Available online at www.sciencedirect.com

ScienceDirect

Computer Standards and Interfaces 00 (2023) 1–21

Computer Std. and Interfaces

Integrating the Manufacturer Usage Description Standard in the Modelling of Cyber-Physical Systems

Sara Nieves Matheu García[a], Adrián Sánchez-Cabrera[a], Enrico Schiavone[b], Antonio Skarmeta[a]

[a]*Department of Information and Communication Engineering, Computer Science Faculty, University of Murcia, Murcia, 30100, Spain*
[b]*Resiltech s.r.l.,Piazza Nilde Iotti, 25, Pontedera, 56025, Italy*

Abstract

The continuous growth of cyber-physical systems (CPS) attacks, especially due to the conflict in Ukraine, has highlighted the need for cybersecurity management mechanisms, due to the catastrophic consequences that a failure or attack on critical infrastructures such as power plants. Indeed, Gartner predicts that by 2025, 30% of critical infrastructures will suffer a cyberattack. In this context, defining the expected behaviour of the system is key to detecting and mitigating possible vulnerabilities both in the design and runtime phases. Modelling emerges as a tool that facilitates the analysis of the security offered by the system even before the system is implemented, allowing an early risk analysis. However, creating such a model is usually challenging due to its intrinsic complexity, or the reconfiguration needed after a security assessment due to a new vulnerability. The situation gets even worse when the system is a complex CPS-of-Systems, where different Constituent Systems (CS) are interconnected since cascade effects and dependencies are stronger and we might not have all the information from the third-party CS. Also, the results of the evaluation are typically used only during the design phase, thus missing out on potential security policies and mitigations that could be used during the system operation. In this sense, the Manufacturer Usage Description (MUD) allows the manufacturer to define access control policies that reduce the attack surface of a device. However, the limited expressiveness of this standard reduces the possibilities of its application in systems with more complex policies beyond the network level. We propose the usage of the MUD standard as a source of information for CPS modelling, providing information on interactions about third-party components of the system. In addition, we define an extended MUD model that deals with the expressiveness problems of the MUD and allows to automatically generate a behavioural profile that integrates the recommendations obtained from the assessment and modelling processes. The extended MUD could be used during runtime to reduce the attack surface of the system, enforce security configuration or even discern if a component is secure enough to be part of the ecosystem. Our approach has been validated in a real use case in the context of smart grid, to show its applicability.

Keywords: CPS, Modelling, MUD standard, Security, Network

1. Introduction

According to Gartner[1], the continuous growth of cyber-physical systems (CPS), which combines cyber and physical aspects, is intended to be in the spotlight of hackers. Indeed, it is predicted that through 2025, 30% of critical

[1]www.gartner.com/en/documents/4008351/predicts-2022-cyber-physical-systems-security-critical-infrastructure-in-focus

infrastructure organisations will experience a security breach that will result in the halting of operations- or mission-critical cyber-physical system[2]. Even ignoring the value of human life, the monetary costs to businesses in terms of compensation, litigation, insurance, regulatory fines, and reputation damage would be substantial. The situation gets even worse when the system is a complex CPS-of-Systems (CPSoS), where different Constituent Systems (CS) are interconnected, cascade effects and dependencies are stronger and we might not have all the information of the third party CS.

In this scenario, managing the security of a system throughout its lifecycle is crucial to minimise, detect and mitigate possible security failures [1][2]. The modelling of a CPS emerges as a tool that facilitates the analysis of the security offered by the system even before the system itself is implemented, allowing an early risk analysis [3]. Modelling is also widely used to automate the derivation and execution of tests through techniques such as model-based testing (MBT), reducing the time and effort needed to evaluate and certify the security of a system [4]. However, the modelling process is usually complex and expensive in terms of time, requiring high expertise and knowledge of all the system components[5], which is almost impossible in the supply chain. Even if we manage to do the effort to model and assess the CPS security, the results of the process (e.g., vulnerabilities found, misconfiguration, suggestions, etc.) are usually considered only during the design phase of the system, losing valuable information that can be used during the operation phase of the system to configure it properly or detect suspicious behaviours.

In this article, we support the modelling phase of a CPS by integrating the information contained in the Manufacturer Usage Description (MUD) [6] associated with the system components. This Internet Engineering Task Force (IETF) standard defines a file format to define the expected behaviour of a device at network level, thus allowing the specification of security policies that, once implemented, are intended to reduce the attack surface of such device. In this direction, the data contained in the MUD file, which indicates the expected communications and indications with other components, ports and protocols is used as input for the modelling process, integrating this information into the CPS model. As the MUD file is intended to be public, basic information about all the system components could be available even if they are black boxes, supporting the supply chain modelling in case of interaction with third-party components.

Since its adoption, the MUD has received significant interest from the research community and standardisation bodies. In particular, the National Institute of Standards and Technology (NIST) proposes the MUD standard as a promising approach to mitigate security threats, and to cope with denial-of-service (DoS) attacks in the Internet of Things (IoT) environment, including home and small-business networks [7]. Additionally, the European Union Agency for Cybersecurity (ENISA) considers the use of MUD as part of IoT security good practices to improve, allowing devices to advertise their supported and intended functionality [8]. However, the capabilities of the MUD are tempered by the limited flexibility the model offers for defining different types of security policies, which are limited to access control policies at network level [9].

Dealing with the presented challenges, this paper aims to

- use the MUD as a support information source for the modelling process, especially when there are third-party components or CS that should be included in the CPS model,
- extend the MUD model to define more fine-grained security policies using the original MUD file, the CPS model and the risk assessment results as input, containing additional information about the expected behaviour and encountered security issues to be used during the CPS operation time,
- partially automate the generation of the proposed extended MUD file,
- validate the solution in a real context (smart grid) using a specific modelling tool (ResilBlockly) by
  - comparing the original and extended MUD, evaluating the reach of the extension in each of the MUD modules
  - evaluating the integration of the MUD extension and the system model and assessment, showing which fields have been manually introduced or generated from the model to build the extended MUD

[2]https://www.gartner.com/en/doc/757423-predictive-analytics-cyber-security

– comparing the generation time of the standard MUD using the well know tool MUD maker and the generation time of the extended version using Resilblockly tool

In this way, we not only take advantage of the MUD as a preventive mechanism but also facilitate the modelling process of a CPS when there are third-party CS. Furthermore, we enrich the functionality and possibilities of the IETF MUD standard, extending the information that the standard is capable to describe while expanding the possibilities of using the MUD beyond the manufacturing phase (e.g., for secure configuration, detection of misbehaviours and vulnerabilities and mitigation) by integrating the results of the modelling and assessment phases, and automating its generation.

The structure of this paper is the following: Section 2 describes the MUD standard, with a special focus on the MUD model and its expressiveness. Section 3 analyses the main drawbacks of both the MUD and the modelling activities, motivating the purpose of this research. Section 4 is the main core of this work, in which we define the extended MUD model and we detail how to integrate the MUD and the extended MUD with the CPS model. The feasibility of the proposed approach is analysed in Section 5 over a real use case, an Information and Communications Technology (ICT) gateway operating in the smart grid context, which is one of the use cases considered within the BIECO project[3]. Finally, Section 6 concludes this paper and summarises the main achievements and future work.

2. The Manufacturer Usage Description standard

The MUD is a behavioural profile standardised in 2019 within the scope of the IETF. 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 built around a set of policies, or Access Control Lists (ACLs), that specify the communication's endpoints. MUD represents a scalable and flexible approach to the definition of network access policies beyond the use of IP addresses to enable communications with other services. A manufacturer could, for example, declare that access to particular cloud services, as well as connection with other manufacturers' devices, should be permitted. MUD also allows specifying protocols and ports for each communication to provide a more fine-grained configuration of access control rules. The standard also enables the extension of the scheme, allowing manufacturers to express other types of conditions or policies based on their needs. For example, while the MUD is focused on network access control regulations, MUD model expansions are being considered for Quality of Service (QoS) [10] aspects of the communications. However, despite this flexibility, the MUD model does not provide mechanisms to describe more fine-grained aspects and additional security restrictions beyond the network layer.

One of the key advantages of the MUD approach is that the manufacturer is responsible for defining the devices' behavioural profiles (instead of the typical network administrator). Indeed, the MUD design and format make it possible to automate the creation of network access policies based on the manufacturer's MUD profile. It should be noted, however, that the instantiation of these profiles may be influenced by the network domain in which the device is deployed.

2.1. The MUD model

The MUD standard restricts IoT device connections by defining ACLs, using the Yet Another Next Generation (YANG) [11] standard to model network restrictions and using JavaScript Object Notation (JSON) [12] for serialisation. Towards this end, the MUD file contains two main blocks: the "mud" and "acls" containers. The "mud" container specifies several features related to the MUD file itself. The property *mud-version* specifies the current version of the MUD file, whereas the *last-update* defines when the current version of the file was generated. The MUD is identified by the *mud-url*, that is, the URL that can be used to retrieve the MUD file. The *mud-signature* (optional) verifies the integrity and authenticates the MUD file to avoid security issues. Furthermore, this container also allows defining optional aspects such as the model of the device (*model-name*), the firmware and software revision (*firmware-rev, software-rev*), the minimum period of time before checking for updates (c*ache-validity*), if the device will receive MUD/software/firmware updates (*is-supported*), additional information (*systeminfo*), link to the device documentation (*documentation*) and additional extensions. Finally, the *to-device-policy* and *from-device-policy* containers represent

[3]https://www.bieco.org/

access list references by indicating the appropriate direction of a specific flow to define the communication pattern of the device.

