UCLouvain

École polytechnique de Louvain

Honeypot for IoT networks using Matter/Thread

Author: **Julien GOURGUE**
Supervisors: **Cristel PELSSER, Ramin SADRE**
Readers: **François DE KEERSMAEKER, Tom ROUSSEAUX**
Academic year 2024–2025
Master [120] in Computer Science

Abstract

The adoption of the Internet of Things (IoT) is growing every year, transforming a wide variety of sectors, including healthcare, cars and smart homes. Smart homes are becoming popular, aiming to make our daily lives easier by performing simple or complex tasks such as closing the curtains at sunset or turning on the heating when a certain temperature is reached. However, this growing connectivity also introduces security challenges. As more devices become connected, new security breaches in home networks may arise. To counter this, research and testing are essential.

In this context, I designed and implemented a Matter over Thread honeypot. Matter over Thread is a new technology standard for IoT that is designed for reliable and secure communication between smart devices. The honeypot simulates a vulnerable smart home network and monitors the behaviour of potential attackers on such a misconfigured network. To prevent the honeypot from compromising the actual home network, I detail methods for its secure isolation. Furthermore, I present three attacker scenarios to demonstrate the types of data that can be collected and analysed by the honeypot.

Acknowledgements

First and foremost, I would like to thank my two supervisors, Professors Cristel Pelsser and Ramin Sadre, as well as François De Keersmaeker, for their availability and for all the feedback they provided during this master thesis. I would also like to thank Tom Rousseaux for agreeing to be part of the jury.

Finally, I would like to thank my family and friends for always listening to me during this project, and my flatmate for letting me install all these devices in our living room.

Contents

List of Figures

1 Introduction

Internet of Things (IoT) adoption is growing every year and will continue to grow in the future. Based on the number of IoT Analytics [27], there are in 2024, over 15 billion connected IoT devices worldwide, this number is expected to double by 2030. In 2024, 70.5% of IT devices are IoT devices. With such numbers, we can understand the huge impact of this field in our daily lives. IoT goes from medical field to industry, cars and even smart home.

Smart homes are also becoming more common. It will facilitate our daily life by performing simple or complex tasks. In 2023, the estimated number of smart home users is about 360 million, the global number of smart homes was forecast to continuously increase between 2023 and 2028 by a total of 425.5 million users. The indicator is estimated to reach 785.16 million users in 2028 based on Statista [23]. Most of the biggest tech companies such as: Apple, Amazon, Samsung, Google, Phillips and many more, have understood the importance of the market and have started to produce IoT devices.

At the beginning of adoption of IoT devices for smart homes, each company had its own IoT environment. To add a device of a certain brand, you need the gateway of the same brand and use their official mobile app to control your devices. As users have most of the time devices of different brands in their smart home, it becomes more and more difficult to manage everything, you had to open different app for different purpose, so the argument that IoT makes the life easier become more and more a nonsense. To counter this problem, four companies i.e. Google, Amazon, Apple and the Zigbee Alliance decided to create a new standard called Matter. A standard can be seen as a universal language that all IoT devices will speak, this approach makes all IoT devices able to talk to each other and force companies to use the same standardised message for a specific action. For example, to turn on a light, the message will be the same for every Matter light, so that every Matter controller can interact with every Matter light. This makes life easier for the user, as they only need one controller to control all their Matter devices, regardless of the manufacturer. Matter operates at the application layer of the OSI network model and can be used with two network protocols: Wi-Fi and Thread. Thread is an IPv6-based protocol designed for IoT.

As IoT becomes more ubiquitous, it also means that cybersecurity vulnerabilities can be added anywhere. For example, today a simple refrigerator can become an open door to your home network if some attackers find a way to compromise it. In other cases, IoT can be used in a more critical context, such as in a nuclear power plant or in healthcare, where the security of the devices is mandatory and a security breach could have catastrophic consequences. Therefore, research and development in this area is essential to protect users, infrastructure and data.

A good way to discover vulnerabilities can be the use of honeypots. Honeypots, especially research-focused ones are intentionally vulnerable systems de-

signed to attract attackers and analyse their behaviour, techniques, and tools. These systems provide valuable insights into real-world attack strategies and can help to discover unknown vulnerabilities in a controlled environment. Another common approach to vulnerability discovery is fuzzing, which involves automatically generating and sending malformed or unexpected inputs to a system in order to provoke crashes or anomalous behaviour that may reveal security flaws.

In this master thesis I developed a honeypot for IoT networks using Matter over Thread. The goal of the honeypot is to be able to capture what an attacker could try to do on a misconfigured network and understand their behaviour and knowledge about the technology. The paper is organised into eight sections. Following this section(1), section 2 provides background information, while section 3 reviews the state of the art and discusses related work. The core of the thesis begins with the design of the network in section 4, followed by the implementation of the sniffer component in section 5, and the honeypot itself in section 6. section 7 evaluates and validates the results obtained from the honeypot. Finally, the thesis finished with the conclusion and outlook in section 8.

2 Background

Matter over Thread is a new technology standard for IoT that is designed for reliable and secure communication between smart devices. As the framework is still at its early stage, I will introduce you to the key elements of these technologies. Additionally, I will also introduce the IEEE802.15.4 standard that is used as the physical layer and medium access layer (MAC) of the Thread network. The figure 1 represents the OSI model of Matter when using over Thread. The Bluetooth Low Energy (BLE) is only used during the commissioning where a controller, like a smartphone, will send Thread data used for commissioning to the commissionee. I will finish this section by defining what a honeypot is.

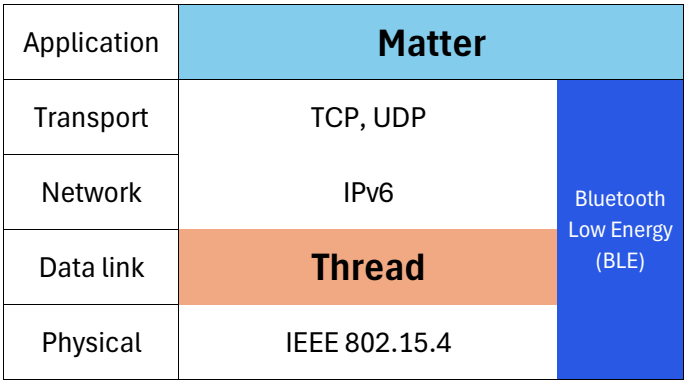


Figure 1: OSI model of Matter/Thread

2.1 IEEE 802.15.4

The IEEE 802.15.4 standard [29] defines the operation of low-rate wireless personal area networks (LR-WPANs). It specifies the physical (PHY) layer and the medium access control (MAC) layer. This standard serves as the foundation for several network protocols, including Zigbee and Thread. Both protocols use the IEEE 802.15.4 standard to ensure robust and energy-efficient communication that is mandatory for Internet of Things (IoT) applications.

2.1.1 Channels

The IEEE 802.15.4 uses channels across different frequency bands to communicate. For Thread networks, the compatible channels are between 11 and 26. The channel is defined during Thread network creation, and all the devices from the PAN should use the same one.

2.1.2 Addressing

The IEEE 802.15.4 standard defines two types of addressing [9]: the short address(16bit) and the extended address(64bit). There is also the PAN ID, which is a 16-bit identifier that defined the personal area network (PAN) to which a device belongs.

The short address *0xFFFF* is reserved for broadcasting messages to all devices in the PAN; devices do not have to ACK these messages. For unicast messages, the short address used is the Thread's RLOC16 2.2.3, this address is unique inside the PAN but can change if there is modification on the network topology. On the other hand, the long address is unique between all devices and so can be used to communicate with a device from a different PAN. To obtain these addresses, devices used the MLE protocol detailed in section 2.2.4.

2.1.3 Frame types

There are three main frame types in the IEEE 802.15.4 standard.

- **Short source and destination addresses**: This is the common frame type used in the network. It is used for unicast messages.
- **Long source and destination addresses**: In some cases, the short address may not be available yet, or the device may prefer to leverage the uniqueness of the long address. In such situations, the long addresses are used for unicast communication instead. Since long addresses are unique, this kind of frame can be used inter-PAN and intra-PAN.
- **Long source and short destination addresses**: This frame type is used during the join process, when the source device does not yet have a short address. If the destination is *0xFFFF*, the frame is a broadcast message.

2.2 Thread

Thread [57] is a network protocol designed for IoT. It is an IPv6-based protocol and uses the power-efficient IEEE 802.15.4 physical and MAC layers to make its wireless mesh network. To understand Thread, I mainly recommend two sources: the official Thread Specification [58] (or a shorter version with only the fundamentals [60]) and the OpenThread guide [66]. The information below mainly comes from these sources.

2.2.1 Benefits

The Thread group [59] defined four mains benefits of using Thread :

- **Built for the internet of things** : Thread is designed specifically for IoT and use low-power and low-latency proven standards.

- **Convergence and Coexistence on Thread** : Thread is application layer agnostic, so it offers the flexibility to choose the desired app layer to connect devices (for example Matter).
- **Thread is Flexible and Future Proof** : Application layer can be easily changed over time due to the application-layer agnosticism of Thread. As Thread is IP-based, developers can maintain a connection to their products and their clients while keeping interoperability for a large amount of their IoT devices.
- **Global Solution** : Thread is based on existing technology in all its layers. Thread is also an open standard that is not maintained by a specific manufacturer, which minimise the risk of incompatibility.

2.2.2 Device Types

There are multiple device types and roles used within a Thread Network.

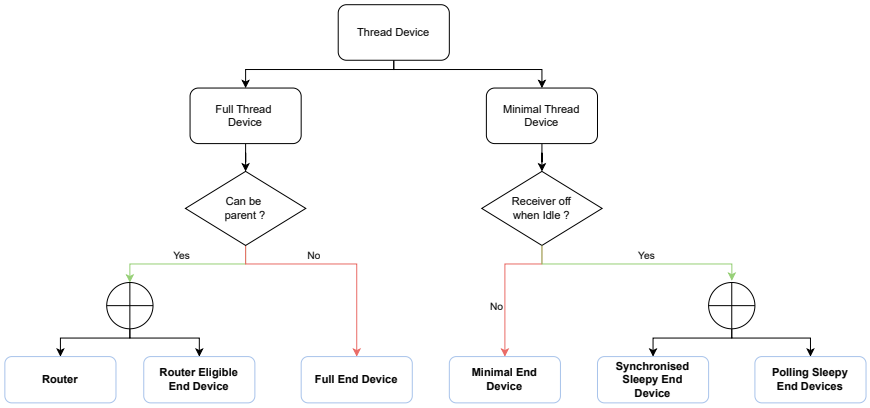


Figure 2: Thread Device Types classification based on chapter 4.1.2 from the Thread Specification [58]

The figure 2 shows all the different types of devices in a Thread network.

- **Router** : router provides routing service to forward packets on the network. It also provides secure commissioning services. To do so, a router device need to always keep its transceiver enabled (i.e. always be awake and ready to receive messages).
- **Router Eligible End Devices** : are devices that can become router but due to network topology or conditions, they are not acting as router yet.
- **Full End Device** At the opposite of Router Eligible End Device, they are nodes that cannot be promoted to a router.

- **Minimal End Device** Are devices that have a significant resource constraints but does not turn off their receiver when Idle.
- **Sleepy end device** : Are devices that can only communicate through their parent router and cannot forward messages. There are two types of sleepy devices :
  - **Synchronised** device synchronises their wake-up with the network
  - **Polling** device wakes-up at regular intervals to check for incoming messages

In addition to device types, there are also roles. There are two specifics and important router roles :

- **Thread Leader** : the thread leader is the router that manage a thread network partition. It is dynamically selected among the routers to ensure fault tolerance. There is only one leader at a time in the network.
- **Border Router** : the border router is a specific type of device that link the thread network to adjacent networks, like for example the Wi-Fi. There may be several on the Thread network.

2.2.3 IPv6 Addressing

A single Thread device has multiple IPv6 unicast addresses, each used for a specific function. There are the different kinds of addresses [32] :

Routing Locator(RLOC) The routing locator is one of the IPv6 addresses that identifies a Thread Interface, based on the location of the device in the network. The last 16-bit value of the RLOC is derived from the Router ID and Child ID and is also called the **RLOC16** (last 16-bit of the RLOC).

Because the RLOC is formed, in part, with the router and child identifier, it can change if the topology changes, for example, if the device is attached to a new router after some time. IEEE802.15.4 sets its Short Address to the Thread RLOC16.

Mesh-Local EID is the address that identifies a Thread interface and is independent of the topology; it will never change. The value is completely random and defined after the commissioning.

Link-Local Address is the address that is used to discover neighbours, configure links and exchange routing information. The address always has a prefix of *fe80::/16*

2.2.4 Thread Network

The network is identified with three unique identifiers :

- PAN ID : 2-byte personal area network identifier
- XPAND ID : 8-byte extended personal area network identifier
- Network name

Mesh Link Establishment (MLE) The initial MLE framework was defined by the IETF draft: *6lo-mesh-link-establishment-00* [33]. However, Thread devices implement a modified version that is described in Chapter 4 of the Thread Documentation [58].

Thread uses MLE for maintaining the route cost between devices, for establishing and configuring the radio connection and for detecting new neighbours. MLE messages use IPv6 and UDP for the network and transport layers. The source address should be a link-local address and an IPv6 unicast or multicast address as the destination. MLE is also used to distribute configuration values that are shared across the network such as the PAN ID, the channel, etc.

Once a new device enter the Thread network, the MLE messages can exchange data with it. The MLE messages distribute or exchange the following information :

- The 16-bit short address (i.e. RLOC16).
- The 64-bit long addresses.
- The type of the device and related data (i.e. the sleep cycle for a Sleepy End Device).
- Neighbour link quality and link cost (if a Router).
- Routing cost to all routers.
- Updates of the network (i.e. modification of the channel or the PAN ID, ...).
- Security material and frame counters between devices.

