# Threat Modeling for Cyber-Physical Systems: A Two-dimensional Taxonomy Approach for Structuring Attack Actions

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*
[2]*University of Duisburg-Essen, Duisburg, Germany*

Keywords: Security Threats, Threat Modeling, Attack Actions, Taxonomy, Catalog.

Abstract: Cyber-physical systems (CPSs) include devices that interaction with the physical world. Hence, attacks against CPSs can lead to substantial damage and endanger life and limb. It is important to consider possible attacks already in the early stages of system development, i.e. during the design phase, by performing threat modeling. Threat modeling aims at identifying, analyzing and documenting potential attacks and threats against a given CPS in a structured way. However, the systematic identification of all relevant threats is not trivial. One challenge is that knowledge about threats or potential attack actions is not documented in a way that makes it easily accessible. To address this challenge, we propose a taxonomy approach for structuring attack actions. The distinguishing feature of the taxonomy approach is the use of two dimensions: attack action types and the attack surface. The attack surface consists of those points of a system at which interaction is possible. Attackers can perform attack actions instead of the intended interaction at these points. As a CPS consists of a range of heterogeneous, connected components that can be accessed in various ways, the attack surface of a CPS is typically large. The attack surface of a specific CPS is defined by its system architecture model. We developed the taxonomy approach to support threat modeling for CPSs. Starting from existing approaches in the context of threat modeling, we extended and modified those in several iterations to meet the challenges of threat modeling for CPSs in industrial projects. While the focus in this paper is on CPSs, the two-dimensional taxonomy approach can be easily applied to other domains.

## 1 INTRODUCTION

Cyber-physical systems (CPSs) are complex systems, based on physical devices and software that are distributed across networks. Such systems are running in many places, for example, industrial plants, public and private infrastructure systems, or energy-generating and distributing facilities. The key tasks are monitoring, automation, and control, which implies interaction with the physical world, and many of these systems are part of critical infrastructures. The IoT (Internet-of-Things) wave is bound to lead to a proliferation of CPSs, also into private homes.

As standard IT is used as well as a specific technology, CPSs are facing at least the same attacks as standard IT systems. The attack surface increases with connectivity, e.g. for remote analytics or remote maintenance.

It is essential to design critical systems with adequate security measures in place, in accordance with security standards like IEC 62443 (IEC 62443, 2018). Those standards require a risk-based approach as a basis for identifying adequate protection measures. To select those measures, relevant attack actions that may have a negative impact on the system's security need to be identified. The term *threat modeling* is commonly used for methods that determine all kinds of threats against a given system and document them. Shostack (Shostack, 2014) defines the term as follows: "Threat modeling is the use of abstractions to aid in thinking about risks." Uzunov and Fernández (Uzunov and Fernández, 2014) 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." Threat libraries collect knowledge about threats, thus assisting software and security engineers in identifying threats for systems under development. Often, such catalogs do not follow a common structure, which makes it difficult to iden-



Maidl, M., Münz, G., Seltzsam, S., Wagner, M., Wirtz, R. and Heisel, M.
Threat Modeling for Cyber-Physical Systems: A Two-dimensional Taxonomy Approach for Structuring Attack Actions.
DOI: 10.5220/0009829901600171
In *Proceedings of the 15th International Conference on Software Technologies (ICSOFT 2020)*, pages 160-171
ISBN: 978-989-758-443-5

tify relevant entries.

A prominent taxonomy used in practice is STRIDE (Kohnfelder and Grag, 2009), a mnemonic consisting of keywords for six threat categories. A difficulty in applying STRIDE's categories is their generic nature, requiring significant security know-how to grasp their meaning for the different elements of the analyzed systems.

Our contribution is to use two dimensions for a taxonomy. The claim of this paper is that our two-dimensional taxonomy makes threat modeling easier for practitioners. One of the dimensions is similar to STRIDE and other taxonomies, as it uses classes of attack actions. 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: Those points of a system at which interaction is possible. Attackers can perform attack actions instead of the intended interaction at these points.

While the taxonomy approach with two dimensions is not restricted to the domain of CPSs, the attack surface dimension has to match the type of system, the technical scope and level of detail that has been chosen for threat modeling of the system. So the attack surface dimension is determined by our target systems, i.e. CPSs. We list the main characteristics of this scope to show where changes would be necessary when transferring the approach to another domain:

- A CPS is composed of various, heterogeneous components: Embedded devices that are connected to sensors and actuators, and workstations or servers that are running standard operating systems and domain-specific applications. The components communicate through a combination of various network protocols. As a result of that complexity, there are various, heterogeneous ways of access, e.g. via physical interfaces like USB, or via standard interfaces of the operating system. The attack surface dimension has to include all ways of access, as all might be misused by attackers.
- Threat modeling should address the system level, where components and protocols are put together into one system. The design of specific elements, e.g. network protocols, is not in scope. Implementation details like the structure of messages or processing of data are not considered for threat modeling.

