DOI: 10.1561/9781638285076.ch6

Chapter 6

Inter-Ledger Platform for Cyber-Threat Information Sharing and Lifecycle Security

By Jesús García-Rodríguez, Ekam Puri Nieto, Juan Francisco Martínez Gil and Agustín Marín Frutos

The security configuration of devices within their deployment context is a key challenge for the secure management of IoT ecosystems in scientific and industrial environments. The challenge is not limited to the initial deployment of the device but branches out to the whole lifecycle of the device where it can be affected by dynamic environments and the arrival of new threats. The cybersecurity of the solution, particularly regarding the latter, hinges on an effective Cyber-Threat Intelligence (CTI) exchange, as pointed out by the NIS2 directive. However, the process of sharing CTI data must be itself secure, privacy-respecting and far-reaching to ensure meaningful participation and results. This chapter describes the solution developed within the ERATOSTHENES project for achieving privacy-preserving CTI sharing in IoT scenarios. It consists of dedicated components within security domains to share and receive this information, backed by an inter-ledger approach as trustworthy backbone for the inter-domain data exchange. Additionally, the chapter introduces the application and extension of Manufacturer Usage Description (MUD) files for applying security configurations and mitigation actions related to devices or threats within a domain. These components help achieve a system that reacts to cyber-security incidents across the whole lifecycle of devices and security domains.

6.1 Introduction

IoT devices are intended to improve people's daily life and business environments. However, several aspects, such as their generally low capabilities, long lifetimes, or ubiquitous access, make them an attractive attack vector, becoming a risk in any deployment. Thus, it is necessary to establish mechanisms for managing security aspects comprehensively throughout their lifecycle, including not only an initial secure deployment but also the protection and reaction to events during their operational phase. Because of the dynamic nature of threats and attacks, it is important to share up to date Cyber-Threat Intelligence (CTI), as highlighted by EU's Network and Information Security (NIS2) Directive. However, there are barriers to the adoption of such measures, such as the potential security and privacy risks that come from data sharing. In this chapter, we delve into the topics and techniques for privacy-preserving CTI sharing, and showcase how, in conjunction with the Manufacturer Usage Description (MUD) initiative for IoT devices and threats, it eases tackling aspects of lifecycle security in IoT ecosystems.

6.2 Privacy-Preserving CTI Sharing

Security-related events incur in many direct and indirect loses for business and industries. Thus, a key challenge in the IoT sector is the need for increased cybersecurity. To achieve meaningful results in terms of detection and mitigation of cyberthreats, it is imperative that information on cyber-threats is exchanged, as pointed out by the NIS2 directive [1]. This is not a trivial matter, with many of the associated challenges already pointed out in said document. This chapter describes a series of components that together provide a platform for inter-domain CTI sharing. Before going into the detailed description of the solution, we give a quick overview highlighting its relationship with the NIS2 directives. The solution applies Distributed Ledger Technology (DLT) as a supporting tool for sharing of CTI and threat data across domains and with the whole ecosystem, including CSIRTs, as specified in multiple directives (e.g., directive such as 18 or directive 26). Immutable DLTs enable the auditability of cyberthreat sharing and actions, taking into account the sentiment of the directive, e.g., established in article 19. For the sharing of information, we consider privacy means like anonymisation and pseudonymisation, as suggested, e.g., in directive 121. Lastly, the solution makes heavy use of open-source security tools and open standards that, as argued in directive 52, can contribute to improved security through transparency, diversification, and interoperability.

6.2.1 Cyber-Threat Intelligence

IDSs, IPSs, SIEMs, Firewalls, among many others, are elements provided for securing networks. This aims to protect the system both proactively and reactively against possible intrusions. The operation of these security elements traditionally consists of analysing traffic and matching it with a database containing signatures and patterns of known attacks, to detect and block possible intrusions. However, since protecting a system depends on many different factors, even with a robust active security plan in place, the security of a system is not guaranteed. When presented with novel attack vectors, it is possible for these devices to trigger false negatives and allow an attacker to intrude. This is why additional methods must be incorporated to secure the systems, complementing the active security they already have.

Cyber intelligence is an information gathering and analysis activity aimed at identifying, tracking and predicting capabilities, intentions and activities of hostile actors in the cybersecurity domain. CTI can be defined as evidence-based knowledge, including context, mechanisms, indicators, implications and actionable advice, about an existing or emerging threat that can be used to make decisions regarding similar threats. CTI is comprised of attributes that give overall meaning to such a report - malicious IP addresses or hashes alone are not considered CTI, but grouped in context along with other information they can form part of a CTI report. One of the crucial elements of Cyber Threat Intelligence are Indicators of Compromise (IoCs). They are the most easily actionable attributes and the ones that most tools working with this information focus on. IoCs are widely used in applications such as Intrusion Detection Systems, web blockers, identification of compromised hosts or malware. These indicators can be easily related to other indicators that have occurred previously, taking advantage of big data analysis techniques on stored indicators.

To facilitate and accelerate the intelligence sharing process between organizations, it is necessary to structure the information so that an automated exchange can take place. Therefore, there are different standards for sharing threat intelligence and different platforms that favour automation through the exchange of this type of information. Survey [2] explains the development paths of the cyber-intelligence domain, its usefulness and use cases, the main standards that have been adopted in the industry, the most widespread platforms, and a comparison between them. The study describes standards like STIX (Structured Threat Information Expression) and TAXII (Trusted Automated Exchange of Indicator Information), as well as OpenTPX, MAEC, IODEF or VERIS. Among these, STIX is currently considered the de facto standard for describing Threat-Intelligence related data and the most widely used in Threat Intelligence Sharing Platforms (TIPs). As for the most widespread platforms for exchanging Cyber Intelligence Information,

```
{
  "Event": {
    "date": "2020-03-12",
    "threat_level_id": "1",
    "info": "testevent",
    "published": "false",
    "analysis": "0",
    "distribution": "0",
    "Attribute": [
      {
        "type": "domain",
        "category": "Network activity",
        "to_ids": false,
        "distribution": "0",
        "comment": "",
        "value": "test.com"
      }
    ]
  }
}
```

Figure 6.1. Example MISP event [4].

the survey highlights MISP, NC4 CTX, ThreatConnect, AlienVault, IBM X-Force Exchange, Anomali, and CrowdStrike among others. In particular, MISP (Malware Information Sharing Platform) [3] is a TIP for collecting, sharing, storing and correlating indicators of compromise (IOCs), targeted attacks, cyber intelligence, financial fraud, vulnerability information, and even counter-terrorism information. MISP is open-source software, and its objective is to foster the exchange of structured information between the computer security community, to support the daily work of incident and malware analysts. and malware analysts. MISP utilizes the MISP Core Format, a subset of JSON, to represent threat information, and events can be published in the platform using the STIX 1.1.1 and STIX 2.0 specifications.

However, CTI in many cases can contain information that should only be transmitted to trusted stakeholders or not transmitted at all, such as Personally Identifiable Information (PII) which is irrelevant to create situation awareness. In wrong hands, this information can lead to a successful attack that could severely damage the reputation of the stakeholder. Therefore, a trust mechanism and a way to protect sensitive data before sharing must be implemented in almost any CTI environment. In recent years, the importance of this mitigation measures has been reflected in various research works. For instance, [5] presented a privacy-preserving protocol for threat intelligence sharing, based on the collaborative training of a decision tree classifier, and exchanging training data between organizations in a private way using homomorphic encryption. Another approach is shown in [6], where authors propose a framework that allows different providers to share their information according to privacy requirements contained in a Data Sharing Agreement (DSA) and apply data mining techniques to extract additional knowledge from it.

