>>> How Do I "BLE Hacking"?

Jose Gutierrez & Ben Ramsey

>>> Make Nicolas Cage Rich Again

Jose Gutierrez & Ben Ramsey

>>> whoami

* Jose Gutierrez
  - Researcher, Merculite Security
  - BS in Electrical & Computer Engineering
  - Perpetual dabbler
  - Latest dabbles: Home automation, BLE security

* Ben Ramsey
  - Research Director, Merculite Security
  - PhD in Computer Science
  - Wireless geek
  - Recent work: Z-Wave attacks
    -DerbyCon 2015
    -ShmooCon 2016
    -PoC||GTFO 0x12

>>> Housekeeping

* What I will be presenting:
  - Basic overview of Bluetooth Low Energy (BLE)
  - Techniques that have worked for our team
  - Which tools to use and when to use them
* What I won't be presenting:
  - In-depth, highly detailed breakdown of BLE
  - Every BLE tool under the sun

>>> Housekeeping

* What I will be presenting:
  - Basic overview of Bluetooth Low Energy (BLE)
  - Techniques that have worked for our team
  - Which tools to use and when to use them
* What I won't be presenting:
  - In-depth, highly detailed breakdown of BLE
  - Every BLE tool under the sun
* Some nomenclature...
  - "Tablet" = Smartphone, tablet, phablet, etc.
  - "Kali" = your favorite Linux distro with the BlueZ bluetooth stack

>>> A story...

Bluetooth Low Energy (aka Bluetooth Smart) is a forefront technology in the realm of "smart" devices and IoT. It's pretty easy to use and implement.

>>> A story...

We knew this. What we didn't know was how much it was actually being used.

>>> A story...

We knew this. What we didn't know was how much it was actually being used. People put that shit on everything.

>>> A story...

Some of those things may have real security implications...
say, a door lock?

>>> A story...

Some of those things may have real security implications...
say, a door lock? An industrial sensor?

>>> A story...

We chose to focus on one particular sensor: the Onset MX1101 temperature and humidity data logger.

>>> A story...

The manufacturer was proud to include testimonials from people using it. One of them caught my eye.

"The HOBO MX1101 is an ideal solution for us to protect such an important historical artifact as the Magna Carta. It's great that I can check the current conditions at a glance, and have the ability to access historical data without interfering with the exhibit."

Emily Naish, Archivist at Salisbury Cathedral

Read Story

>>> A story...

There's no way they're ACTUALLY relying on this thing, right?

>>> A story...

There's no way they're ACTUALLY relying on this thing, right?

>>> A story...

There's no way they're ACTUALLY relying on this thing, right?

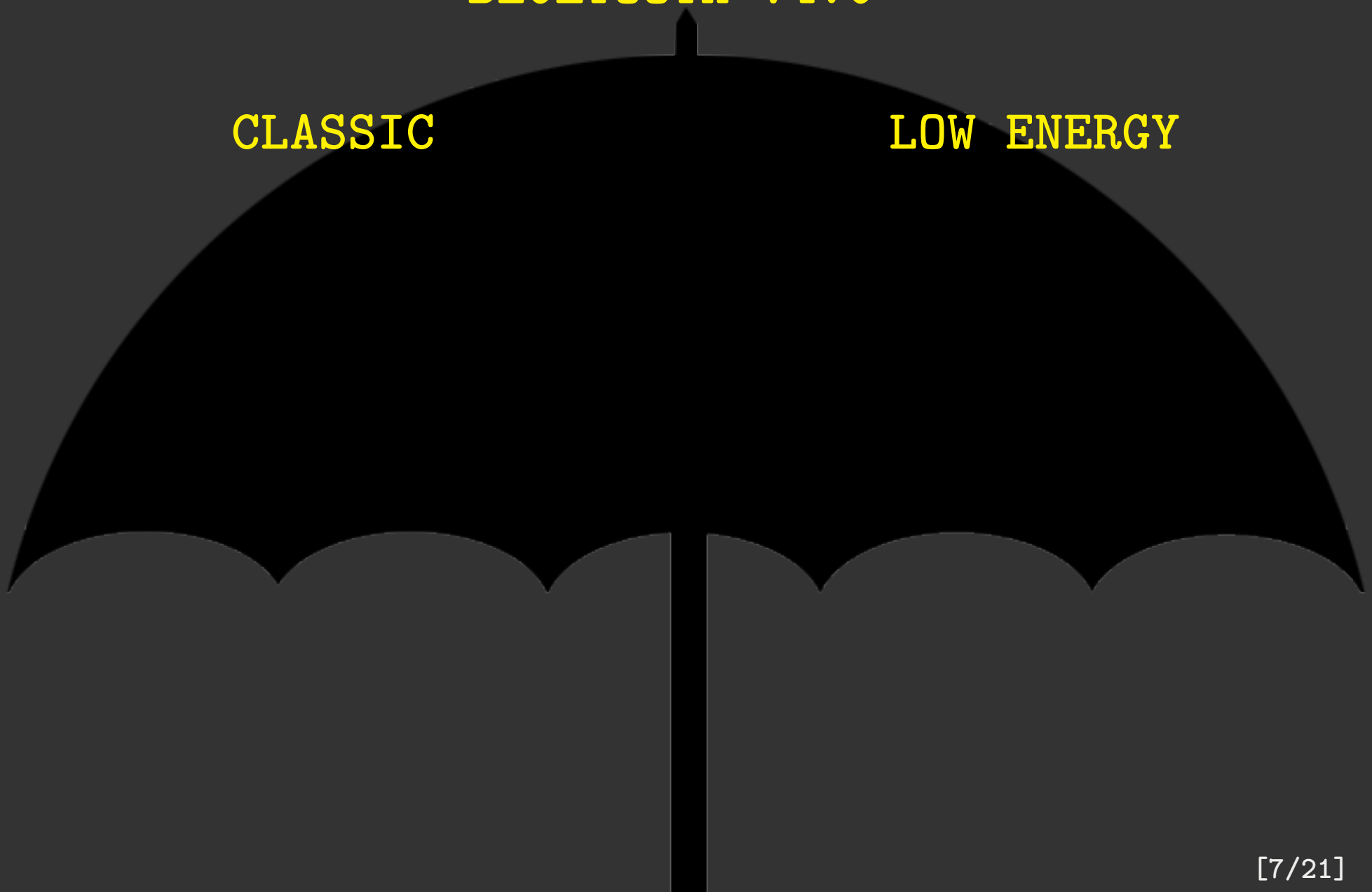
>>> Meet today's target

* MX1101 Data Logger (R57-72)
  - Logs temperature
  - Logs relative humidity
  - Alarms if parameters out of range
  - Uses a password for security

>>> Meet today's target

* MX1101 Data Logger (R57-72)
  - Logs temperature
  - Logs relative humidity
  - Alarms if parameters out of range
  - Uses a password for security
* HOBOMobile App (v1.5.1)
  - Free on App Store, Google Play
  - Configure device
  - Read logged data
  - Update firmware over-the-air

>>> Bluetooth Classic vs Bluetooth Low Energy



>>> Bluetooth Classic vs Bluetooth Low Energy

BLUETOOTH v4.0+

CLASSIC

* High overhead
* Faster data rate
* Good for:
  - Listening to your sick new Bieber album
  - Sharing your cell phone's internet with strangers

LOW ENERGY

>>> Bluetooth Classic vs Bluetooth Low Energy

BLUETOOTH v4.0+

CLASSIC

* High overhead
* Faster data rate
* Good for:
  - Listening to your sick new Bieber album
  - Sharing your cell phone's internet with strangers

LOW ENERGY

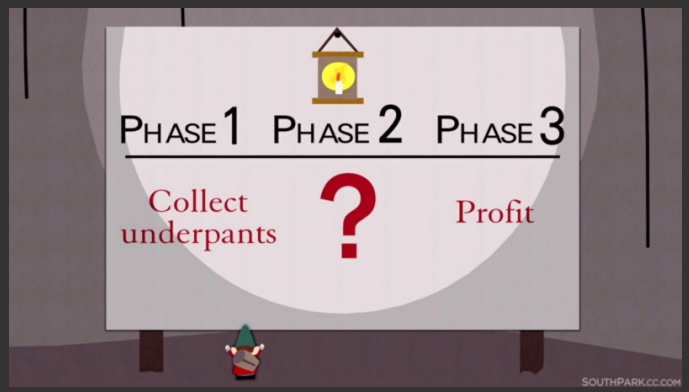
* Different protocol!
* Very low power/overhead
* Slower data rate
* Good for:
  - "IoT"-ing your house
  - Sharing & controlling "state" of things
* A.k.a.
  - Bluetooth Smart
  - BLE

>>> Our basic process

1. Select the target
   - Perform active scan
   - Passively sniff advertisement packets
2. Enumerate its services and characteristics
   - Sniff service discovery messages
   - Connect to the target yourself
3. Reverse the application protocol (fun!)
   - Analyze legitimate traffic
   - Impersonate devices

>>> Our basic process

1. Select the target
   - Perform active scan
   - Passively sniff advertisement packets
2. Enumerate its services and characteristics
   - Sniff service discovery messages
   - Connect to the target yourself
3. Reverse the application protocol (fun!)
   - Analyze legitimate traffic
   - Impersonate devices
4. Pillage, plunder, pwn (more fun!!)
   - ...profit!



