MaDoS: Matter DoS Attacks via Secure Channel Status Reports

Farzam Zohdi
farzam.zohdi@eurecom.fr
EURECOM
Sophia Antipolis, France

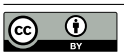
Daniele Antonioli
daniele.antonioli@eurecom.fr
EURECOM
Sophia Antipolis, France

Abstract

Matter is a standard for interoperable smart homes, governed by a consortium of over 200 companies, like Apple, Google, and Amazon. A Matter network, called a fabric, can operate without an Internet connection and uses popular link layers such as Wi-Fi, Thread, or Bluetooth Low Energy. Matter provides a feature-rich application-layer protocol, including secure session-establishment mechanisms that should guarantee confidentiality, integrity, and availability. Prior work has partially explored DoS threats on Matter, although availability is essential and safety-critical for smart homes. For example, no prior work has covered Matter DDoS. We analyze the Matter standard and SDK and uncover two (D)DoS design vulnerabilities in the specification of Matter status report application-layer messages and an issue in the Matter discovery procedures. The three flaws (V1–V3) affect all versions of the Matter standard, including the latest (v1.4.2).

We show how to exploit V1–V3 via three new Matter DoS attack classes: rogue device (RD), spoofer (SP), and Machine-in-the-Middle (MI). Each class maps to a real-world attacker model; e.g., RD is a rogue device that eavesdrops on and injects unauthenticated, unencrypted Matter packets, but has no access to the target fabric. The attacks, dubbed MaDoS, exploit status reports to DDoS a Matter fabric by preventing its devices from establishing secure sessions, i.e., devices can't add new devices or manage existing ones. Since the attacks exploit design issues in the Matter standard, they are effective regardless of the Matter version (1.0–1.4.2), transport layer (TCP, UDP, BTP), or link layer (Wi-Fi, BLE, or Thread).

We create `mados`, a low-cost toolkit for evaluating our attacks in real or simulated environments. The toolkit enables packet injection, Machine-in-the-Middle (MitM), and spoofing attacks, and is reproducible because it uses open-source software and available hardware. We experimentally confirm the effectiveness of the MaDoS vulnerabilities and attacks on real-world devices using `mados`. We exploit 13 Matter devices communicating over Wi-Fi, BLE, and Thread. Our device sample spans Matter versions v1.0–v1.4.2. We also experimentally DDoS a fabric in a controlled environment, rendering its devices unresponsive by blocking all session establishment attempts. We discuss two practical fixes to prevent the attacks by properly specifying status report messages and authenticating operational discovery. We responsibly disclosed our findings to the Matter consortium.



ASIA CCS '26, Bangalore, India

ACM ISBN 979-8-4007-2356-8/26/06
https://doi.org/10.1145/3779208.3807481

CCS Concepts

• **Security and privacy → Security protocols**; **Denial-of-service attacks**; **Mobile and wireless security**.

Keywords

Matter, PASE, CASE, Denial-of-Service

ACM Reference Format:
Farzam Zohdi and Daniele Antonioli. 2026. MaDoS: Matter DoS Attacks via Secure Channel Status Reports. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '26), June 1–5, 2026, Bangalore, India.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3779208.3807481

1 Introduction

Matter is a standard for interoperable, secure smart homes [19]. It enables the management of smart devices such as cameras, presence sensors, and door locks via a vendor-agnostic *application layer (AL) protocol* and related data model. A Matter network is called a *fabric* and is centered on IPv6. Matter supports standard transport layers (TCP, UDP) and link layers (Wi-Fi, Ethernet, BLE, and Thread). It does not require an Internet connection or a backend to operate. According to a recent study [10], more than 5.5 billion Matter devices will be shipped by 2030.

Matter is specified in a semi-open standard maintained by a consortium of companies called Connectivity Standards Alliance (CSA). Its latest public version is *v1.4.2* [20]. It also has a development version behind a $20,000 paywall. The CSA also provides the official Matter Software Development Kit (SDK) [18] for free, which includes a Matter reference implementation, examples, and documentation [17].

The Matter standard has two secure session-establishment protocols, *Passcode Authenticated Session Establishment (PASE)* and *Certificate Authenticated Session Establishment (CASE)*. PASE is a Password Authenticated Key Exchange (PAKE) protocol generating an authenticated session key from a shared password (also called Matter passcode). CASE is a SIGn-and-MAc (SIGMA) based protocol generating authenticated session keys using digital certificates. Additionally, the specification defines *CASE Resume*, an alternative to CASE for quick session re-establishment using a cached Shared Secret. Matter does not rely on TLS; instead, it uses PASE, CASE, and CASE Resume at the AL to protect its messages.

This work focuses on *Denial-of-Service (DoS) threats* on Matter. In a DoS attack, a malicious device targets the *availability* of one or more Matter devices. In a DDoS attack, multiple malicious devices DoS one or more targets in parallel. (D)DoS attacks have severe, large-scale security and safety implications for Matter. Since they target devices' availability, they can cause physical damage to a property and its inhabitants (e.g., missed fire alarms) and unauthorized, stealthy access to the property (e.g., missed intrusion alarms).

In fact, the Matter standard [20, Sec. 13.7] and security and privacy fundamentals [15] consider them a critical risks.

Matter's resilience against DoS attacks has not been systematically explored in academic literature. A master's thesis discusses DoS attacks on Matter [21, 40], while a recent poster [41] shows transport and network layer DoS attacks. A paper studied physical-layer jamming attacks against Thread, one of the Matter link layers [36]. A recent Black Hat talk presented a delayed DoS attack based on CASE message replay [27]. Other research focused on Matter's unauthorized access and eavesdropping attacks [39, 49], testbeds [44], and fuzzing [42]. To date, no academic paper has studied DDoS attacks against Matter.

In this work, we address this relevant gap by presenting the first study of (D)DoS attacks on Matter devices and fabrics using *Matter status-report application-layer messages*. We extensively analyze the latest Matter standard and SDK (v1.4.2) and uncover two DoS vulnerabilities in the specification of Matter status reports. We find that an adversary *not enrolled in a fabric* can (D)DoS a device or the fabric via unauthenticated status updates. The attacker, by DoS'ing CASE, CASE Resume, and PASE, prevents any device in the fabric from communicating and any external device from joining the fabric. Moreover, we find a vulnerability in Matter discovery procedures (V3) that, when combined with V1 and V2, enables additional attacks, i.e., (D)DoS via device impersonation or link-layer MitM.

We unveil **MaDoS**, a family of practical, low-cost, and impactful (D)DoS attacks against Matter exploiting V1–V3. The attacks are grouped into *three classes* according to their attacker model: i) a *rogue device (RD)* eavesdropping and injecting unprotected Matter packets, ii) a *spoofer (SP)* abusing Matter discovery to respond maliciously to CASE, CASE Resume, and PASE sessions, iii) a *link-layer Machine-in-the-Middle (MI)* attacker targeting CASE, CASE Resume, and PASE. In all scenarios, the attacker uses Matter status reports, is not commissioned into the fabric, and does not know any Matter secret material. A high-level representation of the three MaDoS attack classes is shown in Figure 1.

The MaDoS attacks rely on a practical and stealthy attack vector: Matter's *status reports*. These AL messages are effective because they can be sent without authentication to DoS PASE, CASE, and CASE Resume, even when victims are running a secure session. They are stealthy because they are valid Matter messages that do not violate the Matter specification. In particular, the attacks abuse four status messages used to report an invalid parameter (`INV_PAR`), a busy state (`BUSY`), the lack of a root of trust (`NO_TRUST`), and section success (`SES_SUC`).

We provide `mados`, a novel toolkit for testing the MaDoS RD, SP, and MI attacks in both simulated and real-world scenarios. The toolkit includes four modules: i) `PacketInjector`, ii) `PacketManipulator`, iii) `Spoofer`, and iv) `MitMTester`. `mados` is low-cost and reproducible, employing open-source software (Matter SDK and Scapy), and available hardware (ESP32 boards).

We empirically confirm that the attacks are impactful and practical in 25 evaluation scenarios involving 13 real-world Matter devices and 4 applications. Among others, we exploit the official Android, iOS, Linux, and ESP Matter stacks. The target devices employ the commissioner, commissioned, commissionee, session initiator, and session responder roles. The tested attacks are effective regardless

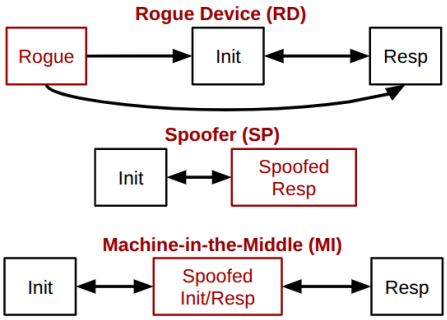


Figure 1: The three MaDoS attack classes. Rogue Device (RD) eavesdrops and injects packets. Spoofer (SP) spoofs a trusted responder to an initiator. Machine-in-the-Middle (MI) has a link-layer MitM position between Init and Resp.

of the Matter version (v1.0–v1.4.2), device role, transport layer (UDP, BTP), link layer (Wi-Fi, BLE, Thread), and vendor-specific implementation details. We further demonstrate the critical, fabric-wide impact of MaDoS testing a DDoS attack against an actual fabric with 6 devices. By launching parallel instances of our attacks, we successfully render the fabric unresponsive, preventing any session from being established.

To address the MaDoS attacks and their root causes, we discuss two countermeasures. F1 provides more precise specifications for CASE, CASE Resume, and PASE status reports to address V1 and V2. While F2 authenticates operational discovery using available Matter credentials to partially fix V3. The fixes require a minimal update to the standard and are easy to integrate, as Matter devices support secure Over The Air (OTA) firmware updates [20, Sec. 13.5]. For space constraints, the fixes are described in Appendix A.

Our contributions are as follows:

- We present the first assessment of Matter against (D)DoS threats. We uncover two design flaws affecting its secure status report messages and show how to exploit them with three new attack classes we call MaDoS RD, SP, and MI.
- We provide `mados`, an open-source, low-cost, and reproducible toolkit for testing MaDoS attack instances in simulated and real-world environments.
- We empirically confirm that the attacks are practical and impactful by exploiting 13 Matter devices and DDoS a real-world Matter fabric.
- We discuss two effective fixes to address the attacks and their root causes by improving the design of Matter's status report and operational discovery.

Disclosure, Ethics, and Availability. We responsibly disclosed our findings to the CSA. We are in contact with their security team as our findings are being evaluated, and we are happy to collaborate with them. We discuss the disclosure details in Appendix B. We conducted our experiments ethically to avoid security and privacy risks. We tested only our devices in a controlled environment, and we did not involve third-party devices or individuals. We provide the `mados` toolkit in a repository at https://github.com/AoiKujira/mados. We will open–source it after the responsible disclosure process.

2 Background

Next, we introduce Matter and explain its AL packet structure and the notation used in the paper. Then, we describe Matter security and privacy mechanisms, including its secure session establishment protocols.

2.1 Matter Introduction

Matter, formerly known as Connected Home IP (CHIP), standardizes a smart home technology based on IPv6. It enables cross-vendor management of smart devices, including safety, security, and privacy-related ones, such as cameras, doorbells, smart lights, water, smoke, and carbon monoxide sensors.

As shown in Figure 9 (in Appendix C), Matter supports popular wired and wireless communication technologies and protocols, including Ethernet, Wi-Fi, Thread, Bluetooth Low Energy (BLE), TCP, and UDP. Moreover, it provides two custom (TCP-like) transport protocols: Bluetooth Transport Protocol (BTP) for Matter over BLE and Message Reliability Protocol (MRP) for reliable exchange of Matter messages over UDP. Matter defines a vendor-agnostic AL protocol for securely exchanging data and commands. The protocol provides a hierarchical data model and secure session establishment based on passcodes (PASE) and digital certificates (CASE, CASE Resume).