In order to increase the privacy level of the raw CTI data itself shared through a platform such as MISP, we focus on applying Privacy Preserving Techniques (PPTs) on the events. These events will thus have masked sensitive data with the main objective of maintaining a satisfactory level of meaning while simultaneously increasing privacy and avoiding risks related to identity disclosure, attribute disclosure or membership disclosure. More specifically, non-perturbative masking techniques such as generalization and suppression, perturbative masking techniques such as noise addition through Differential Privacy (DP), and finally PPTs such as k-anonymity and l-diversity are described and considered.

6.2.2 Data Anonymisation Techniques

There are some risks involved in CTI Sharing that make organizations often reluctant to share information on such platforms. This is partly because they feel that revealing information about intrusions could damage their reputation. Furthermore, this information can also carry IP addresses, email accounts, names, and other PII which can be used against the sharing organization if it falls into the possession of an attacker and the organization has not addressed security issues in their system yet. Particularly relevant with PII is the fact that this information can potentially fall into the hands of a dishonest partner, who can then make fraudulent use of this information. Data anonymisation/pseudonymisation techniques aim to solve these issues by transforming sensitive data in such a way that the new data cannot be used by an attacker to extract sensitive information about the organization, or alternatively the extent of the sensitive information conveyed by the data is reduced enough to render it useless to the attacker. Data pseudonymisation applies a reversible process and allows authorized parties to retrieve the hidden information by using identifiers or pseudonyms, while data anonymisation is used to irreversibly modify the data to guarantee the loss of sensitive information.

For the anonymisation of these identifiers' values, there are privacy preserving and data transformation techniques such as suppression, generalization, sampling, k-anonymity, l-diversity, t-closeness, d-presence, etc. These techniques attempt to minimize data risks related to identity disclosure (the ability of an adversary to correctly associate an individual within a dataset), attribute disclosure (the ability of an attacker to infer the value of an attribute due to the distribution of attribute values in the table), and membership disclosure (the ability of an attacker to be able to determine, with very high confidence, whether or not a particular individual is present in the dataset). They are derived from Statistical Disclosure Control (SDC) [7]. SDC, also known as Disclosure Avoidance, is the discipline that manages the balance between the privacy of respondent data and the usefulness of this data for research purposes. In the following, we give a brief description of some of

	ZIP Code	Age	Disease
1	47677	29	Heart Disease
2	47602	22	Heart Disease
3	47678	27	Heart Disease
4	47905	43	Flu
5	47909	52	Heart Disease
6	47906	47	Cancer
7	47605	30	Heart Disease
8	47673	36	Cancer
9	47607	32	Cancer

Table 1. Original Patients Table

	ZIP Code	Age	Disease
1	476**	2*	Heart Disease
2	476**	2*	Heart Disease
3	476**	2*	Heart Disease
4	4790*	≥ 40	Flu
5	4790*	≥ 40	Heart Disease
6	4790*	≥ 40	Cancer
7	476**	3*	Heart Disease
8	476**	3*	Cancer
9	476**	3*	Cancer

Table 2. A 3-Anonymous Version of Table 1

Figure 6.2. Example of k-anonimity applied to a table of patients[1].

the most widespread mechanisms, which have been implemented in the solution described in this chapter.

K-anonymity is a data protection model/dataset property proposed by Latanya Sweeney in [8]. Its objective is to minimize the risks of re-identification of individuals within anonymised datasets, stating that for a data publication to meet the k-anonymity condition, the information about any one individual in the dataset must be indistinguishable from at least k-1 other individuals in the same dataset. k-anonymity focuses on identity-disclosure by guaranteeing that k records hold the same values in a set of attributes, so that if an attacker had another external database containing data of individuals from the dataset, he would not be able to relate one of them within the privatized dataset to a degree of accuracy of no more than k individuals.

L-diversity [9] improves the k-anonymity proposition by aiming to decrease the attribute-disclosure risks that k-anonymity is susceptible to. To achieve this, it defines a new type of attribute, called sensitive, which contains sensitive data about an individual that is less identifiable than quasi-identifier data but whose distribution in the dataset can reveal information about it. An equivalence class is said to achieve l-diversity if there are at least l well represented values for a sensitive attribute. A dataset has l-diversity when all its equivalence classes satisfy l-diversity. l-diversity defines the concept "well-represented" in three separate ways, resulting in three alternative conditions. The first one defines "distinct l-diversity", where a well-represented element is one that appears at least once. The second one defines "entropy l-diversity", which places a lower bound on the entropy level of the element frequency. Finally, the "recursive (c, l)-diversity" variant seeks to guarantee that the most frequent value within an equivalence class appears less, and the least frequent value does not appear so little.

1. Source: https://blog.csdn.net/scuLVLV/article/details/71077689.

Despite improving the weaknesses of k-anonymity, l-diversity continues to present vulnerabilities related to attribute disclosure to a lesser extent. This is why T-closeness [10] is introduced, to overcome these risks and to introduce a model in which semantic meaning of values is considered. The way t-closeness works focuses on calculating the distance between the distributions of each equivalence class with respect to the rest of the dataset, to avoid problems arising from an uneven or skewed distribution of sensitive values.

Differential privacy [11] establishes a rigorous mathematical definition of the concept of privacy that guarantees the probability that the result of an anonymisation process applied on a dataset is virtually unchanged if any individual in the dataset is removed or added. This model attempts to provide privacy by perturbing the data through the addition of small amounts of noise.

6.2.3 Inter-Ledger Approach for Verifiable Data Sharing

The integration of Distributed Ledger Technologies (DLTs) has been a catalyst for significant advancements in the secure and verifiable exchange of data across various domains. However, in the context of sharing security data, it is usually necessary to connect many different security domains, and particularly make most of the data as available as possible. Thus, it is necessary to adopt DLT solutions that ensure interoperability of various instances and that enable global outreach for data sharing mechanisms. While there exist multiple approaches to this challenge, we in this chapter, we focus on the conceptualised concept of "Blockchain of blockchains," so that it is possible to facilitate smooth and secure transitions through smart contracts (instantiated as chaincodes) among different domains, tailored to the needs of the CTI sharing ecosystem.

The approach is based on the the creation of an inter-domain DLT instantiated as an inter-domain channel within Hyperledger Fabric.[2] This channel serves as the foundation for inter-domain communication. Essential elements include participants from various domains, their respective permissions, and endorsement policies. For example, each domain contributes at least one peer to this channel, and it ensures a majority endorsement policy. This strategy provides robustness and allows for a decentralized governance structure where domains participate in the sharing process as equals. Additionally, this enables the direct use of smart contracts across different channels within the same ecosystem. This approach boosts transaction efficiency and enhances security, interoperability, and scalability.

The inter-ledger approach is designed to address each domain's specific needs for data exchange. Using Hyperledger Fabric, each domain maintains private ledgers

2. https://github.com/hyperledger/fabric.

