A model-based approach for assessing the security of cyber-physical systems

Hugo Teixeira de Castro, Ahmed Hussain, Gregory Blanc, Jamal El Hachem, Dominique Blouin, Jean Leneutre, Panos Papadimitratos

A Model-based Approach for Assessing the Security of Cyber-Physical Systems

Hugo TEIXEIRA DE CASTRO
SAMOVAR, Telecom SudParis,
Institut Polytechnique de Paris
Palaiseau, France
hugo.teixeiradecastro@hotmail.com

Ahmed Hussain
KTH Royal Institute of Technology
Networked Systems Security Group
Stockholm, Sweden
ahmhus@kth.se

Jamal EL HACHEM
Université Bretagne Sud, UMR CNRS
6074, IRISA
Vannes, France
jamal.el-hachem@irisa.fr

Gregory Blanc
SAMOVAR, Telecom SudParis,
Institut Polytechnique de Paris
Palaiseau, France
gregory.blanc@telecom-sudparis.eu

Dominique Blouin
LTCI, Telecom Paris,
Institut Polytechnique de Paris
Palaiseau, France
dominique.blouin@telecom-paris.fr

Jean Leneutre
LTCI, Telecom Paris,
Institut Polytechnique de Paris
Palaiseau, France
jean.leneutre@telecom-paris.fr

Panos Papadimitratos
KTH Royal Institute of Technology
Networked Systems Security Group
Stockholm, Sweden
papadim@kth.se

ABSTRACT

Cyber-Physical Systems (CPSs) complexity has been continuously increasing to support new life-impacting applications, such as Internet of Things (IoT) devices or Industrial Control Systems (ICSs). These characteristics introduce new critical security challenges to both industrial practitioners and academics. This work investigates how Model-Based System Engineering (MBSE) and attack graph approaches could be leveraged to model secure Cyber-Physical System solutions and identify high-impact attacks early in the system development life cycle. To achieve this, we propose a new framework that comprises (1) an easily adoptable modeling paradigm for Cyber-Physical System representation, (2) an attack-graph-based solution for Cyber-Physical System automatic quantitative security analysis, based on the MulVAL security tool, (3) a set of Model-To-Text (MTT) transformation rules to bridge the gap between SysML and MulVAL. We illustrated the validity of our proposed framework through an autonomous ventilation system example. A Denial of Service (DoS) attack targeting an industrial communication protocol was identified and displayed as attack graphs. In future work, we intend to connect the approach to dynamic security databases for automatic countermeasure selection.

CCS CONCEPTS

• **Computer systems organization** → **Dependable and fault-tolerant systems and networks**; *Redundancy*; **Embedded and cyber-physical systems**; *Embedded systems*; Robotics; • **Networks** → **Network properties**; *Network reliability*.

KEYWORDS

Risk Analysis, Security and Privacy for Cyber-Physical Systems, Critical Infrastructures, Threats and Attack Modelling, Usable Security and Privacy, Security by Design.

ACM Reference Format:
Hugo TEIXEIRA DE CASTRO, Ahmed Hussain, Jamal EL HACHEM, Gregory Blanc, Dominique Blouin, Jean Leneutre, and Panos Papadimitratos. 2024. A Model-based Approach for Assessing the Security of Cyber-Physical Systems. In *The 19th International Conference on Availability, Reliability and Security (ARES 2024), July 30-August 2, 2024, Vienna, Austria.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3664476.3670470

1 INTRODUCTION

A Cyber-Physical System (CPS) is a system in which physical and cyber components collaborate to achieve a given goal or function; they are the integration of computation with physical processes [27]. Typical CPSs components include sensors (to retrieve physical values from the environment), computing units (to make decisions based on the sensed values), and actuators (to impact the environment in a desired way). Nowadays, CPSs are used in a large number of applications, often with a significant societal impact, such as energy-aware buildings [24], robotic systems [53], or autonomous vehicles traffic flow management [12].

CPSs are also particularly vulnerable and face increased cyber risk [58]. Indeed, interconnecting the cyber and physical worlds increases the attack surface and gives rise to new impactful security threats. They are also susceptible to physical attacks, with

*This is a personal copy of the authors. Not for redistribution. The final version of the paper is available in the Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES'24)

ARES 2024, July 30-August 2, 2024, Vienna, Austria

ACM ISBN 979-8-4007-1718-5/24/07
https://doi.org/10.1145/3664476.3670470

sensors and actuators geographically distributed, possibly unattended, which can be physically tampered with. Furthermore, CPSs have very specific requirements, such as safety, availability, or time-criticality, which makes traditional countermeasures (e.g., firewalls, anti-viruses, public key cryptography) less effective [49]. In recent years, we witnessed an increase in cyber attacks on CPSs, some of which had a tremendous social and economic impact. The Stuxnet [23] attack damaged about 2000 centrifuges in Iranian nuclear facilities and delayed the entire nuclear program by two years. In 2021, the Verkada hack exposed over 150,000 security cameras in many locations such as factories, jails, or hospitals [21]. Such examples motivate the need for a methodology that enables specifying the expected security requirements and deploying enforcement points.

To prevent such events, a proper Risk Analysis (RA) of the entire system is necessary. Ideally, this RA should be performed from an infrastructure perspective [14], as various works showed that component-wise solutions do not suffice [10, 43]. However, this makes the entire process complex, especially for those with limited cybersecurity knowledge. A promising solution would be to take inspiration from model-based approaches. Indeed, models are simplified representations that can cover multiple scales [47], from high-level perspectives to a component's inner parts and behavior. Furthermore, the relations between the different system components are formalized, and model-checking can be used to verify the overall consistency [6]. These features and layer of abstraction are extremely useful to manage the complexity.

Besides, security experts can use models to design and integrate countermeasures into the rest of the system. Automating the security analysis at this stage is provably more advantageous, as manual assessments are time-consuming, error-prone, and even infeasible in some contexts [51]. Models can also be used at later stages to test countermeasures and run simulations. Ultimately, it could provide a wide overview of the system's security risks, even in complex and intricate CPSs. However, though model-based practices are widely common in systems engineering and software development to model functional and non-functional requirements, their application to security is rare [17, 36]

Research Question. This paper addresses the following research question: *What specific workflows, combining appropriate and accessible modeling approaches, can be used to represent complex CPSs at different levels of abstraction and assess their level of security?*

Contributions. The framework presented in this paper aims to fill this gap. Specifically, we provide the following:

- A model-based framework that represents CPS-specific components and vulnerabilities by extending and combining the SysML and MulVAL tools.
- An accessible and intuitive approach that does not require deep security expertise from the system architects.
- MTT transformations and name-matching algorithm to automate the attack graph generation process from a system model.
- Validation of the effectiveness of our framework by applying it to a simplified small-scale view of a real-world smart building.

