Battery Depletion Attack through Packet Injection on IoT Thread Mesh Network

Poonam Yadav, Nirdesh Sagathia, Dan Wade
University of York, UK
poonam.yadav@york.ac.uk

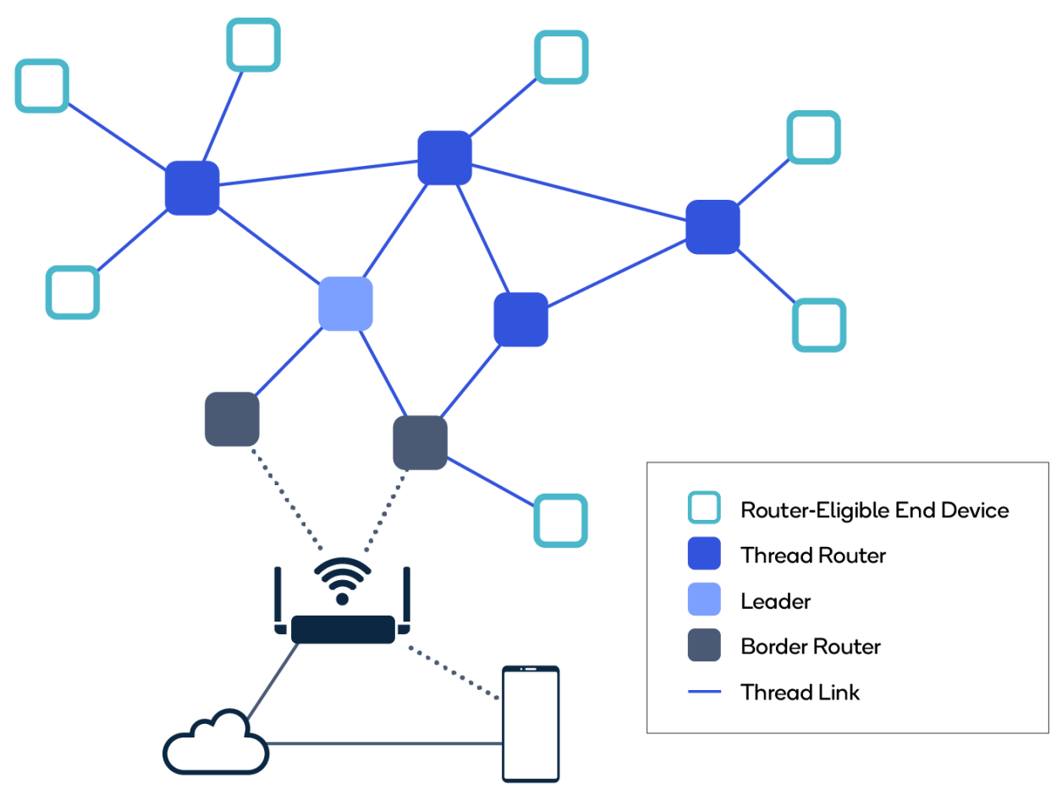
https://github.com/SystronLab/ThreadBatteryAttack/

Abstract

In the rapidly expanding landscape of Internet of Things (IoT) device manufacturing and deployment, concerns about security have become prominent. This demonstration involves practical attacks on a thread-mesh network within a controlled environment, exploiting vulnerabilities in various components of the Thread network stack. Our attack vectors successfully identified nearby Thread networks and devices by gathering 2-byte Personal Area Network ID (PAN ID) and device frequency information, serving as reconnaissance for potential additional attacks. The focus was on investigating susceptibility to replay attacks and packet injection into thread-mesh networks. Although the experiment attempted to capture thread packets to emulate an authorised sender, the cryptographic encryption and sequence numbers employed for integrity checks resulted in packet rejection by the network. Despite this, our successful injection of packets highlights the potential for battery depletion attacks.