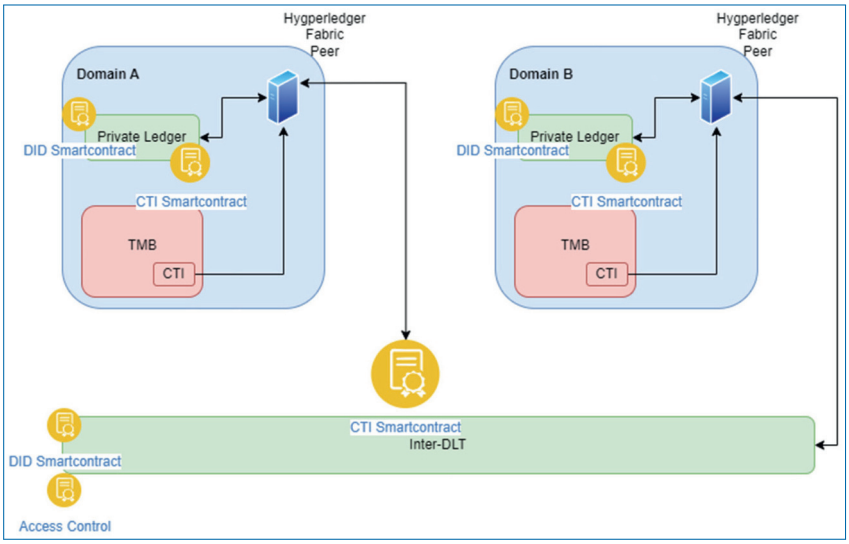


Figure 6.3. Inter-DLT deployment and functional interaction overview.

while also participating in an inter-domain channel. In this way, domains can take advantage of the adequate DLT capabilities according to the characteristics of the data sharing process in mind. For instance, the CTI sharing solution described in this chapter may create private records for auditability while also producing data sharing events that reach the whole ecosystem and relevant stakeholders through the inter-DLT. Smart contract invocations across channels facilitate communication, with data traceability linked to genesis hashes. A gRPC server supports efficient data management by directing it to appropriate channels, ensuring transparent and configurable operations for components handling data in the domain.

The diagram in Figure 6.3 provides a visual representation showcasing the deployment and interaction between different security domains using the inter-DLT framework. The diagram features Domain A and Domain B, each with their private ledgers within Hyperledger Fabric. Within these domains, various smart contracts provide functionality related to the DLT, such as the management of Decentralized Identifiers. Particularly, the CTI smart contracts handle Cyber Threat Intelligence data as part of the security information sharing process. As detailed in the following, this enables an auditable, secure and widespread solution for sharing CTI data across different security domains.

6.2.4 The CTI Sharing Agent

Figure 6.4 shows the detailed description of the CTI Sharing Agent as part of a Trust Manager and Broker, as well as the main communication flows involving the subcomponents.

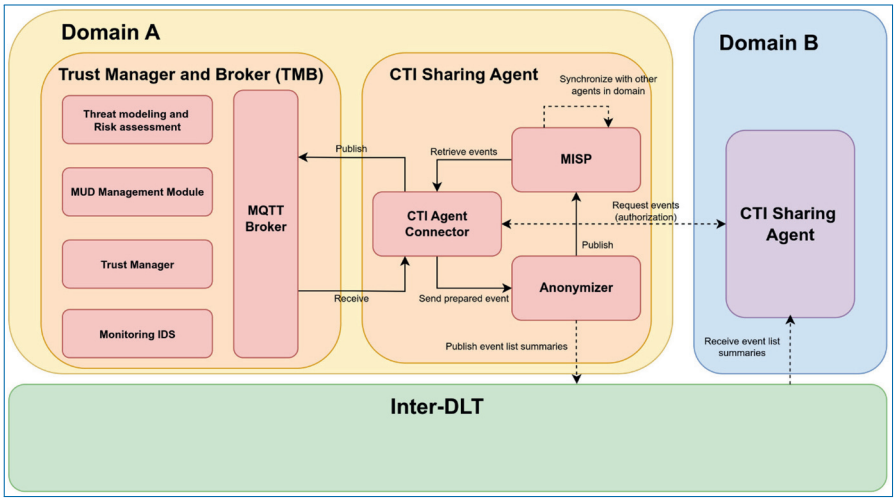


Figure 6.4. CTI sharing agent.

The CTI Sharing Agent consists of:

CTI Agent Connector: This component is in charge of communication to other components of the security and trust framework in the domain. Particularly, our instantiation includes communication through the Trust Manager and Broker using a publish/subscribe MQTT approach. The connector will be the intermediary that receives threat alerts coming from the Monitoring components. It also retrieves events from the CTI sharing platform for later forwarding them to relevant components through the broker. Lastly, it governs the authorization of sharing processes with CTI Sharing Agents from other domains.

Anonymizer: Receives a threat-related event coming from the CTI Agent Connector and applies anonymisation techniques that are specified in the privacy policy file related to that type of event. After the anonymisation process the resultant event is published on the domain's instance of the CTI sharing tool, e.g. MISP. Additionally, the DLT is used for auditability and signalling of this publication process. The techniques implemented in this component are generalization, suppression, k-anonymity, l-diversity, and noise addition by differential privacy.

MISP: An instance of the MISP platform for publication and sharing of security events. It acts as the repository of threat events received. Its synchronization capabilities are useful to keep instances within a security domain synchronized (i.e., for distributed instantiations of the security framework to provide resilience and scalability). What is more, it can be synchronized with external MISP instances, such as those of manufacturers or public CSIRTs/CERTs, so that the database of threats

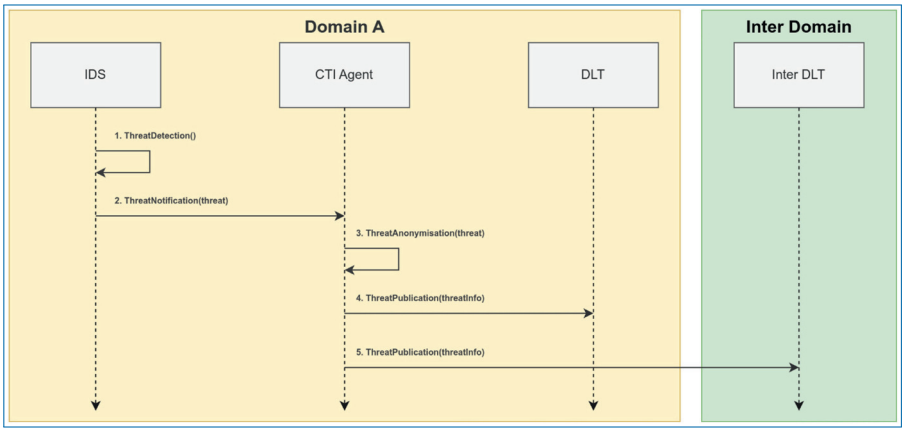


Figure 6.5. Threat detection and sharing.

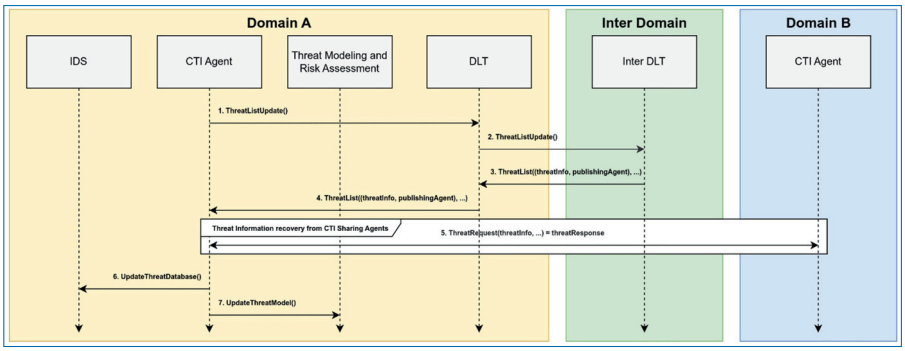


Figure 6.6. Threat list update from CTI sharing.

reaches relevant people in the security sector and improves the security of the overall ecosystem.

Figures 6.5 and 6.6 show diagrams for the threat detection, sharing and threat list update processes. The first flow starts when the monitoring tools detect a threat, at which point they notify the domain's CTI Sharing Agent with detailed information about the threat. The sharing agent then transforms the event data using the Anonymizer to obfuscate sensitive information. The anonymized event is then made available, and the DLT infrastructure is used to register this event. Particularly, the domain's DLT is used for registering the publication of the threat so as to allow for auditability of the process. Additionally, the inter-DLT is used as a means to register and make available summaries of the published information to other domains, so that relevant stakeholders are notified of the event.