Joining phase When a new device wants to join a Thread network, it starts by making a scan for 802.15.4 networks. Once the scan is finished, it configures its Channel, PAN ID, XPAN ID and network name via Thread commissioning, then uses MLE to be attached as a child to a parent device.

2.3 Matter

Matter, initially called Connected Home over IP and abbreviated to Project CHIP, is a freely available standard for smart home IoT. The standard created in 2019 by Google, Apple, Amazon and the Zigbee Alliance. This project is now called the Connectivity Standard Alliance (CSA) [12]. The idea of the standard is to unify all companies to facilitate the utilisation of devices from different brands.

2.3.1 Goals of Matter

The CSA[12] gives four goals of the Matter technology:

- **Simplicity** : Matter wants to be easy to configure and to use.
- **Interoperability** : Matter wants to natively let devices from multiple brands work together.
- **Reliability** : Matter wants to be consistent and responsive.
- **Security** : Matter uses robust security configuration.

2.3.2 Architecture overview

Matter aims to build an IPv6-based communication protocol. The Matter protocol defines the application layer; this layer specifies the functionality to receive and send data from other devices.

The architecture of Matter is divided into multiple layers to help the separation between each main part that has a different role to play in a device. The following stack represents the interaction flow. This section comes from the documentation of the project CHIP [64]. The stack is divided into 7 layers as follows:

- **Application** : The high order logic of a device, for example for a Matter bulb, it may be the logic to turn on/off the light.
- **Data model** : Is the data that is used to support the functionality of the application.
- **Interaction model** : Defines the available interactions that can be perform between a client and a server. The interaction will modify data from the data model layer and the application layer will apply the logic on the modified data to change the state of the device.
- **Action framing** : It is the serialisation of the interaction model to fit the binary format use during network transmission.
- **Security** : This layer encrypts and sign the payload of the frame generated in the previous layer.

- **Message Framing and Routing** : Constructs the payload format with required and optional header fields.
- **IP Framing and Transport Management** : This layer will send this payload to the transport protocol.

2.3.3 Device data model

In Matter, all devices are composed of **nodes**. A **node** in Matter is a resource that can be seen as a functional whole, i.e. light bulb, smart plug, etc. Inside a node, there is a set of **endpoints**; each endpoint is a feature of the device, like, for example, turning on a light, but it can also be used for utilities such as over-the-air updates (OTA). The information in this section comes from the Google developer centre of Matter [37].

Node roles Each node may have one or more roles from the following ones:

- **Commissioner** : A node that performs the commissioning, this will be detailed in the section 2.3.5.
- **Controller** : A node that controls other node(s). Like for example Google Home App, Home Assistant, ...
- **Controlee** : A node that can be controlled by a Controller, for example a Matter light bulb.
- **OTA Provider** : A node that can provide the updates.
- **OTA Requestor** : A node that can request an update.

Clusters A cluster is a group of specific endpoints such as turn on/off a light. Clusters define specific behaviours and data structure for different device types. This approach enables interoperability between manufacturers, as all Matter devices should use the same cluster for the same type of device. These clusters are implemented and provided by the Connectivity Standards Alliance (CSA).

2.3.4 The Fabric

The Matter's fabric is the group of devices that trust each other and communicate with each other. The group is only accessible to devices that have proven themselves trustworthy. During the commissioning phase, the new device will receive a unique security certificate; this certificate identifies the node as a member of the fabric.

2.3.5 Commissioning

The commissioning is the process to add a new device in an existing Matter fabric. To commission a Matter device onto the network, the device needs to have an onboarding payload. This payload includes the vendor ID, product ID, passcode, discriminator and the discovery capabilities bitmask. All these information can be provided in different formats; the most common one is via QR code or manual pairing code.

I will not explain every stage of the commissioning phase but only the more relevant in my sense, more details can be found on the Google Developer Centre [36].

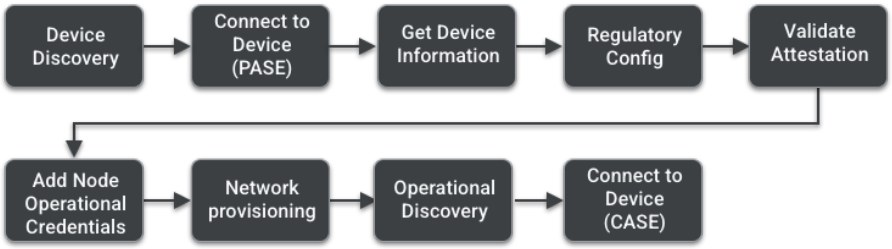


Figure 3: Commissioning flow from [36, Google Developer Center]

The figure 3 shows the complete flow for commissioning in Matter. I will detail some of them:

PASE Passcode Authenticated Session Establishment is used to establish a secure communication channel between the commissioner and the commissionee. Once the session is established, all messages will be encrypted with the PASE-derived key.

Validate attestation The goal of this step is to determine if the device is a real Matter device. To do so, the commissioner extracts the payload of the commissionee and does a challenge request that should be signed with the attestation private key of the manufacturer.

CASE Certificate Authenticated Session Establishment is used to derive an encryption key that will be used to establish a secure communication between the Commissioner and the Commissionee. Once it is created, all unicast messages will use it.

From the user's point of view, the commissioning is easy because of the interoperability, as every Matter device should use the same commissioning process. The user only has to scan the QR code using its controller app on its mobile phone. The phone will use BLE to connect to the Matter device and

exchange data to start the commission. Then, after some seconds, the device will be added to the Matter fabric.

2.4 Honeypot

Honeypots are fake resources and/or services that are intentionally vulnerable. The goal is to attract attackers, making them believe they are in a real system, to analyse their behaviour in such an environment. This computer security technique can be used to detect unknown security breaches in a system by allowing attackers to do what they want. It can also be useful for observing if attackers are interested in certain technologies, such as IoT.

2.4.1 Interaction level

There are two main interaction levels for honeypots: high and low interaction.

Low-interaction honeypots Low-interaction honeypots emulate one or more services, such as port 80 for HTTP. They simulate only simple functions of the service(s) and do not allow an attacker to access the operating system. While they are easier to deploy, they are also easier for an attacker to detect, as they provide only a portion of a computer system.

High-interaction honeypots provide more interaction by simulating the operating system of, for example, a server. They collect information about every action attackers perform on the operating system. Honeypots of this type are riskier because everything they allow attackers to access is on real resources. They are also more complex to set up, but they can gather more data because they simulate a whole network.

2.4.2 Production vs Research Honeypot

The objective of a honeypot is always to lure attackers with fake data, but there are two main types of honeypots: the production honeypot and the research honeypot [6].

Production honeypots are deployed in real-world networks, such as company networks, to protect the system by luring attackers. Most of the time, this kind of honeypot does not have high interaction and is easier to maintain.

Research honeypots are designed to collect information about the methods and tactics that attackers use. Most of the time, these honeypots are high-interaction to understand in detail what an attacker would perform in such a network. Their purpose is to collect data for studying emerging threats, rather than providing direct protection for operational systems.

3 State of the art

As Matter over Thread is a new technology, there is not a lot of research on this topic. This section will present some relevant paper links to my work. I separate these papers into three main themes: Thread, Matter or IoT Honeypot.

3.1 Thread

The paper [61] provides an overview of the Thread protocol and discusses all the network layers used by Thread. From my point of view, the paper is useful to start learning Thread and can be read if you want to go deeper than the section 2.2.

In the paper [3], the authors aim to analyse the security of Thread networks by repurposing tools that were used previously to analyse ZigBee networks. They assume that the attacker does not have any knowledge about cryptographic keys of the network and that the attacker has the necessary hardware at its disposal. They use development boards that were flashed with OpenThread [44] binaries. Then they analyse packet exchange between devices. They were able to develop an energy depletion attack and an online password guessing attack.

3.2 Matter

In the paper [35], the authors developed the first Matter fuzzer called *mG-PTFuzz*. The project was done at the early stage of Matter, and their idea was to use the extensive and detailed information from the Matter specification for test input generation. Converting human-readable documentation to machine-readable information is time-consuming and error-prone, so they leverage a large language model to successfully automate the conversion process. To test their fuzzer, they made an extensive evaluation involving 23 Matter devices. They found 147 new bugs, including 61 zero-day vulnerabilities.

Unlike my work, they simulated an entire Matter environment and used the specifications to find bugs. For my part, Matter is run on real devices with which an attacker can interact. Combining our approaches to determine whether a Matter device complies with its specifications and whether these specifications are secure could give us confidence that Matter is secure.

In the paper [5], the authors use Mealy machine inference to analyse the security of the Matter commissioning process. A Mealy machine is a finite-state machine whose outputs are determined both by its current state and the current inputs. Their Mealy machine captures the system behaviours of Matter commissioning in terms of states, inputs, and outputs. Once the state machine was created, they discovered potential vulnerabilities, including unexpected message handling and unintended state transitions, which can be used for DoS or replay attacks.

In the paper [34], authors analyse the feasibility of a hidden eavesdropping attack in a smart home using Matter. The interoperability of Matter introduces new challenges for the privacy and security aspect. They describe a scenario where an attacker adds its controller to a Matter fabric with the authorisation of the owner, like, for example, during a holiday rental, and then uses its controller to hide a second hub in the house. In such a scenario, the attacker can turn off its "official" hub in front of the owner, the owner can still have access to the network via its original hub. The problem is that the second hidden hub of the attacker is always connected to the network without the owner's knowledge. In such a scenario, the attacker still has access to the matter fabric via the hidden hub. It is a security breach because now the attacker can know the state of IoT devices, such as if the door lock is closed or not. They propose a solution that can be added to Matter to avoid these kinds of attacks, for example, having an admin controller that is able to see every controller of the fabric and remove them if necessary.

Such an attack can be tried remotely on my honeypot, as the attacker can access Home Assistant via the internet (detailed in section 6). As the attack is possible in my network, my honeypot can be used to determine if attackers will try such an attack on a poorly configured Matter/Thread network.

The paper [55] analyses the fact that a Matter device needs a device attestation credential to indicate it comes from a trusted vendor, but for a Matter controller, such a certificate is not required. In other words, everyone can create a Matter controller able to interact with a real commercial Matter device. The authors analyse this design choice of Matter and present scenarios where a malicious controller can exert harm to a healthy Matter system. One of the present scenarios is the following: if a user has a fabric with legitimate devices and then buys a new device. To commission the new device, the user decides to use a non-legitimate app that is in fact infected. The app can control the new device and add it to its own fabric. If, for example, this device is a thermostat, the malicious app can interact with it. But more than that, if in the trusted fabric there is home automation, for example, to open a window if the temperature is too high, the malicious app can increase the temperature, and a trusted device, in response, can open the window. They conclude that a check on the trustworthiness of the controller will increase the security of Matter.

3.3 IoT Honeypot

The paper [15] is related to ZigBee, which is an IoT network protocol that uses the IEEE 802.15.4 standard, like Thread. In this paper, the author creates a honeypot that simulates a gateway to a ZigBee network. The gateway is accessible via SSH and contains fake PCAP files with honey tokens, i.e., fake data. The honeypot logs if the honey tokens are found, for example if the false PCAP file are open or download to the attacker computer. In these PCAP files, The authors added a fake external IP showing where fake sensitive medical data is supposedly sent. The goal was to see if an attacker would analyse the ZigBee

pcap file to find the IP address. The IP points to a PHP server that listens and records all incoming requests. While the honeypot was attacked, no attacks showed any understanding of ZigBee networks.

Apart from the fact that their work is based on ZigBee and mine on Matter/Thread, the aim of our honeypot is to analyse the behaviour of attackers when confronted with information on an IoT network. In my work, in addition to honey tokens, I also give access to a whole real IoT network via Home Assistant to make an attacker even more interested in using the system. Then, I am able to analyse all the actions than attackers performs on the Matter/Thread network.

The master's thesis [24] is an implementation of a honeypot for smart homes. The author analyses an IP camera to gather some interesting results (e.g., port use, use of cloud, ...). Then, he made a honeypot that simulates this IP camera and explains how to implement a similar one. He analyses the result of its IP camera honeypot and concludes that such a device can have a small difference to a real one and so not be easily detected by an attacker, almost impossible, due to the lack of information on the "normal" behaviour of this device. Then he does an analysis of attacks against the honeypot. They discovered SSH attacks on the server but also crypto miner attacks and password leakage. Some attacker even tried to remotely execute malicious code on the camera.

The difference with my work is that he simulates a GUI of a real camera device to analyse the behaviour of an attacker on such a device. Such a device can be added to my honeypot to add a "breach" and enable a richer set of behaviors.

The survey [22] is a complete survey of the research that has been done on honeypots and honeynets for IoT. The paper provides a taxonomy and an analysis of existing honeypots. It also points to future research areas and the need for advanced honeypot systems and collaboration across sectors to improve security.

The paper [8] is, like me, a master's thesis subject at UCLouvain. This thesis was done in 2024 and is the predecessor of mine. The IoT network part of our honeypot has similarities; we both use Matter/Thread devices running on ESP and connected to Home Assistant. The main difference is the honeypot part. To attract attackers, they open the access to Home Assistant. Then, they monitor:

- Home Assistant [26] logs
- OpenThread add-on [2] logs
- Matter add-on [51] logs
- Proxmox [49] logs
- Network traffic

To make all the data human-readable, they decided to use Grafana [25]. To collect data, they open access to the honeypot via the Internet and simulate some famous attacks (man-in-the-middle, DoS). In my work, I do not use anything inside the Home Assistant and its add-ons. As I give attackers access to it, they can decide to stop the logging system, and we will not be able to retrieve the data. I also added a DMZ to the network so as not to compromise the security of the local network.

