A hybrid threat model for smart systems

Fulvio Valenza*, Erisa Karafili*†, Rodrigo Vieira Steiner, Emil C. Lupu

Abstract—Cyber-physical systems and their smart components have a pervasive presence in all our daily activities. Unfortunately, identifying the potential threats and issues in these systems and selecting enough protection is challenging given that such environments combine human, physical and cyber aspects to the system design and implementation. Current threat models and analysis do not take into consideration all three aspects of the analyzed system, how they can introduce new vulnerabilities or protection measures to each other. In this work, we introduce a novel threat model for cyber-physical systems that combine the cyber, physical, and human aspects. Our model represents the system's components relations and security properties by taking into consideration these three aspects. We propose together with the threat model also a threat analysis method that allows understanding the security state of the system's components. The threat model and the threat analysis have been implemented into an automatic tool, called TAMELESS, that automatically analyzes the threats of the system, verifies its security properties, and generates a graphical representation, useful for security administrators to identify the proper prevention/mitigation solutions. We show and prove the use of our threat model and analysis with three cases studies from different sectors.

Index Terms—Threat Analysis, Cybersecurity modelling, Threat model, Cyber-Physical systems

1 Introduction

Nowadays, novel paradigms and technologies such as the Internet of Things (IoT) and Edge Computing, pervade all aspects of the physical world [1]. Cyber-physical systems are ubiquitously present in our everyday life, their devices and environments are "intelligent", interconnected, dynamic, and flexible. We can summarize simply as "*smart*".

The opportunities derived by these smart systems come with their own security challenges. Determining the potential *threats* in these smart systems and providing an adequate amount of protection is more challenging than in traditional computer systems for a variety of reasons that include low computational power, inadequate software quality but also more importantly the fact that such environments combine human, physical and digital (cyber) aspects to the system design and implementation. In these scenarios, the attack surface is broader and extends beyond the realm of the "cyber" domain to the physical and human aspects of the system.

As shown in [2], cyber and physical attacks evolve as fast as the deployment of smart systems and are outpacing efforts to stop them. Such systems are deployed in a physical environment, and in addition to being reachable through their interconnections, they are also physically available to an attacker that can, for example, connect to the hardware interfaces of the device. Finally, these devices interact in several ways with human users. Thus, systems comprising of IoT devices are vulnerable to attacks that exploit their physical, human, and cyber vulnerabilities, as well as to attacks that combine these exploits in any order. For instance, unauthorised physical access to a wind turbine allows an attacker to generate a cyber-attack on the wind farm control network [3]. Whilst a number of methodologies perform threat modelling for cyber-attacks [4], [5], [6], [7], [8], [9], they usually do not take into account attacks on the physical or human aspects of the system and cannot represent the propagation of attacks across the cyber-physical, human-physical, or human-cyber interfaces.

Current threat models and analysis mainly take into consideration only one component between the human, cyber or physical aspects of the analysed system (e.g., [4], [5], [6], [7]), or, occasionally, two of them e.g., the human and cyber aspects in [8], [9]. There is a lack of analyses that take into consideration all three components and how they can introduce new vulnerabilities or protection measures to each other, as well as their interactions. Furthermore, there is a need to develop efficient preventive and mitigative actions that use holistic analysis of the threats and for the construction of threat models for systems that include all three aspects, human, physical, and cyber.

This work aims to take the first steps towards a methodology for performing exhaustive threat analysis, taking into consideration cyber, physical, and human aspects, *as well as the relations between them*. Specifically, we will use the term *"hybrid systems"* to emphasise that we refer to systems in which we take into account their human, physical and cyber perspectives and *"hybrid attacks"* to refer to multi-step attacks that can combine attack steps exploiting human, physical, or cyber vulnerabilities in combination. The security analysis of such systems cannot be done from one perspective alone but must consider the physical context of the system, its cyber resources and connections, and the humans that use or operate it [10].

Threat models for hybrid systems must be able to represent attack scenarios that exploit the interplay between the

- **These authors contributed equally to this work. † Corresponding author. F. Valenza is with the Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Turin, Italy (e-mail:fulvio.valenza@polito.it). E. Karafili is with the School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, United Kingdom (e-mail:e.karafili@soton.ac.uk) R.V. Steiner and E. Lupu are with the Department of Computing, Imperial College London, London SW7 2AZ, United Kingdom. (email:r.vieira-steiner14@imperial.ac.uk, e.c.lupu@imperial.ac.uk).*

human, physical, and cyber aspects of the system. For example, attacks such as those conducted on ATM machines, e.g., jackpot, skimmer, shimmer, cash-out attacks [11], [12], [13], [14]. The "jackpot" attack, in particular, exploits the vulnerability of the physical components and by cutting a small hole next to the PIN pad an attacker can insert a cable connected to a laptop and command the ATM to dispense all the money. Other attacks exploit human vulnerabilities, e.g., by bribing employees to sell customer data or providing sufficient information to conduct successful phishing attacks on the clients.

Hybrid attacks are not specific to the financial sector but also affect other sectors. For example, in the energy sector, researchers have shown that with little effort an attacker can compromise a whole windmill farm network starting by simply breaking a physical lock [3], [15]. Further steps of this attack exploit cyber vulnerabilities such as easy access, lack of encryption in communications, poor default passwords, and insecure remote management interfaces.

The physical security of a system, its cybersecurity, and human security are traditionally dealt separately and by different people/teams inside an organization. In hybrid systems the cyber, physical, and human aspects can all be leveraged in combination as part of the same attack. They typically also complement each other and must be leveraged together to mitigate and respond to threats. For example, human or digital surveillance can be used to monitor a physical space.

Yet at the moment, there is no notation and no framework to be able to represent threats that propagate across the physical, human, and digital aspects of a system, or how to use them in combination. It is not sufficient for such a framework to just combine the different aspects/components of the system, it must also represent the relationships between them, their inter-dependencies and the compositional nature of systems.

The main aim of our work is to design a threat modelling approach that takes into account the human, cyber, and physical aspects of hybrid systems, their interdependencies, and that can analyse their weaknesses and help reason about remediation. We propose the first threat model that permits to describe and derive the security state of smart systems' entities, their relations, and properties. The model permits to describe and analyse the system as part of its physical and human environment rather than in isolation from it. The main contributions of this work are as follows:

- We propose a novel *hybrid threat model* that can represent the relations and security properties of the system's components by taking into consideration their cyber, physical, and human aspects.
- We introduce a *threat analysis* method equipped with a set of derivation rules that permit to understand the properties of the system's components and the overall security state.
- We provide a *tool*[1] that represents the security properties and relations of the system's components. This tool automatically analyzes the threats of the system and can automatically generate a graphical representation of it.

1. https://github.com/FulvioValenza/TAMELESS

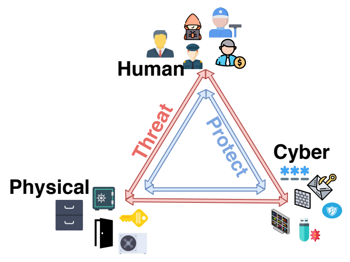

Fig. 1. The main components of a hybrid system threat model

The paper is structured as follows. In Section 2, we introduce the main motivation for this work by means of a case study from smart buildings and provide an overview of the proposed solution together with some practical considerations. In Section 3, we introduce our novel threat model with its components, relations, and properties. In Section 4, we present the rules that permit to gather the facts behind the attack graphs, evaluated during the threat analysis of the system. In Section 5, we present the application of our threat model in three different case studies taken from the smart building scenario and a Wind Farm scenario. Finally, in Section 6, we discuss the related work, and in Section 7 we draw the main conclusions and discuss directions for future work.

2 Problem Statement and Approach

Our main goal is to identify and construct a threat model for hybrid threats to cyber-physical systems that combine cyber, physical, and human aspects and can be attacked on each one of these aspects or in combination.

As previously introduced, a *hybrid system* is considered a system that is composed of one or various components of different aspects/nature, e.g., a component is a physical one, while another is human and other components are cyber. Some components can have more than one nature, and we will explicitly identify their nature during the threat analysis. Therefore, to represent hybrid threats it is necessary to be able to represent the relations between the different aspects of the system: the physical, cyber, and human. Of particular importance are how each one of these aspects can introduce *a threat* to the other, i.e., does compromising one aspect compromise the other? and how they can *protect* each other, i.e., can one aspect help to protect the other?. Broadly, these relations can be represented, as shown in Figure 1, where different components of various nature can protect or introduce/spread vulnerabilities to other components of the system, also of different nature.

A vulnerable physical aspect of a component can threaten the security of a cyber component of the hybrid system. For example, having physical access to a room, where it is possible to connect (unauthenticated) to the wired network, would enable the attacker to compromise the software/network components of the system. Similarly, having physical access to a sensor can enable an attacker to perturb what the sensor is measuring. Furthermore, compromising the human aspects of a system can also compromise its security, e.g., deceiving the user to reveal access details or stealing access keys (both are well known examples). With the increased use of actuators that can affect the physical aspects, compromising the cyber aspects of a system also enables a physical compromise. An obvious example is to compromise a digital lock, or the control system, to open the door to a protected area of a building.

These components can also protect each other. A human can inspect and monitor the physical security, e.g., of an area, a smart building, or a device. A physically secure enclosure can protect both human and digital components - this is why computers or servers are often stored in a secure location. Finally, we increasingly use the digital (cyber) capability to monitor the security of both physical spaces, and the behaviour of humans, e.g., to protect from insider threats.

Our model allows to represent these relations between the human, physical and digital components of the system and to reason about them. In particular, it allows to reason about the propagation of attacks across the different parts of the system and to formulate protective and preventive measures to increase robustness to attacks. Protecting a system can thus be formulated as the combination of human, cyber, and physical interventions whose relative balance depends on the system and the context of use.