>>> Selecting the target

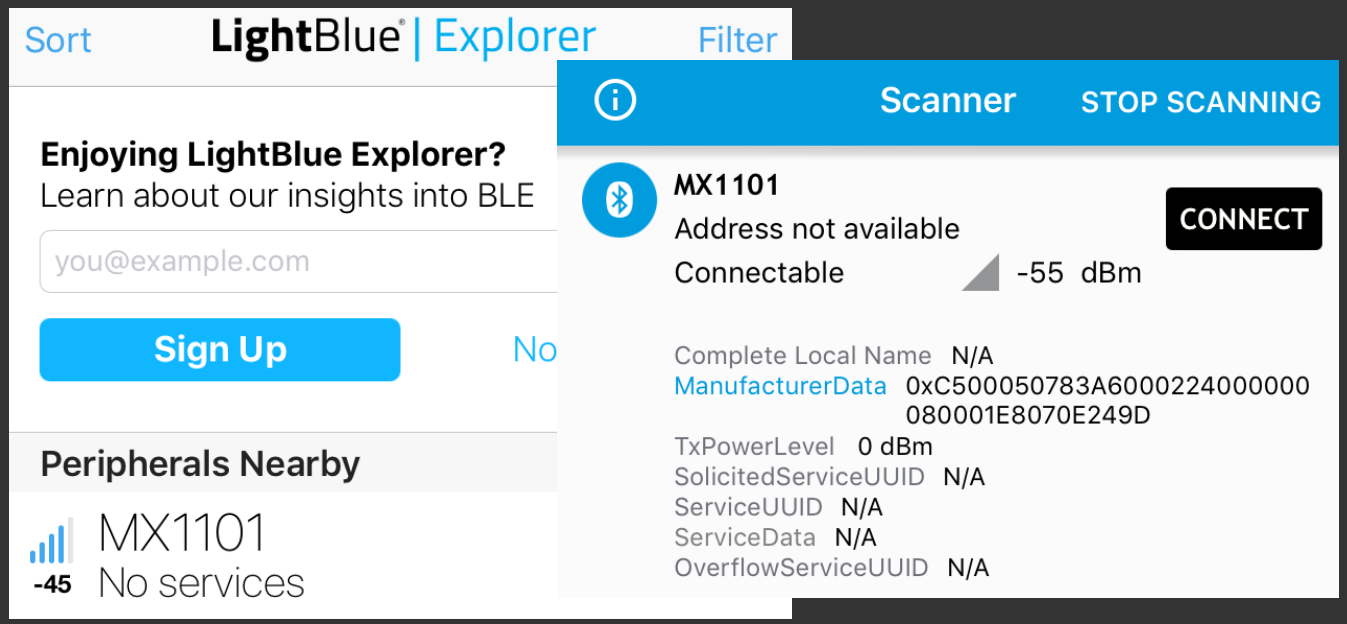
Our goal: Find BLE devices

* All devices use special channels to advertise:
  - Connectability
  - Available services or manufacturer-specific data

>>> Selecting the target

Our goal: Find BLE devices

* All devices use special channels to advertise:
  - Connectability
  - Available services or manufacturer-specific data
* On tablet: use *LightBlue* (iOS) or *nRFConnect* (all)



>>> Selecting the target

Our goal: Find BLE devices

* All devices use special channels to advertise:
  - Connectability
  - Available services or manufacturer-specific data
* On tablet: use *LightBlue* (iOS) or *nRFConnect* (all)
* On Kali: actively scan with *hcitool lescan* or listen for advertisements with *--passive* and *btmon*

```
root@kali:~# hciconfig hci0 up
root@kali:~# hcitool lescan
LE Scan ...
D0:03:4B:ED:D4:68 (unknown)
D0:03:4B:ED:D4:68 (unknown)
52:3F:A1:A2:E4:27 (unknown)
52:3F:A1:A2:E4:27 (unknown)
D1:55:D5:00:68:C7 (unknown)
D1:55:D5:00:68:C7 (unknown)
```

```
> HCI Event: LE Meta Event (0x3e) plen 40
      LE Advertising Report (0x02)
        Num reports: 1
        Event type: Connectable undirected - ADV_IND (0x00)
        Address type: Random (0x01)
        Address: D1:55:D5:00:68:C7 (Static)
        Data length: 28
        Flags: 0x06
          LE General Discoverable Mode
          BR/EDR Not Supported
        TX power: 0 dBm
        Company: Onset Computer Corporation (197)
          Data: 050783a60002240200000800011b08fd259d
        RSSI: -68 dBm (0xbc)
```

>>> Selecting the target

Our goal: Find BLE devices

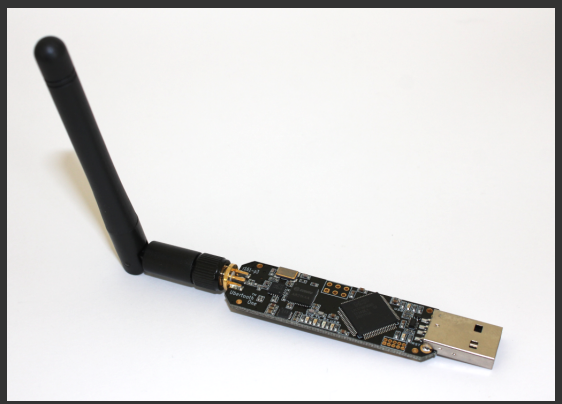
* All devices use special channels to advertise:
  - Connectability
  - Available services or manufacturer-specific data
* On tablet: use *LightBlue* (iOS) or *nRFConnect* (all)
* On Kali: actively scan with *hcitool lescan* or listen for advertisements with *--passive* and *btmon*

If all else fails, use a sniffer.

>>> Smells like traffic

We've tried out three different sniffers:

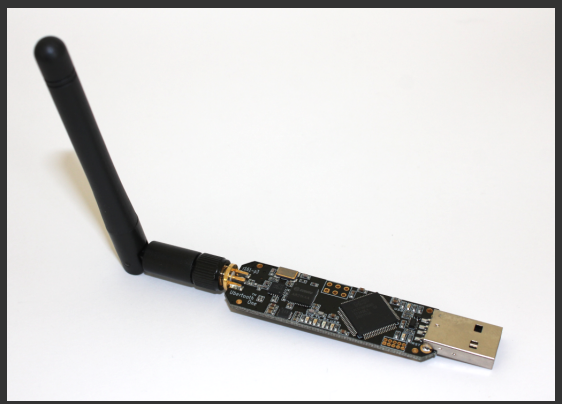
* Ubertooth One ($109 if pre-built)
  - Can sniff both Classic and BLE
  - Open source hardware and firmware
  - Feeds directly into Wireshark
  - Swap antenna for distance level-up

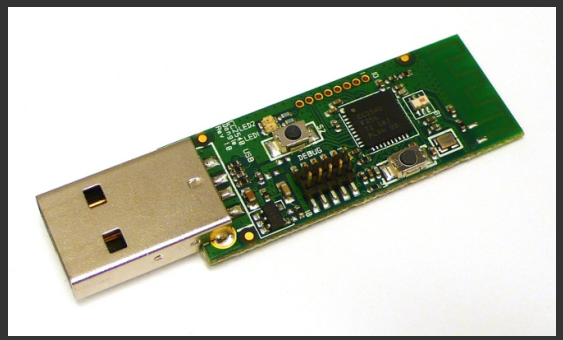


>>> Smells like traffic

We've tried out three different sniffers:

* Ubertooth One ($109 if pre-built)
* TI CC2540 USB Evaluation Kit ($49)
  - Uses the SmartRF Packet Sniffer software
  - Bulky interface, limited configurability