4 Network design

This section describe the design of the network that is used in this work. The objective of a research IoT network is to be as realistic as possible while still being manageable. Multiple choices were made to achieve this goal, which will be discussed in this section.

4.1 Matter devices

First, a Matter over Thread network requires compatible devices. We have several options:

- Fully simulated devices on a computer
- Emulated devices on a real hardware platform (e.g., ESP32, Nordic Semiconductor, etc.)
- Real factory devices

The problem with fully simulated devices on a computer is that they need to be able to use IEEE 802.15.4 to communicate, which requires hardware. The Real factory devices are the more realistic option, as they are devices sold for real-world use. The problem with those is that, firstly, they are expensive, and secondly, there are not many devices that support Matter over Thread yet. You can find on the Matter database website [38] a list of devices that support Matter.

The best option, as I had several ESPs at my disposal, was to use emulated devices on a real hardware platform. The big advantage of this is that it is relatively cheap and easy to set up. On the official Matter GitHub repository [10], we have a bunch of example devices that we can flash on ESP. To do so, it is possible to use ESP-IDF to build and flash a device. Another way to easily flash ESP devices is to use the *ESP-Zerocode*[19] website. This website allows you to select the kind of Matter device you want (e.g., dimmer, plug, etc.) and the ESP device you have. Then it will generate a firmware that you can directly flash on your ESP device from the website.

Device	Matter over Wi-Fi	Matter over Thread
ESP32-C3	Yes	No
ESP32-H2	No	Yes
ESP32	Yes	No
ESP32-C6	Yes	Yes
ESP32-C2	Yes	No
EPS32-S3	Yes	No

Figure 4: ESP devices compatibility with Matter [20]

There are multiple ESP devices, but not all of them support Matter over Thread. The figure 4 shows the compatibility of ESP devices with Matter.

4.2 Controller

Once the devices are selected, the next step is to set up the IoT network like in a smart home. To control the devices, we need a controller. The controller (or hub) is the device that manages the network, sends instructions, and receives data to the devices. There are several proprietary choices for the controller, such as:

- SmartThings hub from Samsung and Aeotec [56]
- HomePod [28] and Apple TV [4] from Apple
- Google Nest Hub [39]
- Amazon Echo [16]
- etc.

The problem with these controllers is that they are proprietary and do not allow for much customisation. The other option is to use a self-hosted open-source controller. The most popular and widely used open-source controller is Home Assistant [26].

4.2.1 Home Assistant

Home assistant [26] is an open-source software that allows you to control all your devices from one place and automate them. It is a really powerful tool for smart homes and is widely used in the community, which means that there are a lot of add-ons and integrations available. In Mars 2025, the number of Active Home Assistant Installations were more than 450.000 based on the Home Assistant analytics website [30]. This number only count users that share their

usage data so it does not reflect the total number of users, they estimate that a third of all Home Assistant users shared them.

By default, Home Assistant does not support Matter, but there are add-ons available.

Matter Server add-on [51] is a Home Assistant add-on that integrates Matter protocol support into Home Assistant. It functions as a Matter Controller Server, enabling Home Assistant to manage and interact with Matter devices using a WebSocket interface. This add-on allows Home Assistant to discover, control, and automate Matter devices. It also provides support for device pairing, updates, and data transmission.

Thread Border Router add-on [2]. To be able to use our Matter over Thread devices, we need a Thread Border Router. This add-on will do that for us. It will act as the leader of the Thread network. The add-on is built from OpenThread and uses a REST API to communicate with the Home Assistant. A REST API is a set of rules that allows applications to communicate over HTTP using standard operations, mainly GET and POST. To be able to communicate with devices, the Thread Border Router must be able to use IEEE 802.15.4, which requires hardware.

For this project, I use Home Assistant as the controller, with the Matter Server and Thread Border Router add-ons.

Home assistant can be installed in multiple ways :

- **HA OS** : is a full operating system that runs Home Assistant.
- **Supervised** : is a Home Assistant installation that runs on a generic Linux system.
- **Container** : which is a Home Assistant installation that runs on a Docker container.
- **Core** : which is a Home Assistant installation that runs on a Python environment.

For this project, I needed to be able to install add-ons for Matter and Thread; the Core and the Container installation do not support add-ons. To be able to manage the add-ons, Home Assistant requires the Supervisor, which is only available in the HA OS and supervised installation. I chose the OS installation, as it is the easiest to set up and manage. I did not want to dedicate a whole machine to Home Assistant OS, so I used a virtual machine that runs the HA OS.

The other advantage of Home Assistant is that we can use the Home Assistant mobile application. This application will let us commission new devices into the network and control them from our phones.

4.3 Demilitarised zone (DMZ)

A DMZ is a network configuration that is used to separate a LAN network from untrusted networks (like the internet). The honeypot network topology described in the previous section should be reachable from the internet. On the other hand, the home network, where personal devices are connected to the internet, should not be exposed. In such a configuration, a DMZ is the perfect way to let the honeypot receive untrusted traffic while preventing traffic from reaching the local network. There are several ways to configure a DMZ; I will present two of them and then explain the pros and the cons for each.

4.3.1 Classic DMZ configuration

The first configuration is the most common one. It consists of two firewalls; the first one is used to filter which traffic can go on the DMZ, and the second one is used to avoid untrusted packets going inside the local area network (LAN).

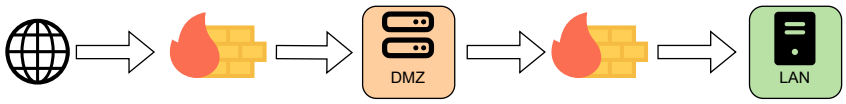


Figure 5: Classic DMZ configuration

Each firewall can run on a router. Most of the time, ISP routers are not fully customisable, as ISP companies often restrict their configuration to limit users misconfigurations. To be able to configure a custom firewall, it is recommended to bridge all incoming traffic from the ISP router to a customisable router. This approach makes it easier to properly configure the firewall and just use the ISP router for providing the internet access.

The pro of such an approach is that we have complete control over the network. The ISP router only sends packets to the first router. The configuration can be customised to suit your needs. But on the opposite side, all the firewalls have to be correctly configured; a small error can completely break the internet connection of the LAN or, worse, can let malicious users send packets to vulnerable devices in the LAN.

4.3.2 Host DMZ configuration

The second approach involves using routers to create two separate LANs: one for the secure home network and another for the honeypot network. This is achieved by configuring the ISP router's DMZ option to redirect untrusted traffic to the DMZ router. However, this option may not be available on all ISP routers. In Belgium, for instance, the Proximus b-box 3V+ router, the VOO Netgear router, and Pegatron routers offer this DMZ feature.

It's important to note that the DMZ host option on the ISP router does not create a true DMZ. Instead, it redirects any packets that are not expected

to the DMZ. The ISP router considers a packet "not expected" if its port is neither part of the NAT nor listed in the port forwarding configuration. These packets are considered unsolicited, as no device within the network specifically requested them.

The three routers in this setup use DHCP with different IP address ranges. NAT is enabled on each router to prevent devices from one LAN from communicating with devices on the other LANs.

Once the DMZ router receives untrusted packets, it determines whether to forward them to a device on the network or drop them if they are not authorised. The router for the secure home LAN is optional, as the ISP will route all untrusted traffic to the DMZ. Devices connected to the ISP router will only receive trusted traffic. However, adding a dedicated router for the secure home LAN can enhance security by providing an additional layer of protection and enabling the use of a custom firewall.

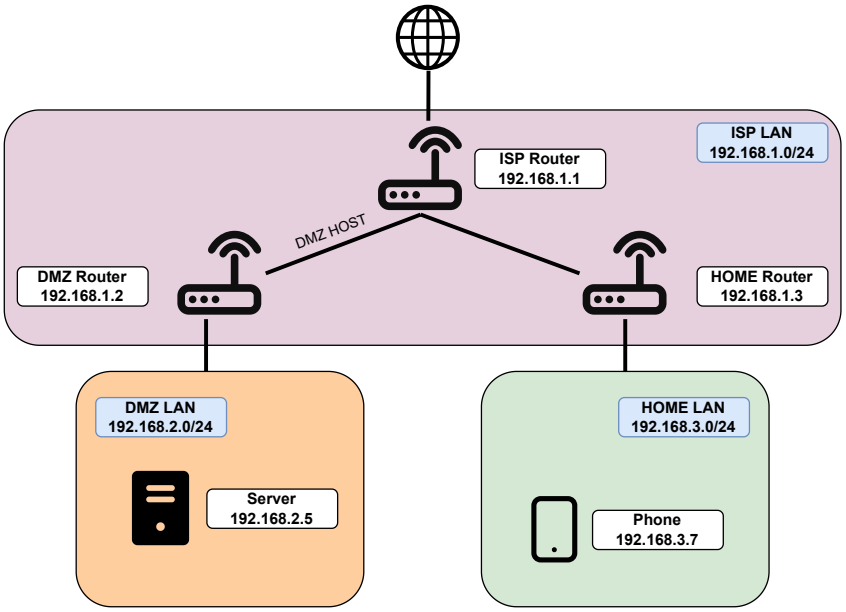


Figure 6: DMZ Host configuration

The pro of this configuration is that it is plug and play; you only need to connect the WAN port of the DMZ and home routers to a LAN port of the ISP router and configure a subnet, and the NAT will by default isolate each LAN. The con is that the ISP router plays a role in the network, and as we cannot always modify it as we want, it can be difficult to ensure a good isolation. This is true only if the DMZ option is available on your ISP router, the feature is not

present in all routers, but it is commonly available on many mid-to-high-end consumer routers (Netgear [40], Cisco [14], TP-Link [65]).

For this project, I decided to use the second approach because it is the easiest to reproduce. I believe that ease of deployment is best, even if we lose some control due to the ISP router.

4.4 Network topology

The network topology used in the experiments is shown in figure 7. The dashed lines represent the IEEE802.15.4 communication over the air between the devices. I decide to use them to represent the full mesh network that is created by the Thread devices. But as the communication is over the air, everyone near the devices can listen to the communication, like the sniffer. The sniffer is completely separated from the network because it is only used to listen to the communication over the air and does not have to be known by the network. The solid lines represent the IP communication in the home network, like Wi-Fi, Ethernet, etc. For the ESPs, I decided to represent them with a light bulb as I flash a Matter smart bulb on them.

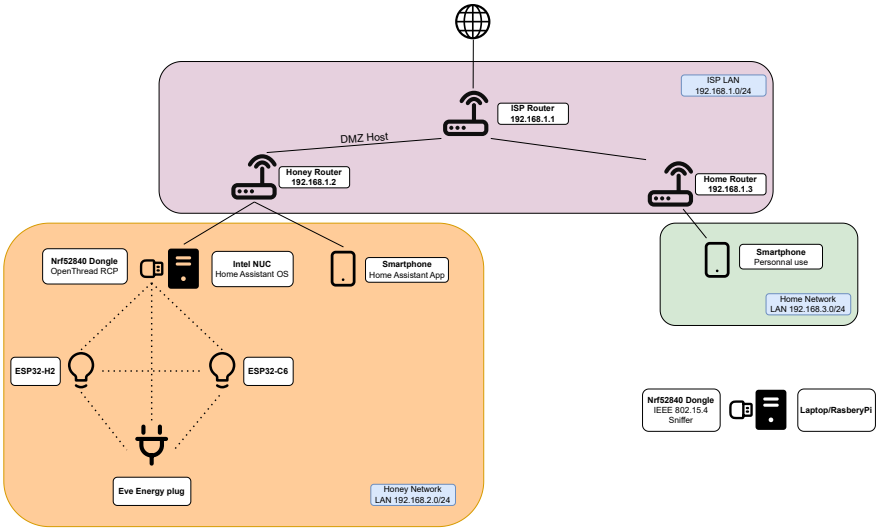


Figure 7: The network topology used in the experiments

4.5 Hardware

4.5.1 IoT devices

As mentioned earlier, I used ESP devices to emulate Matter devices. I used an **ESP32-H2** [18] and an **ESP32-C6** [17] on which I can flash the type of device

I want. I also used an **Eve Energy** [21] smart plug and power meter, which is a real Matter over Thread device.

4.5.2 Intel NUC

For Home Assistant, I have at my disposal an **Intel NUC 8i5BE** [31]. I installed the Ubuntu server OS 24.04 on it. As I will connect to the NUC remotely, I did not have to install an OS with a GUI. I chose Ubuntu because it is an easy-to-use OS, and I am familiar with it. For the virtual machine with Home Assistant OS, I used VirtualBox [46], which is a free and open-source virtualisation software.

4.5.3 Radio co-processor

To be able to use the Thread Border Router add-on, I needed a radio co-processor. The radio co-processor is a device that will allow the NUC to communicate on the Thread network. I used a **Nordic Semiconductor nRF52840** [43] dongle, which is a USB dongle that supports IEEE 802.15.4 and Thread. To be able to use the dongle, I had to install the firmware on it. The firmware and the instructions to install it can be found on the OpenThread GitHub repository [47].

For the sniffer, I also needed a radio co-processor. I chose the same dongle as the Thread Border Router, but I had to install a different firmware on it. The instructions should be found on the Nordic Semiconductor website [41].

4.5.4 Router

As explained in the DMZ section, I use two routers in addition to the ISP one. The two routers are Pegatron routers that run under OpenWrt [45]. OpenWrt is a Linux operating system for routers. I flashed a Raptor GUI (the default for Pegatron routers) on them.

4.5.5 Summary

The figure 8 summarises the hardware used in the experiments.