We illustrate our model and our approach through an example that we will use throughout the remainder of this paper. We consider the case of a smart building that may be subject to attacks combining physical attack steps (such as breaking a door, a window, or picking a lock), cyber-attack steps (compromising the network, digital locks, or the Building Management System) and human attack steps (such as stealing a pass, or losing a key). Our smart building comprises three different locations: the hall, the office area and the rooftop, as shown in Figure 2. The building is equipped with IoT devices and sensors, like smart cameras, temperature sensors etc.. There is a monitoring system on the hall (entrance) where the camera images are controlled by a human (guard) to prevent unauthorised access to the building. Moreover, part of the rooms on the office floor have locks on their doors to prevent unauthorised access. Although this example provides only a small scale illustration of our model and method, scalability aspects to larger systems are considered in Section 2.2 and 6.

2.1 Scenario

To illustrate our approach we consider a scenario shown in Figure 3, which is part of the office floor shown in Figure 2. We are concerned, in particular, about *unauthorized access* to a safe box, denoted by sbox and located in a particular room on the office floor. The safe box is used by the employees to store cash, company check books and confidential documents. The safe box is *physically protected* through a digital lock requiring an access code (Password). This Password is set by the employee physically working in that room. Physical security measures protect the room where the safe box is located: the room can only be accessed through a locked door. The key to open the door (is based on an RFID card) and provided solely to authorised employees (including the employee working in that room). Thus, to access the safe box an *attacker* needs to both access the room and to know the Password required to open the safe. For the purpose of keeping the example simple, we assume that a physical attack on the safe box is not practical.

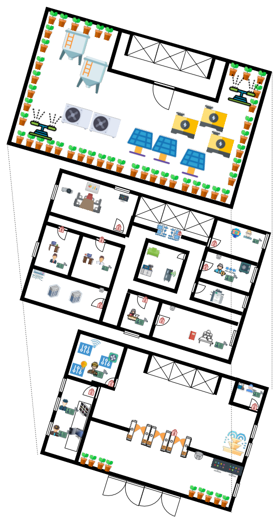
Fig. 2. Example of the smart building used in our case study

For the sake of the example let us assume that the password for the safebox is stored in the employee on duty machine. Therefore, access to the employee's computer (e.g., through phishing) will eventually lead to finding the password. The RFID cards required to open the door can be read and cloned at a moderate distance (e.g., several feet away), leading to the possibility for an attacker to create a duplicate key.

Our model permits us to represent the relations between the human, physical, and cyber aspects of this system. How human vulnerabilities (e.g., phishing or bribery) can lead to threats to the cyber or physical aspects of the system. How cyber vulnerabilities can lead to threats to the physical environment. But also how the different elements of the system relate to each other, for example containment relations (e.g., the safe is in the room) and how they relate to each other (e.g., access to the room is via a digital lock). We can also represent how the different aspects of the system can help to protect each other (e.g., a physical space can be monitored by sensors or by security patrols).

In the introduced scenario, access to the content of the safe requires: exploiting human vulnerabilities (e.g., with a phishing attack) to lead to a compromise of the digital environment, which enables exploiting further digital vulnerabilities to access the employee's machine to obtain the password. An attack would also require cloning the RFID card and using this to access the room where the safe is located, before using the password to open the safe. The scenario was only conceived to illustrate how the different relations can be represented in our model and how the model can be analysed for threats across the different aspects of the system. However, many modern systems exhibit similar characteristics.

To create a comprehensive threat model, our approach allows to represent the properties of the system's components. This includes *security* properties such as vulnerability or detectability as well as *functional* properties such as

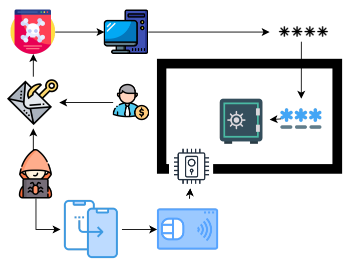

Fig. 3. A threat model for the unauthorised safe box access

device malfunction or recoverability of a system component.

2.2 TAMELESS

We propose a threat analysis model that can derive the current security state of the system and its components, given as input information about the system's components and their security properties. We have implemented the model and threat analysis as an automated tool called Threat & Attack ModEL Smart System (TAMELESS) that is made available with this paper[2], and which relies on an XSB Prolog interpreter[3].

The input for TAMELESS comprises of the specification of the analysed system (i.e., the system's components and threats), the relations between the components, as well as between the components and the threats and the various (security) assumptions. The user (e.g., a security administrator) can then perform the threat analysis by querying TAMELESS to display all the security properties of the system's components. This allows to determine whether the system's components expected states differ from the derived ones (i.e., a *consistency* check). This aspect is important because each smart system has distinct features and is different from others. In some systems, it is fundamental that each element is not compromised, while in others that each element works properly, or is important to know if it is possible to detect an attack, restore or fix the system's status. Our solution allows the users to have a comprehensive view of the security of their system and its risks across the human, physical, and cyber aspects. The complexity of the system, the propagation of threats across different relationships between components, and the number of different possibilities and their combinations make it impossible to conduct this analysis manually for any system of a reasonable size (i.e., beyond toy examples).

TAMELESS provides a graphical representation of the attack propagation that allows the user to add *protection* and *monitor* components in order to change the undersigned security proprieties. The user can run the tool again together with the updated information (i.e., the derived initial model plus the newly added protection and monitor components), and TAMELESS shows the derived threat model. This iteration can be repeated until the user is satisfied with the threat model of the constructed system. With this iterative process, the administrator is able to determine the effectiveness of the selected protective/preventive measures.

2. https://github.com/FulvioValenza/TAMELESS
3. A dialect of Prolog developed by the Stony Brook University [16] (http://xsb.sourceforge.net/index.html).

Moreover, TAMELESS can also help the system administrator to consider also the economical aspects of ensuring the security of the system. For example, if the administrator defines a cost associated to each protection solution, then they can choose, in a simple way, which solutions are more "secure" and efficient with respect to others, by taking into consideration also the costs related to such solutions. The administrator can change a protection/monitor component (human, cyber, or physical) with another one that costs less but that maintains the same security properties for the system. For example, the administrator can now choose between a security guard or a security camera for the surveillance of a physical space.

One of the main current limitations of our threat analysis is that it is a static analysis: in case the relations or properties of the system change, then the analysis needs to run again with the updated system. Another limitation that we plan to fill in future works, deals with the relations and derivation rules. For this version of the threat model, we assumed the relations and derivation rules to always be true, but in the real world, this is not always the case, as a relation or a protection might hold with a certain probability or certainty. Furthermore, in the current threat model, we do not consider the time. The temporal aspect is crucial in modelling the *resilience* aspects of each entity. Indeed each entity has different levels of resilience from different threats and each attack path requires a different amount of time to be performed.

In summary, the threat model we constructed is generic and can be used to analyse the threats of various hybrid systems, e.g., smart buildings, industrial control systems, smart windmill farms, ATM machines, credit cards, smart homes. Once the system components are provided to TAMELESS together with their properties and relations, it can automatically analyse them, and through a graphical representation it provides the possible threats of the analysed system. Furthermore, our threat model helps the system administrator to identify the most appropriate countermeasures for the found threats, as TAMELESS provides a set of possible entities that can be used to prevent or mitigate the threat. The provided entities could be entities that: protect, monitor, detect, restore, repair, or replicate.

3 HYBRID THREAT MODEL

In this section, we introduce the proposed threat model that comprises the security properties of the entities of a hybrid system, the relations that the system's components have with various threats, and the different relations between the system components themselves.

By representing the system's components, their nature, their security properties, and the various relations the threat model permits to identify new vulnerabilities and reason about the necessary mitigation actions including identifying which entities can be used to protect others or what relations can be created or removed to protect the system. Briefly, the security properties represent how the security state of the components can change[4]. While, the relations represent how the system components and aspects are related to each

4. We use the term property instead of state, as it represents the ability to change state, not the state itself.

other given the components' nature (human, physical, and cyber). It is important to distinguish the types of relations as they represent how a threat can be spread or stopped in the system.

3.1 System's Components

Our *threat model* comprises of various entities that can be human, cyber, or physical.

Definition 1. An *entity* is a system or a component of a system that can be of a cyber, physical, or human nature[5]. We denote with $\mathcal{E} = \{A, B, C, \cdots\}$ the set of all entities of our system.

A threat model represents and distinguishes between the various threats defined as below.

Definition 2. A *threat* is one or a sequence of actions that directly or indirectly changes a property that can alter the security state of an entity. We denote with $\mathcal{T} = \{T, T_1, T_2, \cdots\}$ the set of all threats.

We denote with $\mathcal{F}$ the set of all formulas in our system. The set comprises of all formulas that can be expressed by the grammar below:

$$\begin{aligned}\phi ::= \quad & true \mid false \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \\ & p(A) \mid \neg p(A) \mid r(A, A) \mid r'(A, A, A)\end{aligned}$$

where $\phi \in \mathcal{F}$, $A \in \mathcal{E}$, $p \in \mathcal{P}$ and $r, r' \in \mathcal{R}$. $\mathcal{P}$ denotes the set of all security properties and $\mathcal{R}$ denotes the set of all relations. The connectors $\wedge, \vee, \rightarrow, (,)$ and $\neg$ are the standard ones. This language permits to define a set of *rules* to derive the different properties of the considered entities.

3.2 Security Properties of the Entities

The properties of the entities can either explicitly be assumed to be true or can be derived to be true by applying the rules. We distinguish between *basic* properties and *auxiliary* properties. The former describe primary security knowledge about the system's components, while the latter describe the states of an entity, when it is already compromised or malfunctioning due to a threat.

3.2.1 Basic Properties

We denote with $\mathcal{P}_B$ the set of all *basic properties*, where:

$$\mathcal{P}_B = \{\mathrm{Comp}, \mathrm{Malfun}, \mathrm{Vul}\}.$$

Compromised $\mathrm{Comp}(A, T)$: entity A has been compromised by *threat* T;

Malfunctioned $\mathrm{Malfun}(A)$: entity A is malfunctioning and one or more of its functionalities are not working as expected, e.g., a server expected to be running is down;

Vulnerable $\mathrm{Vul}(A, T)$: A has a known vulnerability which makes it vulnerable to *threat* T.