Later, a different security domain may desire to update its threat information database to improve its security. Then, its CTI Sharing Agent queries the inter-domain DLT (through a smart contract) for recorded relevant threat information.

From this query, the agent receives threat event records, which include both threat information as well as contact information for the sharing agent that originally reported the threat. The domain's sharing agent then initiates a process of threat evaluation where, based on the specific domain properties, it selects those threat events which are pertinent to the domain and queries the external sharing agent for additional information. This threat exchange is subject to an authorisation process overviewed by the publisher CTI agent, according to information relevant in the application scenario. For instance, this may include checks of the requestor's identity or its inclusion on trusted entity lists maintained in the own DLT infrastructure. This process is repeated for every relevant threat event. Finally, once all the threat information is gathered, the domain's sharing agent forwards the updated information to both the IDS and the risk assessment module in order to update the relevant databases.

6.3 Manufacturer Usage Description Files

The large-scale nature of IoT deployments, the heterogeneity of device characteristics, the distributed and replicated nature of deployments, or the involvement of people with low or no technical background result in important needs for certification and security configuration for automatized lifecycle management. On the one hand, this topic includes the challenge of easing and automatizing the deployment of devices in a secure way. On the other hand, it is necessary to go beyond those initial steps, covering the whole lifecycle of the device including the necessary adaptations as security contexts change, because of configuration changes, addition of new devices, or the arrival of new threats.

Our proposed solution leverages Manufacturer Usage Description (MUD) files to address these challenges, integrating their use with other components in a domain's security and trust framework. This integration enables the discovery and parsing of behavioural information for devices enrolling in the framework, which is instrumental in enhancing analytical models, trust evaluations, and domain-specific monitoring. Furthermore, we account for the dynamic nature of security requirements by allowing updates to MUD files and incorporating new security information through threat-specific MUDs. In a holistic solution, by incorporating monitoring and cyber-threat intelligence sharing tools, our approach facilitates the exchange of security measures and mitigation strategies among relevant stakeholders, in alignment with guidelines such as those outlined in the NIS2 directive [1]. Overall, the outcomes of our research provide a crucial mechanism for ensuring secure lifecycle management of IoT devices, from their initial security configuration, during deployment and registration, to their eventual

decommissioning, thereby addressing a significant business and technical challenge in the IoT domain.

6.3.1 MUD Standard

The heterogeneity of IoT environments, spanning from dynamic home settings to industrial IoT (IIoT), and the wide variety of devices using different technologies and communication protocols, introduces several challenges. These include ensuring compatibility across devices, managing diverse communication patterns, and addressing security vulnerabilities that vary depending on the specific environment and device capabilities. 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. Despite these complexities, it remains important to understand or predict the expected behaviour of these devices in order to effectively detect and mitigate security threats. Given these challenges, standardization in identifying and managing device behaviour is essential for tackling the unique demands of these varied IoT ecosystems.

In this direction, Manufacturer Usage Description (MUD) standard was published in 2019 by the Internet Engineering Task Force (IETF) [12]. 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) [13] to model network restrictions, and JavaScript Object Notation (JSON) [14] as the serialization format. Figure 6.2 shows an example of a MUD file, where we can appreciate the two core containers of the specification. The first "ietf-mud:mud" container provides information about the MUD file itself, such as its URL, how long it should be cached and information about the device. Additionally, the MUD model is extended with the "ietf-access-control-list:acls" container based on [15], including network communication restrictions to configure the forwarding behaviour in the device, e.g., representing communications with certain hosts, ports or protocols. An example of a MUD file showing these characteristics can be found in Figure 6.7.

Additionally, the standard introduces a comprehensive architecture designed to manage MUDs, and mechanisms for its implementations. Also, it defines several extensions for the transmission mechanism of the MUD such as an DHCP option, an LLDP extension, a X.509 extension that links device's identity and MUD profile together, signature verification and extensibility considerations.

Figure 6.8 shows said architecture, where the device emits the MUD URL (through one of the capable protocols) to router or switch, which forwards the

```
{
    "ietf-mud:mud":{
       "mud-version":1,
       "mud-url":"https://localhost:9443/examples/lightbulb2000.json",
       "cache-validity":48,
       "systeminfo":"The BMS Example Light Bulb",
        ....
       "from-device-policy":{
          "access-lists":{"access-list":[{"name":"mud-76100-v6fr"}]}
       }
    },
    "ietf-access-control-list:acls":{
       "acl":[
          {
             "name":"mud-76100-v6to",
             "type":"ipv6-acl-type",
             "aces":{
                "ace":[
                   {
                      "name":"cl0-todev",
                      "matches":{
                         "ipv6":{
                            "ietf-acldns:src-dnsname":"test.example.com",
                            "protocol":6
                         },
                         "tcp":{
                            "ietf-mud:direction-initiated":"from-device",
                            "source-port":{
                               "operator":"eq",
                               "port":443
                            }
                         }
                      },
                      "actions":{
                         "forwarding":"accept"
                      }
                   }
                ]
             }
          }
       ]
    }
}
```


Figure 6.7. Example of a MUD file [12].

URL to the MUD Manager and then retrieve the MUD 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 [1, 16].

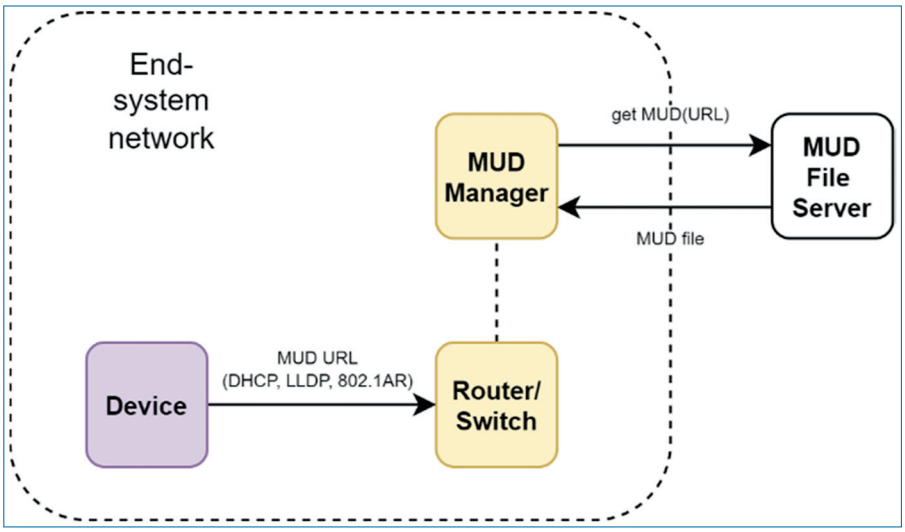


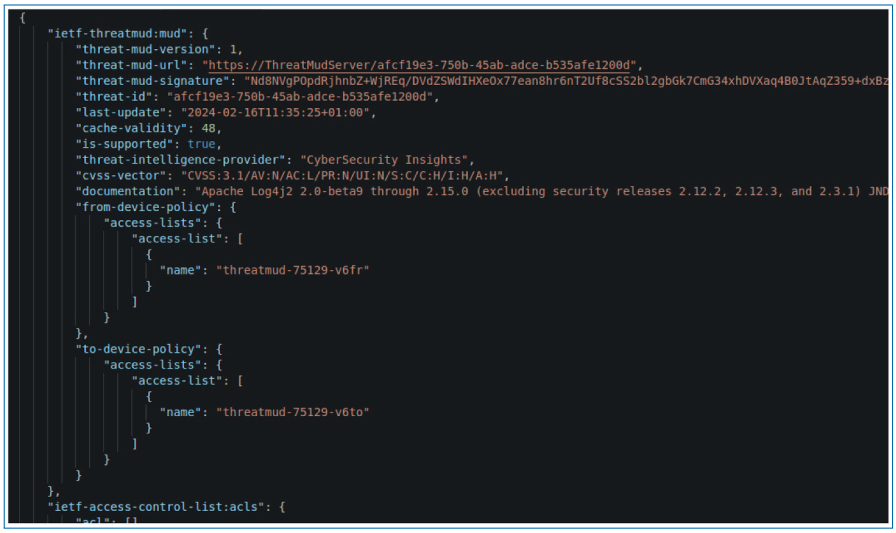
Figure 6.8. MUD Standard Architecture [12].

