Fuzzing Matter(s): A White Paper for Fuzzing the Matter Protocol

Marcello Maugeri[a]
Department of Maths and Computer Science, University of Catania, Catania, Italy

Keywords: Fuzzing, IoT, Security Testing.

Abstract: IoT and smart home devices have transformed daily life, consequently raising more and more concerns about security vulnerabilities. Robust security testing methods are essential to fortify devices against potential threats. While dynamic analysis techniques, such as fuzzing, help identify vulnerabilities, some challenges arise due to diverse architectures, communication channels and protocols. Testing directly on devices overcomes difficulties in firmware emulation, but lack of protocol standardisation still poses hurdles. The recently released *Matter* protocol aims to unify smart home ecosystems, thus also simplifying security testing. In particular, Matter inherits the concept of *Cluster* from *Zigbee* in its *Data Model*. The Data Model clearly defines attributes, commands, status codes and events that could be leveraged to design automated security testing techniques such as fuzzing. This paper proposes the design of a fuzzing framework for Matter-enabled smart home devices. The framework employs stateful fuzzing to cover the inherent state-fullness of IoT devices. Such a framework would bestow benefits upon manufacturers, researchers, and end-users.

1 INTRODUCTION

The increasing growth of smart home devices has led to problems with competing standards and protocols. The *World Economic Forum* estimates that more than 130 million households owned at least a smart speaker – the most common smart home device – in 2022 (World Economic Forum, 2022). As smart home technology rapidly expands, a significant challenge arises for consumers who own devices from multiple vendors. Integrating new devices into their existing smart home ecosystem becomes a tedious task. Typically, each vendor requires users to download their specific app or, in the case of voice assistants such as *Amazon Alexa*, install the corresponding skill to enable communication with the new device. This process not only demands considerable time and effort but also results in a fragmented user experience.

Moreover, an additional concern emerges regarding device usability during service disruptions. In fact, as depicted in Figure 1, many smart home devices rely on cloud services to fully function. As a consequence, when the cloud service experiences downtime or connectivity issues, the device's functionality may be severely affected or temporarily disabled, causing inconvenience and limiting the device's reliability.

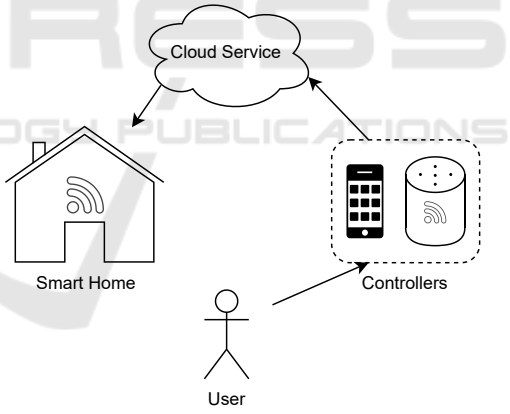

Figure 1: Typical usage of a smart home device.

As a solution for such issues, Matter establishes a new smart home standard that allows devices to be operated relying on the local network, improving user convenience and eliminating the need for multiple vendor-specific applications. To this extent, Matter (Connectivity Standards Alliance, 2023b) enables devices to work offline without requiring continuous access to the cloud, thus increasing the devices' security, especially for sensitive hardware like smart locks and security cameras. In addition, this functionality allows local testing, indirectly empowering end-users and researchers.

Within the realm of security testing techniques,

[a] https://orcid.org/0000-0002-6585-5494

Maugeri, M.
Fuzzing Matter(s): A White Paper for Fuzzing the Matter Protocol.
DOI: 10.5220/0012469200003648

In *Proceedings of the 10th International Conference on Information Systems Security and Privacy (ICISSP 2024)*, pages 446-451
ISBN: 978-989-758-683-5; ISSN: 2184-4356

fuzzing has gained prominence for its automated and effective approach to identifying vulnerabilities. The concept is straightforward—repeatedly execute a target system with inputs designed to provoke unexpected behaviours.