After that, the "acls" container defines those ACLs based on [13]. Each 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). By default, the MUD specifies only the allowed connections, but in certain cases, it can be also useful to define an explicit access restriction to a service (for example, if it has been compromised). 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*. For example, the keywords *manufacturer* and *same-manufacturer* enable the definition of policies to allow or deny the interaction with devices from the same manufacturer. This way, MUD files define the type of communications and access of a certain device in the form of policies or ACLs. Some examples of these restrictions could be "allow the communication to devices of the same manufacturer", "allow the access to a specific DNS service", or "deny the access for a specific port".

2.2. The MUD Architecture

The MUD standard also defines a high-level architecture to allow the network domain where the device is deployed to obtain and enforce this profile. Figure 1 shows the steps and entities of the architecture:

- The device willing to access the deployment network, which will be in charge of sending the MUD URL to the switch (Step 1). The standard proposes the usage of the Dynamic Host Configuration Protocol (DHCP) or certificates to embed the URL, but other mechanisms have been also analysed, for example, integrating the MUD URL in the bootstrapping protocol (e.g., within the Extensible Authentication Protocol) [14].
- The switch will forward the MUD URL to the MUD Manager (Step 2).
- 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 (Steps 3 and 4). 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 (Step 5). Although the standard does not specify how to perform the enforcement, further research has been done to enforce them in a SDNs architecture [15].
- The MUD file server, located in the manufacturer domain, stores all the MUD files from devices of a certain manufacturer.

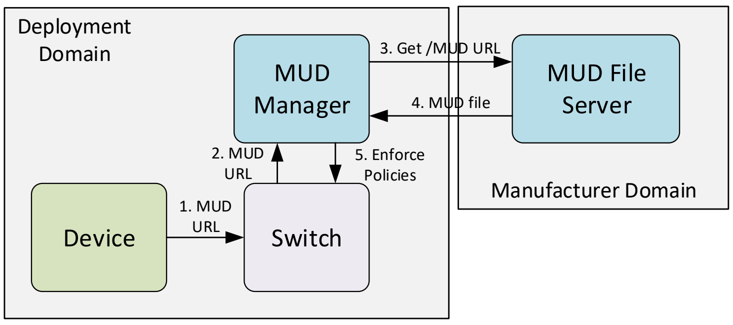


Figure 1: MUD Architecture

3. State of the art, challenges and innovation

This section reviews the main works and current challenges associated with the two building blocks that compose our proposal. In particular, we analyse the modelling challenges to assess the cybersecurity of a CPS, the problem of generating the MUD file in an automated way and how the expressiveness of the MUD limits their usage.

3.1. Third-party components and time-consuming system modelling

ICT systems and solutions developed by different companies, once integrated into a single system give birth to a so-called System-of-Systems (SoS). SoS are typically deployed on very large geographic scales, comprise a very large number of components, are organised in a hierarchical structure, are driven by complex interactions, and their correct operation and availability are essential. However, the efforts and investments required for their design, implementation and maintenance are enormous. Therefore, new methodologies, principles and reliable tools are needed to manage their evolution and address the growing complexity.

A model is an abstraction of a system that highlights its important features while neglecting all those details that are marginal to the objective of the study. Several types of studies and activities can be conducted leveraging models of software-intensive systems; some examples are: i) Model-Driven Engineering (MDE) [16] and Model-Based Systems Engineering (MBSE) [17], that leverage models throughout the software and system engineering lifecycle, ii) Model-Based Evaluation, that encompasses techniques for the evaluation of system dependability and security [18], and iii) Model-Based Testing [19], [20], which is a technique for the automation of software testing activities.

Modelling plays a key role in all of the above activities, especially at design time when the actual system is not available yet. However, numerous challenges related to the modelling of complex software exist [21], and the situation gets even worse when the system under modelling is a complex CPSoS [22], which may integrate third-party components. In the latter case, the model may have to represent heterogeneous aspects across different layers of the technology stack from the physical, sensor and actuator layer, to communication and middleware, up to the application layer [23]. The intrinsic complexity of adopting traditional approaches for modelling, engineering and analysing complex CPSoS becomes high, it may end up generating the so-called spaghetti diagrams, difficult to design and maintain, and the gained result of the entire activity may not be worth the effort. Additionally, a CPS usually integrates third-party components from different companies whose expected behaviour and interaction with other components may be unknown. The lack of information about these black-box components that are part of the CPS makes it difficult and slows down the description and modelling of the system as a whole.

We study how part of the information needed for the model can be obtained semi-automatically from external sources (e.g., the MUD standard), especially that related to the interaction among components, even if they are third-party. Other proposals already consider the MUD as a mechanism to identify a component¡[24] or to visualise the interactions between components [25] [26], but to the best of our knowledge, this is the first time the MUD is considered as input for the modelling process. A previous proposal [27] provides an extension of the MUD model and its integration with the security assessment process, linking the test results with the MUD extended fields. However, there is no link with the model created during the MBT process and therefore, the information extended and generated in the new MUD model is limited to specific aspects.

Our proposal supports the modelling process by linking the CPS model with the system specification described in the MUD standard. We establish a mapping between the system components and their properties and the policies of the MUD. This way, the MUD can be used as input for the modelling process, facilitating the generation of a preliminary model that can be later refined by the user. As a proof of concept, we implemented the integration of the original MUD file with the model using the ResilBlockly modelling tool.

3.2. Underestimated security assessment results

The adoption of models can increase confidence and completeness in risk assessment, provide formal support for a more objective evaluation and documentation of the risk assessment rationale through reviewable artefacts and support change management through traceability of risks to design elements [28]. However, the results of the security evaluation are usually considered only during the design phase to validate or certify the security of the CPS, missing very valuable information that reports the security flaws that the system has and how they could be avoided during the system's operation phase. Previous results [27] paved the way for the usage of the testing report within the MUD as a treatment mechanism. The data aggregated was intended to address or mitigate security issues encountered during the evaluation process, reducing the attack surface to the allowed behaviours. Therefore, the MUD could be used to enforce the recommendations provided and to monitor suspicious behaviours during the operation phase that are outside the ones reflected in the MUD file [15].

In this paper we extended this approach, including additional data about the specific vulnerabilities and weaknesses discovered in the assessment phase. This information complements the security policies included in the MUD, allowing the user to check current vulnerabilities of the system and enforce possible mitigations but also to decide whether the system has the security level required to be part of a bigger ecosystem.

3.3. MUD limited expressiveness

Despite the high benefits of the MUD standard [9], one of the main limitations is its model, which is only capable to describe limited network layer security policies. Current literature has already taken note of this problem [29] [30] [9] and there are already some works proposing augmented behavioural profiles. In this sense, [27] defines a MUD profile generated from a security evaluation process including properties such as recommended key sizes or cryptographic primitives, [14] integrates a new module in the MUD model to include Medium-level Security Policy Language (MSPL) policies and whereas [31] extends the model by considering dynamic security aspects in the context of smart buildings. Other works extend the model to consider additional information such as QoS [10] or directly redefine the MUD concept to describe users' behaviour and interactions with their devices [32]. Finally, although there are similar approaches to the MUD standard such as the Hardware and Software Bill of materials (HBOM, SBOM [33]) describing the different components of a system at hardware at software level and their vulnerabilities, they focus more on the description of the components than on the security configuration.

We propose an extended MUD model from the standard one to describe security policies beyond access control at network level. The extension allows the specification of fine-grained security policies at different TCP/IP layers related to cryptographic configuration, security protocols, endpoints, databases, authorisation requirements, limitations on the connections, and known vulnerabilities, similar to the BOM. The extended MUD is intended to be used during the operation phase of the system to deploy it in a secure way following the recommended configuration or to monitor deviations or attacks exploiting the known vulnerabilities. As an example, we generate the extended MUD of the *ICT Gateway*, a middleware from the smart grid context.

3.4. Manual MUD generation

The process required to create a MUD file is named MUD generation process and, as described in the MUD standard, this process is intended to be performed by the manufacturer. Currently, a considerable number of existing MUD-related proposals presume that each IoT device has its own MUD file. However, in current IoT implementations, where the MUD standard is not widely used, this assumption is not supported. As a result, we found in the current literature different approaches that propose supporting mechanisms for the creation of this MUD file. One of the most representatives is the MUD maker[4] tool, which is used by a wide number of authors to create MUD files for attack detection and mitigation [34] [35] [36] [37] [38]. Although this tool provides a simple interface for the user to introduce the MUD data model fields, it is assumed that users have all this information.

Alternatively, and trying to avoid user involvement, other proposals directly generate the MUD file using the network traffic traces from the device to identify the values for the MUD model. The MUDgee tool, created by [39] is one of the most widely used tools that follow this technique [40] [41]. MUDgee is an open-source application that allows the creation of MUD files from network traffic traces. In particular, the tool developers captured traffic from 28 IoT devices for 6 months and used MUDgee to generate a MUD file for each device based on their traffic traces.

Our proposal provides an automated extended MUD generation process by mapping the attributes of the CPS model and the risk assessment results with the policies of the extended model. It is intended to complement the work done in [27], in which only security assessment results were integrated and no automated generation was developed. As a proof of concept, we implemented an automated generation tool for the extended MUD within the ResilBlockly modelling and assessment tool.

4. Integrating the MUD in the system modelling

Our proposal integrates the MUD in the system modelling phase in a bidirectional way. Next sections detail how the original MUD information is used as input for the system model, provide the details of the MUD extension we propose and describe how the information contained at the end of the assessment process is integrated into the extended MUD to be used during the system operation phase.