For our target scope of CPSs, we use the system elements as proposed in our meta-model for CPSs (Maidl et al., 2019).

The two-dimensional taxonomy can be used to structure a catalog of attack actions for a specific scope and domain. We provide an example of such an attack action catalog, which we have compiled by collecting and reviewing common attacks in the context of industrial projects over many years, including threat modeling, penetration tests, and real-world incidents. Besides illustrating the approach, this attack action catalog is of practical interest as it captures common attack actions in our target scope. Practitioners can find the relevant attack actions efficiently by looking into the appropriate field of the catalog during threat modeling of a CPS. The presented attack action catalog covers common attacks and includes attacks that exploit typical weaknesses in standard IT technology. In order 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.

Overall, the paper shows how a two-dimensional taxonomy, and attack action catalogs that are based on this taxonomy, increase quality, efficiency, and completeness of threat modeling. Furthermore, we discuss how the approach allows collecting knowledge in a systematic way to be reusable for future projects.

Our paper is structured as follows: Section 2 provides a terminology for threat modeling. Section 3 presents the two-dimensional taxonomy approach for structuring attack actions, provides a two-dimensional taxonomy, and illustrates it with a catalog for attack actions for CPSs. We compare our taxonomy to STRIDE and CAPEC in Section 4, and discuss related work in Section 5. Finally, we summarize our contributions and provide an outlook on future research directions in Section 6.

## 2 TERMINOLOGY

The terminology in the context of threat modeling varies between different standards and publications. So for reference, Figure 1 shows the terminology of threat modeling that we use in this paper.

The output of threat modeling is a list of *Threat scenario*s. 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

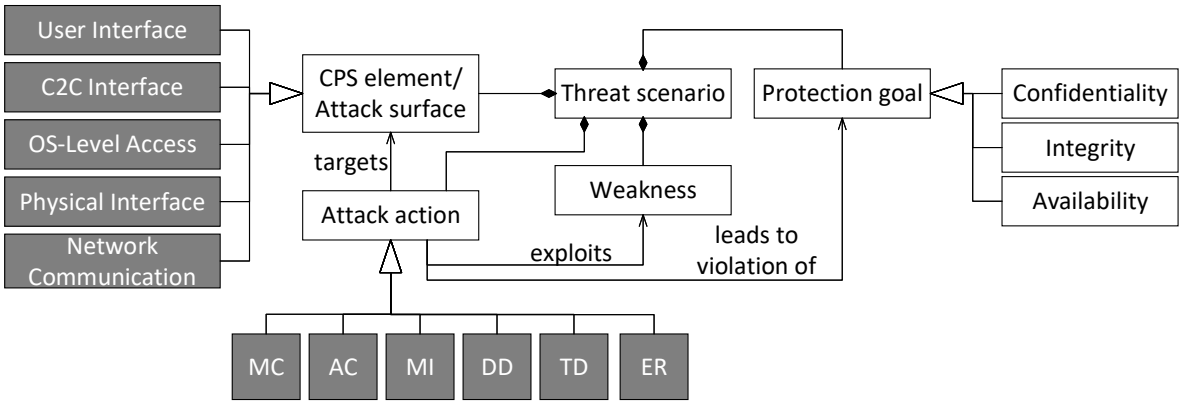


Figure 1: Threat modeling terminology.

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. Figure 1 shows an overview of these categories, and the next section explains the categories in full detail.

# 3 TAXONOMY AND ATTACK ACTION CATALOG FOR CPSs

In the following, we present the main contribution of our paper: The two-dimensional taxonomy approach. We provide a two-dimensional taxonomy of attack actions for the scope of CPSs, and illustrate the taxonomy by presenting an attack action catalog structured according to the taxonomy. We also explain how the taxonomy supports threat modeling. We start with an explanation of the two dimensions of the taxonomy.

## 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 (Maidl et al., 2019). 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 Section 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 Section 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 (BSI, 2016). 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 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 Table 2 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 Section 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 Section 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, 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 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 oper-

Table 1: Two-Dimensional taxonomy.

| | 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 |

ating system.

For network communication, as explained in 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 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.

In Table 2 we provide an example catalog of attack actions against CPSs, structured according to our taxonomy. The catalog in Table 2 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* (BSI, 2016) as well as academic sources like (Uzunov and Fernández, 2014). 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.

Attack actions and their relevance are changing over time, so it is important to emphasize that such a catalog has to be continuously updated.

