Secure management of IoT devices lifecycle through identities, trust and distributed ledgers

D4.2 Secure deployment and registration of IoT devices

Document Summary Information

<table>
<thead>
<tr>
<th>Grant Agreement No</th>
<th>101020416</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acronym</td>
<td>ERATOSTHENES</td>
</tr>
<tr>
<td>Full Title</td>
<td>Secure management of IoT devices lifecycle through identities, trust and distributed ledgers</td>
</tr>
<tr>
<td>Start Date</td>
<td>01/10/2021</td>
</tr>
<tr>
<td>Duration</td>
<td>42 months</td>
</tr>
<tr>
<td>Project URL</td>
<td><a href="http://www.eratosthenes-project.eu">www.eratosthenes-project.eu</a></td>
</tr>
<tr>
<td>Deliverable</td>
<td>Secure deployment and registration of IoT devices</td>
</tr>
<tr>
<td>Work Package</td>
<td>WP4</td>
</tr>
<tr>
<td>Contractual due date</td>
<td>31/03/2023</td>
</tr>
<tr>
<td>Actual submission date</td>
<td>27/03/2023</td>
</tr>
<tr>
<td>Nature</td>
<td>Other</td>
</tr>
<tr>
<td>Dissemination Level</td>
<td>PU</td>
</tr>
<tr>
<td>Responsible author</td>
<td>Jesús García Rodríguez (UMU)</td>
</tr>
<tr>
<td>Lead Beneficiary</td>
<td>UMU</td>
</tr>
<tr>
<td>Authors</td>
<td>Jesús García Rodríguez, Agustín Marín Frutos, Juan Francisco Martínez Gil (UMU)</td>
</tr>
<tr>
<td>Internal reviewers</td>
<td>Hui Song (SINTEF) and Rohit Bohara (DWG)</td>
</tr>
</tbody>
</table>

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Grant Agreement No 101020416.
Revision history (including peer reviewing & quality control)

<table>
<thead>
<tr>
<th>Version</th>
<th>Issue Date</th>
<th>% Complete</th>
<th>Changes</th>
<th>Contributor(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>27/01/2023</td>
<td>5%</td>
<td>Initial table of contents</td>
<td>Jesús García Rodríguez (UMU)</td>
</tr>
<tr>
<td>V0.2</td>
<td>31/01/2023</td>
<td>6%</td>
<td>Revised table of contents and outline</td>
<td>Jesús García Rodríguez, Antonio Skarmeta Gómez (UMU)</td>
</tr>
<tr>
<td>V0.3</td>
<td>15/02/2023</td>
<td>15%</td>
<td>Initial content in sections 3 and 4</td>
<td>Jesús García Rodríguez, Agustín Marín Frutos (UMU)</td>
</tr>
<tr>
<td>V0.4</td>
<td>20/02/2023</td>
<td>30%</td>
<td>Additional content in sections 3 and 4, figures</td>
<td>Jesús García Rodríguez (UMU)</td>
</tr>
<tr>
<td>V0.5</td>
<td>22/02/2023</td>
<td>50%</td>
<td>Finalize sections 3.1, 4.1, 4.3.</td>
<td>Jesús García Rodríguez, Agustín Marín Frutos (UMU)</td>
</tr>
<tr>
<td>V0.6</td>
<td>24/02/2023</td>
<td>80%</td>
<td>Finalize section 3 and 4, section 5 content</td>
<td>Jesús García Rodríguez, Agustín Marín Frutos, Juan Francisco Martínez Gil (UMU)</td>
</tr>
<tr>
<td>V1.0</td>
<td>28/02/2023</td>
<td>90%</td>
<td>Final formatting, content refinement and content in sections 1, 2 and 6.</td>
<td>Jesús García Rodríguez, Agustín Marín Frutos, Juan Francisco Martínez Gil (UMU)</td>
</tr>
<tr>
<td>V1.1</td>
<td>20/03/2023</td>
<td>100%</td>
<td>Review and comments addressed</td>
<td>Hui Song (SINTEF), Rohit Bohara (DWG), Jesús García Rodríguez (UMU)</td>
</tr>
</tbody>
</table>

Disclaimer

The content of the publication herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services.

While the information contained in the documents is believed to be accurate, the authors(s) or any other participant in the ERATOSTHENES consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Neither the ERATOSTHENES Consortium nor any of its members, their officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

Without derogating from the generality of the foregoing neither the ERATOSTHENES Consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

Copyright message

© ERATOSTHENES Consortium, 2020-2025. This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.
Table of Contents

1 Executive Summary ..................................................................................................................................6
2 Introduction..................................................................................................................................................7
  2.1 Mapping ERATOSTHENES Outputs .................................................................................................7
  2.2 Deliverable Overview and Report Structure .....................................................................................9
3 Architecture Orientation and Industrial Requirements ........................................................................11
  3.1 Architectural Positioning and design decisions.................................................................................11
  3.2 Business, Industrial Positioning and End-User Requirements .........................................................12
  3.3 Methodology ......................................................................................................................................15
    3.3.1 Implementation methodology ......................................................................................................15
    3.3.2 Implementation design and development decisions ..................................................................16
    3.3.3 Implementation Issues and risks ...............................................................................................16
  3.4 Code Availability ...............................................................................................................................16
4 Lifecycle management through MUD files ..........................................................................................17
  4.1 Manufacturer Usage Description (MUD) .........................................................................................17
    4.1.1 MUD standard ...........................................................................................................................17
    4.1.2 Threat MUD proposal ...............................................................................................................19
    4.1.3 Shortcomings and extensions to MUD ......................................................................................19
    4.1.4 MUD in ERATOSHTENES .......................................................................................................20
  4.2 Bootstrapping and enrolment in ERATOSTHENES .........................................................................22
    4.2.1 Components ..............................................................................................................................22
    4.2.2 Design choices ............................................................................................................................22
    4.2.3 Integration in ERATOSTHENES ..............................................................................................23
    4.2.4 Interfaces and deployment ........................................................................................................23
    4.2.5 Known future changes ...............................................................................................................25
  4.3 Device protection during operation .................................................................................................25
    4.3.1 Threat monitoring and detection interactions ...........................................................................25
    4.3.2 Security information sharing .....................................................................................................26
    4.3.3 Dynamic security assessment ...................................................................................................26
5 Research and Scientific Innovation ......................................................................................................27
6 Conclusions................................................................................................................................................29
7 References ..................................................................................................................................................30

List of Figures
Figure 1 ERATOSHTENES instantiated architecture (modified) ..............................................................11
Figure 2 MUD file example .....................................................................................................................18
Figure 3 MUD standard architecture, [3] ..............................................................................................18
Figure 4 Instantiation of the MUD components in ERATOSHTENES ..................................................21
Figure 5 PoC enrolment flow extract .....................................................................................................23

List of Tables
Table 1: Adherence to ERATOSTHENES GA Deliverable & Tasks Descriptions ...................................7
Table 2 ERATOSTHENES requirements related to the MUD components ............................................15
## Glossary of terms and abbreviations used