[4]https://www.mudmaker.org/

4.1. From the original MUD to the model

As we mentioned before, a MUD file is associated with a component, indicating the communications that it is allowed to establish with other components and services. This information allows for identifying dependencies between components and accesses to external services, for example, an update server located in the manufacturer domain. From this data, we can create the network architecture in which the device associated with the MUD is integrated. Figure 2 shows an example of the network architecture created by obtaining information from a single MUD file belonging to component A. The MUD file indicates that component A communicates with other system components, B and C, and with a component external to the system (Component D). Additionally, the direction of the communication, interfaces, protocols and ports used in each of these communications are also identified. If instead of a single MUD file, we collect the information contained in the MUD files of all the components of the system, we obtain the dependencies between those components and the accesses that the system will make or receive from external services. In this sense, BOM can be used as support information to identify the system components, so their corresponding MUD files can be later obtained.

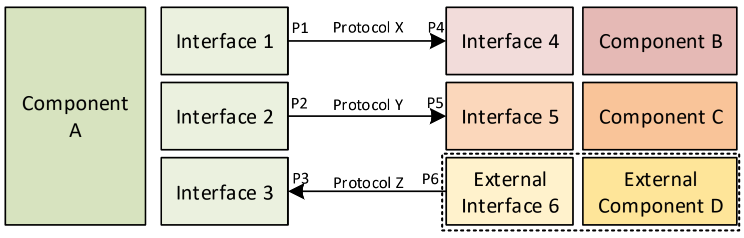


Figure 2: Example of network architecture from a single MUD

The MUD file does not only indicate that there is a relationship between two components, or between a component and a service but also describes the properties of such a relationship. In this way, using the information contained in the MUD files of all the components of a system, we can obtain the architecture of a more complex system in terms of communications, as well as the properties associated with said communications. In particular, the information that we collect from the MUD file to be included in the system model is the following:

- From the "mud" container, we gather metadata related to the component itself (model, software and firmware version, manufacturer).
- From the "acls" container we establish dependencies with other components or services and the attributes associated with the relationship given by the "matches" condition:
  - Name (Uniform Resource Name, URN) of the other component or service from the "mud" submodule.
  - Network protocol used in the interaction (eth, ipv4, ipv6, tcp, udp, icmp) and additional details depending on the protocol (e.g., port, ttl).
  - Egress and ingress interfaces of the component

However, although the relationships between components are useful to facilitate the creation of the system model, the attributes that the MUD is capable of describing are very limited, as we already anticipated in section 3. In the following subsection, we propose an extension of the MUD standard capable of integrating more detailed security information to not only deal with this limitation but also to take advantage of the information produced at the end of the modelling and risk assessment processes.

4.2. *Extended MUD model*

With the objective of improving the MUD model expressiveness, we propose an extension of the information inside that can be linked with information available in the system model. This new version includes additional aspects related to the application level, cryptography, weaknesses and vulnerabilities of the target device. Section 2.1 shows that in a MUD file, we find two constituent modules: "mud" and "acls" containers. Both of them have been extended in this proposal.

Listing 1 shows the extension of the first module. The extension, marked in bold, includes information about the possible weaknesses and vulnerabilities that the target device may have and a link to BOM information ("boms"), which includes the BOM version and the URL. Following the Yang Tree Diagrams syntax [42], this information has been integrated inside the "mud" container. In particular, both vulnerabilities and weaknesses have the Common Weakness Enumeration (CWE) [43]/Common Vulnerabilities and Exposures (CVE) [44] id ("id"), record name ("name") and weakness/vulnerability description ("description"). Also for both blocks, likelihood ("likelihood") and risk estimation ("risk") are included, having as possible values: "very_low", "low", "moderate", "high", "very_high". Specifically, risk estimation is determined according to NIST SP 800-30 matrix [45] as a combination of likelihood and impact. In the case of weaknesses, "impact" field could take the values: "very_low", "low", "moderate", "high", "very_high". Similarly for vulnerabilities, the impact is described in the Common Vulnerability Scoring System (CVSS) [46] base score ("cvss") field, defined as numeric value: "0.0" (none), "0.1 - 3.9" (low), "4.0 - 6.9" (medium), "7.0 - 8.9" (high), "9.0 - 10.0" (critical). As additional information, the weaknesses block defines submission ("date") and last modified date ("last_modified"), whereas the vulnerabilities block contains the creation date of the record ("date").

Listing 1: Extended MUD Container

```
module: ietf-mud
 +--rw mud!
    +--rw mud-version
    +--rw mud-url
    +--rw last-update
    +--rw mud-signature?
    +--rw cache-validity?
    +--rw is-supported
    +--rw systeminfo?
    +--rw mfg-name?
    +--rw model-name?
    +--rw firmware-rev?
    +--rw software-rev?
    +--rw documentation?
    +-- rw boms?
    |    +-- rw bom-url
    |    +-- rw bom-version
    +--rw from-device-policy
    +--rw to-device-policy
    +-- rw [weaknesses]?*
    |    +-- rw id
    |    +-- rw name
    |    +-- rw description
    |    +-- rw date?
    |    +-- rw last_modified?
    |    +-- rw likelihood
    |    +-- rw impact
    |    +-- rw risk
    +-- rw [vulnerabilities]?*
         +-- rw id
         +-- rw name
         +-- rw description
         +-- rw date?
         +-- rw likelihood
         +-- rw cvss
         +-- rw risk
```

The second container, "acls", has been also extended (Listing 2) to add fine-grained information related to the application layer and cryptographic parameters. Listing 2 shows the proposed extension based on the YANG Data Model for Network ACLs standard [13]. The new model includes a database field ("database", line 19) that provides the internet host URI of the used databases in order to allow connections from/to databases, not only from devices. Moreover, we integrate into the extended model a block of cryptographic details and an application layer section to describe the recommended parameters and security restrictions for each application protocol used in the communications.

Listing 2: Extended ACLs Container

```
1   module: ietf-access-control-list
2    +--rw acls!
3       +--rw acl* [name]
4       |   +--rw acls
5       |      +--rw access-list* [name]
6       |         +--rw name
7       |         +--rw type?
8       |         +--rw aces
9       |            +--rw ace* [name]
10      |               +--rw name
11      |               +--rw matches
12      |               |   +--rw mud
13      |               |   |   +--rw manufacturer?
14      |               |   |   +--rw same-manufacturer?
15      |               |   |   +--rw model?
16      |               |   |   +--rw local-networks?
17      |               |   |   +--rw controller?
18      |               |   |   +--rw my-controller?
19      |               |   |   +-- rw database?
20      |               |   +--rw eth?
21      |               |   +--rw ipv4?
22      |               |   +--rw ipv6?
23      |               |   +--rw tcp?
24      |               |   +--rw udp?
25      |               |   +--rw icmp?
26      |               |   +--rw egress-interface?
27      |               |   +--rw ingress-interface?
28      |               |   +-- rw [keys]?*
29      |               |   |   +-- rw kty?
30      |               |   |   +-- rw alg
31      |               |   |   +-- rw crv?
32      |               |   |   +-- rw length
33      |               |   |   +-- rw key_ops
34      |               |   |   +-- rw purpose?
35      |               |   |   +-- rw x5u?
36      |               |   |   +-- rw x5c?
37      |               |   +-- rw [application-protocol]?*
38      |               |       +-- rw protocol
39      |               |       +-- rw version
40      |               |       +-- rw num-connections
41      |               |       +-- rw [resource]?*
42      |               |       |   +-- rw url
43      |               |       |   +-- rw [method] *
44      |               |       |   +-- rw auth?
45      |               |       |       +-- rw key
46      |               |       |       +-- rw value
47      |               |       +-- rw keepAlive?
48      |               +--rw actions
49      +--rw attachment--points
```

The cryptographic block (lines 28-36) is based on the JSON Web Key (JWK) [47] standard, starting with the "keys" parameter, which represents an array of JWK values. Inside this block, for each key, a key type ("kty") parameter is used to identify the cryptographic algorithm family used ("EC" for Elliptic Curves, "RSA" or "oct" are possible values). Next, the algorithm intended to be used with the key is defined ("alg") following the "JSON Web Signature and Encryption Algorithms" registry established in the JSON Web Algorithms (JWA) [48] standard. For the case of Elliptic Curve keys an additional field identifies the cryptographic curve that is used ("crv"). This parameter is also based on the JWA [48] standard. Then, the key length ("length") in bits, the operations ("key_ops", such as sign, encrypt, derive key) and purpose ("purpose", such as ciphering, authentication, confidentiality) for which the key is intended to be used. Finally, two optional values are included: a Uniform Resource Identifier (URI) ("x5u"), referring to a resource for an X.509 public key, and an X.509 Certificate Chain ("x5c").

The application layer block (lines 37-47) contains, for each protocol, information about the protocol name ("protocol"), version ("version") and the maximum number of simultaneous connections ("num-connections") that the device accepts for that protocol. Thanks to this limitation potential DoS attacks can be mitigated. Moreover, a resources block is included in order to define the recommendations about access to the device resources. This block contains the URL for the resource ("url"), the recommended allowed methods ("method") and pairs of key/value ("key", "value") with the attributes needed to be authorised. Finally. we included the minimum amount of time an idle connection has to be kept open ("keepAlive"). It is important to mention that this block should be repeated for each protocol we want to specify.

4.3. From the model to the extended MUD

The proposed approach also deals with the automatic generation of the extended MUD, establishing a mapping between the attributes of the system model and the fields described in the mud. In particular, the extension proposed in the previous section draws mainly on two sources of information, as shown in Figure 3: the refinement of the initial model of the system and the results of the risk assessment process carried out on the model. Tables 1 and 2 show an overview of the mandatory and some optional MUD fields for the ACL and MUD modules obtained from each source of information.