Despite its efficacy, fuzzing encounters challenges in the context of *embedded systems* (Eisele et al., 2022), such as smart home devices, due to their diverse architectures, communication channels, and protocols. While testing directly on devices can overcome issues related to firmware emulation, the lack of protocol standardisation remains a hurdle.

However, since Matter is expected to become the de-facto standard among smart home and IoT devices, this paper relies on the necessity of developing a fuzzer specifically tailored to Matter protocol. While adapting a pre-existing fuzzer to Matter might appear straightforward, the intricacies inherent in the protocol necessitate a specialised solution. Developing an effective framework involves utilising Matter's specifics, including events and attributes for inferring the state model, and leveraging the command list knowledge for precise input generation. Hence, this paper proposes a white paper of a possible solution.

The paper is structured as follows: Section 2 gives a brief introduction to the state-of-the-art of *embedded fuzzing*, followed by a brief description of the Matter protocol. Section 3 outlines the research goals, anticipates challenges, and provides an overview of the proposed fuzzing framework's design. Finally, Section 4 concludes the paper.

2 BACKGROUND

2.1 Embedded Fuzzing

In the pursuit of robust firmware security testing, several vulnerability-finding systems employ static analysis (Costin et al., 2014). While static analysis can uncover potential vulnerabilities, it often reports false positives (Costin et al., 2016) since the target firmware is not actually executed during the analysis. To address this limitation and achieve more accurate results, dynamic analysis is preferred.

Dynamic analysis methods, such as fuzzing, typically involve the emulation of the target firmware in a controlled and efficient environment, allowing security testers and the fuzzer itself to observe the behaviour and interactions with various inputs. Relevant emulation-based works include *Firmfuzz* (Srivastava et al., 2019), *FirmAE* (Kim et al., 2020) and *Icicle* (Chesser et al., 2023). Unfortunately, emulating firmware from different architectures and dealing with obfuscated code still pose significant challenges (Wright et al., 2021), hindering the efficacy of dynamic analysis. As a consequence, researchers and security experts often resort to testing directly on actual devices to overcome the challenges of accurately emulating firmware.

However, testing directly on devices effectively introduces its own set of challenges due to the lack of relevant feedback. *GDBFuzz* (Eisele et al., 2023) addresses the issue effectively by monitoring the target device through the debug interface. However, it is worth noting that not all devices have a debug interface available on their shield, causing this solution to be challenging to apply.

Then, it is preferred to use the network communication channel as an entry point. Another major issue is the limited resources of IoT devices, which may not be able to handle large-scale testing of numerous inputs in a short amount of time. This constraint calls for smart and efficient testing methodologies – for example by leveraging machine learning algorithms (Eceiza et al., 2021; Aichernig et al., 2021) – to maximise the effectiveness of the testing process.

Another significant challenge is the diversity of network communication channels used by IoT devices. The devices may employ different protocols and mechanisms for communication, leading to complex communication pathways. For example, a user might use a companion app on a smartphone to send commands to the cloud through the use of Wi-Fi. The cloud then relays the command to a smart hub on the local network, and the smart hub communicates with the actual device using a protocol such as Zigbee. This intricate network of communication necessitates careful consideration and testing of all communication paths to ensure comprehensive security coverage.

Additionally, the lack of standardisation in the application protocols used to communicate with IoT devices poses a significant hurdle. In some cases, devices may use standard protocols such as HTTP, but with custom payloads or unconventional implementations. Alternatively, *MQTT* may be used to leverage a publish-subscribe pattern for communication. These variations demand the development of tailored testing tools and methodologies for each unique target under test. As a consequence, most of the relevant works, such as *IoTFuzzer* (Chen et al., 2018), *DIANE* (Redini et al., 2021) leverage companion apps provided by the device's respective vendor to communicate directly with the device without inferring the underlying protocol and cipher.