6.3.2 Threat MUD Proposal

The task of securing environments that incorporate IoT devices extends far beyond the initial installation. Numerous vulnerabilities and potential attacks often emerge during the operational lifecycle of these devices, making continuous monitoring and protection essential. For example, in 2022 alone, over 25,000 vulnerabilities were identified [17]. Manufacturers often face challenges in addressing these vulnerabilities promptly, as updates require navigating a complex process. Additionally, the involvement of third-party services can further complicate timely resolutions. Although MUD files can be updated, this method fails to address the majority of vulnerable scenarios. In these circumstances, security-information sharing systems play a crucial role by enabling the rapid, collaborative exchange, analysis, and mitigation of vulnerabilities or attacks, even before a patch becomes available. In fact, the importance of cyber-threat information sharing to strengthen cybersecurity capabilities has been a significant focus of ENISA's NIS2 Directive [1].

In this context, the NIST introduced the concept of Threat Manufacturer Usage Description (Threat MUD) [16], aimed at facilitating the sharing of vulnerability information and corresponding mitigations. Threat MUD builds upon the MUD standard maintaining a similar structural foundation. Despite this close connection, Threat MUD stands as an independent concept focused on mitigation strategies. However, NIST provides only partial guidance on the Threat MUD model, leaving some aspects undefined but still framed within broader guidelines.

In the following section, we examine the Threat MUD model as described in [18]. The NIST guidelines and this work serve as a foundation for our approach,



```
{
    "ietf-threatmud:mud": {
        "threat-mud-version": 1,
        "threat-mud-url": "https://ThreatMudServer/afcf19e3-750b-45ab-adce-b535afe1200d",
        "threat-mud-signature": "Nd8NVgPOpdRjhnbZ+WjREq/DVdZSWdIHXeOx77ean8hr6nT2Uf8cSS2bl2gbGk7CmG34xhDVXaq4B0JtAqZ359+dxBz
        "threat-id": "afcf19e3-750b-45ab-adce-b535afe1200d",
        "last-update": "2024-02-16T11:35:25+01:00",
        "cache-validity": 48,
        "is-supported": true,
        "threat-intelligence-provider": "CyberSecurity Insights",
        "cvss-vector": "CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H",
        "documentation": "Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) JND
        "from-device-policy": {
            "access-lists": {
                "access-list": [
                  {
                    "name": "threatmud-75129-v6fr"
                  }
                ]
            }
        },
        "to-device-policy": {
            "access-lists": {
                "access-list": [
                  {
                    "name": "threatmud-75129-v6to"
                  }
                ]
            }
        }
    },
    "ietf-access-control-list:acls": {
        "acl": [
```


Figure 6.9. Extract of a Threat MUD file.

though we adapt certain elements, such as differences in the way the flows are implemented in the solution's architecture.

A key aspect to highlight about Threat MUD is the contrast in purpose with respect to the original MUD standard. Unlike regular MUDs, which describe the expected behaviour of a device, Threat MUD shifts its focus from the device to the threat itself, specifying network communication rules for sites linked to the identified threat. As a result, Threat MUD does not need to be tied to a particular device. Furthermore, instead of being developed by the manufacturer, the creation of Threat MUD files may fall under the responsibility of different threat intelligence providers and stakeholders, emphasizing its role in threat-centric rather than device-centric mitigation strategies.

As with the standard MUD, the Threat MUD model is structured into two key modules (c.f. Figure 6.9 for an example). The first module provides information about the Threat MUD itself, maintaining fields similar to those in the original MUD standard, such as version, URL, cache-validity, and signature. However, several fields are adapted to align with the threat-centric focus. For instance, the "manufacturer name" is replaced by "intelligence provider," and "device model name" is substituted with "threat name." Also, fields that are no longer relevant, such as those related to the specific device (e.g., system or firmware information), are removed. Additionally, new fields are introduced, such as cvss-vector, related to the CVSS (Common Vulnerability Scoring System) and documentation related, to offer detailed information about the threat and its mitigation strategies.

The second module is related to Access Control Lists (ACLs) that define the conditions and restrictions for network communication. Unlike the traditional MUD,

where these ACLs are tailored to a specific device, Threat MUD applies these configurations more generically, given its focus on threat mitigation rather than device behaviour.

6.3.3 Shortcomings and Extensions

A notable limitation of the standard is the inability to implement security constraints beyond the network layer. This limitation is significant in this scope because it restricts the expressiveness of the MUD files, thereby hindering their potential to address a broader spectrum of security concerns. The definition of rules or conditions that extend beyond this layer could help to add substantial expressive capabilities that enhance the identification and mitigation of a wide range of attacks, including those targeting the application layer such as slow DDoS attacks.

In the following, we describe several potential extensions to enhance the expressiveness of MUD files. These extensions may cover characteristics known at design time or during updates, which the original MUD standard does not address. Additionally, they could be useful for defining mitigation actions through Threat MUDs, particularly in the context of this work.

- Application layer protocols significantly influence how devices should communicate within a network. In the context of the MUD standard, this selection directly relates to the application role of the devices, affecting the device's network behaviour. Allowing the specification of protocol restrictions at the application layer not only influences network layer functionality but also enables more granular control over the device within any given infrastructure.
- Exposed resources offered by the device, like HTTP/CoAP endpoints to retrieve information, which could be used both for discoverability of devices and to consider the expected behaviour for threat and risk models.
- Cryptographic algorithms, specify the supported algorithms of the device and its preferences or add restrictions to the communications. This could be useful for both communications (establishing secure channels) and higher-level needs of the infrastructure, like determining which cryptographic algorithms are preferred or supported, and the appropriate security level for managing identities, e.g. digital signatures that either or not support zero-knowledge proofs depending on the capabilities of the device.
- Known vulnerabilities associated with the device, through continuous MUD file updates, following the CVE entry format and additional details like CVSS scores, which could be associated with mitigations measures or restrictions to reduce the threat risk.
- Software and firmware restrictions designed to support and enhance automatic software update and deployment processes, such as defining a

minimum software version for optimal operation or establishing an update as part of a mitigation action.

Additionally, as identified in [19], the application of the MUD standard confronts several operational challenges, and extension points are being researched in many works on the topic.

Firstly, the standard does not address the dynamic nature of MUD files nor protocol for updates during deployment lifecycle. Another challenge is the necessary authentication during MUD acquisition so that the infrastructure may know that the MUD URL corresponds to the device sending it, and not to some other entity. Moreover, the described method in the standard for obtaining MUD files, typically via requests to switches or routers, may not be suitable for all operational environments. Lastly, the standard currently lacks detailed guidelines for applying MUD in specific sectors such as the Industrial Internet of Things (IIoT), where advanced functionalities and extended MUD models are crucial. This work aligns specifically with these areas of concern, in addition with the extension of the MUD file expressivity.

6.3.4 The Adapted MUD Solution

Within the proposed security and trust framework, the MUD approach is integrated as a key source of behavioral information for devices that enroll a domain. As in the standard, MUD file servers are located outside the framework domains, typically hosted by the device manufacturers. MUD components within each domain can communicate with these external servers to retrieve the relevant MUD files, for instance through a simple REST API. However, different approaches of retrieval could be used such as periodic requests or publish/subscribe models, enabling additional interactions such as MUD file updates. Similarly, threat MUD files will be controlled by external entities, such as cybersecurity agencies. On the other hand, the main functional components of the MUD standard and Threat MUD proposal will be active in each security domain, as part of the security and trust framework (e.g., the Trust Manager and Broker—TMB—component in the ERATOSTHENES solution).