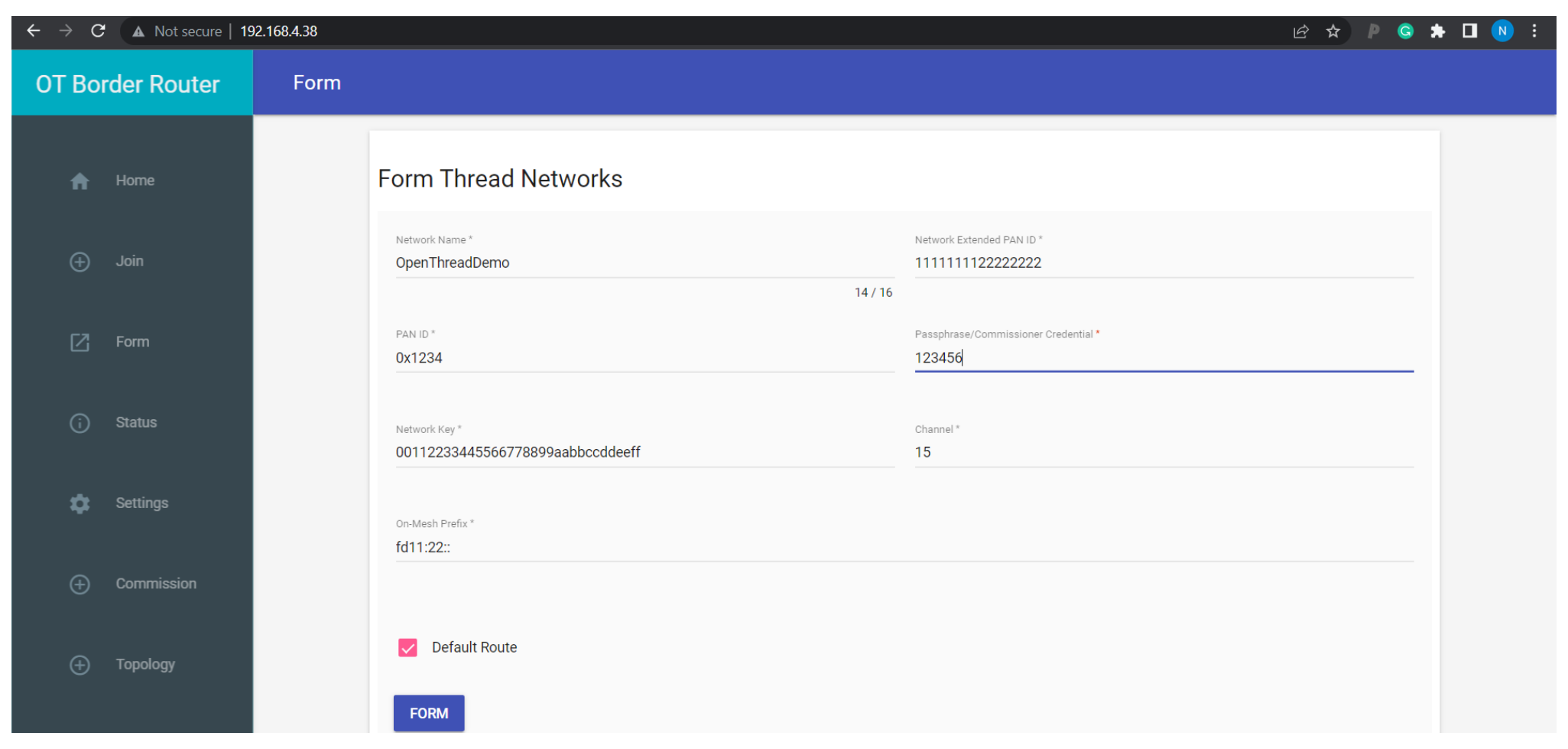
Thread Network Setup

- **Setting Up a Border Router:** A Raspberry Pi 3B is connected to an nRF52840 USB dongle to operate the OpenThread border router software, which is freely accessible from OpenThread. Google introduced OpenThread (openthread.io) as an open-source implementation of Thread, aiming to enhance the accessibility of networking technology used in Google Nest products for a wider developer audience. This move seeks to expedite the development of products for connected homes and commercial buildings. With a concise platform abstraction layer and a small memory footprint, OpenThread (OT) is highly portable, supporting both System-on-Chip (SoC) and Co-Processor (RCP, NCP) designs. The Border Router functions as the external gateway for the Thread network, establishing a connection between the Thread network and other IP-based networks like Wi-Fi or Ethernet. Additionally, it facilitates the formation of a Thread network and supports external commissioning, enabling the seamless addition of new devices to the Thread network using a phone app. Moreover, it offers features to display a network diagram and provide valuable information about the network.



The diagram depicts simplified blocks of a Thread network.

- **Forming a Thread Network:** A network can be created through a web interface on the Border Router. The procedure involves specifying a network name and a passphrase. The passphrase is used to generate the Pre-Shared Key for the Commissioner (PSKc), enabling an external commissioning device to authenticate and commission new devices onto the network.



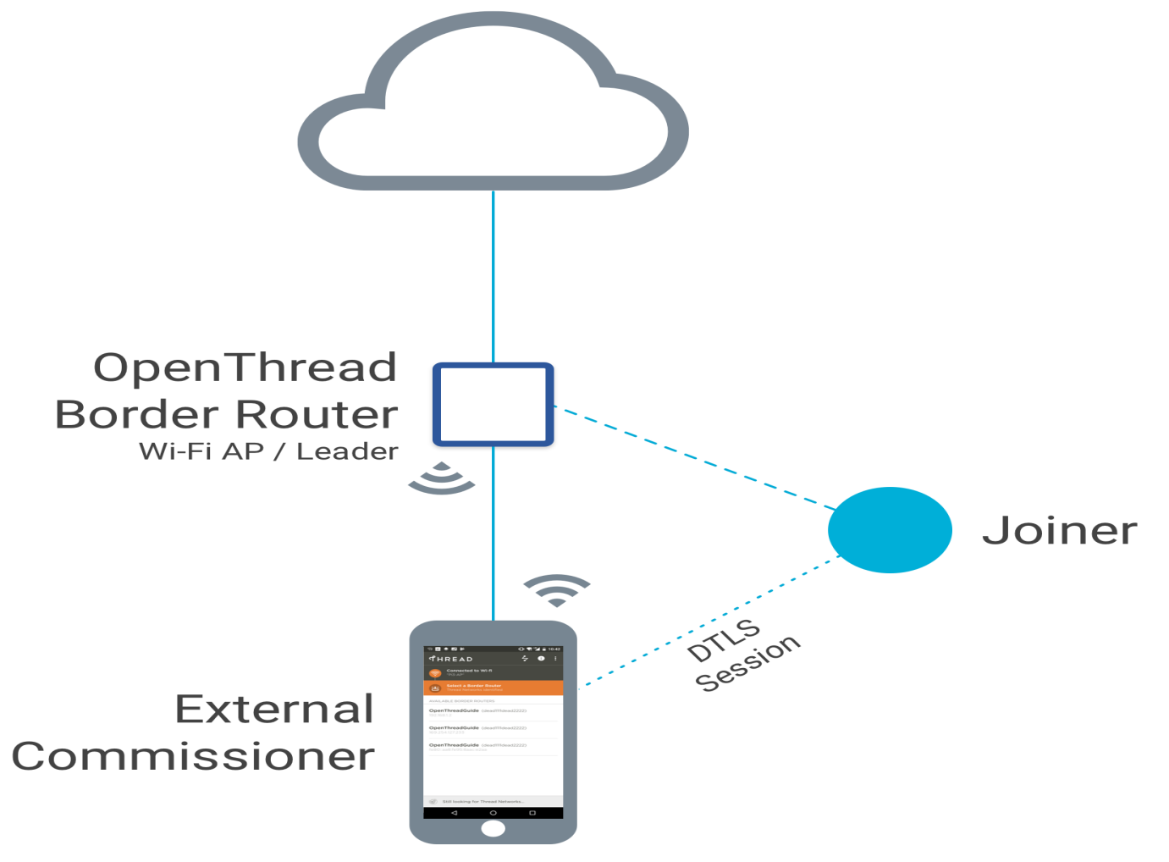
Form Thread Networks

Network Name * OpenThreadDemo (14 / 16)
Network Extended PAN ID * 1111111122222222
PAN ID * 0x1234
Passphrase/Commissioner Credential * 123456
Network Key * 00112233445566778899aabbccddeeff
Channel * 15
On-Mesh Prefix * fd11:22::
☑ Default Route
FORM