Paper Organization. The remainder of this paper is organized as follows: Section 2 provides the preliminary and background knowledge on the topics associated with RA and MBSE. Section 3 illustrates the solution's properties and design, while Section 4 elaborates on the technical implementation details. Section 5 presents the case study used to evaluate the proposed approach. Finally, Section 6 lists all the relevant related work and Section 7 wraps up the findings in this paper and addresses potential future work.

2 BACKGROUND

This section introduces the core concepts of the proposed implementation. We present key elements of RA and discuss the two main paradigms we used in our implementation, namely, MBSE and attack graphs.

2.1 Model-Based System Engineering

Definition and Advantages. Model-Based System Engineering (MBSE) is a standard approach in systems engineering in which the entire development process is centered around models of the system to be built [41]. MBSE is widely adopted in system engineering because of its many advantages [47]. Indeed, models are less complex than real systems, making them more accessible. Furthermore, the models can easily be changed to experiment with various design solutions. Finally, models often have a formal, machine-readable syntax that enables automatic model analysis and verification. The use of MBSE approaches in cybersecurity could have several benefits, such as reduced complexity in the representation of large physical systems or the automatic inference of security risks and vulnerabilities.

Models including details about the CPS components and the functional architecture can thus be automatically analyzed to infer security risks or vulnerabilities.

SysML. The Systems Modeling Language (SysML) [63] was designed as an extension of Unified Modeling Language (UML) through profiles to better support systems engineering. SysML reuses UML's set of diagrams, in addition to integrating its own extensions. For instance, UML class diagrams, renamed Block Definition Diagrams (BDDs) in SysML, can be used to represent different system components using the concept of block. This is considered a more generic object type than a UML class, enabling the representation of abstract, high-level components or sub-parts of the system. Moreover, SysML introduces Internal Block Diagrams (IBDs), static, structural diagrams, owned by a particular block, that illustrate its encapsulated structural contents, such as its parts, properties, connectors, ports, and interfaces.

We used a SysML-based approach for various reasons. Its main advantage is its versatility, which it inherited from UML. It is generic and easily extensible and can be used to model any type of system, including hardware, software, information, processes, personnel, and facilities. Furthermore, SysML inherits UML's profiling features, enabling the creation of any Domain Specific Language (DSL). Subsequently, numerous SysML extensions have been developed [55], including some to model smart buildings and cities [1]. These extension mechanisms rely on the definition of profiles, which consists of stereotypes, allowing designers to extend the vocabulary of UML to create new model elements.

2.2 Attack Graphs

Formalism Description. Attack graphs [40] are a formalism to represent attack scenarios. They offer a visual representation that illustrates the potential paths an attacker could take to exploit vulnerabilities and compromise a system or network. Nodes in the graph represent various elements, and edges represent the possible connections or exploits that an attacker could use to move from one node to another. Attack graphs are often used in Risk Analysis (RA) to formalize threat scenarios and visualize the system's most vulnerable elements. Ultimately, it enables the selection and prioritization of countermeasures. Attack graphs offer a visually intuitive representation of complex attack scenarios, making them easier to understand even for non-experts. Furthermore, graph structures can lead to more advanced automation and computations, for instance, in this case, of an attack's likelihood or impact.

MulVAL. Multi-host, multi-stage Vulnerability Analysis Language (MulVAL) [38] is an attack graph generation tool written in Datalog [8]. Over the years, many contributors have added their own rules and predicates to model various attack scenarios [52]. The predicates represent the machines in the system, their ports, connections, and programs they run, as well as the firewall rules and existing vulnerabilities. Based on these input parameters, MulVAL applies a set of Datalog rules (of the form P :- P1, P2, i.e., predicate P is true if P1 and P2 are true) to generate the attack graph. Additionally, optional commands allow one to define users and their permissions.

Though MulVAL is neither CPS-specific nor model-based, it is open-source and easily customizable. Custom rules and predicates can be introduced to represent any attack scenario and fill any gap. Additionally, MulVAL is performant and can generate attack graphs in polynomial time [38], which is particularly important for large and complex CPSs.

3 METHODOLOGY

The goal is to develop a model-based approach to identify and quantify attack scenarios in CPS. Indeed, the elicitation of scenarios leading to a risk realization is a crucial step of the risk analysis process. As such, this step should be accessible to people with limited knowledge on security. Furthermore, it should be automated or semi-automated to save time and reduce errors. This section provides a high-level overview of the developed methodology and explains the main steps leading to the generation of an attack graph from a model of the system. A SysML-based language is used to create the model, which is then parsed to generate the MulVAL input code. Finally, the MulVAL engine generates possible attack scenarios as attack graphs. These attack scenarios can then serve as inputs to the rest of the RA process. SysML and MulVAL were selected based on a global evaluation of multiple system modeling and security analysis tools. Criteria observed included, among others, CPS-specificity, accessibility to non-security experts, and compatibility between the different formalisms to produce a straightforward mapping.

Figure 1 depicts the steps to generate an attack graph based on a CPS model. The following Subsections 3.1, 3.2, and 3.3 describe the steps required for an attack graph to be generated.

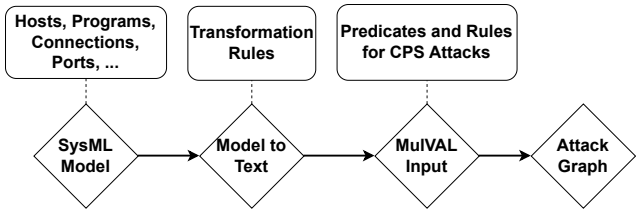

Figure 1: Process for generating an attack graph from a SysML-based model.

3.1 Systems Modeling

The first step is to model the CPS using an extension of SysML. The extension focuses on security-related aspects. Specifically, annotations in the form of UML stereotypes including hosts, programs, and physical devices, are defined. The connections between these components, communication protocols, and ports are also needed to model the system. Finally, stakeholders (e.g., users, employees, etc.) interacting with the system or its parts are modeled, given that they represent potential attack vectors. This information will later be used as input to automatically identify vulnerabilities and attack scenarios.

The SysML extension presented in this paper was created to symbolize all the aforementioned concepts. Indeed, SysML's base metamodel did not meet all the requirements. Basic blocks are too abstract and cannot be used to distinguish different types of components. Besides, SysML only focuses on functional requirements, and no security-related stereotypes are available, which is why we defined the aforementioned extension. Once the model is complete, all the information necessary to perform the security analysis is available.

3.2 Model-to-Text Transformations

Model-To-Text (MTT) transformation refers to the process of generating textual output (e.g., source code or documentation) from a given model or representation [39]. This often involves converting information or data from a structured or machine-readable format into human-readable text. This transformation enables the model parser to identify each of the model's elements and produce the corresponding code. This approach has several advantages. MTT transformations increase productivity, and help the transformation developer use concepts closer to the problem domain at hand, rather than those offered by programming languages [42].

