Demo: Battery Depletion Attack through Packet Injection on IoT Thread Mesh Network

Poonam Yadav
Department of Computer Science
University of York
United Kingdom
poonam.yadav@york.ac.uk

Nirdesh Sagathia
Department of Computer Science
University of York
United Kingdom
ns1584@york.ac.uk

Dan Wade
Department of Computer Science
University of York
United Kingdom
djw585@york.ac.uk

***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.**

***Index Terms*—Internet of Things (IoT), Thread, IEEE 802.15.4, Thread Protocol, Zigbee, Matter Protocol, Interoperability**

I. Introduction

The growing prevalence of Internet of Things (IoT) devices from various emerging companies has sparked considerable apprehension regarding the security of these interconnected systems [1]. In response to this concern, we direct our focus towards the Thread protocol, a successor to the Zigbee protocol that operates on the same radio frequency, namely 802.15.4. Initially, we propose replicating attacks feasible on Zigbee to gain familiarity with the network's behaviors. Our success in executing replay attacks on Zigbee serves as motivation to explore similar exploits within the Thread network [2]. In this demonstration, our goal is to showcase the following objectives:

- Setting up a Thread mesh network [3] where devices establish connections using IPv6.
- Develop a sniffer for IEEE 802.15.4 network [4] with the capability to capture and later replay as an injector on the Thread mesh network. This involves capturing packets from Development kits and Commercial Home automation devices, providing insights from an attacker's perspective. In essence, the objective is to discern what an attacker can deduce from the Thread network [5].
- Explore the potential for replay attacks on Thread networks.
- Examine the viability of battery depletion attacks that could lead to Denial of Service incidents in Thread networks [6]–[8].

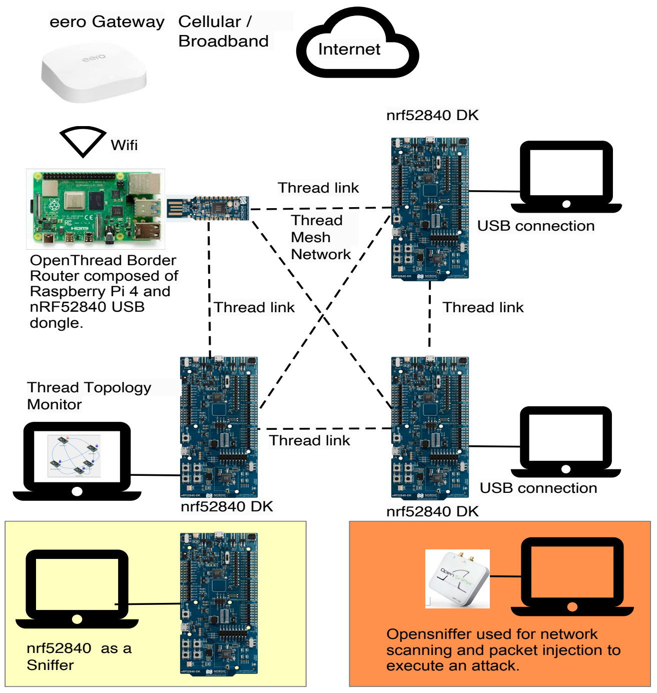
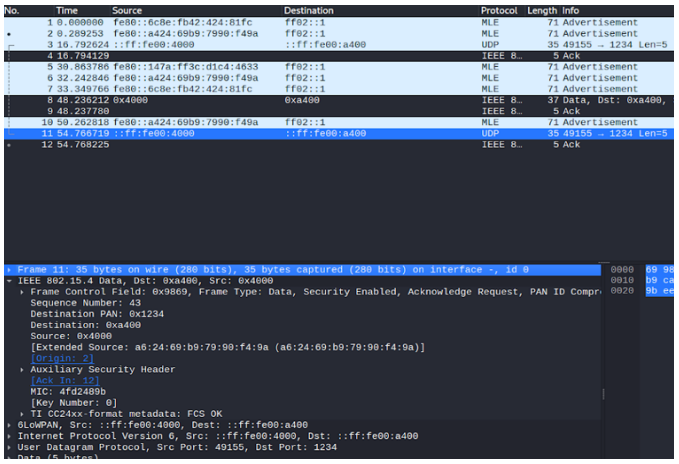