On the one hand, the original MUD generates an initial scheme of the system architecture, integrating attributes related to the communications at the network level. However, this model is limited, and therefore, to be fully functional and faithful to reality it must be refined by the tester, adding additional information that the original MUD is not capable of describing. Part of this information, in particular the one related to the behaviour of the device (at the network level but also at the application level), and the one related to the recommended cybersecurity configuration, is used in the generation of the extended MUD. On the other hand, models are widely used in risk assessment, since a model allows not only to analyse and simulate the behaviour of the system in order to discover new vulnerabilities but also facilitates the usage of automated testing techniques, such as MBT. As a result of this assessment process, we obtain a list of vulnerabilities and weaknesses present in the system, which can be complemented with or taken from the components and associated vulnerabilities already identified in the BOM. Whereas this information could be used to address such security flaws during the design phase if addressing the flaw is difficult or expensive, the product has already been released, or the vulnerability has been discovered after an external certification process, it may be interesting to share this information. In fact, Europe is aware of the value of sharing cybersecurity information among the relevant stakeholders, which could be crucial to apply mitigation or develop a patch but also to detect security flaws in similar devices. Initiatives such as the European Network and Information Security (NIS) directive [49] are following this path.

However, it is worth noting that as the extended MUD contains sensitive cybersecurity information about the system, it must be protected. In this sense, the original MUD represents the public information available to anyone, which can be stored in public databases, and the extended MUD focuses on facilitating the internal sharing of both the information of the risk assessment and the configuration recommended to reduce the risk of an attack among the relevant (and trusted) stakeholders. Additional information on how to use the MUD file can be found in [15], where we show how to perform the obtaining and enforcement of the MUD file using a SDN-based architecture to configure a specific device or in [50], where authors use the MUD file for the detection of volumetric attacks based on machine learning techniques.

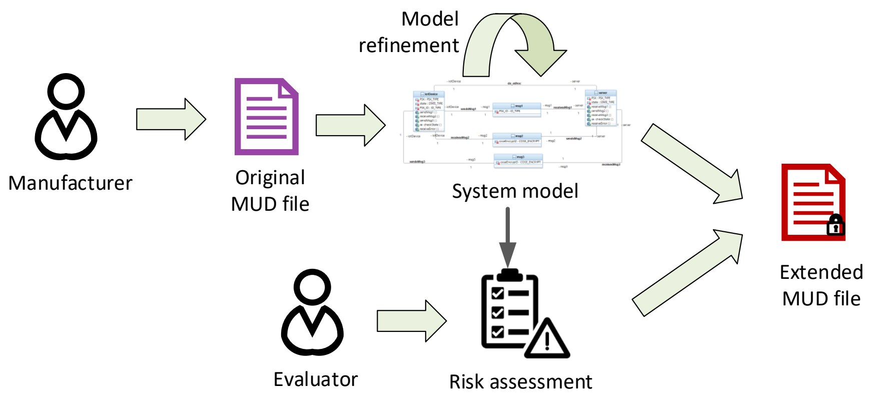


Figure 3: MUD-Modelling integration flow

Table 1: Extraction of extended MUD files - MUD module

MUD field	Taken from risk assessment	Manually introduced	Automatically generated
MUD version			x
MUD URL			x
Last update			x
Signature			x
mfg name		x	
model name		x	
bom		x	
Weaknesses	x		
Vulnerabilities	x		

Table 2: Extraction of extended MUD files - ACL module

MUD field	Taken from model	Manually introduced	Automatically generated
ACL name	x		
ACE name	x		
matches mud	database	rest optional fields	
matches eth, ipv4, ipv6, tcp, udp	x		
keys	x		
application protocol	protocol, version, resource url and method	auth key and value, num connections	
actions			x (default forward)

5. Application to a smart grid use case

The next subsections detail how the integration of the modelling and the MUD is applied to a real use case, an ICT gateway intended for smart grids modelled in the ResilBlockly tool, showing the applicability of our approach. It is worth noting that the objective of this section is not to validate the model developed of the ICT GW, as this paper does not propose a new modelling technique, but to validate the MUD extension and its integration in a modelling process. In this case, we used the Resilblockly tool, but other approaches could be considered as well.

5.1. Description of the use case

The use case comprises two fundamental elements: the ICT Gateway (ICT GW), which is the system under modelling, and ResilBlockly [51], an MDE tool we use to model the system.

5.1.1. ICT Gateway

The ICT GW [52] is a middleware intended for Smart Grids that acts as a mediator between data sourcing, actuation subsystems and domain applications. Its main functionalities are the integration of heterogeneous information from the smart grid, the provision of services for smart energy distribution to the Distribution System Operator (DSO), and ensuring the security and resilience of the system through ICT monitoring, early detection and early diagnosis of anomalies. The heterogeneous information is retrieved from different data and actuation subsystems like smart meters, Remote Terminal Units, Inverters, and Grid Topology Subsystems, while the services offered to DSO mainly belong to the areas of Operation Efficiency, Voltage Quality and Outage Diagnosis.

Figure 4 shows the logical architecture of the ICT GW, with its three architectural layers and the external components (i.e., Graphical User Interface (GUI), Observability Grid Model, Database, Headends and Application Layer). The system is logically divided into four layers: the Adapters Layer, which connects the ICT Gateway to the smart grid; the Domain Logic Layer, which handles interactions with actuation subsystems and contains basic attack and fault detection mechanisms; the Service Layer, which specifies how the ICT Gateway communicates with other components and applications; and the Application Layer, that includes a GUI which allows DSO operators to visually interact with the Applications and the ICT GW functionalities. The other components shown in 4 are: i) Advanced Metering Infrastructure (AMI) Headend, which implements a web interface for providing smart meters network data and related events to the ICT GW; ii) Electrical Measurement (EM) Headend, that provides measurements from the Remote Terminal Units and related alarms; iii) Inverter (INV) Headend, that provides electrical measurements from the Inverter subsystem and notifications related to thresholds exceeding; iv) Grid Topology Headend, that provides access to detailed topological data of the smart grids; v) the Security and Resilience, that implements fault/attack detection mechanisms by identifying anomalies in the ICT network and the measurement devices.

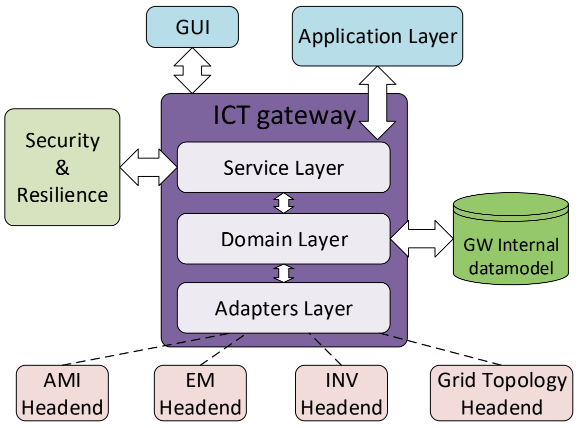


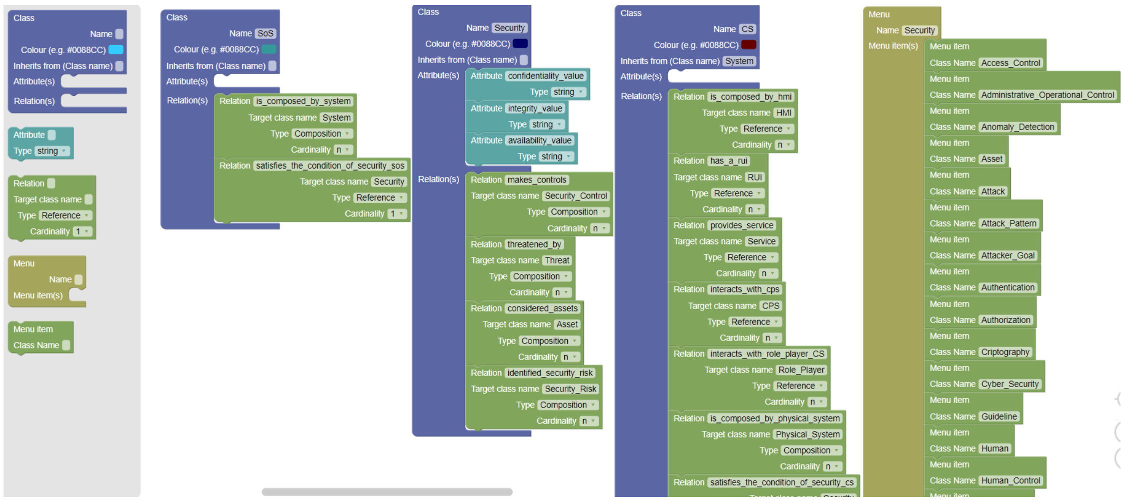
Figure 4: ICT gateway use case

5.1.2. *ResilBlockly*

ResilBlockly is a new MDE tool that has been devised and realised as the evolution of an existing software called Blockly4SoS [53][54]. The outcome is a comprehensive tool capable of modelling the main concepts of CPSoSs, with a reduced cognitive complexity if compared with traditional approaches, also thanks to the adoption of Google Blockly [55]; moreover, a set of new features have been introduced that allow addressing security-related activities, including threat modelling, risk assessment, and matching of risk to system components.

To integrate the usage of the MUD within this tool, Resilblockly has been enhanced with the ability to: i) import existing MUD files and associate them to a component of the modelled system, ii) define new MUD files leveraging an intuitive user interface, iii) extend the MUD file with additional information derived from the security risk assessment process and export it.