A Matter network is called a *fabric* and is identified by a unique 64-bit Fabric ID. A device joins a fabric by being *commissioned*, and it gets a unique 64-bit fabric node ID. A fabric can be established without an Internet connection and a (proprietary) backend. This allows vendor-agnostic management of the smart home. A fabric typically includes a Wi-Fi access point, a smartphone, and several smart home devices. A Matter Border Router can bridge Wi-Fi to a Thread mesh network. A Matter device has one or multiple *roles* in a fabric, including:

- *Commissioner*: commission (onboard) devices
- *Commissionable*: can be commissioned
- *Commissionee*: is being commissioned
- *Commissioned*: is commissioned
- *Controller*: manage controllable devices
- *Controlee*: can be controlled

The public version of the Matter specification is updated twice a year and freely distributed as a PDF upon request [20]. Its main document is called *Matter Core Specification* and includes security and privacy features by design. The standard is maintained by the CSA, a consortium of over 200 companies, including Apple, Google, Amazon, Comcast, IKEA, Huawei, and Schneider. The standard is semi-open, with a public version and a development version behind a paywall. The paper refers to the latest public version (v1.4.2).

A Matter device joins a fabric via a standard Matter procedure called *commissioning*: i) the Commissionee communicates an *onboarding payload* to the Commissioner via an out-of-band channel (e.g., QR code, NFC, or manual input). The payload includes device-specific information and a password. ii) The devices use the password to create a secure session via PASE, and exchange fabric-specific data including credentials and certificates, iii) the Commissioner and the Commissionee confirm that they have the correct fabric certificate via CASE.

Table 1: Matter layer header fields. B: Bytes, opt: optional, var: variable.

Name	B	Description
Message flags	1	ver, enc, ack, routing, …
Session Id	2	key id, 0x00 for PASE, CASE
Security Flags	1	privacy, unicast, multicast, …
Message counter	4	directional, replay protection
Source Id	8	opt, routing
Destination Id	2,8	opt, routing
Msg Ext	var	opt, backward-compliant ext

A Matter AL packet has a *Matter layer*, which contains a header and a *Protocol layer* payload [20, Tables 8, 9]. The Message layer header fields are listed in Table 1, while the Protocol layer ones are in Table 10 in Appendix C. The Protocol payload contains the final payload, e.g., PASE or CASE message. To improve the paper's readability, we adopt a concise notation for the AL messages. The notation is summarized in Table 9, and includes roles, fields, opcodes, message counters, acknowledgment counters, keys, and status reports.

2.2 Matter Security and Privacy

The security and privacy of Matter communication rely on five properties described in the standard [15]:

(1) *Comprehensive protection* provided by device authentication and attestation, encrypted communication, and secure firmware updates.
(2) *Strong mechanism*, such as AES-CCM, SHA256, and ECC over secp256r1, ensure that Matter takes advantage of standard cryptographic ciphers.
(3) *Easy to use* security aspects enable vendors and users to take advantage of reference implementations.
(4) *Resilient approaches* are employed to protect, detect, and recover from security or privacy incidents.
(5) *Crypto agility* enables updating the Matter cryptographic primitives and protocol to address future threats in a backward-compliant way.

Matter provides Authenticated Encryption with Associated Data (AEAD) [20, Figures 8, 9]. It encrypts and authenticates an AL packet starting from the Protocol layer and authenticates the rest as associated data. It uses AES-128-CCM, which employs AES in CTR mode for confidentiality and AES CBC-MAC for message integrity. The AES-CCM session keys are established using PASE and CASE, which are described next.

2.3 PASE

PASE establishes an encrypted and authenticated session between a Commissioner (I) and a Commissionee (R) based on a shared password. It runs at the AL over BLE, Wi-FI, or Ethernet. It is based on *SPAKE2+*, an asymmetric PAKE protocol [34]. To bootstrap PASE, I must know the password, which is typically exchanged via a QR code. I does not store the password but a password verifier (hash).

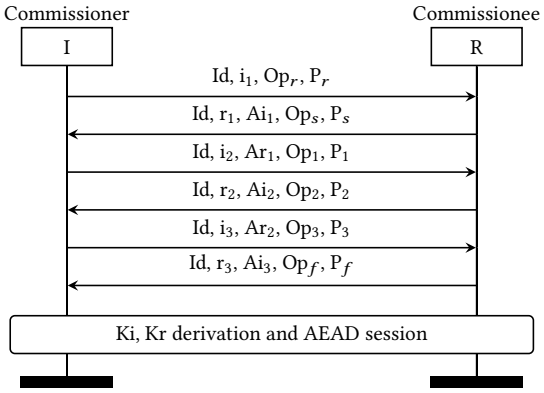


Figure 2: PASE secure session establishment. I and R establish directional session keys (Ki, Kr) using a shared password and SPAKE2+. The six shown messages are neither encrypted nor authenticated. PASE runs over TCP, MRP (UDP), or BTP transports and BLE, Wi-FI, or Ethernet link layers.

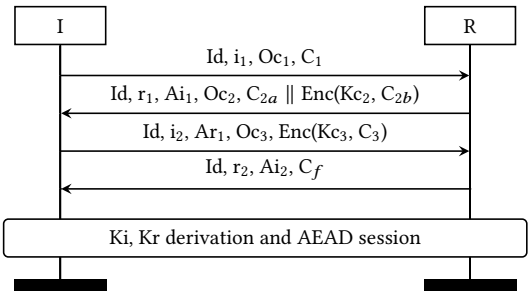


Figure 3: CASE secure session establishment. Some of its messages are (partially) encrypted. C_2 (Sigma2) has a plaintext header (C_{2a}) concatenated (||) with an encrypted footer (C_{2b}). CASE runs over Wi-FI, Ethernet, or Thread. Kc_2 and Kc_3 are intermediate symmetric keys.

Then, I and R find each other using *commissioning discovery*. Over Wi-Fi or Ethernet, the procedure employs Multicast DNS (mDNS) [33] and DNS-Based Service Discovery (DNS-SD) [32] and requires that the devices be connected to the same network. mDNS and DNS-SD enable vendor-agnostic service discovery without a centralized DNS server. For commissioning discovery over BLE, R broadcasts a BLE advertisement with the standard 16-bit Matter BLE UUID (`0xF6FF`) and I discovers R via BLE scanning. The discovery packets are neither encrypted nor authenticated.

After a successful discovery, I and R run the PASE session establishment protocol shown in Figure 2. The protocol runs at the AL over TCP, MRP (over UDP), or BTP (over BLE). I sends a message including some session identifiers (Id), a directional message counter (i_1), a PASE opcode (Op_r), and a P_r (PBKDFParamRequest) PASE payload. R answers with a message containing Id, a directional message counter r_1, a message counter acknowledgment ($Ai_1 = i_1$), Op_s, and P_s (PBKDFParamResponse). The P_r and P_s payloads negotiate the password-based KDF parameters to derive a PASE session key from the password shared during the bootstrap phase.

To describe the next PASE messages, we assume, for brevity, that each party uses the proper message counters and opcodes. I sends a P_1 (Pake1) payload including a public key (pA). R sends P_2 (Pake2), which includes pB and a confirmation code (cB). I responds with P_3 (Pake3), which contains cA. The confirmation codes are computed using the shared password and the public keys. Then, I and R mutually authenticate by verifying cA and cB. R signals a successful PASE session with P_f (PakeFin). So far, the PASE messages are neither encrypted nor authenticated. Table 7 in the Appendix summarizes the payloads used in the PASE handshake.

Next, I and R derive two directional symmetric keys (Ki, Kr) and encrypt the session using AES-128-CCM keyed with Ki (I→R) and Kr (R→I). Then they securely exchange data relevant to the fabric, such as certificates, network credentials, and access control lists. Via PASE devices complete the first commissioning phase. The second and last phase is carried out using CASE.

2.4 CASE and CASE Resume

CASE establishes a secure session between I and R using digital certificates exchanged during commissioning over PASE. It is used in two scenarios: i) by a Commissioner to complete commissioning after PASE, ii) by two Commissioned devices to communicate in the fabric. It runs at the AL over TCP or UDP transports and Wi-Fi, Ethernet, or Thread link layers. It is based on SIGMA, a family of AKE protocols [37].

The CASE discovery procedure, called *operational discovery*, relies on mDNS and DNS-SD and is neither encrypted nor authenticated. For readability, we focus our descriptions on the CASE payloads, assuming correct message counters, CASE opcodes, etc. After successful discovery, I and R run the CASE handshake shown in Figure 3. I sends a CASE opcode (O_{c1}) and a C_1 (Sigma1) payload, including an ephemeral public key. R answers with C_2 (Sigma2), which has a plaintext header (C_{2a}) and a ciphertext payload (C_{2b}) including a certificate and a signature. The latter is AES-CCM-encrypted using Kc_2, an intermediate symmetric key derived via DH from the ephemeral keys.

I replies with a C_3 (Sigma3) message including a certificate and a signature, which is encrypted with an intermediate key (Kc_3). I and R mutually authenticate using the certificates and the signatures. CASE successfully completes with a C_f (SigmaFin) message from R. Next, I and R derive directional symmetric keys (Ki, Kr) and securely exchange messages using AES-CCM.

The CASE Resume protocol enables two commissioned devices to establish a secure session by reusing cached session keys. The protocol, shown in Figure 12, starts with I sending a C_{1r} (Sigma1 with Resumption) payload, which has the Resumption ID field set. R replies with C_{2r}, including a Resume2MIC, which is an AEAD-based MIC, authenticating S2RK. After validation of Resume2MIC

by I, it signals CASE Resume success to R with a C_f message. CASE Resume is faster than CASE, as its handshake is shorter and no certificate verification is involved. Table 8 in the Appendix summarizes the CASE and CASE Resume handshake payloads.

3 Motivation and Threat Model

3.1 Motivation

The security of Matter to date is a relatively unexplored area in academia. [49] highlights a Matter commissioning security issue enabling unauthorized access via an unenrolled channel. [39] presents vulnerabilities in device commissioning and delegation, introducing a hidden eavesdropping attack that exposes sensitive data. In [46], the authors manage to extract Device Attestation Certificate (DAC) private keys using fault-injection attacks.

Other works analyze the security mechanisms in the Matter standard. In [21, 40], the authors evaluate the standard through threat modeling, formal verification, and vulnerability analysis. They identified potential weaknesses, vendor-specific risks, and deviations from security promises, and provided recommendations and symbolic verification models. Authors in [50] examine scenarios where malicious controllers could harm a fabric.

This work presents the first assessment of (D)DoS threats against Matter. Prior research papers present theoretical DoS attacks extrapolated from the standard [21, 40], link-layer jamming attacks on Thread, focusing on OpenThread [29] and its jamming detection mechanism [36]. Moreover, a poster describes a testbed for Syn, LAND, IP fragment, ACK flood, or UDP attacks to DoS Matter at the transport and network layers [41]. A recent Black Hat talk showcased a delayed DoS attack via CASE message replay [27].

We focus on *(D)DoS attacks exploiting the Matter AL specification*, as they are currently unexplored. But they pose large-scale security and safety risks for the Matter ecosystem. Since they target design vulnerabilities in the AL, they are effective against any Matter underlying layers, such as TCP, UDP, Wi-Fi, or BLE, regardless of the hardware and software details of the Matter victims. Moreover, they are challenging to detect as they employ compliant messages.

