Poster: Evaluation of Radio Jamming Countermeasures in IoT Thread Networks

Poonam Yadav
Computer Science Department
University of York, UK
poonam.yadav@york.ac.uk

Anthony Moulds
Computer Science Department
University of York, UK
anthony.moulds@york.ac.uk

Peter Gillingham
Computer Science Department
University of York, UK
peter.gillingham@york.ac.uk

***Abstract*—The Thread mesh networking protocol, designed for IoT communication, ensures network layer interoperability but remains vulnerable to security threats. A major risk is denial-of-service (DoS) attacks, which can overwhelm networks, causing congestion and device failures. While Thread's self-healing mesh mitigates some risks, devices must be robust against resource exhaustion attacks. This paper examines DoS attacks via radio jamming, implementing a jamming system on a bespoke Thread Testbed. A countermeasure using standard channel hopping is proposed and tested, demonstrating its effectiveness.**

I. Introduction

IoT devices use various communication protocols like Wi-Fi [1], Bluetooth [2], Zigbee [3], [4], and Thread [5], [6], each with unique advantages and limitations. However, these networks are susceptible to radio jamming attacks [7], [8], [6], where adversaries interfere with legitimate communication by emitting disruptive radio signal or by replying the packets. In this work we investigate Thread network susceptibility to Denial-of-Service attacks via radio/packet jamming, setting up a testbed, constructing a jamming system, and developing countermeasures.

II. Channel Jamming and Countermeasure Experiment and Results

A. *Thread Testbed Setup*

A bespoke Testbed was developed for Thread networking experiments, featuring a 20-node setup with a Border Router (BR), four Full Thread Devices (FTDs), and 15 Minimal Thread Devices (MTDs). Nordic Semiconductor nRF5340DK boards were used for FTDs, while nRF52840 USB Dongles were chosen for MTDs. Each Thread device was connected to a Raspberry Pi via USB for power and debugging. To emulate real-world conditions, transmit power levels were reduced (-20 dBm for FTDs, -40 dBm for MTDs) to encourage local grouping of End Devices with their nearest Router, maintaining a single unpartitioned network. The Nordic Semiconductor nRF52840-DK is used for network scanning and the Sewio OpenSniffer is used for packet injection.

1) Channel Sniffing and Jamming: Since the jamming attack relies on saturating the channel rather than establishing a connection, only the active channel needs to be identified. The OpenThread CLI's scan command, utilizing the Mesh Link Establishment (MLE) protocol, retrieves the channel number, PAN ID, Extended PAN ID, and network name, helping confirm a valid Thread network. The identified channel is then used for packet injection via the Sewio OpenSniffer. The OpenSniffer injects IEEE 802.15.4 packets with a 1 ms inter-frame spacing (IFS) to disrupt communication. Packet transmission is preferred over a modulated carrier wave as it enables jamming event marking in Wireshark and interferes with the MAC layer, increasing the likelihood of a successful attack. Controlled via HTTP commands over an RJ45 connection, the OpenSniffer transmits payload-free packets through a 12 dBi directional antenna with a 20 dB LNA, ensuring sufficient coverage of the entire network. The transmitted signal power is approximately 15 dBm.

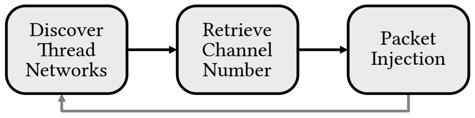


Fig. 1. Flow chart describing Channel Sniffing and Jamming Process.

B. *Jamming Countermeasure*

When active, the single channel radio jamming apparatus, described in this paper, attacks the IEEE 802.15.4 PHY in the Thread stack layer of each node on the Thread network. If sufficient RF energy is received from the transmitted jamming source on the correct channel by the PHY's receiver, the Thread device's MAC sub-layer inhibits transmitting packets due to a poor clear quality assessment (CCA) result in the CSMA/CA mechanism. In addition, reception of any valid IEEE 802.15.4 packets will be rendered impossible due to useless or severely degraded S/N ratio at the PHY receiver, where the 'useful' signal content is lost in the jamming 'noise'. Consequently, the Thread nodes are incapable of both transmitting and receiving IEEE 802.15.4 packets. De-tuning the PHY's narrow-band transceiver to another channel however will remove the jamming event. Hence, a valid countermeasure

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.[23|24]xxxx
www.ndss-symposium.org

is to perform a channel hop to an alternative unoccupied frequency. This section describes an implementation of this countermeasure method on the Testbed; where the Thread devices were programmed with firmware written in the C programming language, incorporating Google's well-known open-source OpenThread API [9]. The developed countermeasure source files are available for download from the research group's public GitHub repository. The developed countermeasure source files are available for download from the GitHub [10].