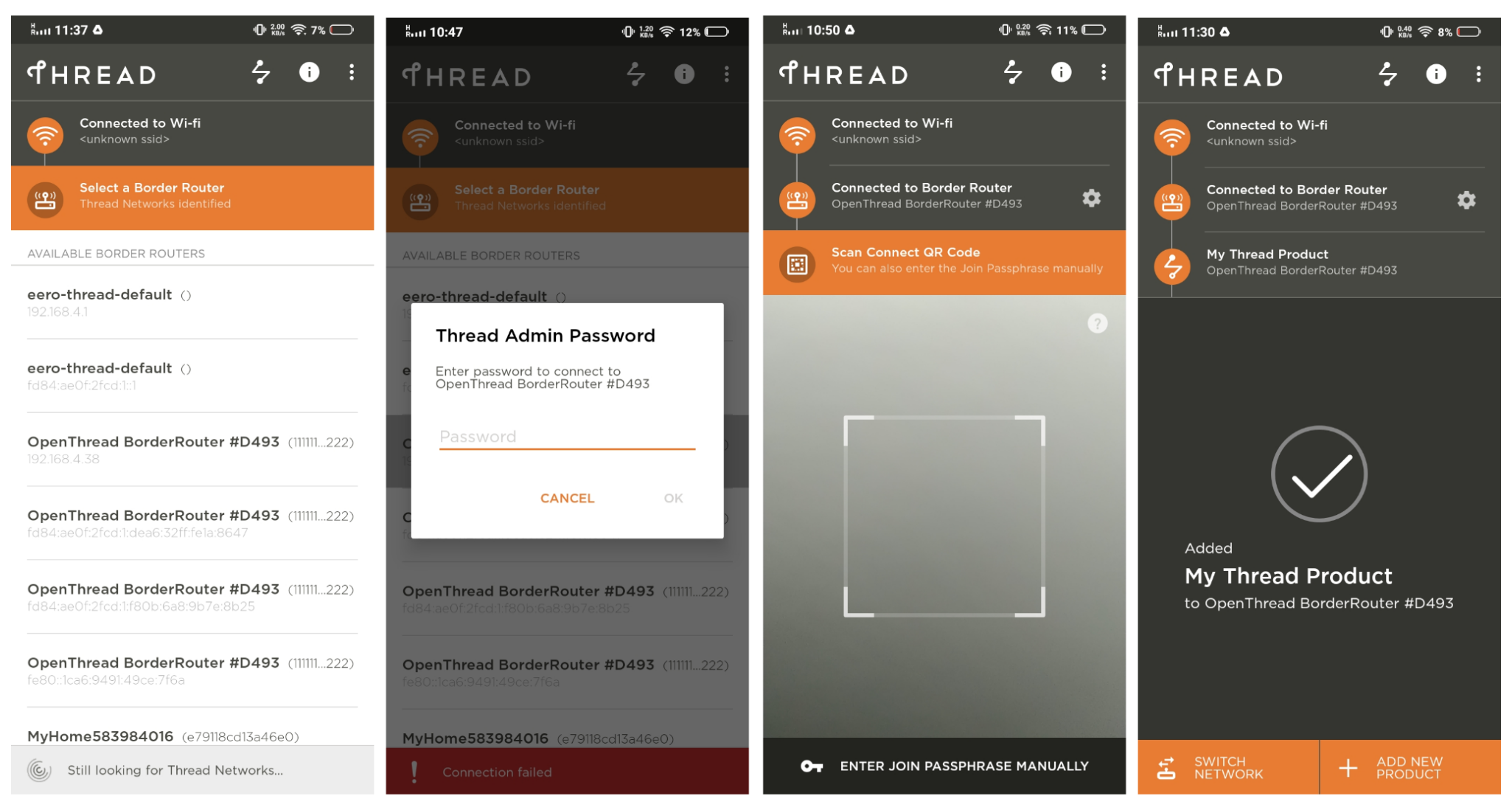
The border router web interface.

- **Commissing a New Device:** The method employed for commissioning new devices onto the Thread Network is known as external commissioning. In this process, a device that is not part of the Thread network commissions new devices onto the network using various methods, such as the command line or a phone app. Additionally, it is possible to commission a new device without an external commissioner, a method known as On-Mesh Commissioning. However, using the app proved to be more straightforward and convenient, eliminating the need for manually running a set of commands.

