Model-Based Threat Modeling for Cyber-Physical Systems: A Computer-Aided Approach

Monika Maidl[1](✉), Gerhard Münz[1], Stefan Seltzsam[1], Marvin Wagner[2], Roman Wirtz[2], and Maritta Heisel[2]

[1] Siemens AG, Otto-Hahn-Ring 6, 81739 Munich, Germany
{monika.maidl,muenz.gerhard,stefan.seltzsam}@siemens.com

[2] University of Duisburg-Essen, Duisburg, Germany
{marvin.wagner,roman.wirtz,maritta.heisel}@uni-due.de

Abstract. Harming the security of a Cyber-Physical System (CPS) can lead to substantial damage and endanger for life because such a system includes many devices that interact with the physical world. Following the principle of security-by-design, the consideration of security should take place as early as possible during software development. However, the current state of the art often lacks systematic documentation of possible threats, and the identification of all relevant threats is not a trivial task.

In previous work, we presented a taxonomy of relevant attack actions for CPSs. The distinguishing feature of the taxonomy is its two-dimensional structure. We map typical attack actions to the attack surface. The attack surface is described by the component's interfaces which can be misused by attackers to gain access to a component, thus potentially harming the security of the system. On top of this taxonomy, we described an example of an attack action catalog. The application of our taxonomy and the attack action catalog still requires manual effort from practitioners, e.g. when looking up relevant attack actions.

Therefore, we developed a tool based on our taxonomy which we present in the present paper. In a first step, we formalized our taxonomy in form of a metamodel. Each threat model is an instance of that metamodel. The metamodel reflects the way in which the taxonomy links attack actions with parts of the system. Furthermore, we created a graphical editor that assists practitioners in creating the threat model. Based on the taxonomy's metamodel and attack action catalogs, the tool pre-filters relevant attack actions and allows to systematically document them in the threat model. Our tool provides different views on the threat model, thus helping to focus on the relevant aspects for a specific task.

Keywords: Security threats · Threat modeling · Attack actions · Taxonomy · Catalog · Tool-support

1 Introduction

Cyber-Physical Systems (CPSs) are running in many places, especially in critical infrastructures. These systems interact with the physical world, e.g. by monitoring values measured with sensors. The recent development in the context of the Internet-of-Things leads to increasing use of CPSs, also into private homes. Usually, a CPS is composed of different components that communicate with each other via interfaces. Due to their connected nature and their dissemination, CPSs are often subject to attacks. Therefore, it is essential to design critical systems with adequate security measures in place, following security standards like IEC 62443 [9].

The term *threat modeling* summarizes methods that deal with identifying and documenting incidents that may have an impact on the system's security.

Shostack [17] defines the term as follows: "Threat modeling is the use of abstractions to aid in thinking about risks." Uzunov and Fernández [20] give an alternative definition: "Threat modeling is a process that can be used to analyze potential attacks or threats, and can also be supported by threat libraries or attack taxonomies."

The knowledge about such threats can be captured in so-called threat libraries which exist for many types of systems. Security engineers can look up relevant threats and document them for their concrete system. However, these libraries often do not follow a common structure for the different threat descriptions.

A well-known taxonomy for security is STRIDE [11]. STRIDE provides six categories for typical threat categories. The disadvantage of the taxonomy is its generic nature, i.e. the categories are not mapped to specific elements of the system. Therefore, the application of STRIDE requires high expertise from security engineers.

In previous work, we presented a two-dimensional taxonomy that addresses the disadvantage of STRIDE by mapping typical attack actions to the attack surface [13]. The first dimension of this taxonomy is similar to STRIDE since it denotes The new approach is to combine this attack action dimension with a second dimension: The second dimension, which we call the attack surface dimension, consists of the system elements that constitute the attack surface of the system. It is described by the elements that allow some interaction, i.e. the component's interfaces. Attackers may perform malicious actions at these points, thus leading to harm to the system's security. The taxonomy allows creating catalogs of typical attack actions to CPSs. Practitioners can use these catalogs to look up relevant attack actions for the system they analyze.

Although our taxonomy and the attack action catalogs assist practitioners in identifying relevant threats, manual effort is still required, e.g. when documenting the identified threats. To support practitioners in systematically identifying and documenting relevant threats to the system, we developed a tool based on our taxonomy following a model-based approach. We first formalize the taxonomy and its dependencies to the system model by developing a metamodel. For the attack surface, we make use of our metamodel for CPS which we presented in

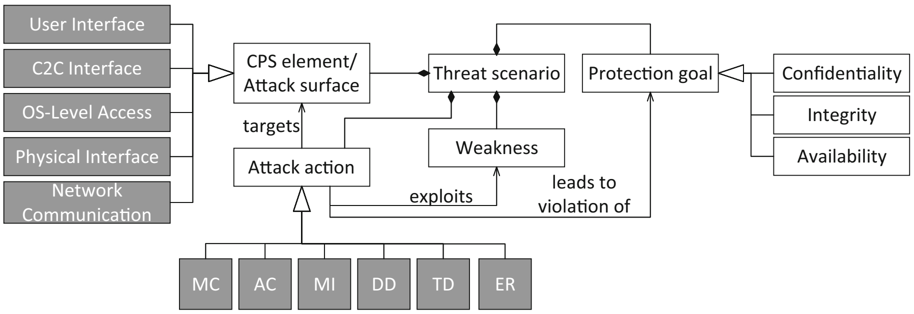


Fig. 1. Threat modeling terminology [13].

previous work [14]. Our metamodel includes the elements of a threat model, i.e. a systematic listing of threats to a system. The focus of our metamodel is to show the dependencies between the aspects of threats and different elements of the system, thereby reflecting the relation between the two dimensions of the taxonomy. To instantiate the metamodel, i.e. to create a threat model for a previously modelled system, we provide a graphical editor. The editor allows to systematically analyze the system, and it pre-filters relevant attack actions based on the provided interfaces. Furthermore, it is possible to embed attack action catalogs into the editor, thus ensuring flexibility for different development projects. To help practitioners in focusing on relevant aspects, our editor provides different views on the model.

The remainder of the paper is structured as follows: In Sect. 2, we describe the terminology which we use in this paper. In Sect. 3, we describe the taxonomy for which we present our tool support in this paper. It is followed by its formalization in form of our metamodel in Sect. 4. Section 5.1 explains how catalogs can be structured with our taxonomy, and we provide an example of a catalog. Our graphical editor to create a threat model is described in Sect. 6. We discuss related work in Sect. 7 and conclude the paper with a summary of our contributions and an outlook on future work in Sect. 8.

2 Terminology

The terminology in the context of threat modeling varies between different standards and publications. To have a common understanding for this paper, we provide an overview of the used terminology in Fig. 1.

The output of threat modeling is a list of *Threat scenarios*. Each threat scenario consists of the following elements: an *Attack action*, a *CPS element* as part of the *Attack surface*, a *Weakness*, and a *Protection goal*. As protection goals, we consider the CIA triad *Confidentiality*, *Integrity*, and *Availability*.

The threat scenario describes how an attack action leads to the violation of a protection goal by exploiting a weakness. The attack action targets an element of the attack surface of the CPS, i.e. an interface or network communication.

In some cases, a sequence of attack actions are required for the violation of protection goals, and these are described as part of the threat scenarios.

