# Defining the Threat Manufacturer Usage Description Model for Sharing Mitigation Actions

Sara N. Matheu-García
*Department of Information and Communication Engineering*
*University of Murcia*
Murcia, Spain
saranieves.matheu@um.es

Antonio Skarmeta
*Department of Information and Communication Engineering*
*University of Murcia*
Murcia, Spain
skarmeta@um.es

***Abstract*—While the growing development of paradigms such as the 5G or the Internet of Things (IoT) enlarges the attack surface due to its high inter-connectivity, an effective prompt response to discovered vulnerabilities arises as a crucial need to guarantee the security of these systems throughout their lifecycle. Based on the threat signalling mechanism proposed by NIST and the Manufacturer Usage Description (MUD) standard, this article defines the threat MUD model, its architecture and usage to share security information about threats, compromised domains and possible mitigation actions in terms of security policies.**

***Index Terms*—threat MUD, MUD, information sharing, CTI, NIST**

## I. Introduction

The advent of the 5G technology and its close relationship with the Internet of Things (IoT) promises to realise the vision of a hyper-connected society, in which humans and devices compose complex interconnected systems leading to a strong cybersecurity interdependence. In this scenario, the final network becomes much more complex and heterogeneous and therefore it can be much more feasible for a vulnerability to affect many more systems and to be propagated very quickly. Due to the borderless nature of the infrastructures and threats involved, any vulnerability or security incident in one country can have catastrophic implications throughout the world. While Europe is leading huge initiatives such as the Cybersecurity Act and the 5G Toolbox to ensure the security of these systems, it is still unclear how to handle new vulnerabilities in such a complex context as 5G.

Only in 2021, more than 20,000 vulnerabilities were detected[1]. The fact is that manufacturers cannot quickly deal with new discoveries since the release of a patch or an update is usually a slow process. In this sense, security information sharing systems propose an efficient, fast and collaborative way of sharing recently discovered vulnerabilities or attacks in order to react in time before a patch is released. The US National Institute of Standards and Technology (NIST) proposed a threat signalling approach using a threat Manufacturer Usage Description (MUD) [1]. The threat MUD is based on the MUD standard for network behavioural specification [2], and they are intended to be structured similarly. However, unlike the MUD standard, the threat MUD is designed as a mitigation mechanism. The NIST gives some indications about the new MUD and a specific build to obtain it. However, the threat MUD model is still unclear.

In this paper, we analyse the NIST indications and the MUD standard model to define the threat MUD model, we generalise the architecture proposed by the NIST to combine the usage of the MUD and threat MUD and we provide two possible scenarios in which the threat MUD file can be used for sharing security information and mitigations. In this way, the proposal guarantees an effective action when a security breach is detected. We do not only identify the systems that may be affected, but we also design a mechanism based on a standard to communicate the existence of said threat and the mitigations that should be applied to avoid more damage even before a patch is released.

The structure of this paper is the following: Section II describes the MUD standard, with a special focus on the MUD model, which is similar to the threat MUD. Section III defines the proposed threat MUD model and its architecture. Moreover, this section also presents two use cases in which the threat MUD file can be applied. Finally, Section IV concludes this paper.

## II. The Manufacturer Usage Description standard

The MUD [2] is a standardised behavioural profile that was established by the Internet Engineering Task Force (IETF) in 2019. The main purpose of the MUD specification is to limit the device's attack surface by allowing manufacturers to specify the network behaviour for their devices. The profile is based on a set of policies, also known as Access Control Lists (ACLs) that define the communication's endpoints. Beyond the usage of IP addresses to enable interactions with other services, the MUD represents a scalable and flexible solution to the specification of network access policies. Indeed, the MUD design and format make it possible to automate the creation of network access policies based on the manufacturer's MUD profile.

The MUD standard restricts IoT device connections by defining ACLs, using the Yet Another Next Generation (YANG) [3] standard to model network restrictions and using JavaScript Object Notation (JSON) [4] for serialisation. Towards this end, the MUD file contains two main blocks: the

[1]https://www.cvedetails.com/browse-by-date.php

"mud" and "acls" containers. The "mud" container specifies several features related to the MUD file itself such as the current version of the MUD file or the URL that can be used to retrieve it and the signature to verify the integrity of the MUD file to avoid security issues. After that, the "acls" container defines those ACLs based on [5]. An ACL has a *name*, a set of conditions to apply the rule (*matches*), and the *actions* to apply in case the conditions are satisfied (e.g., forwarding accept or deny). It's worth noting that the MUD model includes Network ACL extensions to the YANG data model, which are augmented by the MUD standard to specify more expressive terms that facilitates the definition of high-level policies without the need to know the associated IP addresses. These keywords are *manufacturer*, *same-manufacturer*, *model*, *local-networks*, *controller* and *my-controller*.