The Matter standard and security guideline consider (D)DoS a significant risk [15, 20]. For instance, its "Threats and Countermeasures" Section [20, Sec. 13.7] lists three DDoS threats: i) DDoS of Validator Nodes (T169), DDoS of Validator/Observer Node via READ CLI protocol (T180), and DDoS of Observer Nodes (T182). However, it does not explore fabric-wide AL DDoS attacks. This important point further motivates our work.

3.2 Threat Model

System model. We consider a network including heterogeneous Matter devices. Some devices are commissioned in the fabric, while others are to be commissioned. The devices can assume any Matter role, such as Commissioner, Commissioned, Controller, or Controllee. They can use any Matter-compliant technology, including Wi-Fi, Ethernet, BLE, or Thread, and can be from any Matter-certified brand, such as Apple, Google, and Amazon. They run PASE, CASE, and CASE Resume and the discovery procedures specified in the latest Matter specification (v1.4.2). They are Matter-certified and malware-free.

The devices use commissioning and operational networks, which might be the same physical network (e.g., a smart home WLAN) or different ones (e.g., commissioning over BLE and operations over a WLAN). Commissioned devices can be powered on or off as needed and autonomously establish secure sessions. A user owns the devices in the fabric and can add or remove them at will.

Attacker models. We assume an attacker with *no access* to the fabric (i.e., not commissioned), but with access to the network used by the victim to run Matter's session establishment, and device discovery. For example, the attacker can observe the BLE or Wi-Fi packets used during commissioning discovery and PASE, as well as Wi-Fi packets during operational discovery and CASE (Resume). They can observe these packets either within BLE range or by connecting to the same Wi-Fi network as the victim devices. This is realistic, as the Matter standard explicitly identifies a "malicious device or person with local network access" as a high-severity threat in the Matter specification [20, Sec. 13.7] and Matter's security architecture is intended to provide end-to-end (E2E) protection against "untrusted" local devices.

We define a DoS attack as one that renders a Matter device or the fabric non-functional. For example, a DoSsed device cannot commission devices, be commissioned in the network, or communicate with commissioned devices. Or a DoSsed fabric cannot commission new devices or allow commissioned devices to communicate. If the attacker employs more than one device to DoS a target, we define that attack as a DDoS. The attack vectors are Matter AL status reports, and their goal is to (D)DoS PASE, CASE, and CASE Resume, preventing devices from being commissioned and from exchanging operational data.

To ensure a comprehensive representation of real-world (D)DoS threats, we consider *three attacker models*:

(1) *Rogue Device (RD)*: a device eavesdropping and injecting plaintext packets,
(2) *Spoofer (SP)*: a device spoofing a Matter device using plaintext packets,
(3) *MitM (MI)*: a device with a link-layer MitM position between two devices running Matter at the AL.

None of these attackers is commissioned in the fabric and knows any Matter secrets shared by the victims, including passcodes, private keys, and session keys. Hence, the attacker cannot compromise the confidentiality, integrity, and authenticity of the packets protected by PASE, CASE, and CASE Resume. MI uses known techniques to achieve a link-layer MitM position over Wi-Fi, Ethernet, or BLE. For example, they perform IPv6 Neighbour Discovery Protocol (NDP) spoofing, or BLE link-layer spoofing with bettercap [45].

The attackers have Dolev-Yao network capabilities, e.g., sending, blocking, replaying packets, and establishing one or more sessions with one or more victims. They can craft Matter discovery and session packets and discover Matter devices using mDNS and BLE. The attacker can obtain publicly available information about the victims, such as IP addresses, IDs, and message counters. They do not have physical access to the victim devices and cannot tamper with their hardware or software, e.g., pressing buttons, or infecting them with malware.

Table 2: Matter has five status report messages. The PASE (P), CASE (C), and CASE Resume (C_R) columns count ways a status report can be used in a protocol run. If Sec is ✓, the status report is encrypted and integrity-protected. The MaDoS attacks exploit the four unprotected (✗) status reports.

Status	Description	P	C	C_R	Sec
SES_SUC	Session established	1	1	1	✗
NO_TRUST	No shared trust root	0	1	1	✗
INV_PAR	Invalid parameter	3	3	2	✗
BUSY	Device is busy	1	1	1	✗
CLOSE_SES	Session closed	0	0	0	✓

4 MaDoS Attacks

We present *MaDoS*, three new classes of (D)DoS attacks exploiting Matter AL status report messages. As shown in Figure 1, the attacks employ RD, SP, or MI attacker models to (D)DoS connection initiators and responders, preventing them from completing PASE, CASE, and CASE Resume. Next, we describe the MaDoS vulnerabilities and attack classes in detail.

4.1 Vulnerabilities

The MaDoS attacks are caused by *two new design vulnerabilities* we uncover in the specification of Matter status reports (V1, V2) and one design issue in the Matter commissioning and operational discovery (V3). Status reports are supported by all Matter versions (v1.0–v1.4.2) and are used to send AL status updates. The standard defines five status reports [20, Table 19] (we shorten their names for readability):

(1) Session established (SES_SUC)
(2) No shared trust root (NO_TRUST)
(3) Invalid parameter (INV_PAR)
(4) Device is busy (BUSY)
(5) Close session (CLOSE_SES)

As shown in Table 2, status reports are used within the CASE, CASE Resume, and PASE protocols (introduced in Section 2). The PASE (P), CASE (C), and CASE Resume (C_R) columns count how many times a status report message can appear during a protocol run. SES_SUC may be sent as a P_f or C_f while answering P_3, C_3, or C_{2r}. NO_TRUST may be sent in response to C_1 or C_{1r} during CASE (Resume). INV_PAR may be sent in response to P_r, P_2, P_3, C_1, C_2, C_3, C_{1r}, or C_{2r}. BUSY can be sent in response to to C_1, C_{1r}, or P_r. CLOSE_SES is not used during session establishment. Refer to Tables 11 and 12 in the Appendix for more details about these combinations. As shown in Table 2, Sec column, the first four status reports are sent in plaintext without authentication, whereas CLOSE_SES must be encrypted and integrity-protected. The MaDoS attacks exploit the four unprotected status reports.

We analyzed the Matter standard [20] and SDK [18] and uncovered two design vulnerabilities in the status reports specification:

V1: Unencrypted and unauthenticated status reports can terminate CASE, CASE Resume, and PASE, even if the other AL messages are AEAD-encrypted. This is because status reports are specified in isolation from PASE and CASE (Resume). For instance, their security (Sec in Table 2) does not depend on their usage during PASE or CASE (Resume).

V2: The status reports' semantics enable message confusion. For example, the standard provides contradictory C_f (SigmaFin) definitions: one [20, Tables 19 and 25] states that it is an unencrypted SES_SUC message, while another [20, Figure 15] states that it is an encrypted CASE message. Hence, by the standard, C_f is valid as encrypted *and* unencrypted, allowing an attacker to inject a plaintext termination message to abort a CASE handshake.

Table 3: Enumeration of MaDoS attack instances based on possible Status Reports (S). The Total column counts the number of possible S in response to each message used by MI, SP, or RD attack classes, resulting in a cumulative total of 40 instances, including 16 instances for MI, 8 instances for SP, and 16 instances for RD attack classes.

Message	Possible S	MI	SP	RD	Total
$\mathbf{P}_r$	INV_PAR, BUSY	2	2	2	6
$\mathbf{P}_s$	–	0	0	0	0
$\mathbf{P}_1$	–	0	0	0	0
$\mathbf{P}_2$	INV_PAR	1	0	1	2
$\mathbf{P}_3$	INV_PAR, SES_SUC	2	0	2	4
$\mathbf{C}_1$	INV_PAR, BUSY, NO_TRUST	3	3	3	9
$\mathbf{C}_2$	INV_PAR	1	0	1	2
$\mathbf{C}_3$	INV_PAR, SES_SUC	2	0	2	4
$\mathbf{C}_{1r}$	INV_PAR, BUSY NO_TRUST	3	3	3	9
$\mathbf{C}_{2r}$	INV_PAR, SES_SUC	2	0	2	4
Total		16	8	16	40

V1 and V2 lead to (D)DoS threats, as attackers can exploit unprotected and ambiguous status reports to abort PASE and CASE (Resume), even when victims use a secure communication channel. These vulnerabilities are effective in many attack scenarios, as CASE (Resume) and PASE support multiple status reports that can be sent at different stages of a handshake, as shown in Table 2.

We also find that Matter operational and commissioning discovery procedures are neither authenticated nor encrypted, and label this issue as **V3**. V3 introduces more opportunities for (D)DoS, as SP and MI attackers can craft valid discovery packets to trick a victim into initiating a session with them rather than with a trusted or legitimate device. Moreover, an RD attacker can eavesdrop on discovery packets to acquire information about the target they want to (D)DoS. Next, we present the three MaDoS attack classes (RD, SP, and MI) that exploit V1–V3.

4.2 MaDoS Rogue Device Attacks

As defined in Section 3.2, the RD adversary has access to the local network but is not commissioned into the Matter fabric. Hence, while they cannot decrypt the traffic, they can eavesdrop on the plaintext message headers of the session establishment handshake, extract session metadata (e.g., message counters), and disrupt a

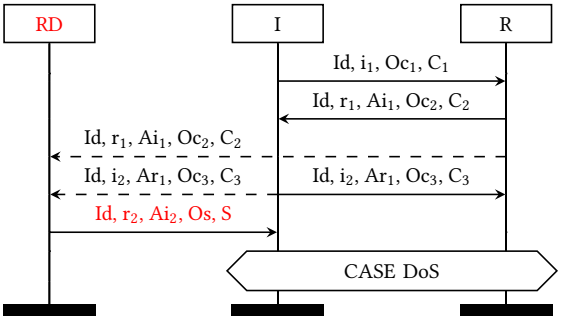


Figure 4: MaDoS RD Attack on CASE Initiator.

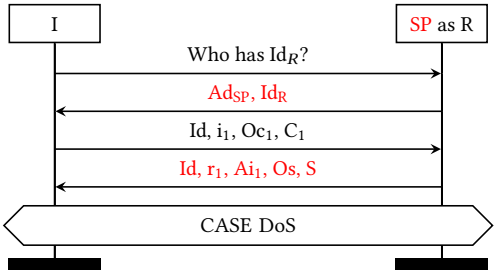


Figure 5: MaDoS SP attack on CASE Initiator.

handshake by injecting unauthenticated, unencrypted status reports by exploiting V1 and V2. In the following scenarios, we demonstrate how this primitive forces a protocol abort against both the Initiator (I) and Responder (R) across CASE, CASE Resume, and PASE.

4.2.1 RD on CASE. Figure 4 shows a CASE RD attack against I. I and R exchange C_1 and C_2, thereby exposing the Id and message counters. The RD adversary sniffs the latter to extract Id, R's message counter (r_1), and I's message counter ($Ai_1 = i_1$) as they are transmitted in plaintext. Then, I sends C_3 to R and expects C_f in response. Once the attacker observes the message, exploiting V1, they inject a plaintext status report including Id, the next expected responder message counter ($r_2 = r_1+1$), the acknowledgment of I's last message ($Ai_2 = Ai_1+1$), O_s, and S.

Upon receiving this status message (S), I terminates the protocol and CASE fails. The attacker can target different stages of the protocol to send S, for example, RD can send `INV_PAR` after C_1 or C_3 to I, or after C_2 to R. By targeting the last stage of the protocol, RD gets a larger attack window. As summarized in Table 3, RD can send `INV_PAR`, `BUSY`, or `NO_TRUST` in response to C_1 or send `INV_PAR` or `SES_SUC` in response to C_3.