For illustration, we describe the example of a threat scenario: An attacker, pretending to be a legitimate device of the CPS, sends manipulated configuration (attack action) to an embedded component that is accessible via a C2C interface. As a result, the configuration of the control program is changed (violating the protection goal 'integrity of configuration'), and the embedded component behaves in an unintended way.

The goal of threat modeling is to consider all relevant attack actions against the CPS. To support this, we use categories of attack surface elements and attack action types. The elements marked in gray provides an overview of the categories which we use for our taxonomy (see Sect. 3). The attack surface denotes the first dimension, and the attack action types are the second dimension.

3 Two-Dimensional Taxonomy

In the following, we describe our two-dimensional taxonomy. We provide a two-dimensional taxonomy of attack actions for the scope of CPSs. Section 5.1 later exemplifies the usage of the taxonomy by describing a catalog of attack actions.

We first describe the two dimensions and show how we combine them for our taxonomy. Finally, we compare the taxonomy with existing ones, i.e. STRIDE [11] and CAPEC [15].

3.1 Attack Surface Dimension

The first dimension lists the parts of a system that form the attack surface, i.e. those points of a system at which an attack action may be performed. The *elements of the attack surface* depend on the type of system, and reflect the technical scope and level of detail typically considered in threat modeling. In this work, a CPS is viewed as a set of different types of components like embedded devices and hosts (workstations and servers) that are running standard operating systems and domain-specific applications and services. The components communicate through a combination of network protocols. In previous work, we proposed a metamodel for CPSs which is intended to be used as a basis of security analysis and specifies the elements of the attack surface of a CPS [14]. These elements form the attack surface dimension, and in the following, we explain them in detail.

The primary parts of an attack surface are the interfaces of the system components, as interfaces are the parts of the system that are open for interaction. Corresponding to the scope and level of detail considered in this paper, the various interfaces related to operating systems are covered by one abstract attack surface element, and the same holds for network communication.

User Interface. User interfaces are designed to let human users interact with the system. User interfaces can be realized in different ways, e.g. as a graphical user interface of an application running on the local computer, as a web-based user interface accessed over the network via a web browser, or as a human-machine interface realized with an embedded device. Apart from interfaces for regular users of the CPS, user interfaces for administration purposes need to be covered as well. User interfaces are usually associated with user accounts to implement user identification, authentication, and authorization.

Component-to-Component (C2C) Interface. These interfaces are similar to user interfaces but are designed to allow interaction between components instead of humans. Typically, an application running on a system component calls a service that runs on another component according to some protocol. C2C interfaces implement protocols and may include authentication and authorization. Typically, the protocol used by some C2C interface is utilizing standard network services that are implemented as part of the operating system. Interfaces (e.g. APIs) that exist internally in a component without being accessible by other components are not considered as C2C interfaces but considered as part of OS level access.

OS Level Access. There are various possible ways of how an attacker can interact with the operating system of a component. This includes local APIs and files, as well as the installation and modification of software, and network services that are implemented as part of the operating system. We use the element OS level access to represent the range of actual OS interfaces. This corresponds to the typical scope and level of detail of security analyses for CPSs, where the interfaces of the operating system are not modeled in all detail.

Physical Interface. These interfaces require physical access or physical proximity to the component to interact with the system. This is often relevant for CPSs with components that are widely deployed across sites. Included are interfaces used to communicate with the component, such as serial ports, USB port, local diagnosis or management interfaces, and near-field communication, e.g. Bluetooth. Other kinds of physical interactions are covered as well, such as manipulating the hardware and removing a hard drive.

Network Communication. User interfaces and C2C interfaces may involve network communication between different components of the CPS, using a protocol. Communication takes place over a potentially complex network infrastructure composed of network cables and network devices like routers and firewalls. We use Network Communication as an element of the attack surface that subsumes all possibilities to attack the communication between components of the CPS. An attacker could e.g. perform wiretapping at an accessible LAN port, or hack into a network device to disturb the communication. This abstraction corresponds to the typical scope and level of detail of the design of CPSs, which builds on an existing network infrastructure such as the Internet or production networks.

3.2 Attack Action Type Dimension

Attack actions are a central part of threat scenarios, as shown in Sect. 2, and describe the action an attacker takes at the attack surface of the system. Hence it is straightforward to use types of attack actions as a dimension of our taxonomy. Actual attack actions are often creative ways to interact with the system in an unintended way, and hence the known attack actions are very heterogeneous. Therefore it is not straightforward to find suitable types. We devised the following guiding principles for the development of our attack action types.

1. Focus on actions that an attacker performs at some location of the attack surface.
2. Strictly differentiate between attack actions and harm. As detailed in Sect. 3.5, after the identification of a relevant attack action for a CPS, it is a separate step to analyze whether a protection goal can be violated by that attack action.
3. Common attack actions should be assignable to one of the attack action types in a straightforward way. As a reference for common attack actions, we use the list compiled from industrial projects, as well as external sources [5]. Coverage of 'esoteric'attack actions has less priority.
4. Keep it simple: For good usability, the list of attack action types should not be too long, and easy to grasp.

As the next step, we considered existing taxonomies, in particular STRIDE and the taxonomy-level of CAPEC. To meet the guiding principles, we performed some adaptations. Section 3.4 contains a detailed comparison of the attack action types with the taxonomies of STRIDE and CAPEC, showing the adaptations.

The following list presents the attack action types, which form the attack action type dimension of our taxonomy. We argue for each case that the first two principles are fulfilled.

MC. Misuse credentials: Attacker obtains the authentication credentials for the account of a legitimate user and uses these to get access.
Note that this type covers all attack actions that relate to passwords, e.g. actions like obtaining passwords by social engineering, or guessing the password. Such attacks are very common indeed. Login interfaces are part of the attack surface. And as misuse of a password is not in itself harmful, the second principle is also observed.

AC. Exploit weakness of access control: Attacker circumvents or breaks access control and gets access.
This type covers the actions of attackers who are confronted with some form of access control. Access control is located at places where interaction with users or other components is expected, and hence the first principle is fulfilled. The second principle is observed by the same argument as for MC. One could argue that credentials are part of access control, but we decided to single out the misuse of credentials as a separate type, as AC is about exploiting (usually technical) weaknesses, while MC is about misusing legitimate credentials.

MI. Submit malicious input: Attacker enters or sends malicious data or commands.
This type comprises many common attack actions, in particular many actions against Web applications like SQL-injection. The first principle is fulfilled since interfaces that take input are open for interaction and hence are part of the attack surface. The second principle is fulfilled as it requires separate considerations to determine harm that might be caused by malicious input.

DD. Disclose data: Attacker reads or sniffs data.
This type comprises attack actions where an attacker can easily read data at the attack surface, e.g. by sniffing clear-text protocols. So the first principle is observed. Concerning the second principle, note that this type stands for various actions in which data is read at a place directly accessible to the attacker. Whether such reading results in harm, by violating the protection goal of confidentiality, is a different (although in this case fairly easy) consideration: Determining whether the data that can be read is sensitive.

TD. Tamper data: Attacker manipulates data.
This type is similar to the type DD. The difference is that this type covers attacks where data is manipulated at the attack surface.

ER. Exhaust resources: Attacker uses up limited, shared resources needed by the system.
This type covers attack actions that exploit the use of shared resources, e.g. CPU, memory, or network bandwidth. The attack surface for these actions is some form of access to the shared resource, e.g. the possibility to run applications on the operating system, or the possibility to send traffic in a network. So the first principle is fulfilled. Concerning the second principle, like in the two previous cases, it might be easy to determine the harm that follows from the exhaustion of a shared resource, but this attack action type focuses on the ways how to perform the exhaustion.