Device	Purpose
ESP32-H2[18]	Matter over Thread device
ESP32-C6[17]	Matter over Thread device
Eve Energy[21]	Matter over Thread real device
Intel NUC 8i5BE[31]	Home Assistant controller
2x Nordic Semiconductor nRF52840[43]	Radio co-processor for HA + Sniffer
2x Pegatron Router	Network isolation

Figure 8: Summary of hardware used in the experiments.

5 Sniffer

The sniffer is a Python package I developed for capturing and analysing a Matter over Thread network. The code of the honeypot can be founded on my Github : Matter-Thread_Sniffer. The tool's objectives are to capture packets and analyse them to find out what an attacker could do with them. The idea is to discover which kind of information is available in packets and if it is possible to extract sensitive information. I also developed an analyser to extract relevant information from captured packets and expose it to the user. For ease of use, the sniffer is fully usable inside a Linux terminal.

To be able to capture packets, the computer must be able to listen to IEEE802.15.4 traffic. To do so, I used an NRF52840 Dongle [43] (as explained in section 4.3.3). The dongle should be flashed using the nRF Sniffer for 802.15.4 tool developed by Nordic Semiconductor [41]. Nordic [54] also provides a Python module that is used by the sniffer [42]. The module allows us to use a Python command to launch and configure the traffic capture.

5.1 Design

The sniffer package is divided into two main modules, **the sniffer** and **the analyser**.

5.1.1 Sniffer module

The sniffer module is a command line interface (CLI), and its main objective is to allow users to capture packets and save them into a PCAP file. The command line interface is built with the Python *click*[62] library. The library, is used to create command line interface in a composable way with as little code as possible. The CLI is the entry point of the package and implements multiple useful commands that I described below.

***help* command** The help command is used to display all the available commands of the sniffer module. Each command listed below also has its own help option, which explains its purpose and how to use it.

***channel-finder* command** As explained earlier in the IEEE802.15.4 background section 2.1, the IEEE 802.15.4 standard uses channels to communicate. All the devices of the network should use the same, but how can we know on which channel to listen? To solve this problem, I implemented the channel founder command. The command will start listening on each channel for 10 seconds. Then, it will read the captured PCAP file and compute the number of packets in it. If the number of packets is greater than a threshold, for example, 2, it will stop and return to the user the channel on which it captures more than the threshold packets. The purpose of the threshold is to limit the number of false positives. In contrast, the threshold cannot be too big to avoid false

negatives. For the correctness of these approach, we have to suppose that only one channel is used. As we are interested in Matter over Thread and a Thread personal area network only used one channel, it's unlikely that several Thread networks will be created in the same smart home. However, if another network protocol that uses IEEE 802.15.4 is present, such as Zigbee [68], it is possible that the channel returned will not be used by the thread network but by Zigbee.

To counter this problem, I had to improve the channel-finder command. Before stopping when receiving multiple packets on a channel, the program will verify if the captured packets are not Zigbee packets. I cannot only verify the first packet because IEEE802.15.4 ACK packets for Thread and for Zigbee are similar, so I have to select the protocol used on data packets.

To implement this, I had to capture some Zigbee traffic to be able to differentiate it from the Thread traffic. To do so, I connected a Phillips Hue bridge [7] and a Phillips Hue bulb [1] to the home assistant configuration. As the bridge uses Zigbee to communicate with the bulb, I simply have to sniff traffic with the capture command described below. Once it was done, I was able to differentiate Zigbee and Thread packets; I simply see that Pyshark [50] was able to recognise Zigbee traffic. Pyshark is a Python wrapper for Tshark, allowing Python packet parsing using Wireshark dissectors.

***capture* command** The capture command is used to start capturing packets. The command uses the module provided by Nordic to start the capture. After X seconds (this value can be set in the command arguments), it stops the capture and saves the packets in a PCAP file. Pyshark [50] is used to extract data from the PCAP file.

***analyser-cli* command** The analyser-cli command is the entry point of the analyser described below.

5.1.2 Analyser module

The analyser module will help the user infer information from a captured PCAP file. As the technology is still new and not well known yet, an analyser will help to collect and analyse important data and present it in a simple way. The analyser is a Python script that starts by analysing the provided PCAP file and then lets the user enter commands to show desired data.

When launching the analyser, it starts by analysing all the PCAP files, so it may take some time. When everything is loaded, the user can interact to retrieve relevant information. All the analysis is shown to the user via a table in the terminal. To simplify the creation and make a readable ASCII table, the Python module PrettyTable [48] is used.

There are all the available commands of the analyser:

***help* command** The help command is used to display to the user all the available commands and their arguments if any.

***nbr_packets* command** The nbr_packets command returns the number of packets in the PCAP file. It also informs the user on the number of Thread packets and the number of MLE packets.

***channel* command** The channel command prints on which IEEE802.15.4 channel the capture was done.

***exit* command** The exit command is used to exit the analyser CLI and close the PCAP file reader correctly.

***communications* command** The communication command displays every communication from a source to a destination (or multiple destinations in the case of a broadcast). This information is represented in a table in the following format:

Communication type	Address IEEE802.15.4. A	Address IEEE802.15.4. B	#Packets A->B	#Packets B->A

- **Communication type** can be the following :
  - *MLE_Data_Broadcast_IPv6*: MLE packets that are broadcast using IPv6.
  - *MLE_Data_IPv6*: MLE packets that are sent to a specific IPv6 destination.
  - *Thread_Data*: Thread packets sent from one source to one destination.
- **Address IEEE802.15.4. A/B** are the source and the destination address of the IEEE802.15.4 layers. The destination of a Broadcast message is *0xFFFF*.
- **#Packets A (B)->B (A)** represent the number of packets from the source address A to B and the number of packets from the source address B to A.

You can also add an argument when calling the communication command. The argument must be a number within $[0, 4]$. This number represents the column that will be sorted in increasing order. The default value is 0. This option can be useful for facilitating the reading of the table when focusing on a specific column.

***sources* command** The command will show each different IEEE802.15.4 source address. This information is represented in a table in the following format:

IEEE802.15.4 Address	Address type	PAN	Mean RSS (dBm)	Mean LQI	#Packets sent

- **IEEE802.15.4 Address** is the IEEE802.15.4 source address of the packet
- **Address type** is either *short_address* if it is an RLOC16 source address or *long_address* if it is an IPv6 address.
- **PAN** is the Personal Area Network in which the packet was sent.
- **Mean RSS (dBm)** The RSS is the Received Signal Strength measured in dBm (decibels milliwatts). This value represents the average strength at which the sniffer reads packets from this source.
- **Mean LQI** The LQI is the Link Quality Indicator, it is another metric used to measure the quality of a wireless communication. As for RSS this value is the mean of LQI measure by the sniffer for the current source.
- **#Packets sent** Is the number of packets send by this source.

As for the communications command we can add the index of the column by which we want to sort the table.

***devices* command** This command is used to link the *short_address* and the *long_address*. As explained earlier, the short address is the Thread RLOC16 and may change over time if the topology changes. In contrast, the long address is unique and never changes. The issue is that most communications are done using the short address, while only MLE packets use the long address in clear text. The easiest way to link them is to decrypt the Thread packet with the Thread Master Key and identify the RLOC in the decrypted part of the MLE packet. The problem is that when sniffing, we are not supposed to know the Thread key, so we have to extrapolate.

To extrapolate which short address is linked with which long address, I decided to use the LQI. As our sniffer will not move during the capture, each packet that we sniff from a device should approximately have the same Link Quality Indicator. As we know that the long address is unique, we can, for each long address, suppose that it is a device. For each RLOC16 source, we have to link them with the devices we define based on the long address. To do so, each RLOC source will compare its mean LQI with the one of the devices and link them with the closest one. As it is an extrapolation, we cannot guarantee the result, but on the network explained in the previous section, the result was always correct.

The result is presented in a table in the following format:

Short IEEE 802.15.4 address	Long IEEE 802.15.4 address	IPv6	LQI delta	#Packets

- **Short IEEE 802.15.4 address**
- **Long IEEE 802.15.4 address**
- **IPv6** The source IPv6 of the MLE packet.
- **LQI delta** The difference between the LQI of the device and the RLOC source linked to it.
- **#Packets** Total number of packets of this device.

5.2 Usage example

In this section, I will show how to use the sniffer and which kind of data an attacker should infer from capturing IoT communication. The example presented below was done in the network environment described in the figure 7. The sniffer was run during the commissioning of the EVE smart plug [21]. I chose to present this scenario because the commissioning is most of the time the most sensible part, prone to bugs and failures. The commissioning is also a good way to capture a lot of packets in a short time period.

The two ESPs were linked to the Intel NUC for power supply, and the EVE plug was 2 meters apart, connected to a wall socket. The NRF52840 used to sniff the IEEE802.15.4 was plugged into a laptop between the NUC and the EVE plug. I will present you each step and the result of it.

The first step is to find out which IEEE802.15.4 channel Thread is communicating on. To do so, I use the channel-finder command with the path to the NRF52840 in the arguments.

```
    sniff channel-finder --dev /dev/ttyACM0 --stop False
```

After several seconds, the command returns *15*. Now that I know the channel, I can start a capture of 90 seconds on channel 15 and save the result in a PCAP file in the logs directory.

```
sniff capture --dev /dev/ttyACM0 --duration 90 --channel 15 logs/
    EVE_commissioning.pcap
```

Now that the capture is finished, we can enter into the analyser command line interface. To do this, I used the analyser-cli command:

```
sniff analyser-cli logs/EVE_commissioning.pcap
```

Once in the analyser, I will show all the available commands and discuss their output.

number_packets This command shows us that Thread packets outnumber MLE packets. It is explained by the fact that Thread handles regular data communication, while MLE packets are used for network management.

```
> nbr_packets
+--------------------+----------------+-------------+
| Number of packets | Thread packets | MLE packets |
+--------------------+----------------+-------------+
|        2854        |      2826      |      28     |
+--------------------+----------------+-------------+
```

Figure 9: Sniffer: nbr_packets

channel The channel command simply returns 15, as it is the channel on which we capture data. The command can be useful if analysing multiple PCAPs with different channels to remember from which it was captured.

```
> channel
Channel: 15
```

Figure 10: Sniffer: channel

communications Here we can see for each IEEE802.15.4 source address, the number of packets. As explained in the section 2.1, the Thread data packets use the RLOC16 of the node as the source and destination address. This functionality aims to reduce the size of the packets.

The MLE broadcast packets contain information on the topology of the network. This information is spread to all the Thread nodes to let them know network information such as the RLOC of their neighbours, the IPv6 address of the border router, etc.

The MLE packets that are directly sent to an IPv6 address are packets containing the node's specific information.

```
> communications
+-------------------------+-------------------------+-------------------------+--------------+--------------+
|    Communication type   |  Address IEEE802.15.4 A |  Address IEEE802.15.4 B | #Packets A->B | #Packets B->A |
+-------------------------+-------------------------+-------------------------+--------------+--------------+
| MLE_Data_Broadcast_IPv6 | 4e:15:8c:de:70:76:99:77 |          0xffff         |       8       |       0       |
| MLE_Data_Broadcast_IPv6 | 6e:5f:41:81:42:ad:b0:1b |          0xffff         |       8       |       0       |
| MLE_Data_Broadcast_IPv6 | a2:09:d9:f1:08:56:c3:fe |          0xffff         |       2       |       0       |
|      MLE_Data_IPv6      | 4e:15:8c:de:70:76:99:77 | a2:09:d9:f1:08:56:c3:fe |       6       |       1       |
|      MLE_Data_IPv6      | 6e:5f:41:81:42:ad:b0:1b | a2:09:d9:f1:08:56:c3:fe |       4       |       1       |
|       Thread_Data       |          0x5c01         |          0x5c00         |      563      |      142      |
|       Thread_Data       |          0x7c00         |          0x5c00         |      143      |      567      |
+-------------------------+-------------------------+-------------------------+--------------+--------------+
```

Figure 11: Sniffer: communications

sources This command shows us each IEEE802.15.4 source that sends Thread packets on the network. We can see that there are 3 short addresses and 3 long addresses, as IEEE802.15.4 uses long addresses in certain cases, as explained

in the section 2.1. We can also see that the PAN ID of the last line was not the same as the previous one. I suppose that when this device uses its long addresses (only 4 packets), it is when it did not yet know the PAN ID.

```
> sources
+--------------------------+---------------+--------+---------------+----------+---------------+
|   IEEE802.15.4 Address   |  Address type |  PAN   | Mean RSS(dBm) | Mean LQI | #Packets sent |
+--------------------------+---------------+--------+---------------+----------+---------------+
|          0x5c00          | short_address | 0x44b9 |     -35.16    |  229.24  |      709      |
|          0x5c01          | short_address | 0x44b9 |     -44.21    |  193.72  |      563      |
|          0x7c00          | short_address | 0x44b9 |     -39.01    |  216.06  |      143      |
| 4e:15:8c:de:70:76:99:77  |  long_address | 0x44b9 |     -38.71    |  217.43  |       14      |
| 6e:5f:41:81:42:ad:b0:1b  |  long_address | 0x44b9 |     -35.1     |  230.0   |       10      |
| a2:09:d9:f1:08:56:c3:fe  |  long_address | 0xffff |     -45.0     |  194.0   |       4       |
+--------------------------+---------------+--------+---------------+----------+---------------+
```

Figure 12: Sniffer: sources

devices The devices command, as explained earlier in the section 5.1.2, is an extrapolation to estimate which short addresses are the same device as which long addresses. To verify the output, I used Wireshark [67]. In Wireshark, we can add a Thread master key to decrypt packets. Once decrypting packets, I was able to see in the MLE packets which short address was announced for which long address. With that, I could verify that the extrapolation in this scenario was the good one.

```
> devices
/!\ MLE packets and Thread packet are used to extrapolate the result of this table. The result may be false
+---------------------------+---------------------------+----------------------+-----------------------------------------------+----------+
| Short IEEE 802.15.4 address | Long IEEE 802.15.4 address |         IPv6         | LQI delta between IEEE802.15.4 and MLE | #Packets |
+---------------------------+---------------------------+----------------------+-----------------------------------------------+----------+
|          0x5c00           |  6e:5f:41:81:42:ad:b0:1b  | fe80::6c5f:4181:42ad:b01b |                0.06                |   719    |
|          0x5c01           |  a2:09:d9:f1:08:56:c3:fe  |  fe80::a009:d9f1:856:c3fe |                0.79                |   567    |
|          0x7c00           |  4e:15:8c:de:70:76:99:77  | fe80::4c15:8cde:7076:9977 |                0.3                 |   157    |
+---------------------------+---------------------------+----------------------+-----------------------------------------------+----------+
```

Figure 13: Sniffer: devices

5.2.1 Discussion

With such a sniffer, attackers should be able to determine the IEEE802.15.4 addresses used in the network. They can know which address communicates more, and so they can make predictions on the type of device, i.e., a smart bulb will usually send fewer packets than a thermometer. As the long address is unique (in opposition to the short one), they can determine the number of IoT devices in the network. With the devices command, they can also link an RLOC to its long address. All of that can be known without having any device on the network, by just capturing IEEE.802.15.4 packets. The nRF52840 used in the experiment is not for long distance, but with a bigger antenna, the sniffer can be performed from outside the smart home.