The NIST [1] presented a threat signalling technique using what is called a threat MUD based on this standardised behavioural profile. Although this threat MUD has a structure similar to the regular MUD format, it is intended to serve as a mitigation mechanism, listing external sites to and from which traffic should be restricted due to their association with a specific threat. Therefore, it is not within the scope of the threat MUD to provide a list of sites with which access should be permitted, nor to establish any rules for local network traffic. As a result, rather than being developed by the manufacturer, the threat MUD is supposed to be created by a threat intelligence provider. However, the NIST only gives some indications about the threat MUD model and its similarity with the MUD standard.

## III. Formalisation of the threat MUD

This section defines the threat MUD model and its architecture taking the NIST guidelines and the MUD standard as a starting point. In addition, two possible usages of the generalised threat MUD are proposed to highlight the interest of the proposal. The first one for sharing encountered threats and the second one for obtaining information about compromised domains and possible mitigations.

### *A. Threat MUD modeL*

The only indication that the NIST gives about the threat MUD document is that the threat MUD model is equal to the MUD model, with the exception of two fields: *model-name* and *mfg-name*, which are substituted by the name of the threat and the intelligence provider, respectively. However, other fields of the MUD standard are not applicable to the threat MUD concept, and additional fields would be required to complete the information about the threat detected. We performed a carefully analysis of the standard MUD model to select which fields are applicable to this new concept of threat MUD and which fields should be added. Listing 2 shows the first module of the threat MUD. This module has been generated from the standard MUD model and the NIST indications with some variations, and includes the following fields:

- **threat-mud-version**, previously named *mud-version*, indicates the current version of the threat MUD file. It can be used to get the last update of the threat MUD or compare different threat MUD files.
- **threat-mud-url**, previously named *mud-url*, indicates the URL associated with the current threat MUD file. This URL can be used to retrieve the file.
- **last-update**, indicates the date of the last update of the threat MUD file, which can be used to get the last version of the file.
- **threat-mud-signature**, previously named *mud-signature*, indicates the URL in which the signature of the file is located. As in the MUD standard, this signature is used to check the integrity of the threat MUD file.
- **cache-validity**, indicates the frequency in hours to check for an update of the current file. This field is especially relevant in the threat MUD, as an updated threat MUD can be created in a short time if new domains are known to be compromised by the associated threat, and therefore, having the last update of the file is crucial to implement the countermeasures.
- **is-supported** indicates if the threat associated with the threat MUD file is currently being addressed by the involved manufacturers.
- **threat-intelligence-provider** substitutes the *mfg-name* field and indicates the threat intelligence provider that detected and alerted about the threat (NIST indication).
- **threat-name** substitutes the *model-name* field and identifies the threat associated with the threat MUD file (NIST indication).
- **cvss-vector** is a new field that indicates the severity of the threat in terms of the Common Vulnerability Scoring System (CVSS) standard [6]. In particular, the CVSS score is represented as a vector string, a compressed textual representation of the values used to derive the score.
- **documentation** points to an URL in which additional information about the threat can be found e.g., a link to the National Vulnerability Database (NVD)[2] entry.
- **extensions**, as in the MUD standard, is reserved for future extensions of the threat MUD model.
- **from-device-policy** and **to-device-policy**, indicate the name of the ACLs that should be applied to mitigate the threat. These ACLs are further detailed in the next module of the threat MUD.
- **system-info**, **firmware-rev** and **software-rev** fields have been removed from the threat MUD model, as they are associated with a specific device.

Listing 4 shows the second module of the threat MUD, which integrates the ACLs that could be applied to mitigate the associated threat in terms of network access control. The module is similar to the MUD standard model. However, as the threat MUD is associated with a specific threat, not with a device and the configuration to apply should be as generic as possible, some fields have been removed from the ace/matches/mud section, in particular, **same-manufacturer,**

[2]https://nvd.nist.gov

**local-networks, controller and my-controller**. The field *same-manufacturer* always had a null value to indicate devices from the same manufacturer specified in the *mfg-name* field from the previous module. However, this field was substituted by the intelligence provider. The field *local-networks* was used to remove or allow access to the whole local network of a particular device. As the threat MUD should be applicable to any kind of device, this field has been removed. Finally, *controller* and *my-controller* were used to indicate the generic controller of a device. As before, this field is not generic enough to be included in the threat MUD.

