encrypted. After the packets sent during the switch-on process of the Matter device's LED were successfully intercepted, the LED was switched off again and it was tried to repeat the switch-on process using a replay attack. For this purpose, the previously used Python script was used in a slightly different form, which can be seen in Figure 22.



```python
1  '''For this replay attack the packet manipulation tool Scapy is used,
2  which takes captured packages from Wireshark,
3  manipulate them and send them to the origin address.'''
4
5
6  # Script for matter test #
7
8  from scapy.all import *
9
10 sniffedpackage = rdpcap("/home/michael/Desktop/esp32replay.pcap")
11
12 for replayattack in sniffedpackage:
13     replayattack[0].show()
14     if "IP" in replayattack [0]:
15         del (replayattack [0][IP].len)             # line 12-18: deletion of length and checksums
16         del (replayattack [0][IP].chksum)
17     if "IPv6" in replayattack [0]:
18         del (replayattack [0][IPv6].len)
19         del (replayattack [0][IPv6].chksum)
20     del(replayattack [0][UDP].len)
21     del(replayattack [0][UDP].chksum)
22     replayattack.show()
23     new = Ether(replayattack[0].build())
24     new.show()
25     sendp(new, iface="wlp4s0")                      # send, check iface with "ls -l /sys/class/net"
```

Figure 22: Matter replay attack python script

After executing the replay attack, it was determined that the Matter device received the packets, which can be seen in Figure 23.



```
I (323100) ROUTE_HOOK: prefix :: lifetime 180
I (326170) ROUTE_HOOK: Received RIO
I (326170) ROUTE_HOOK: prefix :: lifetime 180
I (329140) ROUTE_HOOK: Received RIO
I (329140) ROUTE_HOOK: prefix :: lifetime 180
I (331490) chip[EM]: >>> [E:53772r S:0 M:160604536] (U) Msg RX from 0:2263E4D046072BE1 [0000] --- Type 0000:30 (SecureChannel:CASE_Sigma1)
I (331500) chip[EM]: <<< [E:53772r S:0 M:346315 (Ack:160604536)] (U) Msg TX to 0:0000000000000000 [0000] --- Type 0000:10 (SecureChannel:StandaloneAck)
I (331520) chip[IN]: (U) Sending msg 346315 to IP address 'UDP:[FE80::E37D:75A6:3DB:8673%st1]:34285'
I (331520) chip[EM]: >>> [E:53772r S:0 M:160604537 (Ack:346312)] (U) Msg RX from 0:2263E4D046072BE1 [0000] --- Type 0000:32 (SecureChannel:CASE_Sigma3)
I (331540) chip[EM]: <<< [E:53772r S:0 M:346316 (Ack:160604537)] (U) Msg TX to 0:0000000000000000 [0000] --- Type 0000:10 (SecureChannel:StandaloneAck)
I (331550) chip[IN]: (U) Sending msg 346316 to IP address 'UDP:[FE80::E37D:75A6:3DB:8673%st1]:34285'
I (331560) chip[EM]: >>> [E:53773r S:31123 M:202178795] (S) Msg RX from 1:000000000001B669 [FBF4] --- Type 0001:08 (IM:InvokeCommandRequest)
I (331580) chip[EM]: <<< [E:53773r S:31123 M:81270376 (Ack:202178795)] (S) Msg TX to 1:000000000001B669 [FBF4] --- Type 0000:10 (SecureChannel:StandaloneAck)
I (331590) chip[IN]: (S) Sending msg 81270376 on secure session with LSID: 31123
I (331600) chip[EM]: >>> [E:53772r S:0 M:160604538 (Ack:346314)] (U) Msg RX from 0:2263E4D046072BE1 [0000] --- Type 0000:10 (SecureChannel:StandaloneAck)
E (331620) chip[EM]: OnMessageReceived failed, err = 70
I (332010) chip[EM]: >>> [E:53774r S:31123 M:202178801] (S) Msg RX from 1:000000000001B669 [FBF4] --- Type 0001:08 (IM:InvokeCommandRequest)
I (332010) chip[EM]: <<< [E:53774r S:31123 M:81270377 (Ack:202178801)] (S) Msg TX to 1:000000000001B669 [FBF4] --- Type 0000:10 (SecureChannel:StandaloneAck)
I (332020) chip[IN]: (S) Sending msg 81270377 on secure session with LSID: 31123
I (332060) chip[EM]: >>> [E:53775r S:31123 M:202178803] (S) Msg RX from 1:000000000001B669 [FBF4] --- Type 0001:08 (IM:InvokeCommandRequest)
I (332060) chip[EM]: <<< [E:53775r S:31123 M:81270378 (Ack:202178803)] (S) Msg TX to 1:000000000001B669 [FBF4] --- Type 0000:10 (SecureChannel:StandaloneAck)
I (332090) chip[IN]: (S) Sending msg 81270378 on secure session with LSID: 31123
I (332110) ROUTE_HOOK: Received RIO
I (332110) ROUTE_HOOK: prefix :: lifetime 180
```

Figure 23: Matter replay attack execution result

Despite receiving the packages, however, they were rejected and the LED could not be switched on as a result. To verify this observation, this test was repeated 10 times, each time giving the same result.