<table>
<thead>
<tr>
<th>Abbreviation / Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>Authentication, Authorization and Accounting</td>
</tr>
<tr>
<td>ACL</td>
<td>Access Control List</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CERT</td>
<td>Computer Emergency Response Team</td>
</tr>
<tr>
<td>CoAP</td>
<td>Constrained Application Protocol</td>
</tr>
<tr>
<td>CSIRT</td>
<td>Computer Security Incident Response Team</td>
</tr>
<tr>
<td>CTI</td>
<td>Cyber-Threat Information</td>
</tr>
<tr>
<td>CVE</td>
<td>Common Vulnerabilities and Exposures</td>
</tr>
<tr>
<td>CVSS</td>
<td>Common Vulnerability Scoring System</td>
</tr>
<tr>
<td>(D)TLS</td>
<td>Datagram Transport Layer Security</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>DID</td>
<td>Decentralized Identifier</td>
</tr>
<tr>
<td>DLT</td>
<td>Distributed Ledger Technology</td>
</tr>
<tr>
<td>EAP</td>
<td>Extensible Authentication Protocol</td>
</tr>
<tr>
<td>ENISA</td>
<td>European Union Agency for Cybersecurity</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>IIoT</td>
<td>Industrial internet of things</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>LLDP</td>
<td>Link Layer Discovery Protocol</td>
</tr>
<tr>
<td>MQTT</td>
<td>OASIS messaging standard designed for IoT.</td>
</tr>
<tr>
<td>MSPL</td>
<td>Medium-level Security Policy Language</td>
</tr>
<tr>
<td>MUD</td>
<td>Manufacturer Usage Description</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology</td>
</tr>
<tr>
<td>NoSQL</td>
<td>Not-only Structured Query Language</td>
</tr>
<tr>
<td>PANA</td>
<td>Protocol for Carrying Authentication for Network Access</td>
</tr>
<tr>
<td>PoC</td>
<td>Proof of Concept</td>
</tr>
<tr>
<td>PUF</td>
<td>Physically Unclonable Function</td>
</tr>
<tr>
<td>RADIUS</td>
<td>Remote Authentication Dial In User Service</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>REST</td>
<td>REpresentational State Transfer</td>
</tr>
<tr>
<td>SDN</td>
<td>Software Defined Networking</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>TMB</td>
<td>Trust Manager &amp; Broker</td>
</tr>
<tr>
<td>TMRA</td>
<td>Threat Modelling and Risk Assessment</td>
</tr>
<tr>
<td>XACML</td>
<td>eXtensible Access Control Markup Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>YANG</td>
<td>Yet Another Next Generation</td>
</tr>
</tbody>
</table>
1 Executive Summary

The purpose of the current report is to document the work performed in the context of Task 4.6 “Lifecycle Security of IoT Devices” and its results until M18. The deliverable is of type OTHER, and in fact the main outcomes are the initial software prototypes which can be found in the project’s software repository. However, this document also gives a detailed overview on the components that will result from Task 4.6 to support the lifecycle management of devices, along with their design, their fit into the general ERATOSTHENES framework, the relevant standards and state of art related works, along with their shortcomings and challenges, and how our developments will address those challenges. Thus, it becomes a reference point for the work in this task. Additionally, it will help guide potential users on the goals, functionalities, and specifications of the developed software components. The main outcomes documented in this work are:

- Exhaustive analysis of the status of MUD and threat MUD standards and recent works on the subject, identifying challenges and how they can be tackled by the ERATOSTHENES approach.
- The initial design of the adaptation of MUD and threat MUD proposals to the ERATOSTHENES architecture, including their extension through increased expressivity or complete lifecycle monitoring among others.
- The initial implementation of the related components, and integration with related components, specifically in the context of the bootstrapping and enrolment phases, within the proof-of-concept version of ERATOSTHENES and beyond.
2 Introduction

This document introduces and discusses the first outcomes of task 4.6. They are framed in the initial secure deployment and registration phase and focused on the application of Manufacturer Usage Description (MUD) files. These are the outcomes of the first development efforts spent on the components, for integration in the first integrated proof-of-concept (PoC), and additional work since month 14. These technologies will be further developed, evaluated, and improved in later revisions.

2.1 Mapping ERATOSTHENES Outputs

Table 1: Adherence to ERATOSTHENES GA Deliverable & Tasks Descriptions

<table>
<thead>
<tr>
<th>ERATOSTHENES GA Component Title</th>
<th>ERATOSTHENES GA Component Outline</th>
<th>Respective Document Chapter(s)</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DELIVERABLE</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4.2 Secure deployment and registration of IoT devices</td>
<td>This deliverable includes the design and development of the mechanisms of the framework related with the secure deployment of IoT devices and its registration in the directory. This is the initial outcome of T4.6</td>
<td>Sections 3, 4.1, 4.2, and 5</td>
<td>Section 3 covers the methodology and code availability for the initial outcomes of T4.6 in regard to secure bootstrapping and enrolment/registration, along with the positioning within ERATOSTHENES general architecture and goals. Section 4.1 describes the MUD standard, the main tool in T4.6 for the lifecycle management and specifically during bootstrapping, along with shortcomings, extension points and how it is being applied in ERATOSTHENES. Section 4.2 covers the design of bootstrapping and enrolment from the point of view of T4.6’s lifecycle management through MUDs, along with details on components and current implementation. Section 5 establishes state of art and innovation over it for the topics covered in the previous sections.</td>
</tr>
<tr>
<td><strong>TASKS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4.6 Lifecycle Security of IoT Devices</td>
<td>This task in is charge of developing a framework to manage the security of an IoT device throughout its lifecycle to create a framework for lifecycle management, integrating mechanisms for the secure</td>
<td>Sections 3, 4 and 5</td>
<td>Section 3 covers the positioning of the task activities and initial outcomes in ERATOSTHENES general architecture and requirements. Section 4.1 describes the MUD standard, the main tool in T4.6 for the lifecycle management</td>
</tr>
</tbody>
</table>
deployment, registration, operation, recovery and decommissioning of IoT network devices. It will provide four main mechanisms: a) Deployment of security policies during bootstrapping b) Device registration to the network c) Threats Monitoring/detection d) Dynamic security assessment e) Security information sharing. In the first block, we will consider the enforcement of security policies during the bootstrapping, in a way the device and network are protected before the device could access to any resource. For this, we will consider the usage of the MUD standard, and other sources such as vulnerability databases or previous security evaluations to obtain the appropriate configuration. We will also develop mechanisms to register the security information related with the device, so any entity could consult it at any moment. Towards this, we will consider ongoing standardization efforts, such as the CoRE Resource Directory. In the second block, we will focus on the device protection during its operation, developing mechanisms to monitor/detect threats and dynamically assess its security compliance during its operation phase, so new policies can be enforced as a response to a security flaw. We will consider and adapt the ARMOUR methodology for dynamic

Section 4.2 covers the design of bootstrapping and enrolment from the point of view of T4.6’s lifecycle management through MUDs, along with details on components and implementation as initial outcomes of the task.

Section 4.3 gives an initial overview and design approaches on topics within task 4.6 (monitoring/detection, dynamic assessment, security information sharing) that were not part of the deliverable.

Section 5 establishes state of art and innovation over it for the topics covered in the previous sections.

1 https://doi.org/10.3030/688237
security evaluation based on the information received from the monitoring and from external sources. In the third block, we will develop processes of security information sharing with any relevant entity (e.g., manufacturer, vulnerability databases, other entities of the network, etc.), so this information could be used in the other two blocks, not only for the deployment of the security policies but also for the detection, monitoring and security assessment of new encountered threats. The key outcomes of this task will be the development of the security lifecycle management framework and the different mechanisms for the secure deployment and operation, as well as mechanisms for security information sharing throughout the IoT device lifecycle. The results will be documented in D4.2 and D4.9.