For simplicity we write $\mathrm{Comp}(A)$ when there exists at least one threat T for which A is compromised ($\mathrm{Comp}(A, T)$), and $\neg\mathrm{Comp}(A)$ when no such threat $T \in \mathcal{T}$ exists.

5. In some cases, the same entity might have more than one nature. We will distinguish the various natures of the entity and deal with them separately.

3.2.2 Auxiliary Properties

We denote by $\mathcal{P}_A$ the set of all auxiliary properties of our system.

$$\mathcal{P}_A = \{\mathrm{Det}, \mathrm{Rest}, \mathrm{Fix}\}.$$

Detected $\mathrm{Det}(A, T)$: describes that it has been *detected* that entity A has been compromised by threat T, e.g., a physical or digital unauthorised access has been detected.

Restored $\mathrm{Rest}(A)$: describes that $\mathrm{Control}$ over A is *restored*, usually after A has been compromised, e.g., a room has been secured again, or the system has been cleansed, patched and restored.

Fixed $\mathrm{Fix}(A)$: describes that the *functionality* of A is *repaired*, usually after A malfunctioned (Malfun), e.g., the lock of a door is repaired, or the antivirus has been updated.

We distinguish between the Rest and Fix properties to differentiate between the security and functional properties of a system. Rest is applied after an entity has been compromised (Comp) and a malicious user had control over the entity. Fix concerns the functionality of an entity, and is applied when an entity was malfunctioning (Malfun). Thus, an entity can be restored but still malfunction or fixed but still compromised.

3.2.3 Assumed and derived properties

We distinguish between *assumed* or *derived* properties of an entity. Assumed, properties (prefixed with α) are part of the security assumptions for the system, denote initial knowledge and need to be explicitly declared. Derived properties (prefixed κ) are obtained by applying the derivation rules to the known properties. The set of assumed properties is:

$$\mathcal{P}_\alpha = \{\alpha\mathrm{Comp}, \alpha\mathrm{Malfun}, \alpha\mathrm{Vul}\}.$$

Derived properties indicate security properties that may/can become true (κ), as a malicious user exploits vulnerabilities. The set of all derived properties is:

$$\mathcal{P}_\kappa = \{\kappa\mathrm{Comp}, \kappa\mathrm{Malfun}, \kappa\mathrm{Vul}, \kappa\mathrm{Det}, \kappa\mathrm{Rest}, \kappa\mathrm{Fix}\}.$$

3.3 Relations

The entities of our hybrid system have different relations between each other as well as different relations to the threats. For simplicity, we consider binary and ternary relations; further relation types can easily be added if needed.

3.3.1 Relations between entities

We start by introducing the relations that hold among the entities of the system. $\mathcal{R}$ represents the set of all relations. These include structural relations (containment, interconnection) as well monitoring and control relations influencing the way in which entities can respond to threats.

$$\boxed{\{\mathrm{Contain}, \mathrm{Control}, \mathrm{Connect}, \mathrm{Depend}, \mathrm{Check}, \mathrm{Replicate}\} \in \mathcal{R}}$$

Contain $\mathrm{Contain}(A, B)$: means that A contains B, and represents how the system is composed, e.g., a room contains a server, or a server A contains data B. This permits to represent the structure of the system, and from it how the attack could propagate.

Control $\mathrm{Control}(A, B)$: means that entity A controls entity B, e.g., a person controls the use of their identification badge, or a controller controls the sensors.

Connect $\mathrm{Connect}(A, B, C)$: means that A connects B to C, e.g., the door connects the room with the hall, or network A connects server B with server C. The $\mathrm{Connect}$ relation is unidirectional and allows to identify how an attack can spread laterally or how such spread can be prevented, e.g., by removing the connection.

Depend $\mathrm{Depend}(A, B)$: the functionality of A depends on the functionality provided by B. For example, the air conditioning depends on its outside fan, or the web server depends on the data base. The depend relation is used for example when identifying the spread of the vulnerabilities or the protective measures.

Check $\mathrm{Check}(A, B)$: means A checks that B is functioning normally and thus detects malfunctions.

Replicate $\mathrm{Replicate}(A, B)$: A is a replica of B. The availability of a replica makes it possible to fix an entity.

3.3.2 *Relations between entities and threats*

In the previous section, we presented the relations between the entities of the system where threats can propagate. In this section, we introduce the relations between the entities and their threats. These relations permit to represent which entities are vulnerable to a particular threat, identify which other entities are vulnerable or how entities can protect each other. The set of relations is presented below.

$$\boxed{\{\mathrm{Protect}, \mathrm{Monitor}, \mathrm{Spread}, \mathrm{PotentiallyVul}\} \in \mathcal{R}}$$

Protect $\mathrm{Protect}(A, B, T)$: A protects B from threat T, e.g., a lock protects the door from unauthorised access, or firewalls protect systems from unauthorised traffic. $\mathrm{Protect}$ means that entity A is able to protect entity B from a threat T. If an entity is vulnerable and not protected, then it can be compromised.

Monitor $\mathrm{Monitor}(A, B, T)$: A monitors B from threat T, e.g., an intrusion detection system monitors the system against cyber-attacks, or a camera monitors the room against thieves. This expresses that attacks can be detected, even if they may not be prevented. Thus, if T can compromise B, then $\mathrm{Monitor}(A, B, T)$ describes that A can detect that B is compromised by T. Note that if A monitors B from threat T, this does not mean that A protects B from T. Although both the Check and $\mathrm{Monitor}$ relations monitor the target system, Check refers to the functionality of the systems, whereas $\mathrm{Monitor}$ refers to its security.

Spread $\mathrm{Spread}(A, T)$: A can propagate threat T, e.g., a phishing email can be used to spread a malware, or an IoT device can enable an attack to spread to other devices or to the controller. This property connects a device with the threats it can propagate.

Potentially Vulnerable $\mathrm{PotentiallyVul}(A, T)$: A can become vulnerable to threat T, e.g., a user can be vulnerable to phishing. This property is used to model that an entity can become vulnerable to threat T in certain circumstances, e.g., when it malfunctions.

3.4 High-level properties

Having defined the properties and the relationships among entities and threats, we can now define some high-level properties ($\mathcal{P}_H$) that can be expressed in terms of lower level primitives. The use of these high-level properties enables a user to understand much more easily the security state of the system, e.g., if an entity is defended or monitored from a threat. They also permit to represent the overall state of an entity including its components, dependabilities, and connections. We introduce below the definitions of these high-level properties; Table 1 gives their formal definitions.

$$\mathcal{P}_H = \{\mathrm{Val}, \mathrm{Def}, \mathrm{Safe}, \mathrm{Mon}, \mathrm{Check}, \mathrm{Rep}\}.$$

Valid $\mathrm{Val}(A)$: A is *valid* when it has not been compromised and is not malfunctioning. An entity is *not valid* when it has either been compromised or it is malfunctioning.

Defended $\mathrm{Def}(A, T)$: A is *defended* from threat T when an entity B exists that protects A from T and B is valid. On the contrary, an entity is *not defended* from T when no such B exists, or it exists but is not valid.

Safe $\mathrm{Safe}(A, T)$: A is *safe* with respect to T when A is not vulnerable to T or can be defended from T. A is *not safe* from T when it is vulnerable and not defendable from T.

Monitored $\mathrm{Mon}(A, T)$: A is *monitored* for threat T, when an entity B that monitors A with respect to threat T exists and is valid. Entity A is *not monitored* when no such entity B exists, or it exists but is not valid.

Checked $\mathrm{Check}(A)$: A is *checked*, when an entity B checks the functionality of A and B is valid. Entity A is *not checked* when no such entity B exists, or it exists but is not valid.

Replicated $\mathrm{Rep}(A)$: A is *replicated*, when at least one entity B that replicates A exists and B is valid. Entity A is *not replicated* when no such B exists that replicates A, or it exists but is not valid.

4 Threat Analysis

In this section, we first introduce the derivation rules used for our threat analysis. These rules apply to the relations and properties initially known to derive the new security properties that reflect the state of the system and can be used to protect it better. Through the derivation rules, we deduce which entities can become vulnerable, compromised, and/or malfunction. We can also identify if existing mechanisms can detect when an entity is compromised and whether it is possible to restore it.

We then describe how the derived security properties, help to construct the attack graph for our system, which graphically represents the results of the threat analysis and the attack paths through the system.

4.1 Derivation Rules

Beyond a basic number of derivation rules, which we describe first, there are also other derivation rules for each of the security properties of the entities. These derivation rules can derive when an entity is compromised, malfunctioning, vulnerable, restorable, and fixed.

4.1.1 *Basic derivation rules*

Basic derivation rules state that if a property is assumed true, then naturally it can be derived to be true (Rules 1-3). Furthermore, these rules represent the transitivity of relations such as $\mathrm{Replicate}$ and Depend. Note that $\mathrm{Contain}(A, B)$ and $\mathrm{Control}(A, B)$ are not necessarily transitive when applied to the human, cyber, and physical aspects of the system.

$$\alpha\mathrm{Comp}(A, T) \rightarrow \kappa\mathrm{Comp}(A, T) \quad (1)$$

$$\alpha\mathrm{Vul}(A, T) \rightarrow \kappa\mathrm{Vul}(A, T) \quad (2)$$

$$\alpha\mathrm{Malfun}(A) \rightarrow \kappa\mathrm{Malfun}(A) \quad (3)$$

$$\mathrm{Replicate}(A, B) \wedge \mathrm{Replicate}(B, C) \rightarrow \mathrm{Replicate}(A, C) \quad (4)$$

$$\mathrm{Depend}(A, B) \wedge \mathrm{Depend}(B, C) \rightarrow \mathrm{Depend}(A, C) \quad (5)$$

4.1.2 *Derivation rules for compromised*

We now introduce the derivation rules to reason about how threats can compromise different entities and propagate across the system.