Nevertheless, these tools are tailored to the respective companion app, hence they are not easily reusable

without requiring additional effort. In addition, most apps are obfuscated, nullifying this method. As a consequence, some tools such as *Snipuzz* (Feng et al., 2021) and *FIoTFuzzer* (Xu et al., 2022) work directly by mutating previously acquired messages to send to the target device. Both works employ guided mutation based on feedback inferred from the response provided by the target device. However, certain devices may not generate a detailed response and instead prioritise providing a simplistic error code. Also, the message acquisition phase is time-consuming and could not cover all possible functionalities.

To make a more comprehensive solution, *Ma et al.* (Ma et al., 2023) recently developed *HubFuzzer*, which leverages a device pretending to be a smart hub, to receive allowed commands from the target device directly. Unfortunately, HubFuzzer is not currently open-sourced, and it is designed to work with devices based on Zigbee and *Z-Wave* protocols only. In addition, *Ma et al.* discuss the possible re-using of this approach with the Matter protocol. Still, this paper foresees that it will require extensive additional work to provide an effective solution, due to the Matter's specification complexity.

2.2 Matter Protocol

The Matter protocol defines a seven-layer stack as depicted in Figure 2. At the topmost layer, the *Application Layer* orchestrates the high-level business logic, managing tasks such as turning on or off a smart home device. Situated beneath the first layer, the *Data Model Layer* serves as a fundamental component, elucidating data and verb elements pivotal for the protocol's delineation. Notably, this layer introduces the concept of a Cluster: a specification that intricately defines attributes, commands, behaviours, and dependencies for a set of functionalities.

In particular, a *Command* is a data field set that triggers a behaviour in the receiver. The receiver, upon receiving a command, can respond with a success or error status code. Alternatively, it may remain silent (no response), or reply with a specific response command if the initial command is not a request. Note that the status code could be cluster-specific or a global status code defined from the underlying *Interaction Model*.

Another integral element within a Cluster is an *Attribute*. Attributes encompass queryable or settable states, configurations, or capabilities of the device. For instance, a smart bulb may feature the *On/Off* attribute to indicate its operational status. A more intricate example is the *AcceptedCommandList*, detailing the supported commands. Similar functionalities are achieved through *AttributeList* for attributes or *EventList* for events.

Specifically, *Events* serve as records of past state transitions. While attributes convey current states, events function as a historical journal, equipped with a monotonically increasing counter, timestamp, and priority. This design enables the capture of state transitions and facilitates data modelling that extends beyond the capabilities of attributes.

Moving further down, the Interaction Model layer sets the stage for communication between devices of the same Fabric acting as a client or a server. A Fabric is a set of nodes that interact by accessing data model elements. For completeness, a *Node* is a unique addressable entity, usually a physical device, but can also be a logical instance of it. In addition, a node is composed of one or more endpoints adhering to a specific device type, and the Cluster is an instance of an endpoint.

Subsequently, the *Action Framing* layer is responsible for encoding the message into a prescribed binary format for network transmission. Consequently, security measures are accomplished in the *Security* layer. In this layer the action frame is processed, undergoing encryption and the addition of a message authentication code using the *AES-CCM* mode as defined in *NIST 800-38C* (Dworkin, 2007).

Finally, the *Message Layer* takes charge of constructing the payload format post-encryption. This layer adds required and optional header fields, specifying message properties and logical routing information. The finalised payload is then ready for transmission via the underlying transport protocol, either TCP or the Matter's *Message Reliability Protocol (MRP)* through UDP, for effective IP management of the data. As the data reaches the peer device, it ascends the protocol stack, with each layer reversing the operations performed by the sender.

3 FUZZING MATTER(S)

3.1 Research Goals

Matter offers a comprehensive definition of smart home device functionalities through its Cluster specification. Additionally, the *Matter Application Cluster Library (MACL)*, curated by the Connectivity Standards Alliance, serves as a dynamic repository for cluster functionalities, regularly updated with new features. Essentially, the MACL comprises multiple sets of clusters for general use. The primary research