### 2.2 Deliverable Overview and Report Structure

The deliverable is structured as follows. Section 3 positions the work in the broader context of the ERATOSTHENES architecture and the requirements of the pilots. It also describes the used development methodology and relates the availability of code. Then, Section 4 elaborates on the components for lifecycle management through MUD files, describing the standards, their shortcomings, how the approaches will be adapted and improved throughout the project, and details on design and implementation. Section 5 gives an overview on state of art, challenges and highlights on the innovations and scientific contributions. Finally, Section 6 concludes the report.

Additionally, even if multiple deliverables are related to this work because of the horizontality of the concepts, we remark the following deliverables (past and in the close future) that treat concepts heavily mentioned throughout the document:

- “D1.2 Use cases, requirements and methodological framework”: it was delivered in M6 and provides the requirements associated to the Trust Manager & Broker component [1].
- “D1.3 Preliminary ERATOSTHENES Architecture”: provides interactions between components, specifications regarding utilized communication protocols between them, technologies to be utilized as well as an overview of the proposed platform’s operations and components. This deliverable was delivered in M6 [2].
“D2.1 Trust Broker Mechanism”: Describes the TMB, within which the key components described in this document are located, including information on communication interfaces through MQTT [34].

“D4.3 Inter-ledger platform for Cyber-threat information sharing”: Describes the initial outcomes of the cyber-threat information sharing task, closely related with this task 4.6 as highlighted in section 4.3.

“D4.5 Intrusion detection for IoT-based context and networks”: Describes the initial outcomes of the monitoring and intrusion detection system on ERATOSTHENES, closely related with this task 4.6 as highlighted in section 4.3.
3 Architecture Orientation and Industrial Requirements

In this section, we position the main components related to the task at hand in the broader context of the overall architecture and align them with business needs, the different pilots of the project and their requirements. Then, we discuss the methodology for the development of these components and point to the code related to them.

3.1 Architectural Positioning and design decisions

Figure 1 presents the instantiated version of the architecture as of the time of this deliverable. We note that a slight modification is introduced with this work: the MUD Management Module was previously named MUD Manager and was renamed to avoid confusion with one of its subcomponents.

Various components of said architecture are related to the device lifecycle management, and specifically the secure bootstrapping, deployment and enrolment/registration of the device. This is the case, for instance, of the Trust Manager & Broker (TMB) or Monitoring and IDS components, along with supporting tools like DLT or specific elements that may be integrated into the lifecycle management ecosystem to enable or enhance its characteristics (like trust agent or PUF elements). However, the focus of this deliverable will be the components directly related to the management of Manufacturer Device Description (MUD) files, while interactions with other components will be described, but details on the components themselves will be covered in their respective tasks and deliverables. Namely, we deal with:

- MUD lifecycle management servers: There will be two kinds of servers, those dedicated to storing MUD files, which will be on device manufacturers premises, and those dedicated to saving threat MUD files, which will be controlled by threat intelligence actors (which may not be manufacturers). Current implementation follows the simple standard API as REST server for file retrieval using the MUD URL, but in the future we will consider possible adaptations into a publish/subscribe communication channel for enhanced update flows.
- MUD Management module: This component covers most functionality related with MUD files and will be replicated in each of the instances of a TMB inside the domains. While from the architectural point of view this will be a component that interacts with the rest of the TMB components through the MQTT

Figure 1 ERATSOTHENES instantiated architecture (modified).
communication interface, it will be comprised of various subcomponents. First, a MUD Manager as in the MUD standard will be in charge of retrieving MUD files associated to devices that enrol in an ERATOSTHENES domain. The threat MUD manager will carry out the equivalent functionality for threat MUD files. A translation module will be used to manage the communication with other components, e.g., by transforming MUD files into security policies. Lastly, a database will be used for storage of MUD files, along with information on their corresponding device or threats, lifetime and other metadata. For this, we use MongoDB\(^2\) in consonance with other TMB components.

More details on these components, their instantiation and the relationship to the standard MUD elements can be found in Section 4.1.4.

Note that the MUD Manager component, the central part of the solution, is part of the TMB. This reveals the close relationship with the other components within the TMB. A brief overview of potential interactions with them, which will be expanded upon in Section 4, is:

- **Threat Modelling and Risk Assessment (TMRA):** Behavioural information for devices can be useful for updating threat models on them and establish risk assessments. Threat information from threat MUDs can also be an interesting source of knowledge.
- **Monitoring and IDS:** Expected communications of devices will allow to monitor them and detect potential anomalies or attacks. Additionally, mitigation actions and detection information threat MUD can be implemented to improve monitoring of potential threats.
- **MQTT Broker:** It will manage communication between components. In the case of the MUD Manager, for instance, it serves as the source of MUD URLs for retrieval, which abstracts the network layer from the MUD obtaining phase in contrast with other applications of the standard.
- **CTI Sharing Agent:** It will be a key component as part of task 4.6’s lifecycle management goals, and a complementary source of information to threat MUDs.
- **Trust Manager:** While relationship may not be as direct, the behavioural information and the subsequent monitoring will affect trust evaluation of devices, through security events or the close relationship it has with the TMRA component.

### 3.2 Business, Industrial Positioning and End-User Requirements

One of the key business challenges for the development of IoT scenarios identified in the workshops and user studies carried out in the context WP1 is the need for certification and security configurations for automatized IoT lifecycle management. On the one hand, this refers to the challenge of easing and automatizing the deployment of devices, be it because of the large scale and heterogeneity as highlighted by the industry 4.0 pilot, or because of the distributed and replicated nature of deployments and involvement of people with low or no technical background, as highlighted in the health services pilot. On the other hand, it is necessary to go beyond that step, and cover the whole lifecycle of the device, with the necessary adaptations as security contexts change, because of configuration changes, addition of new devices, new threats. The components described in this deliverable take advantage of MUD files to tackle these aspects, supporting (and being supported by) other elements of the ERATOSTHENES architecture, especially related to its trust framework. They enable the discovery and parsing of behavioural information related to devices enrolling in the framework, which is useful for improving models, trust evaluations and monitoring within the domain. What is more, they consider the mutability through the possibility of updating MUD files and gaining new security information through threat MUDs. In fact, the latter, in combination with the monitoring and cyber-threat intelligence sharing tools within the project, allow the sharing of security and mitigation actions between actors as encouraged in the NIS2 directive [8]. All in all, the results coming from this task will serve as a key tool for achieving secure lifecycle management of devices from their initial security configuration during deployment and registration to the moment they are decommissioned.

The initial requirements for ERATOSTHENES projects were identified in Deliverable D1.2. In this section, we present the coverage of the requirements by the MUD file management components in the context of Task 4.6. To identify the related requirements, we have examined the initial requirements presented in D1.2 and the Component-
Requirement mapping presented in Deliverable D1.3. The results of our coverage analysis are presented in Table 2. For each requirement displayed in the tables, we present its description and importance (from D1.2), along with the Coverage Role and Rationale. The Coverage Role aims to depict the role of the MUD components described in this deliverable in satisfying the requirement. It can be Direct, which means that the components are of prime importance in satisfying the requirements, or Supporting, when the components support the achievement of the requirements in a more indirect way, complementing other components responsible for satisfying the requirements. Lastly, the Coverage Rationale explains the reasoning on these claims, and how the solution helps cover the requirements.