Figure 13 (in Appendix C) shows a CASE RD attack against R. As before, I and R exchange the first two CASE messages. The attacker eavesdrops on the latter and injects a plaintext status report message for R containing Id, r_2, Ai_2, O_s, and S. Here, the only valid S would be `INV_PAR`. As a result, R terminates the handshake and CASE fails.

4.2.2 RD on CASE Resume. Figure 15 shows a CASE Resume RD attack against I. I sends a C_{1r} message to R and the attacker eavesdrops on it. Then, the attacker sends a plaintext message to I including Id, r_1 (which is arbitrary), Ai_1, O_s, and S. As a result, I aborts the protocol and DoS CASE Resume. Figure 16 shows a CASE Resume RD attack against R. I and R exchange the first two CASE Resume messages: C_{1r} and C_{2r}. The attacker captures the latter and injects a status report message for R containing Id, r_2, Ai_2, O_s, and S, and R terminates CASE Resume.

4.2.3 RD on PASE. Figure 14 (in Appendix C) shows a PASE RD attack against I. I and R exchange P_1 and P_2 and the attacker eavesdrops on the latter. Then I and R run the three-way PASE handshake, and the attacker captures the last message. Next, the attacker sends I a status report message containing Id, r_3, Ai_3, O_s, and S. I terminates the protocol and PASE is DoSsed. Similarly, as shown in Figure 17, the attacker can DoS PASE by sending a rogue status report to the responder.

4.3 MaDoS Spoofer Attacks

The Spoofer (SP) attack class exploits the lack of authentication in Matter's discovery procedures (V3). The adversary is not part of the fabric, but can observe multicast discovery traffic from the victims. By impersonating a legitimate device during the discovery phase, the attacker can attract session requests and force a protocol abort, exploiting status reports (V1, V2).

Figure 5 illustrates a SP attack on a CASE initiator. I broadcasts a query to discover which address maps to the responder ID (Id_R). The attacker, spoofing R's identity, replies with a message claiming that R is located at the attacker's address (Ad_{SP}). Since operational discovery is unauthenticated, I cannot distinguish the attacker from the legitimate device. I initiates CASE with SP by sending C_1. SP replies with an unauthenticated S such as `INV_PAR`, `BUSY`, or `NO_TRUST`. Upon receiving S, I terminates the session, while the real R is unaware.

The attack methodology detailed above generalizes to CASE Resume and PASE. SP on CASE Resume follows the exact sequence shown in Figure 5; the only distinction is that I transmits a C_{1r} message instead of a C_1. SP on PASE differs from SP on CASE only in the discovery phase parameters. The attacker listens for and responds to mDNS queries containing commissioning-specific service tags (e.g., `_matterc._udp`) rather than operational node IDs. Then they force PASE to terminate using a status report.

4.4 MaDoS Machine-in-the-Middle Attacks

The MI attack class assumes an attacker with a link-layer MitM position between I and R. MI achieves this position by exploiting Matter's insecure link-layer protocols, like poisoning the IPv6 Neighbor Discovery Protocol (NDP) cache. Then, MI terminates the

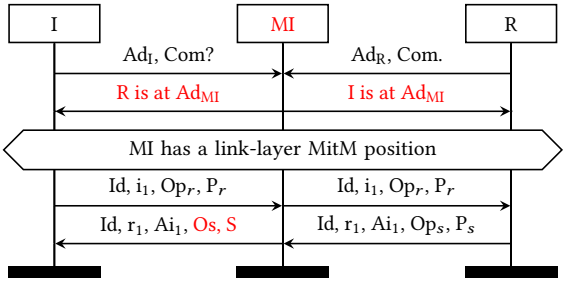


Figure 6: MaDoS MI attack on PASE.

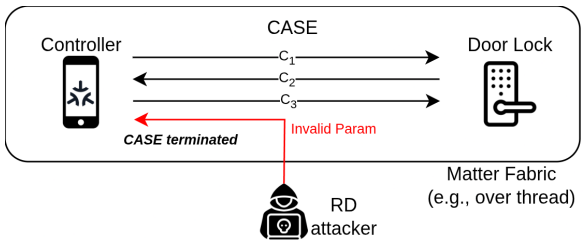


Figure 7: Practical MaDoS RD attack overview. The user tries to lock the door using the controller. The attacker waits for the CASE packets, then injects a plain-text status report to the initiator, causing the target to terminate session establishment and preventing the user from locking their door.

CASE (Resume) and PASE sessions by injecting malicious status reports in plaintext, without decrypting any messages. Next, we detail the MI attacks on PASE, CASE, and CASE Resume.

4.4.1 MI on PASE. Figure 6 shows a MI attack instance against PASE. The attacker captures broadcast Matter commissioning discovery messages from I and R containing their addresses (Ad_I, Ad_R). Then, it mounts a link-layer MitM attack by telling I that R is at Ad_{MI} and R that I is at Ad_{MI}. Then, I initiates PASE by sending P_r to Ad_{MI}. MI forwards the message to R, and R responds with P_s to Ad_{MI}. MI replaces P_s with S and keeps the identifiers and message counters set by R unchanged. I receives S and terminates PASE and MI successfully DoS PASE. As per Table 11, the status report answering P_r can be `INV_PAR` or `BUSY`.

4.4.2 MI on CASE and CASE Resume. Figure 18 shows a MI attack on CASE. MI mounts a link-layer MitM attack as in the prior attack. I starts CASE by sending C_1 to Ad_{MI}, and MI relays it to R. Then, R answers with C_2, sending it to Ad_{MI}, and MI replaces C_2 with an S and sends it to I. For CASE Resume, I starts by sending C_{1r} and MI replaces C_{2r} with an S. As per Table 3, C_1 and C_{1r} can be replied with `NO_TRUST`, `INV_PAR`, or `BUSY`. Next, I receives the status report and terminates CASE.

4.5 MaDoS Impact

The three MaDoS attack classes have critical *security* and *safety* implications on the ecosystem. They can DoS or DDoS one or more devices or an entire fabric. They block the commissioning of new devices and any commissioned devices from operating. For example, they can be used to break into a smart home while preventing communication between motion sensors, surveillance cameras, and their controllers, as in [25], they can prevent a user from locking their door as shown in Figure 7, or to cause physical damage by preventing safety-critical devices, such as fire alarms, from communicating.

The attacks are *scalable* as they are effective regardless of the target devices' Matter version, transport layer, link layer, and hardware and software implementation details, as they exploit design issues in the Matter standard (V1–V3). Furthermore, the attack is efficient as an adversary can maintain a persistent DoS state by injecting a single status report per session attempt, requiring zero cryptographic overhead.

Moreover, they are also *practical* as they do not require the adversary to join a fabric or compromise existing cryptographic keys, passwords, or credentials, but an attacker in the local network. Moreover, they do not need unexpected user interactions. For example, during the attacks on PASE, the user repeatedly fails to commission a victim device after scanning a QR code. We experimentally verified the mentioned impact claims and report our results in Section 6. Next, we explain how we implemented the MaDoS attacks in a toolkit.

5 Implementation

We present `mados`, a novel toolkit for assessing the resilience of Matter devices to DoS attacks. The toolkit enables the reproduction of the MaDoS attack classes (RD, SP, MI) in both simulated environments and real-world deployments. Moreover, it is used to analyze Matter discovery procedures, PASE, CASE, CASE Resume, and status report messages. We provide the toolkit in a repository at https://github.com/AoiKujira/mados.

Figure 8 shows a high-level overview of `mados`, including its control, orchestration, core logic, and infrastructure and drivers components. The toolkit has four modules: `PacketInjector`, `PacketManipulator`, `Spoofer`, and `MitMTester`. The orchestration modules utilize a shared core logic for protocol-specific tasks such as TLV decoding and packet manipulation. These components interact with the target environment via underlying drivers and a patched version of the official Matter SDK. Next, we describe each module in detail.

5.1 PacketInjector

The `PacketInjector` module is responsible for executing the Rogue Device (RD) attack class described in Section 4.2. It tests the resilience of a target against an RD adversary across five scenarios: attacking the PASE Initiator (RDpI), the PASE Responder (RDpR), the CASE Initiator (RDcI), the CASE Responder (RDcR), and the CASE Resume Responder (RDcrR).

This module is implemented in Python and leverages `PacketManipulator` and Scapy to sniff, dissect, and inject Matter packets. The module can operate over Wi-Fi and Ethernet interfaces. It also

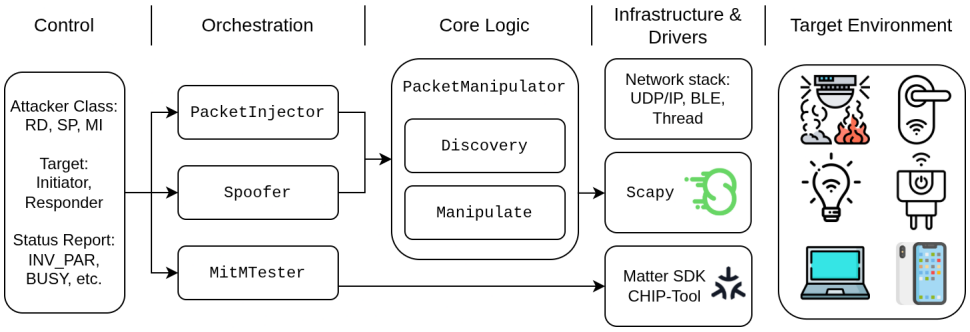


Figure 8: `mados` toolkit architecture and interactions.

enables testing attacks against Thread devices by utilizing a border router. In a standard Matter deployment, a Border Router (BR) bridges the Thread mesh to the local Wi-Fi network. Since Matter relies on end-to-end IPv6 connectivity, `PacketInjector` can target Thread nodes by injecting malicious IPv6/UDP packets at the BR's upstream interface, which the BR then routes into the Thread mesh.

`PacketInjector` operates by monitoring the network for specific session establishment handshakes. Upon launch, the user selects a target protocol (PASE, CASE, or CASE Resume) and an attack instance among five (e.g., RDpI). The script then executes the following logic, using the RDpI scenario (Figure 14) as a reference.

The script filters in UDP packets with Matter source or destination ports (5540). It identifies the opcode and if it equals 0x21 (Op_s), parses the message counter (r_2) and ACK message counter (Ai_1). Then, using the `PacketManipulator` and the extracted values, it constructs a forged StatusReport (S) to be injected in place of a PakeFin message. The forged payload is encapsulated within the same Ethernet/IP/UDP layers, preserving the original addresses of I and R. The module then monitors the channel for the corresponding Pake3 message, and once observed, it transmits the plaintext status report and disrupts PASE.

5.2 PacketManipulator

The `PacketManipulator` module serves as the core parsing and generation engine for `mados`. It addresses a significant gap in the current Matter research ecosystem: the lack of open-source tools to dissect and manipulate Matter packets. Existing tools, such as the reference Matter Wireshark dissector [16] or the popular Scapy Python module [12], either lack support for recent protocol versions (Matter 1.2+) or do not support the protocol entirely.

This module is built on top of Scapy. It consists of two submodules: *discovery* enables testing and tampering with commissionable and operational discovery over mDNS and DNS-SD, and *manipulate* allows crafting standard and custom PASE, CASE (Resume), and status report AL messages. Next, we describe each submodule in more detail.

Discovery submod. The discovery submodule allows an attacker to be discovered as an arbitrary Matter device by legitimate commissioner or commissioned devices. It monitors broadcast discovery packets and extracts devices' hostname, commissioning subtypes, addresses, and IDs. Then it generates spoofed mDNS response packets and transmits them at a configurable transmission frequency. In parallel, it runs a modified Zeroconf [51] service to respond to unicast mDNS queries from the victim Matter devices. Since the original Zeroconf library does not permit duplicate service names and automatically resolves conflicts by appending a suffix, we override `Zeroconf.async_check_service()` to disable this conflict resolution.