The previously introduced SysML model (Sec. 3.1) is not yet exploitable by our attack graph generation tool. Hence, appropriate transformations are needed to convert the SysML model into a MulVAL-readable file. To achieve this, we formally map each Datalog predicate (MulVAL input information) to its corresponding representation in SysML. The mapping is summarized in Table 1. This method enables the (semi)-automatic transformation of CPS architecture elements into executable artifacts (MulVAL code). Based on the later results from the security analysis, new elements can be iteratively integrated in the model until reaching an acceptable level of security.

A benefit of adding SysML as an additional MulVAL input would be accessibility. Indeed, MulVAL's raw syntax (Datalog) is technical and complex and requires specific expertise to use (especially since MulVAL is not as popular as other programming languages). On the other hand, SysML is diagram-oriented, which makes it easier to represent and visualize the system. It can be easily used by system architects without having to match MulVAL's syntax directly.

Besides, we developed an identification algorithm to extract the type of the CPS components. For example, it is possible to automatically know if a model element is a database, a server, or a sensor. This is shown to be useful during the security analysis, as some attack patterns and vulnerabilities are specific to certain software and hardware. More details are available in Section 4.3.

Table 1: Mappings between new SysML's stereotypes and MulVAL's predicates.

Stereotype	MulVAL predicate
Attacker	attackerLocated(Host)
Program	networkServiceInfo(Host, Prog, Protocol, Port, Usr).
Connection	hacl(Src, Dst, Protocol, Port).
Device	isSensor(Device) or isActuator(Sensor).
User	stakeHolder(Username,Exposure)
hasAccount	hasAccount(User,Host,Permission)
physicalAccess	physicalAccess(User,Device,Permission).
Asset	isAsset(Asset, Importance).

3.3 Automatic Graph Generation

The MulVAL input discussed in Sec. 3.2 is used by MulVAL's engine to automatically generate attack graphs, representing attack events in the system. The nodes in the graph represent the system components as well as their vulnerabilities, which an adversary can exploit. The leaves correspond to the entry points; the root is the attacker's final goal, e.g., to compromise a specific machine.

As already discussed, CPSs are more vulnerable than traditional Information Technology (IT) systems. As such, a key goal was to generate CPS-specific scenarios as attack graphs. This would help system architects identify threats early, consider them in the RA, and implement suitable countermeasures.

To that end, we augmented MulVAL's basic ruleset with additional attacks, such as physical tampering and Common Vulnerability Enumeration (CVE) entries. Additionally, using MulVAL enables performing quantitative analysis and computing the probability of success of an attack. Indeed, each transition in the graph is attributed to an elementary likelihood. Moreover, they are aggregated at every step to get the probability of reaching a specific node and, ultimately, the attack goal. These elementary likelihoods can either be manually entered or extracted from the Common Vulnerability Scoring System (CVSS) scores of the vulnerabilities [16, 31]. These probabilities calculations are paramount for risk prioritization and countermeasure selection.

4 IMPLEMENTATION

This section provides technical details about the presented extensions. The goal is to be able to automatically infer attack scenarios from the information contained in the system's model. In what follows, the new SysML stereotypes, additional MulVAL rules and predicates, and Model-To-Text (MTT) templates are described.

4.1 New stereotypes for the SysML Extension

As discussed earlier, SysML was initially designed for system engineering. Hence, it does not represent complex security properties, which must be manually added. A few existing SysML extensions were proposed to address this [3, 15, 50]. However, the introduced concepts are not sufficient to analyze the CPS models and generate corresponding attack graphs. Therefore, we devised specific custom stereotypes, enhancing the representation of these properties. The modified metamodel, including the newly integrated stereotypes, is illustrated in Figure 2.

Initially, we differentiate the CPS components, i.e., machines, physical devices, and software elements, since each of them plays a specific role in an attack scenario. Thus, we derived distinct stereotypes from SysML blocks: *Hosts*, *Programs*, and *Devices*. Custom attributes, such as the software's version number, can also be added to make the model more precise. These stereotypes are all extensions of *Asset*. These assets are represented as system components that hold a particular value or importance. Each have 2 attributes: (i) their level of importance, on a scale from 1 to 5, and (ii) the security property that must be guaranteed among the Confidentiality Integrity Availability (CIA) triad. The former gives insight into an asset's criticality, i.e., the level of impact if compromised. MulVAL uses the latter to generate the needed scenarios (for instance, all scenarios that lead to a confidentiality violation).

Another stereotype introduced is *Connection*, which represents directed network links between the components. It has an optional attribute to indicate the communication protocol (e.g., TCP/IP, Ethernet), and it can also point to traditional SysML ports to represent port numbers. If the connection protocol is not specified, it is assumed any protocol can be used. Connections are extensions of item flows, representing information exchanges between components.

Two stereotypes are used to represent individuals that interact with the CPS: the *Attacker* and the *User*. The latter has several dependencies with the system's blocks. Indeed, the stakeholder can have an account on a machine, which would correspond to a traditional user account with defined permissions. In the case of a device, we indicate if a stakeholder can physically access it and thus possibly tamper with it. With the stakeholder's attributes, one can provide more details about the user's profile. *Competent* is a boolean that indicates whether or not a user can be trivially compromised, for example, by phishing or brute-force attacks. The *Malicious* boolean is used to define an insider attacker, and the *Exposure* integer indicates the probability of the stakeholder being targeted. The attackers and stakeholders are derived from actors, similar to those seen in UML use-case diagrams to represent real users. The new stereotypes allow for their use cases to be similar to usual SysML design patterns, ultimately making it easier for system architects to adopt this approach. Furthermore, our approach minimizes the requirement for the inclusion of security-related information. Modelers are only required to provide basic input of security properties, such as CIA, and to assess the exposure levels of assets and stakeholders. Knowledge about vulnerabilities and

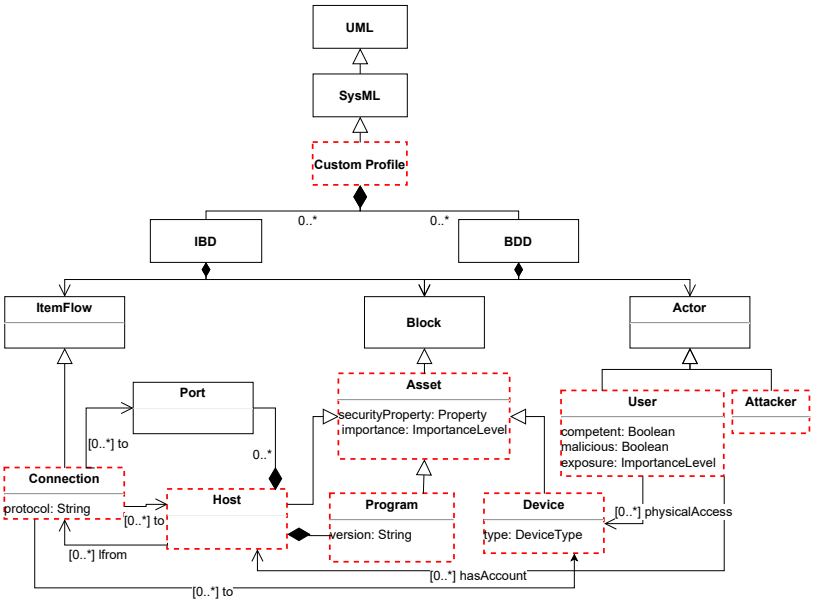

Figure 2: Extension of SysML's metamodel. The boxes in red are the new stereotypes, the ones in black are the default stereotypes.

attack patterns is already included in MulVAL, which lightens the burdens on system architects.