The example attack action catalog in Tables 4 and 5 shows that the third principle is met, by mapping a range of common attack actions to our attack action types.

3.3 Two-Dimensional Taxonomy

As the attack action types of Sect. 3.2 stand for attack actions at the attack surface, it is a natural step to relate the attack action types with the attack surface elements of Sect. 3.1. Table 1 shows the mapping, where the statements in each field express the relation. In most cases, the statements are straightforward, while some statements clarify the relevant aspects of the attack surface. Furthermore, some attack actions are not relevant for certain elements of the attack surface, resulting in empty fields in the table.

The two-dimensional taxonomy helps to systematically cover attack actions for the attack surface of a system.

We provide some explanations for the statements in the table: The attack action types DD and TD are considered for user and C2C interfaces. By design,

Table 1. Two-dimensional taxonomy [13].

	User interface	C2C interface	OS level access	Physical interface	Network comm.
MC	Attacker misuses credential to authenticate to the user interface	Attacker misuses credential to authenticate to the C2C-interface	Attacker misuses credential to obtain access to the operating system	Attacker misuses credential to obtain access to physical interface	
AC	Attacker exploits weakness in the access control of the user interface	Attacker exploits weakness in the access control of the C2C interface	Attacker exploits weakness in the access control of the operating system	Attacker exploits weakness in the access control of the physical interface	
MI	Attacker enters malicious input at the user interface	Attacker sends malicious input to the C2C interface	Attacker sends malicious input to some OS level interface	Attacker enters malicious input at the physical interface	
DD			Attacker reads data out of memory	Attacker reads data via physical interface	Attacker sniffs network communication
TD			Attacker manipulates data stored in memory	Attacker manipulates data via physical interface	Attacker manipulates network communication
ER			Attacker exhausts resources of the operating system		Attacker exhausts network resources

these interfaces display data and provide functionality for editing. Using this functionality is not an attack action. If the access to a user or C2C interface is meant to be restricted, then the attack action types MC and AC apply and cover possible ways an attacker can get access despite the access protection.

The last row of Table 1 shows that the attack action type ER is only considered for OS level access and network access. Only at these elements of the attack surface, an attacker has direct access to limited resources, like CPU, memory, or network bandwidth. In contrast, user interfaces, C2C interfaces, and physical interfaces do not provide direct access to resources. Malformed input to these interfaces that causes the receiving component to crash, e.g. due to overload, is covered by the type MI.

The column for OS level access reflects the fact that this element of the attack surface comprises various interfaces of the operating system. For MC, the user accounts of the operating system are in focus. The attack action type AC refers to the various access control mechanisms of the operating system, e.g. privilege of processes and file permissions. It comprises attacks to exploit weaknesses in these mechanisms, e.g. to obtain higher privileges. Malicious input (MI) can

Table 2. Mapping of taxonomy categories - STRIDE [13].

Category	Description	MC	AC	MI	DD	TD	ER
Spoofing of user identity	Impersonating something or someone else.	✓	✓				
Tampering with data	Modifying data or code					✓	
Repudiation	Denying to have performed an action						
Information disclosure	Exposing information to someone not authorized to see it				✓		
Denial of service	Deny or degrade service to users			✓			✓
Elevation of privilege	Gain capabilities without proper authorization		✓	✓			

take the form of malware that exploits vulnerabilities in the operating system. Malicious input may originate from a user with OS level access who is tricked into downloading and executing malware. Another path of malicious input is specially crafted packets sent to a network service of the operating system.

For network communication, as explained in Subsect. 3.1, the scope and level of detail applied in the design of a CPS usually does not include the network infrastructure. Hence, threat modeling for a CPS focuses on attack actions against the network communication between components. These attack actions are disclosing (DD), tampering (TD), and exhausting resource (ER). The attack action types MC, AC, and MI are not relevant as the network communication does not process credentials, does not implement access control, and does not handle inputs. These tasks are performed by the protocol stack of the corresponding user or C2C interface.

3.4 Comparison with Other Taxonomies

In a systematic literature review on threat analysis of software systems performed by Tuma et al. [19], five methodologies make use of some sort of knowledge base, are applicable to the architectural or design level, and take the architectural design as input. Three of them use STRIDE [7,8,17] as taxonomy, the remaining two refer to CAPEC [2,3]. As our taxonomy also provides a knowledge base and is supposed to be used in the same context of threat analysis, this section provides a detailed comparison with STRIDE and CAPEC.

STRIDE. STRIDE [11] is a well-known categorization model for threats against computer systems. It has been developed by Microsoft and is integrated in the

Microsoft Threat Modeling Tool[1]. STRIDE is a mnemonic for six threat categories: *Spoofing*, *Tampering*, *Repudiation*, *Information disclosure*, *Denial of service*, and *Elevation of privilege*.

We found that some of the STRIDE categories refer to the impact of a successful attack (e.g. denial of service) rather than to the actual action an attacker performs. To avoid confusion, our taxonomy clearly focuses on attack actions that describe what an attacker does. The impact of an attack action can be assessed in a subsequent step by determining the violated protection goals.

Table 2 shows how the STRIDE categories can be mapped to our attack action types. As can be seen, the STRIDE categories *Tampering* and *Information disclosure* are directly related to the attack action types TD and DD. *Spoofing* can be achieved by misusing credentials of existing accounts (MC), or by exploiting an access control weakness (AC). *Denial of service* is typically caused by malicious input (MI), such as a specially crafted packet leading to a segmentation fault, or by exhausting limited resources (ER), e.g. with a flooding attack. Malicious input (MI) as well as bypassing access control (AC) can lead to *Elevation of privilege*.

We did not map the STRIDE category *Repudiation* to any of our attack actions types. This is because we see repudiation as violation of a protection goal (i.e. non-repudiation), not an attack action. In fact, various attack actions can be used with the goal to repudiate an action, such as tampering log files. But our types focus on the action of the attacker rather than the goal of the action.

The main extension of our attack action types compared to STRIDE is the attack action type MI, which includes all kinds of injection attacks, such as SQL injection, code injection through exploitation of a buffer overflow vulnerability, infection of a system with malware etc. In STRIDE, these attacks do not have an explicit category but can only be categorized indirectly by the harm they cause (e.g. denial of service).

STRIDE itself does not include an attack surface dimension. The Microsoft Threat Modeling Tool allows us to associate STRIDE categories with elements of a Data Flow Diagram (DFD), which contains processes, data stores, external interactors, and data flows between them. However, the combination of STRIDE categories and DFD elements is not used to provide a better understanding of a STRIDE category for a DFD element. More importantly, DFDs do not reflect the different parts of the attack surface of a system. So the combination of STRIDE with DFDs lacks the possibility to create a catalog of relevant attack actions for each attack surface element, similar to the ones in Tables 4 and 5.

CAPEC. The Common Attack Pattern Enumeration and Classification (CAPEC) [15], maintained by MITRE[2], provides a catalog of attack patterns.

[1] Microsoft Threat Modeling Tool (last access: May 25, 2021): https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling.

[2] MITRE: https://www.mitre.org/ (last access: May 18, 2021).

Table 3. Mapping of taxonomy categories - CAPEC [13].