Table 2: Attack action catalog for CPSs.

| | User Interface | C2C Interface | OS Level Access | Physical Interface | Network Communication |
|---|---|---|---|---|---|
| **MC** | – Phishing to obtain a user's password<br>– Brute force attack on weak password<br>– Setting password through weak password recovery mechanism | – Extract default or hard-coded passwords<br>– Brute force attack on weak passwords<br>– Misuse fake MAC or IP address to authenticate | – Misuse of temporary or default password<br>– Brute-force attack to guess password of OS account<br>– 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<br>– Access via debugging interface<br>– Misuse of direct object references e.g. in URLs<br>– Session hijacking | – Misuse of client-side authentication or authorization<br>– Security downgrade through algorithm negotiation<br>– Misuse of excessively granted privileges | – Bypass of kiosk mode<br>– Misuse of open network service (e.g. Telnet, VNC)<br>– Misuse of (unnecessarily) high privileges in OS<br>– Misuse of unlocked user session | – Access through server management consoles or BIOS<br>– Re-boot with different OS from CD or USB<br>– Access through unprotected near-field communication protocol<br>– Misuse of hardware interfaces (UART, JTAG)<br>– Misuse of shut-down button | |
| **MI** | – Cross-site scripting<br>– SQL-injection<br>– Malware infection of component through malicious payload | – Fuzzing attack<br>– Malware infection of component through malicious payload<br>– Crash due to overload | – Trick OS-user to install or run malware<br>– Network packet exploiting vulnerability in network protocol implementation of the OS, e.g. ping-of-death | – Malware infection of component through infected USB stick | |
| **DD** | | | – Read sensitive data from files or Windows registry, e.g. passwords, operational data<br>– Read data from process memory by causing a core dump | – Steal media, i.e. SD card, USB stick, or hard disk<br>– Install keylogger<br>– Take a covert look at a display<br>– Read data through hardware interfaces (UART, JTAG) | – Read clear text protocols, e.g. HTTP, FTP<br>– Sniff data sent over unprotected WLAN |
| **TD** | | | – Manipulate data in files or databases<br>– Manipulate configuration or software | – Change data on removable media | – Manipulate or replay message<br>– Man-in-the-middle attack |
| **ER** | | | – (Malicious) application uses up CPU or memory | | – Flooding the network<br>– Occupy wireless interfaces with a jammer |

### 3.5 Using the Taxonomy for Threat Modeling

In the process of threat modeling, our taxonomy and attack action catalog help to obtain a list of threat scenarios as described in Section 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.

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.

### 3.6 Further Benefits

The example catalog of Section 3.4 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.

## 4 COMPARISON WITH STRIDE AND CAPEC

In a systematic literature review on threat analysis of software systems performed by Tuma et al. (Tuma et al., 2018), 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 (Shostack, 2014; Halkidis et al., 2008; Deng et al., 2011) as taxonomy, the remaining two refer to CAPEC (Almorsy

et al., 2013; Berger et al., 2016). 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.

## 4.1 STRIDE

STRIDE (Kohnfelder and Grag, 2009) 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 3 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 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 one in Table 2.

## 4.2 CAPEC

The Common Attack Pattern Enumeration and Classification (CAPEC) (MITRE, 2019), maintained by MITRE[2], provides a catalog of attack patterns. 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 4 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

[1]Microsoft Threat Modeling Tool (last access: December 4, 2019): https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling

[2]MITRE: https://www.mitre.org/ (last access: December 12, 2019)

Table 3: Mapping of taxonomy categories - STRIDE.

| 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 | | ✓ | ✓ | | | |

Table 4: Mapping of taxonomy categories - CAPEC.

| 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 | | ✓ | | | | |

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.

# 5 OTHER RELATED WORK

Almorsy et al. (Almorsy et al., 2013) 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. Section 3.1), our proposed taxonomy can be formalized in a similar way.

The paper by Halkidis et al. ((Halkidis et al., 2008)) evaluates the protection that selected security patterns of Blakley and Heath (Blakley et al., 2004) 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 (Uzunov and Fernández, 2014) 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. Section 4.2) is often used as a comprehensive repository for attack descriptions rather than as a taxonomy. An example is Adams et al. (Adams et al., 2018), where CAPEC is used as source to identify relevant attacks, by using machine learning and natural language processing. Another example is the approach of (Li et al., 2016) 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 (Xiong and Lagerström, 2019). 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 (Dahl et al., 2007), 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. (Shevchenko et al., 2018) 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. (Khan et al., 2017) 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. (Deng et al., 2011) 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.