4.2 Extensions for MulVAL

MulVAL was initially designed to generate attack scenarios in legacy networked information systems. Only traditional elements, such as computers, firewalls, or routers, are identified by MulVAL. Thus, it is not suited to generate CPS-specific graphs. Several researchers improved MulVAL by introducing additional attacks and concepts [35, 46, 48, 52]. However, these extensions are considered use-case-specific. To the best of our knowledge, none of the existing contributions considered implementing extensions for CPS scenarios.

To fill this gap and to support the generation of CPS-specific scenarios, we implemented new MulVAL rules and predicates incorporating several common attacks [2, 9, 26, 29, 44] on CPSs. This paper focuses on the *OPC UA Flooding* [9] attack, a DoS attack that exploits a vulnerability in the OPC UA protocol. As explained by Cavalieri al. [9], the attacker sends a few initial messages, forcing the target to reply with multiple responses [9]. It is CPS-specific since OPC UA is commonly used to facilitate the information exchange between industrial processes [33]. To represent this in MulVAL, we implemented rules to check if an OPC UA connection between 2 hosts can be exploited by an attacker to perform DoS.

4.3 MTT Transformation Rules

Eclipse Acceleo [60] is a template-based code generator, with each template specifying the text that must be generated for each stereotype. We used Acceleo because it is an implementation of the Object Modeling Group MTT transformation standard [61], making it appropriate to parse any metamodel from the Eclipse Modeling Framework (EMF), including UML or SysML. Furthermore, it is open-source, easy to use, and well-documented, with frequent updates from the developers. Finally, it offers powerful support by offering an editor, a debugger, a profiler, and traceability between model and code [5].

Table 1 defines all the custom mappings between SysML stereotypes and MulVAL predicates. The model element's name and custom attributes can be included in the MulVAL predicates. Acceleo enables precise and intricate operations, such as identifying the host on which a program is running. It provides access to the model's hierarchy and retrieves all a given node's predecessors or siblings. Additionally, it is used to convert the SysML model into a Datalog input file for MulVAL.

Besides, a name-based identification algorithm (inspired from [34]) for the model's elements is implemented. Specifically, we used the Common Platform Enumeration (CPE) [11] name format to formally identify specific programs. When a model's element name contains or is contained by the CPE entry, the algorithm creates an association, and a new MulVAL predicate *isA(prog, cpe)* is generated. CPE is connected to the CVE database, which enables the identification of possible vulnerabilities in a software model. Hosts have no official naming convention; hence, we only provide a basic categorization with types such as *web server* or *database*. Additionally, the system architect does not need to enter information about the system's vulnerabilities. This step is performed automatically by the identification algorithm. It is implemented as Java functions, which can be called by Acceleo during the code generation. However, following a proper naming format is required.

5 CASE STUDY: HEATING VENTILATION AIR CONDITIONING SYSTEM

This section applies and evaluates the developed framework on a real-world CPS. The objective is to determine if the presented

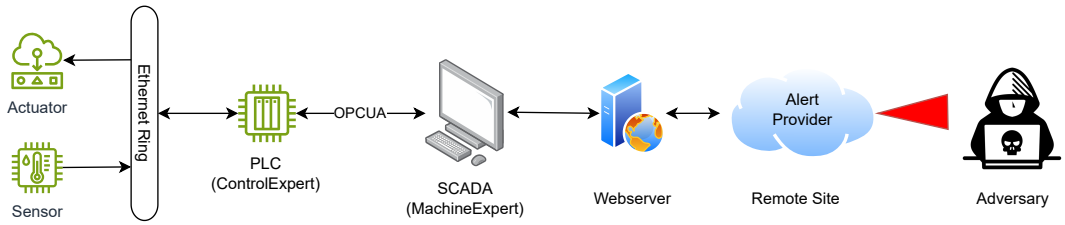

Figure 3: ICS architecture and components.

work can accurately model CPS characteristics and use them to automatically identify attack scenarios. The Heat Ventilation Air Conditioning (HVAC) system and its architecture are presented. Next, we model it using the developed SysML extension before generating the attack scenarios.

5.1 Heat Ventilation Air Conditioning Architecture

The case study is an Industrial Control System (ICS), more specifically, an autonomous HVAC system whose goal is to maintain a stable temperature. For simplicity and demonstration purposes, only a simplified version, depicted in Figure 3, is considered. The goal was to use a sufficiently generic model to prove that our approach could be applied to most systems.

The HVAC is composed of a Supervisory Control and Data Acquisition (SCADA), a control system for high-level supervision of machines and processes. The SCADA monitors the Programmable Logic Controller (PLC), which receives the temperature and humidity values from the sensor, performs computations, and sends commands to an actuator (a fan). The SCADA and PLC communicate using the OPC UA protocol, whereas the physical devices are connected via Ethernet. A web server is connected to the SCADA and serves as an entry point into the system. It can be accessed remotely via VPN, or directly through the employee workstations. Three pieces of software developed by Schneider Electric are used: EcoStruxure Machine Expert and Ecostruxure OPC UA Server Expert for the SCADA, EcoStruxure Control Expert for the PLC [1].

5.2 Scenario and Adversarial Model

The adversary assumed in this work is a cybercriminal who targets the HVAC infrastructure by performing DoS attacks and data thefts, seeking financial gains. We assume that the adversary is located outside the system and initiates the attack remotely. Furthermore, we assume the adversary has enough computational power to exhaust the most vulnerable machines, particularly the CPS components.

The adversary's main strategy is to target critical stakeholders (i.e., the actors with privileged access to the system) and use them as entry points to the system. Particularly, the adversary targets the *security alert provider* who is responsible for detecting abnormal events (e.g., sensors malfunction, abnormal temperatures, etc.) and reporting it to the HVAC's staff. The alert provider is located remotely and connects to the system via a secure Virtual Private Network (VPN) connection[2]. The adversary aims to compromise the credentials for the aforementioned VPN connection through brute-force cracking or using sniffer malware. This VPN connection will enable the adversary to access the internal network and perform the DoS attack.

5.3 Modeling the Case Study

We evaluate the proposed framework using the aforementioned case study. A model of the HVAC system must be first developed using the provided SysML extension. The model elements and diagrams are created using Eclipse Papyrus [59].

Figure 4 defines a high-level abstract block, named HVAC, as well as its basic components. Their interactions, represented by communication arrows and ports, correspond to the architecture discussed in Section 5.2. The SCADA monitors the PLC, which is connected to the sensor and actuator. The OPC UA protocol is given as an attribute of the `Connection` stereotype, and the OPC UA port is represented as a traditional SysML port. All model elements are not shown for readability, but by recursively nesting more blocks, it is possible to model the interactions with other sub-systems. For instance, the webserver is defined as an external interface.