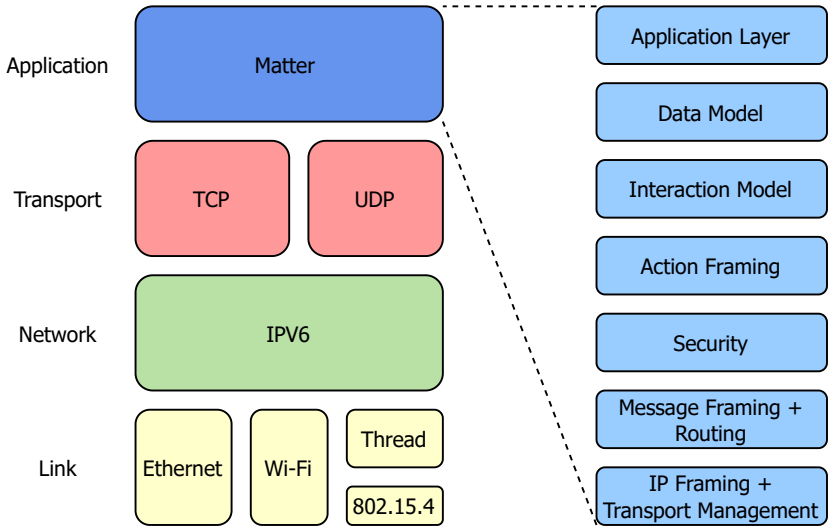

Figure 2: Stack of the Matter Protocol.

goal is to develop a fuzzer capable of effectively covering all functionalities outlined in the MACL.

Moreover, smart home devices inherently possess statefulness. In Matter, this state can be extracted from queryable attributes and events within a Cluster, laying the foundation for the design of a stateful fuzzer (Daniele et al., 2023). This dual-pronged research approach aims to enhance security testing by addressing the unique characteristics of Matter's Cluster functionalities and the inherent statefulness of smart home devices.

3.2 Expected Challenges

The project expects the following challenges:

1. **Benchmark** – Acquiring a relevant set of real case studies with multiple natures to build a benchmark for the design and evaluation of the fuzzer.
2. **Information Gathering from Static Analysis of the Firmware –** Although static analysis cannot be always applicable, existing techniques could be tailored to obtain Matter-relevant information from the firmware.
3. **Information Gathering from Firmware Emulation –** Although firmware emulation is not always possible, an alternative approach could be tailored to perform fuzzing in an emulated environment, improving test speed and validating the obtained results with the real device.
4. **Handling Multiple Link Layers –** Matter primarily operates with Wi-Fi and Thread protocols, but it also supports other protocols such as Zigbee. Therefore, the fuzzer must be compatible with all link layer protocols specified in Matter.
5. **Stateful Fuzzing –** IoT devices often have stateful behaviour, where their actions and responses depend on previous inputs. Fuzzing stateful protocols can be more complicated than stateless ones. The proposed solution will likely need to design stateful fuzzing strategies to adequately cover all possible device states and interactions related to the Matter protocol.
6. **Handling Vendor-Specific Implementations –** Different manufacturers may implement the Matter protocol differently, introducing variations in the behaviour of devices. For example, a manufacturer could define its own Cluster specification. The fuzzer should account for vendor-specific implementations and adapt the fuzzing techniques accordingly.
7. **Validating True Positives** – Fuzzing often produces numerous inputs and potential vulnerabilities. The proposal requires effective methods to validate and reproduce the obtained findings accurately.

3.3 Proposed Framework

The proposed fuzzing framework consists of four components, illustrated in Figure 3.

Initially, the *Input Generator* creates input following the Cluster specification, utilising information such as accepted commands, queryable/settable attributes, and accessible events. It is important to note that, at the current design stage, specific input generation algorithms have not been chosen as any grammar-