All of that is not due to a misconfiguration or Matter/Thread problem; it cannot be avoided without losing all the performance.

6 Honeypot

In this section, I will discuss the honeypot I deployed in the network described in the section 4. The code of the honeypot can be founded on my Github : Matter-Thread_Honeypot. My honeypot is a high-interaction research honeypot as defined in section 2.4. It is a research honeypot, as it is in a false network, and its goal is to know more about attacker behaviour in such an environment. The high interaction is as the network is a real Matter over Thread network, it will fully interact with attackers; it is not a simulation; all of that will be explained in the following sections. I'll start by explaining the objectives I've chosen for my honeypot. Then, I will present the different parts of the honeypot.

6.1 Objectives

The first objective is to see what an attacker will try to do on a Matter/Thread IoT network. As the protocol is still new and wants to be usable by everyone, a poorly configured network may occur. For this reason, I decide to simulate a poorly configured IoT network. The objective of the network is to be as close as possible to a real network while adding a few bad configurations. This network will be like a Matter/Thread network that a lambda user may deploy for personal purposes. As a lambda user doesn't necessarily have any knowledge of IT, let alone security, such an installation can be misconfigured (i.e. open SSH port, poor password, etc.).

The second objective is to be aware if an attacker can be interested in Matter/Thread or know these technologies. To know that, I decide to use an SSH honeypot with some files (i.e. honey tokens) that indicate that there are Matter devices on the network. With such files, we can conclude, based on the input command of the attacker, if such a file may interest an attacker.

To retrieve and analyse all the data to accomplish the objectives, I need to monitor the system. To monitor it, I implemented two layers of monitoring as explain in section 6.4.

6.2 NUC organisation

Figure 14 summarises what is running on the NUC from the honeypot point of view. Cowrie listens on port 2222 for SSH connections, but as the default port for SSH is 22, I decided to add a rule in the firewall to redirect all the packets coming on port 22 to port 2222. The firewall is also used to authorise real SSH connections (moved to port 8228) from the LAN network. The Flask server and the MySQL server are also running on the NUC, as it was the easiest in this project, but for full isolation, we can move them to a remote server.

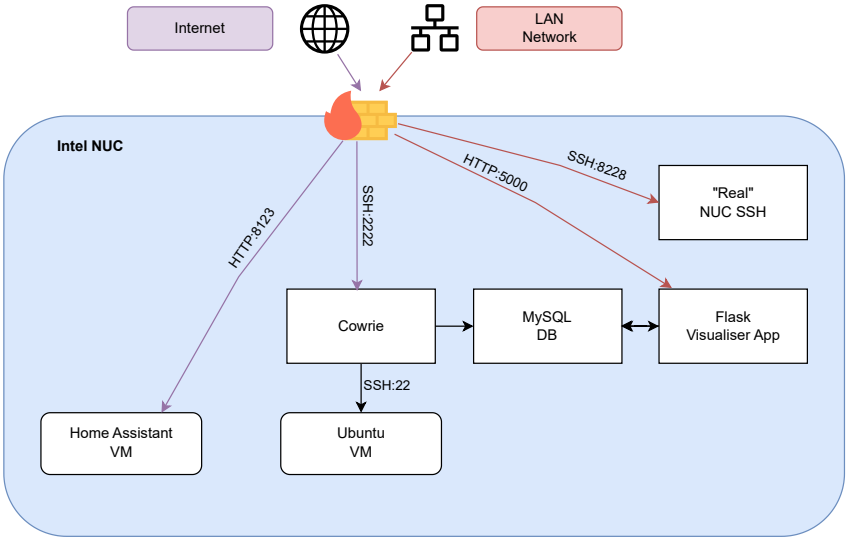


Figure 14: Intel NUC organisation

6.3 Cowrie

Cowrie [11] is an open source medium to high interaction SSH and telnet honeypot designed to log shell interaction performs by attackers. In this work, I only used the SSH part.

The medium interaction mode is a simulation of a UNIX system using Python. The advantage of such an approach, as explained in the subsection 2.4 is that the system is fully simulated, so attackers cannot do what they want, but in return, they can easily detect that they are in a honeypot. The second mode, the high interaction mode, is a SSH proxy. A proxy is a system that acts as an intermediary between a client and a server. In this context, Cowrie listens for SSH communication to the machine and redirects it to another system. The main advantage is that attackers can do what they want in the system, as it is a fully operating system. Of course, I had to limitate some actions such as the internet connection. It should be disabled to prevent attackers, for example, from using the system to attack other servers and be part of a DDoS attack.

Cowrie is a process that runs directly on the Intel NUC of my network. The process simply listens for traffic on port 2222; all the traffic is logged and saved in a MySQL database and then sent locally to a Ubuntu VM. For an attacker, everything looks like a normal SSH connection to a server.

For the authentication, Cowrie lets us define all the authorised login password pairs. As I want the access to the server to be relatively trivial, I decided to accept every connection where the login is *admin*, no matter what the password is.

I decided to redirect the SSH traffic to a VM running Ubuntu Server. The VM is in host-only network mode to prevent it from having access to the local network or the internet. As the purpose of the SSH connection is to give information to the attacker on the context of the network and information on it, I hide multiple pieces of information in the file system of the VM.

```
/home/admin/
├── Downloads/
│   ├── connectedhomeip/ (Github repo)
│   ├── openThread/ (Github repo)
│   └── matter-dissector/ (Github repo + contains Thread Master
│       Key)
├── Config/
│   ├── firewall/
│   │   └── nft.conf (contains port of the HAVM)
│   └── homeAssistant/
│       ├── HAsetup.md (contains credential of HA, owned by sudo)
│       └── docker-compose.yml
└── Logs/
    ├── SnifferPCAP1
    ├── SnifferPCAP2
    ├── SnifferPCAP3
    └── SnifferPCAP4
```

Figure 15: File system tree of the Ubuntu server VM

Figure 15 represents the file system of the VM. I will discuss some files in the following points:

- **matter-dissector** is a Wireshark plugin to display Matter packets correctly. As it needs credentials to decrypt Matter/Thread packets, I hide in a config file the Thread master key of the network.
- **nft.conf** is a false Nftables firewall configuration file. It contains the port number of the HomeAssistant web GUI.
- **HAsetup.md** is a false guide on how to deploy HomeAssistant. I decided to hide in it the credentials needed to access the WEB GUI. As this information is quite sensible, I make the file readable only via the sudo command. To be able to read it, the attacker should find the password.
- **SnifferPCAP** are PCAP files I capture on the network using the sniffer described in the section 5.

All the other files and/or repositories are just there to fill the VM and to not be suspect. Attackers that see an empty server where the access is trivial can suspect that they are in a honeypot.

Cowrie can also use **Discord webhook**. Discord[13] is a messaging app. This functionality will send information about a new SSH connection to the API of Discord. Then, the message will be sent by a BOT in a Discord server. With that, we can know when someone enters the honeypot and a summary of relevant information such as the IP, the duration of the sessions, username, ...

6.4 Monitoring

The monitoring system consists of two parts. The purpose of each is to collect and log all relevant data for subsequent analysis. These two parts are:

- **SSH monitoring** using Cowrie section 6.4.1
- **Home Assistant monitoring** section 6.4.2

6.4.1 Cowrie monitoring

I did not implement something for the monitoring of Cowrie, as it is already done by Cowrie, but I will explain which relevant data is logged. By default, Cowrie logs everything in log files, but in the configuration file of the honeypot, we can decide to send all the logs to a third-party app. In my case, I decided to save all this data in a MySQL database.

When someone tries to connect using SSH to the honeypot, Cowrie will log a new **session** with the starting time and the IP. Once the session is stopped, it will add an end time. After the session has started, attackers should introduce credentials to achieve the **authentication** to the false server on the VM. The username and the password entered by an attacker are stored with a boolean to indicate if the authentication succeeded or not and the timestamp of it. Then, if attackers succeed in connecting to the server, all the **inputs** they enter will be stored.

6.4.2 Home Assistant monitoring

This monitoring part captures relevant data on TCP packets that come from the internet to the Home Assistant VM. To do so, I had two possibilities:

- Log from inside the Home Assistant VM
- Log from outside by capturing network traffic

I opted for the second solution. The reason for this is that attackers can enter Home Assistant and take possession of the system. If this is the case, they may decide to interrupt the data capture or send false information. To overcome this problem, I decided to implement a Python script that would analyse the packets sent over the NUC to Home Assistant and then save the important information.

To implement this, I used Scapy [53]. Scapy is a powerful interactive packet manipulation library for Python. The script will launch two threads. The first

thread will use the sniff command of Scapy to capture all TCP traffic to the Home Assistant VM. It will save the IP and the timestamp of each packet in a temporary CSV file. The second thread will read the CSV file and send it to the MySQL database.

6.5 Dashboard

Now that the data collection is achieved, I need a way to make them human-readable. In order to achieve this objective, I developed a monitoring web server. The objective of the server is to present the data in the clearest way possible. To implement a web server I use Flask [63]. Flask is a web application framework for Python. I decided to use it, as I am familiar with it, and it is quite easy to use.

Sessions table page The first page is the session table page, as shown in figure 16. On this page, we can see each session sorted by their starting time. For each session we can see their starting and ending time and their ID and IP source address. The session ID, i.e. the random unique key of the MySQL database, is clickable and redirects to the specific session page.

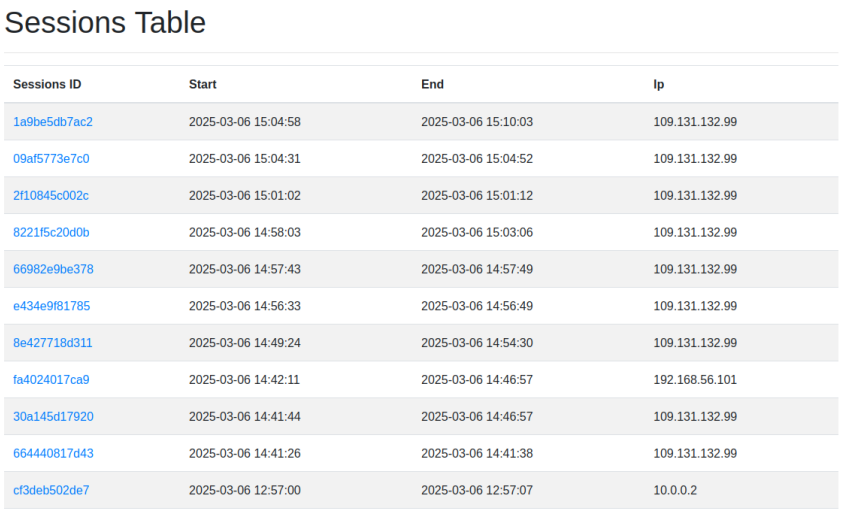
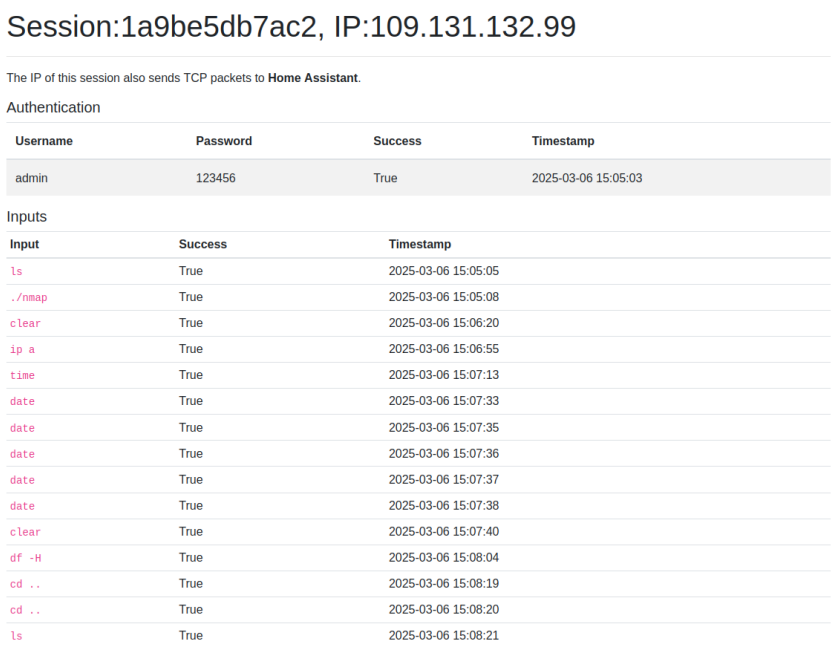


Figure 16: Monitoring server: session table page

Specific session page The specific session page as shown in figure 17 shows all relevant information about a session. Firstly, if the IP of the session also sends packets to Home Assistant, it will be indicated. Then, an authentication table will list all the usernames and passwords used to connect to the server; it will also show if the authentication succeeded and its timestamp. The last part of the page is the list of the inputs entered by the attacker. The list is sorted by ascending time. Representing the list of inputs is mandatory, as we can see what an attacker tried to do on the VM, and if the whole system shutdown, we can understand why and reproduce it if needed.