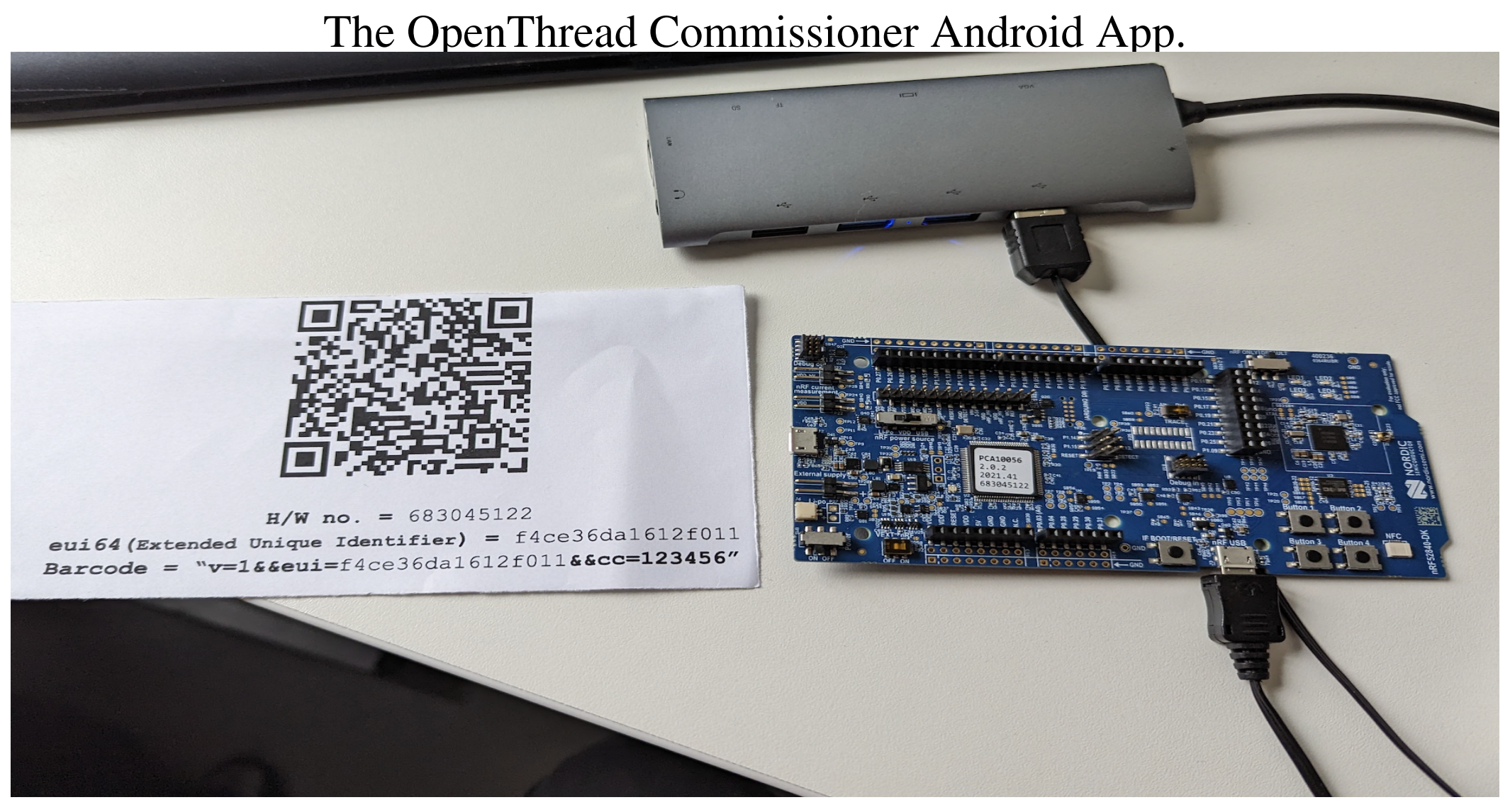
An external device, not part of the Thread network, can add a device to the Thread network by authenticating with the Border Router using the PSKc key generated during the network formation. The commissioner then conveys the details of the new device to the border router, which initiates a connection to the new device using Datagram Transport Layer Security (DTLS) and adds it to the network by sharing the network key.



The figure shows how an external commissioning device interacts with the Thread network.


THREAD
Connected to Wi-fi
<unknown ssid>
Select a Border Router
Thread Networks identified
AVAILABLE BORDER ROUTERS
eero-thread-default
192.168.4.1
eero-thread-default
fd84:ae0f:2fcd:1::1
OpenThread BorderRouter #D493 (111111...222)
192.168.4.38
OpenThread BorderRouter #D493 (111111...222)
fd84:ae0f:2fcd:1:dea6:32ff:fe1a:8647
OpenThread BorderRouter #D493 (111111...222)
fd84:ae0f:2fcd:1:f80b:6a8:9b7e:8b25
OpenThread BorderRouter #D493 (111111...222)
fe80::1ca6:9491:49ce:7f6a
MyHome583984016 (e79118cd13a46e0)
Still looking for Thread Networks...
Thread Admin Password
Enter password to connect to OpenThread BorderRouter #D493
Password
CANCEL
OK
Connection failed
Connected to Border Router
OpenThread BorderRouter #D493
Scan Connect QR Code
You can also enter the Join Passphrase manually
ENTER JOIN PASSPHRASE MANUALLY
My Thread Product
Added
My Thread Product
to OpenThread BorderRouter #D493
SWITCH NETWORK
ADD NEW PRODUCT

The OpenThread Commissioner Android App.

