Eindhoven University of Technology

MASTER

A Framework for the Design of IoT/IIoT/CPS Honeypots

Ichaarine, S.

Award date:
2022

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Department of Mathematics and Computer Science,
Security Group (SEC)

A Framework for the Design of IoT/IIoT/CPS Honeypots

Master Thesis

Shuaib Ichaarine

Supervisors:
Assistant Professor, dr. Savio Sciancalepore

Committee:
Assistant Professor, dr. Savio Sciancalepore
Assistant Professor, dr. Luca Allodi
Associate Professor, dr. Marc Geilen

Eindhoven, July 2022

Abstract

Honeypots are not new in the field of information security, as even in the IoT/IIoT/CPS domain already, there exist numerous works that propose solutions for a particular goal. However, there exists no overarching methodology that is accessible to develop honeypots specific to an individual's needs. This work, therefore, proposes a framework that enables the systematic construction of a honeypot in the IoT/IIoT/CPS domain, given an objective specified by the user. First, a topology of existing honeypot characteristics was developed, based on over twenty works. Then, relevant observation targets were selected from the MITRE ATT&CK for ICS framework, and categorized into phases of the Cyber Kill Chain. By mapping these observation targets with the constructed topology, different sets of honeypot features were provided as output, which were able to facilitate specific observation targets. Part of the mapping was validated and found to be consistent. To demonstrate that the framework is practical, a use case was presented in which a government agency aimed to develop a honeypot for their building automation system. The proposed methodology was followed, which resulted in a honeypot that facilitated two different attack paths and was able to capture all observation targets derived from the user's objective.

Acknowledgements

This thesis has been written to fulfill the graduation criteria for the Embedded Systems Master's program at the Eindhoven University of Technology. I would like to thank Assistant Professor, dr. Sciancalepore for his close supervision and cooperation over the past 9 months. His willingness to provide support whenever necessary, combined with the extremely fast response times to any of my requests helped me a lot during the project. Furthermore, thank you, Mark, for the daily supervision, the discussions, and for making sure that I felt welcome from the beginning. I am grateful to Assistant Professor, dr. Zambon and PhD candidate Kempinski, for the in-depth discussions regarding the direction of the research as well as feedback on the deliverables. I would also like to thank Assistant Professor, dr. Allodi for his help in the early stages when there were still challenges in formulating the project.

Finally, I would like to express my sincere gratitude to my parents, girlfriend, close friends, and all other persons involved for their unconditional and endless support throughout the course of the project.

Contents

List of Acronyms

AJAX Asynchronous JavaScript And XML

API Application Programming Interface

APT Advanced Persistent Threat

BACnet Building Automation and Control Networks

BAS Building Automation System

BMS Building Management System

CIA Confidentiality-Integrity-Availability

CLI Command-Line Interface

CPS Cyber-Physical Systems

DDoS Distributed Denial of Service

DoS Denial of Service

FTP File Transfer Protocol

GUI Graphical User Interface

HMI Human Machine Interface

HTTP Hypertext Transfer Protocol

HVAC Heating, Ventilation and Air Conditioning

ICS Industrial Control System

IDS Intrusion Detection System

IIoT Industrial Internet of Things

IoT Internet of Things

IP Internet Protocol

IT Information Technology

NIDS Network Intrusion Detection System

OS Operating System

OT Operational Technology

PDU Protocol Data Unit

PLC Programmable Logic Controller

PoC Proof of Concept

POP3 Post Office Protocol 3

REST REpresentational State Transfer

RFC Request for Comments

RSSI Received Signal Strength Indicator

RTU Remote Terminal Unit

SCADA Supervisory Control and Data Acquisition

SMTP Simple Mail Transfer Protocol

SSH Secure Shell

TTP Tactics Techniques and Procedures

VM Virtual Machine

XMPP Extensible Messaging and Presence Protocol

Chapter 1

Introduction

Cyber-Physical Systems (CPS) describe systems that measure and control the physical world to achieve a particular goal. They can be found in many industries, such as electric, water, manufacturing and building automation. Over the past years, CPS have been connected to the Internet increasingly, promoting efficiency and ease of use [1]. While there are clear benefits in the increasing connectivity of these systems, it also increases a greater need for their security. Especially since CPS are developed with long life cycles, older systems typically do not have the required modern security measures in place.

In recent years, various incidents have taken place where critical infrastructure, containing CPS had been attacked, resulting in unavailability or even damage to those systems. In 2015, a power outage was caused by an attack on a regional electricity distribution company in Ukraine [2]. A petrochemical plant in Saudi Arabia was attacked in 2017, where the malware was specifically designed to manipulate safety systems in the critical infrastructure [3]. Besides the implications on the system itself, the disruption of large scale physical processes could have significant impact on the security and safety of many people.

One of the defensive countermeasures for protecting networks is the use of honeypots. These 'fake targets' can be placed inside a network to attract attackers by realistically imitating a device, service or application. By connecting and interacting with the honeypot, the owner is alerted and could collect information about the attacker and its method of operating. Even though honeypots can be very effective as security measure, the development of these mechanisms can be time costly as they are often tailor-made for a specific goal and network environment, and therefore not reusable. The aim of this project is to develop a solution such that a user can systematically construct honeypots for a specific objective in the Internet of Things (IoT)/Industrial Internet of Things (IIoT)/CPS domain. The remainder of this work will have the following structure:

In Chapter 2, we provide background information to give an understanding of honeypots and different type of networks. Furthermore, in the same chapter, a literature research has been performed to see what studies have been performed already in the IoT/IIoT/CPS honeypot research domain. Chapter 3 uses the existing literature to identify the gap in the state-of-the-art and thus provide the motivation for the research. Moreover, it states the research question and subquestions on which the research has been based. In Chapter 4, the methodology is built in a stepwise approach, by first creating an IoT/IIoT/CPS honeypot topology based on honeypots in the literature. Chapter 5 describes the results that are obtained while using the methodology. In Section 5.1 we aim to demonstrate that the mapping performs as intended, using honeypots that were not considered before in the development of the methodology. In Section 5.2, a use case is conducted in which we construct a functional honeypot to show that the framework is practical. In Chapter 6, the results and limitations of the research shall be discussed and finally in Chapter 7, we summarize the conclusions and provide some points to consider for future work.

Chapter 2

Background and Literature Research

In this chapter, an extensive state-of-the-art analysis regarding honeypots is performed, based on (recent) scientific literature. To be able to place the research into context, we will first give a description of honeypots, their different types, and considerations. Next, we will explain the characteristics of the IoT/IIoT/CPS, as well as their differences with traditional Information Technology (IT) infrastructure. Then we will provide a detailed overview of a typical Building Automation System (BAS) network, as this application domain plays an important role in the use case of this research. Finally, we discuss relevant literature to identify gaps in the state-of-the-art.

2.1 Honeypots in a Nutshell

Following one of the earliest definitions, according to SecurityFocus, a honeypot is: "an information system resource whose value lies in unauthorized or illicit use of that resource" [4]. Later on, more definitions appeared, such as "A system (e.g., a web server) or system resource (e.g., a file on a server) that is designed to be attractive to potential crackers and intruders, like honey is attractive to bears.", according to Request for Comments (RFC) 4949 [5]. What these definitions have in common is the core purpose of being attackable by potential intruders. A honeypot is a defense mechanism in an active network, which main value is to monitor adversary behaviour, by posing as a production device. Since the decoy has minimum actual practical use in a network, whenever there is a connection made to the honeypot, it can be assumed that there is malicious intent. When comparing honeypots with other well known monitoring mechanisms, such as a Network Intrusion Detection System (NIDS), there are several advantages for its use:

- Low false positive rate: In contrast to a NIDS, there are very few false hits as a NIDS monitors traffic throughout the entire network. In the case of a honeypot, however, any form of actively initiated communication is suspicious by definition [6].
- Small data sets: As honeypots only monitor local interactions, logging files are notably smaller than those generated by NIDS. Therefore, honeypots are a relatively cheap and easy solution in terms of data analysis compared to NIDS [7].
- Likelihood of capturing valuable information: Depending on the complexity of the honeypot, valuable information ranging over the Pyramid of Pain can be gained [8]. Examples can vary from Internet Protocol (IP) addresses and logins with simple honeypots to zero-day exploits and tooling used for complex decoy systems.
- No encrypted data: Since a honeypot acts as an end device, it does not suffer from dealing with encrypted data. Therefore, processes such as malware reversing and mapping an attack on the honeypot are made much more feasible as opposed to NIDS.

There are also some limitations worthwhile mentioning regarding the use of honeypots [9][10]:

- Narrow field of view: Since honeypots only monitor local traffic, a honeypot is only effective when it is the target of the attack. Therefore, honeypots are not suitable as standalone defense systems.
- Detection: In order to attract a targeted attacker, the honeypot needs to resemble the real device as much as possible. There are many tools available that are able to fingerprint well-known honeypots by their configuration or to detect whether an application/operating system is running in a virtual environment.
- Risks: For specifically production honeypots, there are real risks when due to incorrect configuration/segregation the honeypot could be used to attack production systems when it is taken over.

2.2 Types of Honeypots

From a user's point of view, there are two main reasons to deploy a honeypot. The first reason is to use it as a defense mechanism in an active network. Its primary objective then is to distract the adversary from real systems and alarm the user when interactions with the decoy occur. These types of honeypots, namely *production* honeypots, are not necessarily accessible from the Internet. Instead, their operation logic lies with monitoring internal or isolated networks and are, therefore, also well-suited to defend against local attackers. For example, if the honeypot takes the form of some critical resource (database, mail server), malicious employees or automated malware could be attracted and detected. The second reason for deploying a honeypot is to gain information, either for educational purposes or for threat intelligence. This type of honeypot is referred to as a *research* honeypot and focuses mainly on attackers, their targets, motives, and methods. Research honeypots are generally accessible from the Internet, where the user hopes that an adversary attempts a connection such that data is gathered.

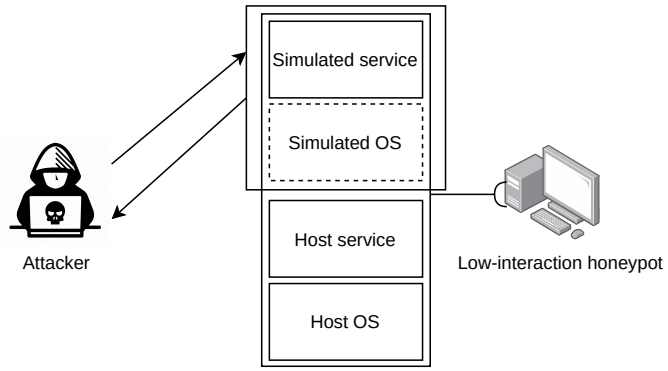

Figure 2.1: Example of layering in a virtualized low-interaction honeypot.

The objective of a user when deploying a honeypot is closely linked to the abilities that the honeypot should possess. In literature, these are categorized into different levels of interaction [10]. While there does not exist a widely accepted or authoritative definition of these different interaction types, somewhat of a description can be given, based on similarities between the different works analyzed in this research. Starting with the simplest one, *low-interaction* honeypots mostly offer a single service that is frequently targeted by attackers. This could for example be an open port, simple web portal, or some (part of a) network service. A typical setup for such a honeypot can be found in Figure 2.1, where the adversary communicates with a sandboxed simulated service. The dashed box in the figure represents the possibility of either presence or

absence of a simulated Operating System (OS) in the honeypot. The reason for it is that this type of honeypot is predominantly script-based, meaning that the attacker has no access to the host operating system. Consequently, low-interaction honeypots have a reduced risk of being taken over and are much easier to set up. On the other hand, the information gathered from this type is limited and consists of often rudimentary data such as raw packets with corresponding origin IP addresses and username/password combinations. Low-interaction honeypots are primarily used in the research domain for statistical analysis of automated attacks, such as scans and spammers. They can also be found as a supplement to an Intrusion Detection System (IDS) when an alarm is triggered as soon as any form of communication with the honeypot occurs. The downside of low-interaction honeypots is that a targeted attacker could detect them relatively fast due to the limited pre-programmed functionality and responses in script-based simulated services.

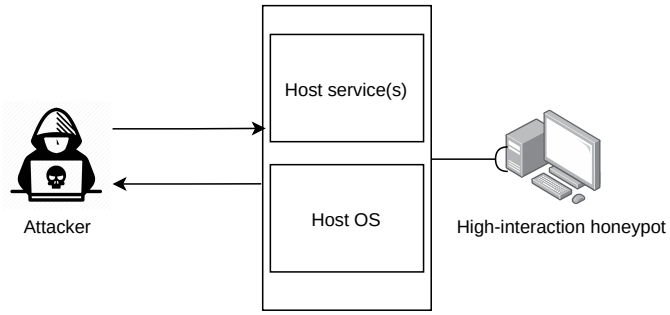

Figure 2.2: Example of layering in a virtualized high-interaction honeypot.

On the other side of the spectrum, *high-interaction* honeypots provide the attacker with much more possibilities. For example, advanced emulation of OS or even the use of real services, allows the adversary to perform more actions on these systems. The goal of this type of honeypot is to gather more complex data from the attacker's activity. Therefore, it is essential that the honeypot behaves as much as a real device as possible, by imitating or even using a real device as shown in Figure 2.2. High-interaction honeypots are generally used in environments where the adversary model is mostly based on the 'targeted attacker', such as Industrial Control Systems (ICSs) [11]. An example would be the use of a real non-production Programmable Logic Controller (PLC) with a mirroring switch in order to log all incoming traffic to the honeypot. Deploying a honeypot in such a scenario would offer the attacker all available services without affecting an actual (cyber-physical) system.
Because of the realistic functionalities of the honeypot, the attacker might use an exploit on an unknown vulnerability, thinking that he successfully intruded into the target network [12]. Logging this type of information can be much more valuable compared to data gathered from a low-interaction honeypot. There are some drawbacks to high-interaction honeypots, however, including high costs due to the complexity or use of a real device, specifically in the IoT/IIoT/CPS domains [13]. Furthermore, there is a higher risk involved since the attacker might take control of the operating system and attempt to use the honeypot for lateral movement towards an actual production device. Therefore, proper configuration and network structure are crucial to prevent more harm than good from being done by the high-interaction type.

2.3 IoT, IIoT and CPS Networks

Being able to identify the attack surfaces and vectors in an IoT, IIoT, or CPS network, requires an understanding of such network structures. Honeypots and other security mechanisms that originated in the IT world, are increasingly being incorporated in the CPS domain [14]. Therefore, using differences between these networks it should become clear what adaptations are necessary for an IoT, IIoT, or CPS honeypot. To determine the differences between the aforementioned

Table 2.1: Key differences between CPS and IoT.

CPS	IoT
Specific domain	No specific domain
Scalable	Highly scalable
Data generation	Very high data generation
Isolated system	Internet-connected

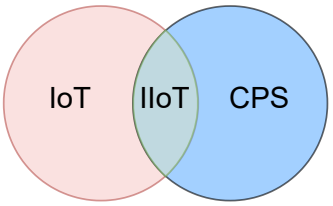

Figure 2.3: Venn-diagram of IoT, IIoT, and CPS.

network types, we first provide their definitions based on available recognized and authoritative sources.

IoT: A global infrastructure for the information society, enabling advanced services by interconnecting physical and virtual things based on existing and evolving interoperable information and communication technologies [15].

CPS: Systems that use computations and communications deeply embedded in and interacting with physical processes to add new capabilities to physical systems [16]. BAS and ICS are examples of a CPS.

IIoT: A subset within IoT that focuses on industrial applications [17].

These concepts and their relationships are visualized in Figure 2.3 by a Venn diagram [18]. It shows that IIoT is a subset of both IoT and CPS, containing the overlap between these two concepts. Therefore, we could say that IIoT consists of CPS of which their embedded systems are inter(net)connected, enabling advanced services. Table 2.1 shows the key differences that distinguish CPS from IoT. Examples of specific domains that CPS refer to, are ICS and our use case, BAS. The main difference between IT and IoT lies in the characteristics, such as the heterogeneous and resource-constrained devices that are present in IoT, together with its ultra-large-scale network that connects all 'things' to the Internet [19].

To get a better understanding of the architecture of typical IIoT and CPS networks, we are using BAS as an example, since it belongs to the CPS domain and will be relevant for our use case. The main differences between traditional IT and BAS boil down to the case of IT versus Operational Technology (OT). Whereas IT mainly deals with data and the flow of digital information, OT deals with the operation of physical processes. In the BAS domain, this translates into services available in buildings, for example, Heating, Ventilation and Air Conditioning (HVAC), lighting, access control mechanisms, and (fire) alarms among others. Consequently, the availability of these services is paramount in BAS, whereas confidentiality and integrity are the main security-related priorities in IT networks. Moreover, the disruption of physical processes inside a building has a direct impact on the persons and things in that building.

In Figure 2.4, the typical structure of a BAS network is depicted. Depending on the terminology used, three or four layers can generally be distinguished. Starting from the top, the highest layer describes devices used for the management of the network. Specific devices on this layer are mostly engineering workstations used by operators, to monitor or control the system. These devices might or might not be connected to the Internet, resulting in different attack paths.
In the middle layer (Automation), the building automation controller connects different field devices within the same subnet. The reason for the naming of the automation layer is that the controller enables autonomous interoperability between subsystems of different vendors using (pre-)configured control logic. Communication between automation layer devices and management layer devices generally takes place using IP-based protocols, such as Hypertext Transfer Protocol

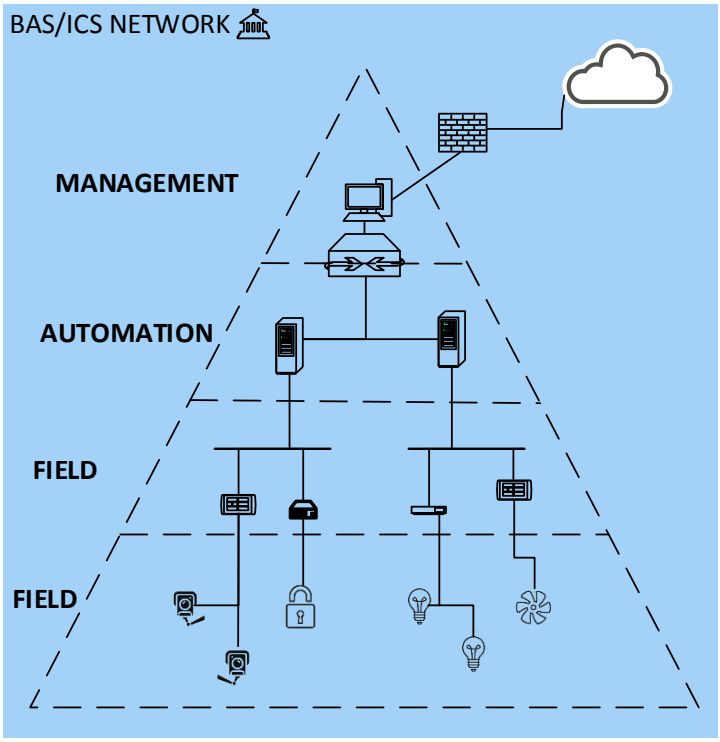

Figure 2.4: Typical network structure of a Building Automation System.

(HTTP), File Transfer Protocol (FTP) or BAS specific protocol stacks, such as LonWorks and Building Automation and Control Networks (BACnet). BACnet is an open protocol stack specifically used in BAS, consisting of application, network, data link, and physical layers. In the example of Figure 2.4, the two controllers in the automation layer both have their separate local networks but are still inter-connected through an IP-based protocol such as BACnet/IP. Even though this interconnectivity can increase efficiency (think about light control in adjacent parts of a building), it creates opportunities for adversary lateral movement, especially since these types of protocols do not have security implemented by design.

The Field layer describes the subsystems that interact with the physical world, such as sensors, actuators, and their corresponding PLCs and Remote Terminal Units (RTUs). The communication of these devices with the automation controller takes place using a variety of field protocols such as Modbus, MS/TP, KNX, LonTalk, and many others.

The attack paths in a BAS heavily depend on the configuration of its network. Attack paths in general can be modeled through the MITRE ATT&CK Framework [20]. This framework enables threat modeling, ranging from information about particular Advanced Persistent Threat (APT) as well as the tactics, techniques, and software used for specific attacks. In the ATT&CK Framework, tactics represent tactical adversary goals during an attack. Techniques describe the means by which the attacker attempts to achieve a certain goal. The ATT&CK Framework will contribute significantly to our research, particularly in Section 4.3.2. Dos Santos et al. from the company Forescout listed potential attack paths on different surfaces, which can be found in Figure 2.5 [21]. The main distinction that can be made is whether a device within the BAS is directly connected to the Internet or not. In the first scenario, the attacker can reach a device in the BAS network from the Internet. There are various ways in which the exposed device could unintentionally provide initial access such as an 'Exploit Public-Facing Application' (in case of a server) or by use of hard-coded/default credentials (in case of an IoT device or PLC) [22][23].

The alternative scenario in Figure 2.5, indicated by the purple path, describes a properly air-gapped network, which requires physical or close access to the network. Techniques such as 'Hardware Addition' or 'Replication Through Removable Media' could be used to gain initial access through the use of USB sticks or by connecting a malicious device to the network.

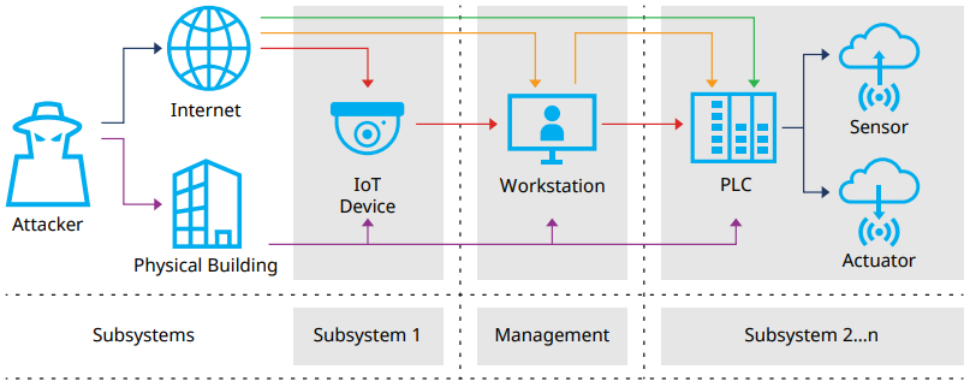

Figure 2.5: Potential BAS attacker paths identified by Forescout [21].