Mechanism of attack	**Description**	**MC**	**AC**	**MI**	**DD**	**TD**	**ER**
Engage in deceptive interactions	Spoofing and social engineering	✓	✓	✓			
Abuse existing functionality	Manipulation of data or system behavior by misusing system functionality		✓	✓		✓	✓
Manipulate data structures	Manipulation of data by exploiting a system vulnerability			✓		✓	
Manipulate system resources	Manipulation of shared resources			✓		✓	
Inject unexpected items	Manipulation of system behavior through malicious input			✓			
Employ probabilistic techniques	Fuzzing and bruteforcing		✓	✓			
Manipulate timing and state	Exploitation of concurrency issues (e.g. race condition)			✓		✓	
Collect and analyze information	Theft of information			✓	✓		
Subvert access control	Exploitation of access control weakness		✓				

In CAPEC version 3.2, attack patterns are classified according to two different schemes. The first scheme is called *domains of attack* and assigns attack patterns to the categories *Software*, *Hardware*, *Communications*, *Supply chain*, *Social engineering*, and *Physical security*. These categories refer to the type of weakness that is exploited, such as a software vulnerability, a weak physical control, or an unaware user. We found that CAPEC attack patterns in the domain *Communications* largely correspond to the attack actions associated to our attack surface element *Network Communication*. Similarly, most attack patterns belonging to *Hardware* and *Physical security* are related to the attack actions of the attack surface element *Physical Interface*. For the other domains, however, we did not find any clear correlation with the different elements of the attack surface.

The second CAPEC classification scheme is called *mechanisms of attack* and refers to general attacking techniques, which is similar to the attack action dimension of our taxonomy. Table 3 shows a mapping of our attack action types to CAPEC mechanisms of attack. Attack patterns belonging to the mechanism *Engage in deceptive interactions* range from attacks targeting user credentials and clickjacking to DLL injection and DNS spoofing. In our taxonomy, these attacks are separated into the attack action types MC, AC, and MI. Similarly, *Abuse of existing functionality* covers a broad spectrum of attack patterns that, in our taxonomy, belong to different attack action types. As can be seen, the attack mechanisms *Manipulate data structures*, *Manipulate system resources*, and *Manipulate timing and state* are related to the attack action types MI and TD. These two types distinguish between attacks sending malicious input to a system interface, and attacks tampering data (e.g. configuration files) directly, whereas the three CAPEC mechanisms differentiate between types of manipulated data and resources. The mechanism *Employ probabilistic techniques* includes password brute-forcing, which relates to the exploitation of an access control weakness (AC), and fuzzing attacks, which corresponds to sending potentially malicious input to an interface (MI). *Collect and analyze information* subsumes active and passive information gathering techniques, belonging to the attack action types MI and DD, respectively.

All in all, we can state that CAPEC's approach to classify attack patterns into mechanisms of attacks has some similarities to the attack action dimension of our taxonomy. The attack surface dimension of our taxonomy, however, is not reflected in CAPEC. Some CAPEC domains of attack are slightly related to specific attack surface elements, but in general, CAPEC domains of attack refer to types of exploited weaknesses. As a consequence, CAPEC lacks the possibility to easily query attack patterns that are relevant for a specific attack surface element of a CPS.

3.5 Using the Taxonomy for Threat Modeling

In the process of threat modeling, our taxonomy helps to obtain a list of threat scenarios as described in Sect. 2. The elements of the attack surface that need to be considered can be directly extracted from the design of the CPS. In the first step of threat modeling, for each of these elements and each relevant attack action type, attack actions are looked up from the catalog.

Once an attack action is found to be relevant, the next step is to analyze whether the attack action could lead to the violation of a protection goal of the CPS. This is an essential step of threat modeling, in which know-how about the architecture of the system is combined with a thorough understanding of the protection goals for the data and functionalities of the system. If a path to the violation of a protection goal has been found, a threat scenario is documented. The threat scenario is completed by describing the weakness of the CPS that is exploited by the attack action. Usually, the attack action is directly associated with a weakness, so this step is not challenging. For example, the infection with malware is exploiting unpatched vulnerabilities, while a brute force attack on a password is exploiting weak passwords. In fact, it would be a natural extension of an attack action catalog to link the attack actions to related weaknesses and hints for security measures. For example, enforcing a strong password policy is a security measure to protect against brute forcing.

After threat modeling has been completed, the weaknesses are used as a basis to select (additional) security measures for the CPS.

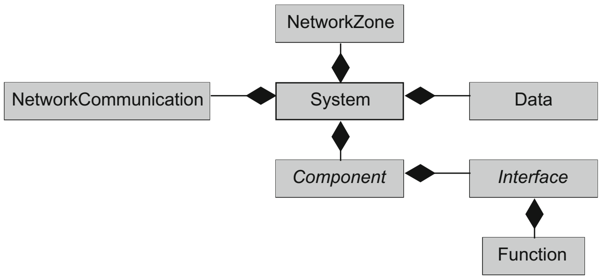


Fig. 2. Relevant artifacts of the system's metamodel.

We point out some aspects of using the taxonomy with an example: A component has several user interfaces and C2C interfaces. The relevant attack action types, namely MC, AC, and MI, are analyzed for each of these interfaces. This helps to identify weaknesses in the design of access control for these interfaces, and weaknesses in the processing of inputs. The component also has several physical interfaces, and the need to adequately protect each of them may have been overlooked during design. Going through the attack action types helps to identify the critical gaps. Furthermore, the component runs a standard operating system that needs to be securely configured and hardened. The attack action types allow the architect to understand which parts of the OS need particular protection, e.g. by encrypting files, disabling unneeded network services, or implementing other hardening measures. For each of the network communications of the CPS, the attack action types DD, TD, and ER are analyzed, and as a result, the architect might decide to use another protocol or a secure channel for a protocol.

4 Metamodel

As a first step towards tool support, we formalize our taxonomy and the relations. Based on the *Eclipse Modeling Framework (EMF)* [18], we create a metamodel for this purpose. The notation of metamodels EMF is similar to UML class diagrams. For better readability and space reasons, we have decomposed the metamodel into four sub-metamodels *System*, *ThreatModel*, *ThreatScenarioListing*, *ProtectionGoal*. The gray elements in the metamodels are taken from our CPS system model [14]. Classes with a cursive name are abstract classes. The focus of our metamodel is on the dependencies between the aspects of threats and the different elements of the system, thereby reflecting the relation between the two dimensions of the taxonomy as presented in Sect. 3.

4.1 CPS Metamodel

We use the metamodel for cyber-physical systems (CPS) from our previous work [14] as a starting point. It contains the elements of the attack surface, i.e. the interfaces and network communication. In the present paper, we focus on the

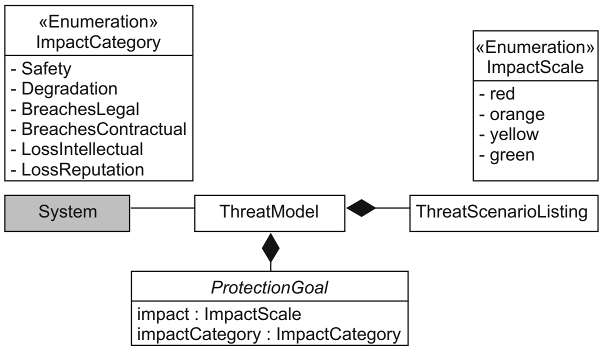


Fig. 3. Metamodel of *ThreatModel*.

crucial parts of the CPS metamodel that are important for the implementation of the taxonomy. Figure 2 shows this compact version. The *System* class is the root element of the sub-metamodel. It contains four other classes: (i) *Network-Zone* representing network zones in a CPS, (ii) *Component* which can be further refined to specific types, e.g. a host, (iii) *NetworkCommunication* representing the communication between two components, and (iv) *Data* (represents data which is processed in a CPS). A component has a set of *Interface*s. These interfaces offers some functionalities (class *Function*) to other components. Later on, we map attack actions to the different interface types (see Sect. 3.1). Our editor uses this information for filtering relevant attack actions (see Table 1).