Rule 6: A can be compromised by threat T when A is not safe from T and an entity B, which controls A, can be compromised and spread threat T.

$$\mathrm{Control}(B, A) \wedge \kappa\mathrm{Comp}(B) \wedge \mathrm{Spread}(B, T) \wedge \neg\mathrm{Safe}(A, T) \rightarrow \kappa\mathrm{Comp}(A, T) \quad (6)$$

$\mathrm{Val}(A) := \neg\mathrm{Comp}(A) \wedge \neg\mathrm{Malfun}(A)$	$\neg\mathrm{Val}(A) := \mathrm{Comp}(A) \vee \mathrm{Malfun}(A)$
$\mathrm{Def}(A,T) := \exists B.\ \mathrm{Protect}(B,A,T) \wedge \mathrm{Val}(B)$	$\neg\mathrm{Def}(A,T) := \nexists B.\ \mathrm{Protect}(B,A,T) \vee (\forall B.\mathrm{Protect}(B,A,T) \wedge \neg\mathrm{Val}(B))$
$\mathrm{Safe}(A,T) := \neg\mathrm{Vul}(A,T) \vee \mathrm{Def}(A,T)$	$\neg\mathrm{Safe}(A,T) := \mathrm{Vul}(A,T) \wedge \neg\mathrm{Def}(A,T)$
$\mathrm{Mon}(A,T) := \exists B.\ \mathrm{Monitor}(B,A,T) \wedge \mathrm{Val}(B)$	$\neg\mathrm{Mon}(A,T) := \nexists B.\ \mathrm{Monitor}(B,A,T) \vee (\forall B\ \mathrm{Monitor}(B,A,T) \wedge \neg\mathrm{Val}(B))$
$\mathrm{Check}(A) := \exists B.\ \mathrm{Check}(B,A) \wedge \mathrm{Val}(B)$	$\neg\mathrm{Check}(A) := \nexists B.\ \mathrm{Check}(B,A) \vee (\forall B\ \mathrm{Check}(B,A) \wedge \neg\mathrm{Val}(B))$
$\mathrm{Rep}(A) := \exists B.\ \mathrm{Replicate}(B,A) \wedge \mathrm{Val}(B)$	$\neg\mathrm{Rep}(A) := \nexists B.\ \mathrm{Replicate}(B,A) \vee (\forall B.\ \mathrm{Replicate}(B,A) \wedge \neg\mathrm{Val}(B))$

TABLE 1
High-level properties definitions

Rule 7: A can be compromised by threat T when A is not safe from T, and A is connected to B through C, B can be compromised and spreads T, and either C can be compromised or C is not protected against T.

$$\boxed{\begin{aligned}&\mathrm{Connect}(C,B,A) \wedge \kappa\mathrm{Comp}(B) \wedge \mathrm{Spread}(B,T)\\ &\wedge\neg\mathrm{Safe}(A,T) \wedge (\kappa\mathrm{Comp}(C) \vee \neg\mathrm{Def}(C,T))\\ &\to \kappa\mathrm{Comp}(A,T)\end{aligned}} \tag{7}$$

Rule 8: A can be compromised by threat T when A is not safe from T, and either A contains B or is contained in B, and B can be compromised and spread T.

$$\boxed{\begin{aligned}&(\mathrm{Contain}(B,A) \vee \mathrm{Contain}(A,B)) \wedge \kappa\mathrm{Comp}(B)\\ &\wedge\mathrm{Spread}(B,T) \wedge \neg\mathrm{Safe}(A,T) \to \kappa\mathrm{Comp}(A,T)\end{aligned}} \tag{8}$$

Example 1. We illustrate the application of the above rules using the scenario introduced in Section 2.1. In this scenario, an employee (E) controls their PC, where the password to open the safe is stored. The employee can access emails (including phishing emails $\mathrm{phishEm}$) on a server (server) that connects the employee with the emails.

$$\mathrm{Control}(\mathrm{E}, PC) \qquad \mathrm{Contain}(PC, D)$$
$$\mathrm{Connect}(PhishAttack, E, PhishEmail)$$

The employee is assumed vulnerable to phishing attacks, denoted by $\mathrm{phishAt}$ and the PC is assumed vulnerable to malware. The password (Pw) is assumed vulnerable to being stolen, denoted by $\mathrm{stealInfo}$. There are no protective measures P in place, against these threats.

$$\begin{array}{ll}\alpha\mathrm{Vul}(\mathrm{E},\mathrm{phishAt}) & \nexists P.\ \mathrm{Protect}(P,\mathrm{E},\mathrm{phishAt})\\ \alpha\mathrm{Vul}(PC, malware) & \nexists P.\ \mathrm{Protect}(P, PC, malware)\\ \alpha\mathrm{Vul}(\mathrm{Pw},\mathrm{stealInfo}) & \nexists P.\ \mathrm{Protect}(P,\mathrm{Pw},\mathrm{stealInfo})\end{array}$$

Furthermore, we know that: the phishing attack ($\mathrm{phishAt}$) can be spread via the phishing emails; the employee can accidentally spread the malware code; the PC spread the steal information threat, as once you have access to the PC, you can get access to the stored information in it.

$$\mathrm{Spread}(\mathrm{phishEm},\mathrm{phishAt}) \qquad \mathrm{Spread}(\mathrm{E}, malware)$$
$$\mathrm{Spread}(PC,\mathrm{stealInfo})$$

Using the high-level properties, we can derive that the employee (E), the PC, and password are not *defended* and not *safe*, while the server is not defended (see below).

$$\begin{aligned}&\neg\mathrm{Def}(\mathrm{Pw},\mathrm{stealInfo}) = \nexists P.\ \mathrm{Protect}(P,\mathrm{Pw},\mathrm{stealInfo})\\ &\neg\mathrm{Def}(PC, malware) = \nexists P.\ \mathrm{Protect}(P, PC, malware)\\ &\neg\mathrm{Def}(\mathrm{E},\mathrm{phishAt}) = \nexists P.\ \mathrm{Protect}(P,\mathrm{E},\mathrm{phishAt})\\ &\neg\mathrm{Def}(\mathrm{server},\mathrm{phishAt}) = \nexists P.\ \mathrm{Protect}(P,\mathrm{server},\mathrm{phishAt})\\ &\neg\mathrm{Safe}(\mathrm{Pw},\mathrm{stealInfo}) = \alpha\mathrm{Vul}(\mathrm{Pw},\mathrm{stealInfo})\wedge\\ &\qquad\qquad\neg\mathrm{Def}(\mathrm{Pw},\mathrm{stealInfo})\\ &\neg\mathrm{Safe}(PC, malware) = \alpha\mathrm{Vul}(PC, malware)\wedge\\ &\qquad\qquad\neg\mathrm{Def}(PC, malware)\\ &\neg\mathrm{Safe}(\mathrm{E},\mathrm{phishAt}) = \alpha\mathrm{Vul}(\mathrm{E},\mathrm{phishAt}) \wedge \neg\mathrm{Def}(\mathrm{E},\mathrm{phishAt})\end{aligned}$$

Using Rule 7 in conjunction with the above, we can derive that the employee can be compromised by a phishing attack.

$$\begin{aligned}&\mathrm{Connect}(\mathrm{server},\mathrm{phishEm},\mathrm{E}) \wedge \kappa\mathrm{Comp}(\mathrm{phishEm})\wedge\\ &\mathrm{Spread}(\mathrm{phishEm},\mathrm{phishAt}) \wedge \neg\mathrm{Safe}(\mathrm{E},\mathrm{phishAt})\wedge\\ &\neg\mathrm{Def}(\mathrm{server},\mathrm{phishAt}) \to \kappa\mathrm{Comp}(\mathrm{E},\mathrm{phishAt})\end{aligned}$$

Using Rule 6, we can then derive that the employee's PC can also be *compromised*.

$$\begin{aligned}&\mathrm{Control}(\mathrm{E}, PC) \wedge \kappa\mathrm{Comp}(\mathrm{E}) \wedge \mathrm{Spread}(\mathrm{E}, malware)\wedge\\ &\neg\mathrm{Safe}(PC, malware) \to \kappa\mathrm{Comp}(PC, malware)\end{aligned}$$

Finally, using Rule 8 we can derive that the password can be *compromised*.

$$\begin{aligned}&\mathrm{Contain}(PC,\mathrm{Pw}) \wedge \kappa\mathrm{Comp}(PC) \wedge \mathrm{Spread}(PC,\mathrm{stealInfo})\\ &\wedge\neg\mathrm{Safe}(\mathrm{Pw},\mathrm{stealInfo}) \to \kappa\mathrm{Comp}(\mathrm{Pw},\mathrm{stealInfo})\end{aligned}$$

Thus, based on the given information we can derive that the employee's password can be compromised by exploiting both the human and the cyber vulnerabilities of the system.