Since security features such as authentication and monitoring are often not integrated or optional in BAS protocols, initial access is generally the hardest tactic to apply. Once the adversary is inside a network, he or she can move around easier in a BAS compared to an IT network. This makes BAS also significantly susceptible to insider threats. For example, an insider could exploit an IoT device on local network A, and move up the hierarchy, through the building automation controllers to local network B being responsible for HVAC or access control.

2.4 State-of-the-Art Analysis

Over the past fifteen years, lots of research and progress has been made in the honeypots research domain. In 2004, Google engineer Niels Provos released one of the first open-source honeypot frameworks for virtual computer systems [24]. Honeyd could be configured with different services, such as Post Office Protocol 3 (POP3), TelNet, Simple Mail Transfer Protocol (SMTP), or FTP such that an adversary could communicate with the honeypot on the network level. While this framework allowed the spawn of thousands of virtual devices on a network, its low interaction severely limited the opportunity of gathering information about the attack(er). However, over the years Honeyd has matured significantly due to its increased configurability resulting in the possibility for high-interaction honeypots. Furthermore, a lot of services, including industrial protocols, were added such that a large number of recent honeypots have and are being constructed using this framework [25][26][27][28][29].

In 2018, Wang et al. developed ThingPot, and proclaimed the system as medium-interaction IoT honeypot [30]. In this research, medium-interaction is described as a combination of high and low interaction elements. According to the definition that we use, which will be further specified in Section 4.2.1, ThingPot can be considered to offer high interaction. The research uses an Extensible Messaging and Presence Protocol (XMPP) client and a REpresentational State Transfer (REST) Application Programming Interface (API) for the use case of imitating a Philips Hue smart light system. Data was collected for 1.5 months, by logging both services and classifying captured requests as 'targeted', 'untargeted', and 'undefined'. This way, the distinction could be made between general scanning activity and specific attacks. Whereas the XMPP service did not receive any direct requests, the REST logs did indicate attacker activity. More interestingly, an attacker methodology was derived in the research, where after general scanning for openings, targeted attacks using brute force or fuzzing would take place.

Another honeypot in the IoT domain, namely Honeytrack, has been developed by Kamoen [31]. This work was built on top of an existing open-source framework and extended the available services with TelNet [32]. Another feature of this honeypot is that it is able to save the virtual machine state for each attacker. Consequently, sessions could be restored, allowing for higher interaction and thus increasing the amount of knowledge that could be gathered.

A recent study by Dodson et al. describes the largest high-interaction ICS honeypot network in the literature yet, mimicking RTUs and PLCs using dispersed geographical locations [33]. Due to the convincing imitation of ICS devices, they were able to identify new ICS exploits injected in the honeypot network. The research states that a clear distinguishment can be made between adversary motivation for targeting IoT and ICS devices. Where IoT is susceptible to large-scale attacks, ICS lacks this interest mostly by its high cost of entry, a fragmented population that uses proprietary software, and the generally limited resources of ICS devices. Therefore, the adversary model is narrowed down to a targeted attacker with particular knowledge, looking for specific devices. Dodson et al. states that, for that reason, an effective ICS honeypot should offer high interaction. A worrying observation is that the convergence of IIoT and IoT domains results in a decrease in the gap between ICS- and IoT-aware hosts, which could result in an increased attractiveness for large-scale ICS attacks in the future. Finally, a set of recommendations is given to improve the effectiveness of ICS honeypots, based on the data collected in a large-scale experiment. Key points are the use of realistic IP addresses, as well as a systematic and continuous deployment to allow a larger attacker window, again matching the modus operandi of the targeted attacker.

The most realistic ICS honeypot in the literature up to the present time is that developed by Hilt et al. [34]. It could be called a honeynet since an entire ICS environment consisting of a multitude of devices was deployed. A factory plant was simulated using four real PLCs combined with three Virtual Machines (VMs) where the goal was to build a honeypot appearing so realistic that not even a 'well-trained control systems engineer' could see through without making the uttermost efforts. What is even more interesting in this paper is the presence of non-technical features that contribute to the attractiveness of the honeypot. A fake company was set up to online presence through a company website with AI-generated employee pictures, and other information such as available phone numbers and mailboxes. They even went a step further, by repeatedly advertising the company's network as vulnerable on forums popular among cybercriminals. The honeypot received a variety of interactions, including multiple successful ransomware attempts, where negotiations with cybercriminals were recorded and published. Furthermore, cryptocurrency miners, different kinds of fraud, and a beaconing attack were found in the logs.

A paper from Cifranic et al. describes the Decepti-SCADA framework that uses a modular, Dockerized design to create a high-interaction ICS honeypot [35]. The architecture consists of two parts. The first part provides the honeypot deployment while the second part is monitoring network traffic and provides a user interface. Due to its modularity, new containers can be added and deployed using the images of arbitrary existing Supervisory Control and Data Acquisition (SCADA) devices. It remains unclear how the device images were retrieved and whether they are publicly accessible. As a result, it is impossible to assess how far high-interaction is achieved in this framework, as the major challenge in an extensible high-interaction honeypot is the realistic imitation of proprietary software from a wide range of manufacturers. Also, contrary to the extensive networking logging present, the paper's architecture shows no indication of host-based logging, which plays a significant role in capturing complex attacks on high-interaction honeypots.

An extensive survey by Franco et al. analyzed a large number of existing honeypots in the IoT and ICS domain [36]. Common characteristics of state-of-the-art honeypots and honeynets were extracted and key design factors and open research problems were discussed. One of these design factors is Resource Level Selection, which determines whether virtual, real, or a combination of

both device types are used in a honeypot. While the paper states that a virtual honeypot costs 12.5 times less to maintain than a physical honeypot, virtual environment detection techniques used by adversaries could decrease the effectiveness of simulated honeypots. Furthermore, the choice of services to provide is crucial in the design. Not only the support of device-specific protocols but also file persistency, response times, and commands for utilities should be realistic to prevent honeypot fingerprintability. Open issues identified by Franco et al. mainly concern emerging technologies and unexplored protocols. For example, out of the 79 honeypots investigated, 43 works concerned CPS while only a single one of those was specifically designed for a building automation system [37]. Next, in accordance with Dodson et al., an emphasis is put on deployment locations, such as cloud providers versus private locations. They claim to have found no study that aims to optimize this deployment location. Also, the majority of honeypots described in the literature are research honeypots, which do not actively participate in securing an IoT/ICS production environment. Therefore, we think it would be interesting to combine a honeypot with an IDS for integration in a production environment as a defense mechanism.

Litchfield et al. proposed HoneyPhy, the BAS honeypot mentioned above, which contributes by realistically modeling process behavior and providing auxiliary information arising from the attached physical system [37]. The purpose of this is to prevent an attacker from being alerted by the lack of delay and deviations from expected process behavior. As a solution, Litchfield et al. introduce the '*hybrid-interaction honeypot*' that could use real devices to interact with a process simulation. As Proof of Concept (PoC), an HVAC honeypot was developed, using a low-interaction imitation of a SEL-751A device and a physics-based heating and cooling simulation model. The major problem with this honeypot framework, however, is that extendibility is very limited for other cyber-physical applications. Devices and realistic process models are generally only valid for a single application, which would require redefining the physical model for every environment the honeypot is placed in.

In 2017, Lin developed a honeypot based on a Siemens APOGEE building automation system using P2 and BACnet protocols [38]. The center of attention in this research was to increase realism by generating network traffic that mimics genuine control systems. The motivation for this work was the claim that the lack of OT traffic on a network can give away the identity of honeypots. As an APT may involve passive network monitoring in a multi-staged attack, realistically generated traffic coming from honeypots may increase the interest of the attacker in that particular device. By extracting the characteristics from collected traffic sets of the Siemens device, a configuration file could be composed that was used to set up virtual hosts through the Honeyd framework.

Only one other honeypot specifically for BAS has been found in the literature, developed by Bauer et al. [39]. This research aimed to investigate whether malicious actors were already carrying out attacks against Internet-connected BAS devices and how to distinguish targeted attacks from widespread attacks on arbitrary devices. By imitating an existing BAS device (DDC4200), an Secure Shell (SSH) server and web interface were set up, connected to the Internet through thought out IP addresses. After a ten-week deployment period, no attacks were found, however, except for a significant number of SSH login attempts. As no adversary information was collected, it is difficult to conclude whether the honeypot was not realistic enough, there was no attacker interest, or if there was another cause.

Chapter 3

Motivation and Research Questions

The analyzed literature provides a great level of detail regarding the capabilities of the developed honeypots. Therefore, it is possible to extract specific features for each honeypot. Table 3.1 contains an overview of the related work, the corresponding application domain, and a set of features derived from each research. This set serves as an example and is merely a selection from a multitude of technicalities that can be derived from the different works. It can be seen that there are significant differences between the honeypots, even within the same application domain. This observation is in line with the state-of-the-art analysis, where each honeypot seemed to possess its own specific capabilities and components. The reason for this heterogeneity remains mostly unclear and in practice even extends beyond the specific set of features listed in Table 3.1.

Work	**Application**	**Features**				
		Interaction	Virtualization	Physical process	Persistency	Size
Honeyd [24]	IT	Low	Service Imitation	No	No	Pot
Bauer et al. [39]	BAS	Low	Service Imitation	No	No	Pot
HoneyPHY [37]	BAS	Low	Service Imitation	Model	No	Pot
Dodson et al. [33]	ICS/BAS	High	Real Device	No	Yes	Pot
Hilt et al. [34]	ICS	High	Real Device	Real plant	Yes	Net
Cifranic et al. [35]	ICS	High	Digital Twin	No	No	Pot
Honeytrack [31]	IoT	Low	Service imitation	No	Yes	Pot
ThingPot [30]	IoT	High	Digital Twin	No	No	Pot

Table 3.1: Overview of related work, including a set of selected features.

This heterogeneous nature of honeypots, however, appears to pose difficulties for an individual who intends to construct one. Specifically, a systematic methodology to select appropriate features in order to meet the user's objective seems absent. It would appear that there is a gap in the state-of-the-art, providing the coupling between security threats, and the composition of a honeypot being able to mitigate these threats. With increasing connectivity and integration of heterogeneous interconnected IoT devices in CPS, the attack surface is expected to increase significantly in the coming years. Therefore, a framework that allows for systematic development of honeypots in IoT/IIoT/CPS networks seems to be well-needed.

This project aims to create a framework for the development of IoT, IIoT, and CPS honeypots. Using this framework, it should become straightforward to develop and deploy an arbitrary honeypot in an IoT/IIoT/CPS environment, given the user's objective, its observation targets, and a set of parameters obtained from the target network environment. Finally, the practical functionality of the framework will be demonstrated by a use case where we plan to develop and deploy a PoC

BAS honeypot. BAS is an interesting domain application domain because of the convergence of IoT and CPS, caused by the incorporation of IoT end devices in buildings [40]. Furthermore, security in the BAS domain is relatively underrepresented in the literature, while new vulnerabilities are being discovered increasingly [21][41]. Based on the described problem, derived from state-of-the-art analysis, the following research question and sub-questions have been composed.

How can we develop a systematic framework for the construction of arbitrary honeypots in IoT, IIoT and CPS networks?"

Sub-questions
1. What (technical) features can be derived by the deconstruction of existing IoT, IIoT, and CPS honeypots?
2. What are the known attacker Tactics Techniques and Proceduress (TTPs) that are used in IoT, IIoT and CPS networks and how can they be translated to observation targets?
3. How can those relevant observation targets be mapped to the derived (technical) features?
4. How can we practically apply the obtained framework to create a PoC honeypot in a BAS use case?

Chapter 4

Methodology

To answer the research question, we aim to create a framework that guides the user when constructing an IoT/IIoT/CPS honeypot. The composition of this proposed framework for systematic honeypot development yields multiple stages. In the first stage, Subsection 4.2, we aim to obtain an IoT/IIoT/CPS honeypot topology by analyzing existing honeypots and deconstructing these works in so-called 'features'. The features allow for the classification of certain technical properties of each honeypot, such that each honeypot can be dissected in a combination of existing or new features. After completion, there should be an as complete as possible overview of technical aspects. Next, in Section 4.3 we specify what interaction needs to be caught by a honeypot, by extracting relevant techniques from the ATT&CK for the ICS framework and categorizing them in phases of the Cyber Kill Chain [42][43]. As the final step of the methodology, we map the obtained observation targets to the required features in Section 4.4.

4.1 Our Methodology in a Nutshell

The methodology that we aim to develop consist of three phases. Each phase represents a step that the user needs to take and where the framework should offer assistance. In the first phase, the user can, based on his objective, select certain observation targets which together form the attack path that the user aims to capture.

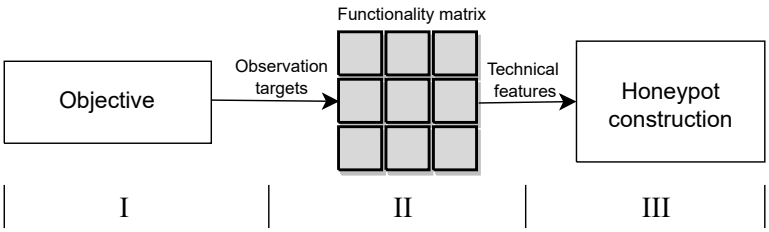

Figure 4.1: Overview of the methodology that we aim to develop.

These observation targets are the input for the second phase, where they are translated to sets of technical features that we will identify in the upcoming section. To perform this mapping, an overview of all known honeypot features is required. In the third phase, the sets of collected features are bundled to a single set, which forms the basis for the constructed honeypot. Figure 4.1 shows an overview of the methodology, with corresponding phases.

4.2 Honeypot Topology

For the first research sub-question, *What (technical) features can be derived by the deconstruction of existing IoT, IIoT, and CPS honeypots?*, we aim to create a representative topology. To do this, over twenty IoT/IIoT/CPS honeypots have been studied in this research. Most literature emphasizes specific characteristics that have been implemented or are being proposed. However, a top-down approach is preferred to allow stepwise refinement in the course of analyzing the honeypots. For that reason, the initial step is to make a distinction between the general purpose of each characteristic, based on the top-level features shown in Figure 4.2. These features form the highest level of abstraction in the topology and are further specified in this section.

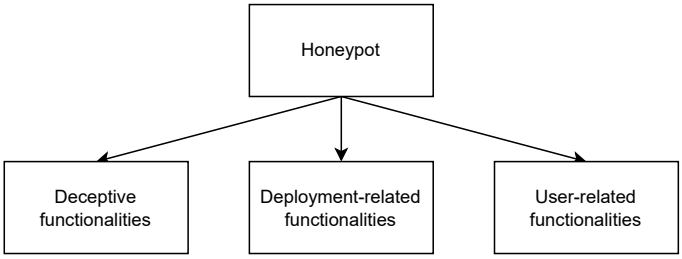

Figure 4.2: Top-level features that describe the general purpose of a characteristic.

- **Deceptive functionalities:** Features belonging to the deceptive functionality have the objective to trick an attacker into thinking that it is facing a real and relevant system. They describe, mostly technical, parts that together compose the perceived system and enable interaction initiated by the adversary.
- **Deployment-related functionalities:** The features within this characteristic do not necessarily translate into a product, but rather provide advice or context about the deployment conditions in order to increase the likelihood of capturing the desired interaction. In order to prevent any influence on the deceptive composition of the honeypot, this top-level feature considers functionalities around a finished product that is ready to be deployed.
- **User-related functionalities:** Features that are related to the user have the objective of capturing the interaction that is enabled by the deceptive functionalities and extracting relevant information if applicable.

Each top-level functionality contains multiple features on a lower level of abstraction that further elaborate on characteristics that together contribute to the top-level objective. For example, a feature that has already been identified in the literature and was mentioned in Section 2.2 is the 'Interaction Level'. This feature belongs to the Deceptive functionalities because it is related to the decoy system that is perceived by the attacker. The interaction level feature, along with many others, is further discussed in Subsections 4.2.1, 4.2.2, and 4.2.3.

4.2.1 Deceptive Functionalities

The features belonging to the objective of providing a decoy system, reside on a lower layer of abstraction. These meta-level features each describe a relevant aspect that together form a basis for the deceptive composition of the honeypot. An overview of identified meta-level features can be found in Figure 4.3. All features described have been derived from existing honeypot research. In the descriptions that will follow, we refer to some of these works. A full overview of the analyzed honeypots and their categorization can be found in Appendix A.

Below the meta-level layer, the tech-level features can be found. Tech-level features represent a value for their parent, where for some features only a single option is allowed. The features *Size*, *Interaction Level*, *Virtualization*, and *Physical Process* all have options whereof only one can be picked (exclusive). For the features *Services* and *External Persuasion*, multiple tech-level options can be selected. For example, a honeypot cannot be low and high interaction at the same time, while it could be possible to have both an Human Machine Interface (HMI) and some protocols integrated into the system.

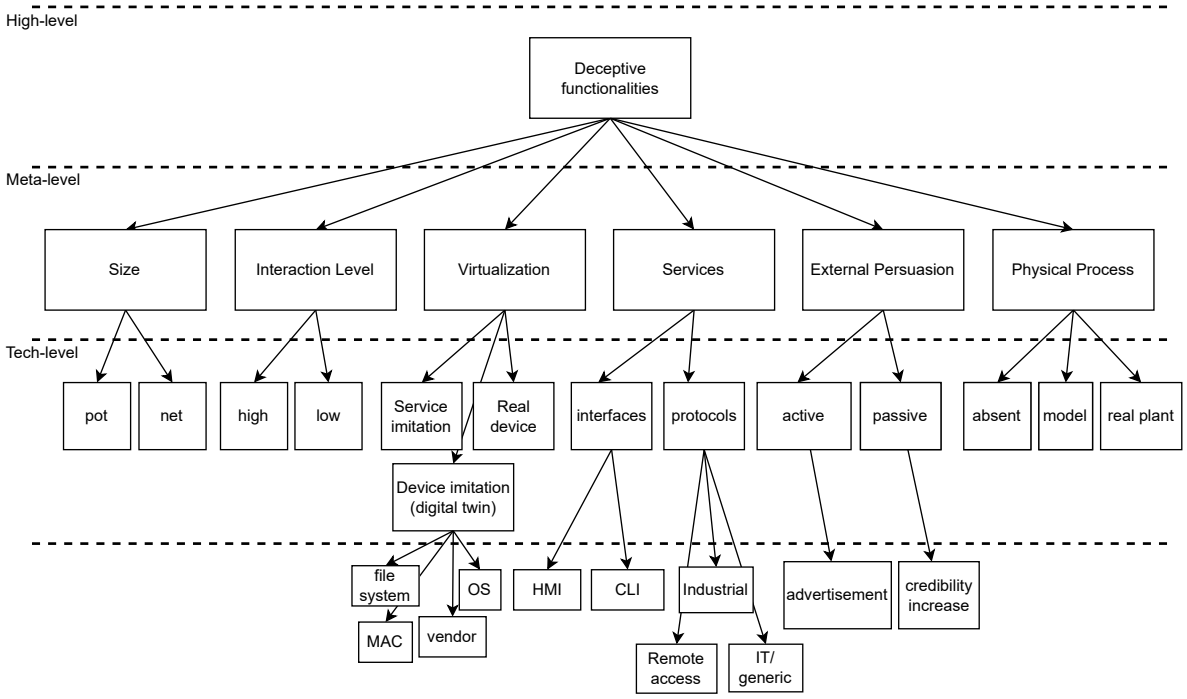

Figure 4.3: An overview of features that belong to the deceptive functionality, on different layers of abstraction.

Furthermore, cost plays a significant role when developing a honeypot. In Section 2.4, we described the cost differences between the use of virtual and real devices, based on the research of Franco et al. [36]. Therefore, we assign relative costs to the exclusive tech-level parameters. This will help in selecting the cheapest option when more than one tech-level feature can be selected within the same exclusive meta-level parent. We will see this becoming relevant in Section 4.4, where it becomes visible what features are necessary to catch specific observation targets. Within the deceptive functionalities, the following features have been identified:

– **Size**: describes the volumetric extent of the honeypot system. The honeypot can present itself as a standalone device, a pot. On the other hand, the decoy system offered to the attacker can be a network of interconnected devices, also referred to as 'honeynet'. The vast majority of available honeypots are developed and presented as a single pot. For that reason, the pot generally imitates a specific service of a device or the device in its entirety. Conpot, for example, contains several templates of which a single be selected upon initialization. Each template contains a set of services that can be attributed to a particular device [44]. Other examples of pots in our examined set are Mimepot, CryPLH, Honeyd, and DiPot [45] [46][24][47].

  Contrary to the standalone pots, some of the studied honeypots clearly present themselves as a net of interconnected devices [34][48][49][50]. This means that the attacker perceivably

has the possibility to interact with more than one device. Furthermore, these devices are then connected to form a network that can exchange information. The honeynet developed by Hilt et al., for instance, contains four PLCs, a file server, a network switch, and other devices that an attacker could target. The net that Piggin and Buffey provided, consists of a PLC, workstation, and switch, however, it remains unclear which exact devices were used.

The tech-level features within the *Size* feature can be described by the following set:

$Size = \{pot, net\}$, where in terms of cost, $pot < net$.
The reasoning behind this cost is quite self-explanatory: a net consisting of two connected similar pots has a higher cost than that of a single instance of the same pot. For the purpose of consistency, we refer to each work as a honeypot, regardless of the value of the *Size feature*. In the case of a net, we use terms along the lines of ‘ a honeypot system that consists of a net of interconnected devices ’.

– **Interaction level**: this feature is a subject of discussion in the honeypot research domain. As mentioned in Section 2.2, there exists no definitive description, resulting in honeypots with self-proclaimed interaction. This research defines the interaction level as the quality of implemented services. If the service is implemented in a way that can enable all interaction that would be possible with that same real service, the honeypot is deemed high interaction. When a service is implemented in a manner such that the intended attacker is limited in interaction through that service, we consider a low interaction honeypot. This definition corresponds to other literature that states that a low-interaction honeypot is software-based [51]. Whereas some authors classify a third category, medium interaction, this term could increase the ‘grey area’ around interpretations of interaction level. Therefore, this research considers most self-proclaimed medium interaction honeypots as low interaction. An example is Kippo, which simulates an (SSH) service and additionally offers a fake filesystem [52]. Execution or modification of those files, however, is not supported which directly shows the limitations of low-interaction decoys. Other honeypots in this category are Conpot, Digital Bond, HoneyPHY, and the research by Simões et al. [44][53][37][54].