Instead of predefined ad-hoc profiles specific for a SoS domain, with ResilBlockly, different profiles can be created from scratch. In this case, a profile is an abstraction of components and relationships for a specific domain and a model is an instance of a profile. This allows for modelling complex systems in any type of context, focusing on the most relevant aspect and giving the model the required structure and refinement. In addition, the System Designer (the modeller) can choose one of the profiles generated by a Profile Expert and instantiate it within a model specific to their use case system; as it is not required to have deep knowledge about the domain (as, instead, the Profile Expert is). Figure 5 shows the profile created for the ICT GW. On the left, we have the building blocks to create the profile. We defined a class SoS with the relation *is composed by system* to model the different components of the CPSoS, called Constituent Systems (CS). Each CS has different relations to model the CS interfaces. For example, we call an interface of a CS where the services of a CS are offered to other CSs a Relied Upon Interface (RUI). Here, the service of the SoS as a whole relies on the services provided by the respective CSs across the RUIs. We call Relied upon Message Interface (RUMI) a message interface where the services of a CS are offered to the other CSs of an SoS. For security aspects, we define a specific relation *satisfies the condition of security CS*, which models a set of security features such as access control, cryptography parameters or authorisation conditions among others.


Class
Name
Colour (e.g. #0088CC)
Inherits from (Class name)
Attribute(s)
Relation(s)

Attribute
Type string

Relation
Target class name
Type Reference
Cardinality 1

Menu
Name
Menu item(s)

Menu item
Class Name

Class
Name SoS
Colour (e.g. #0088CC)
Inherits from (Class name)
Attribute(s)
Relation(s)
Relation is_composed_by_system
Target class name System
Type Composition
Cardinality n
Relation satisfies_the_condition_of_security_sos
Target class name Security
Type Reference
Cardinality 1

Class
Name Security
Colour (e.g. #0088CC)
Inherits from (Class name)
Attribute(s)
Attribute confidentiality_value
Type string
Attribute integrity_value
Type string
Attribute availability_value
Type string
Relation(s)
Relation makes_controls
Target class name Security_Control
Type Composition
Cardinality n
Relation threatened_by
Target class name Threat
Type Composition
Cardinality n
Relation considered_assets
Target class name Asset
Type Composition
Cardinality n
Relation identified_security_risk
Target class name Security_Risk
Type Composition
Cardinality n

Class
Name CS
Colour (e.g. #0088CC)
Inherits from (Class name) System
Attribute(s)
Relation(s)
Relation is_composed_by_hmi
Target class name HMI
Type Reference
Cardinality n
Relation has_a_rui
Target class name RUI
Type Reference
Cardinality n
Relation provides_service
Target class name Service
Type Reference
Cardinality n
Relation interacts_with_cps
Target class name CPS
Type Reference
Cardinality n
Relation interacts_with_role_player_CS
Target class name Role_Player
Type Reference
Cardinality n
Relation is_composed_by_physical_system
Target class name Physical_System
Type Composition
Cardinality n
Relation satisfies_the_condition_of_security_cs

Menu
Name Security
Menu item(s)
Menu item
Class Name Access_Control
Menu item
Class Name Administrative_Operational_Control
Menu item
Class Name Anomaly_Detection
Menu item
Class Name Asset
Menu item
Class Name Attack
Menu item
Class Name Attack_Pattern
Menu item
Class Name Attacker_Goal
Menu item
Class Name Authentication
Menu item
Class Name Authorization
Menu item
Class Name Criptography
Menu item
Class Name Cyber_Security
Menu item
Class Name Guideline
Menu item
Class Name Human
Menu item
Class Name Human_Control

Figure 5: ICT gateway profile

The smart grid ecosystem introduced in the previous section is modelled in Resilblockly as an instance of the described profile. Figure 6 shows an overview of this model, while the complete version is available in ecore format[5]. It encompasses several CS, i.e., the GUI, the Application Layer, the ICT GW, the EM infrastructure, the MQTT Broker, the Security and Resilience, the Database, the Topology infrastructure, the INV infrastructure and the AMI infrastructure. The CSs can be connected directly to the SoS. The interfaces between CSs are modelled through RUMIs, while the messages exchanged and the services provided by the corresponding blocks are available in the SoS profile: respectively message and service. In the model, it can be observed the Smart_Grid_Ecosystem (SoS), the ICT GW (CS) and some of its CSs: Application_REST_API (CS), GUI_REST_API (CS), and GUI (CS).

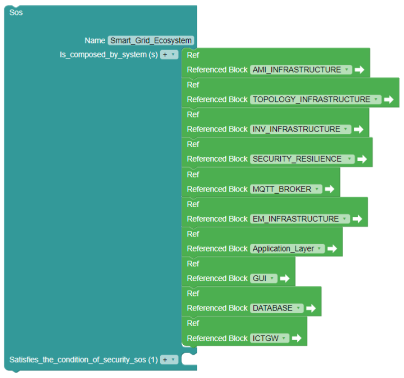

Sos
Name Smart_Grid_Ecosystem
Is_composed_by_system (s) +
Ref
Referenced Block AMI_INFRASTRUCTURE
Ref
Referenced Block TOPOLOGY_INFRASTRUCTURE
Ref
Referenced Block INV_INFRASTRUCTURE
Ref
Referenced Block SECURITY_RESILIENCE
Ref
Referenced Block MQTT_BROKER
Ref
Referenced Block EM_INFRASTRUCTURE
Ref
Referenced Block Application_Layer
Ref
Referenced Block GUI
Ref
Referenced Block DATABASE
Ref
Referenced Block ICTGW
Satisfies_the_condition_of_security_sos (1) +

Figure 6: ICT gateway model - overview

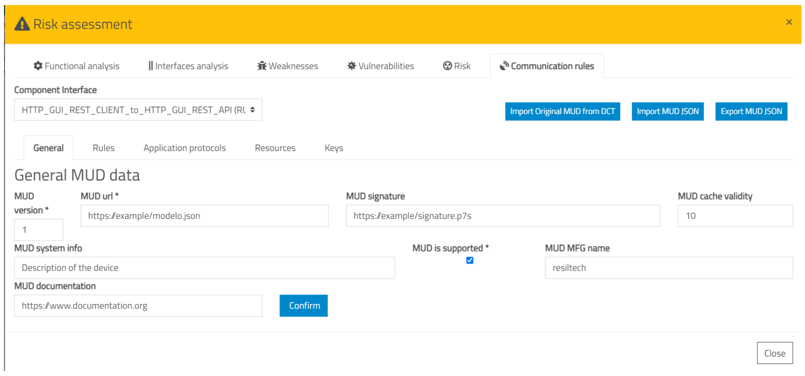
5.2. *Integrating the MUD in the ResilBlockly model*

The integration of the MUD standard within ResilBlockly provides useful support for the user in the task of creating an original MUD file, as well as a source of information for the completeness of the system modelling, especially in complex CPSoS when there are third party components and we need to integrate their interaction in the model.

Using Resilblockly, the user can directly import a MUD file, in its original version, for the selected component of the model. For a complex CPSoS with a high number of subsystems and components, it may be necessary several

[5]https://drive.google.com/file/d/1ZSTD314xiFT4f2eLrhWumSDY-CIVOLeJ/view?usp=share_link

MUD files, which in some cases could not be available for all of them, which is one of the main limitations of the proposed approach. In this case, it would be useful to use external tools such as MUDgee [56], which generates a hypothetical MUD file based on the network traces of a component or MUD Maker, which provides a user friendly GUI to introduce the information to generate the MUD, or Resilblockly, which has been also improved to support the manual generation of the original MUD file in case it is not available. This information is later integrated into the model to enrich it. In this case, it is fundamental that the data in the selected file apply to the model, which means that the value of the field "matches" in the selected JSON of the MUD should be the same as the name of the component interface in the model. Figure 7 shows the graphical user interface (GUI) designed and implemented within Resilblockly to support the integration of the original MUD file. As shown, the user can write manually the fields of the MUD, similarly to existing tools such as MUD Maker, or import the MUD file, which automatically fills in the form fields.


Risk assessment

Functional analysis | Interfaces analysis | Weaknesses | Vulnerabilities | Risk | Communication rules

Component Interface: HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API (RU

Import Original MUD from DCT | Import MUD JSON | Export MUD JSON

General | Rules | Application protocols | Resources | Keys

General MUD data

MUD version *	1
MUD url *	https://example/modelo.json
MUD signature	https://example/signature.p7s
MUD cache validity	10
MUD system info	Description of the device
MUD is supported *	✓
MUD MFG name	resiltech
MUD documentation	https://www.documentation.org

Confirm

Close

Figure 7: GUI implemented for the integration of the MUD file in Resilblockly

Listing 3 in Appendix A (non-bold fields) shows an extract of one of the MUD files we used as input for the ICT Gateway model, whereas part of the model generated as a result of this input is shown in Figure 8. The model takes as input all the information about the system components (e.g., GUI, Application_Layer, ICT GW, EM_INFRASTRUCTURE, etc.) contained in their respective MUD files. In particular, for the ICT GW (CS), we can see how the information contained in the MUD file specifically inside the ACL *HTTP_GW_GUI_REST_CLIENT_to_HTTP_GUI_REST_A* from line 57, defines the communication between the ICT Gateway and the GUI CS. This information has been integrated into the ICT GW component model as an interface RUMI. Similarly, the GUI CS describes this communication in the other direction.

It is worth noting that the MUD file is intended to be created by the manufacturer during the design of the system or component. That means that it is created without knowing the context in which it is going to be deployed. Therefore, when the component is integrated into a CPSoS or deployed in the operational domain, it may occur policy conflicts between the existing policies and the MUD policies. This could imply a manual resolution of conflicts.