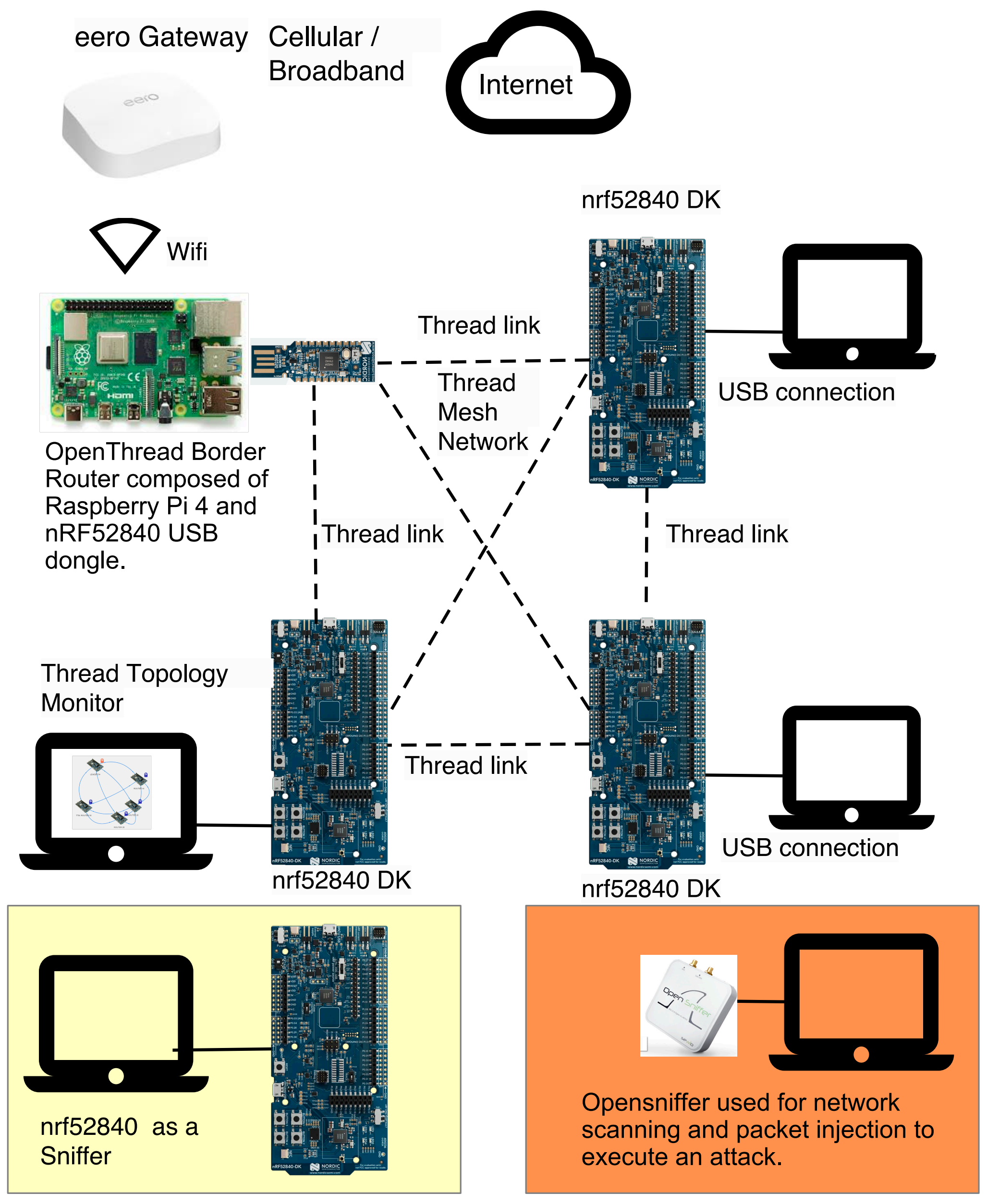

H/W no. = 683045122
eui64 (Extended Unique Identifier) = f4ce36da1612f011
Barcode = "v=1&&eui=f4ce36da1612f011&&cc=123456"

nRF52840 DK used as a Thread node, with its QR code to allow it to be commissioned onto the network.

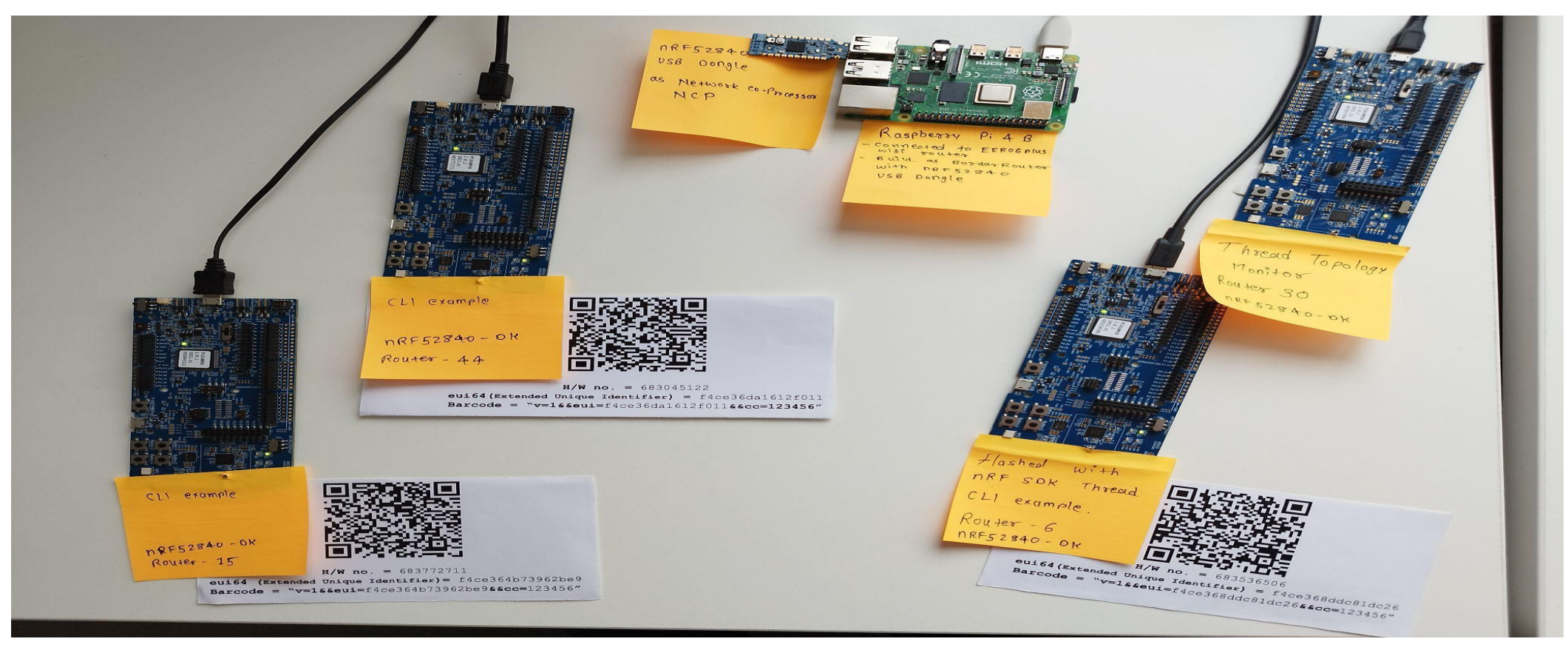
```
v=1&&eui=<new device Extended Unique
Identifer>&&cc=<passphrase>
```

The extended unique identifier (EUI) of the device can be obtained by running the command "eui64" on the new device when connected over serial. Once the QR code is scanned, the app will wait for the device to complete the joining process. This step must be manually completed through the command line on the new device by enabling the network interface (ifconfig up') and initiating the join process (joiner start $< passphrase >$'). Afterward, wait a few minutes to receive a success message both on the command line and in the app. To connect the device to the Thread network, simply execute the command 'thread start'.