1) Jamming Detection: Reliable jamming detection was implemented by monitoring the PHY layer's RSSI in the Thread device firmware. A threshold was set by measuring the minimum RSSI without jamming and adding a -12 dBm margin. If exceeded for 10 seconds, a channel hop was triggered. SED devices, which normally disable their radios to save power, were programmed to periodically assess RSSI like MED devices, detecting jamming when the threshold was surpassed.

2) Channel Hopping: After detecting a jamming event, each Thread node must prepare to jump to an alternative channel. In the experiment, an array of up to 16 IEEE 802.15.4 channels was created by the Thread Leader on initial creation of its network (with no jamming applied) by performing a frequency scan to determine available unoccupied channels, and populated its array. After sorting, the array contents are ordered randomly, with a Hop Index created to point to the next hop in the list. As Thread devices joined the network, the

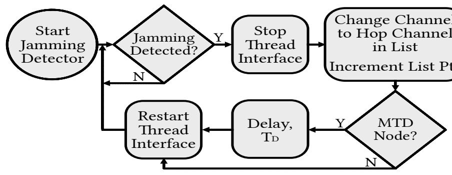


Fig. 2. Flow diagram of the implemented single-channel Countermeasure.

list and index were shared among nodes—Routers fetched data from the Leader, while Child devices obtained it from their Parent. This ensured synchronization for a coordinated channel hop during jamming. The countermeasure's flow diagram and timing are illustrated in figures, with equations defining recovery time. For networks with one FTD and multiple MTDs, hop time factors include jamming detection delay, channel change time, and Leader restart time. In larger networks, a fixed delay for MTDs ensures stable topology. Both MEDs and SEDs followed the same hopping mechanism.

C. Countermeasure Performance

Two experiments were completed on the Testbed to evaluate the performance of the Countermeasure, in increasing Thread network complexity; the first executed on a network topology comprising of just one FTD and up to 19 MEDs, and the second comprising five FTDs and 15 MEDs. Using the timing information contained in the on-device timestamped data and event logs maintained in each Testbed Thread device, it was shown that each node detected the jamming event and successfully performed a channel hop.

D. Jamming Attack Mitigation

The experimental results show that it was possible to repeatedly detect radio jamming of the Thread network and that a Countermeasure, by applying channel hopping, proved reliable. However, if the Jamming equipment is continuously tracking the network, then the networked system is perpetually compromised and unable to function. The network's Border Router(s) can be configured to broadcast an alert message (via Ethernet, cellular network, LoRaWAN, etc.) flagging the DoS attack, so action can be taken. In addition, the individual Thread devices can be programmed or configured to enter a 'safe' mode during the attack period. For example, a door lock should close when attacked or a security lamp should illuminate.

III. Conclusion and future work

We implemented a channel hopping countermeasure using off-the-shelf Thread hardware and demonstrated its effectiveness against jamming attacks on a bespoke Testbed. The countermeasure successfully performed continuous channel hopping until the jammer ceased tracking or the Thread system entered safe mode. Future work includes reducing hop latency, improving partial jamming handling, and exploring alternative defense strategies. All code is publicly available on the research group's GitHub [10]. This work is supported, in part, by EPSRC & DSIT funded projects (EP/X040518/1), (EP/Y037421/1), and (EP/Y019229/1).

References

[1] S. Pradhan, W. Sun, G. Baig, and L. Qiu, "Combating replay attacks against voice assistants," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 3, no. 3, sep 2019. [Online]. Available: https://doi.org/10.1145/3351258

[2] K. Lounis and M. Zulkernine, "Attacks and defenses in short-range wireless technologies for iot," *IEEE Access*, vol. 8, pp. 88 892–88 932, 2020.

[3] I. S. . W. Group, "Ieee standard for low-rate wireless networks," *IEEE Std 802.15.4-2020*, pp. 1–800, 2020.

[4] C. M. Ramya, M. Shanmugaraj, and R. Prabakaran, "Study on zigbee technology," in *2011 3rd International Conference on Electronics Computer Technology*, vol. 6, 2011, pp. 297–301.