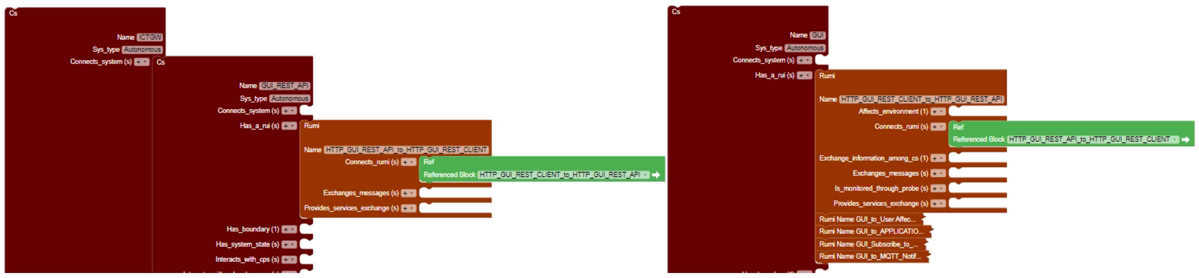

Cs
Name ICTGW
Sys_type Autonomous
Connects_system (s)
Cs
Name GUI_REST_API
Sys_type Autonomous
Connects_system (s)
Has_a_rui (s)
Rumi
Name HTTP_GUI_REST_API_to_HTTP_GUI_REST_CLIENT
Connects_rumi (s)
Ref
Referenced Block HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API
Exchanges_messages (s)
Provides_services_exchange (s)
Has_boundary (1)
Has_system_state (s)
Interacts_with_cps (s)
Cs
Name GUI
Sys_type Autonomous
Connects_system (s)
Has_a_rui (s)
Rumi
Name HTTP_GUI_REST_CLIENT_to_HTTP_GUI_REST_API
Affects_environment (1)
Connects_rumi (s)
Ref
Referenced Block HTTP_GUI_REST_API_to_HTTP_GUI_REST_CLIENT
Exchange_information_among_cs (1)
Exchanges_messages (s)
Is_monitored_through_probe (s)
Provides_services_exchange (s)
Rumi Name GUI_to_User Affec...
Rumi Name GUI_to_APPLICATIO...
Rumi Name GUI_Subscribe_to_...
Rumi Name GUI_to_MQTT_Notif...

Figure 8: ICT gateway model - interfaces

5.3. Risk assessment

ResilBlockly assists the user in conducting risk assessments leveraging the integration with online catalogues of threats to security as CWE, CVE, Common Attack Pattern Enumeration and Classification (CAPEC)[6], National Vulnerability Database (NVD)[7] and scoring systems as CVSS. In the use case context, the purpose of the risk assessment is to determine the risk information about a particular ICT solution, part of the supply chain, that constitutes the system under analysis and is examined for identifying potential weaknesses, vulnerabilities, and attack patterns, as well as impacts to the system components that could, later on, constitute an issue, especially in case of propagation over for the supply chain.

To facilitate this process, ResilBlockly leverages public databases to associate a list of weaknesses and vulnerabilities associated with the system components. However, the list can be appropriately integrated with custom weaknesses and vulnerabilities which may be retrieved from different sources. A risk is associated with each weakness and vulnerability encountered, calculated as the product between the impact (*Base Score* from the CVSS standard, where available, or established by the user otherwise) and the likelihood (established by the user), following the NIST approach and risk matrix [45].

For the ICT GW component, and specifically for the interface mentioned before, *HTTP_GW_GUI_REST_CLIENT_to _HTTP_GUI_ REST_API*, 11 CVEs have been associated with it. We selected three potential weaknesses/vulnerabilities to be included in the extended MUD file:

- CWE-648: Incorrect Use of Privileged APIs - Description: The application does not conform to the API requirements for a function call that requires extra privileges. This could allow attackers to gain privileges by causing the function to be called incorrectly
- CVE-2016-10735: In Bootstrap 3.x before 3.4.0 and 4.x-beta before 4.0.0-beta.2, XSS is possible in the data-target attribute, a different vulnerability than CVE-2018-14041.
- CVE-2018-14041 In Bootstrap before 4.1.2, XSS is possible in the data-target property of scroll spy.

5.4. Generating the extended MUD from ResilBlockly

Listing 3 shows an excerpt of the extended MUD JSON pertaining to interfaces of the ICT GW model generated from ResilBlockly. In particular, the interface from the ICT GW to the GUI (*HTTP_GW_GUI_REST_CLIENT_to_HTTP_GUI _REST_API*) and the interface from the ICT GW to the database (*ICT_GW_to_GW_DATAMODEL_DB*). The information contained in the file is in part retrieved from the modelled components and from the results of the Risk Assessment (e.g., the Weaknesses and Vulnerabilities, as well as the risk-related information), and in part retrieved from the refinement of the user over the model.

As we can see, the information obtained from the risk assessment, in particular the three CVEs selected in the previous section, have been integrated into the weaknesses and vulnerabilities fields (lines 17 to 42). Each vulnerability described includes the impact, likelihood and risk calculated in the risk assessment phase, as well as additional information available in the public database. Lines from 43 to 80 describe the communication between the ICT GW component of the model and the GUI REST API (*HTTP_GW_GUI_REST_CLIENT_to_HTTP_GUI_REST_API* interface), whereas lines from 82 to 101 describe the communication between the ICT GW and the database. All the ACLs are firstly defined (lines from 6 to 16), indicating the direction of the communication e.g., *from* the device or *to* the device. The first ACE includes characteristics of the used protocols at network (ipv4, line 51), transport (tcp, line 55) and application level (HTTP, line 63). Additionally, the MUD integrates fine-grained security information such as the protocol version, the maximum number of connections allowed, and the cryptographic suite configuration (line 70) recommended for the integrity protection of the exchanged messages. The second ACL, related to the communication with the database, shows how the proposed extension can reflect this type of communication (line 88), beyond network components and services.

[6]https://capec.mitre.org/
[7]https://nvd.nist.gov/

Table 3: comparison

Concept	**Extended MUD**	**[27]**	**[59]**	**[57]**	**[58]**	**[14]**
Application layer protocols	x	x (HTTP)		x (DTLS, TLS)		**x**
Databases	x					
Vulnerabilities	x					
Weaknesses	x					
Cryptography	keys and certificates	keys	certificates	keys and certificates		**x**
Connection restriction	x	x				**x**
Authorisation	x	x				**x**
SBOM/HBOM	x				SBOM	

5.5. Evaluation

Figure 9 compares the amount of information specified in the original and the extended MUD files generated for the ICT GW. In the original MUD file of the ICT GW, the MUD container represents 60% of the information, whereas the ACL container describes the 40% remaining. From this data, 71,43% has been integrated into the initial system model. The graph shows how the extended MUD doubles the information defined in the original one, and how the majority of the new information has been placed in the ACL container, which represents now 111,4% of the amount of data with respect to the original MUD.

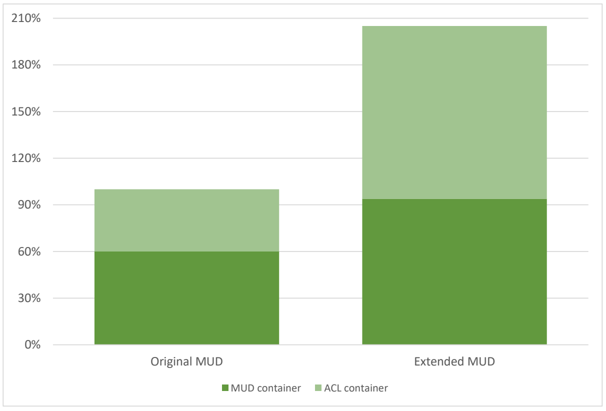


Figure 9: Extension of the MUD model separated by container

Table 3 compares different proposals for an extended MUD. In particular, we focus on similar works that address the concepts that have been considered in the proposed extension. Thus, we find that [27], [57] and [14] also consider some application-level protocols and cryptographic configuration, while other proposals focus on specific aspects such as the inclusion of a reference to the SBOM ([58]). The most complete extension is proposed at [14]. However that extension does not maintain the standard structure of the MUD, but instead adds an appendix to the ACL block to describe more complex policies. Our proposal integrates concepts that have been taken into account in several extensions, as well as additional concepts such as the relationship with databases or the presence of vulnerabilities and weaknesses while keeping the MUD structure. Furthermore, none of these works addresses the automatic generation of the extended MUD that is proposed, while we propose the integration of the extended MUD generation with the model of the system.

Figure 10 compares the generation time of the original MUD file using the well-known tool MUD maker, and the generation time of the extended MUD using Resilblockly functionality. The graph also shows the time spent by a user filling in manually all the fields required for the ICT GW MUD and the time the tool takes to generate the corresponding MUD. The user input time was measured using Selenium IDE in Mozilla Firefox. Naturally, the generation time of the extended MUD, 2’3 s for generation and 21’1 s for user input, is higher than in the original one, which is reduced to 0’9 s for generation and 3’1 s for user input. However, the extended MUD allows to describe details that were not within the scope of the original MUD and the implementation made on Resilblockly allows certain customisation aspects such as the nomination of aces, something that MUD maker does not allow, implying an additional input for the user. In addition, it is important to mention that the MUD generation process is understood as something punctual, at the end of the system development. Therefore, the time spent in such a generation may not be as critical as the amount of information that can be described.