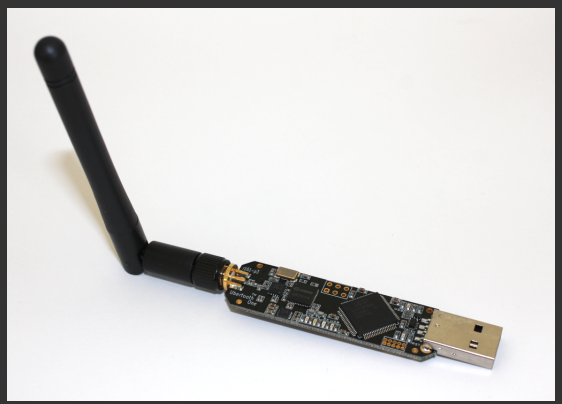


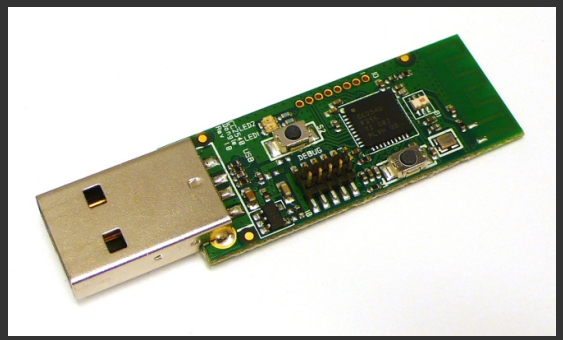


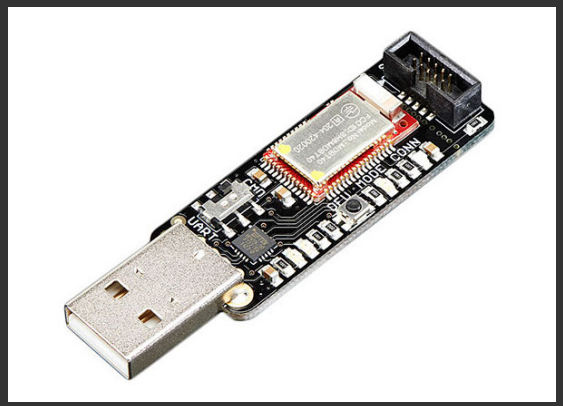
>>> Smells like traffic

We've tried out three different sniffers:

* Ubertooth One ($109 if pre-built)
* TI CC2540 USB Evaluation Kit ($49)
* Bluefruit LE Sniffer ($30)
  - Lots of Windows support; not much on Linux
  - Custom tools can feed into Wireshark
  - Driver comes pre-installed in Kali
  - Kali Wireshark plugin instructions on our repo



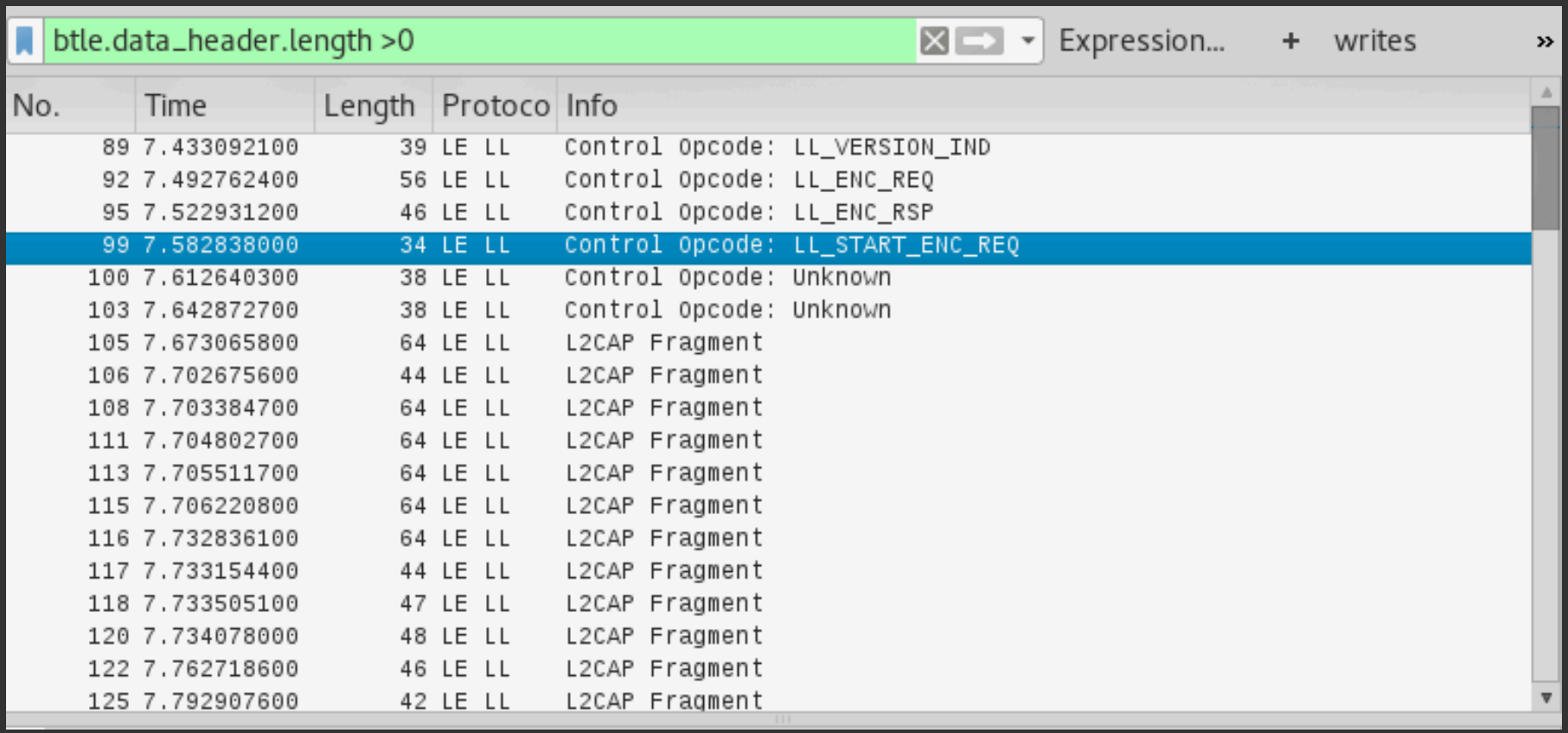




>>> A note on encryption

Your traffic may be encrypted if:

* Device forces you to "pair" with it
* LL_START_ENC_REQ at the top of the capture
* Sniffed traffic contains a ton of L2CAP Fragments



>>> A note on encryption

Mike Ryan gave us *crackle*, which can decrypt traffic if:

1. You capture the full pairing process
2. Pairing mode is "6-digit PIN" or "Just Works"

>>> A note on encryption

Mike Ryan gave us *crackle*, which can decrypt traffic if:

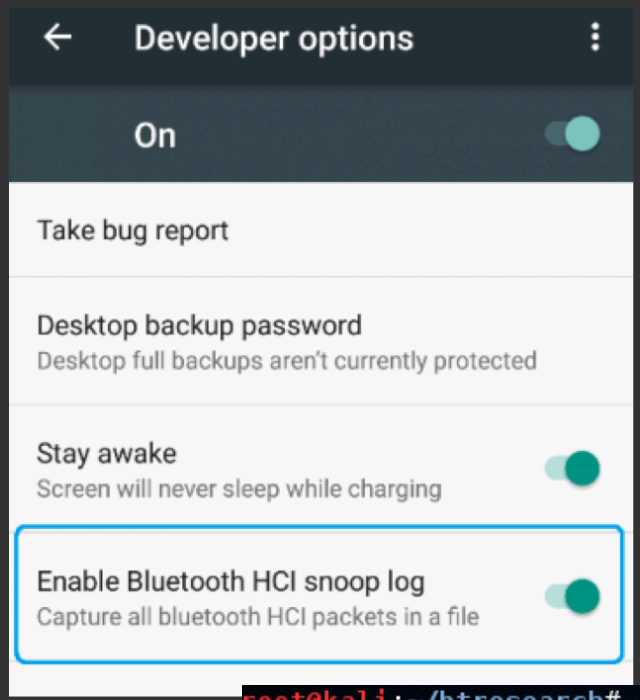
1. You capture the full pairing process
2. Pairing mode is "6-digit PIN" or "Just Works"