4.2 Threat Model

The second sub-metamodel, which is shown in Fig. 3, shows how the other three sub-metamodels are linked. The root element is the *ThreatModel* which has an association to the system. This way, the threat model can make references to the elements of the system, and the automatic mapping of attack actions to the interfaces can be used. A threat model consists of *ThreatRiskAnalysis*s and *ProtectionGoal*s, both of which are further refined in the next sub-sections.

A protection goal has two attributes. First, there is the attribute *impact* of the type *ImpactScale*. This attribute expresses how severe a violation of the protection goal would be. We use an enumeration for the *ImpactScale*. This enumeration has four literals *red*, *orange*, *yellow*, and *green*. Red means a high impact and green means a low impact. Orange and yellow are between them. Orange means a higher impact than yellow.

The second attribute is *impactCategory* which has the type *ImpactCategory* in form of an enumeration. It has the six literals *Safety*, *Degradation*, *BreachesLegal*, *BreachesContractual*, *LossIntellectual*, and *LossReputation*. The impact category describes what kind the impact has to the system. We use these six categories because they are the most common ones.

4.3 Protection Goals

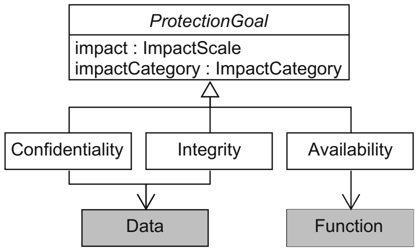


Fig. 4. Metamodel of *ProtectionGoal*.

Figure 4 shows the third sub-metamodel. The class *ProtectionGoal* is an abstract class. It has the three specializations *Confidentiality*, *Integrity*, and *Availability* which are equal to the CIA triad. Confidentiality and integrity have an association to some data processed by the System. This way, the metamodel provides the possibility to document which data shall be protected. The metamodel for CPS as presented in [14] includes the relation of data to components and network communications, i.e. where data is stored and transferred. Availability has an association to a function of the System. An interface of a component offers some functions (see Fig. 2). Thus, an availability goal denotes that the availability of a function shall be preserved.

4.4 Threat Scenario Listing

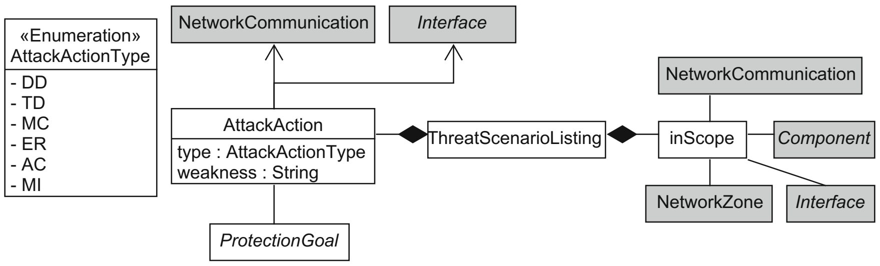


Fig. 5. Metamodel of *ThreatScenarioListing*.

Figure 5 shows the last sub-metamodel. The *ThreatScenarioListing* is part of a threat model. A threat model can have multiple threat scenario listings for different perspectives. Each analysis consists of *inScope*-classes and a set of *AttackActions*. *inScope* has the purpose to express that a part of a system (*NetworkCommunication*, *Component*, *Interface*, or *NetworkZone*) is in scope for that analysis. So, not each element has to be in scope, and it can be filtered what shall be analyzed. Attack actions are also part of a threat scenario listing. Each attack action has the attribute *type* which has the type *AttackActionType* which is an enumeration and has six literals DD, TD, MC, ER, AC, and MI. These are the abbreviations of attack actions as shown in Table 1. They form the second dimension of our taxonomy. Furthermore, there is the attribute *weakness* that documents the weakness exploited by the attack action. The associations from an attack action to a *NetworkCommunication* and an *Interface* are to express that an attack action is performed at this element of the attack surface. They form the other dimension of the taxonomy. Between an attack action and a protection goal, there is another association to express that this action harms the protection goal.

5 Attack Action Catalog

In the following, we describe how attack action catalogs can be structured with the help of our taxonomy which we presented in Sect. 3. Furthermore, we present an example of such a catalog.

5.1 Structuring Attack Action Catalogs with the Taxonomy

While Table 1 helps to focus on relevant attack action types for a certain interface, architects and software developers find it hard to identify specific attack actions based on abstract attack action types: They need an understanding of actual attack actions rather than abstract categories. There are many threat and attack catalogs that contain actual attack actions, but it is hard to find relevant entries, especially for people without a deep security background.

We propose to use our two-dimensional taxonomy to structure catalogs of specific attack actions. This means that each specific attack action is assigned to an attack action type and an element of the attack surface. In that way, the (typically large) set of attack action is clustered into 20 subsets in a way that is meaningful for threat modeling. Practitioners can find the relevant attack actions efficiently by looking into the appropriate field of the structured catalog. Hence the catalog provides a useful way to make security knowledge about attacks available during threat modeling.

5.2 Example Catalog

In Tables 4 and 5, we provide an example catalog of attack actions against CPSs, structured according to our taxonomy. The catalog captures the range of attacks that have been considered in security analyzes for CPSs over many years in industrial projects, and also reflects the results of penetration tests and real world incidents. Besides, the catalog was compared and extended with external resources, e.g. from the *Bundesamt für Sicherheit in der Informationsbranche* [5], as well as academic sources like [20]. The catalog is not aiming for completeness.

Instead, the aim is to cover the most relevant cases and include attacks that exploit typical weaknesses in standard IT technology. To cover attacks that are specific to domain-related technology (e.g. embedded devices, sensors) or attacks to specific components like network devices, the catalog can be augmented.

In Table 4, we show the mapping of the attack actions *Misuse Credentials (MC)*, *Exploit Weakness of Access Control (AC)*, and *Submit Malicious Input (MI)*. For each of the attack actions, we provide examples in the context of a specific interface type or a network communication. An empty cell denotes that the attack action is not relevant for the corresponding attack surface element.

In Table 5, we show the second part of the example catalog. It contains the attack actions *Disclose Data (DD)*, *Tamper Data (TD)*, and *Exhaust Resources (ER)*.

Attack actions and their relevance are changing over time, so it is important to emphasize that such a catalog has to be continuously updated. Furthermore, it is possible to use the taxonomy to create a catalog for a specific context, e.g. for critical infrastructures. To do so, the examples can be refined with more details.

5.3 Further Benefits

The example catalog of Sect. 5.1 illustrates the structuring of attack actions according to the two-dimensional taxonomy. In this section, we discuss further ways to use taxonomy-based catalogs in the context of threat modeling.