Listing 1: Threat MUD module

```
module: ietf-threat-mud
 +--rw threatmud!
     +--rw threat-mud-version
     +--rw threat-mud-url
     +--rw last-update
     +--rw threat-mud-signature?
     +--rw cache-validity?
     +--rw is-supported
     +--rw threat-intelligence-provider
     +--rw threat-name
     +--rw cvss-vector?
     +--rw documentation?
     +--rw extensions*?
     +--rw from-device-policy
     +--rw to-device-policy
```

Listing 2: Threat MUD module

```
module: ietf-mud
 +--rw mud!
     +--rw mud-version
     +--rw mud-url
     +--rw last-update
     +--rw mud-signature?
     +--rw cache-validity?
     +--rw is-supported
     +--rw mfg-name
     +--rw model-name
     +--rw system-info?
     +--rw firmware-rev?
     +--rw software-rev?
     +--rw documentation?
     +--rw extensions*?
     +--rw from-device-policy
     +--rw to-device-policy
```

Listing 3: ACL module

```
module: ietf-access-control-list
  +--rw acls
     +--rw acl* [name]
     |   +--rw als
     |       +--rw acess-list* [name]
     |           +--rw name
     |           +--rw type?
     |           +--rw aces
     |               +--rw ace* [name]
     |                   +--rw name
     |                   +--rw matches
     |                   |   +--rw mud
     |                   |   |   +--rw manufacturer?
     |                   |   |   +--rw model?
     |                   |   +--rw eth?
     |                   |   +--rw ipv4?
     |                   |   +--rw ipv6?
     |                   |   +--rw tcp?
     |                   |   |  +--rw direction-initiated?
     |                   |   +--rw udp?
     |                   |   +--rw icmp?
     |                   |   +--rw egress-interfe?
     |                   |   +--rw ingress-interfe?
     |                   +--rw actions
     +--rw attachment-points
```

Listing 4: ACL module

```
module: ietf-access-control-list
  +--rw acls
     +--rw acl* [name]
     |   +--rw als
     |       +--rw acess-list* [name]
     |           +--rw name
     |           +--rw type?
     |           +--rw aces
     |               +--rw ace* [name]
     |                   +--rw name
     |                   +--rw matches
     |                   |   +--rw mud
     |                   |   |   +--rw manufacturer?
     |                   |   |   +--rw model?
     |                   |   |   +--rw local-networks?
     |                   |   |   +--rw same-manufacturer?
     |                   |   |   +--rw controller?
     |                   |   |   +--rw my-controller?
     |                   |   +--rw eth?
     |                   |   +--rw ipv4?
     |                   |   +--rw ipv6?
     |                   |   +--rw tcp?
     |                   |   |  +--rw direction-initiated?
     |                   |   +--rw udp?
     |                   |   +--rw icmp?
     |                   |   +--rw egress-interfe?
     |                   |   +--rw ingress-interfe?
     |                   +--rw actions
     +--rw attachment-points
```

### B. *Threat MUD architecture*

A particular build to obtain the threat MUD was proposed by the NIST in [1] to combine the usage of the MUD standard and the threat MUD. Figure 1 shows a generalisation of that build including additional components for threat management.

- The device operating in the network. It is responsible for sending the MUD URL to the switch for the MUD obtaining.
- The router or switch responsible to forward or restrict the device traffic.
- The MUD Manager is the main entity of the MUD architecture. It will be in charge of asking for the MUD file to the MUD File Manager using the MUD URL. Additionally, it will also retrieve a signature to validate the integrity of the MUD file. Once the MUD is obtained, it will translate and enforce the MUD policies over the switch. Although the standard does not specify how to perform the enforcement, further research has been done to enforce them in a SDNs architecture [7].
- The MUD file server, located in the manufacturer domain, stores all the MUD files from devices of a certain manufacturer.
- The Threat Agent monitors the DNS traffic from and to the device to detect when a DNS request is not solved. When a domain is suspicious of being compromised, that is, a DNS request returned a null value, it asks for confirmation to the Threat API and alerts the Threat MUD Manager about this domain. Moreover, it also receives information from the monitoring and detection entity about possible threats.
- The DNS service, which receives information from threat intelligence providers about a compromised domain. In case the domain is marked as compromised, it returns a null value.