<table>
<thead>
<tr>
<th>Req. ID</th>
<th>Description</th>
<th>Importance</th>
<th>Coverage Role</th>
<th>Coverage Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1_FR_05</td>
<td>The infrastructure is monitored, and network traffic is analysed to detect intrusions</td>
<td>M</td>
<td>Supporting</td>
<td>The integrity of the communications between the vehicles and the infrastructure is ensured by some intrusion/anomaly detection, which is supported by device behavioural information included in MUD files, along with threat information from threat MUDs</td>
</tr>
<tr>
<td>P1_FR_06</td>
<td>Vehicle re-certification after software change</td>
<td>M</td>
<td>Supporting</td>
<td>After software update/upgrade, it might be necessary for the vehicle to be re-certified, which update flows for MUD files help support</td>
</tr>
<tr>
<td>P1_FR_09</td>
<td>Automatic and trustworthy deployment of software updates</td>
<td>M</td>
<td>Supporting</td>
<td>One of the extensions proposed is the identification of necessary software updates in MUD files</td>
</tr>
<tr>
<td>P1_NFR_14</td>
<td>Avoid single point of failure in the lifecycle management system of vehicle to smart city communications</td>
<td>M</td>
<td>Direct</td>
<td>Ensure the operation of an IoT lifecycle management system even after a/some nodes’ failure, e.g. through a flexible architecture where the TMB, and specifically MUD components, are duplicated in domains</td>
</tr>
<tr>
<td>P2_FR_02</td>
<td>Per-device trust calculation and calibration of trust in devices</td>
<td>M</td>
<td>Supporting</td>
<td>Algorithms to calculate trust at the basis of relevant inputs/events. Expected behaviour and threat information can be useful as variables to establish trust</td>
</tr>
<tr>
<td>P2_FR_04</td>
<td>Devices can be enrolled in the system and associated to user identity</td>
<td>M</td>
<td>Supporting</td>
<td>The information in MUD files is useful for the device enrolment on a domain, as part of the TMB enrolment process</td>
</tr>
<tr>
<td>P2_FR_05</td>
<td>The enrolment process is automatic after user input without direct intervention by Tellu</td>
<td>M</td>
<td>Supporting</td>
<td>The information established by the manufacturer in the MUD file allows for automatic enrolment where device expected behaviour is known and security policies can be derived</td>
</tr>
<tr>
<td>FR/PNFR</td>
<td>Description</td>
<td>Type</td>
<td>Role</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>P2_FR_06</td>
<td>The trust score measurement is assigned to every device and service, internal or external to the infrastructure.</td>
<td>M</td>
<td>Supporting</td>
<td></td>
</tr>
<tr>
<td>P2_FR_08</td>
<td>Manage device lifecycle (certification, deployment, upgrade, decommissioning, recovery)</td>
<td>M</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>P2_FR_09</td>
<td>Automatic deployment and update of software</td>
<td>M</td>
<td>Supporting</td>
<td></td>
</tr>
<tr>
<td>P2_FR_11</td>
<td>Initial Trust Score assignment to IoT devices during the enrolment phase</td>
<td>M</td>
<td>Supporting</td>
<td></td>
</tr>
<tr>
<td>P2_FR_20</td>
<td>The infrastructure is monitored, and network traffic is analysed to detect intrusions</td>
<td>M</td>
<td>Supporting</td>
<td></td>
</tr>
<tr>
<td>P2_NFR_01</td>
<td>Reduce the time on device trusted enrolment and access configuration</td>
<td>M</td>
<td>Supporting</td>
<td></td>
</tr>
<tr>
<td>P3_FR_03</td>
<td>The infrastructure is monitored and network traffic is analysed to detect intrusions</td>
<td>M</td>
<td>Supporting</td>
<td></td>
</tr>
</tbody>
</table>
3.3 Methodology

This section summarizes the methodology followed for implementation of the components described in this document, as well as design decisions, issues, and risks.

3.3.1 Implementation methodology

The development of the MUD-based lifecycle management technology is, following, as other project components, the WP5’s incremental and prototype or pilot-driven methodology. The first development, as denoted by the deliverable’s scope, focuses on the bootstrapping and enrolment phase and initial integration of the components into the ERATOSTHENES architecture and flows, including the PoC deployment developed in WP5 in M14. In subsequent iterations, this will be extended with operational phase integration through MUD updates, threat MUD and collaboration with other TMB components.

Some of the steps resulting from this iterative approach are:

- Implementation of very first version of MUD Manager using Python and an Apache-based server for MUD File Server. Functionality developed: web interface offering basic functionality to retrieve MUD files from the server.
- Second version, progress in the implementation and its integration into the ERATOSTHENES framework through the TMB developed component. Achieved a scenario where the MUD manager and the TMB are connected through MQTT (topic subscription) and added the functionality of retrieving the MUD File using MQTT notification.
- Unit test and integration tests for output validation and deliverable D5.1.
- Dockerization of all components to be deployed to accomplish the PoC and to get them work with the rest of TMB submodules (databases, ports, etc.). Docker images were generated and uploaded to the Nexus repository, in addition to respective repositories for each element.
- Initial implementation efforts beyond PoC, including a translation module that takes MUD files and generates the corresponding MSPL-based policies that will be shared with other components through the MQTT broker, thus giving them access to information in MUD files relevant for risk analysis and threat modelling, monitoring or policy enforcement.

This implementation effort has been integrated into the general work package and task working methodology. Namely:

Meetings:

- Bimonthly meetings were organized by the WP4 leader.
- During these meetings all partners involved in WP4 provided insight and were kept informed on the progress of each component’s design and development.
- Moreover, risks, issues and concerns were raised.

Collaboration tools:
• Slack channel to resolve day to day implementation and integration issues as well as to arrange calls.
• Microsoft Teams meetings
• E-Mails
• GitLab (branches, merge-request)

3.3.2 Implementation design and development decisions

The design of the implementation takes a starting point the MUD standard and threat MUD proposal, but taking into account how they can be adapted to best fit the ERATOSTHENES framework. In particular, the work developed in D1.3, with flows and architectural description, have been a keystone for this development. This has lead to one of the main overall design changes from the traditional MUD applications: the obtention and usage phase of MUDs have been abstracted from the specific network, instead taking advantage of, and supporting, the trust framework components of the ERATOSTHENES architecture. This will help with generalization of the related actions, as well as improved capabilities like application-layer extensions, direct impact on monitoring, or multiple enforcement approaches (i.e., more direct or through indirect means like trust evaluation).

At the very first of the initial development, we selected MongoDB as database for its scalability, flexibility, and ability to handle structured and unstructured data, along with its integration capabilities and advanced features. Besides, other components are using MongoDB as well, so we are searching for compatibility between system components, which will also enable easier deployment and establishment of security and reliability measures like database replication. However, the development is not tied to this decision, so we could move to options that fit better if needed.

For development of the components, we are using Python as a programming language. Python is a popular and widely used programming language. Used along with Flask, a rapid web development framework, and MQTT, a lightweight messaging protocol for IoT, allows for the creation of efficient and scalable web and IoT applications. Python's large developer community and compatibility with various platforms make it an attractive option for business application development.