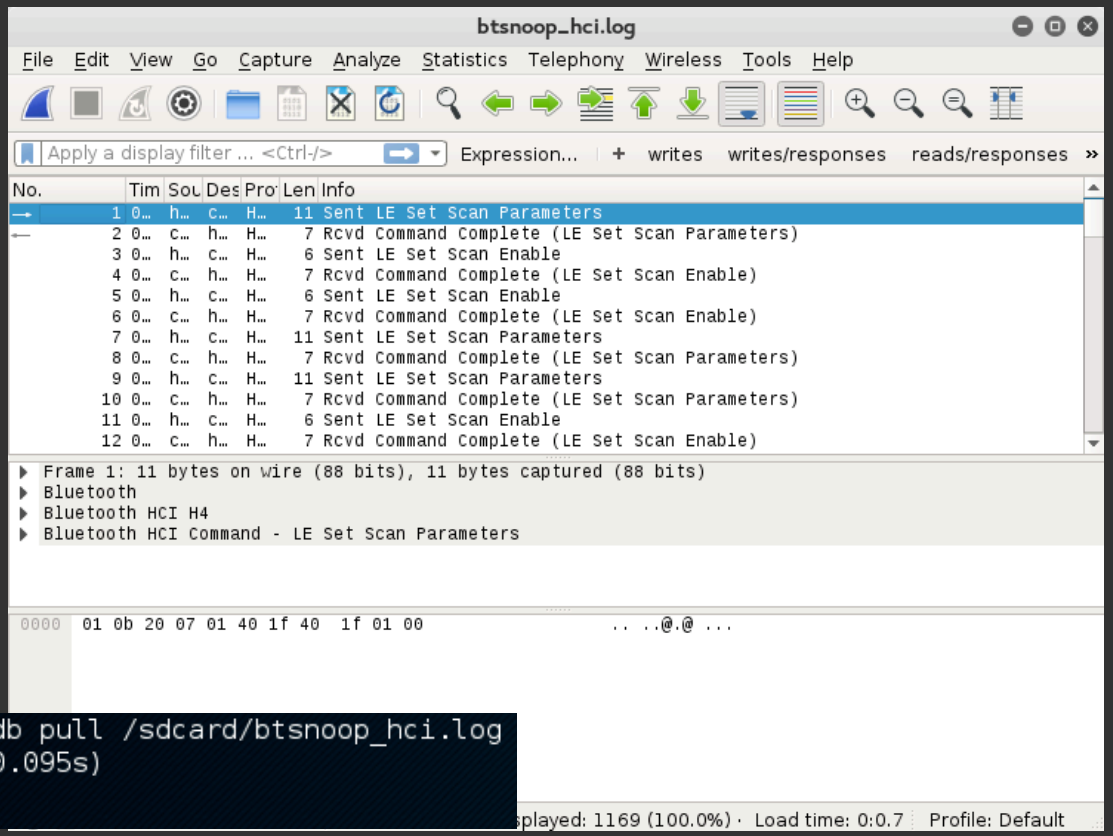
Bluetooth v4.2 added "Numerical Comparison" mode as a countermeasure. However, most devices don't use encryption, or they encrypt at a higher layer.

>>> An alternate "sniffer"

We prefer to bypass link layer encryption altogether...

* Enable Android developer options
* Select "Enable Bluetooth HCI snoop log"
* Grab */sdcard/btsnoop_hci.log* using *adb*
* Open in Wireshark






```
root@kali:~/btresearch# adb pull /sdcard/btsnoop_hci.log
583 KB/s (57238 bytes in 0.095s)
root@kali:~/btresearch#
```

>>> Enumerate

Our goal: Determine what services are running on the target

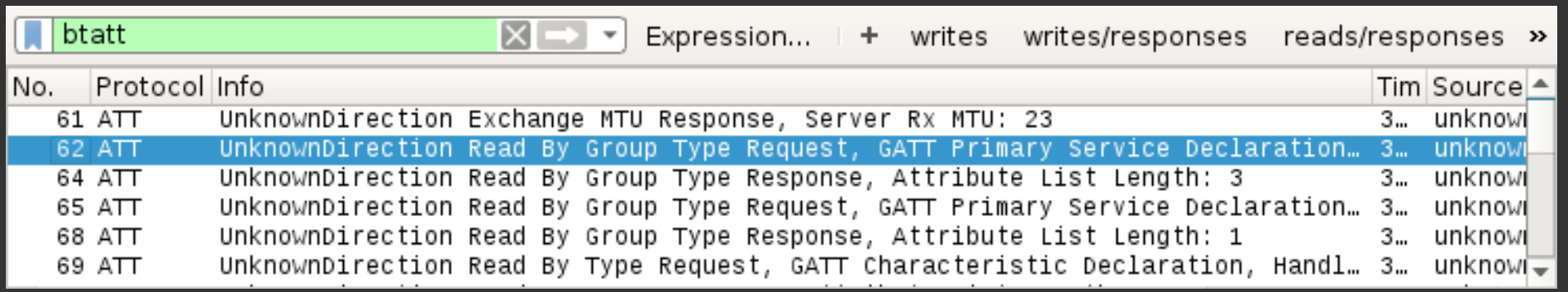
* Active - connect to the device
  - On tablet: connect with nRFConnect to automatically enumerate
  - On Kali: use *gatttool* commands *primary* and *characteristics*

```
[D1:55:D5:00:68:C7][LE]> primary
attr handle: 0x0001, end grp handle: 0x0007 uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle: 0x0008, end grp handle: 0x000b uuid: 00001801-0000-1000-8000-00805f9b34fb
attr handle: 0x000c, end grp handle: 0x0016 uuid: 0000180a-0000-1000-8000-00805f9b34fb
attr handle: 0x0017, end grp handle: 0xffff uuid: 65e16e4f-ed4e-4641-ac49-83ccbce6cbcf
[D1:55:D5:00:68:C7][LE]> characteristics
handle: 0x0002, char properties: 0x0a, char value handle: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0004, char properties: 0x02, char value handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb
handle: 0x0006, char properties: 0x02, char value handle: 0x0007, uuid: 00002a04-0000-1000-8000-00805f9b34fb
handle: 0x0009, char properties: 0x20, char value handle: 0x000a, uuid: 00002a05-0000-1000-8000-00805f9b34fb
handle: 0x000d, char properties: 0x02, char value handle: 0x000e, uuid: 00002a29-0000-1000-8000-00805f9b34fb
handle: 0x000f, char properties: 0x02, char value handle: 0x0010, uuid: 00002a24-0000-1000-8000-00805f9b34fb
handle: 0x0011, char properties: 0x02, char value handle: 0x0012, uuid: 00002a25-0000-1000-8000-00805f9b34fb
handle: 0x0013, char properties: 0x02, char value handle: 0x0014, uuid: 00002a26-0000-1000-8000-00805f9b34fb
handle: 0x0015, char properties: 0x02, char value handle: 0x0016, uuid: 00002a28-0000-1000-8000-00805f9b34fb
handle: 0x0018, char properties: 0x18, char value handle: 0x0019, uuid: 65e16f4f-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001b, char properties: 0x0a, char value handle: 0x001c, uuid: 65e16f50-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001d, char properties: 0x18, char value handle: 0x001e, uuid: 65e16f52-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x0020, char properties: 0x0a, char value handle: 0x0021, uuid: 65e16f53-ed4e-4641-ac49-83ccbce6cbcf
```

>>> Enumerate

Our goal: Determine what services are running on the target

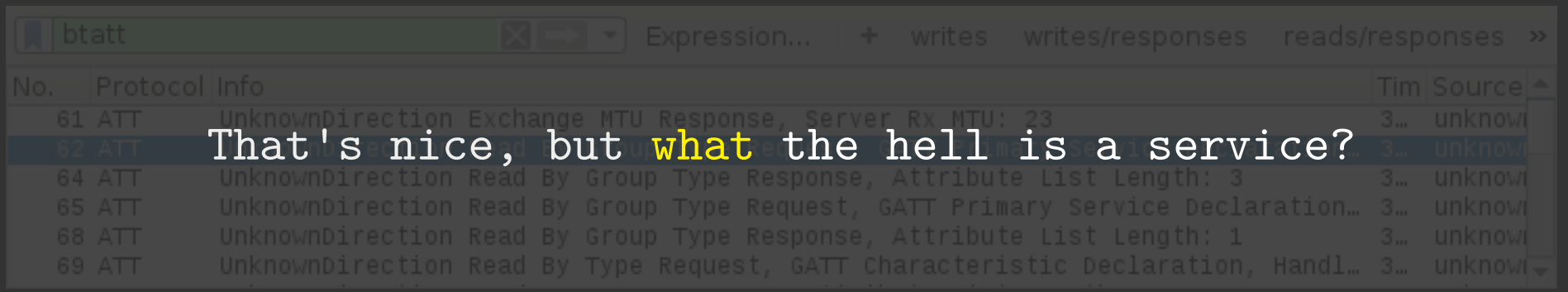
* Active - connect to the device
  - On tablet: connect with nRFConnect to automatically enumerate
  - On Kali: use *gatttool* commands *primary* and *characteristics*
* Passive - listen for service discovery messages
  - Sniff someone else's connection
  - Note service and characteristic handle ranges



>>> Enumerate

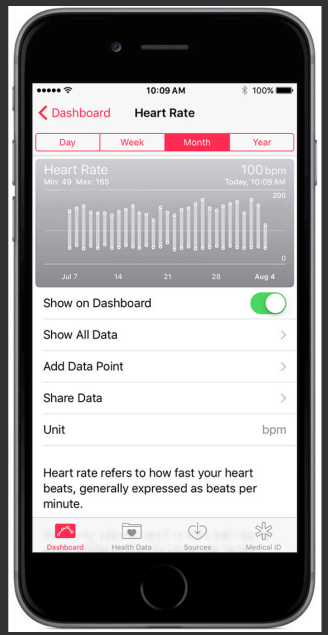
Our goal: Determine what services are running on the target

* Active - connect to the device
  - On tablet: connect with nRFConnect to automatically enumerate
  - On Kali: use *gatttool* commands *primary* and *characteristics*
* Passive - listen for service discovery messages
  - Sniff someone else's connection
  - Note service and characteristic handle ranges



>>> Let's pretend we're working out.