5.3 Further results

When carrying out the tests, some observations could be made which could not be explicitly counted among the experiments, but nevertheless have an added value and are therefore noted at this point.

It was found that the choice of hub device, which is responsible for controlling the Matter devices in a network, has an impact on the usability of non-certified Matter devices. Two hubs were used to conduct the tests. The first hub was a third generation iPad, the second hub device was a second generation Google Nest Hub. Both hub devices recognised the self-flashed microcontroller as a non-certified device. While the iPad warned that the device being paired was an uncertified device and it was possible to decide to take the risk, the Google Nest Hub device aborted the commissioning of the Matter device. Only the notification was displayed that the device to be paired was a non-certified device. It was not possible to accept the risk, so the microcontrollers flashed for the experiments could not be operated with the Google Nest Hub.

It was also determined during the experiments that the integration of the flashed devices into the Matter network was not always successful. In about 20% of the attempts to connect the Matter microcontroller to the iPad as a hub device, commissioning was carried out and feedback was given that the process was successfully completed. However, it was not possible to operate the controller via the hub. The Hub device could not find the Matter device, although the commissioning was classified as successful immediately before. By monitoring the Matter device, it could also be determined that the device has access to the Wi-Fi network, so that a missing connection to the network is considered unlikely. Furthermore, the commissioning was completed, so that a new attempt to integrate the device into the network was not possible because BLE was switched off. Since the hub device could not communicate with the Matter device, it was also not possible to decouple the device from the network. The only solution found to pair the controller with the hub again was to delete the Matter device from the hub and flash the microcontroller again. A conclusive explanation as to why in some cases the connection between the Matter device and the hub did not work could not be found.

References

Baray, E. & Ojha, N. K. (2021) 'WLAN security protocols and WPA3 security approach measurement through aircrack-ng technique', *5th International Conference on Computing Methodologies and Communication.* Erode, India, 08-10 April. IEEE. 23-30. Available from: https://ieeexplore.ieee.org/abstract/document/9418230 [Accessed 19 July 2023].

Combs, G. (N.D.) About Wireshark. Available from: https://www.wireshark.org/about.html [Accessed 17 July 2023].

CSA (2022c) Matter Specification Version 1.0. Available from: https://csa-iot.org/wp-content/uploads/2022/11/22-27349-001_Matter-1.0-Core-Specification.pdf [Accessed 25 April 2023].

Herrero, R. (2022) *Fundamentals of IoT Communication Technologies*. Cham: Springer. Available from: https://link.springer.com/content/pdf/10.1007/978-3-030-70080-5.pdf [Accessed 21 April 2023].

IEEE Standards Association (2021) IEEE Standard for Information Technology--Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks--Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Available from: https://standards.ieee.org/ieee/802.11/7028/ [Accessed 13 April 2023].

Kajor, M., Siu, I., Turon, M., Litvin, A., Zbarsky, B., Lunde, G. S., & Smith, M. (2023) Working with the CHIP Tool. Available from: https://github.com/project-chip/connectedhomeip/blob/master/docs/guides/chip_tool_guide.md [Accessed 13 July 2023].

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

Kristiyanto, Y. & Ernastuti, E. (2020) Analysis of deauthentication attack on ieee 802.11 connectivity based on iot technology using external penetration test. *Communication and Information Technology Journal* 14(1): 45-51. Available from: https://journal.binus.ac.id/index.php/commit/article/view/6337/3826 [Accessed 13 June 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].

Noman, H. A., Abdullah, S. M. & Mohammed, H. I. (2015) An automated approach to detect deauthentication and disassociation dos attacks on wireless 802.11 networks. *International Journal of Computer Science Issues* 12(4): 107-112. Available from: https://www.researchgate.net/profile/Haydar-Mohammed/publication/283354063_An_Automated_Approach_to_Detect_Deauthentication_and_Disassociation_Dos_Attacks_on_Wireless_80211_Networks/links/563729f508ae758841

151f49/An-Automated-Approach-to-Detect-Deauthentication-and-Disassociation-Dos-Attacks-on-Wireless-80211-Networks.pdf [Accessed 13 June 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].

Scapy (N.D.) What is Scapy? Available from: https://scapy.net/ [Accessed 13 July 2023].