Session:1a9be5db7ac2, IP:109.131.132.99

The IP of this session also sends TCP packets to **Home Assistant**.

Authentication

Username	Password	Success	Timestamp
admin	123456	True	2025-03-06 15:05:03

Inputs

Input	Success	Timestamp
ls	True	2025-03-06 15:05:05
./nmap	True	2025-03-06 15:05:08
clear	True	2025-03-06 15:06:20
ip a	True	2025-03-06 15:06:55
time	True	2025-03-06 15:07:13
date	True	2025-03-06 15:07:33
date	True	2025-03-06 15:07:35
date	True	2025-03-06 15:07:36
date	True	2025-03-06 15:07:37
date	True	2025-03-06 15:07:38
clear	True	2025-03-06 15:07:40
df -H	True	2025-03-06 15:08:04
cd ..	True	2025-03-06 15:08:19
cd ..	True	2025-03-06 15:08:20
ls	True	2025-03-06 15:08:21

Figure 17: Monitoring server: specific session page

Home Assistant live tracker page This page displays a graph of the number of TCP packets received over time during a lapse of time for each IP. By default, it plots the number of TCP packets in the last 15 minutes, as shown in figure 18, but we can decide to plot in the last 60 minutes, 12 hours, day and week. This graph will let us see if an IP sends a lot of packets to the HomeAssistant server or even try a DoS attack.

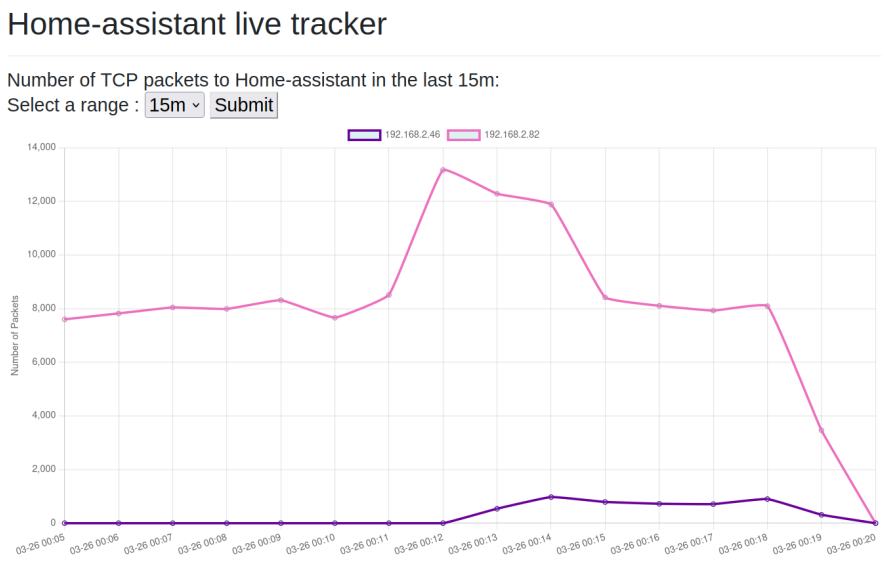


Figure 18: Monitoring server: Home Assistant live tracker page

7 Validation

The goal of the validation process is to prove that the sniffer and the honeypot work as intended. For the validation, I captured 3 scenarios representing what attackers could do in the honeypot and then analysed whether the honeypot was able to catch them. I chose these three scenarios to show 3 different types of attacks that the honeypot can catch. The first scenarios is the *Basic behaviour* where I show how an attacker can explore the honeypot and retrieve all the sensitive information. The second scenario is the *DoS attack*, in this scenario, I show how the honeypot can capture a DoS attack and the impact on the CPU. The third attack, is the *Command flooding*, in this attack I flood the IoT network with commands. Such an attack can make the devices unusable and/or if they are on battery, it will reduce their running time, in this case we can say that it is a battery depletion attack.

7.1 Scenario I : Basic behaviour

The first scenario is the "normal" attack path, in my opinion it is the easy way to get all the system vulnerabilities, attackers only need to connect via SSH and then explore files one by one to be able to interact with the Matter over Thread devices. I will show you step by step what the fake attacker does and then show you what the honeypot captures and what we can get and understand from it.

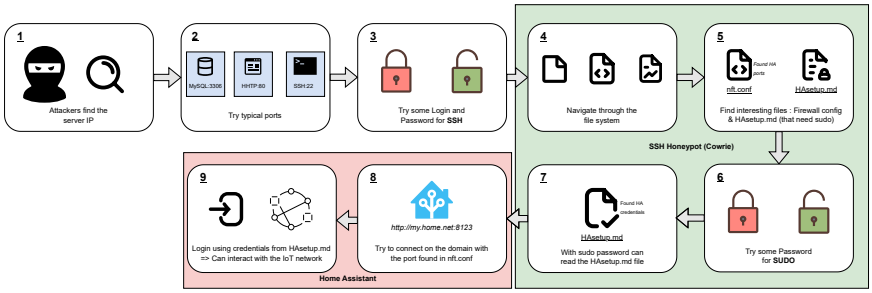


Figure 19: Scenario I : Attackers actions

Figure 19 shows the actions perform by the attackers for the first scenario. Each steps is detailed below :

1. Attackers find the server through its domain name or directly through its public IP.
2. Attackers try some basic ports on the IP address to see if anything is running. They try port 80 for HTTP web servers, 443 for HTTPS, 21 for FTP (File Transfer Protocol), 3306 which is commonly used for MySQL databases, and so on. Then they try to connect to the server using SSH on its default port, 22.

3. To connect via SSH, they need to provide a login and password. A basic login could be *admin* with the password 1234; with these credentials they are able to connect to the server (i.e. the honeypot) via SSH.

4. Once connected, they navigate through the file system to see what they can find on the server. To do this, they use Linux commands (e.g. cd, ls, cat, ...).

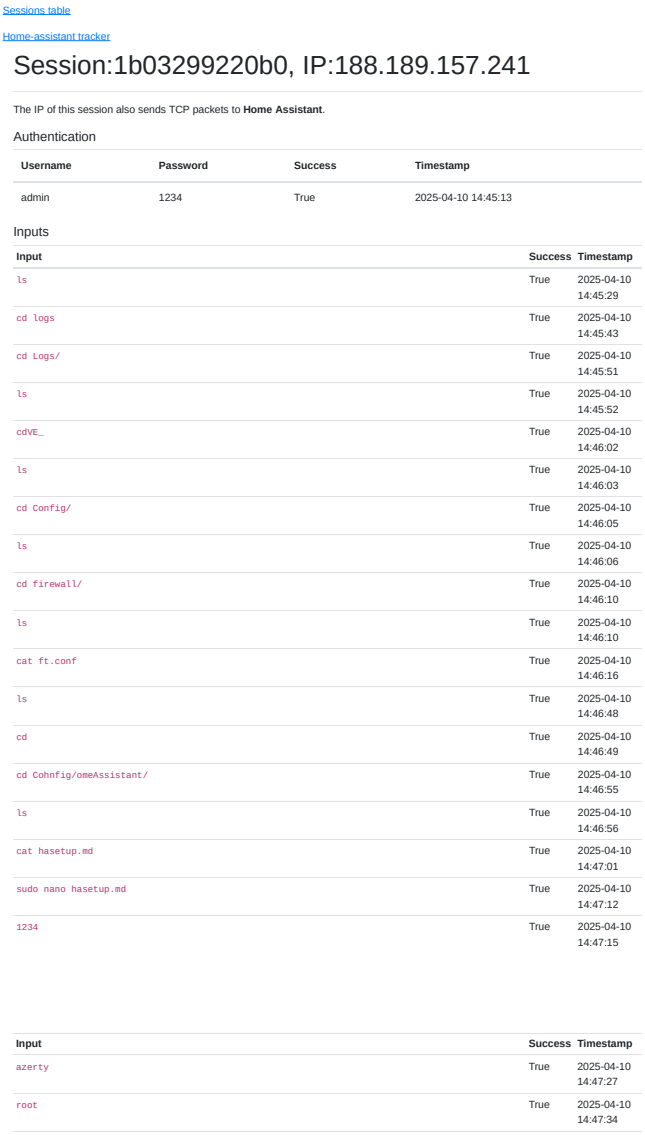
5. They are able to find several interesting things, such as a firewall configuration that lists ports which might be used by services (such as Home Assistant on port 8123). They also find a *HAsetup.md* file that requires sudo privileges to open.

6. To get sudo access, they try several passwords, such as the 1234 of the SSH connection. That didn't work, so they try a second one, azerty, and then a third one, root, which is the good one.

7. With sudo privileges, they can open the *HAsetup.md* and when they read it, they find a pair of credentials.

8. Now that they know the port of a Home Assistant instance and a pair of credentials, they simply try to connect to that specific port on the server domain name, for example : *http://myDomain.com:8123*. They see the login page of a Home Assistant web GUI.

9. Once on the GUI, they enter the credentials they found earlier and are now able to connect to the home assistant and interact with any Matter over Thread IoT devices connected to it.



Sessions table

Home-assistant tracker

Session:1b03299220b0, IP:188.189.157.241

The IP of this session also sends TCP packets to **Home Assistant**.

Authentication

Username	Password	Success	Timestamp
admin	1234	True	2025-04-10 14:45:13

Inputs

Input	Success	Timestamp
ls	True	2025-04-10 14:45:29
cd logs	True	2025-04-10 14:45:43
cd Logs/	True	2025-04-10 14:45:51
ls	True	2025-04-10 14:45:52
cdVE_	True	2025-04-10 14:46:02
ls	True	2025-04-10 14:46:03
cd Config/	True	2025-04-10 14:46:05
ls	True	2025-04-10 14:46:06
cd firewall/	True	2025-04-10 14:46:10
ls	True	2025-04-10 14:46:10
cat ft.conf	True	2025-04-10 14:46:16
ls	True	2025-04-10 14:46:48
cd	True	2025-04-10 14:46:49
cd Cohnfig/omeAssistant/	True	2025-04-10 14:46:55
ls	True	2025-04-10 14:46:56
cat hasetup.md	True	2025-04-10 14:47:01
sudo nano hasetup.md	True	2025-04-10 14:47:12
1234	True	2025-04-10 14:47:15

Input	Success	Timestamp
azerty	True	2025-04-10 14:47:27
root	True	2025-04-10 14:47:34

Figure 20: Scenario I : Basic behaviour list input from monitoring

The figure 20 is the specific session page explained in the section 6.5 of the basic scenario explained above. We can see that the honeypot is able to list all the commands entered by the attackers. It also gives us its public IP. Just below the title, we can also see that the same IP is sending TCP packets to the Home Assistant, indicating that the attackers are at least opening the Home Assistant's WEB GUI.

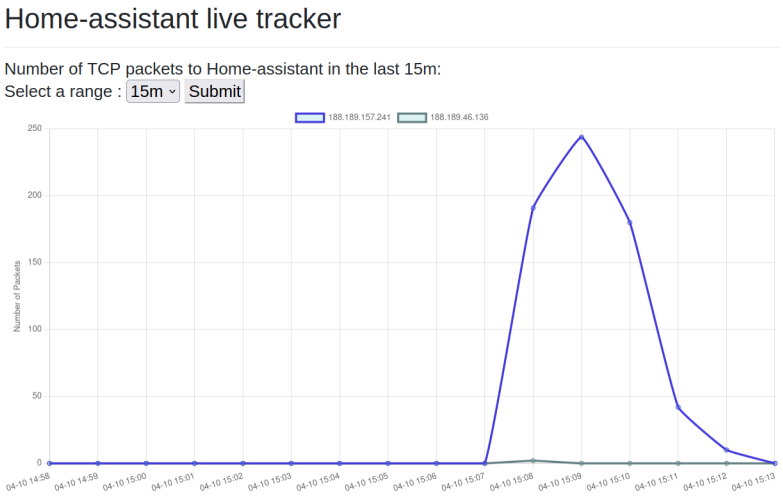


Figure 21: Scenario I : Basic behaviour TCP packets to Home Assistant

In the figure 21 we can see the number of TCP packets over time for the attackers' public IP. We can see when they start sending packets to the Home Assistant and when they stop.

7.2 Scenario II : DoS attack

A denial of service attack (DoS) is an attack that attempts to make the service unreachable by sending too many requests and overloading the system. Since the honeypot is focused on the Matter over Thread network, I will DoS the HomeAssistant service. If attackers are able to DoS the HomeAssistant, they can make the entire IoT network unreachable for legitimate users.

To simulate such an attack, I developed a Python script that uses the Requests [52] library. The Requests library is a simple library for playing around with HTTP. The script will send as many HTTP GET and POST messages as possible to the Home Assistant GUI. The script is a trivial DoS attack and is less powerful than a real DoS attack.