BLE Client



Hey! Whats the heart rate?

116 bpm

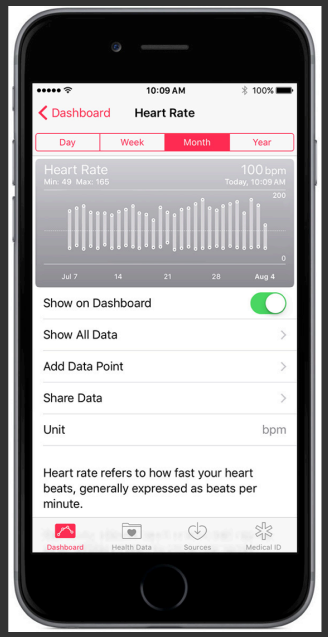
BLE Server

The app on the phone uses a BLE connection to request a heart rate reading from the heart rate monitor (HRM).

>>> Let's pretend we're working out.

BLE Server

BLE Client



Hey! Whats the heart rate?

116 bpm

Required BLE Service 1

Required BLE Service 2

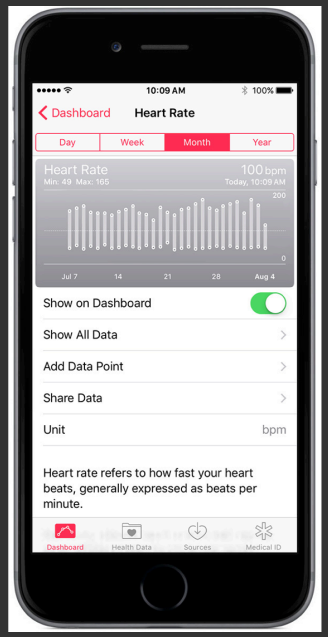
Heart Rate Service

The BLE client uses a BLE connection to communicate with the heart rate service on the BLE server

>>> Let's pretend we're working out.

BLE Server

BLE Client



Hey! Whats the heart rate?

116 bpm

Heart Rate Service

Heart Rate Measurement Char.

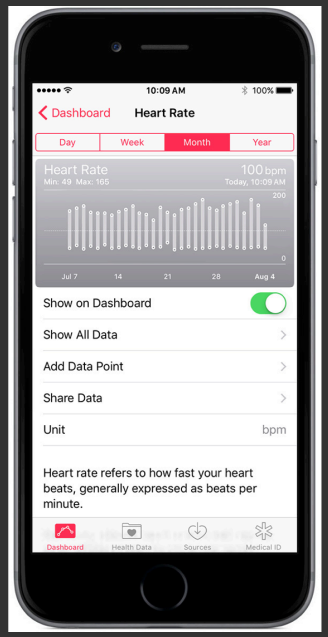
Body Placement Char.

The BLE client uses a BLE connection to read the Heart Rate Measurement Characteristic within the heart rate service on the BLE server.

>>> Let's pretend we're working out.

BLE Server

BLE Client



Hey! Whats the heart rate?

116 bpm

Heart Rate Measurement Char.

Declaration Attribute

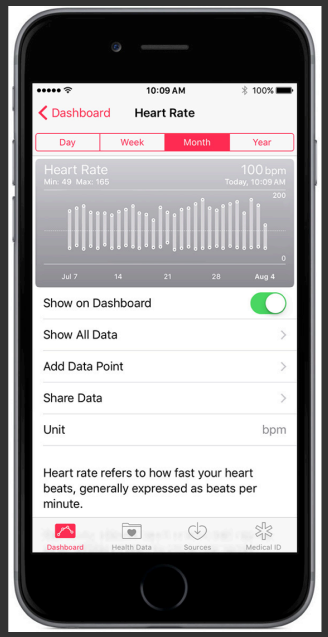
Value Attribute

The BLE client uses a BLE connection to send a Read Request to the Value Attribute of the heart rate characteristic within the heart rate service on the BLE server.

>>> Let's pretend we're working out.

BLE Server

BLE Client



Hey! Whats the heart rate?

116 bpm

- Attribute 1
- Attribute 2
- Attribute 3
- Attribute 4
- Attribute 5
- Attribute 6
- Attribute 7
- Attribute 8
- Attribute 9
- ⋮

In fact, the whole server is simply a collection of Attributes. The Generic Attribute (GATT) Profile interprets the meaning of any given attribute based on its UUID.

>>> MX1101 Services

Here's how it looks for the MX1101 data logger:

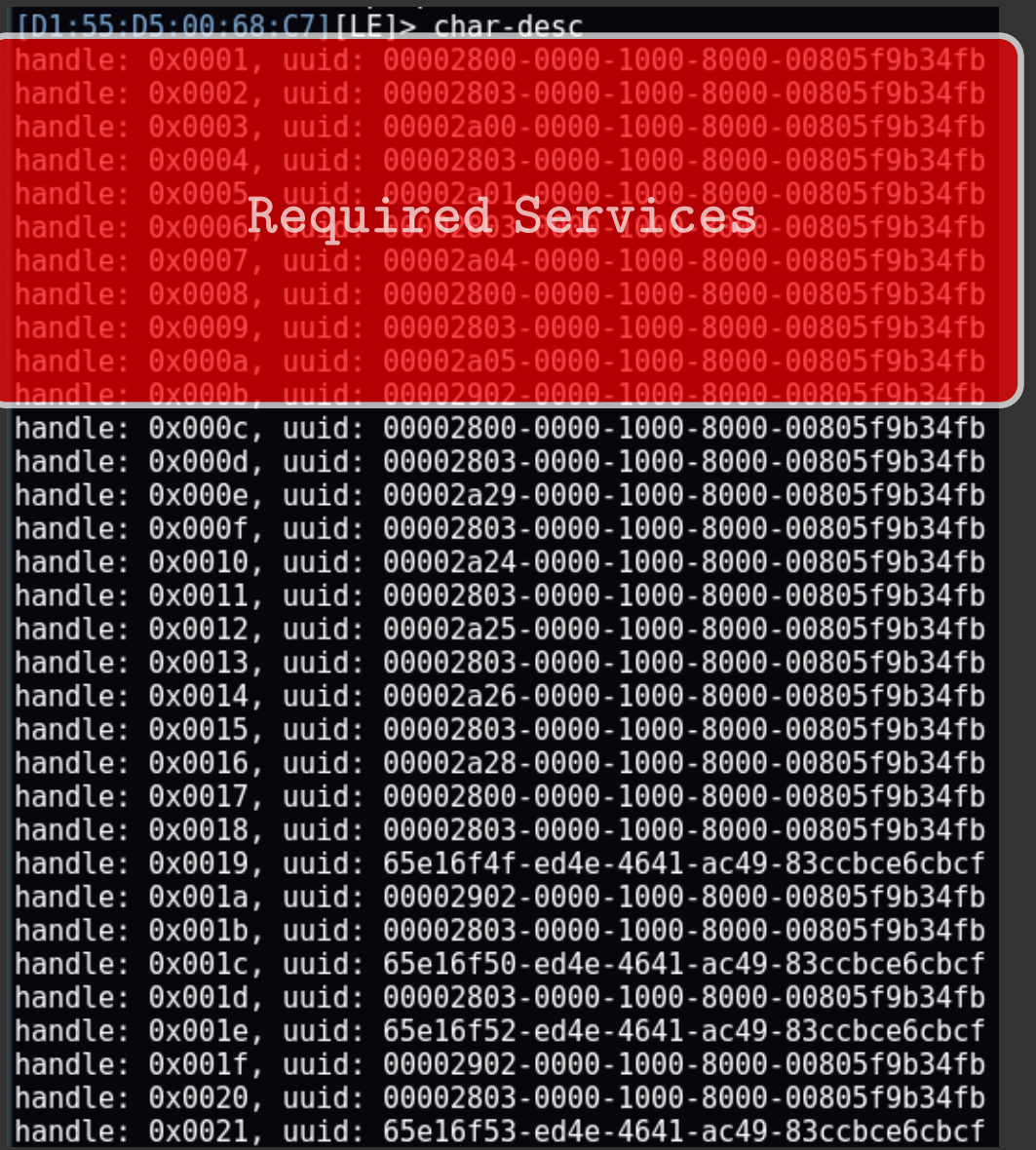
This is a list of all attributes on the BLE server.

```
[D1:55:D5:00:68:C7][LE]> char-desc
handle: 0x0001, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0002, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0004, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb
handle: 0x0006, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0007, uuid: 00002a04-0000-1000-8000-00805f9b34fb
handle: 0x0008, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0009, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000a, uuid: 00002a05-0000-1000-8000-00805f9b34fb
handle: 0x000b, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x000c, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x000d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000e, uuid: 00002a29-0000-1000-8000-00805f9b34fb
handle: 0x000f, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0010, uuid: 00002a24-0000-1000-8000-00805f9b34fb
handle: 0x0011, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0012, uuid: 00002a25-0000-1000-8000-00805f9b34fb
handle: 0x0013, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0014, uuid: 00002a26-0000-1000-8000-00805f9b34fb
handle: 0x0015, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0016, uuid: 00002a28-0000-1000-8000-00805f9b34fb
handle: 0x0017, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0018, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0019, uuid: 65e16f4f-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001a, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x001b, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001c, uuid: 65e16f50-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001e, uuid: 65e16f52-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001f, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x0020, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0021, uuid: 65e16f53-ed4e-4641-ac49-83ccbce6cbcf
```