3.3.3 Implementation Issues and risks

Several issues may occur in the implementation process: compatibility issues related to python and the docker images used for deployment, although the high containerization of the applications, which will be deployed at infrastructure side, mitigates potential issues; there were several implementation problems related to Flask module version and MQTT version, which are as of now solved; we have also encountered other minor difficulties in linking the MUD module with the dockerized TMB, but the integration is currently working and the extensibility through different topics will enable simple future developments.

3.4 Code Availability

Include a link to an online repository where the code is available. We should be consistent on where this folder will be. Include a ‘readme.txt’ in this folder to describe files, compilation etc.

The code for the components described in this deliverable is available on the ERATOSTHENES project’s GitLab repository, with repositories for the MUD Management Module and MUD File Server modules which are described in Section 4.1.4.
4 Lifecycle management through MUD files

In this section, we describe the approach for lifecycle management taken in the project, which has the Manufacturer Usage Description (MUD) files at the centre. In the following, we discuss the MUD standard and Threat MUD proposal, how they fit in our solution, their shortcomings and how we can palliate them, e.g., through extended models for added expressivity. Then, we describe how we take advantage of MUDs for secure bootstrapping and enrolment of devices in ERATOSTHENES, detailing the procedures and the current status of the implementations. Lastly, even if not part of the focus of the deliverable, we give initial positioning and ideas for how lifecycle management will be tackled after enrolment, which will be expanded upon in later deliverables.

4.1 Manufacturer Usage Description (MUD)

The Manufacturer Usage Description (MUD) files, the architecture and lifecycle for managing them, were standardized in 2019. Additionally, a closely related variant of the procedures, known as threat MUD, has also been proposed as a mitigation mechanism during operation. In this section, we will introduce the concepts for these approaches, along with their main limitations, and how we can tackle some of them in the project.

4.1.1 MUD standard

To achieve effective detection and mitigation of security threats in specific IoT environments, it is useful and sometimes necessary to know the expected behaviour of devices beforehand. However, the heterogeneity of IoT environments (from non-critical and dynamic home environments, to IIoT) and of devices themselves, which are based on various technologies and communication protocols. Furthermore, the restrictions inherent to certain IoT devices (e.g., the lack of user interface) make management of IoT devices cumbersome for non-expert users. To cope with these challenges, standardization of the identification and management of device behaviour is key.

In this direction, the Manufacturer Usage Description (MUD) standard was published in 2019 by the Internet Engineering Task Force (IETF) [3]. The MUD specification's major goal is to limit the threat and attack surface of a certain IoT device by allowing manufacturers to establish network behaviour profiles for their devices. Each profile is specified through Access Control Lists (ACLs), which establish policies for communication endpoints. They are defined using Yet Another Next Generation (YANG) [4] to model network restrictions, and JavaScript Object Notation (JSON) [5] as the serialization format. Figure 2 shows an example of a MUD file, where we can appreciate the two main containers of the specification. The “mud” container provides information about the MUD file itself, such as its URL, how long it should be cached, information about the device. Additionally, the MUD data model describes the “acls” container based on [6], including network communication restrictions for the device, e.g., allowing communication with a certain host and port.
Additionally, an architecture with components and flow accompanies the models so that it is possible to obtain this profile in the domain of usage and establish the security context and possible enforcements from it. Figure 3 shows said architecture, where the device sends its MUD URL along with a connection request to router or switch, which forwards the URL to the MUD Manager. It can then retrieve the file from the MUD file server on the manufacturer’s premises.

Since its adoption, MUD has been object of interest both from researchers and standardization bodies. In particular, the National Institute of Standards and Technology (NIST), and the European Union Agency for Cybersecurity (ENISA) consider the use of MUD as part of future IoT security good practices to increase security against cyberattacks in IoT domains.
4.1.2 Threat MUD proposal

The protection of security in environments with IoT devices does not end with the initial device installation. Many vulnerabilities and attacks can, and indeed are, discovered during the operational phase. For instance, only in 2022, more than 25,000 vulnerabilities were detected [7]. Manufacturers cannot always deal with vulnerabilities quickly enough through updates, which follow a complex process, and in many cases the relationship with third-party services makes this even harder. While MUD files can be updated, this process does not cover most vulnerable scenarios. In this sense, security-information-sharing systems enable fast and collaborative sharing, analysing and mitigation of vulnerabilities or attacks which may also be applied before a patch is released. The sharing of cyber-threat information for building cybersecurity capabilities has also been a big focus of the NIS2 Directive of the ENISA [8].

In this direction, the NIST proposed a threat Manufacturer Usage Description (threat MUD) [9], as a way to share vulnerability information and its mitigations. It is based on the MUD standard for device behaviour specification and follows a similar structure. However, despite the close relationship, threat MUD is its own concept intended as a mitigation mechanism. However, the NIST only gives some indications about the threat MUD model and functioning, leaving some details undefined but framed in the guidelines. In the following section, we overview the threat MUD model as presented in [10], which we use as basis, though with differences on how the flows are integrated into the architecture (e.g., receiving threat ids for which to search associated files through the TMB). This work takes NIST guidelines and the MUD standard as a starting point.

A key element to have in mind for the threat MUD is that, even if its structure and operation are similar that of the regular MUD, its purpose is to serve as a mitigation method against a specific threat, in particular through network communication rules for sites that have been associated to a threat. Therefore, a threat MUD does not need to be strictly related to a specific device, nor specify expected behaviour. As a result, rather than being developed by the manufacturer, the threat MUD might be created by a threat intelligence provider.

As with the standard MUD, the threat MUD model has two modules. The first module reflects information about the threat MUD itself. It contains equivalent fields to the MUD standard, such as version, url, cache-validity, signature etc. However, some fields have to be modified (manufacturer name is changed to intelligence provider, device model name is changed to threat name), removed (information about device to which the MUD is associated like system or firmware is no longer needed) or added (CVVS and documentation fields for additional information about the threat) to fulfil the new mitigation goal. The second module is again related to the ACLs that specify the conditions and restrictions. In this case, as the file is tied to a threat and its mitigation and not to a specific device, the configuration to apply should be as generic as possible. Thus, unnecessary fields like same-manufacturer, local-networks, controller and my-controller (which established conditions specific to the device) are removed.

4.1.3 Shortcomings and extensions to MUD

One of the main shortcomings of the standard, although understandable because of its scope, is the reduced expressivity of the models. Indeed, while the standard allows some possibilities for extension (e.g., a field reserved for future extensions, and expansions are being considered for Quality-of-Service aspects) and flexibility, the MUD granularity of the standard specification is somewhat limited, and additional security restrictions beyond the network layer are not possible. The definition of such enriched behavioural profiles could be used to detect/avoid a broader range of potential security attacks, including application layer threats such as slow DDoS attacks3.

In the following, we describe some possible extensions of the expressivity to characteristics that may be known at design time or when updating MUD files, but the original MUD does not cover, or be useful for creating mitigation actions through threat MUDs, specifically in the ERATOSTHENES context.

- Application layer protocols that also define restrictions on the communications. E.g., whether a device should use MQTT, TCP or UDP, affects how they should communicate in the network. Note that, especially in the case of the standard MUD protocol, this would refer to the application role of the device. The applicability of these policies to communications necessary for the ERATOSTHENES infrastructure (e.g., the instantiation

---

3 https://doi.org/10.1504/IJTMCC.2013.056440
of TMB communication through MQTT) will depend on and be decided by the own infrastructure components when creating the security context for devices in the specific deployment and domain.