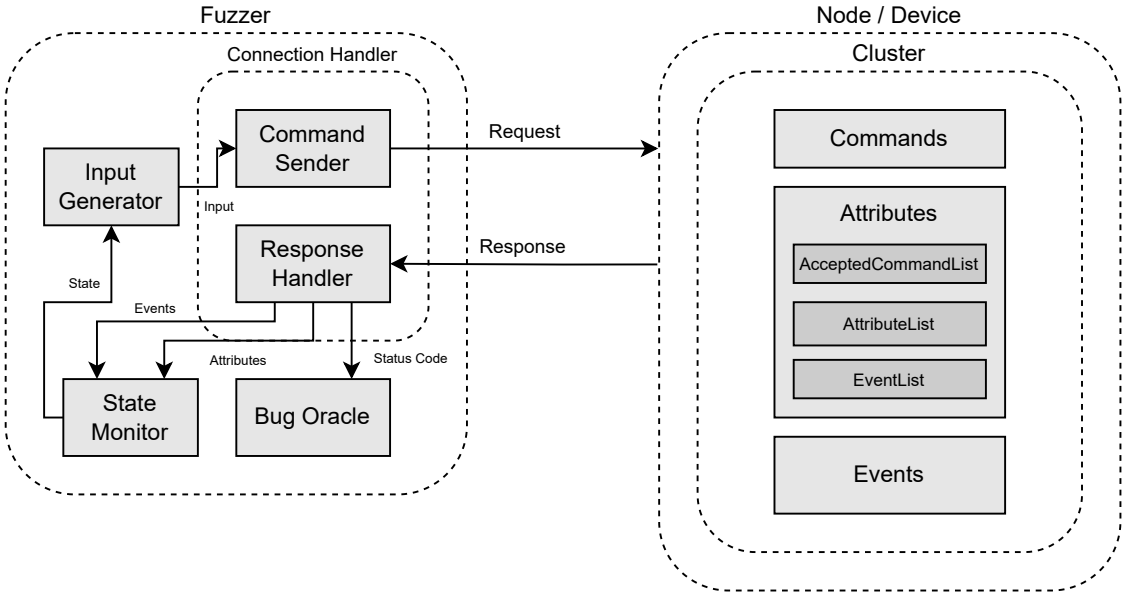

Figure 3: Proposed Framework.

based approach adapted to the problem should fit. However, the modular design of the framework facilitates easy replacement with minimal effort.

Once the input is generated, the *Connection Handler* engages with the target node (or device), managing the transmission of input generated by the Input Generator and receiving any corresponding responses. Given Matter's use of Thread, Wi-Fi, and Ethernet as networking technologies, the fuzzer must support communication with any Matter-compliant device, accommodating various link layer protocols.

Responses can carry diverse information. If the request involves a command, the response should include a success or error status code, signalling potential bugs to the *Bug Oracle*. A bug might manifest as an unexpected success code inappropriately granted or an unforeseen error code in a properly formed command.

Additionally, a response may include the queried attribute or event, updating the *State Monitor* on any state transitions. The fuzzer should account for the stateful nature of smart home devices, considering variables such as device states (e.g., on or off). Understanding the current state proves valuable during input generation, as the same input may trigger different behaviours based on the device's state.

4 CONCLUSIONS

In conclusion, this paper has explored the necessity and design of a fuzzing framework tailored for Matter-enabled smart home devices. By leveraging the inherent structure provided by Matter's Cluster specification (Connectivity Standards Alliance, 2023a) and the MACL, the proposed framework aims to comprehensively cover all aspects of device functionality. The modular nature of the framework allows for flexibility, enabling potential advancement and extensions.

Furthermore, considering the stateful nature of smart home devices, the framework incorporates a State Monitor to capture state transitions during testing. Also, the Connection Handler component facilitates interaction with Matter-compliant devices, ensuring compatibility across different link layers.

The fuzzer aims to be open-sourced and promote collaboration and knowledge sharing within the smart home and fuzzing community, providing valuable testing methods to companies, researchers, and end-users alike. As such, the advances of this project will be published on the web page https://fuzzingmatter.github.io/ which anyone is welcome to join.

REFERENCES

Aichernig, B. K., Muškardin, E., and Pferscher, A. (2021). Learning-based fuzzing of iot message brokers. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 47–58.