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

[6] P. Yadav, N. Sagathia, and D. Wade, "Demo: Battery depletion attack through packet injection on iot thread mesh network," in *2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 2024, pp. 318–320.

[7] E. Bout and V. Loscrí, "An adaptable module for designing jamming attacks in wifi networks for ns-3," in *Proceedings of the 25th International ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 121–124. [Online]. Available: https://doi.org/10.1145/3551659.3559059

[8] E. Bout, V. Loscri, and A. Gallais, "Evolution of iot security: The era of smart attacks," *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 108–113, 2022.

[9] OpenThread, "API," https://openthread.io/reference/, accessed on 7th November 2024.

[10] SystronLab, "Thread Edge Testbed GitHub," https://github.com/SystronLab/thread-edge-testbed, accessed on 7th Nov 2024.

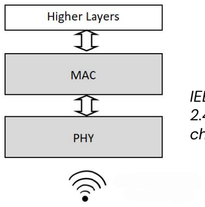
Poster: Evaluation of Radio Jamming Countermeasures in IoT Thread Networks

Poonam Yadav, Anthony Moulds, Peter Gillingham University of York, UK

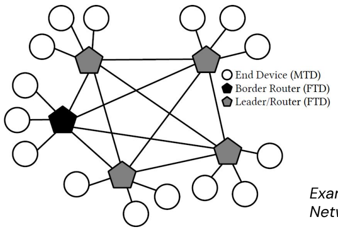
SYS TRON Lab

Problem and Motivation

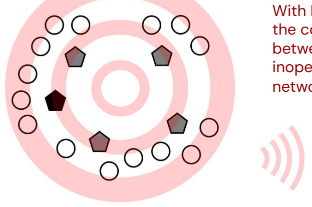
- The Thread protocol implements the IEEE 802.15.4 standard for establishing a radio network between IoT devices. This medium however results in the network being vulnerable to radio channel jamming.



IEEE 802.15.4 PHY & MAC 2.4 GHz (16 available channels)



Example of Thread Mesh Network



With Radio Jamming applied the communication links between Thread devices are inoperable resulting in network collapse.

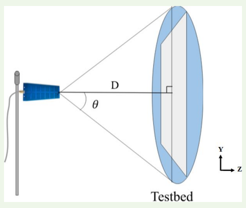
Jamming Signal

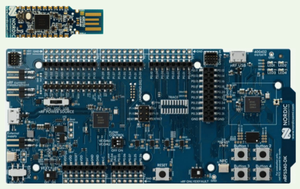
- We need a solution to countermeasure a radio jamming attack, enabling Thread network to recover gracefully.
- Countermeasure must be hardware agnostic and available as software in the form of an API.

Testbed and Setup

A bespoke Testbed was specially designed and built to perform the experiments with Thread networking, enabling easy placement of Thread devices to a wall mounted magnetic panel. A Thread network made up of 20 nodes was used, consisting a Border Router, 4 Full Thread Devices (FTDs) and 15 Minimal Thread Devices (MTDs).

The network jamming attack was applied through IEEE 802.15.4 packet injection from a Sewio OpenSniffer device, injecting empty packets (no payload) into the network with minimal interframe spacing, rendering the network unusable. To conduct the attack, a Yagi antenna was pointed at the testbed at a distance of 1.5 m and transmitting at +17 dB, resulting in a minimum received signal strength indicator (RSSI) value across the testbed of -48 dB.





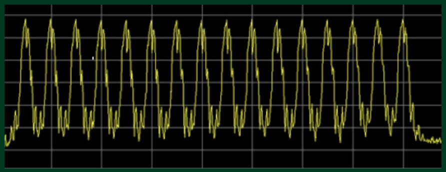
Left: Yagi antenna pointed at the Testbed.

Right: nRF52840 Dongle MTD (top) and nRF5340DK FTD (bottom) used in the experiments

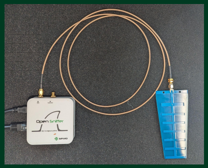
Jamming Attack Scenario

- With 30 billion Internet of Things (IoT) devices set to being connected globally by 2030, there are growing concerns around network security.
- We demonstrated how simple it is to apply a denial-of-service (DoS) attack on a Thread network using readily available off-the-shelf (OTS) equipment.