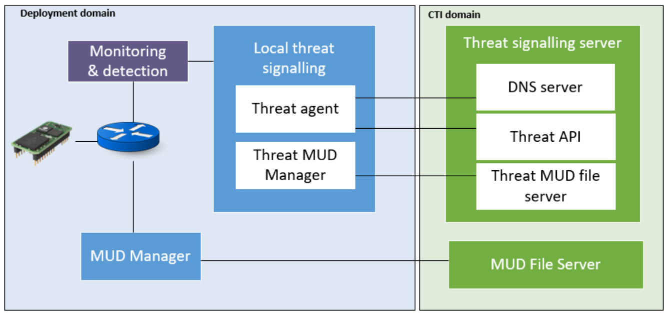

Fig. 1: Threat MUD and MUD architecture

- Threat API receives requests from the Threat Agent to verify whether an unresolved domain is compromised. Moreover, it gives information about the Threat Intelligence provider that identified a compromised domain.
- The Threat MUD Manager, analogous to the MUD Manager, queries the Threat MUD file Server for the threat MUD file and signature. In addition, the Threat MUD Manager enforces the filtering rules in the router. It's worth mentioning that the Threat MUD connected with a threat will list all of the domains that are affected by the threat, as well as the filtering rules that will block access to them. The threat MUD Manager is also responsible for creating a threat MUD file in case a new threat without an existing threat MUD file is detected.
- The threat MUD file server job will consist of storing and delivering Threat MUD files associated with a compromised domain (and threat).
- The Monitoring and detection entity is in charge of monitoring the device communications and alerting the threat agent about possible threats.

### C. *Use Case: Sharing discovered threats*

The threat MUD is integrated into the workflow of a use case whose main objective is to share encountered threats and mitigation with interested stakeholders, as shown in figure 2. In this context, the monitoring and detection entity in charge of monitoring the device communications (step 1) detects a new threat and alerts the local threat signalling service, specifically the threat Agent (step 2). The threat agent validates if this threat was already discovered by asking the threat API (step 3). If the threat API replies that the threat is unknown (step 4), the threat Agent will request the threat MUD manager the creation of a threat MUD associated with the encountered threat, indicating the compromised domains identified by the monitoring and detection entity (step 5). Once the threat MUD file is created, the threat MUD manager will post it on the threat MUD file server (step 6). If accepted, the threat MUD file server will acknowledge it (step 7) and request an update of the threat API database (step 8). In this way, other domains that may be affected by this threat will be able to have access to this information, apply the pertinent mitigations and collaborate in the construction of the new threat MUD file.

### D. *Use Case: Enforcing mitigations*

This second use case depicted in Figure 3 shows how the threat MUD file can be used to deploy security policies to mitigate encountered vulnerabilities. In this case, the detection of compromised domains is performed through the DNS service. The device will eventually make a DNS request to access a certain domain (step 1). The router or switch is responsible to forward the DNS request to the Threat Agent (step 2) and the DNS server (step 3). The DNS service, which receives information from threat intelligence providers about compromised domains will answer the DNS query of the device. In case the domain is marked as compromised, it will return a null value (step 4). The threat Agent, which is monitoring DNS traffic from/to devices to DNS server, will detect a NULL DNS answer, and it will ask for confirmation to the Threat API (step 5). If the threat API confirms that the domain is compromised (step 6), the threat Agent will alert the Threat MUD Manager about this domain to obtain the threat MUD file (step 7). The threat MUD Manager will ask for the associated Threat MUD file (and its signature) to the threat MUD file server (steps 8 and 9). Finally, the threat MUD Manager will translate and enforce the threat MUD policies in the switch (step 10). An integration of this use case architecture within the INSPIRE-5Gplus project is provided in [8].

## IV. Conclusion and Future Work

The management of new vulnerabilities and attacks during the life cycle of a device becomes even more important due to the high inter-connectivity inherent in the IoT. In this article we address the sharing of security information about the threats discovered and possible mitigations to apply based on the threat MUD proposed by NIST and the IETF MUD standard. This new approach to the MUD allows obtaining information on compromised domains and security policies and sharing information on vulnerabilities discovered locally, facilitating the application of mitigations before a patch is available. In the future, we plan to carry out an implementation of the threat MUD following the proposed model, linking it with existing cyber threat information approaches such as the Malware Information Sharing Platform (MISP).

## Acknowledgments

The research is supported by the H2020 EU project INSPIRE-5Gplus project, Grant Agreement Nº 871808 and the H2020 project BIECO, Grant Agreement Nº 952702.