Figure 6.4 shows the detailed instantiation of the MUD components and an overview on their interfaces. Note that the MUD management module is an aggregation of subcomponents, incorporating both the standard MUD and Threat MUD manager functionalities, alongside a Translation module.

- **MUD Manager:** the component is in charge of receiving MUD file resolution requests according to the URL associated to a device, e.g. through an MQTT broker. It then utilizes the URL to retrieve the MUD File from the

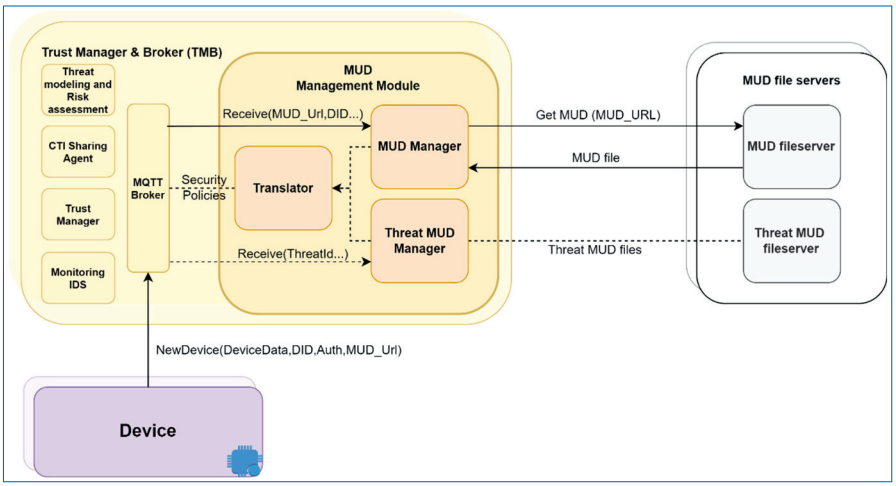


Figure 6.10. Instantiation of the MUD management module.

MUD File Server. Once done it returns the MUD File to the Translator module.

- **Translator:** transforms MUD and Threat MUD files into applicable security policies, ensuring seamless integration with the broader domain management processes. Receives the MUD & Threat MUD File related to a device and converts its conditions and restrictions into corresponding actions in MSPL format. Once translated it adds to the MSPL the information of the device or threat to which it's addressed and sends it to the rest of the infrastructure through the corresponding topic.
- **Threat MUD Manager:** similar to the MUD Manager, collects the Threat MUD File tied to the specific threat identifier received after a trigger from security components. After retrieving the file from the MUD File Server, it forwards it to the Translator module for its translation and sharing.
- **MUD File Server:** this service belongs to the manufacturer premises and contains the MUD files associated to the manufacturer devices. It's a single server for each manufacturer, leading to several servers[3] according to the number of manufacturers in the ecosystem.
- **Threat MUD File Server:** stores Threat MUD files associated to threat identifiers. Unlike the MUD File Servers, in this case the server does not belong to the manufacturers but acts as a common access point for entities within the framework serving mitigation strategies related to the threats present in

3. From a practical point of view this server may instead be distributed to improve scalability and resilience of the solution.

the ecosystem. Various security actors may control Threat MUD servers to create and share policies.

The main contrast with the standard MUD architecture in Figure 6.10 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 domain enrolment phase to get the MUD URL from the device. The integration in such flow, and within the complete ecosystem, also allows the exploration of further improvements for the MUD flow.

For instance, while the server-side communication can be simply secured with standard mechanisms (e.g., HTTPS), there is a security risk in the standard specification regarding the potential spoofing of a MUD URL by the device. In the application-level integration proposed, the authentication of the URL during MUD file obtention can be tackled through the domain's identity framework. Specifically, the manufacturer will associate the URL to the root of trust for identification of the device, for instance through the use of Physical Unclonable Functions as a root identity. When receiving the URL and retrieving the file, it is then possible to check that the device possesses the proper root identity, otherwise rejecting its claim. Furthermore, the extension of the MUD model and language with higher level concepts, like software updates or cryptographic parameters restrictions, is even more relevant through this layer abstraction. Nonetheless, the MUD solution will not ignore networking elements by working at the application level. As MUD files will be associated to devices that are relevant to an application domain, the networking rules will also be relevant to the specific deployment and must simply be translated to the appropriate enforcement measures.

Additionally, in the specific instantiation within ERATOSTHENES, the inclusion of the MUD elements in the TMB allows simple interactions with other domain components relevant to the application of its functionality and the management of the lifecycle of devices. For instance, as previously mentioned, the information provided in the MUD file can be leveraged by the Threat Modelling and Risk Assessment (TMRA) component to influence the trust level assigned to a device. Additionally, monitoring systems such as Intrusion Detection Systems (IDS) can utilize MUD data to understand the expected traffic patterns for each device. This allows IDS to more effectively identify deviations from expected behaviour and respond swiftly to potential threats. These interactions open various enforcement possibilities, not only at the network level through direct policy application, but also through indirect control by continuously evaluating device behaviour and adjusting its trust level based on its interactions within the domain. This approach strengthens the overall security posture and response capability within the network.

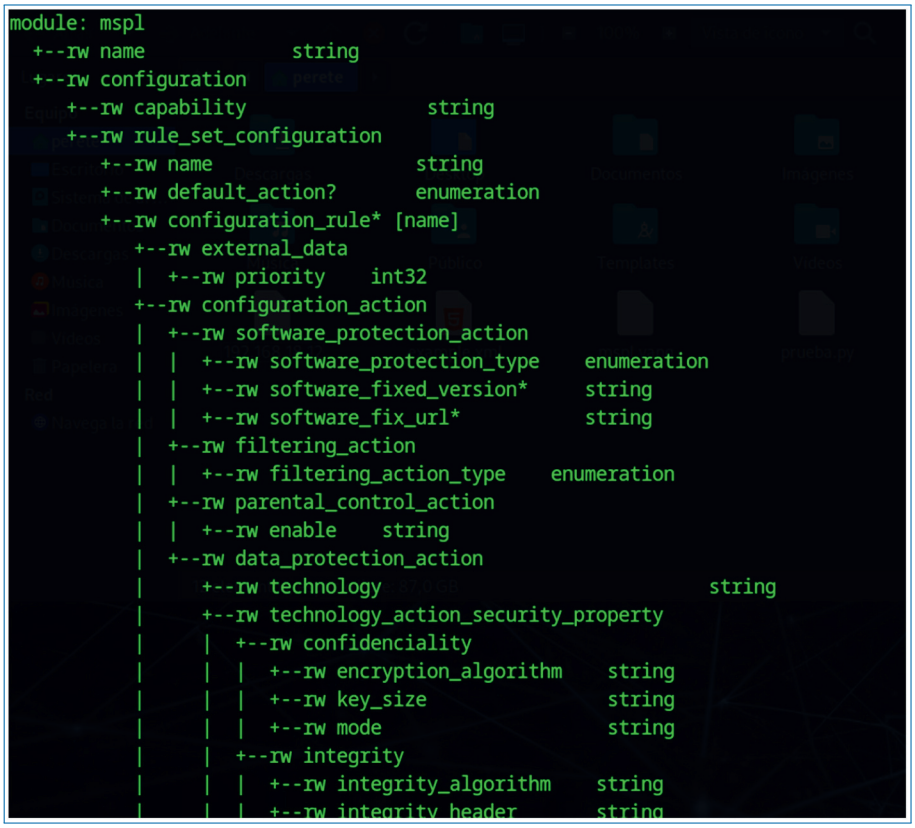


Figure 6.11. Extract of the MSPL YANG model tree.