- Cryptographic algorithms, which not only add restrictions to the communications, but also specify the supported algorithms of the device and its preferences. This may be useful for communications (e.g., as parameters for establishing secure channels) but also for higher level needs of the ERATOSTHENES infrastructure, like the preferred/supported algorithms and security level for identity management elements (e.g., signatures that support zero-knowledge proofs or not, depending on its capabilities).
- Exposed resources offered by the device, like HTTP/CoAP endpoints to retrieve information, which could be used in ERATOSTHENES both for discoverability of devices and to take into account the expected behaviour for threat and risk models.
- Known vulnerabilities (introduced in updates to the MUD file) associated with the device, following a format of a CVE entry and additional information like CVSS scores, which could be associated with mitigations or restrictions.
- Software/firmware restrictions, for instance defining a minimum software version for optimal operation, or establishing an update as part of a mitigation action. This would be useful for and closely related to ERATOSTHENES automatic software update and deployment flows.

Additionally, as identified in [11], multiple challenges arise from the application of the MUD standard, and extension points are being researched in many works on the topic. The area is very extensive, with multiple challenges arising from the standard and proposals. The work in this project will be especially related to the following challenges, along with the aforementioned extension of the MUD file expressivity. The mutability of MUD files, and how they are kept up to date in deployments, is not covered in the standard. Another challenge is the necessary authentication during MUD obtention so that the infrastructure may know that the MUD URL corresponds to the device sending it, and not to some other entity. Another avenue of research is the analysis of traffic based on MUD information, to better detect potential attacks. Also, the MUD obtention process in the standard, through requests to the switch or router, may not be applicable to many use cases. Lastly, the standard does not cover the particularities of application to relevant sectors like IIoT or smart vehicles, especially when advanced features through extended MUD models are in play. More details on the how the work in this project is relevant for challenges are given in the following section, along with Section 5.

### 4.1.4 MUD in ERATOSHTENES

MUD is integrated into ERATOSHTENES as a source of behavioural information of the devices that enrol in a domain. As shown in Section 3, the MUD file servers are outside of the ERATOSHTENES domains, e.g., on the manufacturer premises. MUD components from all domains may communicate with them to retrieve MUD files. This communication is currently through simple REST API, but we will explore ways for MUD updates like periodical requests or communication through publish/subscribe technologies. On the other hand, the main functional components of the MUD standard and threat MUD proposal will be active in each domain, as part of the Trust Manager & Broker component.
Figure 4 shows the detailed instantiation of the MUD components in the ERATOSTHENES architecture. Note that the MUD management module in ERATOSTHENES is an aggregation of subcomponents, including the functionalities of both standard MUD and threat MUD managers, plus a translation module for the interaction with other ERATOSTHENES components, e.g., by transforming MUD files into corresponding security policies.

The main contrast with the standard MUD architecture in Figure 3 is the abstraction of the network layer. Indeed, one of the key advances over the MUD obtaining process in the standard comes from the integration of the MUD processing into a full-fledged trust framework. Now, we take advantage of the ERATOSTHENES domain enrolment phase to get the MUD url from the device, through the publish/subscribe approach used for communication between TMB components. The integration in such flow, and within the ERATOSTHENES ecosystem, also allows the exploration of further improvements for the MUD flow.

For instance, while the communication with the MUD file servers can be easily secured in any usual way for webservers (e.g., HTTPS), there is a glaring gap in the standard regarding the possible spoofing of a MUD URL by the device. This authentication of the URL during MUD file obtention can be tackled through the project’s identity framework. Specifically, the manufacturer (at bootstrapping time, along with PUF and MUD URL installation) will associate the PUF of a device (as a root of trust for identity) to the respective URL. Alternative approaches can be explored in the case of not having PUF available as a root of trust, following the project’s general approach for tackling this case, e.g., by taking it into account when threat models, risk and trust for the device are evaluated.

Of course, specific network elements will still have to be considered in specific instantiations, but this will be partly solved by the own MUD mechanism: MUD files will be associated to devices that are relevant to the application domain, which will make the rules included in them relevant to the domain deployment by default. What is more, the advantages of the extension of the MUD model with higher level concepts, like software updates or cryptographic parameters restrictions, will be even more relevant through this abstraction.

Additionally, the inclusion of the MUD elements in the TMB allows simple interactions with other components very relevant to the application of its ideas and the management of the lifecycle of devices. For instance, as mentioned before, the information contained in the MUD file can be considered in the TMRA component and influence trust on the device. What is more, monitoring elements like IDS can also use the MUD information to perform traffic analysis, and be able to detect and respond to possible threats. These interactions enable multiple possibilities for enforcement, as it can still be direct enforcement at network level, but also indirectly controlled through evaluation of the device behaviour and adjustment of its trust level as considered in its domain interactions. For these communications, we introduce a translation unit of MUD files into intermediate security policies. In the specific implementation, we are currently working with Medium-level Security Policy Language (MSPL) [17], a security policy language with medium level of abstraction, that provides a set of actions suitable by the most common applicable security settings. The MSPL’s structure is defined in a YANG model, which allows using XML or JSON as encoding format. This allows
flexibility when applying the information contained in MUD files. Other components will take the medium-level policy from the topic and take advantage of it for their purposes, possibly with another translation into their own structures. For instance, the TMRA may use the policy directly to increase its knowledge base for building models, while the IDS may translate policies into rules that detect related events and further enforcement policies may be derived by PDPs.

### 4.2 Bootstrapping and enrolment in ERATOSTHENES

In this section, we cover the initial implementation efforts, that have been focused on the bootstrapping and enrolment phases. The development of the PoC has also been a staple within the development process. In Section 3, we already introduced our approach on designing and implementing these prototypes. Here, we overview the current status of the implementation of the components, their integration and interfaces, as well as identified future plans for the implementation.

#### 4.2.1 Components

In this phase, the main acting components are the MUD Manager and Translator subcomponents of the ERATOSTHENES MUD Management module, along with the MUD file server. In particular:

- **MUD Manager**: developed with python’s Flask framework, it’s subscribed to the MQTT topic of the broker, from which it receives the MUD URL related to a device and retrieves the MUD File from the MUD File Server. Once done, returns the MUD File to the Translator module.

- **Translator**: also developed with python’s Flask framework. Receives the MUD File related to a device and transforms the conditions and restrictions it states to the corresponding actions in a MSPL format. Once translated, adds to the MSPL policy the information of the device to which it’s addressed and sends it to the MQTT corresponding topic.

In addition, even if outside the bootstrapping/enrolment phase, work on two other components implementation has started:

- **Threat MUD Manager**: similar to the MUD Manager, it collects the Threat MUD File associated with the threat identifier received from the CTI sharing topics. Once it is retrieved from the MUD File Server, sends it to the Translator module that transform the mitigation actions from the Threat MUD File to the corresponding MSPL policy. This service and the Threat MUD File Server are being implemented using Flask as well.

- **Threat MUD File Server**: stores the Threat MUD Files related to the threat identifiers, similar to the MUD File Server but in this case the server does not belong to the manufacturers, it’s a single server for all entities.

#### 4.2.2 Design choices

The MUD Manager is a web server using flask (python), with different routes to interact with. Besides, it uses MQTT to interact with other components and obtain MUD URLs.

The MUD File server is a simple web server hosting the files.

For the database, between NoSQL and SQL, we chose the former due to several reasons:

- SQL is a relational database, a database which model data as records in tables with logical links between them, has fixed-defined schemas for data record and could have high-cost of infrastructure.
- NoSQL are usually documents-based (key pair, JSON) databases, with flexible schemas, high availability and throughput and low-cost infrastructure.