Fig. 1: The diagram illustrates the experimental network configuration at Systron Lab. It features nRF52840 Development Kits [9] as Thread nodes and a Border Router, consisting of a Raspberry Pi connected to an nRF52840 USB dongle, for establishing a Thread mesh network. The Border Router interfaces 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.



```
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     0000  69 98
⌄ IEEE 802.15.4 Data, Dst: 0xa400, Src: 0x4000                                                0010  b9 ca
  › Frame Control Field: 0x9869, Frame Type: Data, Security Enabled, Acknowledge Request, PAN ID Compr  0020  9b ee
    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)
```

Fig. 2: In this instance, we witness the successful replay of the previously captured UDP packet into the network. The original packet is designated as packet number 3, the first replayed packet is 8, and the subsequent replayed packet, after the removal of the last two bytes, is identified as packet number 11 in the figure. Both of these packets are acknowledged, although, at the upper layer, they were not received.

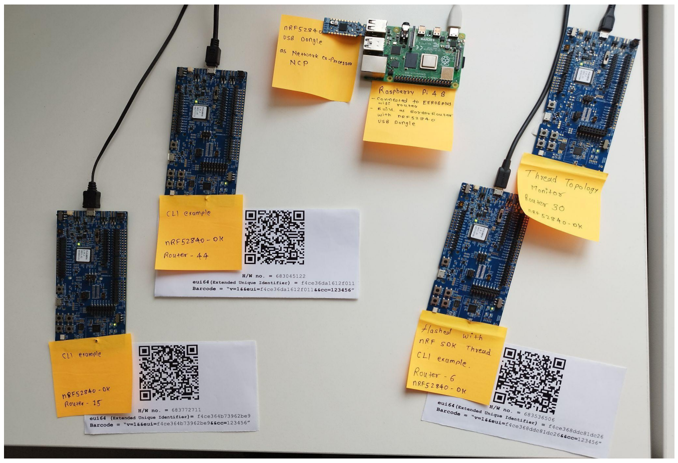

nRF52840 USB Dongle as Network co-Processor NCP
Raspberry Pi 4 B - Connected to EEROGplus wifi router - Build as Border Router with nRF52840 USB Dongle
CLI example nRF52840 - DK Router - 44
H/W no. = 683045122
eui64 (Extended Unique Identifier) = f4ce36da1612f011
Barcode = "v=1&&eui=f4ce36da1612f011&&cc=123456"
Thread Topology Monitor Router 30 nRF52840-DK
CLI example nRF52840 - DK Router - 15
H/W no. = 683772711
eui64 (Extended Unique Identifier)= f4ce364b73962be9
Barcode = "v=1&&eui=f4ce364b73962be9&&cc=123456"
Flashed with nRF SDK Thread CLI example. Router - 6 nRF52840 - DK
H/W no. = 683536506
eui64 (Extended Unique Identifier) = f4ce368ddc81dc26
Barcode = "v=1&&eui=f4ce368ddc81dc26&&cc=123456"

Fig. 3: The network setup includes all nRF52840 Development Kits [9] and a Border Router, consisting of a Raspberry Pi connected to an nRF52840 USB dongle, for establishing a Thread mesh network. All Nordic Development kits are wirelessly linked via IEEE 802.15.4 (Thread protocol), and USB connections are exclusively used for accessing serial connections.

II. Failed endeavor to execute a Replay attack

To validate our assumption regarding the replay attack, we executed an experiment involving the extraction of raw hex data and UDP message packets from the PCAP files containing the captured packets. Subsequently, we injected these packets into the Thread network using an OpenSniffer device. Initially, the first attempt was unsuccessful due to a packet length mismatch between the sent packet and the sniffed packet by the OpenSniffer. For example, we observed that the received packet had an additional 2 bytes in Wireshark when replaying packet number 8, as shown in Fig. 2. This packet was not recognised as UDP and displayed a length discrepancy of 37 bytes, deviating from the original 35 bytes.