## References

[1] NIST, "Securing Small-Business and Home Internet of Things Devices: NIST SP 1800-15," 2019.
[2] E. Lear, D. Romascanu, and R. Droms, "Manufacturer Usage Description Specification (RFC 8520)," 2019. [Online]. Available: https://tools.ietf.org/html/rfc8520
[3] M. Bjorklund, "The YANG 1.1 data modeling language (RFC 7950)," 2016. [Online]. Available: https://tools.ietf.org/html/rfc7950
[4] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format (RFC8259)," 2017. [Online]. Available: https://tools.ietf.org/html/rfc8259

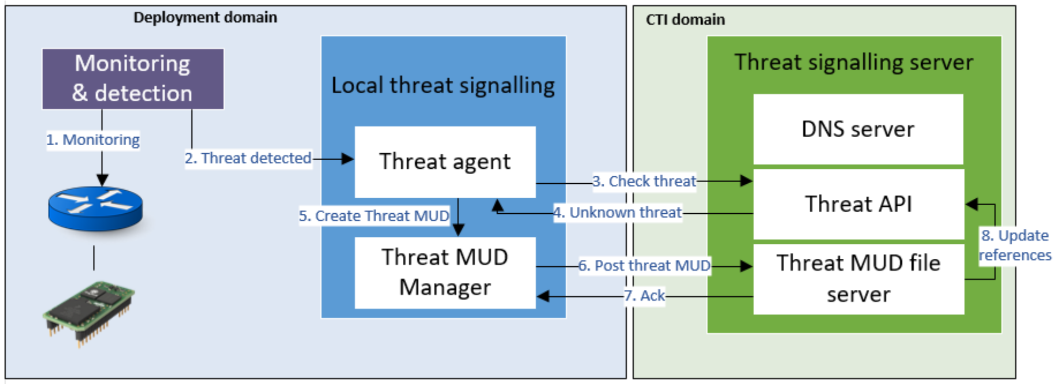

Fig. 2: Use case: Sharing discovered threats

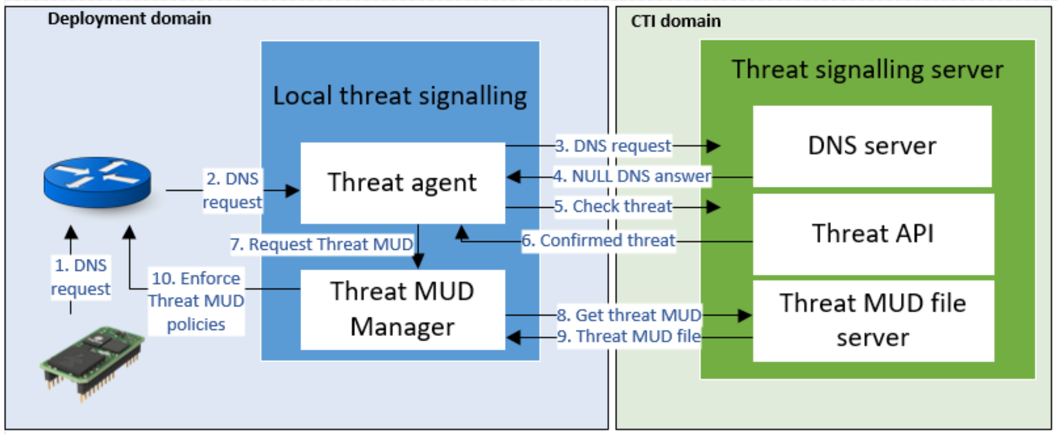

Fig. 3: Use case: Enforce mitigation

[5] M. Jethanandani, D. Blair, L. Huang, and S. Agarwal, "YANG Data Model for Network Access Control Lists (RFC8519)," 2019. [Online]. Available: https://tools.ietf.org/html/rfc8519

[6] FIRST, *Common Vulnerabilities Scoring System (CVSS)*, 2014. [Online]. Available: https://www.first.org/cvss

[7] S. N. M. García, A. Molina Zarca, J. L. Hernández-Ramos, J. B. Bernabé, and A. S. Gómez, "Enforcing Behavioral Profiles through Software-Defined Networks in the Industrial Internet of Things," *Applied Sciences*, vol. 9, no. 21, p. 4576, 2019.

[8] N. P. Palma, S. N. Matheu-García, A. M. Zarca, J. Ortiz, and A. Skarmeta, "Enhancing trust and liability assisted mechanisms for ZSM 5G architectures," in *2021 IEEE 4th 5G World Forum (5GWF)*. IEEE, Oct. 2021, pp. 362–367.