4.1.3 Derivation rules for malfunctioned

Rule 9 specifies that an entity A, when compromised by threat T, can malfunction. **Rule 10** specifies that when A depends on entity B and B malfunctions, then A can also malfunction.

$$\kappa\mathrm{Comp}(A,T) \to \kappa\mathrm{Malfun}(A) \tag{9}$$
$$\mathrm{Depend}(A,B)\ \wedge\ \kappa\mathrm{Malfun}(B) \to \kappa\mathrm{Malfun}(A) \tag{10}$$

Example 2. Let us assume a physical web server (physical entity) that hosts a $\mathrm{website}$ (cyber entity). The functionality of the website strictly depends on the functionality of its web server: $\mathrm{Depend}(\mathrm{website},\mathrm{server})$.
If the server can be physically compromised for a threat T, then we can derive that it can malfunction by using Rule 9, as shown below.

$$\kappa\mathrm{Comp}(\mathrm{server}, T) \to \kappa\mathrm{Malfun}(\mathrm{server})$$

Given the dependency between the website and its server, we can derive that the website can also malfunction by applying Rule 10.

$$\begin{aligned}&\mathrm{Depend}(\mathrm{website},\mathrm{server}) \wedge \kappa\mathrm{Malfun}(\mathrm{server})\\ &\qquad\to \kappa\mathrm{Malfun}(\mathrm{website})\end{aligned}$$

4.1.4 Derivation rule for vulnerabilities

When A malfunctions, and A can be vulnerable to threat T, then A can become vulnerable to T, **Rule 11**.

$$\boxed{\kappa\text{Malfun}(A) \land \text{PotentiallyVul}(A, T) \to \kappa\text{Vul}(A, T)} \tag{11}$$

Example 3. For example, a broken lock can become vulnerable to being open by unauthorised users.

$$\kappa\text{Malfun}(lock) \land \text{PotentiallyVul}(lock, breakOpen) \\ \to \kappa\text{Vul}(lock, breakOpen)$$

4.1.5 Derivation rule for detecting threats

Rule 12 states that when A can be compromised by threat T and A is monitored for threat T, then T can be detected for A.

$$\boxed{\kappa\text{Comp}(A, T) \land \text{Mon}(A, T) \to \kappa\text{Det}(A, T)} \tag{12}$$

Example 4. For example, if a system is monitored for intrusions (e.g., with an IDS), then the intrusions can be detected.

$$\kappa\text{Comp}(\text{system}, \text{intrusion}) \land \text{Mon}(\text{system}, \text{intrusion}) \\ \to \kappa\text{Det}(\text{system}, \text{intrusion})$$

4.1.6 Derivation rule for restoring services

Rule 13 states that when threat T can be detected for A and A has been replicated, then A can be restored.

$$\boxed{\kappa\text{Det}(A, T) \land \text{Rep}(A) \to \kappa\text{Rest}(A)} \tag{13}$$

Example 5. Restoring a system from backup stands as an immediate example.

$$\kappa\text{Det}(\text{system}, \text{intrusion}) \land \text{Rep}(\text{system}) \to \kappa\text{Rest}(\text{system})$$

4.1.7 Derivation rule for fixed

Rule 14 states that when A can *malfunction* and is checked (i.e., its malfunction can be detected) then A can be fixed.

$$\boxed{\kappa\text{Malfun}(A) \land \text{Check}(A) \to \kappa\text{Fix}(A)} \tag{14}$$

Example 6. For example, if the air conditioning (AC) malfunctions, and it is detected, then the AC can be fixed.

$$\kappa\text{Malfun}(\text{AC}) \land \text{Check}(\text{AC}) \to \kappa\text{Fix}(\text{AC})$$

4.2 Application of the threat analysis

The derivation rules introduced above are used automatically, given certain inputs to derive new (security) properties for the system's entities. It can also identify possible entities that will protect or monitor other entities, or how the threat will propagate in the system from one entity to the other.

Given the initial relations and properties of the entities, the threat analysis will derive new security properties, which together with answers to other queries will be used to output the threat analysis model. We also provide a graphical representation for this in the form of an attack graph.

4.2.1 Attack graphs

The derivation rules allow us to compute the basic and auxiliary properties (i.e., can be compromised, malfunction, vulnerable, detected, restored, and fixed). This also enables us to construct an attack graph [17] for a target in our system and its properties that we want to verify. For example, we can build the attack graph to compromise a web server or to a door malfunction. This graph offers a graphical and simple representation of the result of the threat analysis.

Attack graphs are a powerful tool for security assessment by analysing network vulnerabilities and the paths attackers can use to compromise system resources. Attack graphs permit a priori analysis of the possible avenues an attacker can exploit to compromise the system. Thus, they can be used to focus on the most-effective threats and produce a better countermeasures selection [18], which is also known as static analysis.

We show in Figure 4 part of a generated attack graph, where we also show the relations amongst the derivation rules. Specifically, in Figure 4, the nodes are assumed and derived properties of the entities, the dashed links are the relationships among entities, while the solid red links are the connections created by the triggered derivation rules. In the next section, we show the attack graphs of two different case studies taken from a smart building described in Section 2.1.

5 CASE STUDY EVALUATION

In this section, we apply our model to two use cases taken from the smart building scenario introduced in Section 2.1 and one use case from a critical infrastructure system. Given the space constraints, we describe in detail the first scenario and provide only the significant steps for the remaining two. Our threat model is able to analyse the given information automatically. It provides the users with the derived security properties as well as a graphical representation of the security properties and relations of the entities composing the system, in the form of an attack graph.

5.1 Unauthorized Safe Box Access

We continue with the Safe box example introduced in Section 2.1 and Example 1. Using our model, we will show how the attacker can access the safe box and all the steps leading to the successful attack. We will explain in detail how the security properties of the various entities change. This detailed analysis is performed by our model automatically and the result is provided as a list of answers to the users' queries and a graphical representation, see Figure 5.

The contents of a safe box are protected through a combination that requires a password. The safe box is located in a room that can be accessed through a door, which can be open only with an RFID card. Using the results of Example 1, we know that the password can be compromised as it can be stolen through a phishing attack: $\kappa\text{Comp}(\text{Pw}, \text{stealInfo})$. However, the attacker Att, still needs to physically access the safe to use the password and the room is protected from unauthorised access by the door lock, denoted by lock.

We also know that the door lock can also be compromised, by following the steps shown in the attack graph generated by our threat model, presented in Figure 5. Specifically, the door lock is *controlled* by its RFID card (denoted as key) and is vulnerable to unauthorised users controlling the card or making a copy of it (unAuthUser). Moreover, no other protection measure is in place to defend the door lock such as a human guard. Thus, the door lock is not safe from being opened by an unauthorised user. We provide below the relations and security properties for these entities.

$$\text{Control}(\text{key}, \text{lock}), \quad \alpha\text{Vul}(\text{lock}, \text{unAuthUser}), \\ \nexists P.\ \text{Protect}(P, \text{lock}, \text{unAuthUser}), \quad \neg\text{Def}(\text{lock}, \text{unAuthUser}), \\ \neg\text{Safe}(\text{lock}, \text{unAuthUser})$$

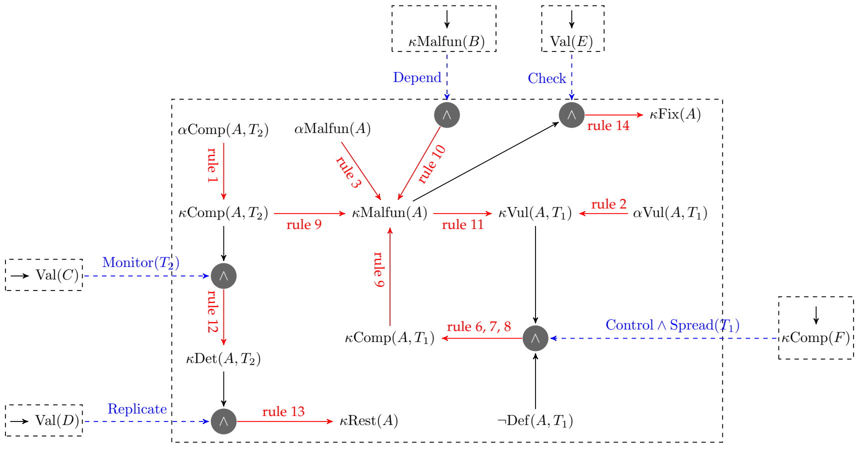

Fig. 4. Part of a Generated Attack Graph from our Threat Analysis

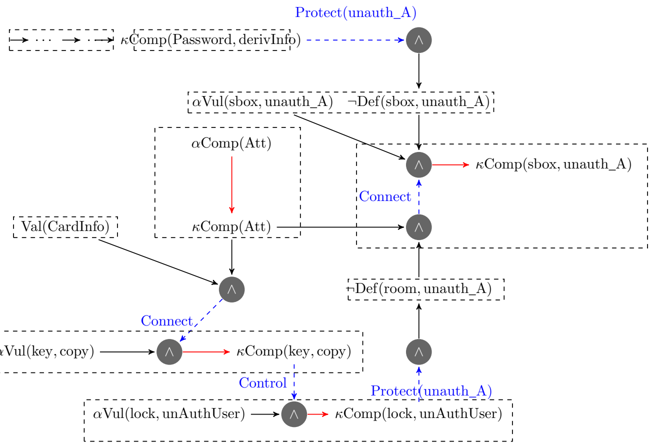

Fig. 5. Attack Graph for the Safe Box Scenario

The RFID card is assumed to be vulnerable to the threat of being physically copied (denoted as copy), and there are no protection measures in place that prevent the card from being copied. Therefore, using the high-level property definition for Safe, we can state that the RFID card is not safe with respect to the threat of being copied (copy). Any user can open the door, using the card, thus, if the card is compromised, then it can spread the unAuthUser threat.

$$\mathrm{Vul(key, copy)}, \quad \neg\mathrm{Def(key, copy)}, \quad \neg\mathrm{Safe(key, copy)}, \\ \mathrm{Spread(key, unAuthUser)}$$

We assume that the Att has malicious purposes, thus, can be considered compromised, ($\kappa\mathrm{Comp(Att)}$), and they have information about the RFID card (e.g., who is the employee that has the card[6]). The $CardInfo$ connects the attacker to the RFID card. Furthermore, once the attacker has this information, we can consider the card information to be compromised ($\kappa\mathrm{Comp}(CardInfo)$). Together with the high-level property about the RFID card described above (the RFID card is not safe to the copy attack) and that the attacker can spread the copy attack, we can derive (using Rule 7) that the RFID card can be compromised by the copy card threat, see below.