>>> MX1101 Services

Here's how it looks for the MX1101 data logger:

These top 11 attributes define the two required BLE services.



>>> MX1101 Services

Here's how it looks for the MX1101 data logger:

These next 11 attributes define the Device ID Service. It's one of the standard services defined by the Bluetooth SIG.

```
[D1:55:D5:00:68:C7][LE]> char-desc
handle: 0x0001, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0002, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0004, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb
handle: 0x0006, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0007, uuid: 00002a04-0000-1000-8000-00805f9b34fb
handle: 0x0008, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0009, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000a, uuid: 00002a05-0000-1000-8000-00805f9b34fb
handle: 0x000b, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x000c, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x000d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000e, uuid: 00002a29-0000-1000-8000-00805f9b34fb
handle: 0x000f, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0010, uuid: 00002a24-0000-1000-8000-00805f9b34fb
handle: 0x0011, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0012, uuid: 00002a25-0000-1000-8000-00805f9b34fb
handle: 0x0013, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0014, uuid: 00002a26-0000-1000-8000-00805f9b34fb
handle: 0x0015, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0016, uuid: 00002a28-0000-1000-8000-00805f9b34fb
handle: 0x0017, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0018, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0019, uuid: 65e16f4f-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001a, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x001b, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001c, uuid: 65e16f50-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001e, uuid: 65e16f52-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001f, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x0020, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0021, uuid: 65e16f53-ed4e-4641-ac49-83ccbce6cbcf
```

Device ID Service

>>> MX1101 Services

Here's how it looks for the MX1101 data logger:

Some of its characteristics include a Manufacturer Name String,

```
[D1:55:D5:00:68:C7][LE]> char-desc
handle: 0x0001, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0002, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0004, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb
handle: 0x0006, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0007, uuid: 00002a04-0000-1000-8000-00805f9b34fb
handle: 0x0008, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0009, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000a, uuid: 00002a05-0000-1000-8000-00805f9b34fb
handle: 0x000b, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x000c, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x000d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000e, uuid: 00002a29-0000-1000-8000-00805f9b34fb
handle: 0x000f, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0010, uuid: 00002a24-0000-1000-8000-00805f9b34fb
handle: 0x0011, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0012, uuid: 00002a25-0000-1000-8000-00805f9b34fb
handle: 0x0013, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0014, uuid: 00002a26-0000-1000-8000-00805f9b34fb
handle: 0x0015, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0016, uuid: 00002a28-0000-1000-8000-00805f9b34fb
handle: 0x0017, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0018, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0019, uuid: 65e16f4f-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001a, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x001b, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001c, uuid: 65e16f50-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001e, uuid: 65e16f52-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001f, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x0020, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0021, uuid: 65e16f53-ed4e-4641-ac49-83ccbce6cbcf
```

>>> MX1101 Services

Here's how it looks for the MX1101 data logger:

Some of its characteristics include a Manufacturer Name String, and a Model Number String.

```
[D1:55:D5:00:68:C7][LE]> char-desc
handle: 0x0001, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0002, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0004, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb
handle: 0x0006, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0007, uuid: 00002a04-0000-1000-8000-00805f9b34fb
handle: 0x0008, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0009, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000a, uuid: 00002a05-0000-1000-8000-00805f9b34fb
handle: 0x000b, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x000c, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x000d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000e, uuid: 00002a29-0000-1000-8000-00805f9b34fb
handle: 0x000f, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0010, uuid: 00002a24-0000-1000-8000-00805f9b34fb
handle: 0x0011, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0012, uuid: 00002a25-0000-1000-8000-00805f9b34fb
handle: 0x0013, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0014, uuid: 00002a26-0000-1000-8000-00805f9b34fb
handle: 0x0015, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0016, uuid: 00002a28-0000-1000-8000-00805f9b34fb
handle: 0x0017, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0018, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0019, uuid: 65e16f4f-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001a, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x001b, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001c, uuid: 65e16f50-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001e, uuid: 65e16f52-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001f, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x0020, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0021, uuid: 65e16f53-ed4e-4641-ac49-83ccbce6cbcf
```

>>> MX1101 Services

Here's how it looks for the MX1101 data logger:

These last 11 attributes define an application-specific service created by the manufacturer.

```
[D1:55:D5:00:68:C7][LE]> char-desc
handle: 0x0001, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0002, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0004, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb
handle: 0x0006, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0007, uuid: 00002a04-0000-1000-8000-00805f9b34fb
handle: 0x0008, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0009, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000a, uuid: 00002a05-0000-1000-8000-00805f9b34fb
handle: 0x000b, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x000c, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x000d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000e, uuid: 00002a29-0000-1000-8000-00805f9b34fb
handle: 0x000f, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0010, uuid: 00002a24-0000-1000-8000-00805f9b34fb
handle: 0x0011, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0012, uuid: 00002a25-0000-1000-8000-00805f9b34fb
handle: 0x0013, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0014, uuid: 00002a26-0000-1000-8000-00805f9b34fb
handle: 0x0015, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0016, uuid: 00002a28-0000-1000-8000-00805f9b34fb
handle: 0x0017, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0018, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0019, uuid: 65e16f4f-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001a, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x001b, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001c, uuid: 65e16f50-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001e, uuid: 65e16f52-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001f, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x0020, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0021, uuid: 65e16f53-ed4e-4641-ac49-83ccbce6cbcf
```

Application-Specific Service

>>> MX1101 Services

Here's how it looks for the MX1101 data logger:

```
[D1:55:D5:00:68:C7][LE]> char-desc
handle: 0x0001, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0002, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0004, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb
handle: 0x0006, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0007, uuid: 00002a04-0000-1000-8000-00805f9b34fb
handle: 0x0008, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0009, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000a, uuid: 00002a05-0000-1000-8000-00805f9b34fb
handle: 0x000b, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x000c, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x000d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000e, uuid: 00002a29-0000-1000-8000-00805f9b34fb
handle: 0x000f, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0010, uuid: 00002a24-0000-1000-8000-00805f9b34fb
handle: 0x0011, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0012, uuid: 00002a25-0000-1000-8000-00805f9b34fb
handle: 0x0013, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0014, uuid: 00002a26-0000-1000-8000-00805f9b34fb
handle: 0x0015, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0016, uuid: 00002a28-0000-1000-8000-00805f9b34fb
handle: 0x0017, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0018, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0019, uuid: 65e16f4f-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001a, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x001b, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001c, uuid: 65e16f50-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001e, uuid: 65e16f52-ed4e-4641-ac49-83ccbce6cbcf
handle: 0x001f, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x0020, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0021, uuid: 65e16f53-ed4e-4641-ac49-83ccbce6cbcf
```

This characteristic is creatively used as a command and control point. The client writes commands to it, and the characteristic responds with data by using notifications.