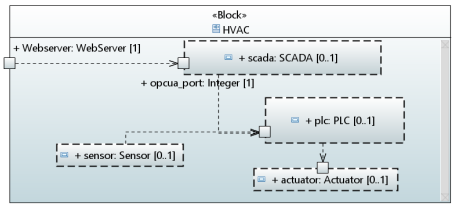

Figure 4: HVAC Internal Block Diagram.

5.4 Generated Attack Graphs

By applying Model-To-Text (MTT) transformations to the model, a MulVAL input is generated, enabling the automatic generation of attack graphs for the analyzed system. Various attack goals can be specified to create distinct scenarios. Figure 5 presents an attack graph in which the SCADA host is targeted. A *rectangle* node corresponds to a MulVAL predicate, while the *diamond-shaped* nodes represent the rules applied for each transition. The number on the node's left side is an arbitrary step number, and the one on the

[1]The described software can be found at https://www.se.com/uk/en/product-range/548-ecostruxure-control-expert-unity-pro and https://www.se.com/uk/en/product-range/2226-ecostruxure-machine-expert

[2]The alert provider sends a connection request to the web server through a VPN tunnel, allowing it to access the ICS, including the SCADA system.

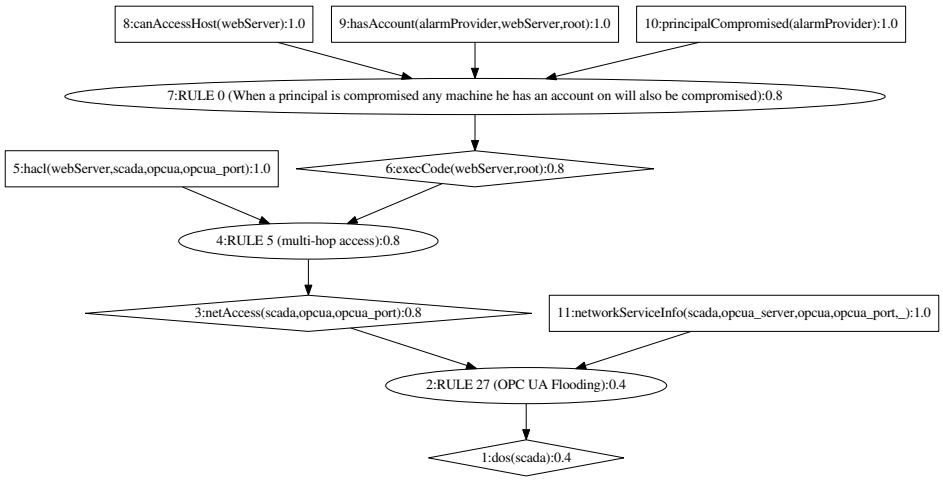

Figure 5: Generated attack graph for the OPCUA-Flooding attack.

right is the probability of reaching that node[3]. In the following, the nodes are referred to by their step number in between parentheses.

The web server is open to the Internet and runs a VPN service for remote connections. MulVAL concludes that the web server can be accessed using a log-in service (8). On the other hand, the alarm provider has an account used to connect to the CPS via the web server (9). The attacker compromises the password (10) and gains complete access to the web server (6).

The SCADA is directly connected to the server (5). The connection between these two hosts operates with the OPC UA protocol, and the OPC UA port is open. Furthermore, the SCADA hosts an OPC UA server program, which also communicates with the same protocol and port (11). Hence, the attacker can send specifically crafted OPC UA messages to overflow its resources, ultimately leading to DoS (1, 2).

Besides generating the attack graph, MulVAL performs a quantitative analysis by computing the probabilities. Theleaf nodes have a probability of 1 because they are assumed to be true information about the system. However, the rules are assigned a certain likelihood corresponding to the difficulty in exploiting the attack step. MulVAL then aggregates the probabilities by multiplying subsequent values.

6 RELATED WORK

6.1 Security Modeling

Various modeling languages were developed to represent security properties in engineering systems. Apvrille et al. [3] presented *SysML-Sec*, an extension of SysML that supports custom diagrams tailored for security. It introduces the `Security Requirement` stereotype and provides mechanisms to measure the impact of cryptographic countermeasures on the system's performance. SysML-Sec also introduces specific formalisms for attack graph representation [4]. However, it lacks several important security features, such as concepts for threats, vulnerabilities, and goals [20]. Additionally, El Hachem et al. [15] presented a SysML extension, named *SoSSecML*, to represent vulnerabilities and their pre and post-conditions. The extension includes modifications of the `Block` diagram with added elements such as `Goal`, `Organization`, `Environment`, `Operation` or `Threat`. Though it has been validated by real case studies, it does not include the elicitation of assets.

On the other hand, Messe et al. [34] developed an asset-based metamodel to identify the most critical components of the system. It relies on name-matching and type hierarchies to classify the components. It uses security databases, such as Common Weakness Enumeration (CWE) and Common Attack Pattern Enumeration and Classification (CAPEC), to automatically discover vulnerabilities and suggest appropriate countermeasures. The authors designed their approach to bridge the gap between system architects and security experts. However, it is not CPS-specific and does not provide model-checking for system engineering properties. Similarly, Vasilevskaya [64] introduced the Security for EmbeddED Systems (SEED) method to be accessible to non-security experts. The method is decomposed into three basic steps. First, a system model is analyzed to identify the parts that need security protection. Second, the security knowledge previously built by an expert is consulted to retrieve a set of relevant security properties. Finally, the selected set of security mechanisms is studied with respect to a potential resource overhead. Several limitations were pointed out in [20]: The security knowledge base needs to be created and maintained by an expert, and the expression of security properties is limited, as metrics were defined to measure losses for confidentiality and integrity only.

Outside of system modeling, other approaches were also developed by the security requirements engineering community. Secure Tropos [32] is an extension to Tropos methodology that enables developers to consider security issues throughout the development process of multi-agent systems. It utilizes several tools and techniques. The Secure Tropos Modeling Language is a graphical language used to model actors, goals, dependencies, and security constraints, helping visualize the relationships and security requirements in the system. Threat analysis techniques, such as STRIDE,

[3]Details on how the probabilities are computed can be found in [46]

are used to identify potential threats and mitigations. However, Secure Tropos has some limitations. Namely, modeling actors, goals, dependencies, and security constraints can be challenging, especially for large systems. Furthermore, it can be difficult to grasp and apply effectively by non-security experts. Finally, it is not a system modeling language, making it hardly scalable for complex CPSs.