Specific Catalogs for Types of Components. A CPS consists of heterogeneous components like controllers, network devices, and standard IT components. By providing a separate catalog for each type of component, attack actions that are specific to the technologies of that component type can be listed and provided to practitioners. Such catalogs could either complete or replace a generic catalog.

Reusing Threat Modeling Results. In practice, often a certain type of CPS is used as a blueprint for industrial projects. After performing threat modeling for that type of CPS, the knowledge generated by that process of threat modeling can be captured in the form of a specific attack action catalog. More precisely, the entries in the generic catalog(s) can be replaced by more specific and relevant attacks for the blueprint. In that way, it is possible to make knowledge reusable for future projects.

Using Catalogs in Tooling for Threat Modeling. The benefits of taxonomy-based catalogs are significantly increased by automation: We have developed a prototype for a tool, and are in the piloting phase. Our tool guides practitioners through the process of threat modeling and presents relevant attack action types when the practitioner is working in a certain part of the system.

Table 4. Attack action catalog for CPSs Part 1 [13].

	User Interface	C2C Interface	OS Level Access	Physical Interface	Network Comm.
MC	– Phishing to obtain a user's password – Brute force attack on weak password – Setting password through weak password recovery mechanism	– Extract default or hard-coded passwords – Brute force attack on weak passwords – Misuse fake MAC or IP address to authenticate	– Misuse of temporary or default password – Brute-force attack to guess password of OS account – Misuse of shared password (e.g. shared between sites)	– Social engineering to obtain password to server management consoles or BIOS	
AC	– Misuse of client-side authentication or authorization – Access via debugging interface – Misuse of direct object references e.g. in URLs – Session hijacking	– Misuse of client-side authentication or authorization – Security downgrade through algorithm negotiation – Misuse of excessively granted privileges	– Bypass of kiosk mode – Misuse of open network service (e.g. Telnet, VNC) – Misuse of (unnecessarily) high privileges in OS – Misuse of unlocked user session	– Access through server management consoles or BIOS – Re-boot with different OS from CD or USB – Access through unprotected near-field communication protocol – Misuse of hardware interfaces (UART, JTAG) – Misuse of shut-down button	
MI	– Cross-site scripting – SQL-injection – Malware infection of component through malicious payload	– Fuzzing attack – Malware infection of component through malicious payload – Crash due to overload	– Trick OS-user to install or run malware – Network packet exploiting vulnerability in network protocol implementation of the OS, e.g. ping-of-death	– Malware infection of component through infected USB stick	

Table 5. Attack action catalog for CPSs Part 2 [13].

	User Interface	C2C Interface	OS Level Access	Physical Interface	Network Comm.
DD			– Read sensitive data from files or Windows registry, e.g. passwords, operational data – Read data from process memory by causing a core dump	– Steal media, i.e. SD card, USB stick, or hard disk – Install keylogger – Take a covert look at a display – Read data through hardware interfaces (UART, JTAG)	– Read clear text protocols, e.g. HTTP, FTP – Sniff data sent over unprotected WLAN
TD			– Manipulate data in files or databases – Manipulate configuration or software	– Change data on removable media	– Manipulate or replay message – Man-in-the-middle attack
ER			– (Malicious) application uses up CPU or memory		– Flooding the network – Occupy wireless interfaces with a jammer

6 Tool-Support

We use a model-based approach for our tool based on two frameworks. The first framework is EMF [18] (see Sect. 4) and the second framework is called Sirius[3].

6.1 Sirius

Sirius allows you to create your own eclipse graphical modeling workbench and diagrams. It builds on EMF and the *Acceleo Query Language (AQL)*[4]. AQL is a specification language similar to the *Object Constraint Language (OCL)*[5]. AQL expressions are used to interact with the model, e.g. to manipulate model instances or to query data from the model. Sirius has some built-in graphical elements, for example text fields, nodes, and containers. Diagrams can be specified in a hierarchical tree structure. The diagrams provide a user-friendly view of model instances.

6.2 Workflow of Our Tool

The metamodels of Sect. 4 provide the fundamentals for the Sirius editors. We implemented two editors for different purposes, each having several diagrams. The first editor enables the user to model a CPS with different components, interfaces, communications, and data. That editor builds on our previous work [14]. Therefore, we do not further discuss it in this paper. However, the CPS editor is essential for the second editor, which uses our taxonomy from Sect. 3. We provide a simple example to show how our editor works. Figure 6 shows a CPS which is modeled with our editor. There is the component *Host 1* with a user interface (small box), which can be accessed remotely from a *User Browser* via a network communication with the *https* protocol.

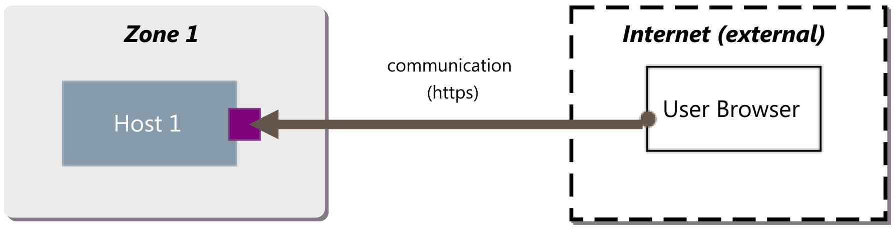


Fig. 6. Example of a CPS.

[3] https://www.eclipse.org/sirius/.
[4] https://www.eclipse.org/acceleo/documentation/.
[5] https://www.omg.org/spec/OCL/2.4/PDF.