>>> Reversing the application

Our goal: Understand the application protocol

* Strategies are endless!
* Generate and analyze legitimate traffic
* Impersonate target using *bleno* (node.js)
  - Create arbitrary BLE server on Kali
  - Implement as much functionality as needed
  - Use captured traffic to your advantage

```
var service_17= new bleno.PrimaryService({
        uuid: '65E16E4F-ED4E-4641-AC49-83CCBCE6CBCF',
        characteristics:[
                new bleno.Characteristic({
                        uuid: '65E16F4F-ED4E-4641-AC49-83CCBCE6CBCF', // Handle: 0x19
                        properties: ['write','notify'],
                        secure: [],
                        value: null,
                        onWriteRequest: function(data,offset,withoutResponse,callback){
                                callback(this.RESULT_SUCCESS);
                                respondHandle19(data,this.updateValueCallback);
                        }
                }),
                new bleno.Characteristic({
                        uuid: '65E16F50-ED4E-4641-AC49-83CCBCE6CBCF',
                        properties: ['read', 'write'],
                        secure: [],
                        value: 4096,
                }),
```

>>> Findings on the MX1101

* Password is sent in cleartext

btatt.opcode==0x12 || btatt.opcode==0x1b Expression... + writes writes/responses »

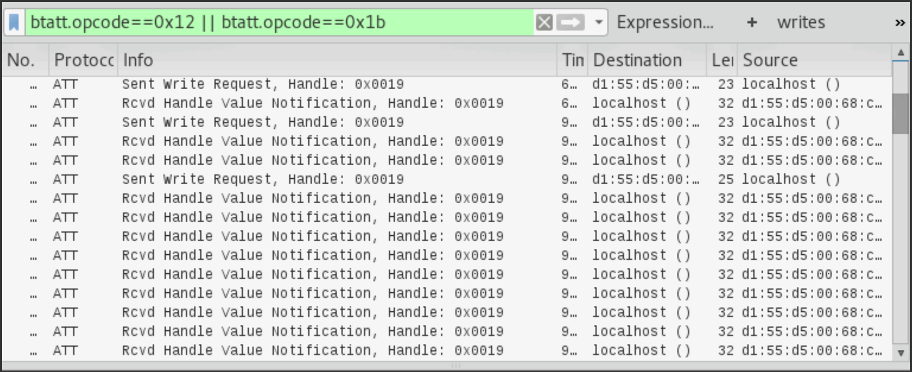
No.	Protocc	Info	Tin	Destination	Lei	Source
58	ATT	Sent Write Request, Handle: 0x001a	2…	d1:55:d5:00:…	14	localhost ()
60	ATT	Sent Write Request, Handle: 0x001f	2…	d1:55:d5:00:…	14	localhost ()
63	ATT	Sent Write Request, Handle: 0x0019	2…	d1:55:d5:00:…	28	localhost ()
65	ATT	Rcvd Handle Value Notification, Handle: 0x0019	2…	localhost ()	32	d1:55:d5:00:68:…
71	ATT	Sent Write Request, Handle: 0x0019	2…	d1:55:d5:00:…	25	localhost ()
74	ATT	Rcvd Handle Value Notification, Handle: 0x0019	2…	localhost ()	32	d1:55:d5:00:68:…
75	ATT	Rcvd Handle Value Notification, Handle: 0x0019	2…	localhost ()	32	d1:55:d5:00:68:…

▶ Bluetooth L2CAP Protocol
▼ Bluetooth Attribute Protocol
  ▶ Opcode: Write Request (0x12)
  Handle: 0x0019
  Value: 01010d051c0168694445464 34f4e2100

```
0000  02 40 00 17 00 13 00 04  00 12 19 00 01 01 0d 05   .@...... ........
0010  1c 01 68 69 44 45 46 43  4f 4e 21 00               ..hiDEFC ON!.
```

>>> Findings on the MX1101

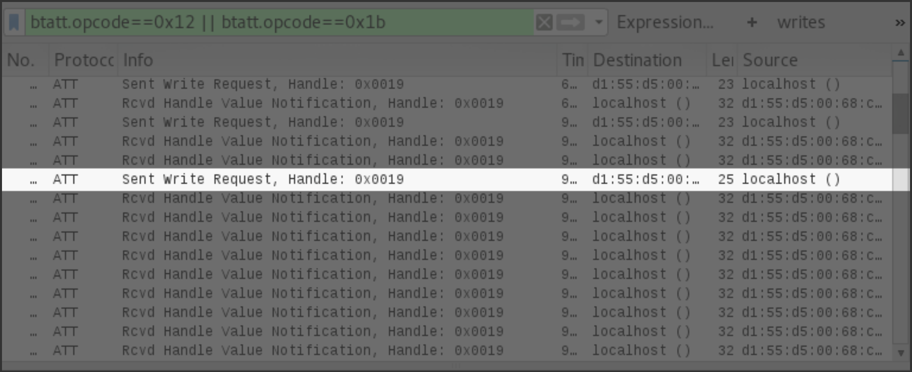
* Password is sent in cleartext
* Custom C2 and data transfer protocols
  - Write commands to handle 0x0019
  - Receive notifications from the same


btatt.opcode==0x12 || btatt.opcode==0x1b   Expression...   +   writes   »

No.	Protocc	Info	Tin	Destination	Lei	Source
…	ATT	Sent Write Request, Handle: 0x0019	6…	d1:55:d5:00:…	23	localhost ()
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	6…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Sent Write Request, Handle: 0x0019	9…	d1:55:d5:00:…	23	localhost ()
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Sent Write Request, Handle: 0x0019	9…	d1:55:d5:00:…	25	localhost ()
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…

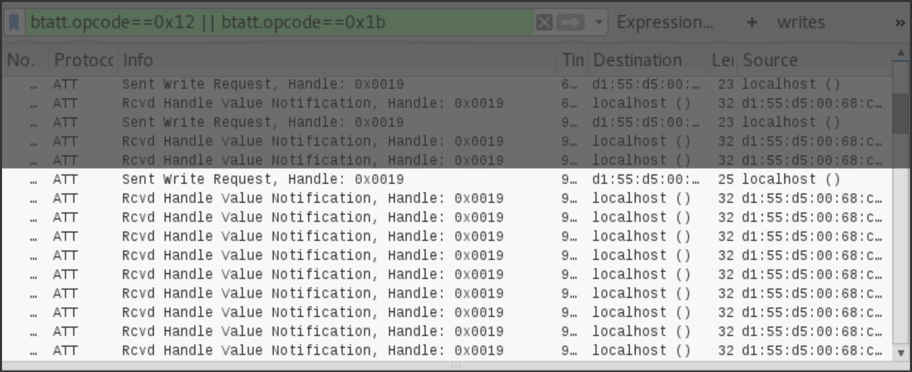
>>> Findings on the MX1101

* Password is sent in cleartext
* Custom C2 and data transfer protocols
  - Write commands to handle 0x0019
  - Receive notifications from the same



>>> Findings on the MX1101

* Password is sent in cleartext
* Custom C2 and data transfer protocols
  - Write commands to handle 0x0019
  - Receive notifications from the same

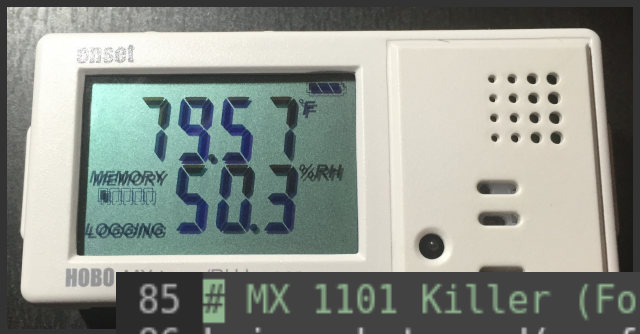


btatt.opcode==0x12 || btatt.opcode==0x1b   Expression...   +   writes

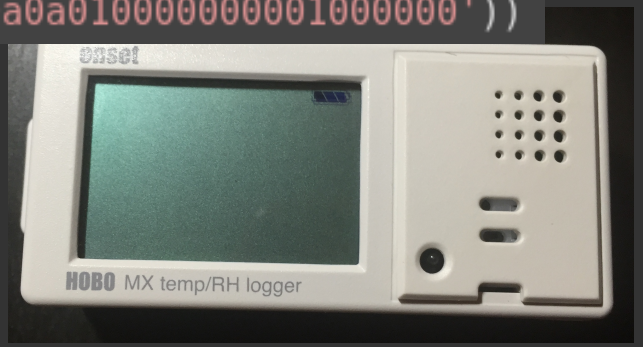
No.	Protocc	Info	Tin	Destination	Lei	Source
…	ATT	Sent Write Request, Handle: 0x0019	6…	d1:55:d5:00:…	23	localhost ()
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	6…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Sent Write Request, Handle: 0x0019	9…	d1:55:d5:00:…	23	localhost ()
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Sent Write Request, Handle: 0x0019	9…	d1:55:d5:00:…	25	localhost ()
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…
…	ATT	Rcvd Handle Value Notification, Handle: 0x0019	9…	localhost ()	32	d1:55:d5:00:68:c…