Similarly, ARCHSEC [7] is an architectural security model aimed at enhancing the security of computer systems by integrating security measures directly into the architecture of hardware and software. By providing a graphical representation of the system, based on EMF, as well as data flow diagrams of the different services, the software system can be automatically analyzed w.r.t to vulnerabilities and threats. The analysis relies on graph queries to identify situations where the security requirements of an application are violated. Despite its numerous advantages, full automation of the approach is not possible, and the runtime of the graph-based flaw identification can grow quite large. Lastly, it is also software-centric, and the queries are not necessarily appropriate for CPS-specific vulnerabilities.

6.2 Attack Graph Generation

Several tools similar to MulVAL enable the generation of attack scenarios as attack graphs. Johnson et al. [25] presented Meta Attack Language (MAL), a meta-model for cyber threat analysis. It is inspired by CySeMol [45] and designed to allow any security analyst to easily create a DSL. MAL's compiler requires knowledge about the system's topology and possible attack steps. For example, abstract `Assets` (e.g., a computer), `Instances` (e.g., a MacBook), and `Associations` representing their connections can be defined. It requires introducing `Attack steps` and `Defenses` for each asset. It can be used for certain analyses such as attack graph generation or Time To Compromise (TTC) calculation. However, MAL introduces a new and unique meta-model with its own syntax [56], making it incompatible with already existing system engineering frameworks. Domain experts thus need experience in applying MAL to their fields.

Temple et al. [22] presented CyberSecurity Argument Graph Evaluation (Cybersage), an activity-centric tool that can represent a threat agent's workflow and compute the probability of an attacker reaching his goal. Its models are built around workflows that describe critical system functions or attacker actions. Those models are populated with information about system devices, configuration, and security controls. Attacker capabilities can be defined, such as skill, resources, accesses, and intentions. Additionally, it takes as input *mal-activity diagrams*, i.e., a modified version of UML's traditional sequence diagrams to represent malicious scenarios. However, these attack scenarios must be manually written, which makes the process impractical and error-prone. Besides, modeling very large systems has proven to be tedious [20], limiting the potential of Cybersage.

Wortman et al. [57] introduced Translation of AADL Model to Security Attack Tree (TAMSAT), a recent tool based on attack trees. Its main feature is to automatically and seamlessly convert an Architecture Analysis and Design Language (AADL) model [18] into an attack tree. The modeler defines *assets of importance*, which will serve as the root nodes of the generated attack tree. The AADL model is then parsed into a JSON dictionary, and vulnerabilities can be automatically identified by querying a locally stored CVE database. Finally, the generated format is compatible with the Security Model Adversarial Risk-based Tool (SMART) [54], which can perform financial risk analysis. A limitation is that the vulnerability database used is user-maintained (since it is locally stored). As such, actions must be taken to make sure it is not outdated.

Carter et al. [13] proposed another SysML-based methodology for CPS security modeling named Cybok. After the system and its low-level components have been modeled, they are encoded into an XML graph structure used for security analysis. Specifically, the graph nodes are used to request various security databases (including CVE, CWE, and CAPEC) and construct exploit chains, i.e. a succession of attack patterns and vulnerability exploits performed by an attacker. Cybok offers a seamless and intuitive approach for CPS attack scenario identification which is model-driven and accessible to non-security experts. However, it does not support important risk analysis features such as the probability assessment of attack scenarios.

Table 2: Features of different Model-based security tools

Tool	CPS-specific	Assets	Stakeholders	No Expertise Required	Model-checking
SysML-Sec [3]	✓	✗	✗	✗	✓
SEED [64]	✓	✓	✗	✓	✓
MoRiAML [62]	✗	✓	✓	✗	✓
MBCA [37]	✗	✓	✓	✗	✓
Asset-based [34]	✗	✓	✓	✓	✗
SoSSecML [15]	✗	✗	✓	✗	✓
SecureTropos [32]	✗	✓	✓	✗	✓
ARCHSEC [7]	✗	✗	✗	✓	✓
This work	✓	✓	✓	✓	✓

Table 3: Features of different Attack Graph generation tools

Tool	CPS-specific	Model input	Attacker Model	No Expertise Required	Probability Assessment
CySeMol [45]	✗	✗	✗	✓	✓
MAL [56]	✗	✗	✗	✓	✓
ADVISE [28]	✓	✗	✓	✗	✓
Cybersage [22]	✓	✗	✓	✗	✓
FAST-CPS [30]	✓	✓	✓	✓	✗
TAMSAT [57]	✓	✓	✗	✓	✗
ADTOOL [19]	✗	✗	✓	✗	✓
Cybok [13]	✓	✓	✓	✓	✗
This work	✓	✓	✓	✓	✓

6.3 Comparison with Existing Solutions

The proposed framework is compared to the previously described works. Table 2 and 3 summarize the different comparison criteria. Few existing security-oriented modeling languages were specifically designed for CPSs. Cyber and physical components are represented in the same way, and their differences are rarely considered. On the other hand, when a language is CPS-specific, it is often system-centric, making it hard to analyze non-functional requirements or external threats. The presented approach fills that gap by introducing specific stereotypes for component types, assets, and stakeholders. Besides, most modeling tools require the intervention of a security analyst. Our extension of SysML's metamodel only includes basic security concepts (CIA), making it usable by non-security experts.

Another challenge is the difficulty of accurately representing CPS-specific vulnerabilities, while various tools only consider traditional IT system attacks. Adding new predicates and rules in MulVAL makes it possible to generate more diverse and precise

scenarios, as we illustrate with the OPC UA Flooding attack. Furthermore, few model-based solutions are used in vulnerability or attack scenario identification. Our MTT transformations automatically convert a SysML model into a MulVAL input, thus bridging the gap between system architects and security experts.

7 DISCUSSION AND FUTURE WORK

We selected a HVAC system to validate the presented framework and generate attack scenarios. Although this system is not as complex as a real CPS system, we showed that it is possible to model its components and their interactions for two different attacks. This information was then used to automatically generate attack graphs. However, further validation on larger systems is required to verify if the generated attack graphs remain readable and if the modeling process requires extensive resources.

Furthermore, the *OPC UA Flooding* attack was added as MulVAL rules. This attack and its impact on CPSs have been extensively investigated [9], and integrating it in MulVAL enables the generation of more complex and specific attack graphs. This method can further be extended with additional rules to represent more diverse attacks. One could generalize the approach with limited efforts by choosing generic and well-crafted predicates.

The results of the performed evaluation confirm that the proposed framework can model complex and CPS-specific scenarios. This will ultimately help identify suitable countermeasures such as, in the described case study, port hardening to prevent unnecessary OPC UA connections and intensive physical access control. However, the identified attack scenarios were not tested on a real-life system. Additional experiments are needed to build a physical platform representing the case study and to carry out the attacks. For these experiments, an appropriate set of metrics should be defined to evaluate the approach's effectiveness. Such metrics could include, for instance, graph generation runtime, concept similarity with other ontologies, or comparison between the probability calculations from our framework and other methods.