Manipulate submod. This submodule handles the complex serialization of Matter application payloads. It introduces a custom Scapy class, `MatterHeader`, to dissect the Message Layer fields defined in Table 1. Implementing a robust encoder/decoder was necessary due to Matter's complex Data Model, which relies on a context-dependent Type-Length-Value (TLV) format. Unlike standard TLV schemes, Matter employs variable-length tags, nested container structures, and control-byte prefixes, rendering general-purpose decoders incompatible. The module is specifically engineered to parse and craft the AL messages required for secure session establishment. It provides full support for generating and dissecting PASE, CASE, and CASE Resume handshake messages, as well as Status Report messages. This capability allows the toolkit to both analyze legitimate traffic flows and construct the specific packet sequences needed to verify MaDoS vulnerabilities.

5.3 Spoofer

`Spoofer` is a Python script that executes the SP attacks from Section 4.3. It utilizes `PacketManipulator` to spoof Matter devices and parse and craft custom Matter packets. While the module is extensible to all session types, our implementation focuses on the SP on CASE attack instances (SP_c), which enables an attacker to impersonate operational devices and isolate valid nodes within the fabric. To reproduce the evaluation scenario illustrated in Figure 5, the module executes a three-stage sequence beginning with reconnaissance. It first monitors the network for mDNS queries to capture identifiers, including the Fabric ID and the Node ID of the victim's intended destination. Using this data, it executes the discovery scripts from `PacketManipulator` to broadcast falsified operational advertisements.

After spoofing operational discovery packets, the script listens on port 5540 for incoming Sigma1 message from I. Upon receiving Sigma1, if it requests an acknowledgment, our script responds with the appropriate StandAloneACK, including the acknowledged message's counter value. Next, depending on the specific attack variation, it sends a StatusReport message with `INV_PAR`, `BUSY`, or

NO_TRUST errors. This creates a state where the Initiator perceives the Responder as alive and reachable but actively rejecting the session with an application error, forcing I to abort the handshake while masking the legitimate device's true availability.

5.4 MitMTester

MitMTester tests five MI attacks on PASE (MIp) and eleven attacks on CASE (MIc) and CASE Resume (MIcr). It patches CHIP-Tool [18] to use it both as a MI victim and MI attacker without conducting an end-to-end MitM attack. It modifies the session-handling logic and inserts a simulated MitM attacker at the application layer. The attacker intercepts and modifies messages as a real node would, and we use them to test target responders and initiators. To implement this feature, we patch CASESession.cpp, CASEServer.cpp, and PASESession.cpp to verify the MaDoS vulnerabilities.

Under the hood, MitMTester hooks the CHIP-Tool send and receive handler functions such as HandleMsgSigma1() and SendMsgSigma1(), to change the packet handling logic in PASESession.cpp for MI on PASE (MIp) and in CASESession.cpp for MI on CASE (MIc) and CASE Resume (MIcr). Depending on the attack instance, it replaces the corresponding handler with code that invokes the SendStatusReport() function with the appropriate error codes. This setup allows testing MIp and MIc utilizing INV_PAR, NO_TRUST, and SES_SUC status report messages.

To test the MIc attacks while sending BUSY, the module hooks the OnMessageReceived() function in CASEServer.cpp. For MIp, CHIP-Tool does not include logic to handle a device already engaged in PASE that responds with a BUSY status report. We extended the CASE-handling logic to support PASE, allowing us to test MIp with the BUSY status report message. We do this by reusing the SendBusyStatusReport() function from the CASEServer.cpp and calling it in the HandlePBKDFParamRequest() function in PASESession.cpp.

6 Evaluation

Next, we describe our experimental setup and present our results.

6.1 Experimental Setup

6.1.1 Devices and ES. To evaluate the three MaDoS attack classes, including 40 attack instances, we select a subset of these instances, thereby enabling us to demonstrate the exploitability of the PASE, CASE, and CASE Resume protocols while maintaining a manageable experimental scope. We evaluate 16 MI, 6 SP, and 5 RD attack instances out of 40 instances, covering PASE, CASE, and CASE Resume. For RD attacks, we select the instances with the largest attack windows.

We test the attack classes across *25 Evaluation Scenarios (ES)* using mados (presented in Section 5). In each scenario, we target a specific victim device/application pair to verify the effectiveness of the attack primitives. These scenarios replicate typical household interactions, such as adding new devices to a fabric or managing existing ones via mobile apps. Our device sample includes 13 physical devices and 4 Matter Admin applications from nine vendors (like Apple, Google, and Amazon). The device selection spans the full range of Matter roles (Commissioner, Commissionee, Controller, Controllee) and supports all link layers (Wi-Fi, Thread, BLE). The targets run Matter versions ranging from v1.0 to v1.4.2.

We test high-end smartphones (e.g., iPhone 16 Pro Max) running the latest versions of iOS Home and Google Home, as well as smart hubs (Nest Hub 2nd gen, Echo Show 5) and a Linux workstation running the standard CHIP-Tool. Our workstation runs Ubuntu 22.04 with a 4.8 GHz Intel i7 CPU and 16 GB RAM. We use ESP32-C6 and ESP32-H2 development boards running the ESP-Matter SDK reference applications (Lighting, Door Lock) to evaluate the ESP stack across different transport layers. Moreover, we assess commercial Matter-certified peripherals, including the LEDVANCE Smart Plug, Nanoleaf Bulb, and Aqara Door Sensor P2.

6.1.2 Network. Our experimental network supports Wi-Fi, Thread, and BLE. Our workstation, running a 2.4 GHz Wi-Fi hotspot [3], serves as the Access Point, allowing us to monitor and inject traffic via standard Wi-Fi interfaces. The victim devices are connected to this network and discover one another via DNS-SD. We utilize an OpenThread Border Router (OTBR), deployed on an ESP32-S3 and ESP32-H2 combination [24]. As described in Section 5.1, the attacks against Thread devices are executed by injecting IPv6 packets at the OTBR's upstream interface, which are then routed into the mesh.

We evaluate MaDoS over BLE using the MitMTester module to simulate MI attacks. We limit our testing to this scope because standard Bluetooth controllers do not allow the injection of raw, arbitrary link-layer frames required for RD attacks without specialized hardware (e.g., SDRs). However, since the vulnerabilities reside in the Matter application layer, MitMTester validates that the protocol state machine remains susceptible to Status Report injection regardless of the underlying transport.

6.1.3 RD, SP, and MI DoS Attacks. We execute RD attacks against PASE and CASE (Resume) using PacketInjector running on the Linux Wi-Fi AP. This position allows the attacker to monitor all fabric traffic and inject malicious Matter frames without requiring an external Wi-Fi adapter in monitor mode. To evaluate the RD attack resilience to race conditions, we automated the execution of evaluation scenarios in which CHIP-Tool served as I (ES14-ES16). For these scenarios, we executed each attack 100 times and recorded the success rate. For scenarios involving manual interaction with mobile apps or physical buttons, we performed 10 tries per scenario.

We test SP on CASE (SPc) with a Linux machine acting as the attacker and running Spoofer. We recall that Spoofer can be configured to reproduce the three SPc attack instances by sending arbitrary status reports in response to Sigma1. We test the BUSY message with different delays, including the minimum and maximum values specified in the standard (i.e., 0 and 0xffff milliseconds).

To test MIp and MIc against R, we run a controller app modified with MitMTester on our Linux workstation. To test attacks against I, we load the modified CHIP-Tool on ESP32 devkits, with the ESP32 acting as both R and MI. For example, to test the attack in Figure 6, we enable the MADDOS_PASE macro in MADDOS.h and the invpar_pase_attack_pbreq constant in PASESession.cpp. Changing the relevant constants enables the other MIp and MIc attack instances.

6.1.4 DDoS Attack. To evaluate our attacks in a distributed manner, i.e., with multiple attack devices targeting the fabric, we conduct a DDoS experiment. As shown in Figure 11, the experiment runs two

Table 4: MIp, MIc, and SPc evaluation. Attacks are effective against the 13 tested evaluation scenarios and the 13 Matter devices on all Matter secure session establishments (PASE, CASE, or CASE Resume) regardless of the victim role (I or R), Matter version (v1.0–v1.4), and link layer (BLE, Wi-Fi, or Thread). ES: Evaluation Scenario, Ver: Matter Version, Thd: Thread, n/a: not applicable.

	Target				MIp			MIc/MIcr		SPcI	
ES	Device	App	Role	Ver	BLE	Wi-Fi	Thd	Wi-Fi	Thd	Wi-Fi	Thd
1	iPhone 16 Pro Max	iOS Home	I	1.4	✓	✓	✓	✓	✓	✓	✓
2	iPhone 13	iOS Home	I	1.4	✓	✓	✓	✓	✓	✓	✓
3	Pixel 8, Nest Hub	Google Home	I	1.4	✓	✓	✓	✓	✓	✓	✓
4	Linux machine	CHIP-Tool	I	1.4	✓	✓	✓	✓	✓	✓	✓
5	Pixel 8, Echo	Alexa	I	1.3	✓	✓	✓	✓	✓	✓	✓
6	ESP32-C6	Light	R	1.4	✓	✓	✓	✓	✓	✓	✓
7	ESP32-C6	Light	R	1.2	✓	✓	✓	✓	✓	✓	✓
8	ESP32-C6	Door lock	R	1.4	✓	✓	✓	✓	✓	✓	✓
9	ESP32-H2	Light	R	1.4	✓	n/a	✓	n/a	✓	n/a	✓
10	ESP32-H2	Door lock	R	1.4	✓	n/a	✓	n/a	✓	n/a	✓
11	Nanoleaf bulb	3.2.0	R	1.0	✓	n/a	n/a	n/a	✓	n/a	✓
12	Aqara Sensor P2	1.0.0.0	R	1.0	✓	n/a	n/a	n/a	✓	n/a	✓
13	LEDVANCE Plug EU	1.1.0	R	1.1	✓	n/a	n/a	✓	n/a	✓	n/a

Table 5: The five evaluated RD attacks are effective across vendors and Matter versions. Our smart bulb and plug can only perform PASE over BLE, so our setup can't test corresponding attacks against them. *: success rate over 10 trials.

	Setup				Attack				
ES	Initiator (I)	Ver	Responder (R)	Ver	RDpI	RDpR	RDcI	RDcR	RDcrR
14	CHIP-Tool	1.4.2	ESP32-C6 Light	1.4.2	100%	100%	100%	100%	100%*
15	CHIP-Tool	1.4.2	Linkind Bulb	1.4	n/a	n/a	100%	100%	100%*
16	CHIP-Tool	1.4.2	LEDVANCE Plug EU	1.1	n/a	n/a	100%	100%	100%*
17	iOS Home	1.4	ESP32-C6 Light	1.4.2	100%*	100%*	100%*	100%*	100%*
18	iOS Home	1.4	Linkind Bulb	1.4	n/a	n/a	100%*	100%*	100%*
19	iOS Home	1.4	LEDVANCE Plug EU	1.1	n/a	n/a	100%*	100%*	100%*
20	Amazon Alexa	1.4.1	ESP32-C6 Light	1.4.2	100%*	100%*	100%*	100%*	100%*
21	Amazon Alexa	1.4.1	Linkind Bulb	1.4	n/a	n/a	100%*	100%*	100%*
22	Amazon Alexa	1.4.1	LEDVANCE Plug EU	1.1	n/a	n/a	100%*	100%*	100%*
23	Google Home	1.5	ESP32-C6 Light	1.4.2	100%*	100%*	100%*	100%*	100%*
24	Google Home	1.5	Linkind Bulb	1.4	n/a	n/a	100%*	100%*	100%*
25	Google Home	1.5	LEDVANCE Plug EU	1.1	n/a	n/a	100%*	100%*	100%*

instances of RDcR and one instance of RDpI, MIp, and SPc against six victims. The victim devices are: i) A Linux machine running CHIP-Tool as I, ii) ESP32-C6 running light app as R_1, iii) ESP32-C6 light bulb as R_2, iv) LEDVANCE Smart Plug EU as R_3, v) Linkind light bulb as R_4, and vi) ESP32-C6 running door-lock app as R_5.