>>> Findings on the MX1101

* Password is sent in cleartext
* Custom C2 and data transfer protocols
  - Write commands to handle 0x0019
  - Receive notifications from the same
* Some unhandled commands can force logging to stop

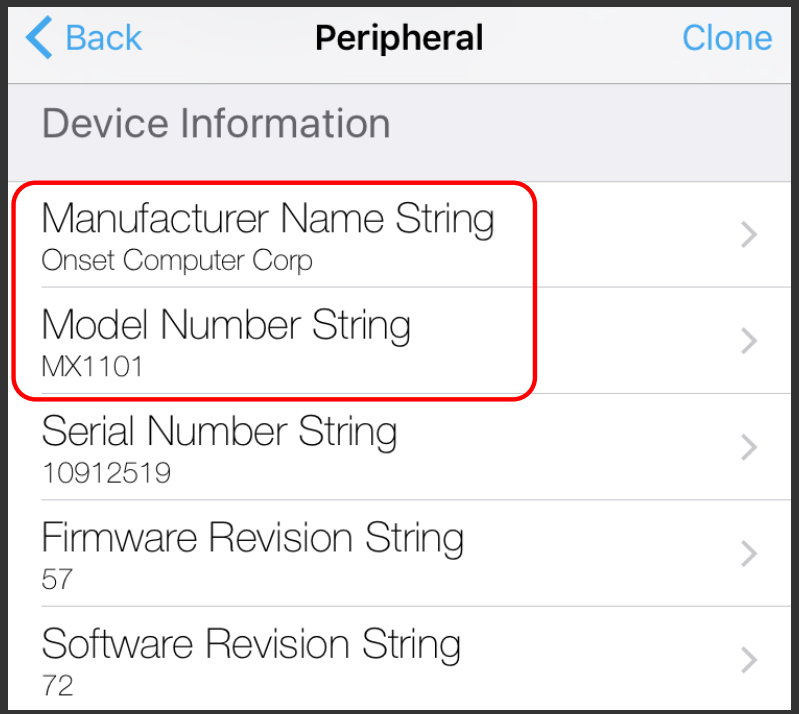



```
85 # MX 1101 Killer (Force Reset)
86 hci_socket.send(craftWriteReq(0x0019,'01010a0a0100000000001000000'))
```



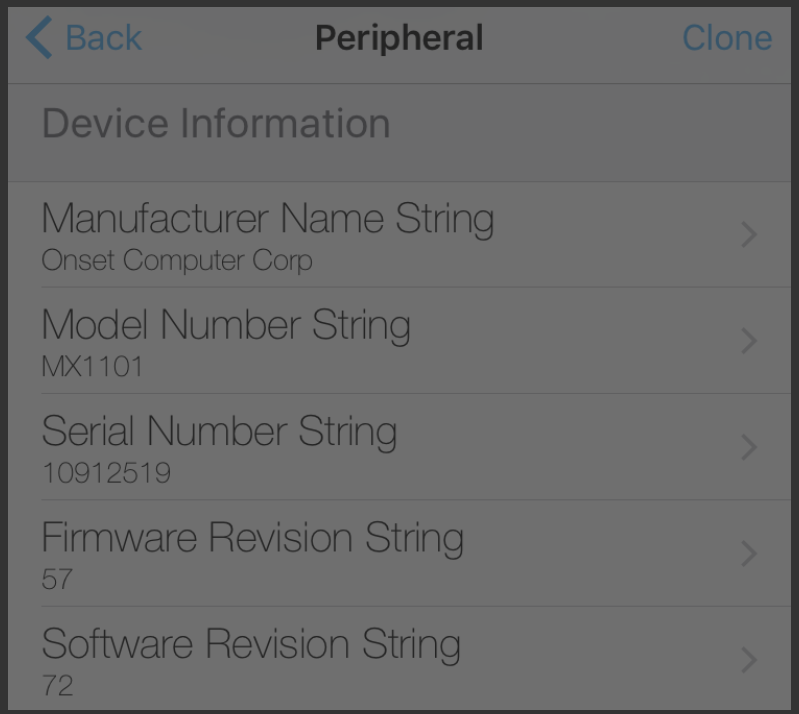
>>> Findings on the MX1101

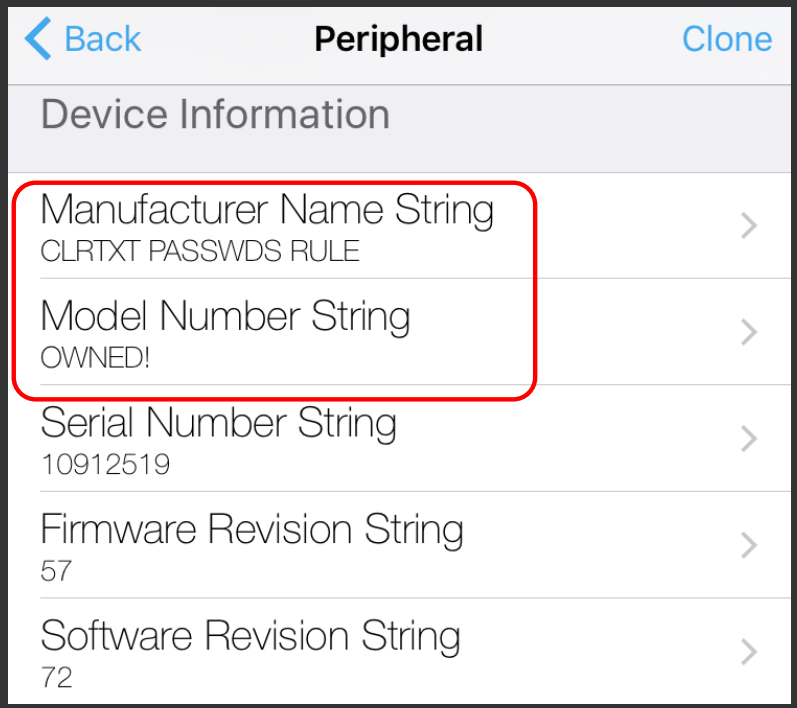
* Password is sent in cleartext
* Custom C2 and data transfer protocols
  - Write commands to handle 0x0019
  - Receive notifications from the same
* Some unhandled commands can force logging to stop
* Can perform over-the-air firmware update



>>> Findings on the MX1101

* Password is sent in cleartext
* Custom C2 and data transfer protocols
  - Write commands to handle 0x0019
  - Receive notifications from the same
* Some unhandled commands can force logging to stop
* Can perform over-the-air firmware update





>>> Now, do something with it!

Our goal: pillage, plunder, pwn

* On Kali: write *scapy* scripts
  - Write your own custom functions using L2CAP and ATT layers
  - Use *pyBT* python egg, which does it for you
* On tablet: use *nRFConnect*
  - Read from and write to handles
  - Subscribe to characteristic notifications
  - Useful if commands aren't very long

>>> Now, do something with it!

Our goal: pillage, plunder, pwn

* On Kali: write *scapy* scripts
  - Write your own custom functions using L2CAP and ATT layers
  - Use *pyBT* python egg, which does it for you
* On tablet: use *nRFConnect*
  - Read from and write to handles
  - Subscribe to characteristic notifications
  - Useful if commands aren't very long

Proof-of-concept tools for the MX1101, available on repo:

* Password sniffer & brute forcer (*scapy*)
* Data dumper (*scapy*)
* Device impersonator (*bleno*)
* Firmware uploader (*scapy*)

>>> More resources

Some of the resources that we found valuable:

* Bluetooth Low Energy: The Developer's Handbook, by Robin Heydon
* Mike Ryan's research, lacklustre.net/bluetooth/ & github.com/mikeryan
* Bleno documentation, github.com/sandeepmistry/bleno
* Bluetooth specs, www.bluetooth.com

>>> Special thanks

These folks helped a lot:

* Ben Ramsey,
* Anthony Rose,
* Caleb Mays,
* Mike Ryan (indirectly),
* The lady at home.

>>> Questions?

Code: github.com/merculite/BLE-Security

Have comments, compliments, or cash?
Contact us: team @ merculite.net

MERCULITE
SECURITY