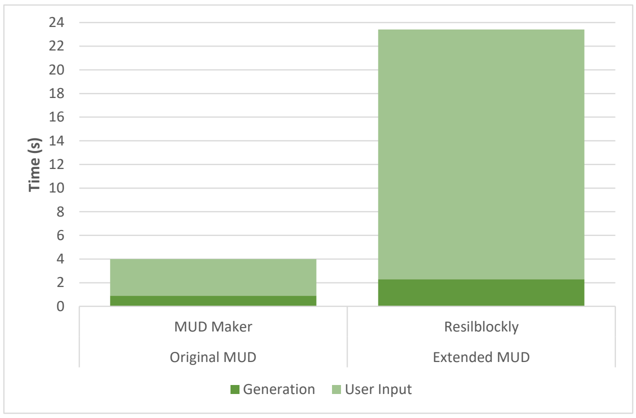


Figure 10: Generation time comparison between MUD Maker (original MUD) and Resilblockly (extended MUD)

6. Conclusions and future work

The high hyper-connectivity to which CPSs are subject creates new attack surfaces through which the system can be compromised, so early and continuous risk management is key to reducing possible cyberattacks. In this paper we have analysed some of the main challenges of modelling techniques, a technique focused on describing the expected behaviour of the system with the aim of detecting malfunctions or security vulnerabilities through a subsequent risk assessment process. Among these challenges, we highlight the complexity of the CPSoS modelling process, which usually has black box components created by third parties that must be integrated into the model. Our work analyses the possibilities of integrating the modelling process with the MUD standard, which despite having certain expressiveness limitations, allows the manufacturer to describe the behaviour of a component in a standardised and public way. We have studied which information from the MUD could be integrated into the model, filling in the gaps of external components.

As a second result, we have discussed the benefits of taking advantage of the modelling and risk assessment results to automatically generate an extended behavioural profile based on the MUD structure to increase the expressiveness of the file and extend the utility of the MUD standard. In this way, the possibility of using it during the runtime phase opens up to monitor suspicious behaviours, launch recommended security policies or mitigations after a security analysis or even discern if a component is secure enough to be part of a larger system.

The usefulness of both results has been validated by applying them to a use case in a real CPS, in the context of smart grid, using the Resilblockly modelling tool developed in the BIECO project. The results obtained help to visualise the scope of the profile extension, which doubles the information contained in the original MUD, and its integration in the modelling, since 71% of the information has been obtained from the system model and the risk assessment process.

Future work has been planned for some of the limitations analysed in the valuation; i) the policy conflict resolution in case there is a mismatch between the policies of the system and the policies described in a new system component, which could derive in conflicts when integrating the MUD in the model, or ii) the necessity of a MUD file for the integration, which requires the consideration of tools to generate a draft of the profile and may not be accurate enough. Finally, another working path focuses on the subsequent use of the extended profile. Although there are already works that address its use in attack monitoring and detection, as well as deployment of security policies, one of the advantages of CPSoS is related to the feasibility of analysing whether a component is adequate in terms of security to be part of a complex system, applying possible reconfigurations to enhance it.

Acknowledgements

This work has been partially funded by the European Commission through the projects H2020-952702 BIECO, H2020-830929 CyberSec4Europe and H2020-101021936 ELECTRON.

Appendix A. ICT GW MUD model

Listing 3: Extended MUD Exported from ResilBlockly ICT GW model

```
{
"ietf-mud:mud": {
  "mud-version": "1",
  "mud-url": "https://www.ICTGatewayGUI.com/ictgw.json",
   ...
  "from-device-policy": {
     "access-lists": {"access-list": [
     {"name": "HTTP_GW_GUI_REST_CLIENT_to_HTTP_GUI_REST_API"},
     {"name": "ICT_GW_to_GW_DATAMODEL_DB"},
      ...
     ]}},
  "to-device-policy": {
     "access-lists": {"access-list": [{
        "name": "HTTP_GW_GUI_REST_CLIENT_from_HTTP_GUI_REST_API"},
         ...
  ]}},
  "weaknesses":
     [{ "id": "CWE-648",
        "name": "Incorrect Use of Privileged APIs",
        "description": "The application does not conform to...",
        "date": "2021-06-22",
        "likelihood": "Low",
        "impact": "Moderate",
        "risk": "Low"
     }],
  "vulnerabilities":
     [{ "id": "CVE-2016-10735",
        "name": "CVE-2001-1494",
        "description": " In Bootstrap 3.x before 3.4.0 and 4.x-beta ...",
        "date": "2021-06-14",
        "likelihood": "Moderate",
        "risk": "Moderate",
        "cvss": "6.1"
     },
     {  "id": "CVE-2018-14041",
        "name": "CVE-2001-1494",
        "description": "In Bootstrap before 4.1.2, XSS is possible ...",
        "date": "2021-06-14",
        "likelihood": "Moderate",
        "risk": "Moderate",
        "cvss": "6.1"
     }]},
"ietf-access-control-list:acls": {
  "acl": [
     { "name": "HTTP_GW_GUI_REST_CLIENT_to_HTTP_GUI_REST_API",
        "type": "ipv4-acl-type",
        "aces": {
           "ace": [
              { "name": "5074-Rule1",
                 "matches": {
                    "ipv4": {
                       "protocol": 6,
                       "ietf-acldns:dst-dnsname": "ict-gateway-gui"
                    },
                    "tcp": {
                       "source-port": {
                          "port": 8883
                    },
                    "destination-port": {
                      "port": 8884
                    }
                  },
                  "applicationProtocol": [
                        {  "protocol": "HTTP REST",
                           "version": "3.1.1",
                           "numConnections": 1,
                           "keepAlive": 60
                        }
                     ],
                  "keys": [
                     {  "kty": "EC",
                        "alg": "ECDH-ES",
                        "crv": "P-256",
                        "length": 256,
                        "key_ops": "enc",
                        "purpose": "integrity"
                     }]
                },
                "actions": {"forwarding": "accept"}
              }]]}
              {
        "name": "ICT_GW_to_GW_DATAMODEL_DB",
        "type": "ipv4-acl-type",
        "aces": {
          "ace": [
            { "name": "5074-Rule1-frdev",
                "matches": {
                  (*\bfseries "ietf-mud:mud": \{"database": "datamodel"\}, } @*)
                  "ipv4": {
                    "protocol": 6,
                    "ietf-acldns:dst-dnsname": "gw-datamodel"},
                  "tcp": {
                    "source-port": {
                      "port": 8020},
                    "destination-port": {
                      "port": 8021
                    }}
                },
                "actions": {
                  "forwarding": "accept"
        }}]}}
        ...
      ]}}
```

References

[1] E. Parliament, REGULATION (EU) 2019/881 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 17 April 2019 on ENISA (the European Union Agency for Cybersecurity) and on information and communications technology cybersecurity certification (Cybersecurity Act) (2019).
URL `https://eur-lex.europa.eu/eli/reg/2019/881/oj`

[2] NIST, Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1, Tech. rep., National Institute of Standards and Technology (2018).
URL `https://doi.org/10.6028%2Fnist.cswp.04162018`

[3] S. J. Oks, M. Jalowski, A. Fritzsche, K. M. Möslein, Cyber-physical modeling and simulation: A reference architecture for designing demonstrators for industrial cyber-physical systems, Procedia CIRP 84 (2019) 257–264, 29th CIRP Design Conference 2019, 08-10 May 2019, Póvoa de Varzim, Portgal. `doi:https://doi.org/10.1016/j.procir.2019.04.239`.
URL `https://www.sciencedirect.com/science/article/pii/S2212827119308868`

[4] S. N. Matheu-García, J. L. Hernández-Ramos, A. F. Skarmeta, G. Baldini, Risk-based automated assessment and testing for the cybersecurity certification and labelling of IoT devices, Computer Standards & Interfaces 62 (2019) 64–83.

[5] S. N. Matheu, J. L. Hernández-Ramos, A. F. Skarmeta, G. Baldini, A Survey of Cybersecurity Certification for the Internet of Things, ACM Computing Surveys (CSUR) 53 (6) (2020) 1–36.

[6] E. Lear, D. Romascanu, R. Droms, Manufacturer Usage Description Specification (RFC 8520) (2019).
URL `https://tools.ietf.org/html/rfc8520`

[7] T. Polk, M. Souppaya, W. C. Barker, Mitigating IoT-Based Automated Distributed Threats (2017).
URL `https://www.nccoe.nist.gov/sites/default/files/library/project-descriptions/iot-ddos-project-description-draft.pdf`

[8] ENISA, Good Practices for Security of IoT - Secure Software Development Lifecycle.
URL `https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot-1`

[9] J. L. Hernandez-Ramos, S. N. Matheu, A. Feraudo, G. Baldini, J. B. Bernabe, P. Yadav, A. Skarmeta, P. Bellavista, Defining the Behavior of IoT Devices Through the MUD Standard: Review, Challenges, and Research Directions, IEEE Access 9 (2021) 126265–126285. `doi:10.1109/ACCESS.2021.3111477`.
URL `https://ieeexplore.ieee.org/document/9531614/`

[10] E. Lear, J. Henry, R. Barton, Determining Nominal Quality of Service needs of a Device.
URL `https://www.tdcommons.org/dpubs_series/1625`

[11] M. Bjorklund, The YANG 1.1 data modeling language (RFC 7950) (2016).
URL `https://tools.ietf.org/html/rfc7950`

[12] T. Bray, The JavaScript Object Notation (JSON) Data Interchange Format (RFC8259) (2017).
URL `https://tools.ietf.org/html/rfc8259`

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

[14] S. N. Matheu, A. Robles Enciso, A. Molina Zarca, D. Garcia-Carrillo, J. L. Hernández-Ramos, J. Bernal Bernabe, A. F. Skarmeta, Security architecture for defining and enforcing security profiles in DLT/SDN-Based IoT systems, Sensors 20 (7) (2020) 1882.

[15] S. N. M. García, A. Molina Zarca, J. L. Hernández-Ramos, J. B. Bernabé, A. S. Gómez, Enforcing Behavioral Profiles through Software-Defined Networks in the Industrial Internet of Things, Applied Sciences 9 (21) (2019) 4576.