$$\mathrm{Connect}(CardInfo, \mathrm{Att}, \mathrm{key}) \wedge \kappa\mathrm{Comp(Att)} \wedge \\ \mathrm{Spread(Att, copy)} \wedge \neg\mathrm{Safe(key, copy)} \wedge \\ \kappa\mathrm{Comp}(CardInfo) \rightarrow \kappa\mathrm{Comp(key, copy)}$$

Using the information above that the card can be compromised (i.e., copied) we can derive using Rule 6 that the door lock can be compromised by copying the card. In detail, we know that the lock is controlled by the RFID card and is not safe to unAuthUser threat and that the RFID card can spread this threat. Together with the derived information that the card can be compromised, we can derive that the lock can also be compromised, see below.

$$\mathrm{Control(key, lock)} \wedge \kappa\mathrm{Comp(key, copy)} \wedge \\ \mathrm{Spread(key, unAuthUser)} \wedge \neg\mathrm{Safe(lock, unAuthUser)} \\ \rightarrow \kappa\mathrm{Comp(lock, unAuthUser)}$$

If the lock is compromised, it is no longer valid (see high-level property definitions) and the room that the door leads to is no longer defended from unauthorized access (unauth_A), as the only protection in place is no longer valid.

$$\neg\mathrm{Val(lock)}, \quad \mathrm{Protect(lock, room, unauth\_A)}, \\ \neg\mathrm{Def(room, unauth\_A)}$$

From Example 1, we know that the password (Pw) can be compromised as it can be stolen through a phishing attack: $\kappa\mathrm{Comp(Pw, stealInfo)}$. Thus, we can define the Pw to not be valid ($\neg\mathrm{Val(Pw)}$), following the high-level property definition.

$$\neg\mathrm{Val(Pw)} = \mathrm{Comp(Pw)}$$

As the Pw was protecting the sbox from unauthorised accesses but is not valid any more, and there is no other way to protect the safe box, then we can define that it is not defendable.

$$\neg\mathrm{Def(sbox, unauth\_A)} = \mathrm{Protect(Pw, sbox, unauth\_A)} \\ \wedge\neg\mathrm{Val(Pw)}$$

Following the definition of $\neg$Safe we can state that sbox is not safe any more, see below.

$$\neg\mathrm{Safe(sbox, unauth\_A)} = \mathrm{Vul(sbox, unauth\_A)} \\ \wedge\neg\mathrm{Def(sbox, unauth\_A)}$$

We know that the room connects the attacker to the safe box and that the attacker can spread the unauthorised access. Furthermore, we derived that the safe box is not safe to unauthorised access and the room is not defendable. We can derive from the information above by applying Rule 7 that the safe box can be

6. This information is useful for the attacker to then copy the RFID card even by staying at a distance from the employee and the card.

compromised through an authorised access, as the attacker can physically access the room[7], by compromising its protection measures, and they can insert the password of the safe box.

$$\begin{aligned}\mathrm{Connect}(\mathrm{room}, \mathrm{Att}, \mathrm{sbox}) \wedge \kappa\mathrm{Comp}(\mathrm{Att}) \wedge \\ \mathrm{Spread}(\mathrm{Att}, \mathrm{unauth\_A}) \wedge \neg\mathrm{Safe}(\mathrm{sbox}, \mathrm{unauth\_A}) \wedge \\ \neg\mathrm{Def}(\mathrm{room}, \mathrm{unauth\_A}) \rightarrow \kappa\mathrm{Comp}(\mathrm{sbox}, \mathrm{unauth\_A})\end{aligned}$$

To summarise, in order to compromise the safe box, the attacker needs to exploit the vulnerabilities of the *human* component of the system, i.e., the employee's vulnerability to phishing attacks, the vulnerabilities of the *cyber* component, i.e., the employee's PC, and the vulnerabilities of the *physical* component, i.e., the RFID card of the door lock.

As shown in this example, thanks to the use of our threat analysis the security administrator is able to understand which part of their system can be compromised. Moreover, due to the representation of the attack graph, the administrator is able to choose the best place where to put in place a new security mechanism (i.e., protection and/or monitor component) to interrupt the attack propagation or to prevent the occurrence of the attack. For example, it is possible to avoid the compromise of the safe box in several ways: (i) by installing an anti-phishing system on the email server; (ii) using a door lock where entry cards cannot be copied; (iii) add a guard in the room or (iv) adding a camera in the room. Clearly each solution has its advantages and disadvantages. However, through TAMELESS, the administrator has a holistic view on how to provide the security of a system, and they can consider each solution based on effectiveness, logistic, economical, and ethical aspects.

5.2 Compromise a Web Server

In this section, we discuss a second scenario where the attacker needs to exploit the vulnerabilities of different components of our smart building to compromise a particular entity.

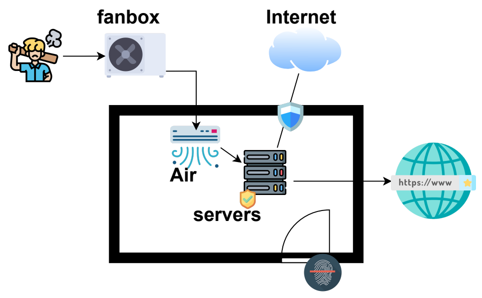

Fig. 6. A threat model for compromising a Web Server

The case study takes place in two different rooms, as shown in Figure 6. We consider a web server, denoted by server, located in one of the rooms, which is protected from unauthorised access, denoted by $\mathrm{unauth\_A}$, through biometric authentication, $\mathrm{bioMAuth}$. An air conditioning unit, denoted by AC, is located in the same room. The server functionality depends on the functionality of the AC, because if the AC is not working, then the server overheats and shuts down or can be damaged. The server hosts a particular $\mathrm{website}$, and is protected against cyber-attacks on the website ($\mathrm{Safe}(\mathrm{server}, \mathrm{cyberAttack})$). The functionality of the website *depends* on the functionality of the web server. The server is *connected* to the $\mathrm{internet}$. The functionality of the AC depends on the functionality of its fan-box, denoted by fan, located on the terrace of the building. We introduce below the main properties of this scenario.

7. For simplicity, we ignore that the attacker needs to physically get to the room, e.g., by entering the building.

$$\begin{aligned}&\mathrm{Contain}(\mathrm{server}, \mathrm{website}),\ \mathrm{Depend}(\mathrm{website}, \mathrm{server}),\\ &\mathrm{Contain}(\mathrm{room}, \mathrm{server}),\ \mathrm{Contain}(\mathrm{room}, \mathrm{AC}),\\ &\mathrm{Depend}(\mathrm{AC}, \mathrm{fan}),\ \mathrm{Depend}(\mathrm{server}, \mathrm{AC}),\\ &\mathrm{Connect}(\mathsf{path}, \mathrm{Att}, \mathrm{fan}),\ \mathrm{Connect}(\mathsf{path2}, \mathrm{Att}, \mathrm{room}),\\ &\alpha\mathrm{Vul}(\mathrm{fan}, \mathrm{PhysAtt}),\ \alpha\mathrm{Vul}(\mathrm{AC}, \mathrm{PhysAtt}),\ \alpha\mathrm{Comp}(\mathrm{Att}),\\ &\alpha\mathrm{Vul}(\mathrm{server}, \mathrm{cyberAttack}),\ \alpha\mathrm{Vul}(\mathrm{website}, \mathrm{cyberAttack}),\\ &\mathrm{Spread}(\mathrm{Att}, \mathrm{PhysAtt}),\ \mathrm{Protect}(\mathrm{bioMAuth}, \mathrm{room}, \mathrm{unauth\_A}),\\ &\mathrm{Safe}(\mathrm{server}, \mathrm{cyberAttack}),\ \mathrm{Safe}(\mathrm{bioMAuth}, \mathrm{falsification})\end{aligned}$$

A malicious attacker is not able to compromise the website or the web server using a cyber-attack, e.g, DoS, or malware, or by compromising *cyber components* related to them, as there are protection measures in place that are assumed not vulnerable. The attacker is also not able to compromise any *human component*, as the access to the room is restricted to a small group of trusted people. Furthermore, the attacker is also not able to access the server room, thus, they cannot compromise the server physically or take physical control over it.

Let us now analyse the given properties and relations for this case study. The AC unit is driven by an external fan-box. As this is placed outside, the box is vulnerable to *physical attacks*. Therefore, we can derive that it is not defendable and not safe with respect to physical attacks.

$$\neg\mathrm{Def}(\mathrm{fan}, \mathrm{PhysAtt}),\ \neg\mathrm{Safe}(\mathrm{fan}, \mathrm{PhysAtt}),\ \neg\mathrm{Def}(\mathsf{path}, \mathrm{PhysAtt})$$

This is automatically derived by Rule 7 as shown below.

$$\begin{aligned}\mathrm{Connect}(\mathsf{path}, \mathrm{Att}, \mathrm{fan}) \wedge \kappa\mathrm{Comp}(\mathrm{Att}) \wedge \neg\mathrm{Safe}(\mathrm{fan}, \mathrm{PhysAtt})\\ \wedge\mathrm{Spread}(\mathrm{Att}, \mathrm{PhysAtt}) \wedge \neg\mathrm{Def}(\mathsf{path}, \mathrm{PhysAtt})\\ \rightarrow \kappa\mathrm{Comp}(\mathrm{fan}, \mathrm{PhysAtt})\end{aligned}$$

We can derive that the fan-box can *malfunction* given that it can be compromised, using rule 9, which states that if an entity is compromised, then it can malfunction.

$$\kappa\mathrm{Comp}(\mathrm{fan}, \mathrm{PhysAtt}) \rightarrow \kappa\mathrm{Malfun}(\mathrm{fan})$$

As the functionality of the AC depends on the functionality of its fan-box, we can then derive that if the fan-box malfunctions the AC can also malfunction, using rule 10.

$$\kappa\mathrm{Malfun}(\mathrm{fan}) \wedge \mathrm{Depend}(\mathrm{AC}, \mathrm{fan}) \rightarrow \kappa\mathrm{Malfun}(\mathrm{AC})$$

As the functionality of the server depends on the AC, we can further derive that the server can malfunction if the air conditioning malfunctions using rule 10.

$$\kappa\mathrm{Malfun}(\mathrm{AC}) \wedge \mathrm{Depend}(\mathrm{server}, \mathrm{AC}) \rightarrow \kappa\mathrm{Malfun}(\mathrm{server})$$

As the functionality of the $\mathrm{website}$ depends on its server, we can derive that the $\mathrm{website}$ can malfunction, as shown also in Example 2.

$$\begin{aligned}\kappa\mathrm{Malfun}(\mathrm{server}) \wedge \mathrm{Depend}(\mathrm{website}, \mathrm{server})\\ \rightarrow \kappa\mathrm{Malfun}(\mathrm{website})\end{aligned}$$

Thus, in this case, the attacker can compromise the functionality of a digital component of the system (the website) by performing a *physical attack* on a physical component, that at first sight is not directly connected with the website. In particular, as shown in the attack graph (Figure 7), the attacker does not need to physically access the server room but can create a cascading malfunction between the different components leveraging the fact that they depend on each other. This use case shows the user that there is no need to add more protection to the room, the server, or the employee, but better physical security to the space outside e.g., the terrace and the fan-box.