6.2 Experimental Results

We evaluate the efficacy of the RD, SP, and MI attack classes to DoS or DDoS Matter devices and fabrics. The results, summarized in Tables 4 and 5, empirically demonstrate that the MaDoS attacks are practical and impactful on the Matter ecosystem. We expect that any Matter device following the spec up to v1.4.2 is vulnerable to the MaDoS attacks. Next, we describe our results in detail.

6.2.1 MI and SP DoS. As shown in Table 4, the MI and SP attack classes (MIp, MIc, SPc) are effective across all evaluation scenarios (ES) regardless of Matter version, Vendor, or network type. A ✓ in the table means all possible error messages mentioned in Table 3 cause DoS on session establishments. For instance, in ES4 (CHIP-Tool), the attacker consistently replaced the PBKDFParamRes message with `INV_PAR`, forcing an immediate session abort without any race conditions. For the SPc scenarios, we verified the attack logic by isolating the legitimate device and simulating an attacker who has successfully won the discovery race (consistent with the Spoofer attacker model). In these trials, I accepted SP as the legitimate device and initiated CASE. The SP attacker then terminated every attempt by transmitting a malicious S message.

6.2.2 RD DoS and Attack Windows. As shown in Table 5, the RD attacks demonstrated high reliability, achieving a 100% success rate in our automated trials (ES14–ES16). Unlike MI and SP, the RD attacker does not control the traffic flow; it must inject the malicious Status Report (S) such that it reaches the victim before the valid encrypted message (e.g., P_3 or C_3). Although theoretically susceptible to a race condition, our results indicate that triggering the injection upon receipt of the StandAloneACK creates a practical attack window. This optimization effectively neutralizes the race condition, allowing the attacker to inject the malicious payload before the victim runs its cryptographic operations or processes the subsequent valid message.

6.2.3 DDoS. In the DDoS experiment shown in Figure 1, the simultaneous execution of one MI_p, one RD_p, two RD_c, and one SP_c

instance resulted in a complete fabric lockout. The attackers successfully blocked the commissioning of both new devices (R_1 and R_2). MI_p prevented the Controller (I) from pairing with R_1, while RD_p disrupted the commissioning of R_2. The combination of RD_c and SP_c attacks severed all CASE handshakes between I and commissioned devices (R_3, R_4, and R_5). As a result of the DDoS attack, the fabric was unusable. The targeted devices couldn't establish a session as they are continuously terminated by the MaDoS agents, rendering the smart home environment unresponsive to user commands.

7 Related work

Matter Security. In [42], the authors introduce mGPTFuzz, utilizing LLMs to automate test input generation based on Matter's specification. Evaluated on 23 devices, mGPTFuzz discovered 147 new bugs, including three CVEs, outperforming existing IoT fuzzers.

Authors in [11] provide a concise description of the Matter standard, survey research work by academia and industry, and offer insights on addressing current limitations. [44] examines the role of Matter in enhancing smart home interoperability, providing insights from an academic testbed setup.

mDNS DNS-SD. In [8], the authors present a detailed analysis of the mDNS and DNS-SD specifications and discuss their attack surface, including eavesdropping, impersonation, and MitM threats. Other research touched on similar issues [28, 47], including getting mDNS responses from outside the local network using a unicast query [14]. Attacks and defenses on proprietary Apple discovery services using mDNS and DNS-SD, including AirDrop, were presented in [9, 23, 52].

Wi-Fi, Thread, BLE. Significant research efforts have been devoted to Wi-Fi, Thread, and BLE, the three standard wireless technologies that Matter uses. What follows is a sample from the many research works touching these wireless technologies. Borisov showed practical attacks on Wired Equivalent Privacy (WEP) capable of breaking its security guarantees [13]. Vanhoef presented how to force a key reuse attack during the 4-way handshake used by WPA2 [53]. Moreover, he demonstrated a password-partitioning attack on the WPA3 SAE handshake [54].

Regarding Thread, its side-channel resistance to differential EM analysis was assessed in [22]. The Thread commissioning protocol, called J-PAKE, was formally modeled and analyzed [1]. Moreover, in [4], the authors study battery depletion and online password-guessing attacks on a Thread network. MUDThread [30] shows how to integrate the Manufacturer Usage Description (MUD) IoT standard into a network of constrained Thread devices.

Ryan showed how to compromise BLE legacy pairing with a practical key brute-force attack [48]. Antonioli found BLE vulnerable to the KNOB attack, lowering the entropy of the BLE pairing key and brute-forcing it [6]. Moreover, he showed how to compromise BLE pairing via Cross-Transport Key Derivation (CTKD) [7]. Wu demonstrated spoofing attacks against BLE reconnections by disabling encryption [56].

Matter Alternatives. Before the Matter standard, several works analyzed the security and privacy of proprietary and competing smart home systems. These works are orthogonal to ours. Fernandes in [25] discussed security flaws on Samsung SmartThings. Then, Fernandes proposed FlowFence [26], a system based on a finer-grained permission model, to defend against the prior attack.

Wang in [55] discussed ProvThings, a tool to instrument IoT home automation apps to add centralized audit logging capabilities, and tested its efficacy against 26 IoT attacks on the SmartThings platform. Alrawi in [5] presented a scorecard-based methodology to analyze and systematize the security of a home automation system and evaluated it across 45 IoT devices. In [58], the authors compared five smart home platforms and modeled their cloud, IoT, and mobile app components using state machines.

Kumar [38] described a large-scale case study involving 83M devices and 16M households, showing that home automation is widespread across continents and involves heterogeneous devices. Similarly, the IoT Inspector work [31] shows that by collecting an extensive, labeled dataset of smart home communication, one can infer security and privacy issues across vendors. In [2], the authors present a multi-stage attack that uses machine learning methods to infer smart home device data even when it is encrypted.

Proprietary smart home management systems were analyzed in [35] and shown to be uncoordinated and vulnerable to access control attacks. In [57], the authors systematize research on privacy-preserving smart hub devices across 10 industrial proposals and 37 research papers. Mandalari in [43] evaluated eight popular IoT security services, mostly provided via commercial routers, and found that they provide limited security and privacy protection.

8 Conclusion

Matter is a standard technology for smart homes centered around IPv6. (D)DoS attacks against Matter are critical, but research on them remains limited. We address this gap by uncovering MaDoS, a family of three (D)DoS attacks that exploit design flaws in Matter status report messages (V1–V2) and in operational discovery (V3). We demonstrate that RD, SP, and MI attackers can disrupt PASE, CASE, and CASE Resume sessions. These attacks have a large-scale, critical impact on the Matter ecosystem by exploiting design issues in the standard. The attacks require a weak attacker model, i.e., an attacker in the local network who is not commissioned in the fabric and has no access to any key. They are stealthy as they do not trigger user interaction and rely on compliant Matter messages.

We provide `mados` to test MaDoS attacks using a low-cost, reproducible setup. Our evaluation across 25 evaluation scenarios involving 13 devices from major vendors (e.g., Amazon, Google, Apple) confirms that MaDoS is effective regardless of the target's Matter version, role, or link layer. We propose two countermeasures to fix the MaDoS attacks and their root causes by design. F1 provides a dedicated and unambiguous specification for PASE and CASE (Resume) status reports to fix V1 and V2. F2 adds authentication to operational discovery using available credentials and partially addresses V3.

References

[1] Michel Abdalla, Fabrice Benhamouda, and Philip MacKenzie. 2015. Security of the J-PAKE password-authenticated key exchange protocol. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 571–587.

[2] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and Selcuk Uluagac. 2020. Peek-a-boo: I see your smart home activities, even encrypted!. In *Proceedings of*