Jamming a Thread network is achievable by saturation of the network with continuous IEEE 802.15.4 packets, only requiring knowledge of the network's radio channel number. The channel can be discovered by using OTS hardware, such as the nRF5340DK (shown above).



The single-channel jamming system developed for the research is capable of tracking and jamming all sixteen Thread network radio channels in succession.

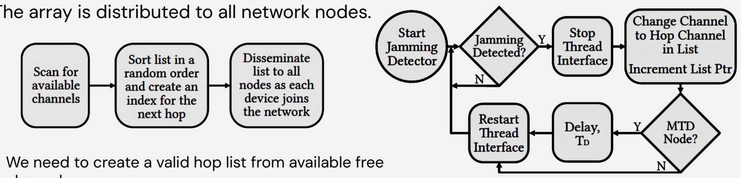


The Sewio OpenSniffer is a readily available OTS device for use in packet injection onto a given IEEE 802.15.4 network channel.

With the OpenSniffer combined with an nRF5340DK, forming the jamming system, a potential attacker could easily and inexpensively detect and disrupt a Thread network, leading to sensors and security cameras not transmitting for example, or IoT locks not being operable and security alarms being unusable.

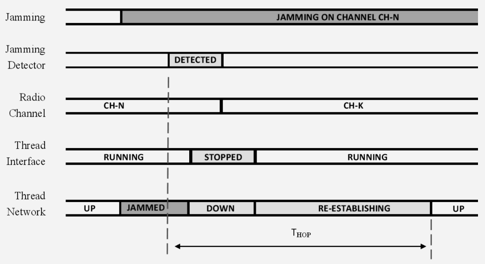
Countermeasure Design and Results

- Implements 'channel hopping' to countermeasure continuous jamming attacks; uses OpenThread API to ensure compatibility with most Thread development platforms.
- Capable of continuously hopping channels specified in an array determined by the Thread Leader device; the channel hop array is randomized and ordered.
- The array is distributed to all network nodes.


Scan for available channels
Sort list in a random order and create an index for the next hop
Disseminate list to all nodes as each device joins the network
Start Jamming Detector
Jamming Detected?
Y
N
Stop Thread Interface
Change Channel to Hop Channel in List
Increment List Ptr
MTD Node?
Y
N
Delay, TD
Restart Thread Interface

We need to create a valid hop list from available free channels.

Countermeasure flow diagram.



$T_{COUNTERMEASURE} = T_{JD} + T_{HOP}$

$T_{HOP} = T_C + T_L + \max\{T_{MTD1}, \ldots T_{MTDN}\}$ for single FTD

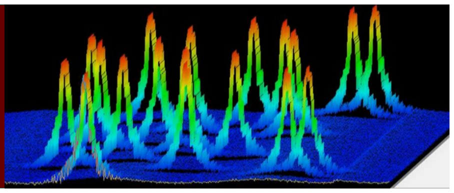
$T_{HOP} = T_C + T_L + \max\{T_{D1}, \ldots T_{DN}\}$ when $T_D > T_R$

T_{JD} = Jamming detection time
T_C = Channel change time
T_L = Time for Leader to restart network
T_{MTD} = Time for MTD devices to re-attach
T_D = Optional device attachment delay
T_R = Time for Router devices to re-attach

Results show Channel Hop T_{HOP} in 30 sec (max) for a Thread mesh network of 20 devices.

A 3D Spectrogram (X,Y,Z: frequency, power, time) of network channel frequency, showing the countermeasure applying consecutive channel hops during continuous jamming (where the jammer is capable of tracking network channel changes).

All sixteen channels are shown hopped.



Conclusion and Further Work

We demonstrated how relatively simple it is to jam or disrupt Thread radio links and apply a network-wide DoS. An implementation of a channel hopping countermeasure using real off-the-shelf Thread-capable hardware was proposed, and its effectiveness demonstrated. We showed the success of this defense mechanism against an applied jamming attack using a bespoke Testbed. Its ability to successfully channel hop continuously through a list of randomized available channels was shown. The countermeasure runs until either the jammer stops tracking the network or the Thread system enters a safe-mode fallback.

Further Work: reduce countermeasure channel-hop latency, better handling of partial network jamming, research defence against multi-channel jamming. This work is supported by EPSRC & DSIT funded projects (EP/X040518/1), (EP/Y037421/1), and (EP/Y019229/1).

Read about our research

systronlab.github.io/projects
poonam.yadav@york.ac.uk