For MUD files and threat MUD files, the system does not really need file relationships. Also, within a domain, different devices will load different MUD files from different manufacturers, which could include extra elements/information (e.g. metadata, dates, snapshot data) that could affect the final MUD structures. The system then searches a flexible database. In addition, the ERATOSTHENES’ approach aims at a scalable and distributed system, requirements that NoSQL databases fully meet. Lastly, as part of the aim of integration, we considered a desirable goal to choose the same kind of technology used by other TMB components, to homogenize storage in the component, with the benefits it entails (ease of management, homogeneous and stricter security measures etc.).
4.2.3 Integration in ERATOSTHENES

Components that are already integrated in ERATOSTHENES are:

- **MUD Manager**: web interface using flask and python which interacts with the TMB using MQTT:
  - MUD manager is subscribed to the enrolment MQTT topic ‘RiskAssessment’.
  - Receive a MUD File request in the topic ‘RiskAssessment’ in the enrolment phase.
  - Retrieve the MUD File from the MUD File server.
  - Store the MUD File in the database (MongoDB).
  - Send the MUD File to the specific topic of the device and the MUD file: ‘device/deviceDID/MUDFileJSON’.
- **MUD File Server**: Apache Server hosting MUD Files.
  - Clients download files through HTTP requests.

In development components:

- **Threat MUD Manager**: In relation with TMB, the next steps are to receive the mud request in a specific MQTT topic and share it in a specific topic (for each device).
- **Translator**: Transform MUD File content to MSPL as an intermediate security data format is done, while it is still necessary to integrate with the next version of TMB through the definition of a topic for sharing the results.
- **Threat MUD File Server**: Server hosting Threat MUD Files.

Future work:

- TMB collaboration to use the MUD Files to participate in the trust score calculation.
- Regarding other TMB components and modules, it is important to share MUD files and threat MUD files with CTI and IDS, both within and between domains.

4.2.4 Interfaces and deployment

For the initial development and deployment of the MUD module inside the TMB and the file servers, the PoC specification and flows, an extract of which is shown in the following figure, have been a key driving force. The deployment has been carried out through docker containers (docker-compose⁴ was defined for automatic deployment) and tested in the PoC (c.f. Figure 5).

![Figure 5 PoC enrolment flow extract](https://docs.docker.com/compose/)

- **MUD FileServer**:

---

⁴ [Link to Docker Compose documentation](https://docs.docker.com/compose/)
This server contains files created by the manufacturer.

- **Interface:**

  **MUDRequest (MUD_URL) = MUDFileJSON**
  HTTP Request to MUD URL and Returns JSON MUD File based on the MUD URL
  Example:
  
  
  Returns JSON MUD file

- **Instantiation:**

  Apache Server serving MUD Files

- **MUD Manager:**

  This manager is responsible for capturing MUD URLs and retrieving MUD files for the servers.

- **Interface:**

  **getMUDFile (DeviceDID, MUD_URL) = MUDFileJSON**
  HTTP Request to MUD URL with params device ID and MUD_URL
  Example:
  
  
  Returns JSON MUD File based on the MUD URL

- **Instantiation:** REST service that receives requests from network monitoring script/module (modified DHCP Server)

- **Note that the interface is intended for the component as a standalone. In practice, the communication with other TMB components. Including the retrieval of the MUD url, is done through the MQTT broker. In particular, the MUD manager automatically subscribes to the topic ‘RiskAssessment’ (c.f. D2.1 [34] for further details), from where the manager will receive the MUD URLs.**

Up to this point, we covered the implementation status of the components involved in the PoC. Beyond that, we have continued work on the other components that form the MUD management framework in ERATOSTHENES. For instance, we defined and integrated the translation module, for which we show the interface in the following.

- **Translator:**

  The Translator Service is responsible for translating the MUD File actions linked to a device to a MSPL policy in XML format.
4.2.4.1 Deployment

The mud components are deployed using docker-compose. Both current developed components for the PoC are inside the same docker-compose file and in the same docker network so there is communication between both components. Beyond the PoC, the process will be the same, with the additional translation and threat MUD components. The MUD Manager needs the endpoint of the TMB to automatically subscribe to the topic ‘RiskAssessment’, from where the manager will receive the MUD URLs to download them from the MUD File server and share them in the corresponding topic of the specific device for MUD file sharing.

4.2.5 Known future changes

The current implementation is a first version that will evolve throughout the project. Apart from improvements related to the engineering itself, and possible changes after feedback from integration with other components and pilots, there are some known changes that will happen to it to cover all the functionality related to bootstrapping and enrolment. In particular, while the current implementation already deals with translation of MUD files into security policies, the integration with other components like TRMA or IDS for enforcement is not completed. Additionally, the process for MUD retrieval does not yet include verification of the holder of the MUD URL so that we are sure that the MUD file actually corresponds to the device, which will be tackled mainly using PUFs.

4.3 Device protection during operation

While the focus of this deliverable is the initial deployment and registration of IoT devices in the ERATOSTHENES framework from the point of view of the lifecycle management task, this section gives a brief overview on the initial and future work on other topics considered in the task, which entail applications during the operational phase, and will be covered more thoroughly in future deliverables.

4.3.1 Threat monitoring and detection interactions

As pointed out in previous sections, monitoring and attack detection are key for secure operation of environments, and closely related with the lifecycle constructs relevant to this deliverable. The information in MUD files will be useful for detection. During operation, the security policies coming from the enrolment phase will be relevant, but also those related to potential updates. What is more, information on threats and mitigations obtained through threat MUDs may also enable further monitoring capabilities. In the implementation scope, the interaction will be carried out through the shared MQTT broker, where the relevant components will subscribe and/or publish information in topics tailored...
to the communication case. E.g., we can consider a topic \textit{MUDpolicies} where the translated information from MUD files associated to devices or threats is published by the MUD manager, allowing other interested components (like TMRA and IDS) to get the information.

The actual threat monitoring and detection efforts in the ERATOSTHENES project will be covered by tasks 4.4 and 4.5. For instance, the IDS approach, and its possible integration with the elements of this deliverable, will be treated with more details in deliverable D4.5, and an advanced approach for detection based on federated learning techniques in D4.4.

4.3.2 Security information sharing

One of the key topics of the NIS2 directive by ENISA \cite{ENISA} is the need to enable secure and privacy-preserving sharing of security information and cyber-threat intelligence (CTI). This must include any relevant entity (e.g., manufacturer, vulnerability databases, other entities of the network, etc.), and will be useful not only for deployment of the security and mitigation policies but also for the detection, monitoring and security assessment of new encountered threats. Part of this functionality may be covered by threat MUD files, which have been overviewed throughout this document. Additionally, further efforts in this regard will be carried out within the project’s task 4.3, which deals with CTI sharing across domains with (inter-)DLT as a supporting mechanism. Again, the CTI sharing agents will be a component within the TMB because of the close relationship with others like IDS or the own threat MUD manager. More details on the approach for CTI sharing in ERATOSTHENES will be given in deliverable D4.3.