To facilitate these communications, we rely on the Translator component, which converts MUD files into intermediate security policies. The policies provide a set of actions suitable to the most common applicable security settings. For this medium level of abstraction, we work with the Medium-level Security Policy Language (MSPL) [20]. The MSPL language specification is based on a YANG model, allowing it to be encoded in XML or JSON formats, providing flexibility in how MUD information is utilized (c.f. Figure 6.11).

This flexibility ensures that the data contained in MUD & Threat MUD files can be efficiently applied across different components. Assets in the security domain will have access to the medium-level policies and take advantage of them for improving their performance in their tasks, 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 authorization enforcement policies may be derived by PDPs.

6.4 Lifecycle Security Through Information Sharing

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, with different phases along lifecycles. 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. 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 is carried out during the operation of a security domain. This knowledge can be taken advantage of in security domains with tools that can take into account and apply the threat information and mitigation actions defined in these files.

On this note, a key topic of the NIS2 directive by ENISA [1] is the importance of providing mechanisms for privacy-preserving and secure CTI sharing. As described above, the sharing processes should include relevant entities such as manufacturers, vulnerability databases, or cyber-response teams. With such solution, it is possible to improve detection, monitoring and assessment of threats and security domains. What is more, it enables widespread awareness of threats, resulting in a more secure ecosystem, for instance through the creation of mitigation policies. With this in mind, the tools and processes introduced in this chapter can be used to enhance security along the lifecycle of devices and IoT ecosystems, as we discuss in the following.

6.4.1 Secure Deployment of IoT Devices

A critical security event during the lifecycle of devices is their deployment within the domain of operation. During this process, several checks must be made to ensure the safety of the domain, from the identification of the device to the initial evaluation of its characteristics in a zero-trust approach. These processes are further explored in other chapters in the book. Here, we focus on how the security information coming from the MUD solution can be used to enhance the initial bootstrapping and enrolment process, acting within the context of a holistic solution. In this sense, we part from the MUD management solution based on an application-level interaction with the trust services of a security domain.

Figure 6.12 shows an extract of an enrolment process in such an integrated ecosystem, highlighting the interactions related to MUD files. The process takes

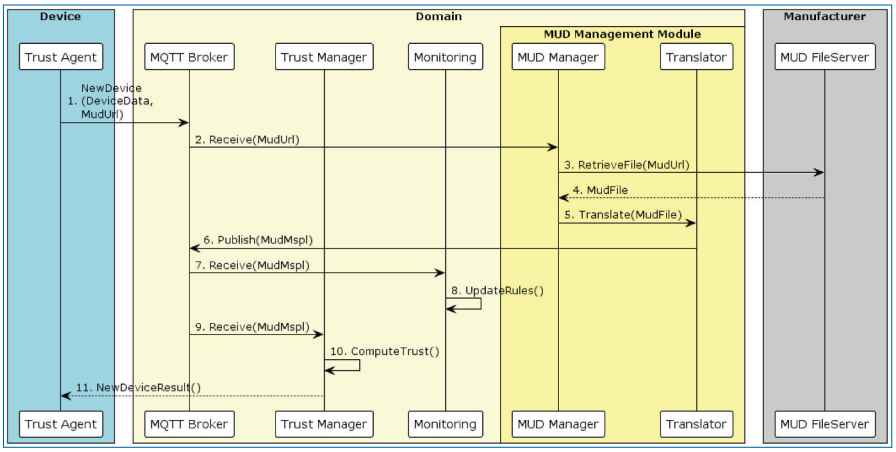


Figure 6.12. MUD Management during bootstrapping and enrolment.

advantage of the enrolment of a device within the trust framework of a domain to kickstart the MUD retrieval process. This MUD file contains information about expected behaviour, known vulnerabilities and general security data about the device. The MUD Management Module then translates the file into security policies in the MSPL language, which are shared throughout the security and trust framework of the domain. Thus, actions adapted to the security context can be taken to ensure a safe onboarding process. As related in the figure, it is possible, for instance, to take advantage of other tools described in this book, improving the monitoring capabilities of the detection systems and adapting the evaluation of the trustworthiness of the device according to the gathered information.

During the deployment of the device, its identification and the provisioning of identity in the domain are important steps for the security of the process and later operation of the device. Of course, this also affects the application of security measures according to device information, as it is necessary to ensure that the retrieved MUD file is adequate for the device. To do so, the MUD URL associated to a device can be linked to the root of trust for its identification process, which is set up by the manufacturer during the initial provisioning of the device. For instance, the MUD URL can be bound to a Physical Unclonable Function output, achieving strong unforgeability guarantees.

A simple extension of this secure deployment process can be done to account for the changing nature of security aspects throughout device lifecycles. MUD files are stored within the MUD Manager during their lifetime, and once expired they are retrieved again, checking for changes and if necessary, updating the MSPL policy associated with the device so that the other tools can take advantage of the new information. In some deployments, the fileserver may instead allow a

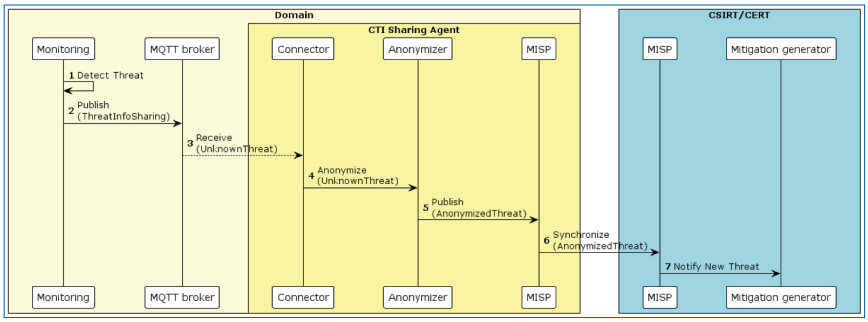


Figure 6.13. Threat detection and sharing.

subscribe-publish approach, so that immediately after a new version of a MUD file is generated, it will reach the relevant MUD managers.

6.4.2 Threat Sharing and Mitigation

While the previous section focuses on the aspects related to adopting initial and updated security configurations of a device within a domain, in the following the focus within the lifecycle security is on the functionalities achieved through the sharing of CTI information. As previously discussed, this is a critical process for ensuring the security of both the devices and the domain throughout their entire lifecycle, as well as enabling appropriate responses to information received in order to implement necessary mitigation measures.

Figure 6.13 gives a general overview of a threat sharing process that involves a new threat for which mitigation actions will be generated. First, a threat or attack is detected through the monitoring tools within a security domain, and the CTI information reaches the CTI Sharing Agent. The agent parses the data and applies anonymization techniques to protect sensitive data. The anonymized information about the threat is then shared outside the security domain, through sharing tools like MISP. The data will reach relevant cybersecurity actors, such as CSIRT or CERT teams, for instance of a device manufacturer The identified threat may include new information that can be analyzed by the appropriate tools or personnel, initiating the process of generating a suitable mitigation response. In some cases, the threat may already be known, including mitigation actions related to it, so that the mitigation process automatically starts from the domain tools.

In case the new mitigation is generated, it can be published as a Threat MUD file. A notification of the creation of the new file is sent through the CTI sharing network, so that it can reach any security domain where the threat is relevant. Particularly, following the previous example, the CTI Sharing Agent of the security domain receives the information and notifies through the broker about the existence of the new Threat MUD file associated to the previously detected threat.