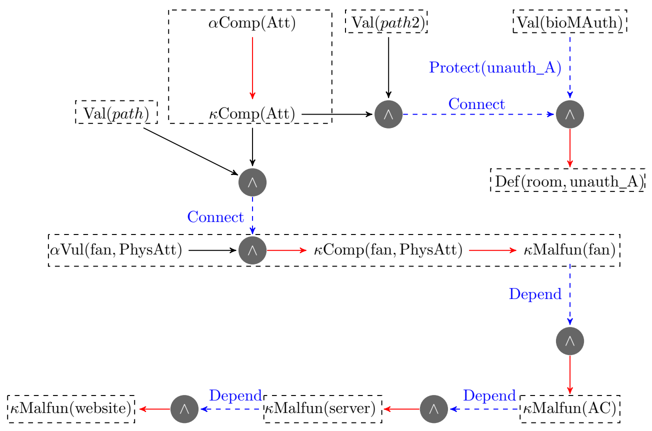

Fig. 7. Attack Graph for the Web Server Scenario

5.3 Attack on a Wind Farm

We now show a representation in our model of an attack on a wind farm, first presented and realised in [3], [15]. In this case, the attacker is able to physically access the wind turbine by physically breaking the lock on the wind turbine door. Once the attacker accesses the wind turbine, they place a Rasberry Pi into the network switch, thus, enabling remote digital access. The network switch is connected through the network with the Wind Farm Control Network (WFCN). Thus, the device newly plugged in can now access the WFCN that has no protection measures on the inside network. Several attacks can now occur, e.g., implanting malware in the WFCN, obtaining information, network configuration, protocols, as well as, the disruption of the Wind Farm operation (that brings economical losses) and damage to key physical components given the architecture of the Wind Turbine and the Wind Farm.

In our model, it is possible to see that the Wind Turbine (WT) is potentially vulnerable to physical unauthorised access (unauth_A) as it is protected against this attack by a lock, which is vulnerable to be physically broken $(physBreak)$. Given that the lock is not protected against this type of attack we can state that it is not defended and not safe against this attack.

$$\alpha\text{Vul}(\text{WT}, \text{unauth\_A}),\ \text{Protect}(\text{lock}, \text{WT}, \text{unauth\_A}),\\ \neg\text{Def}(\text{lock}, \text{physBreak})$$

Given the above information, we can derive that the lock is compromised by the attacker (Att) through a physical attack. As the attacker can now physically get to the Wind Turbine, which is usually located in remote areas and with no surveillance and thus not defended.

$$\text{Connect}(\text{path}, \text{Att}, \text{lock}) \wedge \alpha\text{Comp}(\text{Att}) \wedge\\ \text{Spread}(\text{Att}, \text{physBreak}) \wedge \neg\text{Safe}(\text{lock}, \text{physBreak}) \wedge\\ \neg\text{Def}(\text{path}, \text{physBreak}) \rightarrow \text{Comp}(\text{lock}, \text{physBreak})$$

Once the lock has been compromised, the attacker can physically access the Wind Turbine.

$$\text{Connect}(\text{lock}, \text{Att}, \text{WT}) \wedge \text{Comp}(\text{Att}) \wedge\\ \text{Spread}(\text{Att}, \text{unauth\_A}) \wedge \text{Comp}(\text{lock}) \wedge \neg\text{Safe}(\text{WT}, \text{unauth\_A})\\ \rightarrow \text{Comp}(\text{WT}, \text{unauth\_A})$$

Subsequently, the attacker by having physical access, can connect the malicious Rasberry Pi in the network switch located inside the Wind Tower.

$$\text{Contain}(\text{WT}, \text{switch}) \wedge \kappa\text{Comp}(\text{WT}) \wedge \neg\text{Safe}(\text{switch}, \text{unauth\_A})\\ \wedge \text{Spread}(\text{WT}, \text{unauth\_A}) \rightarrow \text{Comp}(\text{switch}, \text{unauth\_A})$$

The Rasberry Pi can then attack the Wind Farm Control Network e.g., through a malware program (Mal)[8].

$$\text{Connect}(\text{network}, \text{switch}, \text{WFCN}) \wedge \kappa\text{Comp}(\text{switch}) \wedge\\ \text{Spread}(\text{switch}, malware) \wedge \neg\text{Safe}(\text{WFCN}, \text{Mal}) \wedge\\ \neg\text{Def}(\text{network}, malware) \rightarrow \text{Comp}(WFCN, malware)$$

An analysis of this example also shows the possible mitigations that can be deployed including: (i) putting a physical guard or a camera to monitor the entrance to the wind turbines; (ii) improving the quality of the lock; (iii) protecting the internal network from internal attacks e.g., using a 0-trust strategy.

6 Related Work

The work presented here relates to a number of different research studies ranging from threat modelling and analysis, to formal specifications of smart systems. Specifically, we focus on the closest related approaches that seek to analyse the security proprieties in cyber-physical and smart systems and the existing studies that focus on attack graph generation from threat modelling and analysis.

Generally, a significant amount of effort has been devoted in the literature to the development of threat models and analysis. A recent survey of existing threat models and threat analysis for other application domains is provided in [19].

6.1 Threat analysis of cyber-physical systems

In recent years, threat analysis for cyber-physical and smart systems is becoming a popular topic in cybersecurity. Some of the most promising results are shown in [9], [20], [21], [22], where the authors present novel threat models but none of these studies models together the system's cyber, physical and human aspects together.

Tsigkanos *et al.* [20], [21] described an approach for topology aware adaptive security for cyber-physical systems, focusing on the interplay between cyber and physical spaces characterising the operational environment. Even though this work shares some ideas with our paper, their goal is different: they aim to identify potential violations of security requirements (speculative threat analysis), whilst we aim to identify the untrusted elements of our system.

Lemaire *et al.* [9] presented a tool that automates in a formal way the threat analysis of ICS (Industrial Control Systems). Using a knowledge-based system, the tool extracts vulnerabilities both at the component and system level. When the vulnerabilities are extracted, the security administrators can adapt the system to mitigate or remove them.

Oladimeji *et al.* [22] proposed a goal-oriented approach for analysing cyber threats. The approach provides support for guiding the threat analysis process using the notions of negative soft goals for detecting cyber threats and identifying solutions for threat mitigation.

Akella *et al.* [23] defined a formal method to detect threats in confidential communications. The proposed approach detects threats by analysing the interactions between physical systems with the cyber components. Sensitive information about a physical component can be inferred through behaviour observation about the related cyber components.

Finally, in [6] the authors constructed a model for threat analysis on Virtualized Systems called FATHoM. We took inspiration from this work's formal description of the system components relations and security properties. In our work we extend, change and enrich this model, by adding more types of relationships, security proprieties, derivation rules and defining

8. The network and the other components might be protected from external malware from outside components, but the internal network is considered trusted and thus, no protective measures are in place.

an attack graph representation, in order to properly model and detect hybrid threats in smart systems. In particular, our framework represents entities that can be of cyber, physical, or human nature, while FATHoM is constructed to analyse the threat models for Virtualized (Cloud) Systems. Given the triple nature of our cyber-physical and smart systems, we include properties and relations that cannot hold in Virtualized Systems. Furthermore, our model represents also the functionality property of the system's entities.

6.2 Threat Detection via Attack Graphs

Several studies [5], [8], [24], [25], [26], [27] have investigated the generation of attack graphs from the threat model and analysis of the system. These studies use the generated attack graphs to detect and show threats relations and possible mitigations, in a graphical representation. To the best of our knowledge, we list and present below the more relevant studies.

MulVAL (Multihost, multistage Vulnerability Analysis) is a framework for modelling and analyzing the interaction of software bugs with system and network configurations. As it uses First Order Logic, it is able to automatically infer system vulnerability and derives the attack graphs with the probability that an adversary could successfully conduct an attack. MulVAL's language and infer system was extended in several works and by several research groups. The most significant improvements are reported in [28], [29]. In this work we use MulVAL for the automatic graph generation provided the inputs from our hybrid threat model.

In [8], the authors proposed an attack graph generation tool that builds upon MulVAL. In their representation, a node in the graph is a logical statement, i.e., representing some aspect and propriety of the network. While, the edges of the graph, specify the causality relations between network configurations and an attacker's potential privileges. The attack graph, in this way, illustrates snapshots of attack steps and causes of the attacks ("how and why the attack can happen").

In [24], the authors presented methodologies that starting from the information of the MulVAL model are able to: (i) automatically identify portions of an attack graph that do not help a user to understand the core security problems and so can be trimmed, and (ii) automatically group similar attack steps as virtual nodes in a model of the network topology, to increase the understandability of the data.