To address this issue, we removed the final four hex characters during packet injection. Following this adjustment, packet number 11, as shown in Fig. 2, was modified to precisely match the original UDP format and length of 35 bytes. Despite achieving a structurally accurate packet, the receiving device acknowledged its reception at the Link layer, and the packet acknowledgment occurred in packet 12, indicating reception but rejection at the MAC (Medium Access Control) layer. This requires further detailed investigation.

The research recognises the encryption challenges posed by the captured packets and highlights potential time-consuming trial-and-error methods for identifying valuable packets for replay attacks. However, this experiment motivated us to investigate Denial of Service (DoS) attacks in this instance by understanding energy utilisation in processing the replayed packets and their impact on overall node/device availability and functionality.

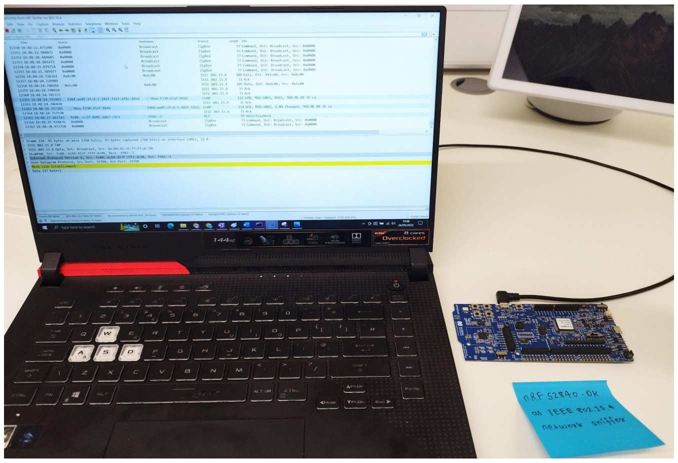


Fig. 4: An nRF52840 Development Kit, flashed with a sniffer binary hex file, is connected to a Windows PC via USB to capture packets on Wireshark.

III. Battery Depletion Attack

We conducted a battery depletion attack using a Raspberry Pi as a monitoring tool and Sewio OpenSniffer for packet capture and replay. Subsequently, we attempted to ping the OpenThread device's IPv6 address during the battery depletion attack. However, once the attack commenced, the target nRF52840 Development Kit with a battery (CC1352P1) couldn't respond to ICMPv6 Echo Request messages due to the overwhelming flood of UDP packets, typically occurring at around 85–100 packets per second [10].

Pausing traffic flooding to track the outage duration proved ineffective due to the high volume of packets. The Sewio OpenSniffer's proprietary web interface, which allows in-

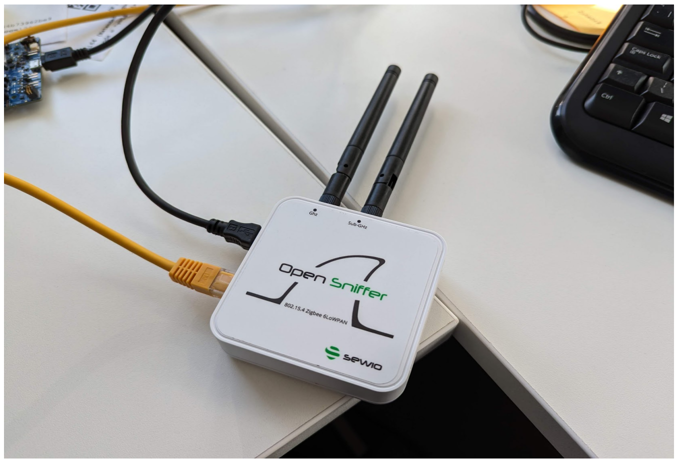


Fig. 5: A Sewio OpenSniffer is utilised for packet capture and injection.

jecting only a specific number of packets, faced issues like overflow and negative numbers with large values. In contrast, Wireshark packet captures were analysed for reverse engineering the underlying frame structure. We developed a Python script with a requests module that repeatedly sent GET requests to inject the endpoint into a loop. Additionally, we incorporated 5-second delays after any failed or timeout injection requests to enforce stability within the script.