  In our analyzed set, we also classified honeypots as high-interaction, of which a few are Cifranic, Honware, and Piggin and Buffey [35][55][50]. The first two used device emulations which allowed for service implementations close to real services. The latter claimed to have used PLC hardware, implying that the corresponding services are real and thus per definition high interaction.

  The tech-level features within the *Interaction Level* feature can be described by the following set:

  $Interaction\ level = \{low, high\}$, where in terms of cost, $low < high$.

– **Virtualization**:

  Virtualization should be considered as the extent to which a service/device is imitated or real, and does not necessarily reflect the level of interaction offered. We distinguish three categories of virtualization. In the first category, named service imitation, the focus of the honeypot is on one or more particular services that are offered to the adversary. Concretely, the entire honeypot consists of the implemented services and a handler to interpret any input from the attacker. In practice, these honeypots are generally script-based and contain a preprogrammed set of commands that they are capable to handle. This category involves the largest portion of current existing honeypots. SSH or Telnet honeypots, such as those of Kippo and Cowrie, are common examples that fall into this group [52] [56].

  The second category within this feature is device imitation, which in literature is also referred to as a digital twin [57]. In this category, the honeypot resembles a digital copy of a specific device, consisting of an underlying OS, file system, and software related to the device it mimics. Consequently, more attack paths, such as through software vulnerabilities, could be introduced since file execution is possible in these honeypots. In addition, these types

of deceptive systems facilitate persistence, such that the attacker is able to resume the session at another moment. Due to the aforementioned characteristics of this category, the implementations are in practice not script-based but rather bundled into a virtual machine. An example that falls into this category is Thingpot, which mimics a Philips Hue smart lighting system by not only providing similar communication protocols but also implementing the device-specific backend API and a representative frontend [30]. Other instances of device imitations are Digital Bond, providing a VM that resembles a Modicon Quantum PLC, Decepti-SCADA which uses Docker images to replicate real operation systems, and Honware which is self-adaptive and able to emulate a variety of devices [53][35][55].

The third category, real device, entitles the use of the actual physical device. The added value of this category is the presence of hardware, enabling a variety of attack paths that depend on physical access to the device. Furthermore, the services in the honeypot have the highest level of realism, compared to the other categories of the *Virtualization* feature. Some of the analyzed works that make use of real devices are Piggin and Buffey which provided real PLC hardware, Pliatsios et al. which have used RTU devices, and SIPHON which used real IP-cameras [50][48][58].

In practice, a correlation can be observed between interaction level and virtualization. Real devices offer high interaction by default, since the services contained in that device are real and thus maintain the highest level of quality. Service imitations, however, generally provide low interaction. Conpot, for example, uses scripts to simulate services that are able to handle a fixed set of commands. The latter does not imply that high interaction service imitations are impossible by default, but it takes considerably more effort to mimic a service, compared to implementing the real service in a device imitation.

The tech-level features within the *Virtualization* feature can be described by the following set:

$Virtualization = \{service\ imitation, device\ imitation, real\ device\}$, where in terms of cost, $service\ imitation < device\ imitation < real\ device$.

The reasoning behind the cost is that a device imitation consists of at least the same services as a service imitation while having additional effort in resembling the device by providing other characteristics. The same holds for a real device, that has at least the same software as the device imitation, but additionally contains the device-specific hardware.

– **Services**:

We describe services as (software) functionalities through which an attacker can interact with the deceptive system. Through the services present in the honeypot, the attacker can attempt to reach objectives such as disruption, theft of information, or other forceful activities. Taking this definition into account, it logically follows that all honeypots should have at least one service implemented in order to provide a possibility for interaction. Two main types of services are distinguished within this feature. Firstly, the interface type that the adversary interacts with, and secondly, the underlying protocols that facilitate the interaction.

With respect to the interface types, we differentiate between a Graphical User Interface (GUI) and a Command-Line Interface (CLI). As the name indicates, the first type provides a graphical overview of the system and/or the user's possible actions. GUIs are especially common in CPS since they can supply the user with a clear visual overview of states within a physical process. For instance, Bauer et al. integrated a control web interface displaying information about the mimicked device and its in/outputs [39]. SIPHON contained web interfaces where, upon successful authentication, the IP cameras could be controlled and live images could be seen [58]. Gridpot used specific HMI software of which the details remain unknown, according to the authors, to prevent fingerprintability [49].

The CLI is a text-based interface used to run programs, manage files and interact with a device in general. It is often provided in cases where there is no need for visual interaction. This type of interface is generally implemented when the attacker is able to navigate through a decoy file system or send remote commands which will be discussed in the protocols below.

The protocols in this research are divided into three categories, as can be seen in Figure 4.3. IT protocols are application layer protocols that originate from the IT domain, with well-known examples as DNS, HTTP, FTP, and SNMP. Even though these protocols are not specific to CPS, often they are still integrated with devices tied to this domain, to facilitate web services, file transfer or network configuration and management.

The next category describes protocols that facilitate remote access to a machine. Upon successful connection and authentication, these protocols can be used to control a remote machine. Examples of widely used remote access protocols are RDP, RAS, SSH, and Telnet. This is generally done in the form of the earlier described CLI that is provided to the user. Therefore, the use of these remote access protocols is coupled with the use of command-line interfaces within this same meta-level feature.

The third category involves protocols that are explicitly used for industrial purposes. Their objective lies mainly in the automation of physical processes, through sensor readings, computational logic, and actuator control. The most frequent industrial protocols encountered in the analyzed set are Modbus and S7comm. Some of the industrial protocols are even more domain-specific, such as BACnet which is predominantly used in the BAS domain. An extensive overview of all protocols that were implemented in the honeypots that we investigated, can be found in Table 4.1.

The tech-level features within the *Services* feature can be described by the following set:

$Services = \{interfaces$: GUI, CLI, $protocols$: IT, Remote Access, Industrial$\}$ As mentioned earlier in this section, only features where a single value needs to be picked, are assigned relative costs. Since multiple interface and protocol types are simultaneously possible in a honeypot, no cost is assigned.

– **External Persuasion**: describes methods to lure potential adversaries or to increase credibility of the honeypot. This takes place outside the technical part of the honeypot system and can be realized in either active or passive methods. Within our set, only Hilt et al. implemented this feature [34]. Passive methods are meant to support the existence of the decoy such that an attacker 'believes' that the system is real and belongs to some company or organization. Hilt et al. put significant effort into this method, by constructing a fictitious company including a website and contact details that were in fact reachable and handled incoming messages.

  Active methods constitute a more aggressive form, in which the honeypot is deliberately promoted as a real vulnerable system. Hilt et al. applied this method by posts on Pastebin that presented the honeypot as a vulnerable robotics workstation. This category also provides opportunities to leak, for example, credentials that could be used in one of the honeypot services. Even though only one honeypot in our set contained this feature, other IT honeypots implemented this feature in some way. John et al. developed a 'heat-seeking' honeypot, that dynamically generates web pages that resemble web pages that are frequently targeted by attackers [61]. Similarly, Mphago et al. tried to maximize attack surfaces in their web application honeypot, as a form of advertisement to attackers [62]. We believe that this feature could be interesting for future research in the attraction of targeted attackers and have, therefore, included it in the topology.

  The tech-level features within the *External Persuasion* feature can be described by the following set:

  $External\ Persuasion = \{Passive$: fictitious company, $Active$: advertisement $\}$

Honeypots	**Implemented protocols**		
	IT	**Remote Access**	**Industrial**
SCADA Honeynet project [26]	HTTP, FTP	Telnet	Modbus/TCP
Digital Bond [53]	HTTP, FTP, SNMP	Telnet	Modbus/TCP
Serbanescu et al. [59]	SNMP, TFTP, XMPP		Modbus, IEC 70870-5-104, DNP3, ICCP
Conpot [44]	HTTP, FTP, SNMP, IPMI, TFTP		BACnet, Guardian AT, Kamstrup, Modbus, S7comm, EtherNet/IP
Pliatsios et al. [48]			Modbus/TCP
DiPot [47]	HTTP, SNMP, IPMI		Modbus, Kamstrup, BACnet, Guardian AST, S7comm
Hilt et al. [34]		VNC	S7comm, EtherNet/IP, Omron FINS
Honeyd [24]	FTP, SMTP, IIS, POP	Telnet	
Dodson et al. [33]	SOAP		S7comm, BACnet, , IEC-104 DNP3, Modbus
HoneyPHY [37]			DNP3
GridPot [49]	HTTP		IEC 61850 GOOSE/MMS, Modbus
Piggin and Buffey [50]	HTTP	SSH, RDP	
CryPLH [46]	HTTP, SNMP		S7 ISO-TSAP
Simoes et al. [54]	SNMP, FTP		Modbus
Thingpot [30]	XMPP, HTTP		Zigbee
MimePot [45]			Modbus/TCP
Bauer et al [39]		SSH	
Vd. Lelie et al. [60]		SSH	
Honware [55]	HTTP, DHCP, UPnP, MDNS, TFTP	Telnet, SSH	
SIPHON [58]	HTTP, RTSP	Telnet, SSH	
Honeytrack [31]	HTTP, TFTP	Telnet	

Table 4.1: An overview of the implemented protocols in the analyzed honeypots.

– **Physical Process**: In contrary to most of the other meta-level features, this feature is very specific to the CPS domain. Since these types of systems have both computational and physical elements integrated, there needs to be a way how modifications in digital components are reflected in the physical world. Therefore, this feature describes the method by which a physical process is incorporated into the decoy system. Targeting a CPS with the intention to disrupt the regulated physical process, requires comprehension of that specific process rather than the sole knowledge of conventional IT-focused skills [63]. Without this knowledge, it is unlikely that an attacker can achieve a complex or undetectable disruption.

  The investigated honeypots each have different methods in which the physical process is considered. For the vast majority of investigated honeypots, a representation of the process has not been taken into account at all. As a result, an attacker does not see a reflection

of any of his modifications done. The absence of a process either modeled or physical does not come across as realistic, especially with an attacker that has disruptive intentions. Even though initial interaction is caught, it could lead to disengagement and thus missing out on the true intentions of an attacker.

Besides the absence of process implementation, some of the considered honeypots have integrated a certain form of process in their deceptive system. In these cases, no real physical process representation, but a model has rather been implemented instead. The definition of a process model can be widely interpretable, however, we describe it as the reflection of the attacker's actions, performed on a system that appears to directly control a process. Following this line of reasoning, a model could consist of very simple state changes that imply that a valve is opened or closed, or whether a door is locked or unlocked. On the other hand, some of the works have implemented more complex physics-based models with the aim to increase realism [37][49][45][50]. Only one of the analyzed honeypots utilized a real plant, in the form of a robotic workstation that could be controlled through a HMI [34].

The tech-level features within the *Physical Process* feature can be described by the following set:

$Physical\ Process = \{absent, process\ model, real\ process\}$, where in terms of cost, $absent < process\ model < real\ process$.

	Size	Interaction level	Virtualization	Services	External Persuasion	Physical Process
Size		i	i	a	i	i
Interaction level	i		a	i	i	i
Virtualization	i	a		a	i	a
Services	a	i	a		i	i
External Persuasion	i	i	i	i		i
Physical Process	i	i	a	i	i	

a:	affect
p:	prevent
i:	independent

Table 4.2: Relationship matrix of meta-level features in Deceptive Functionalities.

Now that all features within the deceptive functionalities have been identified, we aim to identify the relationships between these meta-level features. The relations are relevant for determining the coexistence of certain feature combinations. Table 4.2, shows the relationships between each feature that is part of the deceptive system. The independent relation implies that, after selecting a value for the feature on the vertical axis, all values remain possible for the feature on the horizontal axis. An example, whether the value for *Size* is pot or net, the user's choice for *Interaction Level* remains open.
An affect relation means that picking certain values for the first feature, in some way influences the possibilities for the second feature. When we revert to the previous example but replace *Interaction Level* with the feature *Services*, the affect relation can be noticed. The reasoning is that a net, per definition, requires more services than a pot, due to the communication that takes place between the interconnected devices. More serious affect relations are identified around the

Virtualization feature. When opting for a low-interaction honeypot, using a real device is not possible as was mentioned earlier in this section. Furthermore, controlling a real plant with the corresponding hardware is also not feasible using a service imitation as this type of honeypot does contain the proper physical necessities. These types of limitations are important to consider for Section 4.4, where sets of features are assembled. The elaboration on the remaining relationships can be found in Appendix B.

4.2.2 Deployment-related Functionalities

The functionalities within this feature should provide considerations when aiming to deploy the deceptive system composed in the previous subsection. Even though these functionalities do not directly contribute to the composition of the honeypot, they have a role in determining whether it is likely to catch the desired interaction. The corresponding meta-level features can be found in Figure 4.4, together with the tech-level features that reside on a lower level of abstraction.

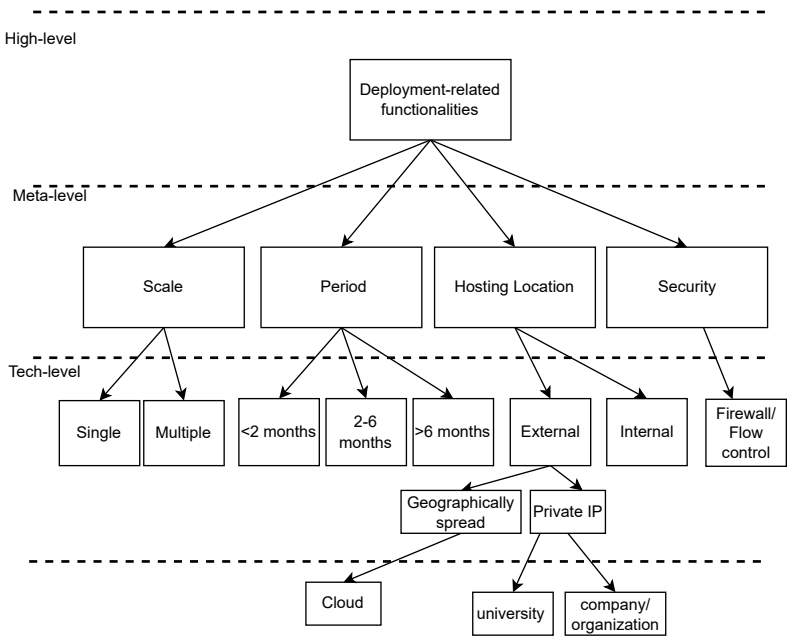

Figure 4.4: Features within the deployment functionality.

– **Scale**: This feature represents the magnitude of the deployment of a deceptive system. One would think that a honeypot would be connected to either the Internet or an internal network, using a single running instance. However, a significant number of investigated honeypots have the possibility to deploy multiple instances of the same decoy. This allows numerous attackers to simultaneously connect with one of the instances, hence increasing the attack surface of the deceptive system. As a result, the chances of being interacted with are significantly higher than when using a single instance. This large-scale deployment started with Honeyd that could host up to 65536 hosts at once already [24].

  It is important to notice that some compositions of deceptive functionalities make it easier or more difficult to deploy more than one instance. When using virtual hosts, in the form of a service imitation, it is possible to create multiple virtual hosts [64]. However, when using a real device, it generally becomes harder or seriously more expensive to deploy multiple honeypots. Dodson et al. still managed to deploy a high number of instances by using proxies for real devices [33]. This way each real device was capable of hosting multiple virtual IP addresses.

The tech-level features within the *Scale* feature can be described by the following set:

$Scale = \{single, multiple\}$

– **Deployment period**: describes the length that the honeypot should be deployed to increase the likeliness or frequency of desired interaction. It is, obviously, not possible to guarantee any form of interaction when a honeypot is deployed for a certain amount of time. However, especially in CPS, real-life attacks have shown to be spread out over months, if not, years [65][66].

From the analyzed honeypots, we distinguish three possibilities for the duration of the deployment. The group with the shortest duration considers a deployment period of fewer than two months. The impact of this relatively short period is illustrated by Serbanescu et al., which collected data for 28 days [59]. Most of the logged interactions resembled reconnaissance activities, where no targeted attacks were captured. The same situation occurred for CryPLH which, after being deployed for a month, did not receive PLC-specific attacks [46].

For the middle value of this feature, a deployment period between two and six months has been considered. The honeypots in our set that fall into this category are SIPHON, DiPot, Bauer et al., and Honeytrack, which showed a significantly higher amount of caught interactions. However, most of this interaction could still be attributed to automated attacks which were noted by lack of manual interaction on, for example, web interfaces.

The last category considers the longest period, with durations over 6 months. During a 7-month deployment, Hilt et al. received, even though not CPS specific, a number of targeted attacks including ransomware and beaconing [34]. Dodson et al. had a 13-month deployment, where only 0.01% of the captured packets were both ICS specific and malicious, including Denial of Service (DoS) and command replay attacks [33].

The tech-level features within the *Period* feature can be described by the following set:

$Period = \{<2\ months,\ 2\text{-}6\ months, >6\ months\}$, where in terms of costs,
$<2\ months <$ 2-6 $months <\ >6\ months$.

– **Hosting location**: describes the network location from which the honeypot is hosted. The tech-level features are extracted by differentiating between the production and research honeypots mentioned in Section 2.2. Research-oriented honeypots are directly accessible from the Internet and therefore hosted on an external network. This external network could be set up from various locations. Serbanescu et al., DiPot, and SIPHON all used cloud service providers to host their honeypot instances [59][47][58]. CryPLH and Honeytrack were deployed from the IP range consistent with the universities involved in the research [46][31]. Hosting the decoy from realistic IP addresses are an important aspect to avoid suspicion. Realistic implies an IP address in the range of the mimicked company/organization, or at least geographically consistent.

Production honeypots, however, are typically placed in an internal network of a running process, with the aim of detecting an intruder. The decoy is then placed close to other machines that the adversary might deem interesting. Unfortunately, these honeypots remain undisclosed more often than not, for the reason of protecting modus operandi. An example is illustrated by Lelie et al. that used a dataset from honeypots that were placed in monitored networks by the Dutch National Cyber Security Centre (NCSC-NL) [60].

The tech-level features within the *Hosting Location* feature can be described by the following set:

$Hosting\ Location = \{internal,\ external$: Cloud, University, Company/Organization, misc.$\}$

– **Security**: describes the method of controlling traffic flow towards and from the honeypot. The main purpose of this feature is to prevent misuse of the decoy, for example as a pivot to

actual production devices on the same network. Furthermore, due to incorrect configuration, a honeypot could be used for malicious activities such as the distribution of illegal material, or to launch attacks on other systems which could have legal implications [67]. Digital Bond used a security mechanism which can be placed in front of a honeypot [53]. This mechanism, named Honeywall, is able to stop the outbound traffic from the compromised honeypot. Simoes et al. also refer to a firewall as containing measure which should allow all incoming traffic but deny any outbound connections which are not made back to the attacker [54].

Besides prevention of misuse, a firewall also provides the opportunity to filter out some of the incoming traffic that the user does not consider relevant, such as automated attacks when aiming to attract a targeted attacker.

The tech-level features within the *Security* feature can be described by the following set:

$Security = \{firewall\}$

Similar to the deceptive functionalities, we analyze the relationships between these meta-level features to make sure that dependencies are identified. For instance, when determining the *Scale* of deployment, one should consider the effects on the *Hosting Location*. A large-scale deployment might not be feasible or realistic on an internal network, but instead, distributed over different geographical nodes. Additionally, the *Hosting Location* determines whether security measures are necessary, to protect potential production devices on the same network as the honeypot.

	Scale	Deployment Period	Hosting Location	Security
Scale		i	a	i
Deployment Period	i		i	i
Hosting Location	a	i		a
Security	i	i	a	

a:	affect
i:	independent

Table 4.3: Relationship matrix of meta-level features in Deployment-related functionalities.

4.2.3 User-related Functionalities

Whereas the deceptive and deployment-related functionalities mainly focused on creating and launching a system that adversaries could target, the user-related functionalities should ensure that all relevant information is extracted from the decoy. Figure 4.5 contains an overview of features within the user-related functionalities.

– **Logging**: describes the method in which interaction with the deceptive system is being captured and made available to the user. In theory, all actions performed by a remote attacker could be considered loggable through network traffic. However, in practice, encryption could prevent the user to interpret the use of any techniques. Also, one could prefer to see the result of the adversaries' actions on the targeted system, for example, the creation of some malicious process. There is a large difference between network-based and host-based logging. The latter becomes especially relevant when the attacker has full access to the device's OS,

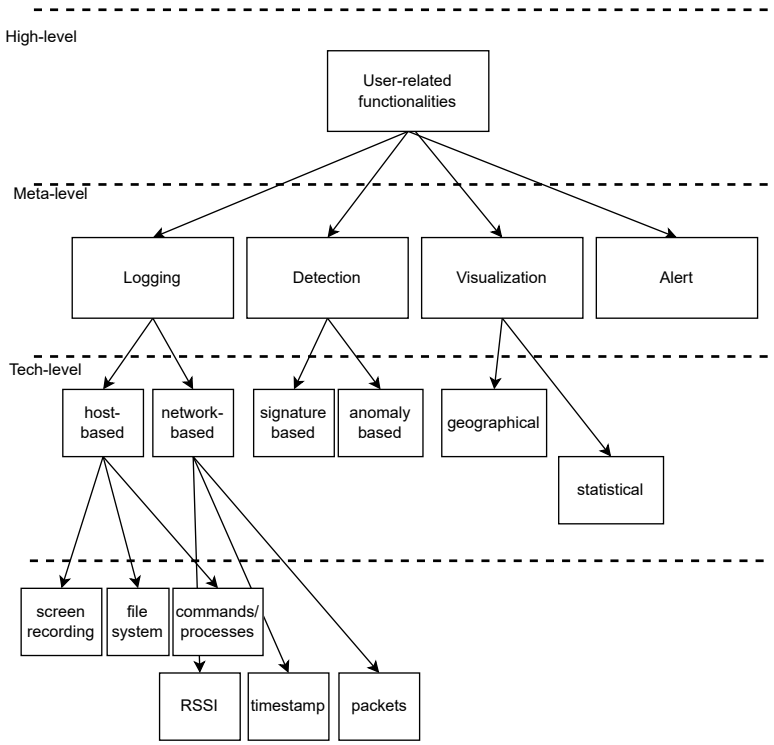

Figure 4.5: Features within the user functionality.

i.e., possibilities for file modification and execution. It also means that we are more likely to see host-based logging methods in device imitations or real devices. This expectation is confirmed by the honeypots that we have investigated. Hilt et al. used VirtualBox screen recording, where the recording was started when there had been an indication that someone accessed the honeypot [34]. Digital Bond implemented a rootkit named Sebek in its VM, which is able to log keystrokes, commands executed, and system calls [53]. Honware modified an existing kernel function to log programs invoked by the attacker [55].