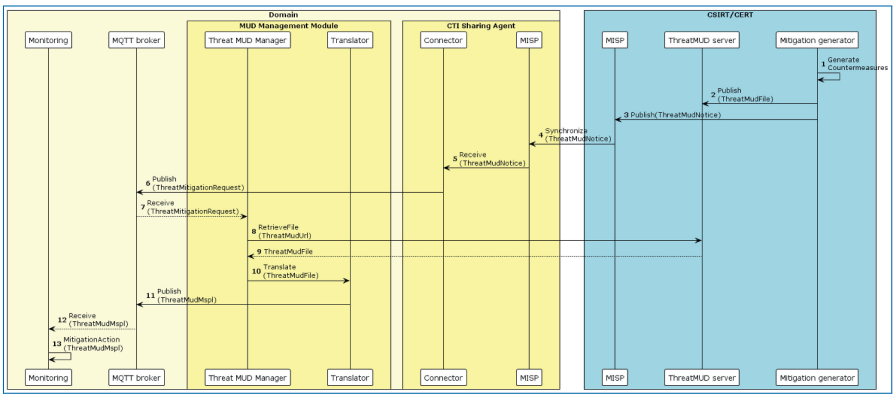


Figure 6.14. Threat mitigation sharing and enforcement.

The Threat MUD Manager receives the notification and retrieves the file, then forwards it to the Translator. The Translator generates an MSPL language policy based on the file's contents, tailoring the mitigation to the specific domain (e.g., adjusting IP addresses to those relevant within the domain). This MSPL policy contains mitigation actions for the specified threat. The policy is then published on the MQTT broker, so that its enforcement can take place according to the necessary actions. For instance, the flow in the image includes a mitigation action at the network level, so that the monitoring and protection tools can execute the policy by adding rules for detecting and stopping specific traffic. Other mitigation actions may be used depending on the context of the threat and the domain, such as having the digital twin fleet management tools issue a software update after a compromise.

These procedures are fully automated, except for specialized personnel who may be needed to develop appropriate mitigation actions. This automation enables the ecosystem to respond to cybersecurity incidents throughout the entire lifecycle of the devices and security domains. The approach is flexible, allowing various components to address different situations, including known threats and their mitigations. Moreover, these processes enhance the overall security ecosystem by ensuring that relevant CTI information—covering existing threats and their mitigations—reaches all stakeholders. Additionally, the tools facilitate the adaptation of measures to specific domain scenarios by considering the security context during information parsing and utilizing generic policies that can be modified by various tools for different enforcement procedures, such as traffic management and automated software upgrades. For instance, threat modelling and risk assessment tools can dynamically incorporate both the existence of a threat and the implementation of a mitigation action, allowing models to evolve in response to changing conditions. This adaptability can influence the perceived trustworthiness of participants and inform decisions on whether to authorize their actions.

Acknowledgements

The research leading to this work has been co-funded by the European Union's Horizon 2020 research and innovation program under Grant Agreement No 883335 ERATOSTHENES, the European Union's Horizon CL3 Increased Cybersecurity 2021 under grant number agreement 101069471, and Spanish Ministry of Economy and Digital Transformation, under the project CERBERUS-HERMES (Grant No. TSI-063000-2021-44). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the Spanish Ministry, European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Directive (EU) 2022/2555 of the European Parliament and of the Council of 14 December 2022 on measures for a high common level of cybersecurity across the Union, amending Regulation (EU) No 910/2014 and Directive (EU) 2018/1972, and repealing Directive (EU) 2016/1148 (NIS 2 Directive) (Text with EEA relevance). ELI: http://data.europa.eu/eli/dir/2022/2555/oj.

[2] Wagner, Thomas; Mahbub, Khaled; Palomar, Esther and Abdallah, Ali. Cyber Threat Intelligence Sharing: Survey and Research Directions. Computers & Security. 87, 101589. 2019. DOI: 10.1016/j.cose.2019.101589.

[3] Wagner, Cynthia; Dulaunoy, Alexandre; Wagener, Gérard and Iklody, Andras. MISP: The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform. Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security 49–56. 2016. URL: https://www.misp-project.org/.

[4] Preuveneers, Davy; Joosen, Wouter; Bernal Bernabe, Jorge and Skarmeta, Antonio. Distributed Security Framework for Reliable Threat Intelligence Sharing. Security and Communication Networks. 2020. 1–15. https://doi.org/10.1155/2020/8833765.

[5] Badsha, S.; Vakilinia, I.; and Sengupta, S. Privacy Preserving Cyber Threat Information Sharing and Learning for Cyber Defense, 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, pp. 0708–0714, 2019 DOI: 10.1109/CCWC.2019.8666477.

[6] Alishahi, Mina; Saracino, Andrea; Martinelli, Fabio and Marra, Antonio. Privacy preserving data sharing and analysis for edge-based architectures. International Journal of Information Security. 21. 1–23. 2022. 10.1007/s10207-021-00542-x.

[7] Elliot, M., and Domingo-Ferrer, J. The future of statistical disclosure control. arXiv preprint arXiv:1812.09204. 2018.

[8] Sweeney, Latanya. K-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl. Based Syst. 10, 5 557–570. 2002. https://doi.org/10.1142/S0218488502001648.

[9] Machanavajjhala, A.; Kifer, D.; Gehrke, J., and Venkitasubramaniam, M. l-diversity: Privacy beyond k-anonymity. Acm Transactions on Knowledge Discovery from Data (Tkdd), 1(1), 3. 2007. https://doi.org/10.1145/1217299.1217302.

[10] Li, N., Li, T. and Venkatasubramanian, S. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. 2007 IEEE 23rd International Conference on Data Engineering, Istanbul, Turkey, pp. 106–115. 2007. https://doi.org/10.1109/ICDE.2007.367856.

[11] Harvard University Privacy Tools Project. Differencial Privacy. https://privacytools.seas.harvard.edu/differential-privacy.

[12] Lear, E.; Droms, R.; and Romascanu, D. Manufacturer Usage Description Specification. RFC Editor. 2019. https://doi.org/10.17487/RFC8520.

[13] Björklund, M. The YANG 1.1 Data Modeling Language. RFC Editor. 2016. https://doi.org/10.17487/RFC7950.

[14] Bray, T. The JavaScript Object Notation (JSON) Data Interchange Format. RFC Editor. 2017. https://doi.org/10.17487/RFC8259.

[15] Jethanandani, M., Agarwal, S., Huang, L., and Blair, D. YANG Data Model for Network Access Control Lists (ACLs). RFC Editor. 2019. https://doi.org/10.17487/RFC8519.

[16] Securing Small-Business and Home Internet of Things Devices: NIST SP 1800-15, NIST, Gaithersburg, MD, USA. 2019. https://doi.org/10.6028/NIST.SP.1800-15.

[17] "CVE vulnerabilities by date," May 2022, [Online]. Available: https://www.cvedetails.com/browse-by-date.php.

[18] Matheu-García, S. N. and Skarmeta, A. Defining the threat manufacturer usage description model for sharing mitigation actions. In 2022 1st International Conference on 6G Networking (6GNet) (pp. 1–4). IEEE. 2002. DOI: 10.1109/6GNet54646.2022.9830415.

[19] Hernández-Ramos, J. L., Matheu, S. N., Feraudo, A., Baldini, G., Bernabe, J. B., Yadav, P., … and Bellavista, P. Defining the behavior of IoT devices through

the MUD standard: Review, challenges, and research directions. *IEEE Access*, *9*, 126265–126285. 2021. DOI: 10.1109/ACCESS.2021.3111477.

[20] García, S. N. M., Molina Zarca, A., Hernández-Ramos, J. L., Bernabé, J. B., & Gómez, A. S. (2019). Enforcing behavioral profiles through software-defined networks in the industrial internet of things. *Applied Sciences*, *9*(21), 4576. 2019. DOI: 10.3390/app9214576.