Experimental trials indicated that the CC1352P1 battery discharges within 1-2 hours. This means the battery voltage drops from 3.3V to 2.7V, resulting in total energy depletion and the inability to power on again even with a new battery. Remarkably, another trial lasted for over nine hours at a higher rate of 100 packets per second without disabling, although the voltage declined to 2.8V, indicating erratic behavior in battery chemistry. For example, demoting an OpenThread router and promoting a battery-powered device to the leader are unintended side effects that illustrate the self-healing of the mesh network but require rebooting to be corrected. This work is in progress, and you can find the latest updates on Systron Lab's GitHub account [11].

IV. Demo Requirements

To deliver a high-quality exhibition, the following conditions and requirements must be met for this demonstration:

1) A test area measuring 2.5 meters in length and featuring a 1-meter-wide table to accommodate the two laptops.
2) To minimise potential interference with other demonstrations, it is recommended to consider an alternative arrangement that avoids conflicts, especially considering that the thread network will broadcast on Channel 11 of radio 802.15.4.

V. Demo Design

For creating a Thread Mesh Network, the following hardware setup will be created by the demo presenters.

1) Two laptops or PCs for sniffing and injecting network traffic packets.
2) The Thread Network comprises a Raspberry Pi 4 functioning as an OpenThread Border Router, connected to a Nordic nrf52840 dongle. Additionally, there are four Nordic nrf52840 development kits, each flashed with full-thread devices, a Sleepy End device, and a Thread Topology monitor, as illustrated in Fig. 3.
3) An IEEE 802.15.4 sniffer is set up using a Nordic nrf52840 Development kit flashed with a sniffer binary hex file, as illustrated in Fig. 4.
4) The Sewio OpenSniffer [12], which features a Web-GUI as depicted in Fig. 5, is employed to inject packets into the Thread mesh network. Additionally, it possesses the capability to sniff packets and scan the Thread channel operating on the IEEE 802.15.4 radio.

Acknowledgment

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

References

[1] D. Dinu and I. Kizhvatov, "Em analysis in the iot context: Lessons learned from an attack on thread," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, p. 73–97, Feb. 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/833

[2] F. Farha, H. Ning, S. Yang, J. Xu, W. Zhang, and K.-K. R. Choo, "Timestamp scheme to mitigate replay attacks in secure zigbee networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 342–351, 2022.

[3] Threadgroup, "Overview of Thread," https://www.threadgroup.org/What-is-Thread/Overview, accessed on 25th Oct 2022.

[4] N. Semiconductor, "nRF Sniffer for 802.15.4," https://github.com/NordicSemiconductor/nRF-Sniffer-for-802.15.4, accessed on 23rd September 2023.

[5] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on iot security: Application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82 721–82 743, 2019.

[6] D.-G. Akestoridis, V. Sekar, and P. Tague, "On the security of thread networks: Experimentation with openthread-enabled devices," in *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 233–244. [Online]. Available: https://doi.org/10.1145/3507657.3528544

[7] A. Feraudo, D. A. Popescu, P. Yadav, R. Mortier, and P. Bellavista, "Mitigating iot botnet ddos attacks through mud and ebpf based traffic filtering," in *25th International Conference on Distributed Computing and Networking (ICDCN)*, 2024, pp. 1–12.

[8] D. Wade and P. Yadav, "Performing a Replay and Battery Depletion Attack against Thread networks," BSc Disseration, https://github.com/SystronLab/ThreadBatteryAttack/, accessed on 1 Oct 2023.

[9] N. semiconductor, "nRF52840 SoC," https://www.nordicsemi.com/products/nrf52840, accessed on 25th Oct 2022.

[10] V.-L. Nguyen, P.-C. Lin, and R.-H. Hwang, "Energy depletion attacks in low power wireless networks," *IEEE Access*, vol. 7, pp. 51 915–51 932, 2019.

[11] SystronLab, "Thread Battery Attack," https://github.com/SystronLab/ThreadBatteryAttack/, accessed on 15th December 2023.

[12] Sewio, "Open Sniffer," https://www.sewio.net/open-sniffer/, accessed on 23rd September 2023.