4.3.3 Dynamic security assessment

One of the main aspects of the cybersecurity world is the ever-changing nature of security contexts and threats. Thus, there is a need to dynamically change the security assessment and measures, especially in the heterogeneous and changeful world of IoT. Multiple elements of the ERATOSTHENES framework will be part of the efforts for tackling this aspect. At the device level, while MUD files are especially relevant during bootstrapping and enrolment, they can be updated with new information (e.g., known vulnerabilities, recommended software updates) by the manufacturers, and uploaded to file servers. Those updates can then be used to improve the security protocols around the device and avoid obsolete information. The standard does not establish a way for carrying out these updates. In the project, we will explore possibilities for doing these updates in a scalable and efficient way, either through the traditional flow or departing more from it with publish/subscribe protocols. Another tool for dynamic assessment, in this case mostly focused on threats and potential malicious actors themselves, is the sharing and analysis of threats through threat MUD files, which will be carried out during operation. In ERATOSTHENES, the capabilities provided by these tools will be enhanced through the specific trust management framework. Particularly, the TMRA component allows dynamic threat modelling and risk assessment that can take advantage of the aforementioned data. Also, the monitoring and IDS tools will both serve as a potential source of information, and as a mean for acting on the received security updates. Lastly, the CTI sharing capabilities of the ERATOSTHENES framework will enable transference of knowledge between domains and CERT/CSIRT teams.
5 Research and Scientific Innovation

The main starting point for the developments in this task, and particularly the components described in this document, come from the recent Manufacturer Usage Description (MUD) standard [3], and the related but complementary threat MUD proposal [9]. While these technologies have received growing attention (e.g., by ENISA and NIST) and are considered a promising approach (many research works on IoT security [11-30]), including a complete initial standardization process for MUD, multiple limitations and research directions have been identified, e.g., in [11]. In this project, we do not only innovate through creating the necessary adaptations for application in relevant sectors like IIoT or smart vehicles, but we go beyond the standards and other recent works in addressing multiple challenges related to them.

In a vastly different approach than existing research works, which focus on network protocols like DHCP, RADIUS (c.f., e.g., [9,13,15,16]) for the **obtainment process of MUD urls**, we provide an architecture that abstracts from network elements and takes advantage of (and supports) the ERATOSTHENES trust framework (and its enrolment phase), as well as the identity framework, and specifically PUF, for **authentication** during this phase. This is also the case for the **enforcement** phase, where most works focus on network enforcement, e.g., through SDNs (e.g.[19-23]), although a few do go into application level extensions [18, 24, 29] through the ERATOSTHENES architecture, we not only support direct enforcement, but also indirect enforcement through trust evaluation of devices. What is more, the **monitoring and traffic analysis** tools will be enhanced by the information in MUD files. Additionally, the shortcoming of a lack of **expressiveness** of the standard MUD models has been identified and partially tackled by various authors (e.g., [20,29,30]). In ERATOSTHENES, we not only consider extensions for the MUD model for application level or key ERATOSTHENES concepts like software updates, but they are enhanced by the fit within the architecture with application-level enforcement, threat modelling and risk assessment, trustworthy software updates. Lastly, we remark that the project will tackle the issue of efficient and scalable **MUD updates and dynamicity of security contexts**, with the inclusion of update flows, threat MUDs and other means for sharing threat information. In short, the approach taken in the project for lifecycle management through MUD files will innovate over existing scientific contributions by providing comprehensive mechanisms at multiple levels (e.g., not only network level, or one-time configurations) that tackle multiple challenges identified in the MUD standard and other recent works and ensure the continuous management throughout the lifecycle including collaboration through sharing of cybersecurity information from many sources.

In the following, we give more details on various state of art works related to the MUD standard and how they compare with our approaches. In [13], current DHCP functionality is extended to transport MUD URL during MUD obtainment phase. Similarly, the use of vendor-specific extension of LLDP [9]. However, both cases are highly vulnerable to possible spoofing of MUD URL [3]. Other approaches seek to include MUD URL in Authentication, Authorization and Accounting (AAA) infrastructure [14], relying on X.509 certificates for it, as considered by NIST [9] and various works [15,16], or along with pre-shared key authentication through EAP [17,18]. However, in all of them, the network layer plays the key role in MUD obtainment, while our approach abstracts this information.

Similarly, most works consider network enforcement, either directly through routers or switches [16, 19, 20, 21], or using SDN as a more flexible, scalable and following current trends approach [17, 22, 23]. In general, both approaches have to deal with the issue of possible explosion of rules and can only work on network-level rules (and sometimes even more limited, depending on capabilities of the deployment). More closely to our approach [18, 24] also include application-level enforcement by translating some of the (extended) rules into XACML policies. However, the solution still relies heavily on network infrastructure, for instance for MUD obtention, and does not consider further elements like “soft” enforcement through IDS or device trust, nor MUD updates and sharing of CTI data. Nonetheless, we note that our approach (similar to theirs) does not rule out direct network enforcement, for instance through SDN (which is interesting because of its flexibility), but it will depend on specific deployment aspects.

Some other works, like [20, 25, 26] do propose MUD files as source for improving IDS. However, they do not consider further complementary technologies like trust evaluation (affected by IDS events). Also, they do not focus on how they are obtained and linked to device identity and its place in the network. What is more, they do not consider potential changes of “normal” device behaviour. In our approach, we will consider MUD file updates, and additionally threat MUD files parting from the model in [10], with a different architectural approach for retrieving (e.g., data coming from IDS/CTI sharing) or applying this kind of knowledge. Our approach thus addresses the lack of a security lifecycle
management framework to monitor the devices’ security compliance fostering the collaboration among certification stakeholders (e.g., users, manufacturers, authorities).

Lastly, the lack of expressiveness has been identified multiple times (c.f., [11, 27, 28]). A few works have proposed specific extensions. Subsequent works [18, 24, 29] included some application layer information, like data privacy policies, known vulnerabilities or security configuration parameters. In [20], a proposed extension includes physical layer parameters and flow statistics that depend on the environment where the device is deployed. [30] proposes the addition of rules for establishing (D)TLS parameters. In this sense, [31, 32, 33] represent initiatives that seek standardization of MUD extensions, e.g., related to bandwidth, versioning/dependencies or update information. This denotes the interest on the subject and encourages our work in increased expressivity of models and, particularly, the necessary tools and processes to take advantage from that information (e.g., through secure software updates).
6 Conclusions

This deliverable describes the initial outcomes of task 4.6, and its support of ERATOSTHENES lifecycle management through Manufacturer Usage Description (MUD) files. The deliverable discusses the MUD standards, their shortcomings and the design of their application and improvement in ERATOSTHENES. This includes the two main components that are outcomes of the task, the MUD management module and MUD servers, their position in the general ERATOSTHENES architecture, and current instantiation and implementation results, along with the groundwork for future development and implementation. This work has also included an overview on state-of-art in terms of the discussed technologies and standards, and how the work in ERATOSTHENES tackles various challenges bringing innovation to the scientific field. While the focus of the deliverable is the bootstrapping and enrolment phase, it also includes initial positioning and development ideas for the operational phase, including potential synergies with other components and tasks. Further results on this task will be detailed in D4.9.

In parallel to these development, existing integration as part of the proof of concept, and future integrations and validations (in the three pilot cases) will provide valuable feedback that will be taken into account in subsequent development of these technological enablers.
7 References

[1] ERATOSTHENES project, "D1.2 Use cases, requirements and methodological Framework", Delivered in M3.
4. Secure Deployment and Registration of IoT Devices


[34] ERATOSTHENES project, "D2.1 Trust Broker Mechanism", Delivered in M14.