For the majority of honeypots, which are service imitations, the possibility to implement host-based logging is very limited. The reason is that most service imitations are low-interaction, and thus do not have a functional file and/or operation system. Consequently, there are generally not many options to integrate a functionality that registers modifications on such system. Therefore, we observe a shift towards network-based logging in that category. The logging functionality is being integrated into the service imitation script, resulting in captured application layer data with additional metadata as output for the user. Conpot, for example, registers the type of request or response regarding the implemented protocol for each incoming session [44]. On top of that, timestamps, source and destination IP addresses, and port numbers are logged as well. Furthermore, tools such as Wireshark, TCPdump, and TShark were seen to be commonly implemented for capturing incoming packets.

Besides application and network layer data, it is also important to consider other connection properties that could provide useful information regarding an attack. For example, the Received Signal Strength Indicator (RSSI) in wireless networks, which in case of close access could supply information about the location from where the attack is launched. Even though none of the honeypots in our set considered this property, there are other (non-CPS) honeypots that took RSSI into account [68][69].

The tech-level features within the *Logging* feature can be described by the following set:

$Logging = \{network\text{-}based, host\text{-}based\}$

- **Detection**: describes the functionality to detect specific attack methods within interaction with the honeypot. This feature, therefore, extends further than logging, which simply ensures that the data itself is captured and stored. We distinguish different types of detection, signature-based and anomaly-based IDS. The first mentioned scans through a live set of activities and attempts to find matches with another signature that already exists in the signature database [70]. Serbanescu et al., Digital Bond and HoneyPHY all implemented Snort, which is the most used signature-based IDS because of its open-source software [71].

  Anomaly-based detection relies on deviations with respect to regular operation. In order to implement this type of detection, a normal model of the behavior of a computer system is created. Whenever the live set of activities deviates a certain amount from this normal model, it is considered an anomaly which should trigger an alarm for the user. Any significant deviation between the observed behavior and the model is regarded as an anomaly, which can be interpreted as an intrusion. GridPot implemented this type of detection in an interesting manner by focusing on anomalies in the physical process instead of in the network traffic [49]. Besides GridPot there is other recent literature that investigates this physics-based anomaly detection, specifically in the CPS domain [72][73].

  The tech-level features within the *Detection* feature can be described by the following set:

  $Detection = \{signature\text{-}based,\ anomaly\text{-}based\}$

- **Visualization**: describes the method in which the captured data is presented to the user. Visualization methods by giving the user a quick and perceptible overview of the collected information. DiPot implemented a data visualization interface where the user can view a world map with geographical indications of incoming connections to the honeypot nodes [47]. Furthermore, statistics regarding protocol and source IP address distribution were presented in a visual way. Lelie et al. focused on visualizing SSH honeypot data which resulted in a dashboard that could assist security analysts in analyzing a large amount of data [60].

  The tech-level features within the *Visualization* feature can be described by the following set:

  $Visualization = \{geographical, statistical\}$

- **Alert**: describes the method in which the user is alerted when relevant interaction has occured. This feature could be considered as supplementary to the *Detection* feature. However, it focuses more on when and how security/alarm messages are generated rather than the detection of malicious activity. Consequently, an alert action should only be triggered when an event is deemed critical.

  Simoes et al. used specific security event messages based on a standard data format designed for IDS [54]. Furthermore, Digital Bond offered report/alerting preferences in the Honeywall manager [53].

The relationship matrix for within the user functionalities is relatively straightforward and can be found in Table 4.4. The implemented *Logging* feature heavily influences the possibilities for the other features, as those depend on the captured information.

	Logging	Detection	Visualization	Alert
Logging		a	a	a
Detection	a		i	a
Visualization	a	i		i
Alert	a	a	i	

a:	affect
i:	independent

Table 4.4: Relationship matrix of meta-level features in User-related functionalities.

4.3 Observation Targets

This section describes the formation of observation targets that we use for the development of our methodology. By doing this, we aim to answer the second research sub-question, *"What are the known attacker TTPs that are used in IoT, IIoT and CPS networks and how can they be translated to observation targets?"*. We define observation targets as specific interactions, caused by the adversary, that the user aims to capture. In Subsection 4.3.1, we discuss how structure can be applied to cyberattacks, using well-known attack frameworks. In addition, we explain why these frameworks are relevant for our research. The final Subsection 4.3.2 then discusses how we use the earlier obtained information to extract our observation targets.

4.3.1 Structuring an Attack

The Cyber Kill Chain can be used to model, recreate or analyze the offensive actions of the attacker in the IT domain [74]. To apply structure, the kill chain breaks an attack down into subsequent stages that each have a specific goal. CPS, however, are different types of systems compared to the traditional IT infrastructure. Consequently, CPS also attract attackers with different objectives. Therefore, the SANS Institute introduced a modified version of the original Cyber Kill Chain, specifically for ICS which we consider part of CPS [43]. The ICS version of the kill chain can be found in Figure 4.6. Compared to the traditional version, The difference lies in the addition of a second stage after the intrusion has taken place. The reason behind the multi-staged approach is that the CPS-specific part is generally initiated quite sometime after the first, intrusion, stage. The intrusion stage itself has been left unmodified because even in CPS attacks, the first point of entry generally resides in the IT part of the organization's network, as has happened with real-world attacks that involved BlackEnergy3, Stuxnet, and Triton [2][75][76]. Upon the final step of the intrusion stage, attackers are able to collect detailed information about the inner workings of the CPS to craft a tailor-made attack for the device types and software of that CPS.
The relevance of the kill chain for our research lies in the translation from the user's objective to the observation targets. For example, a user that intends to perform statistical research on port scanning activity needs to consider other stages of the kill chain, compared to someone that aspires to learn details on how attackers deliver their second stage malware.
To use the kill chain, we need to identify which phases are in fact observable from the target system's point of view. The red lining in Figure 4.6 indicates whether this phase is observable from the targets' point of view. For example, the Weaponization phase, where a specific type of malware is being developed, cannot be detected by the defender. In the Delivery phase, however, the conveyance of this piece of malware onto the defenders system can be observed. The identified observable phases will form the highest abstraction layer of observation targets in our research.

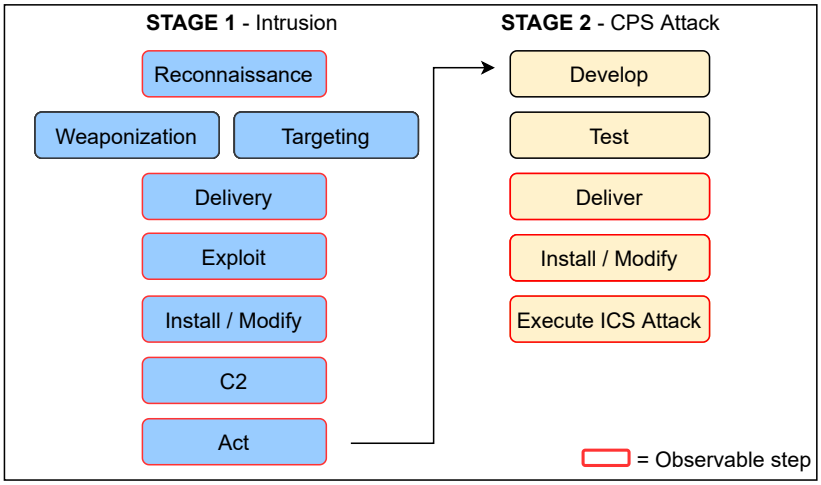

Figure 4.6: ICS Cyber Kill Chain with red contours that represent the observable phases.

Another acclaimed method for deconstructing and analyzing attacks is the MITRE ATT&CK framework, as has been mentioned in Section 2.3. Whereas the Cyber Kill Chain provides a series of subsequent phases, the ATT&CK matrix consists of lists of tactics that could be used by the adversary during an attack. The MITRE framework, therefore, resides on a lower level of abstraction. We specifically use the ATT&CK framework for Industrial Control Systems [42]. From this framework, we first select the tactics that are relevant for each phase in the kill chain and that can be perceived by a honeypot system. Subsequently, we do the same for each technique that is listed within the tactics. For example, a tactic that can be applicable to honeypots is *'Initial Access'*. Within that tactic, there are multiple techniques available. The 'Exploit Public-Facing Application'-technique is very relevant for honeypots, as it describes how the adversary may leverage weaknesses to exploit software reachable from the Internet. Techniques that do not appear to be suitable or relevant for a honeypot are omitted. After performing these steps for each technique in the ATT&CK matrix, the following techniques have been excluded:

- Drive-by Compromise *(T0817)*
- Supply Chain Compromise *(T0862)*
- Screen Capture *(T0852)*

The reason for excluding the first two techniques is that they are beyond the research scope. Our research only considers server honeypots, whereas 'Drive-by Compromise *(T0817)*' would be suitable for client honeypots. Furthermore, we consider 'Supply Chain Compromise *(T0862)*' too broad for our research as we aim for more direct forms of attack by the adversary. Finally, 'Screen Capture *(T0852)*' has been omitted because neither the facilitation nor the logging of the technique are accessible by the honeypot.

Reconnaissance
Active Scanning *(T1595)*
Gather Victim Org Information *(T1591)*
Phishing for Information *(T1598)*
Wireless Sniffing *(T0887)*

Table 4.5: Categorization of MITRE techniques in the Reconnaissance phase of the Cyber Kill Chain.

4.3.2 Formation of Observation Targets

Now that we have a selection of relevant techniques, we categorize them into observable phases of the Cyber Kill Chain. The reason is that it eventually creates structure in the large matrix that will be used for the mapping. Consequently, it is easier, for someone that intends to construct a honeypot, to translate his objective into the necessary observation targets. This categorization is possible because of the different layers of abstraction that the frameworks reside on. Therefore, an attacker could use different techniques during a phase of the kill chain. For instance, when considering the Reconnaissance step, four techniques can be categorized in that phase, as shown in Table 4.5.
It is important to understand that a technique is not necessarily limited to one specific phase of the Cyber Kill Chain. There are multiple examples of techniques that could occur in more than a single attack phase. One of those is 'Valid Accounts' (T0859), which we have placed in the 'Exploit' phase as a user has been exploited and the attacker gained a foothold in the target system. However, one could rightly argue that this technique could also be relevant in the 'Act' phase, where the attacker could utilize the credentials to move to another part of the network. Nonetheless, for readability purposes, each technique only occurs once in the matrix that will be used for the mapping of observation targets with features. Therefore, those specific techniques have been assigned to the earliest phase of the kill chain that they occur in. With the list of relevant and structured techniques, we can start investigating how to incorporate these into a honeypot. To do that, we aim to answer the following questions for each technique:

- *What functionality is required to ensure that the attacker could apply a technique?*
- *What functionality is required to ensure that the user could observe the use of that technique?*

We extract the information necessary to answer these questions from both the existing honeypots that managed to capture such techniques and the ATT&CK framework that supplies information about real-world examples and their detection. Having answered the questions, we know what each technique entails, what deceptive functionalities are required, and what is needed to log the technique. The answers together form the basis for our observation targets. This method can be considered an intermediate step toward the mapping. It also prevents a narrow field of vision by forcefully coupling an observation target with the existing fixed set of features that has been assembled in Section 4.2. The example below shows how the information about the relevant target is annotated.

- **Active scanning *(T1595)***
  To enable: Service(s) running on one or multiple ports, which an external IP address can communicate with.
  To log: Code/software that captures network traffic or registers incoming connection requests from source IP to specific ports of the host.

The first line provides a brief description of the required deceptive functionalities. In the example, it concerns a running service that is exposed to and reachable from the Internet. The second line merely defines that there should be a functionality that captures an incoming connection request. This annotation has been assembled for all observation targets and can be found in Appendix C.

4.4 Mapping

Even though the topology in Section 4.2 provides detailed information about characteristics that need to be contemplated when constructing a honeypot, it is not yet suitable for a one-to-one mapping against the observation targets that we acquired in the previous section. The reason is that, while some features increase the likelihood of attracting attackers, or enhance the processing of captured interaction, they are not strictly necessary to capture the particular interaction that we have called an observation target. For instance, deployment functionalities such as *Period* and

Scale do not determine whether an observation target can be enabled and captured. Furthermore, *Detection*, *Visualization*, and *Alert* are nice-to-haves, but not decisive in the facilitation or logging of any of the observation targets. Therefore, we distinguish functional features from non-functional ones, where we use the former to construct the mapping. Figure 4.7 shows the functional features, where it can be seen that the deceptive functionalities have remained intact. For the user-related functionalities, only both logging features have been included.

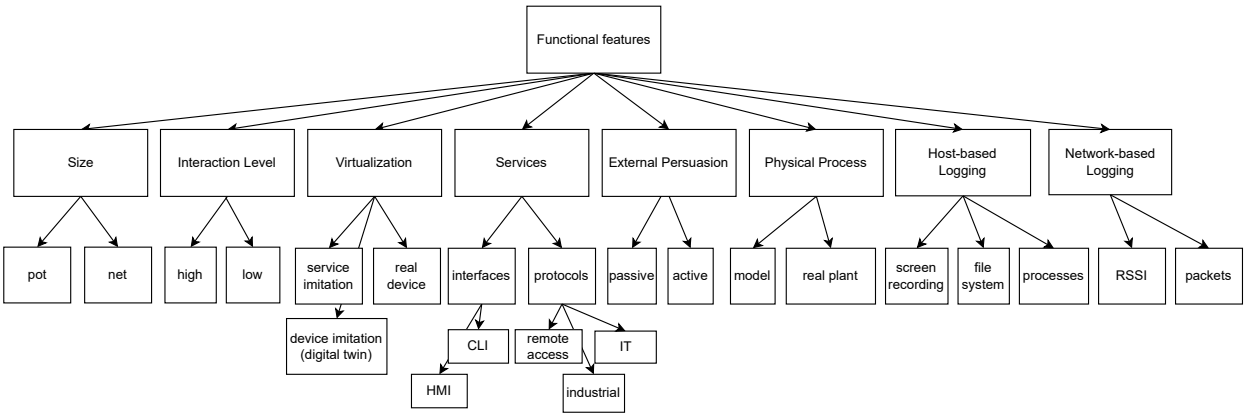

Figure 4.7: Selection of functional features that can be used for the mapping.

The result of the mapping can be found in Table 4.6. The rows of the table consist of the observation targets, grouped in the labeled kill chain phases. For readability purposes, each observation target is represented by its MITRE technique number. An overview of all technique numbers with corresponding names can be found in Appendix C. The columns of the table present the same functional features as shown in Figure 4.7.

		Size		**Inter.**		**Virt.**			**Services**					**Ext.Per.**		**Ph.Proc.**		**Host**			**Netw.**	
		pot	net	high	low	service imitation	device imitation	real device	HMI	CLI	remote access	industrial	IT	passive	active	model	real plant	screen recording	file system	processes	RSSI	packets
Recon.	T1595	x			x	x					x	x	x									x
	T1591	x			x	x							x	x	x							x
	T1598	x			x	x							x	x	x							x
	T0887	x		x				x				x	x									
Delivery I	T0865	x			x	x							x	x	x							x
	T0822	x			x	x				x	x											x
	T0883	x			x	x			x	x	x	x	x									x
	T0847	x		x				x											x	x		
	T1200		x	x				x														x
	T0864		x	x			x															x
	T0848		x		x		x					x										x
Exploit	T0860	x		x				x				x	x								x	x
	T0866	x		x		x					x											x
	T0819	x		x			x		x		x	x										x
	T0807	x			x	x				x	x	x	x					x		x		x
	T0823	x			x	x			x			x	x					x				x
	T0871	x		x			x					x										x
	T0834	x		x			x													x		
	T0853	x		x		x				x										x		x
	T0863	x		x		x													x	x		
	T0859	x			x	x			x	x	x	x		x	x			x				x
	T0812	x			x	x			x	x		x	x					x				x
	T0858	x		x			x													x		

Continued on next page

Table 4.6 – continued from previous page

		Size		Inter.		Virt.			Services						Ext. Per.		Ph. Proc.		Host			Netw.	
		pot	net	high	low	service imitation	device imitation	real device	HMI	CLI	remote access	industrial	IT	passive	active	model	real plant	screen recording	file system	processes	RSSI	packets
Install/Mod. I	T0857	x		x				x												x		
	T0839	x		x				x												x		
	T0874	x		x			x												x	x		
	T0873	x		x			x												x			
	T0872	x		x		x													x			x
	T0849	x		x		x													x	x		x
	T0851	x		x			x													x		
C2	T0885	x		x		x				x	x	x	x									x
	T0884	x		x		x				x	x	x	x									x
	T0869	x		x		x				x	x	x	x									x
Act	T0840		x		x	x				x		x	x							x		x
	T0842		x		x	x						x	x							x		
	T0846		x		x	x				x		x	x							x		
	T0868	x		x			x															x
	T0877	x		x		x														x		
	T0802		x	x			x			x		x							x	x		x
	T0861		x	x			x			x		x										x
	T0811	x			x	x													x			x
	T0801	x			x	x			x	x		x				x						x
Del.II	T0867		x	x			x			x			x					x	x			x
	T0843	x		x		x						x										x
	T0886	x			x	x			x	x	x	x										x
Install/Mod. II	T0830		x		x	x					x	x	x									x
	T0836	x			x	x			x	x		x	x									x
	T0806		x	x			x					x				x						x
	T0835	x		x			x												x			
	T0805	x		x		x					x	x	x									x
	T0889	x		x			x					x								x		x
	T0856		x		x	x						x				x						x
	T0809	x		x			x			x									x	x		
	T0814	x			x	x					x	x	x									x
	T0816	x			x	x			x	x	x	x								x		x
	T0881	x		x			x			x	x									x		x
	T0855	x			x	x						x										x
	T0838	x		x			x					x								x		x
	T0878	x		x			x					x								x		x
	T0803		x	x				x				x										x
	T0804		x	x				x				x				x						x
ICS att.	T0879	x		x				x				x					x			x		
	T0813	x			x	x			x	x		x				x				x		x
	T0815	x		x		x			x			x				x						x
	T0882	x			x	x						x				x						x

Table 4.6: Mapping of observation targets to honeypot features.

The mapping clearly shows that the relevance of certain features depends on the phase that an attack is in. External persuasion seems most relevant in the early phases, where an attacker is still aiming to gain information, or when credentials need to be leaked in order to provide the attacker access to the deceptive system. Contrarily, any representation of the physical process appears to become increasingly important as the second stage of the Cyber Kill Chain progresses. Furthermore, the table shows that, when aiming to facilitate techniques within the first installation phase, a high-interaction pot needs to be constructed, which certainly requires some form of host-based logging to capture the observation target. Besides these rather explainable relationships, some entries may be cause for confusion. For instance, there is one technique that has not been mapped to any of the logging features, which is 'Wireless Sniffing *(T0887)*'. The absence of

logging features lies in the fact that the application of this technique is likely to remain unnoticed by the user. Wireless sniffing is a passive activity that does not generate any data or trace on the target system [77]. The reason for still involving the techniques in the mapping is that it plays a significant role in other techniques that are plausible to follow or precede it. Therefore, one would want to make sure that an attacker could practice such undetectable technique, to enable an attack path that involves other techniques that are detectable. Concretely, this means that a subsequent technique could be 'Wireless Compromise *(T0860)*', which is part of the Exploitation phase and can be captured.

Moreover, for some observation targets, multiple values within the same meta-level feature are selected. This has been done because the observation target is not specific enough to attribute it to a single value. A clear example can be found in the first entry of Table 4.6, where for 'Active Scanning *(T1595)*', all three types of protocols have been selected. Since any type of service implemented on an exposed port could be scanned from the Internet, all of them need to be considered. Therefore, in those cases, the environment, determined by the use case, ultimately decides which of the options is most suitable.
To show how we can use Table 4.6, we composed a simplified attack path, which can be seen in Figure 4.8. The attack path, of which the first two observation targets have been discussed earlier, needs to be translated to a single set of features that form the basis for the construction of a honeypot.

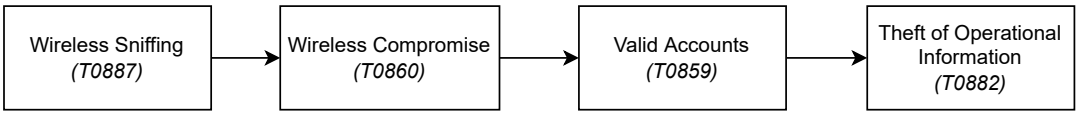

Figure 4.8: An exemplary combination of observation targets that form an attack path.

When feeding each individual observation target to the mapping, we obtain the following sets of features:

Wireless Sniffing *(T0887)*: Size:*pot*, Interaction:*high*, Virtualization: *real device*, Protocols: *industrial, IT*.

Wireless Compromise *(T0860)*: Size:*pot*, Interaction:*high*, Virtualization: *real device* Protocols: *industrial, IT*, Logging: *RSSI, packet capturing*.

Valid Accounts *(T0859)*: Size:*pot*, Interaction:*low*, Virtualization: *service imitation*, Interfaces: *HMI, CLI*, Protocols: *remote access, industrial, IT*, External Persuasion: *active, passive*, Logging: *screen recording, packet capturing*.

Theft of Operational Information *(T0882)*: Size:*pot*, Interaction:*low*, Virtualization: *service imitation*, Interfaces: *command-line*, Protocols: *IT/Remote Access*, Physical Process: *model*, Logging: *packet capturing*.

The next step is combining the sets of the desired observation targets into a single set that facilitates all of them. In the case of the same values for the meta-level features, it is straightforward to select the matching features. However, whenever the values vary, the feature with the highest cost should be selected for the reason that the ability of the costliest option includes the option with lower costs. For example, for the feature *Size*, a net includes pots. Regarding *Virtualization*, when using a real device, atleast the same services have been implemented compared with using a service imitation. Finally, when opting for high interaction, it includes at least the possibilities that low interaction has to offer, with additional quality. By applying this approach to the previously acquired sets, we can produce the following combination of features:

Combination *(T0887, T0860, T0859, T0882)*: Size:*pot*, Interaction:*high*, Virtualization: *real device*, Interfaces: *HMI, CLI*, Protocols: *remote access, industrial, IT*, External Persuasion: *active, passive*, Physical Process: *model*, Logging: *packet capturing*.