In [25], the authors proposed a static analysis approach where attack trees are automatically inferred from a process algebraic specification. In their algebraic specification, they identify an attack as a set of channels that an adversary has to know in order to attain a given location in the system.

In [26] was presented P^{2}CySeMoL (Predictive, Probabilistic Cyber Security Modelling Language), an attack graph tool that can be used to estimate the likelihood that professional penetration testers are able to accomplish different attacks on enterprise architectures within time(s) designated by a user. The proposed tool automatically generates attack graphs by CySeMoL specification. P^{2}CySeMoL as well as CySeMol combine attack graphs and system models through the use of a language that is not flexible. In order to introduce flexibility, the Meta Attack Language (MAL) was introduced in [30] that is a domain specific language for probabilistic threat modelling and attack simulations. Starting from the MAL language, powerLang [31] was proposed to create and evaluate a MAL-based domain-specific languages for the representation and simulation of cyber-attacks for the power domain (i.e., power grids, energy providers, and other critical infrastructure).

Finally, in [27] the attack graphs are generated using an ontology and SWIRL (Semantic Web Rule Language) rules to express cause-consequence relationships of all known attack scenarios.

The main difference between the existing approaches with our work is that these studies do not consider simultaneously the physical, cyber, and human aspects. Moreover, in the approaches proposed by Mulval, P^{2}CySeML and MAL a user or an item has only a static role. Furthermore, no relationships between the system's components are considered, as well as aspects like monitoring. Except for the works based on MulVAL no logical inference is applied to the attack graph analysis.

7 Conclusion

In this work, we propose a novel hybrid threat model and analysis that permits us to describe and derive the security state of smart systems. Our approach includes in its reasoning the cyber, physical, and human aspects as well as relations between them and the relationships between the architectural components of the system. To the best of our knowledge, this is the first threat model that describes and analyses threats for smart hybrid systems, where the system's components can be of cyber, physical, and/or human nature.

The novel threat model introduced is able to represent the various relations between the components of the system, their security properties, and their relations with the possible threats. We analyse the different aspects and properties of the system's components that are not simply considered as possible sources of threat, but also considering their defensive and preventive capabilities. An important aspect is the analysis of the human aspects of some components both in terms of their vulnerabilities but also in their role of protecting the physical and digital aspects of the system.

Our model and analysis are implemented in a Prolog based tool called TAMELESS. We tested TAMELESS in several scenarios of which two smart systems and a critical infrastructure attack example were presented in this paper. We provided in detail the threat analysis steps performed by our threat model for one of the scenarios and an overview for the other two. Our tool automatically generates attack graphs that can be used by the security analyst/system administrator to understand the current state of the system and its components. In particular, the threats and vulnerabilities of the system and system's components are identified. The result of the tool can be used also to identify the most-effective countermeasures, as the user given the current state of the system can decide to make some appropriate changes to it. Before making the changes in the real system, the user can simulate the state of the system with the changes using TAMELESS and decide if they are appropriate or not.

An interesting direction for future work is the automatic extraction of the relations and security properties of the system's components. While in this work, we provided them to the threat model, many could be extracted automatically from system designs, physical plans, Building Information Models etc.. We see the smart building scenario as a good starting point, as the increased use of Building Information Modelling readily makes available not only the physical plans of the building but also its interconnections with the digital aspects of the system and the building management systems. Industrial systems such as in Industry 4.0 or industrial processes are also good candidates where the design documents include much of the information needed to automatically extract the model.

The relations, security properties, and rules used by our threat model are static. In the future, we also aim to investigate how such techniques can be applied in dynamic systems where new components can join or leave the system. Finally, we aim to enrich the expressivity and reasoning capabilities of our model by adding time constraints and probabilistic assumptions on the relations and rules, in order to model more complex systems, such as resilient components or devices able to autorepair.

References

[1] S. K. Khaitan and J. D. McCalley, "Design techniques and applications of cyberphysical systems: A survey," *IEEE Systems Journal*, vol. 9, no. 2, pp. 350–365, 2015.

[2] Verizon, "Data Breach Investigations Report," 2019.

[3] J. Staggs, D. F. Ferraiolo, and S. Shenoi, "Wind farm security: attack surface, targets, scenarios and mitigation," *IJCIP*, vol. 17, pp. 3–14, 2017.

[4] D. Sgandurra and E. Lupu, "Evolution of attacks, threat models, and solutions for virtualized systems," *ACM Comput. Surv.*, vol. 48, no. 3, pp. 46:1–46:38, Feb. 2016. [Online]. Available: http://doi.acm.org/10.1145/2856126

[5] X. Ou, S. Govindavajhala, and A. W. Appel, "Mulval: A logic-based network security analyzer." in *USENIX security symposium*, vol. 8. Baltimore, MD, 2005, pp. 113–128.

[6] D. Sgandurra, E. Karafili, and E. Lupu, "Formalizing threat models for virtualized systems," in *Data and Applications Security and Privacy XXX*, S. Ranise and V. Swarup, Eds. Cham: Springer International Publishing, 2016, pp. 251–267.

[7] H. Holm, T. Sommestad, M. Ekstedt, and L. Nordström, "Cysemol: A tool for cyber security analysis of enterprises," in *22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013)*, June 2013, pp. 1–4.

[8] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 336–345. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180446

[9] L. Lemaire, J. Vossaert, J. Jansen, and V. Naessens, "Extracting vulnerabilities in industrial control systems using a knowledge-based system," in *Proceedings of the 3rd International Symposium for ICS & SCADA Cyber Security Research*, ser. ICS-CSR '15. Swindon, UK: BCS Learning & Development Ltd., 2015, pp. 1–10. [Online]. Available: https://doi.org/10.14236/ewic/ICS2015.1

[10] Y. Mathov, N. Agmon, A. Shabtai, R. Puzis, N. O. Tippenhauer, and Y. Elovici, "Challenges for security assessment of enterprises in the iot era," *arXiv preprint arXiv:1906.10922*, 2019.

[11] ENISA, "ATM cash-out attacks," https://www.enisa.europa.eu/publications/info-notes/atm-cash-out-attacks, 2018.

[12] T. J. Horan, "Double-Digit ATM Compromise Growth Continues in US," https://www.fico.com/blogs/double-digit-atm-compromise-growth-continues-us, 2017.

[13] EAST, "ATM Physical Attacks in Europe on the increase," https://www.association-secure-transactions.eu/atm-physical-attacks-in-europe-on-the-increase/, 2019.

[14] N. Scaife, C. Peeters, and P. Traynor, "Fear the reaper: Characterization and fast detection of card skimmers," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, 2018, pp. 1–14.

[15] J. Staggs, "Adventures in attacking wind farm control networks," 2017.

[16] T. Swift and D. s. Warren, "Xsb: Extending prolog with tabled logic programming," *Theory Pract. Log. Program.*, vol. 12, no. 1-2, pp. 157–187, Jan. 2012.

[17] H. S. Lallie, K. Debattista, and J. Bal, "A review of attack graph and attack tree visual syntax in cyber security," *Computer Science Review*, vol. 35, p. 100219, 2020.

[18] L. Muñoz-González, D. Sgandurra, M. Barrère, and E. C. Lupu, "Exact inference techniques for the analysis of bayesian attack graphs," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 2, pp. 231–244, March 2019.

[19] W. Xiong and R. Lagerström, "Threat modeling – a systematic literature review," *Computers & Security*, vol. 84, pp. 53–69, 2019.

[20] C. Tsigkanos, L. Pasquale, C. Ghezzi, and B. Nuseibeh, "Ariadne: Topology aware adaptive security for cyber-physical systems," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2, May 2015.

[21] ——, "On the interplay between cyber and physical spaces for adaptive security," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 466–480, May 2018.

[22] E. A. Oladimeji, "Security threat modeling and analysis: A goal-oriented approach," 2006.

[23] R. Akella, H. Tang, and B. M. McMillin, "Analysis of information flow security in cyber-physical systems," *International Journal of Critical Infrastructure Protection*, vol. 3, no. 3, pp. 157 – 173, 2010.

[24] J. Homer, A. Varikuti, X. Ou, and M. A. McQueen, "Improving attack graph visualization through data reduction and attack grouping," in *Visualization for Computer Security*, J. R. Goodall, G. Conti, and K.-L. Ma, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 68–79.

[25] R. Vigo, F. Nielson, and H. R. Nielson, "Automated generation of attack trees," in *2014 IEEE 27th Computer Security Foundations Symposium*, July 2014, pp. 337–350.

[26] H. Holm, K. Shahzad, M. Buschle, and M. Ekstedt, "P^{2}CySeMoL: Predictive, probabilistic cyber security modeling language," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 6, pp. 626–639, Nov 2015.

[27] S. Wu, Y. Zhang, and X. Chen, "Security assessment of dynamic networks with an approach of integrating semantic reasoning and attack graphs," in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, Dec 2018, pp. 1166–1174.

[28] E. Bacic, M. Froh, and G. Henderson, "Mulval extensions for dynamic asset protection," Cinnabar Networks INC Ottawa (Ontario), Tech. Rep., 2006.

[29] C. Wang, K. Li, Y. Tian, and X. He, "Network risk assessment based on improved mulval framework and hmm," in *Security and Privacy in New Computing Environments*, J. Li, Z. Liu, and H. Peng, Eds. Cham: Springer International Publishing, 2019, pp. 298–307.

[30] P. Johnson, R. Lagerström, and M. Ekstedt, "A meta language for threat modeling and attack simulations," ser. ARES 2018. New York, NY, USA: ACM, 2018. [Online]. Available: https://doi.org/10.1145/3230833.3232799

[31] S. Hacks, S. Katsikeas, E. Ling, R. Lagerström, and M. Ekstedt, "powerLang: a robabilistic attack simulation language for the power domain," *Energy Informatics*, vol. 3, no. 1, pp. 1–17, 2020.