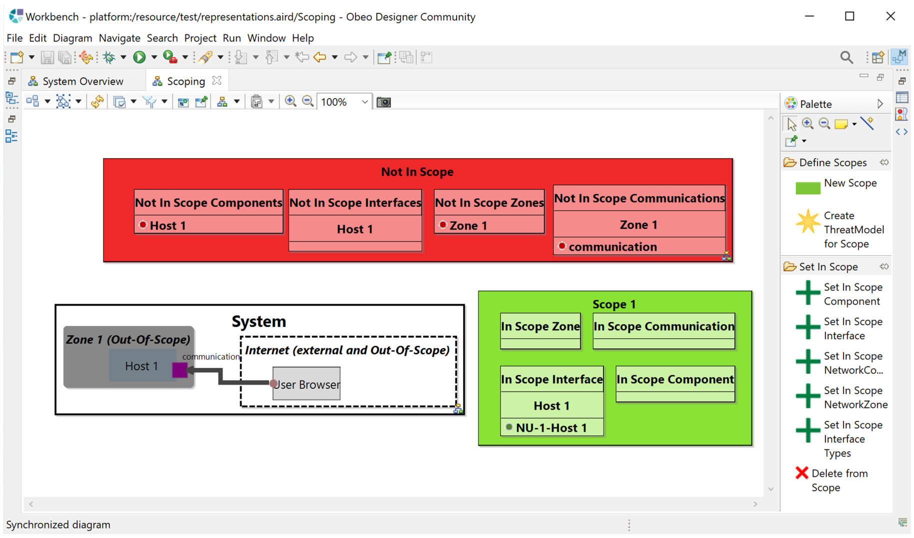


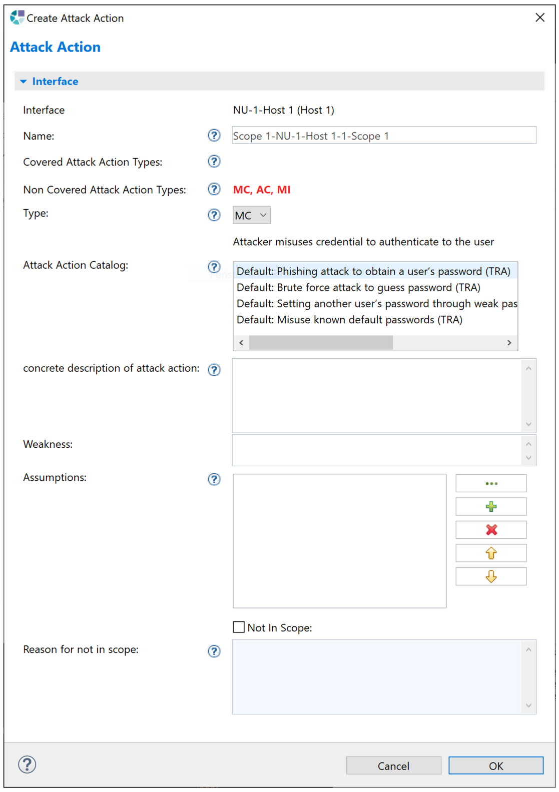
Fig. 7. Scoping.

Scoping. The first step is shown in Fig. 7. Its purpose is to set the focus for the threat analysis. The user can obtain some information from that diagram. First, there is a graphical representation of the CPS which is extracted from the CPS model. The box entitled *Not in scope* denotes the CPS elements that are not in scope. The box *Scope 1* contains the elements that are in scope for the analysis. It is possible to define different scopes within a model, each of them focusing on different elements.

The palette on the right side provides different tools for users to create or manipulate the model. The graphical representation and the *Not in Scope* container have only an informative use, i.e. they only extract some information from the model and present it to the user.

A new scope, e.g. *Scope 1*, can be created with the tool *New Scope*. Each Scope is an instance of the class *ThreatScenarioListing* from Fig. 5. Users have multiple dialogs to set elements of the system in scope. For example, when selecting an interface to be in scope, they get a list of all available interfaces. Via checkboxes, it is possible to select the desired ones. To document that an element is in scope, we instantiate the class *inScope* of the metamodel (see Fig. 5). Furthermore, the corresponding element is shown in the green box of the scope.

Analysis. The second step is the core part of threat modeling, i.e. the identification and documentation of the attack actions. This step is supported by our taxonomy from Sect. 3. For all interfaces that are specfied to be in scope, the user is guided to identify and document attack actions with the dialog shown in Fig. 8. For each documented attack action, a class *AttackAction* from Fig. 5 will be instantiated. Our tool filters relevant attack actions according to the



Create Attack Action

Attack Action

Interface

Interface: NU-1-Host 1 (Host 1)

Name: Scope 1-NU-1-Host 1-1-Scope 1

Covered Attack Action Types:

Non Covered Attack Action Types: MC, AC, MI

Type: MC

Attacker misuses credential to authenticate to the user

Attack Action Catalog:
Default: Phishing attack to obtain a user's password (TRA)
Default: Brute force attack to guess password (TRA)
Default: Setting another user's password through weak pas
Default: Misuse known default passwords (TRA)

concrete description of attack action:

Weakness:

Assumptions:

Not In Scope:

Reason for not in scope:

Cancel OK

Fig. 8. Dialog to document attack actions.

taxonomy, and presents the examples contained in the catalog to the user. Furthermore, the user dialog informes the user about which attack action types have already covered for the given interface, thereby showing the progress of coverage. Using the provided text fields, users can describe the attack action more precisely, give a weakness, and assign assumptions, Alternatively, if there are reasons why an attack action type is not in scope for that interface, users document that reason in the dialog.

To filter relevant attack actions, we use following AQL expression as shown Listing 1. It realizes the mapping of our taxonomy given Tables 2 and 3.

```
if (element.oclIsTypeOf(system::C2CInterface)
  or element.oclIsTypeOf(system::NetworkUserInterface)
  or element.oclIsTypeOf(system::LocalUserInterface))
then
  threats::AttackActionType.eLiterals→select(a|a.name='MC' or
       a.name='AC' or a.name='MI')
else if (element.oclIsTypeOf(system::PhysicalInterface))
then
  threats::AttackActionType.eLiterals→select(a|a.name='AC' or
       a.name='DD' or a.name='MI' or a.name='TD')
else if (element.oclIsTypeOf(system::OSLevelInterface))
```

```
then
  threats::AttackActionType.eLiterals→select(a|a.name='MC' or
      a.name='TD' or a.name='DD' or a.name='ER' or a.name='MP' or
      a.name='AC')
else
  threats::AttackActionType.eLiterals→select(a|a.name='AC' or
      a.name='MI' or a.name='TD' or a.name='DD' or a.name='ER')
endif endif endif
```

Listing 1. AQL expression for filtering relevant attack action types.

element is the CPS's element for which attack actions shall be identified, i.e. an interface or network communication. With the expression, we check which type *element* is and select the corresponding attack action types for it. Afterward, users can choose one of them and document it in the model.

7 Related Work

Almorsy et al. [2] introduced a new architecture software security analysis. They use OCL to formalize system architectural security attack scenarios and security metrics. Since our approach is model-based (cf. Sect. 3.1), our proposed taxonomy can be formalized in a similar way.

The paper by Halkidis et al. [8] evaluates the protection that selected security patterns of Blakley and Heath [4] offer against attacks. As attack categories, the authors make use of STRIDE. The analyzed system is annotated with stereotypes in order to check whether security patterns have been used sufficiently. This approach of using stereotypes can be compared with our interface types, e.g. there is a stereotype *ApplicationEntryPoint* that corresponds to our user interface. The difference to our taxonomy is that the annotations are not associated with attack actions, but are associated with security patterns.

Uzunov and Fernández [20] introduce system elements (called decomposition layers) to describe threat patterns. The system elements are similar to our attack surface elements, e.g. the decomposition layer 'User interaction'corresponds to a user interface. In contrast to our work, the authors do not use the system elements for structuring the threat patterns.

CAPEC (cf. Sect. 3.4) is often used as a comprehensive repository for attack descriptions rather than as a taxonomy. An example is Adams et al. [1], where CAPEC is used as source to identify relevant attacks, by using machine learning and natural language processing. Another example is the approach of [12] to leverage the CAPEC repository for finding relevant attacks, based on problem patterns, solution patterns, and context patterns.

Xiong and Lagerström performed a literature review on threat modeling [21]. This literature review lists many papers on threat modeling approaches that are based on (semi-)formal methods for representing threats, like game theory, Petri nets, Dolev-Yao threat model, PrT nets, Hidden Markov models, Byzantine model, flow model, and others. The usage of taxonomies in these approaches is different to our use. The taxonomy does not represent threats, but provides a structure for knowledge databases. Other papers covered in that literature review describe the use of threat modeling in a specific domain.

There are numerous risk management processes, e.g. CORAS [6], that require a detailed identification of threat scenarios. CORAS has its own modeling language and provides guidelines on how the method can be carried out. The method is model-based and has tool-support. The identification of threat scenarios is often performed in brainstorming sessions which does not necessarily follow a systematic procedure. Our taxonomy can be used as an input for those sessions to create CORAS diagrams.