the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks. 207–218.
[3] Lakindu Akash. 2025. Feature-rich wifi hotspot creator for Linux. https://github.com/lakinduakash/linux-wifi-hotspot.
[4] Dimitrios-Georgios Akestoridis, Vyas Sekar, and Patrick Tague. 2022. 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*. 233–244.
[5] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security Evaluation of Home-Based IoT Deployments. In *2019 IEEE symposium on security and privacy (sp)*. IEEE, 1362–1380.
[6] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. 2020. Key Negotiation Downgrade Attacks on Bluetooth and Bluetooth Low Energy. In *ACM Transactions on Privacy and Security (TOPS)*, Vol. 23. Association for Computing Machinery (ACM), New York, NY, USA, 1–28. doi:10.1145/3394497
[7] Daniele Antonioli, Nils Ole Tippenhauer, Kasper Rasmussen, and Mathias Payer. 2022. BLURtooth: Exploiting Cross-Transport Key Derivation in Bluetooth Classic and Bluetooth Low Energy. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS)*.
[8] Antonios Atlasis. 2017. An Attack-in-Depth Analysis of Multicast DNS and DNS Service Discovery. https://sfc6326dbff511243.jimcontent.com/.
[9] Xiaolong Bai, Luyi Xing, Nan Zhang, XiaoFeng Wang, Xiaojing Liao, Tongxin Li, and Shi-Min Hu. 2016. Staying secure and unprepared: Understanding and mitigating the security risks of Apple zeroconf. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 655–674.
[10] Swift Beacon. 2025. 40+ Smart Home Technology Facts and Statistics in 2022-2024. https://swiftbeacon.com/blogs/news/smart-home-technology-statistics.
[11] Dimitri Belli, Paolo Barsocchi, and Filippo Palumbo. 2024. Connectivity Standards Alliance Matter: State of the art and opportunities. *Internet of Things* 25 (2024), 101005.
[12] Philippe Biondi. 2003–present. Scapy. https://scapy.net. Accessed: 2025-04-22.
[13] Nikita Borisov, Ian Goldberg, and David Wagner. 2001. Intercepting mobile communications: The insecurity of 802.11. In *Proceedings of the 7th annual international conference on Mobile computing and networking*. 180–189.
[14] chadillac. 2013. MDNS Recon. https://github.com/chadillac/mdns_recon.git.
[15] Connectivity Standards Alliance (CSA). 2022. Matter Security and Privacy Fundamentals. https://csa-iot.org/wp-content/uploads/2022/03/Matter_Security_and_Privacy_WP_March-2022.pdf.
[16] Connectivity Standards Alliance (CSA). 2023. Wireshark plugin for parsing Matter protocol messages. Status: Experimental. https://github.com/project-chip/matter-dissector.
[17] Connectivity Standards Alliance (CSA). 2024. Matter Official Documentation. https://project-chip.github.io/connectedhomeip-doc/index.html.
[18] Connectivity Standards Alliance (CSA). 2024. Matter Official SDK (connectedhomeip). https://github.com/project-chip/connectedhomeip.
[19] Connectivity Standards Alliance (CSA). 2024. Matter: The Foundation for Connected Things. https://csa-iot.org/all-solutions/matter/.
[20] Connectivity Standards Alliance (CSA). 2025. Matter Specifications 1.4.2. https://csa-iot.org/developer-resource/specifications-download-request/.
[21] Schutzwerk DE. 2023. Security Considerations for Matter Developers. https://www.schutzwerk.com/en/blog/matter-security-considerations/.
[22] Daniel Dinu and Ilya Kizhvatov. 2018. EM analysis in the IoT context: Lessons learned from an attack on Thread. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), 73–97.
[23] Mark Dowd. 2015. Malwairdrop: compromising iDevices via AirDrop. *Ruxcon, October* (2015).
[24] espressif. 2025. Espressif OpenThread Border Router example setup. https://github.com/espressif/esp-idf/tree/master/examples/openthread/ot_br.
[25] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security analysis of emerging smart home applications. In *2016 IEEE symposium on security and privacy (SP)*. IEEE, 636–654.
[26] Earlence Fernandes, Justin Paupore, Amir Rahmati, Daniel Simionato, Mauro Conti, and Atul Prakash. 2016. FlowFence: Practical data protection for emerging IoT application frameworks. In *25th USENIX security symposium (USENIX Security 16)*. 531–548.
[27] Béla Genge and Ioan Pădurean. 2024. Breaking Matter: vulnerabilities in the Matter protocol. https://i.blackhat.com/EU-24/Presentations/EU-24-Genge-BreakingMatterVulnerabiltiesInTheMatterProtocol-wp.pdf.
[28] GNUcitizen. 2008. Name (mDNS) Poisoning Attacks Inside The LAN. https://www.gnucitizen.org/blog/name-mdns-poisoning-attacks-inside-the-lan/.
[29] Google. 2025. OpenThread: an open-source implementation of the Thread networking protocol. https://github.com/openthread/openthread.
[30] Luke Houben, Thijs Terhoeve, and Savio Sciancalepore. 2024. MUDThread: Securing Constrained IoT Networks via Manufacturer Usage Descriptions. *IEEE Communications Magazine* (2024).
[31] Danny Yuxing Huang, Noah Apthorpe, Frank Li, Gunes Acar, and Nick Feamster. 2020. Iot inspector: Crowdsourcing labeled network traffic from smart home devices at scale. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 2 (2020), 1–21.
[32] IETF. 2025. DNS-Based Service Discovery. https://www.rfc-editor.org/rfc/rfc6763.
[33] IETF. 2025. Multicast DNS. https://www.rfc-editor.org/rfc/rfc6762.
[34] IETF. 2025. SPAKE2+, an Augmented PAKE. https://www.ietf.org/archive/id/draft-bar-cfrg-spake2plus-01.html.
[35] Yan Jia, Bin Yuan, Luyi Xing, Dongfang Zhao, Yifan Zhang, XiaoFeng Wang, Yijing Liu, Kaimin Zheng, Peyton Crnjak, Yuqing Zhang, et al. 2021. Who's In Control? On Security Risks of Disjointed IoT Device Management Channels. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 1289–1305.
[36] Felix Klement, Emily Vorderwülbeke, and Stefan Katzenbeisser. 2023. One Standard to Rule Them All? Assessing the Disruptive Potential of Jamming Attacks on Matter Networks. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, 1–6.
[37] Hugo Krawczyk. 2003. SIGMA: The 'SIGn-and-MAc'approach to authenticated Diffie-Hellman and its use in the IKE protocols. In *Annual international cryptology conference*. Springer, 400–425.
[38] Deepak Kumar, Kelly Shen, Benton Case, Deepali Garg, Galina Alperovich, Dmitry Kuznetsov, Rajarshi Gupta, and Zakir Durumeric. 2019. All things considered: an analysis of IoT devices on home networks. In *28th USENIX security symposium (USENIX Security 19)*. 1169–1185.
[39] Song Liao, Jingwen Yan, and Long Cheng. 2024. WIP: Hidden Hub Eavesdropping Attack in Matter-enabled Smart Home Systems. In *Workshop on Security and Privacy in Standardized IoT (SDIoTSec)*.
[40] Melissa Loos. 2023. Security analysis of the Matter protocol.
[41] Andrew Losty and Anna Maria Mandalari. 2024. Poster: An Investigation of Matter Smart Home Mechanisms to Mitigate Denial-of-Service (DoS) Attacks. In *International Conference on Embedded Wireless Systems and Networks*, Vol. 1. ACM.
[42] Xiaoyue Ma, Lannan Luo, and Qiang Zeng. 2024. 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)*. 4783–4800.
[43] Anna Maria Mandalari, Hamed Haddadi, Daniel J Dubois, and David Choffnes. 2023. Protected or porous: a comparative analysis of threat detection capability of IoT safeguards. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3061–3078.
[44] Ravindra Mangar, Jingyu Qian, Wondimu Zegeye, Abdulrahman AlRabah, Ben Civjan, Shalni Sundram, Sam Yuan, Carl A Gunter, Mounib Khanafer, Kevin Kornegay, et al. 2024. Designing and Evaluating a Testbed for the Matter Protocol: Insights into User Experience. In *Workshop on Security and Privacy in Standardized IoT (SDIoTSec)*.
[45] Simone Margaritelli. 2025. Bettercap Homepage. https://www.bettercap.org/.
[46] Nozomi. 2024. Trust Matters: Uncovering Vulnerabilities in the Matter Protocol. https://www.nozominetworks.com/blog/trust-matters-uncovering-vulnerabilities-in-the-matter-protocol.
[47] G. Pickett. 2011. Port Scanning Without Sending Packets. https://defcon.org/images/defcon-19/dc-19-presentations/Pickett/DEFCON-19-Pickett-Port-Scanning-Without-Packets.pdf.
[48] Mike Ryan. 2013. Bluetooth: With low energy comes low security. In *7th USENIX Workshop on offensive technologies (WOOT 13)*.
[49] Narmeen Shafqat and Aanjhan Ranganathan. 2024. Seamlessly Insecure: Uncovering Outsider Access Risks in AiDot-Controlled Matter Devices. In *2024 IEEE Security and Privacy Workshops (SPW)*. IEEE, 281–288.
[50] Kumar Shashwat, Francis Hahn, Xinming Ou, and Anoop Singhal. 2023. Security Analysis of Trust on the Controller in the Matter Protocol Specification. In *2023 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 1–6.
[51] Jakub Stasiak and contributors. 2024. python-zeroconf: A Python implementation of multicast DNS-SD. https://github.com/jstasiak/python-zeroconf. Python library for Multicast DNS Service Discovery.
[52] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir, and Matthias Hollick. 2019. A billion open interfaces for eve and mallory: MitM, DoS, and tracking attacks on iOS and macOS through Apple Wireless Direct Link. In *28th USENIX Security Symposium (USENIX Security 19)*. 37–54.
[53] Mathy Vanhoef and Frank Piessens. 2017. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 1313–1328.
[54] Mathy Vanhoef and Eyal Ronen. 2020. Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 517–533.
[55] Qi Wang, Wajih Ul Hassan, Adam Bates, and Carl Gunter. 2018. Fear and logging in the internet of things. In *Network and Distributed Systems Symposium*.
[56] Jianliang Wu, Yuhong Nan, Vireshwar Kumar, Dave Jing Tian, Antonio Bianchi, Mathias Payer, and Dongyan Xu. 2020. BLESA: Spoofing attacks against reconnections in Bluetooth Low Energy. In *14th USENIX Workshop on Offensive Technologies (WOOT 20)*.

[57] Igor Zavalyshyn, Axel Legay, Annanda Rath, and Etienne Rivière. 2022. SoK: Privacy-enhancing smart home hubs. *Proceedings on Privacy Enhancing Technologies* (2022).
[58] Wei Zhou, Yan Jia, Yao Yao, Lipeng Zhu, Le Guan, Yuhang Mao, Peng Liu, and Yuqing Zhang. 2019. Discovering and understanding the security hazards in the interactions between IoT devices, mobile apps, and clouds on smart home platforms. In *28th USENIX security symposium (USENIX security 19)*. 1133–1150.

A MaDoS Fixes

We discuss *two fixes (F1, F2)* to address by design the three MaDoS attack classes and their three root causes (V1–V3). F1 provides a specification for status reports compatible with CASE (Resume) and PASE, and addresses V1 and V2. F2 authenticates operational discovery with available credentials to partially address V3. The fixes require a minor amendment to the Matter standard and can be shipped with a secure OTA firmware update.

A.1 Root Cause Analysis

Table 6: Mapping MaDoS attacks and vulnerabilities.

MaDoS Attack	V1	V2	V3
RD on PASE	✓	✓	✗
RD on CASE	✓	✓	✗
RD on CASE Resume	✓	✓	✗
SP on PASE	✓	✓	✓
SP on CASE	✓	✓	✓
SP on CASE Resume	✓	✓	✓
MI on PASE	✓	✓	✗
MI on CASE	✓	✓	✗
MI on CASE Resume	✓	✓	✗

We conducted a root cause analysis of MaDoS and mapped the attacks and vulnerabilities, as shown in Table 6. As shown in the Table, all attacks exploit V1 and V2 to send (unencrypted) status reports to (D)DoS PASE and CASE (Resume). The SP attack class exploits V3 by spoofing R to I using unauthenticated operational discovery messages. Based on this analysis, we created F1 and F2 as follows.

A.2 Status Reports Specifications (F1)

PASE Status Reports. We propose eliminating all status report messages during PASE except for `SES_SUC`, which we require to be encrypted with AES-CCM. This strategy prevents MaDoS attacks against PASE because the attacker cannot send valid, unauthenticated status reports during PASE. PakeFin, which is encoded as `SES_SUC`, should be encrypted with Kr (R2IKey) session key derived after Pake3.

We eliminate `INV_PAR` and `BUSY` as they pose security risks and are not essential to PASE. `BUSY` can be replaced with timeout-based retry logic on the initiator side, simplifying the protocol and reducing the attack surface. In the case of an `INV_PAR` response to PBKDFParamReq, the initiator is already capable of validating the parameters it sends, making such a response redundant. For other uses of `INV_PAR` (i.e., in response to Pake2 and Pake3), the failure should be implicitly detected via a timeout or an inability to complete the handshake, without revealing the responder state through an explicit message.

CASE (Resume) Status Reports. The CASE (Resume) `BUSY` status report should be replaced by timeouts or retry strategies at the initiator side. The three remaining insecure status reports (i.e., `SES_SUC`, `NO_TRUST`, and `INV_PAR`) should be AEAD to prevent RD, MI, and SP against CASE and CASE Resume. Encryption can use CASE intermediate security keys (Kc_2, Kc_3). For example, `INV_PAR` in response to Sigma2 can be encrypted by the initiator with Kc_2.

To send a status report after Sigma1 the responder shall generate Kc_2 and encrypt the status report message using such key. Then it can include its ephemeral public key and a random number in the response, enabling the initiator to compute the shared secret, derive the same key, and decrypt the status report.

A.3 Authenticated Operational Discovery (F2)

F2 authenticates Matter operational discovery using available credentials and secrets to fix SPc and partially address V3. The fix mandates the commissioner/controller to be a trusted distributor of Node Operational Certificate (NOC) public keys to every commissioned device. Moreover, it forces every node to append a signature to every operational discovery message computed from a valid NOC private key. In this setup, the adversaries cannot spoof an operational discovery message to another node because they cannot produce a valid signature. Even if the attacker joins the fabric in some way, they still can not sign an operational discovery message as another node because they do not know their NOC private key.