The MTT transformations from a SysML model to MulVAL source code make the proposed approach seamless and accessible even to non-security experts. However, the CPS-specific vulnerabilities and attacks have to be added manually, possibly by a security expert, making it impractical. A solution would be to dynamically connect to security databases, such as CVE or CAPEC, to automatically generate the corresponding rules and predicates. However, such databases do not always follow a standard, machine-readable format. Future work on uniformizing security-related databases is thus necessary.

Finally, the new metamodel can be used to accurately design the system and its components. This system information is then used by MulVAL to identify threat origins and generate attack scenarios, along with a probabilistic assessment. However, risk treatment and countermeasure selection are currently not supported. Mechanisms to facilitate the graph's interpretation are also required. This could be solved by implementing additional MulVAL rules that integrate countermeasures or by connecting to databases that consider mitigations, such as CAPEC. The countermeasures could then be translated back into the SysML input, with dedicated stereotypes, to update the model before running a new simulation, thus closing the feedback loop.

8 CONCLUSION

CPSs are complex and vulnerable systems, and their security must be considered from an architectural perspective. Despite the potential of models, few existing solutions apply MBSE practices to represent precise and CPS-specific scenarios. We presented a new model-based framework to automatically identify and assess attack scenarios in CPSs to fill this gap. Our approach combines two specific tools: SysML and MulVAL. SysML was extended with CPS-specific and security-related stereotypes to better integrate security elements directly into the CPS model. Additionally, new MulVAL predicates and rules were developed to generate common attacks in industrial systems, such as OPC UA Flooding. Finally, a set of MTT transformations enable the automatic generation of a MulVAL attack graph from a SysML model.

The presented framework was evaluated on a real-world inspired smart building. It was possible to identify the vulnerabilities and attack patterns leading to a system intrusion and a denial of service. The MulVAL tool also computes success probabilities, indicating the difficulty of carrying out these attacks. Such calculations can be used to identify the most vulnerable components and prioritize defenses.

Future work includes validation on a larger case study and mechanisms to query security databases. Ultimately, the framework should be able to automatically identify the components' types and vulnerabilities, and suggest appropriate countermeasures with only the information provided by the model. These steps are necessary to guarantee the development of safer and more secure CPSs.

ACKNOWLEDGMENTS

This work is funded by the French Defense Innovation Agency (AID) under contract n° 2021650010 (CERES).

REFERENCES

[1] Omar Doukari, Boubakar Seck, David Greenwood, Haibo Feng, and Mohamad Kassem. [n. d.]. Towards an Interoperable Approach for Modelling and Managing Smart Building Data: The Case of the CESI Smart Building Demonstrator. 12, 3 ([n. d.]).

[2] Amin et al. 2013. Cyber Security of Water SCADA Systems—Part I: Analysis and Experimentation of Stealthy Deception Attacks. *IEEE Transactions on Control Systems Technology* 21, 5 (2013), 1963–1970.

[3] Apvrille et al. 2013. SYSML-SEC: A SYSML ENVIRONMENT FOR THE DESIGN AND DEVELOPMENT OF SECURE EMBEDDED SYSTEMS. Yokohama, Japan.

[4] Apvrille et al. 2015. SysML-Sec Attack Graphs: Compact Representations for Complex Attacks. In *GraMSec@CSF*.

[5] Brambilla et al. 2012. *Model-Driven Software Engineering in Practice*. Vol. 1.

[6] Balaji et al. 2015. Models, abstractions, and architectures: The missing links in cyber-physical systems. In *52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6.

[7] Berger et al. 2019. The Architectural Security Tool Suite — ARCHSEC. In *2019 19th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. 250–255. https://doi.org/10.1109/SCAM.2019.00035

[8] Ceri et al. 1989. What you always wanted to know about Datalog(and never dared to ask). *IEEE transactions on knowledge and data engineering* 1, 1 (1989), 146–166.

[9] Cavalieri et al. 2010. Evaluating impact of security on OPC UA performance. In *3rd International Conference on Human System Interaction*. 687–694.

[10] Cárdenas et al. 2011. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security* (Hong Kong, China) *(ASIACCS '11)*. New York, NY, USA, 355–366.

[11] Cheikes et al. 2011. *Common platform enumeration: Naming specification version 2.3.*

[12] Chen et al. 2017. Cyber-physical system enabled nearby traffic flow modelling for autonomous vehicles. In *IEEE 36th International Performance Computing and Communications Conference (IPCCC)*. 1–6.

[13] Carter et al. 2019. *Cyber-Physical Systems Modeling for Security Using SysML*. 665–675. https://doi.org/10.1007/978-3-030-00114-8_53

[14] Dibaji et al. 2019. A systems and control perspective of CPS security. *Annual Reviews in Control* 47 (2019), 394–411.

[15] El Hachem et al. 2016. Model Driven Software Security Architecture of Systems-of-Systems. In *23rd Asia-Pacific Software Engineering Conference (APSEC)*. 89–96.

[16] El Hachem et al. 2019. Using Bayesian Networks for a Cyberattacks Propagation Analysis in Systems-of-Systems. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*. 363–370.

[17] El Hachem et al. 2020. Modeling, analyzing and predicting security cascading attacks in smart buildings systems-of-systems. *Journal of Systems and Software* 162 (2020), 110484.

[18] Feiler et al. 2005. An Overview of the SAE Architecture Analysis and Design Language (AADL) Standard: A Basis for Model-Based Architecture-Driven Embedded Systems Engineering. In *Architecture Description Languages*, Pierre Dissaux, Mamoun Filali-Amine, Pierre Michel, and François Vernadat (Eds.). Boston, MA, 3–15.

[19] Fila et al. 2019. Attack–Defense Trees for Abusing Optical Power Meters: A Case Study and the OSEAD Tool Experience Report. In *Graphical Models for Security*, Massimiliano Albanese, Ross Horne, and Christian W. Probst (Eds.). Cham, 95–125.

[20] Geismann et al. 2020. A systematic literature review of model-driven security engineering for cyber-physical systems. *Journal of Systems and Software* 169, 110697 (2020).

[21] Gartenberg et al. 2021. *Security startup Verkada hack exposes 150,000 security cameras in Tesla factories, jails, and more.* https://www.theverge.com/2021/3/9/22322122/verkada-hack-150000-security-cameras-tesla-factory-cloudflare-jails-hospitals

[22] G. Temple et al. 2022. CyberSAGE: The cyber security argument graph evaluation tool. *Empirical Software Engineering volume* 28, 18 (2022).

[23] Holloway et al. 2015. *Stuxnet Worm Attack on Iranian Nuclear Facilities.* http://large.stanford.edu/courses/2015/ph241/holloway1/

[24] Huang et al. 2018. Towards Modeling Cyber-Physical Systems with SysML/MARTE/pCCSL, Vol. 1. Tokyo, Japan, 264–269.

[25] Johnson et al. 2018. A Meta Language for Threat Modeling and Attack Simulations. In *Proceedings of the 13th International Conference on Availability, Reliability and Security* (Hamburg, Germany) *(ARES '18)*. Association for Computing Machinery, New York, NY, USA, Article 38, 8 pages.