[16] M. Brambilla, J. Cabot, M. Wimmer, Model-driven software engineering in practice, Synthesis lectures on software engineering 3 (1) (2017) 1–207.

[17] K. Henderson, A. Salado, Value and benefits of model-based systems engineering (mbse): Evidence from the literature, Systems Engineering 24 (1) (2021) 51–66.

[18] D. M. Nicol, W. H. Sanders, K. S. Trivedi, Model-based evaluation: from dependability to security, IEEE Transactions on dependable and secure computing 1 (1) (2004) 48–65.

[19] J. Zander, I. Schieferdecker, P. J. Mosterman, Model-based testing for embedded systems, CRC press, 2017.

[20] T. Ahmad, J. Iqbal, A. Ashraf, D. Truscan, I. Porres, Model-based testing using uml activity diagrams: A systematic mapping study, Computer Science Review 33 (2019) 98–112.

[21] A. B. Feeney, S. Frechette, V. Srinivasan, Cyber-Physical Systems Engineering for Manufacturing, 2017. `doi:10.1007/978-3-319-42559-7_4`.

[22] A. Bondavalli, S. Bouchenak, H. Kopetz, Cyber-physical systems of systems: foundations–a conceptual model and some derivations: the AMADEOS legacy, Vol. 10099, Springer, 2016.

[23] A. Bennaceur, C. Ghezzi, K. Tei, T. Kehrer, D. Weyns, R. Calinescu, S. Dustdar, Z. Hu, S. Honiden, F. Ishikawa, et al., Modelling and analysing resilient cyber-physical systems, in: 2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), IEEE, 2019, pp. 70–76.

[24] N. Mazhar, R. Salleh, M. Zeeshan, M. M. Hameed, Role of Device Identification and Manufacturer Usage Description in IoT Security: A Survey, IEEE Access 9 (2021) 41757–41786. `doi:10.1109/ACCESS.2021.3065123`.

[25] V. Andalibi, J. Dev, D. Kim, E. Lear, L. J. Camp, Is Visualization Enough? Evaluating the Efficacy of MUD-Visualizer in Enabling Ease of Deployment for Manufacturer Usage Description (MUD), in: ACSAC '21: Annual Computer Security Applications Conference, Association for Computing Machinery, New York, NY, USA, 2021, pp. 337–348. `doi:10.1145/3485832.3485879`.

[26] V. Andalibi, E. Lear, D. Kim, L. J. Camp, On the Analysis of MUD-Files' Interactions, Conflicts, and Configuration Requirements Before Deployment, arXiv`arXiv:2107.06372`, `doi:10.48550/arXiv.2107.06372`.

[27] S. N. Matheu, J. L. Hernandez-Ramos, S. Perez, A. F. Skarmeta, Extending MUD profiles through an Automated IoT Security Testing Methodology, IEEE Access (2019) 1–20`doi:10.1109/ACCESS.2019.2947157`.

[28] M. Rocchetto, A. Ferrari, V. Senni, Challenges and Opportunities for Model-Based Security Risk Assessment of Cyber-Physical Systems, in: Resilience of Cyber-Physical Systems, Springer, Cham, Switzerland, 2019, pp. 25–47. `doi:10.1007/978-3-319-95597-1_2`.

[29] M. H. Mazhar, Z. Shafiq, Characterizing Smart Home IoT Traffic in the Wild`arXiv:2001.08288`.
URL `http://arxiv.org/abs/2001.08288`

[30] R. Fontein, E. Khan, For Whom the IoT-Bell Tolls 3.
URL `https://telluur.com/utwente/master/SSI%20-%20Security%20Services%20for%20the%20IoT/Project/For_Whom_the_IoT_Bell_Tolls.pdf`

[31] Z. Jin, Y. M. Lee, C. H. Copass, Y. Park, Building system with dynamic Manufacaturer Usage Description (MUD) files based on building model queries, uS Patent App. 16/666,005 (Apr. 30 2020).

[32] M. Hanes, C. Byers, J. Clarke, G. Salgueiro, Human usage description for 5G networks endpoints.
URL `https://www.tdcommons.org/dpubs_series/1254`

[33] SOFTWARE BILL OF MATERIALS | National Telecommunications and Information Administration, [Online; accessed 25. Apr. 2022] (Apr. 2022).
URL `https://ntia.gov/SBOM`

[34] M. Al-Shaboti, I. Welch, A. Chen, M. A. Mahmood, Towards Secure Smart Home IoT - Manufacturer and User Network Access Control Framework, in: IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA), pp. 892–899. `doi:10/gfwwpn`.

[35] L. Chang, A Proactive Approach to Detect IoT Based Flooding Attacks by Using Software Defined Networks and Manufacturer Usage Descriptions.

[36] H. J. Hadi, S. M. Sajjad, K. un Nisa, BoDMitM: Botnet Detection and Mitigation System for Home Router Base on MUD, in: 2019 International Conference on Frontiers of Information Technology (FIT), pp. 139–1394. `doi:10.1109/FIT47737.2019.00035`.

[37] S. M. Sajjad, M. Yousaf, H. Afzal, M. R. Mufti, eMUD: Enhanced Manufacturer Usage Description for IoT Botnets Prevention on Home Wifi Routers, IEEE Access 8 (2020) 164200–164213.

[38] G. Baldini, J. L. Hernandez-Ramos, S. Nowak, R. Neisse, M. Nowak, Mitigation of Privacy Threats due to Encrypted Traffic Analysis through

a Policy-Based Framework and MUD Profiles, Symmetry 12 (9) (2020) 1576.
[39] A. Hamza, D. Ranathunga, H. H. Gharakheili, M. Roughan, V. Sivaraman, Clear as MUD: Generating, validating and applying IoT behavioral profiles, in: Proceedings of the 2018 Workshop on IoT Security and Privacy, 2018, pp. 8–14.
[40] A. Hamza, H. H. Gharakheili, V. Sivaraman, Combining MUD Policies with SDN for IoT Intrusion Detection, in: Proceedings of the 2018 Workshop on IoT Security and Privacy, ACM, NY, USA, 2018, pp. 1–7. `doi:10/gfth8h`.
[41] A. Hamza, D. Ranathunga, H. H. Gharakheili, T. A. Benson, M. Roughan, V. Sivaraman, Verifying and Monitoring IoTs Network Behavior using MUD Profiles, arXiv:1902.02484 [cs].
URL `http://arxiv.org/abs/1902.02484`
[42] L. Bjorklund M., Berger, RFC 8340 - YANG Tree Diagrams (2018).
URL `https://datatracker.ietf.org/doc/html/rfc8340`
[43] CWE - Common Weakness Enumeration.
URL `https://cwe.mitre.org`
[44] CVE - CVE.
URL `https://cve.mitre.org`
[45] J. T. F. T. Initiative, Guide for Conducting Risk Assessments, CSRC | NIST`doi:10.6028/NIST.SP.800-30r1`.
[46] Common Vulnerability Scoring System SIG.
URL `https://www.first.org/cvss`
[47] M. Jones, JSON Web Key (JWK), RFC 7517 (May 2015). `doi:10.17487/RFC7517`.
URL `https://rfc-editor.org/rfc/rfc7517.txt`
[48] M. Jones, JSON Web Algorithms (JWA), RFC 7518 (May 2015). `doi:10.17487/RFC7518`.
URL `https://rfc-editor.org/rfc/rfc7518.txt`
[49] Directive (EU) 2016/1148 of the European Parliament and of the Council of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union (NIS directive) (2016).
URL `https://eur-lex.europa.eu/eli/dir/2016/1148/oj`
[50] A. Hamza, H. H. Gharakheili, T. A. Benson, V. Sivaraman, Detecting Volumetric Attacks on IoT Devices via SDN-Based Monitoring of MUD Activity, in: Symposium on SDN Research (SOSR), California, USA, 2019, pp. 36–48.
[51] E. Schiavone, N. Nostro, F. Brancati, A mde tool for security risk assessment of enterprises, in: Anais do X Latin-American Symposium on Dependable Computing, SBC, 2021, pp. 5–7.
[52] N. Nostro, et al., ICT Analysis and Gateway Design". Deliverable D3.1 of H2020 project Net2DG, "Leveraging Networked Data for the Digital Electricity Grid (2018).
[53] A. Babu, S. Iacob, P. Lollini, M. Mori, Amadeos framework and supporting tools, in: Cyber-Physical Systems of Systems, Springer, 2016, pp. 128–164.
[54] Blockly4sos.
URL `https://blockly4sos.resiltech.com`
[55] Google Blockly.
URL `https://developers.google.com/blockly/`
[56] A. Hamza, D. Ranathunga, H. H. Gharakheili, M. Roughan, V. Sivaraman, Clear as MUD: Generating, Validating and Applying IoT Behavioral Profiles (Technical Report), arXiv`arXiv:1804.04358`, `doi:10.1145/3229565.3229566`.
[57] MUD (D)TLS profiles for IoT devices, [Online; accessed 21. Mar. 2023] (Mar. 2023).
URL `https://datatracker.ietf.org/doc/html/draft-reddy-opsawg-mud-tls-05#page-5`
[58] Discovering And Accessing Software Bills of Materials, [Online; accessed 21. Mar. 2023] (Mar. 2023).
URL `https://datatracker.ietf.org/doc/html/draft-lear-opsawg-sbom-access-00#page-5`
[59] [Online; accessed 21. Mar. 2023] (Feb. 2021). [link].
URL `https://openconnectivity.org/specs/OCF_Security_Specification_v2.2.2.pdf`