The feature set forms a solid basis for the construction of a honeypot that captures the selected observation targets. The final step would be to integrate environmental parameters that should be extracted from the use case. These parameters further decide which protocol should be implemented or whether an HMI and/or CLI is applicable. For example, when the attack path in Figure 4.8, the use case contains some replay attack such as the Polish tram incident, we can only select the *industrial* protocol from the feature set [78]. The same principle holds for the interface types, where observations targets such as 'Valid Accounts' alone cannot determine whether credentials should be entered through an HMI or CLI. A more extensive and real-world example of the construction of a honeypot will be presented in Section 5.2 by means of a use case.

Chapter 5

Results

We have developed a methodology by creating a topology and using parts of it to map against interactions that attackers could carry out. Now we ought to demonstrate that the mapping is correct and that the methodology is suitable for practical application. Therefore, this section is split into Section 5.1, which treats the constructed mapping, and Section 5.2, which demonstrates a real-world use case to show the practicality of our methodology.

5.1 Validation

To show that the mapping is representative, a set of seven honeypots, which have not been used earlier in this research, has been composed. A higher number of honeypots would not necessarily result in the validation of a larger part of the mapping, as the vast majority of the available research captures the same observation targets. Many of our observation targets, such as T0865, T0847, T1200, and T0864 to only name a few, have not even been found implemented in any available CPS honeypot. Hence, increasing the number of works would not increase the number of validated observation targets.

Honeypots	**Observation Targets**
Bodenheim [79]	Active Scanning *(T1595)*, Default Credentials *(T0812)*
Mashima et al. [80]	External Remote Services *(T0822)*, Remote Services *(T0886)*, Network Sniffing *(T0842)*, Monitor Process State *(T0801)*, Unauthorized Command Message *(T0855)*, Manipulation of Control *(T0813)*
iHoney [81]	Remote Services *(T0886)*, Graphical User Interface *(T0823)*, Network Sniffing *(T0842)*, Data Destruction *(T0809)*, Unauthorized Command Message *(T0855)*, Manipulation of Control *(T0813)*
Antonioli et al. [82]	Active Scanning *(T1595)*, External Remote Services *(T0822)*, Remote Services *(T0886)*, Scripting *(T0853)*, Man in the Middle *(T0830)*, Denial of Service *(T0814)*, Manipulation of Control *(T0813)*
HosTaGe [83]	Active Scanning *(T1595)*, Internet Accessible Device *(T0883)*, External Remote Services *(T0822)*
Belqruch et al. [84]	Active Scanning *(T1595)*, External Remote Services *(T0822)*
S7commTrace [85]	Active Scanning *(T1595)*, Remote Services *(T0886)*, Program Download *(T0843)*, Device Restart/Shutdown *(T0816)*

Table 5.1: Set of honeypots with their observation targets that have been used for validation.

Rather than aiming to validate the entire mapping, we aim to demonstrate that the mapping performs as intended using honeypots that were not considered before in the development of the

methodology. Table 5.1 contains the works that have been used for comparison together with the observation targets they claim to facilitate. The works have been selected based on their diversity in identified observation targets, to prevent an overrepresentation of honeypots that solely capture scanning activities.
To see how our mapping performs, first combine the features of each honeypot, derived from their observation targets to a single set, as was done with the example in Section 4.4. This combination is represented by the *Mapping*-label in Table 5.2. Then, we deconstruct the honeypot, based on the architectural description that has been given in the corresponding research paper. This set of features is specified using the *Actual*-label in Table 5.2.

	Size		**Inter.**		**Virt.**			**Services**					**Ext. Per.**		**Ph. Proc.**		**Host**			**Netw.**	
	pot	net	high	low	service imitation	device imitation	real device	HMI	CLI	remote access	industrial	IT	passive	active	model	real plant	screen recording	file system	processes	RSSI	packets
Bodenheim [79]																					
Mapping:	x			x	x			x	x	x	x	x									x
Actual:	**x**		**x**				**x**	**x**			**x**	**x**									**x**
Mashima et al. [80]																					
Mapping:		x		x	x			x	x	x	x	x			x						x
Actual:		**x**		**x**	**x**				**x**	**x**	**x**				**x**						**x**
iHoney [81]																					
Mapping:		x	x			x		x	x	x	x	x			x		x	x	x		x
Actual:		**x**	**x**				**x**	**x**	**x**	**x**	**x**	**x**			**x**			**x**	**x**		**x**
Antonioli et al. [82]																					
Mapping:		x	x		x			x	x	x	x	x			x				x		x
Actual:		**x**	**x**		**x**			**x**	**x**	**x**	**x**				**x**				**x**		**x**
HosTaGe [83]																					
Mapping:	x		x		x			x	x	x	x	x									x
Actual:	**x**		**x**		**x**			**x**	**x**	**x**	**x**	**x**									**x**
Belqruch et al. [84]																					
Mapping:	x			x	x				x	x	x	x									x
Actual:	**x**			**x**	**x**				**x**	**x**											**x**
S7CommTrace [85]																					
Mapping:	x		x		x			x	x	x	x	x							x		x
Actual:	**x**		**x**		**x**				**x**		**x**										**x**

Table 5.2: Comparison between the honeypot features expected through the mapping and the actual features.

Five out of the seven works match very well with the features that our mapping has produced. Regarding the other two honeypots, Bodenheim and iHoney, differences can be noticed within the Interaction and Virtualization feature. Whereas Bodenheim used a real device, which we have linked by definition to high interaction, our mapping indicates that only a service imitation was required to capture their observation targets. This observation raises the suspicion that Bodenheim might have developed a decoy with costlier choices than necessary, through our definition of cost. Upon investigating, this suspicion is strengthened by the fact that the honeypot only investigated fingerprinting by scanning using Nmap and exposed a web interface through the HTTP service which displayed device-specific characteristics. Concerning iHoney, our mapping indicates that for *Virtualization* a device imitation would suffice for the targets it aimed to capture, while the authors decided to use real devices. Our mapping output was caused by the 'Data Destruction' target that requires a functional file system with an underlying OS, but not necessarily a real device. The research mentions that the ICS system was implemented with help from a specialized contractor, which may indicate that they already had access to physical devices.

The key takeaways from the rather brief validation are: if the mapping suggests that a costlier feature is needed, while researchers accomplished it with something less costly, the mapping is clearly incorrect. This, however, has not been the case in our research. Nonetheless, when mapping suggests a less expensive solution, while researchers accomplished it with a feature with higher

costs, it simply indicates that they used a more costly solution than what might have been required. Therefore, our mapping contributes to the suggestion that their observation targets could be captured with fewer costs. However, it is important to understand that in practice there might be other reasons to pick a certain composition. For example, if an organization or individual is already in possession of a real PLC device, it would be preferable to use that device for a honeypot, rather than developing their own service or device imitation. Moreover, virtualization in ICS is not trivial [86]. Therefore, buying a real device might be cheaper than developing the virtualization techniques needed.

5.2 Use Case: Building Automation Access Control

The previous sections have shown why the framework is needed, how it has been built and whether the mapping is consistent when compared with a different set of existing honeypots. Now, finally, we aim to demonstrate that the framework is suitable in terms of practicality. A real-life use case is presented where there is demand for a honeypot in a specific scenario. First, we explain the scenario of the use case, which embodies the objective of the user and the environment that will be considered. Then the framework is instantiated by applying the produced methodology. In other words, by using the developed approach, we guide the user from its objective to a set of features that together form the basis for a honeypot. To take it even further, we implement the identified features to realize a functional honeypot. Ultimately, we close the circle by demonstrating that the honeypot is able to detect the observation targets and therefore meets the objectives of the user.

5.2.1 Scenario Description

A reputable European government agency has implemented an access control mechanism in its building automation system. This mechanism allows personnel to use a smartcard for entering specific spaces in the building. Security officers are able to monitor the state of offices to see which doors are open or locked and to collect logs through a workstation that is directly connected to the Building Management System (BMS). Furthermore, engineers have access to an HMI, which permits them to regulate access by manually override the state of office doors in case of maintenance or emergency. The BAS uses a network that is mainly local and from the automation layer downwards decoupled from the IT infrastructure of the organization. The structure of this network is shown in Figure 5.1 where, for visibility purposes, only a single instance of each field device type is displayed.

The internal network is not strictly air-gapped, however, as the service contract for the BAS involves polling the Internet for updates periodically. This means that the workstation has internet access to provide the BMS with updates when necessary. Furthermore, the computers in the management layer are also coupled with the IT infrastructure to exchange information on whether a smartcard associated with person X is or is not authorized to unlock the door of a specific room. The devices in the management layer are used to control the HMI as well as to configure the building controller. In the automation layer, the HMI allows valid users to directly control the system using clickable buttons. The building controller functions as a gateway that translates commands from IP-based protocols to field protocols, and vice versa, sensor and state information from field protocols to IP-based protocols. Additionally, the computers and PLCs in the automation and field layer are spread throughout the building, and not necessarily limited in physical access. Based on the described system architecture, a concern has been raised for the manipulation of the access control mechanism by potential intruders. The organization is worried that attackers might have access to a building automation controller and disrupt the physical process by manually locking or unlocking specific doors. For example, incorrect configuration within the network could result in exposed HMIs and occurs regularly in practice [87]. The organization wants to be able to make a distinction between different types of attackers in terms of knowledge and capabilities. More

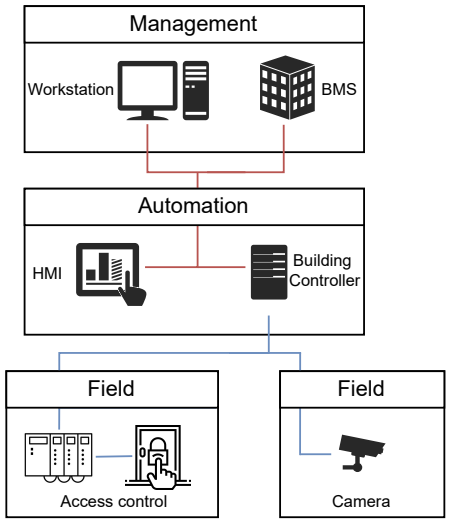

Figure 5.1: Overview of the organization's building automation system, where the red and blue interconnections represent IP-based and field protocols, respectively.

specifically, they would like to have a solution that is able to investigate if different approaches are being used to reach a similar goal, where each approach requires a different particular skill. This requirement will have a significant role in the derivation of observation targets in the next section.

While there are lots of honeypots available on the market, the organization could not find any that meets their specific objective and that fits in their environment. There is no existing plug and play solution to address the specific demands of the government agency. More than that, there is no systematic methodology available on how to compose an effective honeypot for a specific need. As mentioned in Chapter 3, existing solutions lack the ability to couple the objective of an arbitrary user to the required composition of the honeypot. Therefore, we use the developed methodology to construct a honeypot tailored to the user's objective.

5.2.2 Application of the Methodology

In order to apply the framework to the given use case, the required steps need to be specified. The first step is to extract particular observation targets from the scenario discussed in 5.2.1. We do this by defining attack paths that can be deconstructed into observation targets, that are aimed to be caught. In the second step, the obtained mapping is used to identify which features should be supported to facilitate and register that target. The user is left with multiple sets of features at the end of this stage, depending on the number of observation targets that have been used in the mapping. In the third and final step, the sets are combined into a single one, that supports the facilitation of all involved observation targets. Furthermore, the environment parameters are considered in this stage as well, as they determine the further details that are required for the implementation of the honeypot. Figure 5.2 shows the framework in action, instantiated by a use case that can be dissected in the following two elements:

- **Objective**: Detection of a potential intruder that has the intention to disrupt the physical process. Being able to detect attackers with different capabilities by facilitating multiple approaches to reach the goal. Consequently, the framework should output a group of features that can provide all approaches that will be specified.

- **Environment**: Parameters from the target environment that are needed to develop the honeypot, such as the device type, services used, and other properties of the network where

the device is located. Due to security limitations stated by the organization, it has not been possible to disclose the device type and number that is used in their network. Therefore, in the implementation phase, a different device type but with similar services has been used.

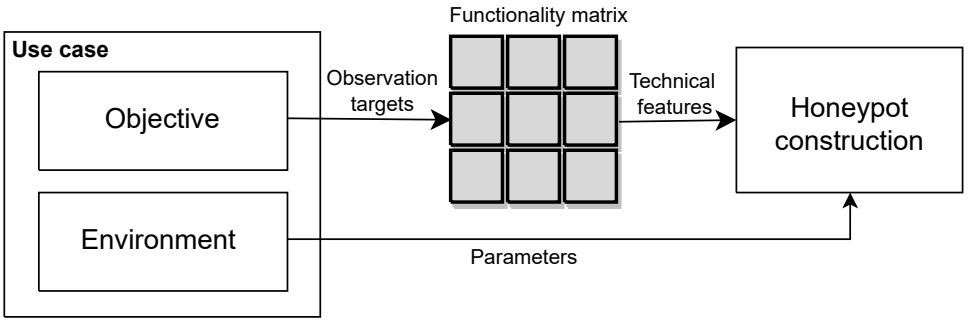

Figure 5.2: Framework in action employing a use case.

From Objective to Observation targets
The first step in applying the methodology consists of the derivation of observation targets from the use case in the previous Subsection 5.2.1. In the scenario, we aim to enable multiple approaches that could result in the manipulation of control. For this step in the methodology, it means that there will be multiple attack paths, consequently, resulting in distinct observation targets. The first path should be fairly approachable for an ordinary user, i.e., a regular cybercriminal. The regular cybercriminal should not be required to have very technical or environment-specific knowledge and skills. For the second path we design a path where the majority of attackers disengages due to lack of knowledge, and attacking the system is possible only for an advanced cybercriminal. At the same time, there needs to be an incentive for an attacker that has skills, to takes this approach. Hence, the reward for taking this path is higher, in the form of the ability to unlock a door that is not controllable from the HMI.

Path 1
The first attack path that we consider is through an exposed HMI. In the scenario, the user interface has become directly accessible from the Internet, due to improper network configuration in the management layer. Using HTTP in the web browser, arbitrary users can access the HMI when in possession of the corresponding IP address and port number. Furthermore, it is also possible for insiders to reach this device through the internal network. An attacker that solely visits the Web address, however, does not match the intentions that we aim to capture. Therefore, the HMI provides an overview of available actions that the user can perform to initiate a physical operation such as unlocking a door. The actions on the HMI are then registered to distinguish "accidental visitors" from users with malicious intent. Though the HMI is accessible for all users, valid credentials are required to allow modification of parameters. This adds another layer of security that the adversary needs to overcome. Also, it allows for future extension of the honeypot by having the possibility to leak specific sets of credentials on arbitrary locations. The complete combination of observation targets, together composing the first attack path can be found in Figure 5.3.

The next step for this attack path will be the translation to the required features for the honeypot that will be constructed.

Path 2
For the second attack path, a port of the building controller that uses an industrial protocol is

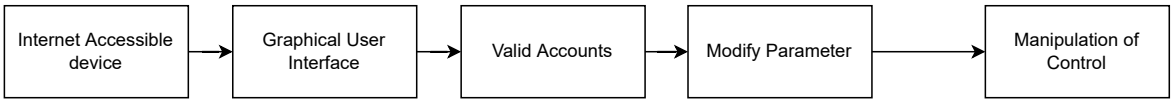

Figure 5.3: First path: attack through an exposed HMI.

exposed to a potential attacker. This protocol facilitates the modification of object parameters through the use of commands, resulting in changes within physical processes. In this scenario we assume that no authentication mechanism has been implemented, meaning that any user is allowed to send arbitrary commands to the building controller. Such a scenario matches real-life situations where these mechanisms, despite their availability nowadays, are widely omitted in practice [88] [89]. Whereas the modification of parameters on an HMI happens visually, industrial protocols use particular syntax that requires specific knowledge about that service. Therefore, we state that manipulation of control through this path, which is shown in Figure 5.4, requires more knowledge than through the path presented in Figure 5.3.

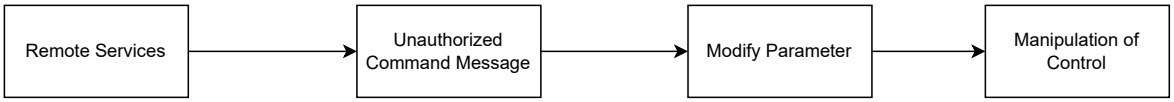

Figure 5.4: Second path: attack through an unauthenticated industrial protocol.

Similar to the previous path, each individual observation target will be translated to a set of required features.

From Observation targets to Features
Now that the objective of the user has been translated to the specified observable actions, the mapping that has been presented in Section 4.4 can be used to acquire the technical features for the organization's honeypot. Firstly, we list the feature set that corresponds with each observation target for both paths. Then, we analyze the sets to determine which similarities or differences are present within the meta-level features. Based on these matches we combine them into a single set, that is able to enable and log the affected targets. Finally, the environment is considered, which determines the implementation details such as specific services or device types that will be used in the next phase.

For the path through the exposed HMI, the following features have been identified:

Internet accessible device *(T0883)*: Size:*pot*, Interaction:*low*, Virtualization: *service imitation*, Interfaces: *CLI*, Protocols: *IT/Remote Access*, Logging: *packet capturing.*

Graphical user interface *(T0823)* : Size:*pot*, Interaction:*low*, Virtualization: *service imitation*, Interfaces: *HMI*, Protocols: *IT*, Logging: *screen recording, packet capturing.*

Valid Accounts*(T0859)*: Size:*pot*, Interaction:*low*, Virtualization: *service imitation*, Interfaces: *web portal, CLI*, Protocols: *IT/Remote Access*, External Persuasion: *blog posts*, Logging: *screen recording, shell, packet capturing.*

Modify Parameter *(T0836)*: Size:*pot*, Interaction:*low*, Virtualization: *service imitation*, Interfaces: *HMI/ CLI*, Protocols: *IT/Industrial*, Logging: *packet capturing.*

Manipulation of Control *(T0813)*: Size:*pot*, Interaction:*low*, Virtualization: *service imitation*, Interfaces: *HMI/CLI*, Protocols: *IT/Industrial*, Physical Process: *model*, Logging: *packet capturing*.

The second path, through the use of the industrial protocol, is mapped to the following sets of features:

Remote Services *(T0886)*: Size:*pot*, Interaction:*low*, Virtualization: *service imitation*, Interfaces: *CLI*, Protocols: *Remote Access/Industrial*, Logging: *packet capturing*.

Unauthorized Command Message *(T0855)*:
Size:*pot*, Interaction:*low*, Virtualization: *service imitation*, Protocols: *Industrial*, Logging: *packet capturing*.

Modify Parameter *(T0836)*: Size:*pot*, Interaction:*low*, Virtualization: *service imitation*, Interfaces:*HMI/CLI*, Protocols: *IT/Industrial*, Logging: *packet capturing*.

Manipulation of Control*(T0813)*: Size:*pot*, Interaction:*low*, Virtualization: *service imitation*, Interfaces: *HMI/ CLI*, Protocols: *IT/Industrial*, Physical Process: *model*, Logging: *packet capturing*.

Based on the features that were obtained in the above steps, the most consistent composition involves a single low-interaction pot that provides two services to the attacker. Also, the sets clearly show that in terms of virtualization, a service imitation should suffice in enabling the observation targets we aim to capture. The first service that should be offered is an HMI, to enable execution through a graphical user interface. Since this HMI is accessible from the Internet through a web browser, it logically follows that the HMI should be facilitated by an IT protocol (such as HTTP). The second service should be an industrial protocol, of which the specifics depend on parameters acquired from the environment. Furthermore, a physical process model is required in order to reflect a change in the state caused by modifications that the attacker performs. Finally, packet capturing is necessary to register the interaction that the attacker has made. Although the screen recording appeared in some of the feature sets, this will not be necessary since a service-imitation does not have such ability compared to device imitation where the attacker is compromising the host machine rather than a standalone service. Also, script-based service imitation provides a good opportunity to integrate logging of all attacker inputs.

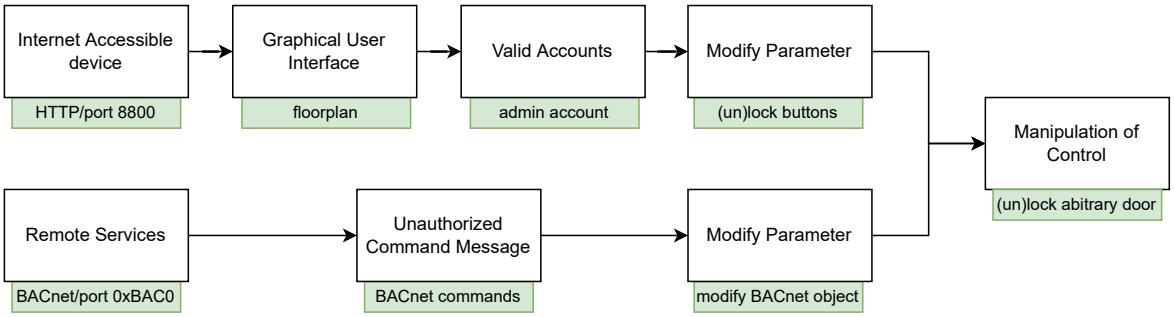

Figure 5.5: Combined attack paths with relevant information of the environment.

The environment that the use case considers is an internal production network, as the target building controller, an EasyIO FW-08 is used in the automation layer. This means that the controller is used to control field devices such as room controllers and corresponding sensors/actuators through the BACnet protocol. Furthermore, the device offers a proprietary software programming tool,

CPT tools, that allows for the creation of graphical webpages and widgets that can be hosted on the device's web server.

5.2.3 Implementation

For the implementation, existing frameworks have been investigated to determine whether any would be suitable as starting point. Based on the feature set composed in Section 5.2.2, we are looking for a framework that offers a low-interaction standalone pot, which is able to imitate services. At the same time, it should have enough flexibility to support the integration of the other features in our set.