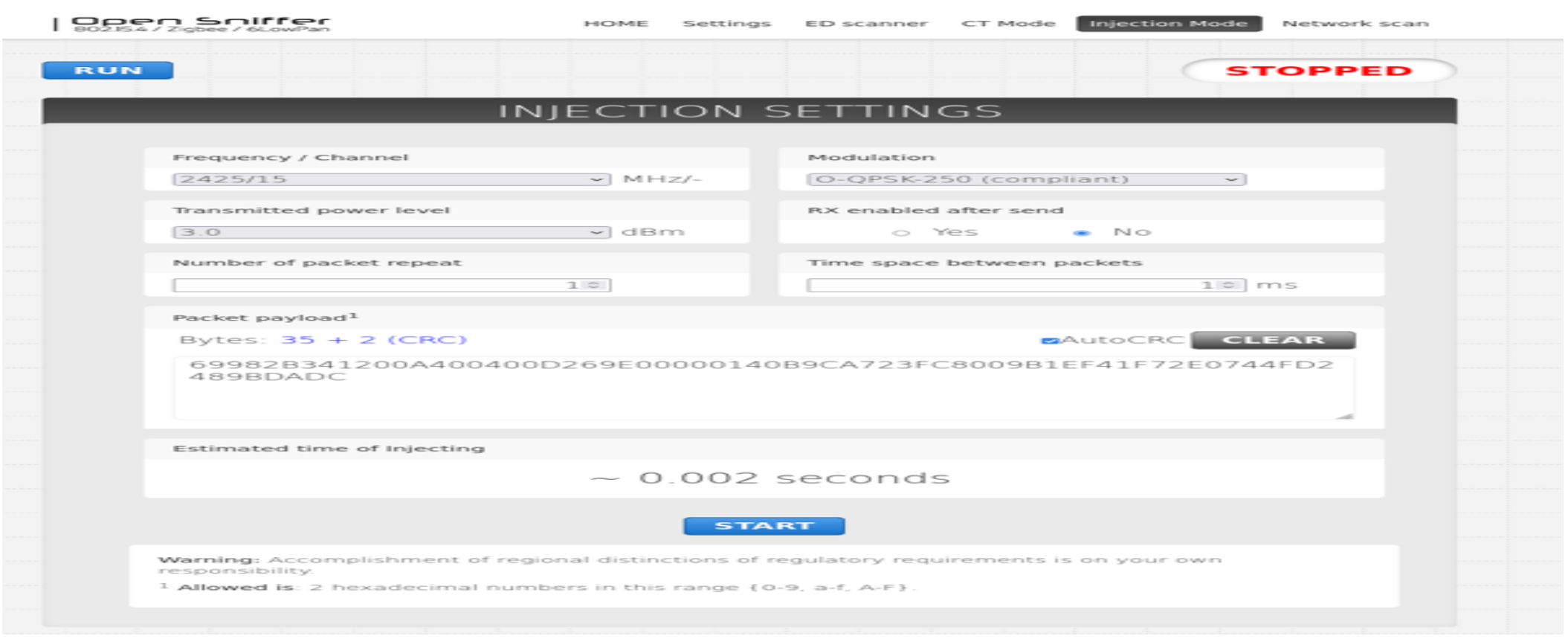
Replay and Battery Depletion Attack Setup



The diagram illustrates the experimental network configuration at Systron Lab. It features nRF52840 Development Kits (https://www.nordicsemi.com/Products/nRF52840) as Thread nodes and a Border Router, consisting of a Raspberry Pi connected to an nRF52840 USB dongle, for establishing a Thread mesh network. The Border Router interfaces with an eero gateway. All Nordic Development kits communicate wirelessly via IEEE 802.15.4, with USB connections to laptops/computers exclusively used for serial connections. An nRF52840 connected to a PC acts as a sniffer. Network scanning and packet injection for conducting attacks are carried out using the Sewio OpenSniffer. Please note that both the packet sniffer and packet injector exist externally to the network.