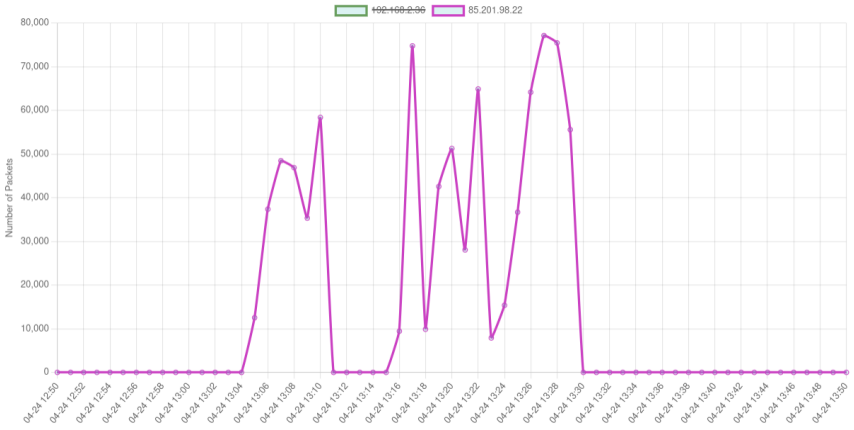


Figure 22: Scenario II : DoS attack number of TCP packets

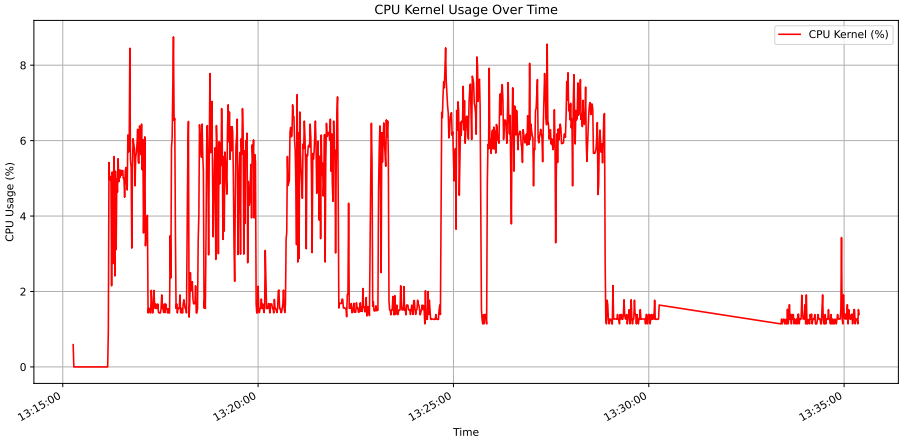


Figure 23: Scenario II : DoS attack CPU kernel usage

In figure 22 we can see that the number of TCP packets over time is huge, with a peak of 78,000 packets in one minute. With such a graph we can conclude that the attacker is performing a DoS attack. We can also imagine this for a Distributed Denial of Service (DDoS). In the opposite of a DoS attack, a DDoS attack come from multiple systems that coordinate to flood the target simultaneously making the attack stronger and harder to mitigate. For a DDoS, the graph will be roughly the same, not for just one IP, but for the IPs of all of the attacker's machines. To understand if such an attack has an impact on the HomeAssistant virtual machine, I also record the CPU usage of the VM's kernel, I only capture the kernel as it is in the kernel that the networking part is done. Figure 23 shows us that the impact of a DoS attack on the CPU load is

visible, but the impact is not large enough to make the HomeAssistant system unusable. In idle mode the load is about 2% and goes up to 8% during a DoS attack. I suspect that a real DDoS attack with multiple machines will have a greater impact and could probably make the system unreachable for its users.

Limitation When performing this scenario, I discovered a limitation. During the attack, I reached a bandwidth of 18.1 Mbps. As the network on which the attack was performed has an upload link of approximately 20 Mbps, it means that the attacker sends packets to the Home Assistant VM at its maximum speed. Another important factor is the internet connection of the honeypot network; here, I measured download speeds of 90.5 Mbps and upload speeds of 19.75 Mbps. These values can actually provide protection against DoS attacks as the internet connection limits the number of packets our network can receive. If the network gateway were more powerful, it would be possible to send more packets and have a larger impact, but in my network, a DoS attack that completely breaks the system does not seem possible.

7.3 Scenario III : Command flooding

In the previous scenarios, we had no way of knowing whether the attackers were using Home Assistant to send packets over the Matter over Thread network. We can only know the number of TCP packets sent to the Home Assistant server. To know if they are sending packets on the IoT network, we can use the sniffer described in section 5.

In this scenario, I present an attack where the attackers continuously send instructions to the IoT device, i.e. command flooding. Such an attack can have several effects, it can render the IoT devices unusable by continuously changing their state (i.e. open/closed). It can also force the IoT devices to stay awake and so, if they are battery powered devices, they may lose all power earlier than in a normal situation, this attack is called a battery depletion attack. The simplest way to perform this attack in my network is simply to have access to the home assistant once and continuously send instructions to devices, for example, always turn on/off a light.

To monitor this attack, I used the sniffer. I do a capture of 120 seconds on the IoT network, during this capture I do nothing on the network, it is the reference capture, so it is the number of packets sent to maintain the network and no instruction is sent to any particular device. Then I start a new capture of 120 seconds where I send a lot of instructions to each device. When the two captures are finished, I compare the bandwidth for each short IEEE802.15.4 address (i.e. RLOC address).

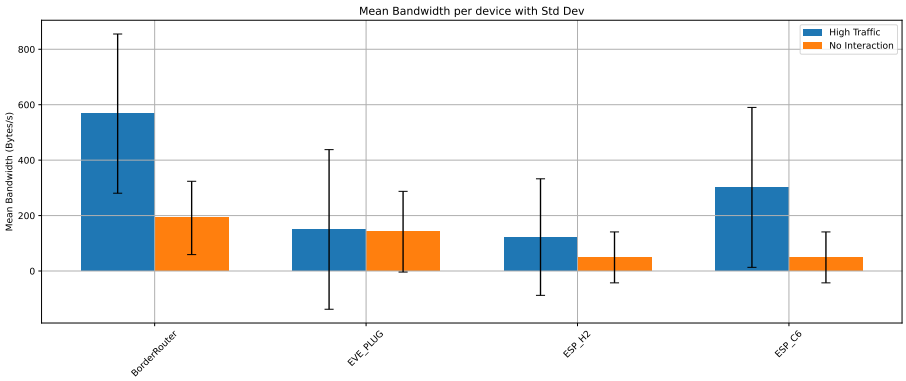


Figure 24: Scenario III : Comparison of bandwidth for high and no command traffic

Figure 24 shows what this type of attack can look like. We can see that there are the 4 devices, i.e., the two ESPs (ESPH2 and ESPC6), the border router and the EVE smart plug. We can also see that the node with the higher bandwidth, even in the high traffic scenario than in the no interaction scenario, is the border router node, which is normal as any message coming from outside the Thread network enters it via the border router. We can clearly see the attack, the capture where the attackers are flooding the IoT devices has a higher bandwidth than the other because more messages are circulating in the network. When we capture an activity like this, we can conclude that our IoT devices are being used more than in idle situation and that someone is interacting with them.

7.3.1 Battery depletion attack

As explain above, a command flooding attack can results on a battery depletion attack where the goal of the attacker is to drain the device's battery faster than in a normal scenario. I don't have at my disposal a Matter over Thread device that works on battery but I got ESP(I make the experience with the ESP-C6 [17] but it will be the same with others ESP) and a EVE smart plug [21].

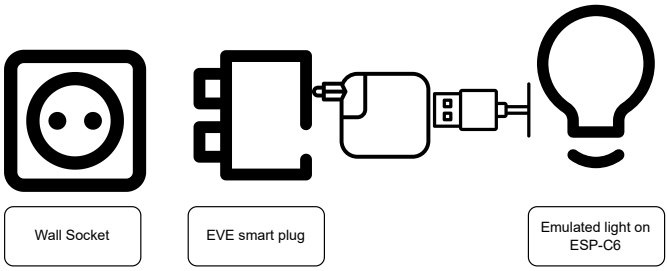


Figure 25: Scenario III : Depletion attack devices configuration

Figure 26 shows the configuration I used to measure the power usage of the ESP device. I simply used the EVE smart plug [21] , which can measure voltage, current, and power while also controlling power on/off. The ESP-C6 emulates a smart light that can be turned on/off. After connecting everything as shown in figure 26, I was able to measure the power consumption of the light.

I started the measurement with the device idle for 10 minutes, then used Home Assistant's history feature to download the power consumption values for this period. I downloaded a CSV file containing the values and timestamps when power consumption changed. With this data, I calculated the mean consumption and standard deviation. I then repeated scenario III, continuously sending on/off commands to the device for 10 minutes, and retrieved the data via Home Assistant.

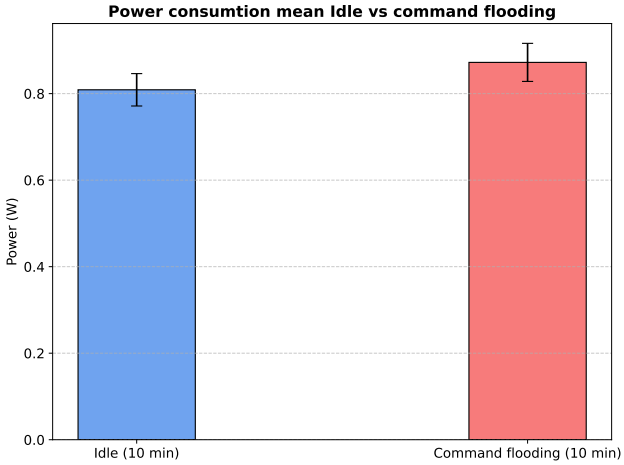


Figure 26: Scenario III : Comparison of power consumption for high and no command traffic

The results of the experiment are shown in Figure 25. We can see that the mean power consumption (in Watts) is greater during the attack than during idle time (with the light off, i.e., the ESP LED off).

Limitation The difference is not substantial and can be explained by multiple factors. Firstly, I collected data from an ESP device which is not a commercial Matter/Thread device. Additionally, the code running on the ESP is designed primarily for testing purposes rather than for real-world use. It is reasonable to assume that some energy optimisations are not enabled for such a device, and that a real device will likely add more functionalities, making the code heavier to run. We can also note that the power consumption of the ESP's LED is very small, so results with a real device that uses more power when turning on might be more significant (i.e. robot vacuums, smart lock, etc.). Another point is that measurements were taken using an EVE smart plug rather than precise laboratory measuring equipment; with such small values, the EVE smart plug is likely not designed for accurate measurement at this scale. Nevertheless, we can observe that there is an impact, even if it's not substantial in my case, it would still reduce the battery life of a battery-powered device. For the reasons explained above, we can also conclude that battery depletion would be even more substantial with a real commercial device.

8 Conclusion and Outlook

The Matter over Thread protocol aspires to become the most widely used IoT standard for smart homes devices. As with any widely adopted protocol, ensuring its security is critical. However, formally proving the security of such a complex standard remains a significant challenge.

In this master's thesis, I developed a honeypot and a packet sniffer for Matter over Thread networks. The honeypot is designed to emulate a realistic Matter over Thread IoT network including deliberate misconfigurations in order to attract potential attackers. By observing their behaviour, we can assess their familiarity with the protocol and identify possible vulnerabilities. The packet sniffer can be used to detect attacks at the IoT level or simply to retrieve data from surrounding devices.

The deployed network is composed of a genuine Matter over Thread device (the Eve Smart Plug [21]) and microcontrollers that emulate a smart bulb. To mitigate potential threats, the honeypot is isolated within a DMZ to prevent attackers from gaining access to the entire network. If attackers realise that they are in a honeypot, they may attempt to erase their traces from the logs. To prevent this, I have completely segregated the logs from the IoT network, ensuring that logs are transmitted outside the attackers' reach, thereby preserving all data collected during an attack. To leave some breach and misconfigurations, I included sensitive data within a SSH honeypot (cowrie [11]), where attackers can connect and retrieve it. To demonstrate the capabilities of the honeypot I showed 3 scenarios as validation (section 7) : *Basic behaviour*, *DoS attack*, *Command flooding*.

To take this work further, the honeypot should be publicly deployed to attract attackers to a larger Matter over Thread network with real devices. As the protocol was still new at the time I wrote this, I could not find a large Matter over Thread network available for testing on which to run my honeypot. Another improvement would be to simulate a Matter controller and give attackers access to it, to see what kind of packets they intend to send to the network.

While it remains premature to make definitive claims about the overall security of the Matter over Thread protocol, the use of such a honeypot can help us discover possible vulnerabilities and so be able to continuously improve it.

References

[1] *A19 - E26 smart bulb - 75 W (2-pack)*. [Online; accessed 7. Feb. 2025]. Feb. 2025. URL: https://www.philips-hue.com/en-us/p/hue-white-and-color-ambiance-a19-e26-smart-bulb-75-w/046677563257.

[2] *addons/openthread_border_router at master · home-assistant/addons*. [Online; accessed 3. Jan. 2025]. Jan. 2025. URL: https://github.com/home-assistant/addons/tree/master/openthread_border_router.

[3] Dimitrios-Georgios Akestoridis, Vyas Sekar, and Patrick 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*. WiSec '22: 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks. San Antonio TX USA: ACM, May 16, 2022, pp. 233–244. ISBN: 978-1-4503-9216-7. URL: https://dl.acm.org/doi/10.1145/3507657.3528544 (visited on 10/20/2024).

[4] *Apple TV 4K*. [Online; accessed 1. Apr. 2025]. Apr. 2025. URL: https://www.apple.com/uk/apple-tv-4k.