Taking all considerations into account, the existing framework Conpot was selected to build upon [44]. The original version, however, does not provide all the features that are deemed necessary. Specifically, the features that enable 'Graphical User Interface' and 'Valid Accounts' are missing, while for the other targets, the environment parameters need to be integrated. In the first step, Conpot's HTTP service has been extended to serve as an HMI, accessible through the web browser. The HMI, shown in Figure 5.6, contains a floorplan from the scenario description to provide an overview of the building section. Furthermore, the state of each access door is represented by a red or green color, indicating a locked or unlocked state respectively.

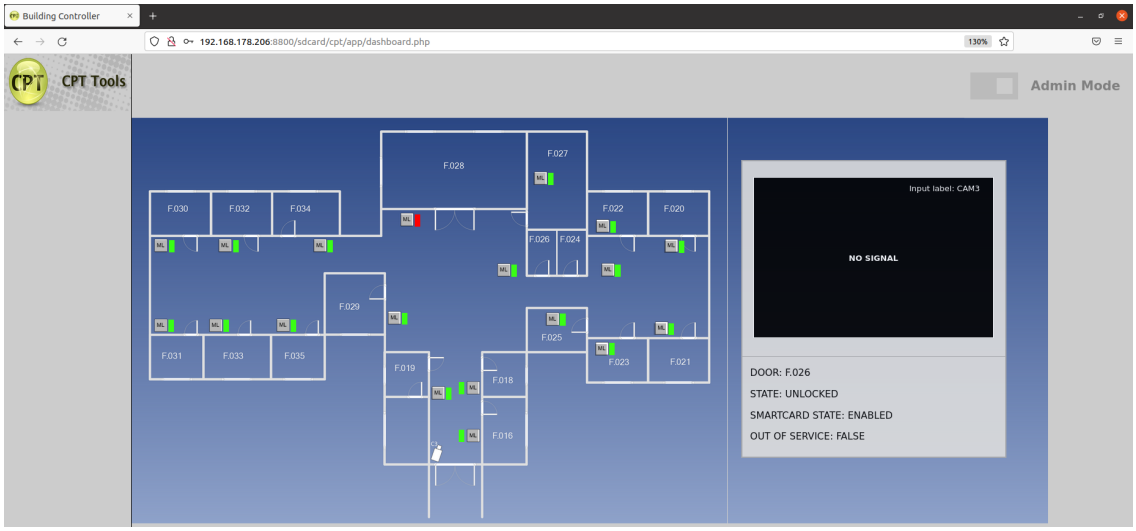

Figure 5.6: An overview of the HMI offered through HTTP on port 8800.

To add an extra layer of security by enabling the 'Valid Accounts' observation target, the HMI has two modes. Upon reaching the network address, the default mode is *Viewer*, where no modification through the interface is possible. When switching to *Admin* mode, the sign-in window shown in Figure 5.7 is displayed, which requires a valid set of credentials. In this implementation, these credentials are stored in the source code, but the organization is free to release them in other preferable ways such as through honeytokens on an internal server, or blog posts [34]. After completing the sign-in, the user can open any individual access door object. The faceplate, which can be seen in Figure 5.8, describes the actions that a user can generally perform. The actions that can be performed consist of locking and unlocking a door, and enabling or disabling the smartcard reader on the door. The reader allows personnel to locally unlock the door of a room that they intend to enter, assuming that the smartcard contains the prescribed authorization. As specified in the scenario description, the floorplan in the HMI contains one room, F.028, that cannot be unlocked regularly, in the sense that all buttons on the faceplate have been disabled. Furthermore, certain read-only properties regarding the access door, such as state of service, are shown as well. It is important to notice that Conpot's HTTP service does not provide any form

of persistence on the server's end. Therefore, the changes made by the attacker on the HMI are stored in Javascript on the client-side. This means that the attacker has access to the source code, though, there exist code obfuscation methods to decrease readability and thus the likelihood that the HMI is identified as deceptive [90].
With a finished deceptive system for this path, the remaining part is the logging functionality. We want to capture an attacker accessing the HMI on the decoy device, any credentials that might be entered, and any interactions and modifications carried out on the HMI. However, as the interaction caused by the attacker is only stored on the client-side, it does not automatically end up at server-side where we could extract that information. Also, upon a modification by the attacker, we cannot just reload the entire page by invoking an HTTP request with logging information to the server. The door states would then be reloaded to their initial state, which might rise a suspicion. As a solution, we utilized Asynchronous JavaScript And XML (AJAX) to send parameters through HTTP GET requests to the server, as soon as any of the aforementioned interaction has taken place. This way, data is send to webserver in background, without reloading the entire page.

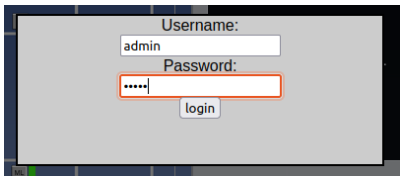

Figure 5.7: Sign-in window required for switching to Admin mode.

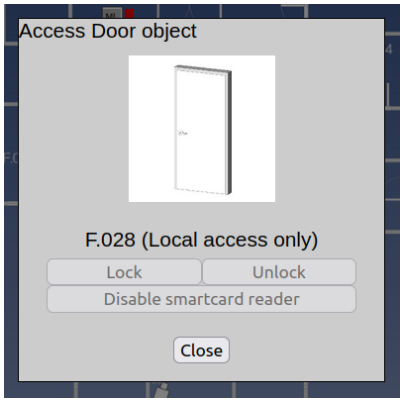

Figure 5.8: Faceplate with the inability to modify any parameter for room F.028.

To accomplish the second attack path, it has been built further upon Conpot's native implementation. The original template offers the possibility to create BACnet objects that represent physical items or descriptions of processes. An access door is one of the existing object types and provides a variety of properties specific to that object type [91]. These access doors have been added to the original version, with modifications to match the objects with those shown in the GUI and that is consistent with the use case. The significant difference with the first attack path here is that using the BACnet service, it is possible to manipulate controls for the door that could not be performed on the HMI. Figure 5.9 shows the address objects that are contained by the FW-08 device, seen from the attacker's perspective. This view has been obtained by using the BACnet client *Yabe* on another machine while connecting to the IP address and port number of the honeypot [92]. Once an object has been selected, which in this case is the door of room F.028, the client transmits a read request to obtain the object properties and their corresponding values. The implemented properties can be found in Figure 5.10. The *Present Value* property contains the writable command, where the values 0 and 1 represent lock and unlock requests, respectively. Upon modification, the client transmits the message to the receiver, which in this case is the BACnet server of our honeypot. By performing this action for the room with restricted access, the attacker has successfully circumvented the limitations in the HMI.
Again, the remaining functionality that requires implementation is the logging feature. According to our composed sets of required features, network-based logging should suffice in capturing all stated observation targets in this path. The native implementation already contained some logging functionality. For example, it can capture whether a connection, using the BACnet service, has been established including source IP address and port number. Moreover, it is able to detect the

transmission of Protocol Data Units (PDUs) from and to the honeypot. What is missing, however, are the details regarding the content of the PDU. Consequently, it is not clear whether reading or writing actions take place, and what values are being read or written. To solve this issue, we coded additional logging lines in the functions that initiate read and write actions. As a result, whenever any form of interaction takes place, the user receives information consisting of atleast the relevant *ObjectName* and its *PresentValue*.

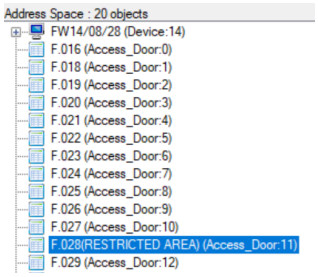

Figure 5.9: BACnet device containing the Access Door objects.

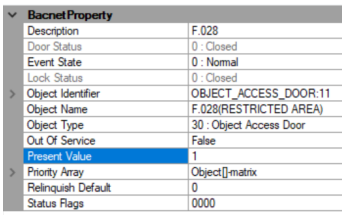

Figure 5.10: Properties of Access Door object, where present value has been modified to '1' (unlock action).

5.2.4 Verification

Now that an implementation has been created, we need to demonstrate that the honeypot is effective and succeeds in capturing the defined observation targets. For the first attack path, we can show that the target device is accessible from the attacker's machine with another IP address. In the snippet that can be found in Figure 5.11, the incoming HTTP session is logged at the moment that the attacker tries to connect. Specifically, the source IP address, source port number, timestamp, and metadata such as user-agent are attained. The second observation target, 'Graphical User Interface', can only be verified using the interaction that takes place on that interface. The interaction that is offered on the HMI consists of a sign-in action and the possibility to click specific buttons. As a result, this observation target is verified inevitably, by verification of the targets 'Valid Accounts' and 'Modify Parameter'. Figure 5.12 shows the capture of information that is relevant to these observation targets. The credentials entered by the attacker are stored at sign-in. Upon entering a set of valid credentials, the modifications that can be performed on most doors are logged in the form of locked/unlocked actions with the corresponding room number. Since the parameters in this honeypot are directly tied to state in the physical process, its modification can be interpreted as the intention to manipulate control. This means that as soon as the attacker clicks one of the lock buttons, it implies that in the real world, a disruption of the physical process would have taken place, which concludes the verification of the first attack path.

```
2022-05-19 15:36:27,113 New http session from [redacted] (9e486c3d-9
37f-4745-9cd5-974aecd75328)
2022-05-19 15:36:27,114 HTTP/1.1 GET request from ('[redacted]', 425
30): ('/sdcard/cpt/app/dashboard.php', [('Host', '[redacted]:8800'),
 ('User-Agent', 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:98.0) Gecko/
```

Figure 5.11: Logging an incoming HTTP session from the attacker.

To verify that honeypot is able to capture all observation targets for the second attack path, we take on the role of the attacker again. When using the *Yabe* client to connect to the honeypots BACnet service, the incoming connection is captured. Figure 5.13 shows that record, including source address and port number. This means that both the facilitation and logging of the observation

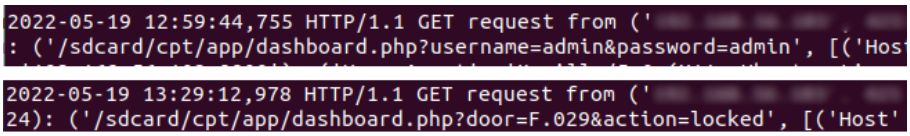

Figure 5.12: Logging the submitted credentials as well as parameter modification by the attacker.

target 'Remote Services' is functional. After establishing the connection, in contrary to the first attack path, we are able to send command messages for the room with restricted area, F.028, to the controller. At the same time, the command consists of a writeable value that represents the desired state of an object property. Therefore, by sending this command with an updated written value, both the targets 'Unauthorized Command Message' and 'Modify Parameter' are being applied. The upper snippet in Figure 5.14 shows the logging of these targets by capturing that a write action is performed against the selected object, using the selected value. The lower snippet shows a read action, after the attacker modified the parameter. We can see that the current *DoorValue* has changed to unlock, compared to the locked state before the attack which can be seen in Figure 5.13. This final observation demonstrates that an adversary could inflict 'Manipulation of Control' and that the honeypot is able to capture it, hence the final observation target is verified.

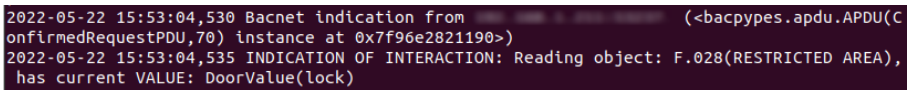

Figure 5.13: Logging an incoming BACnet connection from the attacker.

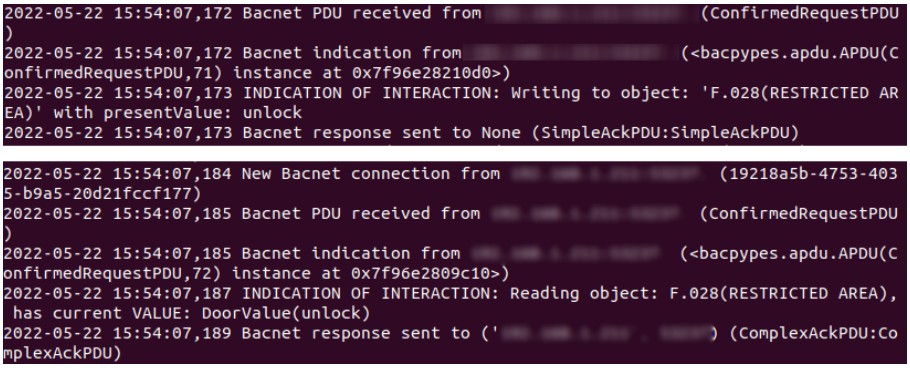

Figure 5.14: Logging the transmission of an unauthorized command message and an updated value after parameter modification.

To conclude, a honeypot has been constructed that meets the objective of the organization. Based on observation targets, derived from the stated objective, a set of features was composed that formed the basis of the honeypot. For the implementation, we used an existing platform and extended it with functionalities to create a system that included the necessary features for our use case. This resulted in a honeypot where it is possible, through two different attack paths, to manipulate parameters that would, in the real world, affect the physical process. While the attacker is interacting with the deceptive system, the owner of the honeypot receives specific information about the attackers' activities. Hence, the organization is able to retrieve insight into both the intentions and capabilities of an adversary.

Chapter 6

Discussion

In this chapter, we elaborate on our findings during the development of the methodology. Furthermore, we interpret the obtained results and state the limitations of the framework.

The first step in developing the methodology was to form a topology of honeypot features based on existing research. When considering the identified features, significant differences in the support for certain features can be seen. Features such as *Logging*, *Size*, and *Interaction Level* are prevalent in the analyzed literature compared to *External Persuasion*. However, as can be seen in the mapping, even these features with lower support are required to capture certain observation targets.
Furthermore, the definition of *Interaction Level* is not fixed, resulting in honeypots that could proclaim their own interaction level. Some literature states that high-interaction entitles the use of a real device by definition [11]. We think that the *Interaction Level* and *Virtualization* features should remain decoupled, as in our research, especially with a new generation of virtualization techniques allowing for accurate imitation and thus a high quality of implemented services [93].

One important topic that has not been included in the formation of observation targets, but was considered thoroughly is the attacker profile and its role in forming attacker models for a honeypot. We identified the following three relevant profiles for the industrial and governmental nature of the systems in our research, based on existing literature [94].

- **Nation-State**: highly capable and well-resourced attacker sponsored by a state/nation. Potential targets consist of public infrastructure systems, mass transit, power or water systems, and general intelligence [95]. According to the Microsoft Threat Intelligence Center (MSTIC), nation-state actors are mainly focused on government agencies, intergovernmental organizations (IGOs), nongovernmental organizations (NGOs), and think tanks for traditional espionage or surveillance objectives [96].

- **Insider**: attacker that has physical access to the system, either a disgruntled employee or a person motivated by a third party. There are two forms of attacks, directly against a system, or providing critical information to a third party for intelligence gathering or enabling an attack [97].

  Now, due to the convergence of CPS and IoT domains, traditional CPS such as Industrial Control Systems and Building Automation Systems become Internet-connected. As the physical barrier for accessing a target network is removed in these systems, another attacker type becomes relevant:

- **Cybercriminal**: an attacker with extensive security knowledge and skills. Driven mainly by economic motivation/gaining profit. With growing connectivity due to, for example, the rise in remote access, cybercriminals increasingly gain access to CPS networks [98].

We aimed to attribute observation targets to certain attacker profiles. For instance, observation targets that require physical access to a system, such as 'Hardware Additions *(T1200)*', or 'Replication through Removable Media *(T0847)*', could be attributed to the profile of an insider. Consequently, the user of our developed methodology could only retrieve the observation targets that were relevant for his use case, based on the expected attacker profiles. For example, a company could base its honeypot on observation targets associated with cybercriminals, or, a government agency could choose from a set of observation targets that are mostly associated with nation-states or insiders. However, while some observation targets could be attributed fairly well, most of the targets are in practice not specific to an attacker profile. That is, among others, because the TTP of nation-states and (advanced) cybercriminals are becoming increasingly alike [99]. Therefore, it becomes significantly harder to make a distinction in observation targets based on the attacker's capabilities. Hence it has been omitted in the development of our methodology. However, for future research, it would be very useful to be able to align the honeypot with the type of attackers it aims to attract.

In the mapping, we can observe a number of relations that can be explained well. For example, in the installation/modification phases, the vast majority of techniques require high interaction. This observation corresponds with expectations, as the techniques in this phase often rely on file and/or operation systems to be used. Also, when considering the validation, the mapping is fairly consistent compared to the validation set. However, the main problem with proper validation lies in the fact that most of the MITRE techniques have, in the available literature, not yet been caught by honeypots. A reason could be that the MITRE for ICS framework is relatively new, published in 2020. The techniques from the framework have been identified during actual attacks on physical production devices. As a result, there does not yet exist a known reference or source that can be used to validate the mapping, especially for observation targets that we claim could be captured using virtualized honeypots. Therefore, this framework could be seen as a beginning, such that future honeypot research could document the recognized techniques that they have captured.

Despite the impossibility to validate the entire mapping, we were able to show consistency in a part of it. The validation set of honeypots demonstrated that for an arbitrary set of observation targets, the predicted features mostly correspond with the actual features that the works had implemented. For the deviating values, we noticed that some options more expensive than necessary might have been taken by the researchers of those works. This, however, is also a limitation of the framework. We assumed relative costs based on the functional differences within a meta-level feature. For most features, such as *Size* and *Interaction Level* it works well. However, specifically for the *Virtualization* feature, the assumption that a real device costs more than a device or service imitation becomes problematic. In practice, there are various ways in which the procurement of a physical device can be less costly than developing a digital twin. Therefore, in future research, a cost function or similar method should be constructed to decide whether to select a device imitation or a real device if both options would suffice in capturing an observation target.

Another limitation of the framework can be seen through the difficulty of integrating all dependencies between certain features. For example, when merging the output of the mapping to a single set of features in the use case in Chapter 5.2.2, the combination of host-based logging with service imitation as *Virtualization* feature occurred. In practice, this combination is unusual as the options for host-based logging are limited for this level of virtualization, due to the absence of a functional operating system. Therefore, we recommended that the output of the mapping is always checked before the features are adopted for implementation.

Finally, the use case shows that it is completely possible to build upon an existing honeypot framework, as long as that framework supports the possibility to integrate any missing features that the user aims to implement. Conpot initially did not offer an HMI or the type of network-based logging that we required for the user's observation targets. However, since it did support

an HTTP server and a basic, incomplete, logging functionality, it was possible to facilitate and verify all observation targets that belonged to the agency's objective.

Chapter 7

Conclusion and Future Work

In this chapter, we provide a conclusion by summarizing the answers to our research questions. Moreover, we come up with suggestions for future work based on the findings of our research.

7.1 Conclusion

The motivation for this research originated from the identified heterogeneity of honeypots in the IoT/IIoT/CPS domain. While there are lots of well-known honeypots available in the literature, it is difficult for users to fabricate a honeypot according to their specific objectives. The goal of the research was to develop a systematic framework for the construction of arbitrary IoT/IIoT/CPS honeypots for the particular objective that a user might have. To accomplish this objective, we first required an overview of all known characteristics of honeypots in our domain. Then, we needed to identify the known attacker TTPs used in our domain. As a third step, the honeypot characteristics and observation targets should be linked together. To fulfill the final objective, we needed to demonstrate that the methodology we wanted to develop could also be used in practice.

The first sub-question has been answered by the creation of a topology of honeypot features. This topology provides a clear overview of identified features, based on a set of existing IoT/IIoT/CPS honeypots. The features were structured hierarchically and grouped depending on their characteristics.

To identify the attacker TTPs that were relevant for our research, we used two existing reputable frameworks which were specific to our research domain. From these frameworks, we made a strict selection of observation targets relevant to honeypots.

For the third research sub-question, we first investigated what functionalities would be needed to facilitate and register the observation target. Then, we constructed a mapping, using the honeypot topology made in the first sub-question, in which we clarify what set of features is required to capture a specific observation target.

To demonstrate that the framework is not only theoretical but can also be applied in real-world scenarios, we presented a use case. In the use case, the methodology is followed, starting with a user's objective and ending with a functional and verified honeypot.
The framework has certain limitations, such as the assumptions that were made to determine the relative costs of features. Also, the output of the mapping should be interpreted before implementation, to make sure that complicated dependencies have not created a set in which some features are unlikely to coexist. As a final result, we have come up with a framework for a user with a specific objective that intends to construct a honeypot in the IoT/IIoT/CPS domain.

7.2 Future Work

This section describes three recommendations for possible improvements and directions for further research.

In our opinion, virtualization has high potential in the honeypot research domain. Realistic virtualized honeypots could decrease physical location dependencies and increase scalability and adaptability. For future research, we would recommend a feasibility study regarding virtualization techniques specifically for CPS devices. Consequently, improved estimations of costs could be made such that users can have a more substantiated choice between the use of device imitations or physical devices.

Our second recommendation concerns the need for a consensus on the definition of Interaction Level. As we have seen researchers interpret this feature differently, it results in confusion and mismatches when comparing honeypots from different works. By proposing a well-defined topology consisting of specific requirements to proclaim a system as high-interaction, future honeypots could be classified and compared objectively.

Finally, we need to investigate how to include the environment of the user in the framework. Until now, the environment parameters are only considered after the set of features has been composed. Consequently, the user still needs to investigate how to implement a specific protocol in the constructed honeypot. By integrating the specifics of the target environment in the framework, one could aim for automated development of the honeypot, when the user specifies the objective and the target environment.

References

[1] D. Ratasich et al. "A Roadmap Toward the Resilient Internet of Things for Cyber-Physical Systems". In: *IEEE Access* 7 (2019), pp. 13260–13283. DOI: 10.1109/ACCESS.2019.2891969.

[2] *Analysis of the Cyber Attack on the Ukrainian Power Grid*. Tech. rep. Electricity Information Sharing and Analysis Center (E-ISAC). 2016. http://www.mandiant.com/resources/triton-actor-ttp-profile-custom-attack-tools-detections.

[3] N. Perlroth and C. Krauss. *A Cyberattack in Saudi Arabia Had a Deadly Goal. Experts Fear Another Try*. Tech. rep. New York Times. Mar. 2018. https://www.nytimes.com/2018/03/15/technology/saudi-arabia-hacks-cyberattacks.html.

[4] L. Spitzner. *Definitions and Value of Honeypots*. SecurityFocus. May 2003. https://www.tracking-hackers.com/papers/honeypots.html.