Chen, J., Diao, W., Zhao, Q., Zuo, C., Lin, Z., Wang, X., Lau, W., Sun, M., Yang, R., and Zhang, K. (2018). Iotfuzzer: Discovering memory corruptions

in iot through app-based fuzzing. In *Network and Distributed System Security Symposium*.

Chesser, M., Nepal, S., and Ranasinghe, D. C. (2023). Icicle: A re-designed emulator for grey-box firmware fuzzing. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2023, page 76–88, New York, NY, USA. Association for Computing Machinery.

Connectivity Standards Alliance (2023a). Matter Application Cluster Specification 1.2.

Connectivity Standards Alliance (2023b). Matter Core Specification 1.2.

Costin, A., Zaddach, J., Francillon, A., and Balzarotti, D. (2014). A Large-Scale analysis of the security of embedded firmwares. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 95–110, San Diego, CA. USENIX Association.

Costin, A., Zarras, A., and Francillon, A. (2016). Automated dynamic firmware analysis at scale: A case study on embedded web interfaces. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '16, page 437–448, New York, NY, USA. Association for Computing Machinery.

Daniele, C., Andarzian, S. B., and Poll, E. (2023). Fuzzers for stateful systems: Survey and research directions. *arXiv preprint arXiv:2301.02490*.

Dworkin, M. (2007). Recommendation for block cipher modes of operation: The ccm mode for authentication and confidentiality. Technical report, National Institute of Standards and Technology.

Eceiza, M., Flores, J. L., and Iturbe, M. (2021). Fuzzing the internet of things: A review on the techniques and challenges for efficient vulnerability discovery in embedded systems. *IEEE Internet of Things Journal*, 8(13):10390–10411.

Eisele, M., Ebert, D., Huth, C., and Zeller, A. (2023). Fuzzing embedded systems using debug interfaces. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2023, page 1031–1042, New York, NY, USA. Association for Computing Machinery.

Eisele, M., Maugeri, M., Shriwas, R., Huth, C., and Bella, G. (2022). Embedded fuzzing: a review of challenges, tools, and solutions. *Cybersecurity*, 5(1):18.

Feng, X., Sun, R., Zhu, X., Xue, M., Wen, S., Liu, D., Nepal, S., and Xiang, Y. (2021). Snipuzz: Black-box fuzzing of iot firmware via message snippet inference. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 337–350, New York, NY, USA. Association for Computing Machinery.

Kim, M., Kim, D., Kim, E., Kim, S., Jang, Y., and Kim, Y. (2020). Firmae: Towards large-scale emulation of iot firmware for dynamic analysis. In *Annual Computer Security Applications Conference*, ACSAC '20, page 733–745, New York, NY, USA. Association for Computing Machinery.

Ma, X., Zeng, Q., Chi, H., and Luo, L. (2023). No more companion apps hacking but one dongle: Hub-based blackbox fuzzing of iot firmware. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*, MobiSys '23, page 205–218, New York, NY, USA. Association for Computing Machinery.

Redini, N., Continella, A., Das, D., De Pasquale, G., Spahn, N., Machiry, A., Bianchi, A., Kruegel, C., and Vigna, G. (2021). Diane: Identifying fuzzing triggers in apps to generate under-constrained inputs for iot devices. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 484–500.

Srivastava, P., Peng, H., Li, J., Okhravi, H., Shrobe, H., and Payer, M. (2019). Firmfuzz: Automated iot firmware introspection and analysis. In *Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things*, IoT S&P'19, page 15–21, New York, NY, USA. Association for Computing Machinery.

World Economic Forum (2022). The market for smart home devices is expected to boom over the next 5 years.

Wright, C., Moeglein, W. A., Bagchi, S., Kulkarni, M., and Clements, A. A. (2021). Challenges in firmware rehosting, emulation, and analysis. *ACM Comput. Surv.*, 54(1).

Xu, Z., Huang, W., Fan, W., and Cheng, Y. (2022). Fiotfuzzer: Response-based black-box fuzzing for iot devices. In *2022 IEEE/ACIS 22nd International Conference on Computer and Information Science (ICIS)*, pages 239–244.