Results

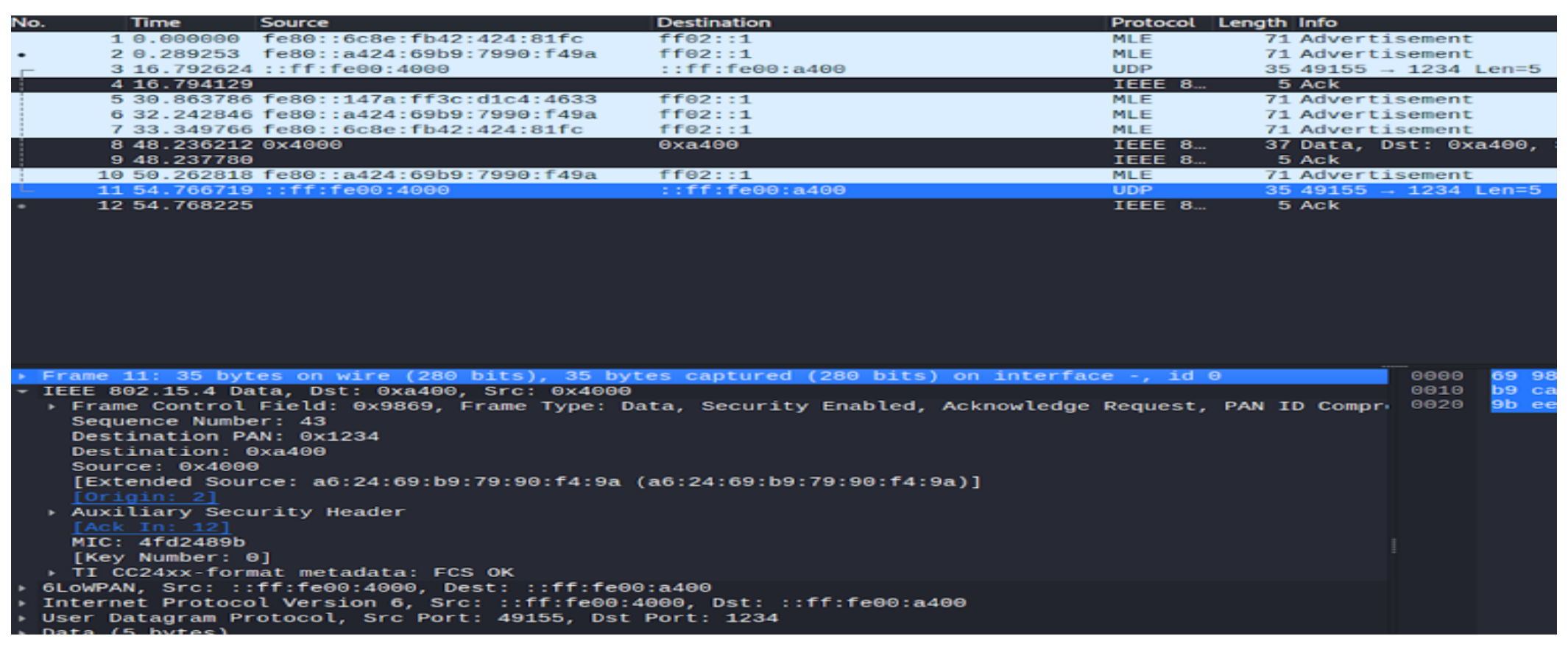


INJECTION SETTINGS

Frequency / Channel: 2425/15 MHz/-
Modulation: O-QPSK-250 (compliant)
Transmitted power level: 3.0 dBm
RX enabled after send: ○ Yes ● No
Number of packet repeat: 1
Time space between packets: 1 ms
Packet payload¹
Bytes: 35 + 2 (CRC) ☑ AutoCRC
69982B341200A400400D269E00000140B9CA723FC8009B1EF41F72E0744FD2489BDADC
Estimated time of injecting: ~ 0.002 seconds

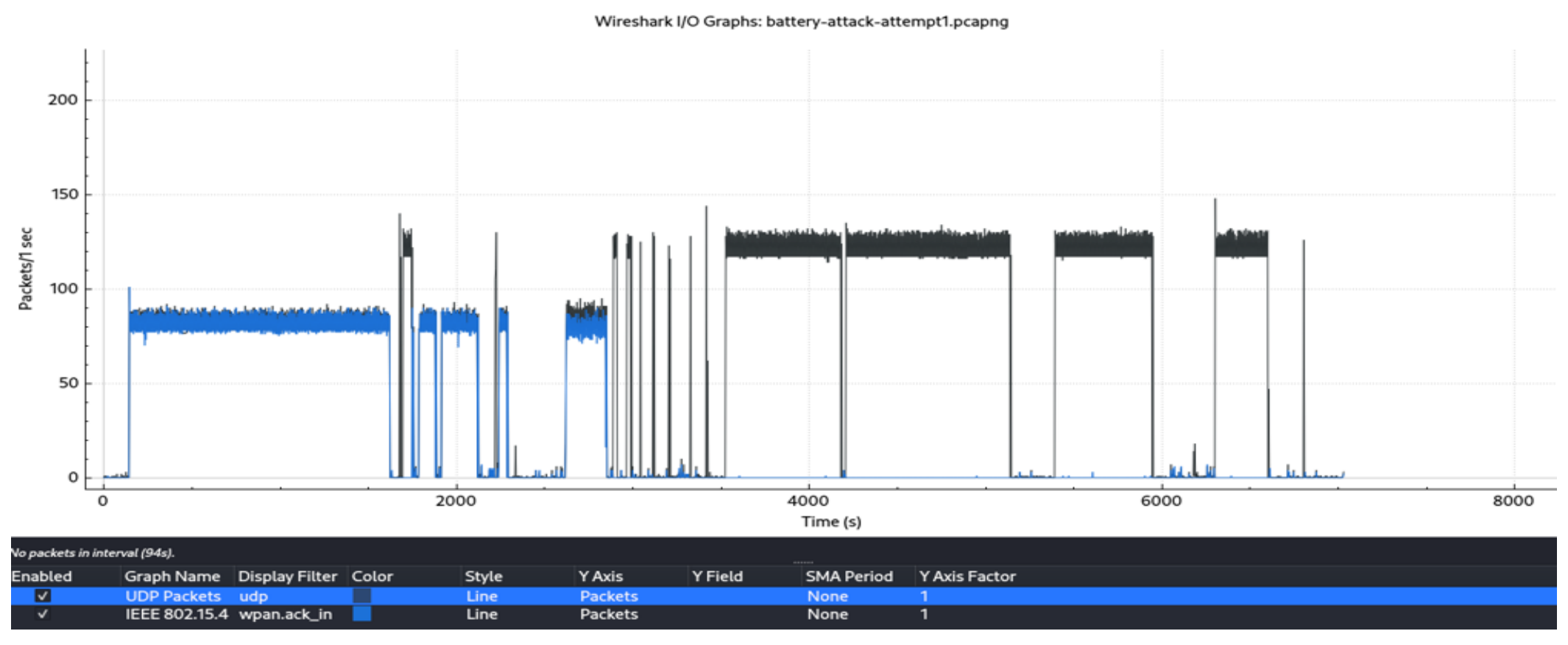
Warning: Accomplishment of regional distinctions of regulatory requirements is on your own responsibility.
¹ Allowed is: 2 hexadecimal numbers in this range {0-9, a-f, A-F}.

Open Sniffer Packet Injection Setting.