SP to spoof R would have to forge a discovery message containing Ad_{SP}, Id_R, and Sign(M, k2), where k2 is R's NOC private key unknown to SP. Sign(M, k2) is a digital signature computed over the operational discovery message. Initiator (I) can verify the signature using K2, R's NOC public key. K2 is securely distributed to I by the commissioner/controller using CASE either when I joins the fabric or when R joins the fabric. The attacker can obtain K2 (R's public key), but still cannot convince the controller that a signature made with an attacker-controlled private key belongs to R, as the controller is the trusted public key distributor and does not distribute AT's public key.

F2 entails minimal changes to the Matter standard. The mDNS discovery message should include a signature field, and the signature can be computed using available primitives, like ECDSA with SHA256. To distribute the public keys, the commissioning flow should include one extra message during CASE to transmit them to the commissioned device. Moreover, the controller can establish CASE sessions with the commissioned devices to share new valid public keys when a trusted device joins the fabric. F2 scales across fabrics as NOCs differ among devices and fabrics. Hence, the signatures differ by device and fabric.

B Responsible Disclosure Details

We responsibly disclosed our findings and maintained active communication with the Connectivity Standards Alliance Security team (CSA) for over eight months. We initiated the responsible disclosure

process via the official CSA reporting portal [1] on May 29, 2025. After a follow-up on August 19, the CSA provided technical comments on October 9 (over 4 months after our initial disclosure). After several email exchanges (Oct 13, Oct 29, Oct 30), we requested an official statement on the vulnerabilities. The CSA informed us on December 4 that they were "not in a position to provide an official statement on this report". On December 14, we shared significant new findings regarding the RD attacker class and are currently awaiting a response. At the time of publication, we have not received a response to our latest findings, and we are unaware of any formal mitigation plan currently in place by the CSA.

C Extra Figures, Tables, and MSC

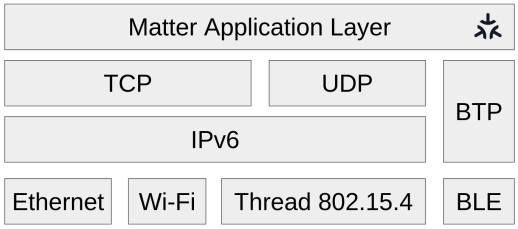


Figure 9: The Matter stack is centered around IPv6 and supports popular transport and link layers. This work focuses on the Matter Application Layer.

Figure 10: We evaluate MaDoS attacks against 13 devices from nine vendors.

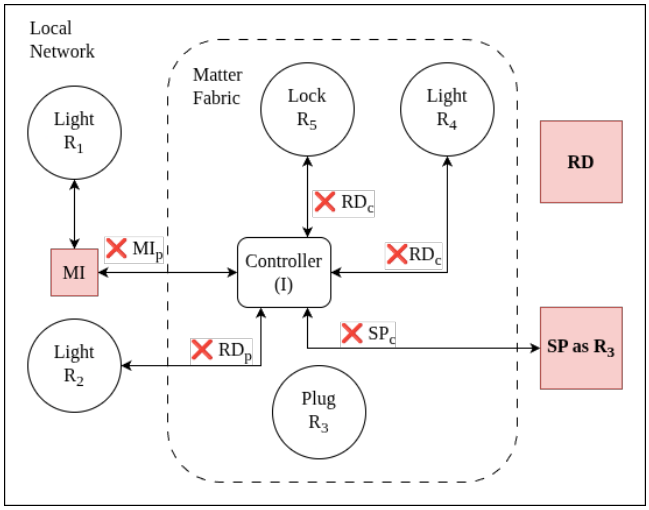


Figure 11: Experimental testbed for the MaDoS fabric-wide DDoS evaluation. The topology comprises five victim nodes: a controller (I), two commissionee (R_1 and R_2), and three commissioned devices (R_3, R_4, and R_5). We execute five simultaneous attack instances to disrupt all commissioning and operational traffic. RD prevents I from establishing a session with R_2, R_4, and R_5. SP is spoofing R_3 and runs SP_c whenever I tries to initiate a session with R_3 and MI is running MI_p and prevents I from commissioning R_1.

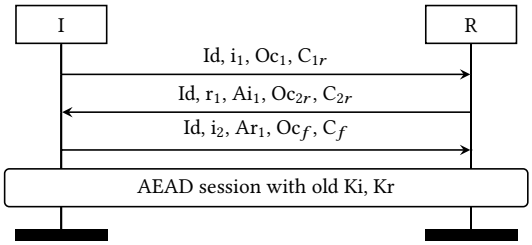


Figure 12: CASE Resume secure session. C_{1r} uses the same CASE opcode as C_1, while C_{2r} employs a dedicated one. The three CASE messages are neither encrypted nor authenticated. I and R reuse Ki and Kr from a previous CASE session.

[1] https://csa-iot.org/vulnerability-reporting/

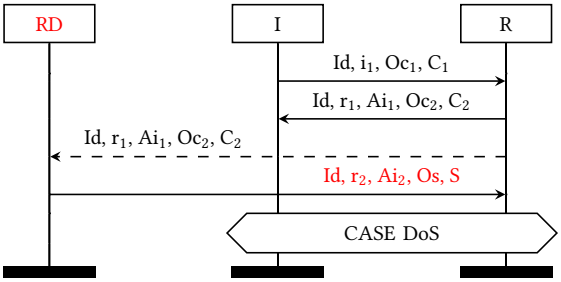


Figure 13: MaDoS RD Attack on CASE Responder.

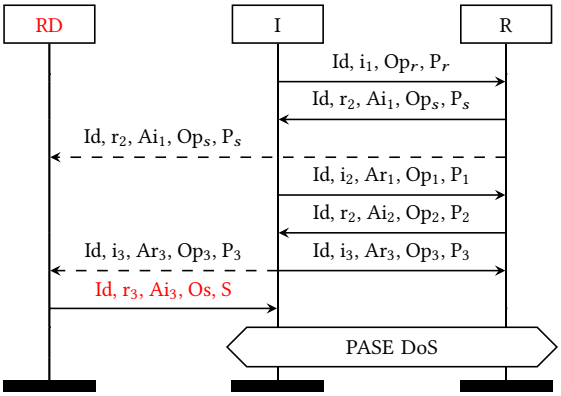


Figure 14: MaDoS RD Attack on PASE Initiator.

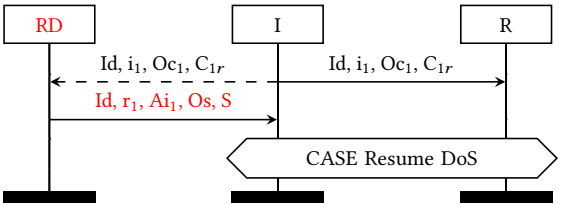


Figure 15: MaDoS RD Attack on CASE Resume Initiator. r_1 is arbitrary.

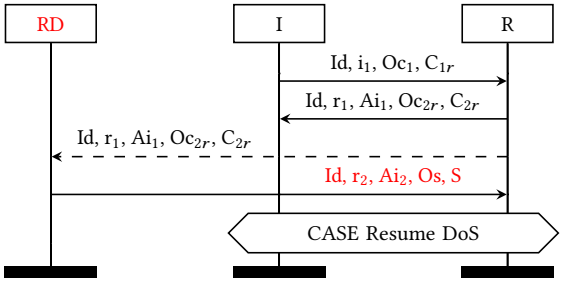


Figure 16: MaDoS RD Attack on CASE Resume Responder.

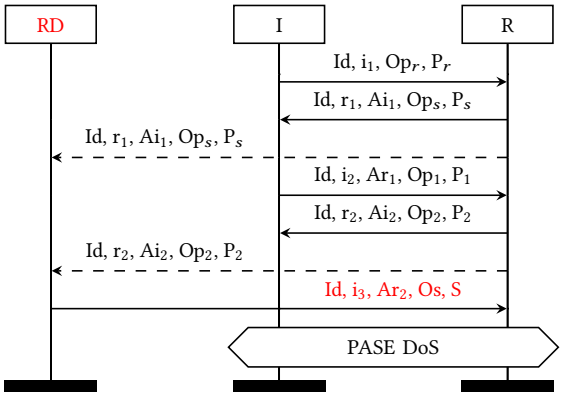


Figure 17: MaDoS RD Attack on PASE Responder.

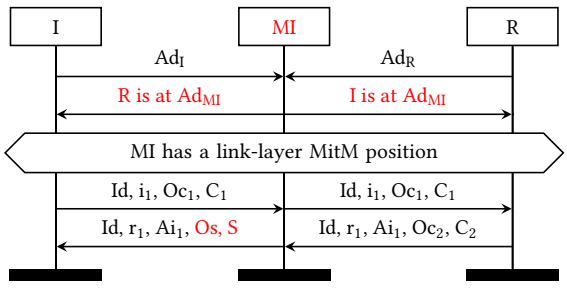


Figure 18: MaDoS MI attack on CASE.

Table 10: Protocol layer header fields. B: Bytes, opt: optional, var: variable.

Name	B	Description
Exchange flags	1	ack, …
Protocol opcode	1	opcode
Exchange ID opcode	2	unique session number
Protocol Vendor ID	2	opt, proprietary
Protocol ID	2	0x00 for PASE, CASE
Ack Counter	4	opt, rec message counter
Secured Extension	var	opt

Table 11: Status reports in answer to PASE messages. ✓ in column X row Y means that replying to message X with status report Y is possible.

Message	P_r	P_s	P_1	P_2	P_3	P_f	Total
SES_SUC	✗	✗	✗	✗	✓	✗	1
NO_TRUST	✗	✗	✗	✗	✗	✗	0
INV_PAR	✓	✗	✗	✓	✓	✗	3
BUSY	✓	✗	✗	✗	✗	✗	1
CLOSE_SES	✗	✗	✗	✗	✗	✗	0
Total	2	0	0	1	2	0	5

Table 12: Status reports in answer to CASE and CASE Resume messages. ✓ in column X row Y means that replying to message X with status report Y is possible.

Message	C_1	C_2	C_3	C_{1r}	C_{2r}	C_f	Total
SES_SUC	✗	✗	✓	✗	✓	✗	2
NO_TRUST	✓	✗	✗	✓	✗	✗	2
INV_PAR	✓	✓	✓	✓	✓	✗	5
BUSY	✓	✗	✗	✓	✗	✗	2
CLOSE_SES	✗	✗	✗	✗	✗	✗	0
Total	3	1	2	3	2	0	11

Table 7: PASE Payloads. I: Initiator, R: Responder.

Id	Description	Sender
P_r	PBKDFParamRequest	I
P_s	PBKDFParamResponse	R
P_1	Pake1	I
P_2	Pake2	R
P_3	Pake3	I
P_f	PakeFin	R

Table 8: CASE (Resume) Payloads. I: Initiator, R: Responder.

Id	Description	Sender
C_1	Sigma1	I
C_2	Sigma2	R
C_{2a}	Sigma2 plaintext head	R
C_{2b}	Sigma2 encrypted foot	R
C_3	Sigma3	I
C_f	SigmaFin	I, R
C_{1r}	Sigma1 with Resum.	I
C_{2r}	Sigma2 with Resum.	R

Table 9: Paper notation.

Id	Description
I	Initiator
R	Responder
Ad	IP, IPv6 or BLE address
Id	Concat of exc, source, and dest IDs
i	Msg counter from I
r	Msg counter from R
Ar	Ack msg counter from I
Ai	Ack msg counter from R
Op	PASE opcode, e.g., Op_1 for Pake1
P	PASE payload, e.g., P_1 is Pake1
O_c	CASE opcode, e.g., O_{c1} for Sigma1
C	CASE payload, e.g., C_1 is Sigma1
Kc	CASE intermediate key
Ki	I to R session key, i.e., I2RKey
Kr	R to I session key, i.e., R2IKey
O_s	Status report opcode
S	Status report payload