[5] R. W. Shirey. *Internet Security Glossary, Version 2*. RFC Editor. Aug. 2007. DOI: 10.17487/RFC4949.

[6] L. Spitzner. *Honeypots: Catching the Insider Threat*. Honeypot Technologies Inc. Oct. 2003. https://www.acsac.org/2003/papers/spitzner.pdf.

[7] A. Verma. *Production Honeypots: An Organization's view*. SANS Institute. Oct. 2003. https://www.giac.org/paper/gsec/3585/production-honeypots-organizations-view/105831.

[8] D. Bianco. *The Pyramid of Pain*. Tech. rep. SANS Institute, Mar. 2013. http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html.

[9] M. Dornseif, T. Holz and S. Müller. "Honeypots and limitations of deception". In: *Heute schon das Morgen sehen, 19. DFN-Arbeitstagung über Kommunikationsnetze in Düsseldorf*. Ed. by J. V. Knop, W. Haverkamp and E. Jessen. Bonn: Gesellschaft für Informatik e.V., 2005, pp. 235–252. https://dl.gi.de/20.500.12116/28613.

[10] L. Spitzner. "Honeypots: Tracking Hackers". In: Addison Wesley, 2003. Chap. 4. ISBN: 0-321-10895-7.

[11] S. Maesschalck et al. "Don't get stung, cover your ICS in honey: How do honeypots fit within industrial control system security". In: *Computers & Security* 114 (2022), p. 102598. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2021.102598.

[12] M. Constantin, E. Mirica and R. Deaconescu. "Detecting and Analyzing Zero-Day Attacks Using Honeypots". In: *2013 19th International Conference on Control Systems and Computer Science*. 2013, pp. 543–548. DOI: 10.1109/CSCS.2013.94.

[13] A. Tiwari and D. Kumar. "Comparitive Study of Various Honeypot Tools on the Basis of Their Classification & Features". In: *SSRN Electronic Journal* (Jan. 2020). DOI: 10.2139/ssrn.3565078.

[14] P. Simoes et al. "On the use of Honeypots for Detecting Cyber Attacks on Industrial Control Networks". In: July 2013.

[15] *Overview of the Internet of Things*. Tech. rep. International Telecommunication Union, 2012. https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=y.2060.

[16] *Cyber-Physical Systems: Executive Summary*. CPS Steering Group. 2008. http://iccps.acm.org/2011/_doc/CPS-Executive-Summary.pdf.

[17] *Industrial Internet of Things: Unleashing the Potential of Connected Products and Services*. Tech. rep. World Economic Forum, Jan. 2015. https://www3.weforum.org/docs/WEFUSA_IndustrialInternet_Report2015.pdf.

[18] R. Basir et al. "Fog Computing Enabling Industrial Internet of Things: State-of-the-Art and Research Challenges". In: *Sensors* 19.11 (Nov. 2019), p. 4807. DOI: 10.3390/s19214807.

[19] M. A. Razzaque et al. "Middleware for Internet of Things: A survey". In: *IEEE Internet of Things Journal* 3 (Jan. 2015). DOI: 10.1109/JIOT.2015.2498900.

[20] *Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK)*. MITRE. Oct. 2018. https://attack.mitre.org.

[21] D. Dos Santos, C. Speybrouck and E. Costante. *Cybersecurity in Building Automation Systems (BAS)*. 2018. https://www.forescout.com/securing-building-automation-systems-bas/.

[22] *Exploit Public-Facing Application (T0819) - ATT&CK for Industrial Control Systems*. MITRE. 2020. https://attack.mitre.org/techniques/T0819/.

[23] *Default Credentials (T0812) - ATT&CK for Industrial Control Systems*. MITRE. 2020. https://attack.mitre.org/techniques/T0812/.

[24] N. Provos. "A Virtual Honeypot Framework". In: *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*. SSYM'04. San Diego, CA: USENIX Association, 2004, p. 1. https://dl.acm.org/doi/10.5555/1251375.1251376.

[25] H. Naruoka et al. "ICS Honeypot System (CamouflageNet) Based on Attacker's Human Factors". In: *Procedia Manufacturing* 3 (2015). 6th International Conference on Applied Human Factors and Ergonomics (AHFE 2015) and the Affiliated Conferences, AHFE 2015, pp. 1074–1081. ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2015.07.175.

[26] V. Pothamsetty and M. Franz. *SCADA HoneyNet Project: Building Honeypots for Industrial Networks*. Jan. 2004. http://scadahoneynet.sourcefourge.net/plc.html.

[27] M. Winn et al. " Constructing Cost-Effective and Targetable ICS Honeypots Suited for Production Networks ". In: *International J. of Critical Infastructure Protection* (Sept. 2015). DOI: 10.1016/j.ijcip.2015.04.002.

[28] J. K. Gallenstein. "Integration of the Network and Application Layers of Automatically-Configured Programmable Logic Controller Honeypots". MA thesis. Air Force Institute of Technology Air University, Mar. 2017. https://scholar.afit.edu/etd/1572/.

[29] S. Abe et al. "Developing Deception Network System with Traceback Honeypot in ICS Network". In: *SICE Journal of Control, Measurement, and System Integration* 11.4 (2018), pp. 372–379. DOI: 10.9746/jcmsi.11.372.

[30] M. Wang, J. Santillan and F. Kuipers. "ThingPot: an interactive Internet-of-Things honeypot". In: *CoRR* abs/1807.04114 (2018). arXiv: 1807.04114.

[31] S. Kamoen. "Honeytrack: Persistent honeypot for the Internet of Things". MA thesis. TU Delft, July 2018. http://resolver.tudelft.nl/uuid:344bd7aa-0a17-47dc-92fd-bd6f7e7b08c8.

[32] R. Verhoef. *About Honeytrap*. DTACT (formerly DutchSec. Mar. 2018. https://medium.com/@remco_verhoef/about-honeytrap-db2d696c7025.

[33] M. Dodson, A. R. Beresford and M. Vingaard. "Using Global Honeypot Networks to Detect Targeted ICS Attacks". In: *2020 12th International Conference on Cyber Conflict (CyCon)*. Vol. 1300. 2020, pp. 275–291. DOI: 10.23919/CyCon49761.2020.9131734.

[34] S. Hilt et al. *Caught in the Act: Running a Realistic Factory Honeypot to Capture Real Threats*. Tech. rep. Trend Micro, 2020. https://documents.trendmicro.com/assets/white_papers/wp-caught-in-the-act-running-a-realistic-factory-honeypot-to-capture-real-threats.pdf.

[35] N. Cifranic. et al. "Decepti-SCADA: A Framework for Actively Defending Networked Critical Infrastructures". In: *Proceedings of the 5th International Conference on Internet of Things, Big Data and Security - IoTBDS,* INSTICC. SciTePress, 2020, pp. 69–77. ISBN: 978-989-758-426-8. DOI: 10.5220/0009343300690077.

[36] J. Franco et al. "A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems". In: *CoRR* abs/2108.02287 (2021). arXiv: 2108.02287. URL: https://arxiv.org/abs/2108.02287.

[37] S. Litchfield et al. "Rethinking the Honeypot for Cyber-Physical Systems". In: *IEEE Internet Computing* 20.5 (2016), pp. 9–17. DOI: 10.1109/MIC.2016.103.

[38] H. A. Lin. "A Framework for Industrial Control System Honeypot Network Traffic Generation". In: vol. 1585. Theses and Dissertations, 2017. URL: https://scholar.afit.edu/etd/1585.

[39] J. Bauer et al. "Honeypots for Threat Intelligence in Building Automation Systems". In: *2019 Computing, Communications and IoT Applications (ComComAp)*. 2019, pp. 242–246. DOI: 10.1109/ComComAp46287.2019.9018776.

[40] G. Bode et al. "Cloud, Wireless Technology, Internet of Things: the Next Generation of Building Automation Systems?" In: *Journal of Physics: Conference Series* 1343 (Nov. 2019), p. 012059. DOI: 10.1088/1742-6596/1343/1/012059.

[41] D. dos Santos, S. Dashevskyi and A. Amri. *NUCLEUS:13, Dissecting the Nucleus TCP/IP stack*. Tech. rep. Forescout Technologies, Nov. 2021. https://www.forescout.com/resources/nucleus13-research-report-dissecting-the-nucleus-tcpip-stack/.

[42] *ATT&CK for Industrial Control Systems*. MITRE. 2020. https://collaborate.mitre.org/attackics/.

[43] M. J. Assante and R. M. Lee. "The Industrial Control System Cyber Kill Chain". In: (2015). Sans Institue. https://sansorg.egnyte.com/dl/HHa9fCekmc.

[44] L. Rist et al. "Conpot ICS/SCADA Honeypot". In: *IEEE Transactions on Industrial Informatics* (2020). http://conpot.org.

[45] G. Bernieri, M. Conti and F. Pascucci. "MimePot: a Model-based Honeypot for Industrial Control Networks". In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. 2019, pp. 433–438. DOI: 10.1109/SMC.2019.8913891.

[46] B. Daniel et al. "CryPLH: Protecting Smart Energy Systmes from Targeted Attacks with a PLC Honeypot". In: 2014, pp. 9–17. DOI: 10.1007/978-3-319-10329-7_12.

[47] J. Cao et al. "DiPot: A Distributed Industrial Honeypot System". In: *Smart Computing and Communication*. 2018, pp. 300–309. ISBN: 978-3-319-73829-1. DOI: 10.1007/978-3-319-73830-7_30.

[48] D. Pliatsios et al. "A Novel and Interactive Industrial Control System Honeypot for Critical Smart Grid Infrastructure". In: 2019. DOI: 10.1109/CAMAD.2019.8858431.

[49] O. Redwood, J. Lawrence and M. Burmester. "Symbolic Honeynet Framework for SCADA System Threat Intelligence". In: 2015, pp. 103–118. ISBN: 978-3-319-26566-7. DOI: 10.1007/978-3-319-26567-4_7.

[50] R. Piggin and I. Buffey. "Active Defence Using an Operational Technology Honeypot". In: 2016, 15 (6 .)–15 (6 .) DOI: 10.1049/cp.2016.0860.

[51] "Chapter 12 - Using Canary Honeypots for Detection". In: *Applied Network Security Monitoring*. Ed. by C. Sanders and J. Smith. Boston: Syngress, 2014, pp. 317–338. ISBN: 978-0-12-417208-1. DOI: https://doi.org/10.1016/B978-0-12-417208-1.00012-X.

[52] U. Tamminen. *Github - Kippo*. Tech. rep. 2009. https://github.com/desaster/kippo.

[53] "Digital Bond SCADA Honeypot". In: (2006). URL: https//web.archive.org/web/20111215085656/http://www.digitalbond.com/tools/scada-honeynet.

[54] P. Simões et al. "Specialized Honeypots for SCADA Systems". In: *Intelligent Systems, Control and Automation: Science and Engineering*. Vol. 78. 2014. ISBN: 978-3-319-18301-5. DOI: 10.1007/978-3-319-18302-2.

[55] A. Vetterl and R. Clayton. "Honware: A Virtual Honeypot Framework for Capturing CPE and IoT Zero Days". In: *2019 APWG Symposium on Electronic Crime Research (eCrime)*. 2019, pp. 1–13. DOI: 10.1109/eCrime47957.2019.9037501.

[56] M. Oosterhof. *Github - Cowrie*. Tech. rep. 2014. https://github.com/cowrie/cowrie.

[57] D. Holmes et al. "Digital Twins and Cyber Security – solution or challenge?" In: *6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*. 2021, pp. 1–8. DOI: 10.1109/SEEDA-CECNSM53056.2021.9566277.

[58] J. D. Guarnizo et al. "SIPHON: Towards Scalable High-Interaction Physical Honeypots". In: Association for Computing Machinery, 2017, pp. 57–68. ISBN: 9781450349567. DOI: 10.1145/3055186.3055192.

[59] A. V. Serbanescu, S. Obermeier and D.-Y. Yu. "ICS Threat Analysis Using a Large-Scale Honeynet". In: *3rd International Symposium for ICS & SCADA Cyber Security Research* (2015). DOI: 10.14236/ewic/ICS2015.3.

[60] J. van der Lelie and R. Breuk. "A visual analytic apporoach for analyzing SSH honeypots". In: University of Amsterdam. 2012. https://rp.os3.nl/2011-2012/p26/report.pdf.

[61] J. P. John et al. "Heat-Seeking Honeypots: Design and Experience". In: *Proceedings of the 20th International Conference on World Wide Web*. WWW '11. Hyderabad, India: Association for Computing Machinery, 2011, 207–216. ISBN: 9781450306324. DOI: 10.1145/1963405.1963437. URL: https://doi.org/10.1145/1963405.1963437.

[62] B. Mphago et al. "Self-Advertising Attack Surfaces for Web Application Honeypots: New Technology to Managing Cyber Disasters". English. In: *Information & Security: An International Journal* 40 (2018).

[63] B. Green, M. Krotofil and A. Abbasi. "On the Significance of Process Comprehension for Conducting Targeted ICS Attacks". In: *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and Privacy*. Association for Computing Machinery, 2017, pp. 57–67. ISBN: 9781450353946. DOI: 10.1145/3140241.3140254.

[64] T. Vollmer and M. Manic. "Cyber-Physical System Security With Deceptive Virtual Hosts for Industrial Control Networks". In: *IEEE Transactions on Industrial Informatics* 10.2 (2014), pp. 1337–1347. DOI: 10.1109/TII.2014.2304633.

[65] S. Miller et al. *TRITON Actor TTP Profile, Custom Attack Tools, Detection and ATT&CK Mapping*. Tech. rep. 2019. http://www.mandiant.com/resources/triton-actor-ttp-profile-custom-attack-tools-detections.

[66] A. Shehod. *Ukraine Power Grid Cyberattack and US Susceptibility: Cybersecurity Implications of Smart Grid Advancements in the US*. Tech. rep. Massachusetts Institute of Technology. 2016. http://www.mandiant.com/resources/triton-actor-ttp-profile-custom-attack-tools-detections.

[67] M. Tasanko. *Honeypots: Security Boost for Critical Infrastructure or Criminal Facilitation?* Tech. rep. Master's thesis. Tilburg University, 2019. arno.uvt.nl/show.cgi?fid=148201.

[68] R. Siles. "HoneySpot: Wireless Honeypot. Monitoring the Attacker's Activities in Wireless Networks". In: http://citeseerx.ist.psu.edu/viewdoc/summary?=doi=10.1.1.158.8428. The Spanish Honeynet Project (SHP), 2007.

[69] R. Goel, A. Sardana and R. C. Joshi. "Wireless Honeypot: Framework, Architectures and Tools". In: *International Journal of Network Security* 15 (2013). http://ijns.jalaxy.com.tw/contents/ijns-v15-n5/ijns-2013-v15-n5-p373-383.pdf, pp. 373–383.

[70] A. Khraisat et al. "Survey of intrusion detection systems: techniques, datasets and challenges". In: *Cybersecurity* 2 (Dec. 2019). DOI: 10.1186/s42400-019-0038-7.

[71] M. Roesch. "Snort - Lightweight Intrusion Detection for Networks". In: LISA '99. Seattle, Washington: USENIX Association, 1999, 229–238.

[72] J. Giraldo et al. "A Survey of Physics-Based Attack Detection in Cyber-Physical Systems". In: *ACM Comput. Surv.* 51.4 (2018). ISSN: 0360-0300. DOI: 10.1145/3203245. URL: https://doi.org/10.1145/3203245.

[73] M. R. G. Raman and A. P. Mathur. "A Hybrid Physics-Based Data-Driven Framework for Anomaly Detection in Industrial Control Systems". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2021), pp. 1–12. DOI: 10.1109/TSMC.2021.3131662.

[74] E. Hutchins, M. Cloppert and R. Amin. "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains". In: *Leading Issues in Information Warfare & Security Research* 1 (2011). Lockheed Martin Corporation.

[75] N. Falliere, L. O. Murchu and E. Chien. *W32.Stuxnet Dossier*. Tech. rep. Symantec, 2011. https://www.wired.com/images_blogs/threatlevel/2010/11/w32_stuxnet_dossier.pdf.

[76] B. Johnson et al. *Attackers Deploy New ICS Attack Framework "TRITON" and Cause Operational Disruption to Critical Infrastructure*. Tech. rep. 2017. https://www.mandiant.com/resources/attackers-deploy-new-ics-attack-framework-triton.

[77] B. Prabadevi et al. "Lattice Structural Analysis on Sniffing to Denial of Service Attacks". In: *ArXiv* abs/1907.12735 (2019).

[78] J. Bill. "Hacked Cyber Security Railways". In: (2017). https://www.londonreconnections.com/2017/hacked-cyber-security-railways/.

[79] R. C. Bodenheim. *Impact of the Shodan Computer Search Engine on Internet-facing Industrial Control System Devices*. Tech. rep. Master's Thesis. Air Force Institute of Technology Air University, 2014. https://apps.dtic.mil/docs/citations/ADA601219.

[80] D. Mashima et al. "Towards a grid-wide, high-fidelity electrical substation honeynet". In: *2017 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. 2017, pp. 89–95. DOI: 10.1109/SmartGridComm.2017.8340689.

[81] Navarro, S. Balbastre and S. Beyer. "Gathering Intelligence Through Realistic Industrial Control System Honeypots: 13th International Conference, CRITIS 2018, Kaunas, Lithuania, September 24-26, 2018, Revised Selected Papers". In: Jan. 2019, pp. 143–153. ISBN: 978-3-030-05848-7. DOI: 10.1007/978-3-030-05849-4_11.

[82] D. Antonioli, A. Agrawal and N. O. Tippenhauer. "Towards High-Interaction Virtual ICS Honeypots-in-a-Box". In: *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*. CPS-SPC '16. Vienna, Austria: Association for Computing Machinery, 2016, 13–22. DOI: 10.1145/2994487.2994493.

[83] E. Vasilomanolakis et al. "Multi-stage attack detection and signature generation with ICS honeypots". In: *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. 2016, pp. 1227–1232. DOI: 10.1109/NOMS.2016.7502992.

[84] A. Belqruch and A. Maach. "SCADA Security Using SSH Honeypot". In: *Proceedings of the 2nd International Conference on Networking, Information Systems and Security*. NISS19. Rabat, Morocco: Association for Computing Machinery, 2019. ISBN: 9781450366458. DOI: 10.1145/3320326.3320328. URL: https://doi.org/10.1145/3320326.3320328.

[85] F. Xiao, E. Chen and Q. Xu. "S7commTrace: A High Interactive Honeypot for Industrial Control System Based on S7 Protocol". In: *Information and Communications Security*. Ed. by S. Qing et al. Cham: Springer International Publishing, 2018, pp. 412–423. ISBN: 978-3-319-89500-0.

[86] T. Cruz et al. "Leveraging Virtualization Technologies to Improve SCADA ICS Security". In: *Journal of Information Warfare 1445-3312 (Printed) ISSN 1445-3347* 15 (Oct. 2016).

[87] S. Hilt et al. *Exposed and Vulnerable Critical Infrastructure: Water and Energy Industries*. Tech. rep. Trend Micro, 2018. https://documents.trendmicro.com/assets/white_papers/wp-exposed-and-vulnerable-critical-infrastructure-the-water-energy-industries.pdf.

[88] M. Peacock, M. N. Johnstone and C. Valli. "Security Issues with BACnet Value Handling". In: *International Conference on Information Systems Security and Privacy*. 2017.

[89] Q. Wanying et al. "The Study of Security Issues for the Industrial Control Systems Communication Protocols". In: *Joint International Mechanical, Electronic and Information Technology (JIMET)*. 2015.

[90] W. Xu, F. Zhang and S. Zhu. "The Power of Obfuscation Techniques in Malicious JavaScript Code: A Measurement Study". In: *7th International Conference on Malicious and Unwanted Software*. 2021, pp. 9–16. DOI: 10.1109/MALWARE.2012.6461002.

[91] *Addendum to Standard 135-2004, BACnet - A Data Communication Protocol for Building Automation and Control Netowrks*. Tech. rep. ASHRAE, 2006. http://www.bacnet.org/Addenda/Add-135-2004f-PPR2-1.pdf.

[92] fchaxel and illishar. *Yet Another Bacnet Explorer*. http://sourceforge.net/projects/yetanotherbacnetexplorer/.

[93] H. Forbes. *Virtualization and Industrial Control*. Tech. rep. ARC Advisory Group. World Economic Forum, Jan. 2018. https://www.arcweb.com/blog/virtualization-industrial-control.

[94] M. Rocchetto and N. O. Tippenhauer. "On Attacker Models and Profiles for Cyber-Physical Systems". In: vol. 9879. Sept. 2016, pp. 427–449. ISBN: 978-3-319-45740-6. DOI: 10.1007/978-3-319-45741-3_22.

[95] E. Doynikova, E. Novika and I. Kotenko. "Attacker Behaviour Forecasting Using Methods of Intelligent Data Analysis: A Comparative Review and Prospects". In: *Information* 11.3 (2020). DOI: 10.3390/info11030168.

[96] J. Lambert. "Nation State Threats". In: *'Microsoft Digital Defense Report*. https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWMFIi?id=101738. 2021. Chap. 3, pp. 47–70.

[97] J. B. Sheldon. "State of the Art: Attackers and Targets in Cyberspace". In: *Journal of Military and Strategic Studies* 14.2 (2012). https://jmss.org/article/view/58029/43672.

[98] I. Murphy. *Dragos warns cybercriminals increasing access to ICS/OT networks*. Tech. rep. Enterprise Times. Feb. 2022. https://www.enterprisetimes.co.uk/2022/02/25/dragos-warns-cybercriminals-increasing-access-to-ics-ot-networks/.

[99] E. Goncharov. *Threats to ICS and Industrial Enterprises in 2022*. Tech. rep. Kaspersky ICS Cert, Nov. 2021. https://ics-cert.kaspersky.com/media/Kaspersky-ICS-CERT-threats-to-ics-and-industrial-enterprises-in-2022-En.pdf.

[100] E. Sisinni et al. "Industrial Internet of Things: Challenges, Opportunities, and Directions". In: *IEEE Transactions on Industrial Informatics* 14.11 (2018), pp. 4724–4734. DOI: 10.1109/TII.2018.2852491.

[101] Janita. *DDoS attack halts heating in Finland amidst winter*. http://metropolitan.fi/entry/ddos-attack-halts-heating-in-finland-amidst-winter. Nov. 2016.