# 6 CONCLUSION

In the present paper, we proposed a taxonomy approach for structuring attack actions. The taxonomy approach consists of using two dimensions: (i) the system elements representing the different parts of the attack surface, and (ii) attack action types. Such a taxonomy can be used to collect and document existing knowledge about common attack actions in a systematic manner, which provides essential support for threat modeling.

The two-dimensional taxonomy presented in this paper is targeted for the domain of CPSs. As an example, we provided a catalog of typical attack actions against CPSs that is structured according to our taxonomy.

As future work, we plan to use the taxonomy approach in other domains, for example, cloud applications. The transfer requires us to adapt the first dimension, i.e. to define the elements of attack surfaces in that domain. For example, a cloud-based system is not structured into components with OS-level interface and C2C-interfaces. Instead, a cloud-based system can be structured into containers and platform services, thus leading to new interface types. With regard to attack actions, first experiments have shown that these are also suitable in the domain of cloud applications.

As mentioned in Section 5, we also plan to extend our taxonomy to be applicable in the context of privacy analysis. The LINDDUN method will be a good starting point since it uses STRIDE. The mapping of our attack action types to STRIDE categories is not that difficult, as we have shown in Section 4.

Last but not least, we have developed a prototype of a tool that supports the identification of threats based on our taxonomy. We have started piloting this tool in industrial projects, and this will enable us to evaluate the tool on a large scale.

# REFERENCES


Adams, S. C., Carter, B. T., Fleming, C. H., and Beling, P. A. (2018). 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, August 1-3, 2018*, pages 490–497.

Almorsy, M., Grundy, J., and Ibrahim, A. S. (2013). Automated software architecture security risk analysis using formalized signatures. In *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*, pages 662–671.

Berger, B. J., Sohr, K., and Koschke, R. (2016). Automatically extracting threats from extended data flow diagrams. In *Engineering Secure Software and Systems - 8th International Symposium, ESSoS 2016, London, UK, April 6-8, 2016. Proceedings*, pages 56–71.

Blakley, B., Heath, C., and of The Open Group Security Forum, M. (2004). Security design patterns. Technical guide, TheOpen Group.

BSI (2016). 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).

Dahl, H., Hogganvik, I., and Stlen, K. (2007). Structured semantics for the coras security risk modelling language. In *Proc. of 2nd International Workshop on Interoperability solutions on Trust, Security, Policies and QoS for Enhanced Enterprise Systems (IS-TSPQ'07)*.

Deng, M., Wuyts, K., Scandariato, R., Preneel, B., and Joosen, W. (2011). A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requir. Eng.*, 16(1):3–32.

Halkidis, S. T., Tsantalis, N., Chatzigeorgiou, A., and Stephanides, G. (2008). Architectural risk analysis of software systems based on security patterns. *IEEE Trans. Dependable Sec. Comput.*, 5(3):129–142.

IEC 62443 (2013-2018). Industrial communication networks - Network and system security - Security for industrial automation and control systems. International standard, International Electrotechnical Commission (IEC).

Khan, R., McLaughlin, K., Laverty, D., and Sezer, S. (2017). Stride-based threat modeling for cyber-physical systems. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE.

Kohnfelder, L. and Grag, P. (2009). The threats to our products. Technical report, Microsoft Cooporation. https://adam.shostack.org/microsoft/The-Threats-To-Our-Products.docx.

Li, T., Paja, E., Mylopoulos, J., Horkoff, J., and Beckers, K. (2016). Security attack analysis using attack patterns. In *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, pages 1–13.

Maidl, M., Wirtz, R., Zhao, T., Heisel, M., and Wagner, M. (2019). Pattern-based modeling of cyber-physical systems for analyzing security. In *Proceedings of the 24th European Conference on Pattern Languages of Programs*, EuroPLop '19, pages 23:1–23:10, New York, NY, USA. ACM.

MITRE (2019). Common Attack Pattern Enumeration and Classification (CAPEC). https://capec.mitre.org.

Shevchenko, N., Frye, B. R., and Woody, C. (2018). Threat modeling for cyber-physical system-of-systems: Methods evaluation. Technical report, Carnegie Mellon University Software Engineering Institute.

Shostack, A. (2014). *Threat modeling - Designing for security*. Wiley Publishing, 1st edition.

Tuma, K., Calikli, G., and Scandariatoa, R. (2018). Threat analysis of software systems: A systematic literature review. *The Journal of Systems & Software*, 144:275–294.

Uzunov, A. V. and Fernández, E. B. (2014). An extensible pattern-based library and taxonomy of security threats for distributed systems. *Computer Standards & Interfaces*, 36(4):734–747.

Xiong, W. and Lagerström, R. (2019). Threat modeling - A systematic literature review. *Computers & Security*, 84:53–69.