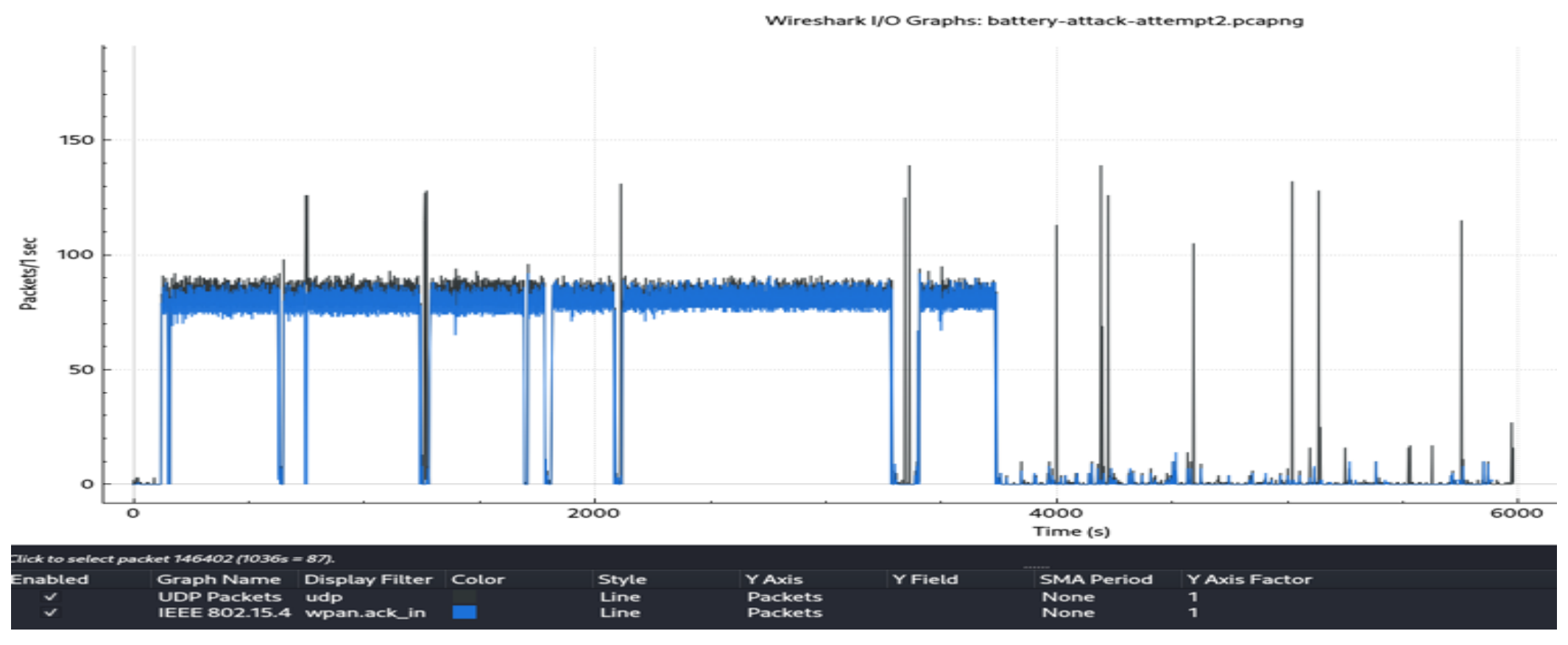
```
No.  Time       Source                            Destination          Protocol  Length Info
  1  0.000000   fe80::6c8e:fb42:424:81fc          ff02::1              MLE           71 Advertisement
  2  0.289253   fe80::a424:69b9:7990:f49a         ff02::1              MLE           71 Advertisement
  3 16.792624   ::ff:fe00:4000                    ::ff:fe00:a400       UDP           35 49155 → 1234 Len=5
  4 16.794129                                                          IEEE 8...      5 Ack
  5 30.863786   fe80::147a:ff3c:d1c4:4633         ff02::1              MLE           71 Advertisement
  6 32.242846   fe80::a424:69b9:7990:f49a         ff02::1              MLE           71 Advertisement
  7 33.349766   fe80::6c8e:fb42:424:81fc          ff02::1              MLE           71 Advertisement
  8 48.236212   0x4000                            0xa400               IEEE 8...     37 Data, Dst: 0xa400,
  9 48.237780                                                          IEEE 8...      5 Ack
 10 50.262818   fe80::a424:69b9:7990:f49a         ff02::1              MLE           71 Advertisement
 11 54.766719   ::ff:fe00:4000                    ::ff:fe00:a400       UDP           35 49155 → 1234 Len=5
 12 54.768225                                                          IEEE 8...      5 Ack

▸ Frame 11: 35 bytes on wire (280 bits), 35 bytes captured (280 bits) on interface -, id 0
▾ IEEE 802.15.4 Data, Dst: 0xa400, Src: 0x4000
   ▸ Frame Control Field: 0x9869, Frame Type: Data, Security Enabled, Acknowledge Request, PAN ID Compr
     Sequence Number: 43
     Destination PAN: 0x1234
     Destination: 0xa400
     Source: 0x4000
     [Extended Source: a6:24:69:b9:79:90:f4:9a (a6:24:69:b9:79:90:f4:9a)]
     [Origin: 2]
   ▸ Auxiliary Security Header
     [Ack In: 12]
     MIC: 4fd2489b
     [Key Number: 0]
   ▸ TI CC24xx-format metadata: FCS OK
▸ 6LoWPAN, Src: ::ff:fe00:4000, Dest: ::ff:fe00:a400
▸ Internet Protocol Version 6, Src: ::ff:fe00:4000, Dst: ::ff:fe00:a400
▸ User Datagram Protocol, Src Port: 49155, Dst Port: 1234
▸ Data (5 bytes)
```

We observe the UDP packet 3, which we captured earlier, being successfully replayed into the network as packet number 11.



No packets in interval (94s).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
✓	UDP Packets	udp	■	Line	Packets		None	1
✓	IEEE 802.15.4	wpan.ack_in	■	Line	Packets		None	1



Click to select packet 146402 (1036s = 87).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
✓	UDP Packets	udp	■	Line	Packets		None	1
✓	IEEE 802.15.4	wpan.ack_in	■	Line	Packets		None	1

Acknowledgment

This work is supported in part by EPSRC grants EP/X525856/1, EP/X040518/1 and EP/Y019229/1.