[5] Navodit Bhardwaj et al. "Mealy Machine Inference for Secure Matter Protocol Commissioning". In: *2024 IEEE 6th PhD Colloquium on Emerging Domain Innovation and Technology for Society (PhD EDITS)*. 2024 IEEE 6th PhD Colloquium on Emerging Domain Innovation and Technology for Society (PhD EDITS). Nov. 2024, pp. 1–2. DOI: 10.1109/PhDEDITS64207.2024.10838691. URL: https://ieeexplore.ieee.org/abstract/document/10838691 (visited on 02/15/2025).

[6] BlogPoster. "Production vs Research Honeypots: What's the Difference?" In: *Logix Consulting Managed IT Support Services Seattle* (Jan. 2021). URL: https://logixconsulting.com/2020/06/22/production-vs-research-honeypots-whats-the-difference.

[7] *Bridge*. [Online; accessed 7. Feb. 2025]. Feb. 2025. URL: https://www.philips-hue.com/en-us/p/hue-bridge/046677458478#overview.

[8] Théophile Coche and Théo Hernandez. "Honeypot on IoT Using Matter over Thread Protocol". PhD thesis. UCLouvain, 2024.

[9] *Connect Tutorial 6 - IEEE 802.15.4 Addressing*. [Online; accessed 30. Dec. 2024]. Dec. 2023. URL: https://community.silabs.com/s/article/connect-tutorial-6-ieee-802-15-4-addressing?language=en_US.

[10] *connectedhomeip*. [Online; accessed 3. Jan. 2025]. Jan. 2025. URL: https://github.com/project-chip/connectedhomeip.

[11] *cowrie*. [Online; accessed 18. Mar. 2025]. Mar. 2025. URL: https://github.com/cowrie/cowrie.

[12] *CSA-IOT*. [Online; accessed 11. Feb. 2025]. Feb. 2025. URL: https://csa-iot.org.

[13] *Discord*. [Online; accessed 18. Mar. 2025]. Mar. 2025. URL: https://discord.com.

[14] *DMZ Options for RV160/RV260 Routers*. [Online; accessed 1. Apr. 2025]. Apr. 2019. URL: https://www.cisco.com/c/en/us/support/docs/smb/routers/cisco-rv-series-small-business-routers/smb5875-dmz-options-for-rv160-rv260-routers.html.

[15] Seamus Dowling, Michael Schukat, and Hugh Melvin. "A ZigBee honeypot to assess IoT cyberattack behaviour". In: *2017 28th Irish signals and systems conference (ISSC)*. IEEE. 2017, pp. 1–6.

[16] *Echo Dot (5th Generation, 2022 Model) | Powerful Bluetooth and Wi-Fi Smart Speaker with Alexa | Navy Blue*. [Online; accessed 1. Apr. 2025]. Apr. 2025. URL: https://www.amazon.fr/-/en/echo-dot-2022/dp/B09B8RF4PY.

[17] *ESP32-C6 Wi-Fi 6 & BLE 5 & Thread/Zigbee SoC | Espressif Systems*. [Online; accessed 3. Jan. 2025]. Jan. 2025. URL: https://www.espressif.com/en/products/socs/esp32-c6.

[18] *ESP32-H2 Thread/Zigbee & BLE 5 SoC | Espressif Systems*. [Online; accessed 3. Jan. 2025]. Jan. 2025. URL: https://www.espressif.com/en/products/socs/esp32-h2.

[19] *Espressif ZeroCode*. [Online; accessed 3. Jan. 2025]. Dec. 2024. URL: https://zerocode.espressif.com.

[20] *Espressif's SDK for Matter documentation*. [Online; accessed 1. Apr. 2025]. Apr. 2025. URL: https://docs.espressif.com/projects/esp-matter/en/latest/esp32/introduction.html.

[21] *Eve Energy | evehome.com*. [Online; accessed 3. Jan. 2025]. Jan. 2025. URL: https://www.evehome.com/en/eve-energy.

[22] Javier Franco et al. "A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems". In: *IEEE Communications Surveys & Tutorials* 23.4 (2021). Conference Name: IEEE Communications Surveys & Tutorials, pp. 2351–2383. ISSN: 1553-877X. DOI: 10.1109/COMST.2021.3106669. URL: https://ieeexplore.ieee.org/abstract/document/9520645 (visited on 10/20/2024).

[23] *Global: smart home number of users 2019-2028| Statista*. [Online; accessed 6. May 2025]. May 2025. URL: https://www.statista.com/forecasts/887613/number-of-smart-homes-in-the-smart-home-market-in-the-world.

[24] Markus Helmut Gollmann. "Design of a Honeypot for Smart Home". In: (). URL: https://epub.fh-joanneum.at/obvfhjhs/content/titleinfo/7942213.

[25] *Grafana: The open and composable observability platform | Grafana Labs*. [Online; accessed 22. Mar. 2025]. Mar. 2025. URL: https://grafana.com.

[26] Home Assistant. *Home Assistant*. [Online; accessed 3. Jan. 2025]. Jan. 2025. URL: https://www.home-assistant.io.

[27] *Homepage*. [Online; accessed 6. May 2025]. Apr. 2025. URL: https://iot-analytics.com.

[28] *HomePod*. [Online; accessed 1. Apr. 2025]. Apr. 2025. URL: https://www.apple.com/uk/shop/buy-homepod/homepod.

[29] "IEEE Standard for Low-Rate Wireless Networks". In: *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)* (2020), pp. 1–800.

[30] *Installations | Home Assistant Analytics*. [Online; accessed 27. Mar. 2025]. Mar. 2025. URL: https://analytics.home-assistant.io.

[31] *Intel NUC8i5BE*. [Online; accessed 3. Jan. 2025]. URL: https://www.intel.com/content/dam/support/us/en/documents/mini-pcs/nuc-kits/NUC8i3BE_NUC8i5BE_NUC8i7BE_TechProdSpec.pdf.

[32] *IPv6 Addressing*. [Online; accessed 28. Dec. 2024]. Sept. 2023. URL: https://openthread.io/guides/thread-primer/ipv6-addressing.

[33] Richard Kelsey. *Mesh Link Establishment*. Internet-Draft draft-ietf-6lo-mesh-link-establishment-00. Work in Progress. Internet Engineering Task Force, Dec. 2015. 19 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-6lo-mesh-link-establishment/00/.

[34] Song Liao, Jingwen Yan, and Long Cheng. "WIP: Hidden Hub Eavesdropping Attack in Matter-enabled Smart Home Systems". In: *Proceedings 2024 Workshop on Security and Privacy in Standardized IoT*. Workshop on Security and Privacy in Standardized IoT. San Diego, CA, USA: Internet Society, 2024. ISBN: 979-8-9894372-6-9. DOI: 10.14722/sdiotsec.2024.23089. URL: https://www.ndss-symposium.org/wp-content/uploads/2024/07/sdiotsec2024-89-paper.pdf (visited on 02/15/2025).

[35] Xiaoyue Ma, Lannan Luo, and Qiang Zeng. "From One Thousand Pages of Specification to Unveiling Hidden Bugs: Large Language Model Assisted Fuzzing of Matter {IoT} Devices". In: *33rd USENIX Security Symposium (USENIX Security 24)*. 2024, pp. 4783–4800.

[36] *Matter : Commissioning*. [Online; accessed 23. Feb. 2025]. July 2022. URL: https://developers.home.google.com/matter/primer/commissioning.

[37] *Matter : The Device Data Model*. [Online; accessed 23. Feb. 2025]. Apr. 2023. URL: https://developers.home.google.com/matter/primer/device-data-model.

[38] *Matter Devices List*. [Online; accessed 3. Jan. 2025]. Jan. 2025. URL: https://www.matterdatabase.com.

[39] *Nest Hub (2nd Gen)*. [Online; accessed 1. Apr. 2025]. Apr. 2025. URL: https://store.google.com/gb/product/nest_hub_2nd_gen?hl=en-GB&pli=1.

[40] Netgear. *What is the De-Militarized Zone (DMZ) feature on my NETGEAR router?* [Online; accessed 1. Apr. 2025]. Apr. 2025. URL: https://kb.netgear.com/25891/What-is-the-De-Militarized-Zone-DMZ-feature-on-my-NETGEAR-router.

[41] *nRF Sniffer for 802.15.4*. [Online; accessed 3. Jan. 2025]. Jan. 2025. URL: https://docs.nordicsemi.com/bundle/ug_sniffer_802154/page/UG/sniffer_802154/intro_802154.html.

[42] *nRF-Sniffer-for-802.15.4/nrf802154_sniffer/nrf802154_sniffer.py*. [Online; accessed 8. Jan. 2025]. Jan. 2025. URL: https://github.com/NordicSemiconductor/nRF-Sniffer-for-802.15.4/blob/master/nrf802154_sniffer/nrf802154_sniffer.py.

[43] *nRF52840 Dongle*. [Online; accessed 3. Jan. 2025]. Jan. 2025. URL: https://www.nordicsemi.com/Products/Development-hardware/nRF52840-Dongle.

[44] *OpenThread*. [Online; accessed 8. Jan. 2025]. Dec. 2024. URL: https://openthread.io.

[45] *OpenWrt*. [Online; accessed 8. Mar. 2025]. Mar. 2025. URL: https://openwrt.org/start.

[46] *Oracle VirtualBox*. [Online; accessed 3. Jan. 2025]. Jan. 2025. URL: https://www.virtualbox.org.

[47] *ot-nrf528xx*. [Online; accessed 3. Jan. 2025]. Jan. 2025. URL: https://github.com/openthread/ot-nrf528xx.

[48] *prettytable*. [Online; accessed 7. Feb. 2025]. Feb. 2025. URL: https://github.com/prettytable/prettytable.

[49] *Proxmox Server Solutions*. [Online; accessed 22. Mar. 2025]. Mar. 2025. URL: https://www.proxmox.com/en.

[50] *pyshark*. [Online; accessed 23. Mar. 2025]. Mar. 2025. URL: https://github.com/KimiNewt/pyshark.

[51] *python-matter-server*. [Online; accessed 3. Jan. 2025]. Jan. 2025. URL: https://github.com/home-assistant-libs/python-matter-server.

[52] *Requests: HTTP for Humans™ — Requests 2.32.3 documentation*. [Online; accessed 12. Apr. 2025]. Jan. 2025. URL: https://docs.python-requests.org/en/latest/index.html.

[53] *Scapy*. [Online; accessed 21. Mar. 2025]. Nov. 2024. URL: https://scapy.net.

[54] Nordic Semiconductor. *Nordic Semiconductor | Specialists in Low Power Wireless*. [Online; accessed 1. Apr. 2025]. Apr. 2025. URL: https://www.nordicsemi.com.

[55] Kumar Shashwat et al. “Security Analysis of Trust on the Controller in the Matter Protocol Specification”. In: 2023 IEEE Conference on Communications and Network Security (CNS). Orlando, FL, USA: IEEE, Oct. 2, 2023, pp. 1–6. ISBN: 979-8-3503-3945-1. DOI: 10.1109/CNS59707.2023.10288747. URL: https://ieeexplore.ieee.org/document/10288747/ (visited on 02/17/2025).

[56] *Smart Home Products - Aeotec*. [Online; accessed 1. Apr. 2025]. Nov. 2024. URL: https://aeotec.com/products.

[57] *Thread*. [Online; accessed 27. Dec. 2024]. Dec. 2024. URL: https://www.threadgroup.org.

[58] *Thread 1.4.0 Specification*. Dec. 2024. URL: https://www.threadgroup.org/ThreadSpec.

[59] *Thread Benefits*. [Online; accessed 27. Dec. 2024]. Dec. 2024. URL: https://www.threadgroup.org/What-is-Thread/Thread-Benefits.

[60] *Thread Network Fundamentals*. [Online; accessed 8. Jan. 2025]. May 2020. URL: https://www.threadgroup.org/Portals/0/documents/support/Thread%20Network%20Fundamentals_v3.pdf.

[61] Ishaq Unwala, Zafar Taqvi, and Jiang Lu. “Thread: An IoT Protocol”. In: *2018 IEEE Green Technologies Conference (GreenTech)*. 2018 IEEE Green Technologies Conference (GreenTech). ISSN: 2166-5478. Apr. 2018, pp. 161–167. DOI: 10.1109/GreenTech.2018.00037. URL: https://ieeexplore.ieee.org/abstract/document/8373620 (visited on 10/20/2024).

[62] *Welcome to Click — Click Documentation (8.1.x)*. [Online; accessed 3. Jan. 2025]. Dec. 2024. URL: https://click.palletsprojects.com/en/stable.

[63] *Welcome to Flask — Flask Documentation (3.1.x)*. [Online; accessed 21. Mar. 2025]. Jan. 2025. URL: https://flask.palletsprojects.com/en/stable.

[64] *Welcome to Matter’s documentation — Matter documentation*. [Online; accessed 23. Feb. 2025]. Feb. 2025. URL: https://project-chip.github.io/connectedhomeip-doc/index.html.

[65] *What is a DMZ and how to configure DMZ host | TP-Link*. [Online; accessed 1. Apr. 2025]. Apr. 2025. URL: https://www.tp-link.com/us/support/faq/28.

[66] *What is Thread?* [Online; accessed 27. Dec. 2024]. Sept. 2023. URL: https://openthread.io/guides/thread-primer.

[67] *Wireshark*. [Online; accessed 23. Mar. 2025]. Mar. 2025. URL: https://www.wireshark.org.

[68] *Zigbee | Complete IOT Solution*. [Online; accessed 7. Feb. 2025]. Feb. 2025. URL: https://csa-iot.org/all-solutions/zigbee.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**