[26] Long et al. 2005. Denial of service attacks on network-based control systems: impact and mitigation. *IEEE Transactions on Industrial Informatics* 1, 2 (2005), 85–96.

[27] Lee et al. 2008. Cyber Physical Systems: Design Challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. 363–369.

[28] LeMay et al. 2011. Model-based Security Metrics Using ADversary VIew Security Evaluation (ADVISE). In *Eighth International Conference on Quantitative Evaluation of SysTems*. 191–200.

[29] Lee et al. 2014. A Passivity Framework for Modeling and Mitigating Wormhole Attacks on Networked Control Systems. *IEEE Trans. Automat. Control* 59, 12 (2014), 3224–3237.

[30] Lemaire et al. 2017. A logic-based framework for the security analysis of Industrial Control Systems. *Automatic Control and Computer Sciences* 51, 2 (March 2017), 114–123.

[31] Mell et al. 2006. Common Vulnerability Scoring System. *IEEE Security & Privacy* 4, 6 (2006), 85–89.

[32] Mouratidis et al. 2007. Secure Tropos: A Security-Oriented Extension of the Tropos methodology. *International Journal of Software Engineering and Knowledge Engineering* 17 (04 2007). https://doi.org/10.1142/S0218194007003240

[33] Mahnke et al. 2009. *OPC unified architecture.*

[34] Messe et al. 2020. An Asset-Based Assistance for Secure by Design. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*. 178–187.

[35] McCormack et al. 2020. Security Analysis of Networked 3D Printers. In *2020 IEEE Security and Privacy Workshops (SPW)*. 118–125.

[36] Nguyen et al. 2017. Model-based security engineering for cyber-physical systems: A systematic mapping study. *Information and Software Technology* 83 (2017), 116–135.

[37] Naouar et al. 2021. Towards the Integration of Cybersecurity Risk Assessment into Model-based Requirements Engineering. Notre Dame, IN, USA, 334–344.

[38] Ou et al. 2005. MulVAL: A logic-based network security analyzer, Vol. 14. Baltimore, MD, 8–8.

[39] Oldevik et al. 2005. Toward standardised model to text transformations. In *European Conference on Model Driven Architecture-Foundations and Applications*. Springer, 239–253.

[40] Phillips et al. 1998. A Graph-Based System for Network-Vulnerability Analysis. In *Proceedings of the 1998 Workshop on New Security Paradigms* (Charlottesville, Virginia, USA) *(NSPW '98)*. New York, NY, USA, 71–79. https://doi.org/10.1145/310889.310919

[41] Ramos et al. 2012. Model-Based Systems Engineering: An Emerging Approach for Modern Systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 1 (2012), 101–111. https://doi.org/10.1109/TSMCC.2011.2106495

[42] Sendall et al. 2003. Model transformation: the heart and soul of model-driven software development. *IEEE Software* 20, 5 (2003), 42–45.

[43] Sandberg et al. 2010. On Security Indices for State Estimators in Power Networks.

[44] Smith et al. 2011. A Decoupled Feedback Structure for Covertly Appropriating Networked Control Systems. *IFAC Proceedings Volumes (IFAC-PapersOnline)* 18 (Aug. 2011). ISBN: 9783902661937.

[45] Sommestad et al. 2013. The Cyber Security Modeling Language: A Tool for Assessing the Vulnerability of Enterprise System Architectures. *IEEE Systems Journal* 7, 3 (2013), 363–373.

[46] Sembiring et al. 2015. Network Security Risk Analysis using Improved MulVAL Bayesian Attack Graphs. *International Journal on Electrical Engineering and Informatics* 7 (12 2015), 735–753.

[47] Shevchenko et al. 2020. An Introduction to Model-Based Systems Engineering (MBSE). Carnegie Mellon University, Software Engineering Institute's Insights (blog).

[48] Stan et al. 2022. Extending Attack Graphs to Represent Cyber-Attacks in Communication Protocols and Modern IT Networks. *IEEE Transactions on Dependable and Secure Computing* 19, 3 (2022), 1936–1954.

[49] Sultan et al. 2022. Safety, Security and Performance Assessment of Security Countermeasures with SysML-Sec. In *10th International Conference on Model-Driven Engineering and Software Development*. Vienna, Austria.

[50] Sultan et al. 2022. Safety, Security and Performance Assessment of Security Countermeasures with SysML-Sec. Vienna, Austria.

[51] Tuma et al. 2020. Automating the early detection of security design flaws. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems* (Virtual Event, Canada) *(MODELS '20)*. Association for Computing Machinery, New York, NY, USA, 332–342.

[52] Tayouri et al. 2023. A Survey of MulVAL Extensions and Their Attack Scenarios Coverage. *IEEE Access* 11 (2023), 27974–27991. https://doi.org/10.1109/ACCESS.2023.3257721

[53] Victorino et al. 2022. Cyber–physical system modeling with Modelica using message passing communication. *Simulation Modelling Practice and Theory* 117 (2022), 102501.

[54] Wortman et al. 2020. SMART: security model adversarial risk-based tool for systems security design evaluation. *Journal of Cybersecurity* 6, 1 (02 2020), tyaa003.

[55] Wolny et al. 2020. Thirteen years of SysML: a systematic mapping study. *Software and Systems Modeling* 19 (2020), 111–169.

[56] Wideł et al. 2023. The meta attack language - a formal description. *Computers and Security* 130 (2023), 103284.

[57] Wortman et al. 2023. Translation of AADL model to security attack tree (TAMSAT) to SMART evaluation of monetary security risk. *Information Security Journal: A Global Perspective* 32, 4 (2023), 297–313.

[58] Yaacoub et al. 2020. Cyber-physical systems security: Limitations, issues and future trends. *Microprocessors and Microsystems* 77 (2020), 103201.

[59] Eclipse Foundation. 2019. *Eclipse Papyrus ™ Modeling environment.* https://eclipse.dev/papyrus/

[60] Eclipse Foundation. 2023. *Eclipse Acceleo.* https://projects.eclipse.org/projects/modeling.m2t.acceleo

[61] Object Management Group. 2008. *ABOUT THE MOF MODEL TO TEXT TRANSFORMATION LANGUAGE SPECIFICATION VERSION 1.0.* https://www.omg.org/spec/MOFM2T/1.0/

[62] Douraïd Naouar. 2022. *MoRiA A model-based method for cybersecurity risk analysis : application to a complex naval defense system.* doctoral thesis. Ecole nationale supérieure Mines-Télécom Atlantique.

[63] OMG. 2023. *ABOUT THE OMG SYSTEM MODELING LANGUAGE SPECIFICATION VERSION 2.0 BETA.* https://www.omg.org/spec/SysML/2.0/Beta1/About-SysML

[64] Maria Vasilevskaya. 2015. *Security in Embedded Systems: A Model-Based Approach with Risk Metrics.* Ph. D. Dissertation. Linköping University, Department of Computer and Information Science, Faculty of Science and Engineering.