Shevchenko et al. [16] evaluates methods for threat modeling of cyber-physical systems. They list twelve methods and rate them according to 5 criteria. The usage of an attack action catalog is no criteria. Some of the methods can be enhanced with an attack action catalog.

Khan et al. [10] apply STRIDE-based threat modeling to cyber-physical systems and apply their adapted method on a real world example. They state 10 possible threat consequences (TC) as an example. The authors use data flow diagrams (DFD) to model a cyber-physical system and link the DFD elements to TCs. The method of the paper is on a high level and our taxonomy can be applied after their method.

Currently, our taxonomy only allows us to analyze a system with regard to security. The LINDDUN methodology of Deng et al. [7] introduces privacy threat categories which have been derived from STRIDE. The relation of STRIDE to privacy may help to transfer our taxonomy into the privacy context, as well.

8 Conclusion

After having presented a two-dimensional taxonomy in previous work, we presented our tool support in the present paper. We first presented a metamodel that formalizes the taxonomy and the dependencies to the system model's elements, i.e. the attack surface. Based on that metamodel, we developed a graphical editor that filters relevant attack actions and that allows documenting the threat model systematically. To provide flexibility, the tool provides functionalities to import different attack action catalog. The current piloting of the tool and the taxonomy in the industrial context promises good results. Further feedback from practitioners will continuously be integrated into the tool.

Currently, our taxonomy, the attack action catalogs, and the tool are limited to the domain of CPS. The transfer to other domains requires us to adapt the first dimension, i.e. to define the elements of attack surfaces in other domains. Currently, we are working on system models for cloud-based systems which shall be used as input for security analyses, too. The new attack surface can be derived from that model, which makes it easy to adopt our taxonomy for cloud-based systems. Concerning attack action types, first experiments have shown that the types used in this paper are also suitable in the domain of cloud applications.

Another important aspect will be the consideration of other software qualities such as privacy. LINDDDUN [7] will be a good starting point for this adoption since it brings STRIDE to the context of privacy. Another important topic would be to identify overlaps between different qualities. For example, the protection goal of confidentiality is relevant for both security and privacy. The same countermeasures can therefore be applied to improve both qualities.

About our tool, we plan to add more catalogs to it and to make these catalogs publicly available via the Internet. Other practitioners and research may contribute to this resource with their own catalogs.

References

1. Adams, S.C., Carter, B.T., Fleming, C.H., Beling, P.A.: Selecting system specific cybersecurity attack patterns using topic modeling. In: 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE 2018, New York, NY, USA, 1–3 August 2018, pp. 490–497 (2018). https://doi.org/10.1109/TrustCom/BigDataSE.2018.00076
2. Almorsy, M., Grundy, J., Ibrahim, A.S.: Automated software architecture security risk analysis using formalized signatures. In: 35th International Conference on Software Engineering, ICSE 2013, San Francisco, CA, USA, 18–26 May 2013, pp. 662–671 (2013). https://doi.org/10.1109/ICSE.2013.6606612
3. Berger, B.J., Sohr, K., Koschke, R.: Automatically extracting threats from extended data flow diagrams. In: Caballero, J., Bodden, E., Athanasopoulos, E. (eds.) Proceedings of the Engineering Secure Software and Systems - 8th International Symposium, ESSoS 2016, London, UK, 6–8 April 2016, pp. 56–71. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30806-7_4
4. Blakley, B., Heath, C.: The open group security forum: security design patterns. Technical guide. TheOpen Group (2004)
5. BSI: Industrial control system security - top 10 threats and countermeasures 2016. Bsi-cs 005e—version 1.20 of 08/01/2016, federal office for information security (BSI) (2016). https://www.allianz-fuer-cybersicherheit.de/ACS/DE/_/downloads/BSI-CS_005E.pdf?__blob=publicationFile&v=3
6. Dahl, H., Hogganvik, I., Stølen, K.: Structured semantics for the CORAS security risk modelling language. In: Proceedings of 2nd International Workshop on Interoperability solutions on Trust, Security, Policies and QoS for Enhanced Enterprise Systems (IS-TSPQ'07) (2007)
7. Deng, M., Wuyts, K., Scandariato, R., Preneel, B., Joosen, W.: A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. Requir. Eng. **16**(1), 3–32 (2011). https://doi.org/10.1007/s00766-010-0115-7
8. Halkidis, S.T., Tsantalis, N., Chatzigeorgiou, A., Stephanides, G.: Architectural risk analysis of software systems based on security patterns. IEEE Trans. Dependable Secur. Comput. **5**(3), 129–142 (2008). https://doi.org/10.1109/TDSC.2007.70240
9. IEC 62443: Industrial communication networks - network and system security - security for industrial automation and control systems. In: International Standard, International Electrotechnical Commission (IEC) (2013–2018)
10. Khan, R., McLaughlin, K., Laverty, D., Sezer, S.: Stride-based threat modeling for cyber-physical systems. In: 2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), pp. 1–6. IEEE (2017)

11. Kohnfelder, L., Grag, P.: The threats to our products. Technical report. Microsoft Co-oporation (2009). https://adam.shostack.org/microsoft/The-Threats-To-Our-Products.docx
12. Li, T., Paja, E., Mylopoulos, J., Horkoff, J., Beckers, K.: Security attack analysis using attack patterns. In: 2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS), pp. 1–13 (2016). https://doi.org/10.1109/RCIS.2016.7549303
13. Maidl, M., Münz, G., Seltzsam, S., Wagner, M., Wirtz, R., Heisel, M.: Threat modeling for cyber-physical systems: a two-dimensional taxonomy approach for structuring attack actions. In: van Sinderen, M., Fill, H., Maciaszek, L.A. (eds.) Proceedings of the 15th International Conference on Software Technologies, ICSOFT 2020, Lieusaint, Paris, France, 7–9 July 2020, pp. 160–171. ScitePress (2020). https://doi.org/10.5220/0009829901600171
14. Maidl, M., Wirtz, R., Zhao, T., Heisel, M., Wagner, M.: Pattern-based modeling of cyber-physical systems for analyzing security. In: Proceedings of the 24th European Conference on Pattern Languages of Programs. EuroPLop 2019, pp. 23:1–23:10. ACM, New York, NY, USA (2019). https://doi.org/10.1145/3361149.3361172. https://doi.acm.org/10.1145/3361149.3361172
15. MITRE: Common Attack Pattern Enumeration and Classification (CAPEC). https://capec.mitre.org (2019)
16. Shevchenko, N., Frye, B.R., Woody, C.: Threat modeling for cyber-physical system-of-systems: methods evaluation. Carnegie Mellon University Software Engineering Institute. Technical report (2018)
17. Shostack, A.: Threat Modeling - Designing for Security, 1st edn. Wiley, Hoboken (2014)
18. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework 2.0, 2nd edn. Addison-Wesley Professional, Boston (2009)
19. Tuma, K., Calikli, G., Scandariatoa, R.: Threat analysis of software systems: a systematic literature review. J. Syst. Softw. **144**, 275–294 (2018)
20. Uzunov, A.V., Fernández, E.B.: An extensible pattern-based library and taxonomy of security threats for distributed systems. Comput. Stand. Interfaces **36**(4), 734–747 (2014)
21. Xiong, W., Lagerström, R.: Threat modeling - a systematic literature review. Comput. Secur. **84**, 53–69 (2019). https://doi.org/10.1016/j.cose.2019.03.010