[102] S. Abe et al. “Tracking attack sources based on traceback honeypot for ICS network”. In: *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. 2017, pp. 717–723. DOI: 10.23919/SICE.2017.8105603.

[103] J. Brown. “Identifying Honeypots simulating Internet-connected Industrial Control System Devices”. In: *NPA thesis* ().

Appendix A

Deconstruction of Analyzed Set

Size	
pot	Scada Honeynet project
	Digital Bond
	Conpot
	CryPLH
	Simoes et al.
	MimePot
	HoneyPHY
	Honeyd
	Dodson et al.
	Bauer et al.
	Cifranic et al.
	Serbanescu et al.
	DiPot
	Lelie et al.
	Honware
	SIPHON
	Thingpot
	Honeytrack
net	Pliatsios et al.
	Hilt et al.
	GridPot
	Piggin and Buffey
Interaction Level	
low	Scada Honeynet project
	Digital Bond.
	Serbanescu et al.
	Conpot
	Honeyd
	HoneyPHY
	Simoes et al.
	DiPot
	Lelie et al.
	CryPLH
	Honeytrack
	Gridpot
high	Piggin and Buffey
	Pliatsios et al.

	MimePot
	Bauer et al.
	Cifranic et al.
	Honware
	SIPHON
	ThingPot
	Hilt et al.
	Dodson et al.
Virtualization	
service imitation	Scada Honeynet project
	Honeyd
	HoneyPHY
	GridPot
	Lin et al.
	Simoes et al.
	MimePot
	Serbanescu et al.
	Lelie et al.
	Honeytrack
	Conpot
	Bauer et al.
device imitation	Digital Bond
	CryPLH
	ThingPot
	Cifranic et al.
	DiPot
	Honware
real device	Piggin and Buffey
	SIPHON
	Pliatsios et al.
	Hilt et al.
	Dodson et al.
Services	
interfaces	Pliatsios et al.
	Hilt et al.
	Gridpot
	Piggin and Buffey
	Bauer et al.
	SIPHON
protocols	*can be found in Table 4.1*
External Persuasion	
active	Hilt et al.
passive	Hilt et al.
Physical Process	
process model	HoneyPHY
	GridPot
	Piggin and Buffey
	MimePot
real plant	Hilt et al.
Scale	
single	Scada Honeynet project
	Digital Bond
	Conpot

	CryPLH
	Simoes et al.
	MimePot
	HoneyPHY
	Pliatsios et al.
	Hilt et al.
	GridPot
	Piggin and Buffey
multiple	Honeyd
	Dodson et al.
	Bauer et al.
	Cifranic et al.
	Serbanescu et al.
	DiPot
	Lelie et al.
	Honware
	SIPHON
	Thingpot
	Honeytrack
Deployment Period	
<2 months	CryPLH
	Honware
	Thingpot
	Serbanescu et al.(
2-6 months	SIPHON
	Bauer et al.
	Honeytrack
>6 months	DiPot
	Hilt et al.
	Dodson et al.
	Lelie et al.
Hosting Location	
external	Serbanescu et al.
	DiPot
	SIPHON
	Holczer
	Dodson et al.
	Honeytrack
	ThingPot
internal	Simoes et al.
	Cifranic et al.
	Lelie et al.
	Hilt et al.
Security	
firewall	Piggin and Buffey
	Simoes et al.
	Digital Bond
	Pliatsios et al.
	Honware
Logging	
network-based	Conpot
	HoneyPHY
	Piggin and Buffey
	CryPLH

	Dodson et al.
	Gridpot
	Digital Bond
	Cifranic et al.
	Serbanescu et al.
	Pliatsios et al.
	DiPot
	Lelie et al.
	SIPHON
	Simoes et al.
	Hilt et al. (
	Bauer et al.
host-based	Hilt et al.
	Digital Bond
	Honware
Detection	
signature-based	Digital Bond
	Serbanescu et al.
	HoneyPHY
	Piggin and Buffey
	Simoes et al.
	Cifranic et al.
	Lelie et al.
anomaly-based	GridPot
Visualization	
geographical	Lelie et al.
	DiPot
statistical	Lelie et al.
	Hilt et al.
	Digital Bond
	Cifranic et al.
Alert	
	Simoes et al.
	Digital Bond

Appendix B

Feature Relationships

Relations between meta-level features within the Deceptive functionalities

- **(1,1)** —
- (1,2) - Size-Interaction - independent: *Since interaction is defined as the quality of implemented service(s), the size of the honeypot system has no influence on the interaction possibilities.*
- (1,3) - Size-Virtualization - independent - *The size of the honeypot system has no influence on virtualization possibilities, besides the costs which are explicitly excluded here.*
- (1,4) - Size-Services - affects - *A net needs to have additional services to enable communication between the devices. Therefore, the size of the honeypot system affects the (number of) services that the honeypots requires.*
- (1,5) - Size-External Persuasion - independent - *The size of the honeypot system has no influence on possibilities for external persuasion, such as the creation of a cover company, or active advertisement of the honeypot.*
- (1,6) - Size-Physical Process - independent - *In case of multiple independently deployed pots, you will need multiple instances of physical processes (either models or real plants). However, theoretically it is possible, since cost is not included in this matrix.*

- (2,1) - Interaction-Size - independent - *The quality of the implemented service(s), has no influence on the size (pot, pots, net) of honeypot system.*
- **(2,2)** —
- (2,3) - Interaction-Virtualization - affect - *The quality of implemented service does affect virtualization possibilities. When choosing high interaction, you could either go virtualized, or use a real device. However, when opting for low-interaction, you could only go virtual, as a real device has high quality of service(s) per definition*
- (2,4) - Interaction-Services - independent - *A higher interaction level, implies a higher quality of implemented service(s) and does not influence the number or type of service.*
- (2,5) - Interaction-External Persuasion - independent - *No influence, since both low and high interaction honeypots could have any form of external persuasion*
- (2,5) - Interaction-Physical Process - independent - *No influence, since both low and high interaction honeypots could have any form of physical process (simulation)*

- (3,1) - Virtualization-Size - independent - *The type of virtualization has no influence on possibilities for the size of the honeypot system, except for increased costs when multiple real devices are deployed.*
- (3,2) - Virtualization-Interaction - affect - *The virtualization option does have influence of interaction level. When going for a real device, the interaction level is high per definition. However, when using a digital twin or service imitation, all levels of interaction can be*

reached.
- **(3,3)** —
- (3,4) - Virtualization-Services - affect - *The level of virtualization determines whether which (number of) services should be implemented, and if the user needs to do this himself.*
- (3,5) - Virtualization-External Persuasion - independent - *Trivial (1,6)*
- (3,6) - Virtualization-Physical Process - affect - *It seems pointless or even impossible to implement a real sensor/actuator network when using a service imitation, even though, in the other cases all combinations can independently occur.*

- (4,1) - Services-Size - affect - *The choice for the number and types of services, has an influence on the size of the honeypot system. For example, to be able to realize a honeynet, multiple services are needed to interconnect the system.*
- (4,2) - Services-Interaction - independent - *The amount or type of services has no influence on the quality of those implemented services.*
- (4,3) - Services-Virtualization - affect - *The choice for a limited number and type of services is incompatible with the use of a real device as the device itself comes with it fixed services. Also, there may be vendor-specific proprietary services, which can only be realized by using real devices.*
- **(4,4)** —
- (4,5) - Services-External Persuasion - independent - *The selection of services has no influence on the possibilities for external persuasion*
- (4,6) - Services-Physical Process - independent - *The selection of services has no influence on the possibilities for physical process (simulation) implementations*

- (5,1) - External Persuasion-Size - independent - *Non-technical external persuasion measures such as blog posts or a cover company have no influence on the possible size of the honeypot system.*
- (5,2) - External Persuasion- Interaction - independent - *Non-technical external persuasion measures such as blog posts or a cover company have no influence on the interaction level.*
- (5,3) - External Persuasion-Virtualization - independent - *Non-technical external persuasion measures such as blog posts or a cover company have no influence on the virtualization options.*
- (5,4) - External Persuasion-Services - independent - *Non-technical external persuasion measures such as blog posts or a cover company have no influence on the technical services offered to the attacker.*
- **(5,5)** —
- (5,6) - External Persuasion-Physical Process - independent - *Non-technical external persuasion measures such as blog posts or a cover company have no influence on the possibilities for (the simulation of) a physical process.*

- (6,1) - Physical Process-Size - independent - *Presence and type of physical process has no influence on the size of the honeypot*
- (6,2) - Physical Process-Interaction - independent - *Presence and type of physical process has no influence on quality of implemented services*
- (6,3) - Physical Process-Virtualization - affect - *Presence and type of physical process affects the virtualization, since a real plant requires proper input (in terms of energy) of a specific control device in order to function.*
- (6,4) - Physical Process-Services - independent - *Presence and type of physical process has no influence on the services that can be offered*
- (6,5) - Physical Process-External Persuasion - independent - Presence and type of physical process has no influence possibilities for external persuasion

- **(6,6)** —

Relations between meta-level features within the Deployment-related functionalities

- **(1,1)** —
- (1,2) - Scale-Deployment Period - independent - *The number of deployed honeypots does not influence the period that they could or should be deployed.*
- (1,3) - Scale-Hosting Location - affect - *The number of honeypots deployed affect the hosting location. When hosting a large volume of decoys, one should think about where to host from (cloud, private IP range).*
- (1,4) - Scale - Security - independent - *The number of honeypots should not influence the measures that can be taken to control traffic from and to these honeypots.*

- (2,1) - Deployment Period-Scale - independent - *The duration of deployment has no influence on the number of honeypots that can be deployed*
- **(2,2)** —
- (2,3) - Deployment Period-Hosting Location - independent - *The duration of deployment has no influence on the possible location that the honeypot is deployed on.*
- (2,4) - Deployment Period-Security - affect - *The duration of deployment has no influence on possible security measures/configurations for the honeypot*

- (3,1) Hosting Location-Scale - affect - *When a location is determined, it could affect the number of pots that can be deployed (for example when renting some cloud instance to host from)*
- (3,2) Hosting Location - Deployment Period - independent - *The location where the honeypot is hosted from, has no influence on the possible duration of deployment*
- **(3,3)** —
- (3,4) - Hosting Location-Security - affect - *When a honeypot is hosted inside an internal (production) network, certain security measures might be in place to control traffic or limit misuse of the system. However a standalone honeypot, hosted from an arbitrary location does not necessarily have those (similar) security measures.*

- (4,1) Security-Scale - independent - *Traffic control from and to the honeypot does not influence the number of honeypots that could be deployeed*
- (4,2) Security-Deployment Period - independent - *The security measures/configurations for the honeypot have no influence on possible duration of deployment.*
- (4,3) Security-Hosting Location - affect - *While a honeypot without any traffic restrictions would suffice if it was standalone and Internet-connected, it would not be suitable to host a similar honeypot inside an active production. network.*
- **(4,4)** —

Relations between meta-level features within the User-related functionalities

- **(1,1)** —
- (1,2) - Logging-Detection - affect - *The type and extent of logging determines what kind of detection algorithmes can be implemented*
- (1,3) - Logging-visualization - affect - *The type and extent of logging determines what kind of visualization can be applied.*
- (1,4) - Logging-Alert - affect - *The type and extent of logging determines on which interactions an alert/security event can be triggered*

- (2,1) - Detection-Logging - independent - *Detection needs a logging functionality in order to analyze interaction. However, logging itself is not affected by a certain detection implementation.*
- **(2,2)** —
- (2,3) - Detection-visualization - independent - *Detection does not affect visualization possibilities*
- (2,4) - Detection-Alert - affect - *The outcome of detection algorithms determine whether malicious activity has taken place, and thus whether an alert/security event needs to happen.*

- (3,1) Visualization-Logging - independent - *visualization choices do not affect logging functionalities.*
- (3,2) Visualization-Detection - independent - *visualization choices do not affect Detection functionalities*
- **(3,3)** —
- (3,4) Visualization-Alert - independent - *visualization choices do not affect alert functionalities*

- (4,1) Alert-Logging - independent - *A certain type of security event manager or other alerting function does not influence logging functionalities.*
- (4,2) Alert-Detection - independent - *A certain type of security event manager or other alerting function does not influence Detection functionalities.*
- (4,3) Alert-visualization - independent - *A certain type of security event manager or other alerting function does not influence visualization functionalities.*
- **(4,4)** —

Appendix C

Observation Targets

Reconnaissance

- **Active scanning** ***(T1595)***
  To enable: Interface that attacker IP address can communicate with, consisting of 1 or multiple ports.
  To log: Code/software that captures network traffic or registers incoming connection requests from source IP to specific ports of host.

- **Gather Victim Org Information** ***(T1591)***
  To enable: Publicly available resources that contain information about target (website, social media).
  To log: Functionality in resource that registers visitors (Web Server logs).

- **Phishing for information** ***(T1598)***.
  To enable: Electronic conversation service (Email, chat).
  To log: Functionality that saves chats.

- **Wireless sniffing** ***(T0887)***
  To enable: Wireless access point that is transmitting signals.
  To log: Not possible when passively sniffing

Delivery

- **Spearphishing Attachment** ***(T0865)***.
  To enable: Electronic conversation service that supports file attachments.
  To log: Functionality that saves chats.

- **External Remote Services** ***(T0822)***
  To enable: an access mechanism service (VPN, citrix, etc), mostly via a corporate network. (Access to valid accounts is often a requirement)
  To log: Capture of network traffic.

- **Internet Accessible Device** ***(T0883)***
  To enable: An Internet-connected service that is publicly available (VNC/RDP/etc).
  To log: Capture of network traffic.

- **Replication Through Removable Media** ***(T0847)***
  To enable: Either physical device, or virtual machine (with file system) that is connected

with a physical device (can be then be targeted via shared network drives).
To log: Software/code that registers Drive, File or Process creation and/or file access.

- **Hardware Additions *(T1200 Enterprise)***
  To enable: Physical device or network access point that provides opportunity for physical connection (USB/Ethernet port)
  To log: piece of software on device that registers the physical connection of attacker's hardware.

- **Transient Cyber Asset *(T0864)***
  To enable: A possibility to physically connect the the asset to target network.
  To log: Capture network traffic from the transient asset.

- **Rogue Master *(T0848)***
  To enable: Multiple devices exchanging network traffic
  To log: Functionality that captures in and outgoing traffic to see if honeypot is being misused.

Exploit

- **Wireless compromise *(T0860)***
  To enable: Wireless access point that accepts connection from attacker.
  To log: Functionality that registers incoming wireless signals (Signal strength, timestamps,

- **Exploitation of Remote Services *(T0866)***
  Similar to T0822

- **Exploit Public-Facing Application *(T0819)***
  To enable: Application on a device (HMI, PLC) that is directly accessible from the Internet
  To log: Functionality on host that registers processes created by that application. No cybercriminals because an application is generally too specific for general exploits, and thus needs advanced knowledge of that specific application.

- **Command-Line Interface *(T0807)***
  To enable: shell simulation or real command line.
  To log: keystroke logger, syslogging, saving user input (script), command-line auditing/process creation auditing.

- **Execution through API *(T0871)***
  To enable: Functionality that responds to preprogrammed set of API commands, or real API including backend server.
  To log: Functionality that register API function calls (either by saving user input to script, or by integrating logging functionality in API backend server)

- **Native API *(T0834)***
  To enable: Functionality that responds to preprogrammed set of OS API commands, or the use of an actual OS.
  To log: Process creation events (windows)/ system/exec call auditing/tracing (Unix).

- **Scripting *(T0853)***

- **User Execution *(T0863)***
  To enable: Operating system (for process & file creation, and command execution).
  To log: Process creation events (windows)/ system/exec call auditing/tracing (Unix)

- **Valid Accounts *(T0859)***
  To enable: Login functionality (scripted/simulation or real), and set of leaked valid credentials.
  To log: Functionality that registers login attempt and credentials used.

- **Default credentials *(T0812)***
  To enable: Login functionality (scripted/simulation or real), and set of valid credentials of device/service imitated.
  To log: Functionality that registers login attempt and credentials used.

Install Modify (stage I)

- **Change Operating Mode *(T0858)***
  To enable: Physical device with keyswitch, or well-implemented API.
  To log: Functionality that registers (unexpected) changes in device state, such as restarting, shutdown etc.

- **System Firmware *(T0857)***
  To enable: Physical device that contains hardware that is compatible with the intended malicious firmware upgrade
  To log: Functionality deeply embedded in host that captures modification of firmware.

- **Module Firmware *(T0839)***
  To enable: Physical entity within device that contains hardware that is compatible with the intended malicious firmware upgrade
  To log: Functionality deeply embedded in host that captures modification of firmware.

- **Hooking *(T0874)***
  To enable: Program or Operating system that uses API functions to redirect calls. Example hooking method is DLL injection in Windows (containing API functions).
  To log: Functionality that registers modification or replacement of files
  Different types of Hooking, can also be embedded in tools (such as Carberp, Empire,)

- **Project File Infection *(T0873)***
  To enable: CPS specific engineering program/software that uses objects, variables, configurations stored in project files.
  To log: Functionality that registers the modification of such a file.

- **Indicator Removal on Host *(T0872)***
  To enable: command-line (for remove-action), file-system (to host the indicator that is being removed)
  To log: Keylogger (to catch 'rm filename'-commands), or scan process list for SDelete (Windows).

- **Masquerading *(T0849)***
  To enable: Filesystem on which malicious applications or executables can be disguised.
  To log: Functionality that registers file paths, process names such that they can be compared

with reference (normal) path/name combinations

- **Rootkit** ***(T0851)***
  To enable: Access to kernel (presence of drivers), Operating systems
  To log: Functionality that is able to register modifications made to the kernel

C2

- **Commonly Used Port** ***(T0885)***
  To enable: Service allowing outgoing network traffic
  To log: Functionality that captures (outgoing) network traffic

- **Connection Proxy** ***(T0884)***
  To enable: Service allowing outgoing network traffic
  To log: Functionality that captures destination IP-address of (outgoing) network traffic. Possibly compare that with a list of known malicious proxies.

- **Standard Application Layer Protocol** ***(T0869)***
  To enable: Service providing outbound L7 protocol (HTTP, FTP, DNS, etc).
  To log: Capture of network traffic (log IP addresses, use signatures).

Act

- **Network connection Enumeration** ***(T0840)***
  To enable: System tools (netstat) to inspect connections on host, network adapters that enable the inspections and multiple devices on network (net).
  To log: functionality on host that registers use of these tools.

- **Network sniffing** ***(T0842)***
  To enable: Available network interface to be able to capture traffic. Multiple devices on network (net) needed for creation of traffic
  To log: Functionality on host that registers the use of a network interface.

- **Remote System Discovery** ***(T0846)***
  To enable: Network interface on host, and multiple devices (net) such that they can be discovered on the network.
  To log: Functionality that captures the remote service scans (network traffic)

- **Detect Operating Mode** ***(T0868)***
  To enable: Functionality that show current operating mode of device. Can either be
  To log: Difficult, use signature to detect keystate detection (Triton)

- **I/O Image** ***(T0877)***
  To enable: Process (model) or other functionality that mimics the input and output of a PLC.
  To log: Functionality that registers the access to the memory region where this Image is stored.

- **Automated collection** ***(T0802)***
  To enable: OS to run automated script/tool, and services (such as OPC) that can be used to enumerate/gather information about connected resources/servers/devices.
  To log: Functionality that audits process creation (by running the tool or executing the script).

- **Point & Tag Identification** ***(T0861)***
  To enable: Industrial protocols that exchange process environment specific information such as points (inputs, outputs, memory locations, variables) and tags (the corresponding identifiers)
  To log: Functionality that captures network traffic, based on (abnormal) enumeration of inbound or outbound traffic using industrial protocols.

- **Data from Information Repositories** ***(T0811)***
  To enable: An accessible filesystem that contains (fake) sensitive data
  To log: Functionality that captures network traffic (stream) to identify file transfer. Or functionality that registers when a file is being accessed (Windows - Object access: File System - Audit policy)

- **Monitor Process State** ***(T0801)***
  To enable: Physical process (model) and a service in order to retrieve information about the states in that process (model)
  To log: Functionality that registers the use of a service used to retrieve process state information. For example: capturing network traffic when using an industrial protocol to request the values of operational variables.

Delivery (stage II)

- **Lateral Tool Transfer** ***(T0867)***
  To enable: More than one (virtual) machines, with a service that offers file transfer
  To log: Functionality that captures traffic over internal network

- **Program Download** ***(T0843)***
  To enable: Vendor-specific management protocol that provides transfer of a user program to a controller
  To log: Functionality on host that registers program modification. Possibly also network traffic capture that registers specific inbound protocols associated with program download).

- **Remote Services** ***(T0886)***
  To enable: At least two network interfaces/services (net) that provide a communication channel between them.
  To log: Functionality that registers the flow data over the channel.

Install/Modify (stage II)

- **Modify Parameter** ***(T0836)***
  To enable: Program on (virtual) control device where an attacker can change values.

To log: Functionality on host that registers the modification of values (screen capture, keylogger, saving parameter values).

- **Brute Force I/O *(T0806)***
  To enable: Program (for example CODESYS,) on control device that offers modification of I/O point values.
  To log: Either a functionality in the application/program that registers the modification of I/O point values, or capturing network traffic (excessive I/O connections).

- **Manipulate I/O Image *(T0835)***
  To enable: Similar as above, or access to direct memory for overriding the I/O table.
  To log: Functionality on host that registers modification in image table (in PLC's internal storage.

- **Block Serial COM *(T0805)***
  To enable: (Virtual) serial COM port that is accessible from the device.
  To log: Functionality that registers opening of COM connection (on operating system)

- **Man in the Middle *(T0830)***
  To enable: Operating system and network interface on which adversary can sniff and/or modify incoming/outgoing network traffic
  To log: Capture network traffic (source, destination, modifications).

- **Spoof Reporting Message *(T0856)***
  To enable: Service on at least two devices that can exchange messages (PLC and Workstation for example)
  To log: capture network traffic between devices.

- **Modify Program *(T0889)***
  To enable: Well-implemented industrial software/program and the ability to make changes to functions inside the program.
  To log: Functionality that registers modification of file and or verifies integrity (CRC, checksums).
  Requires in-depth knowledge about specific target CPS devices AND/OR interaction with physical process.

- **Data Destruction *(T0809)***
  To enable: Operating system and files such that attacker can either use standard system commands or specific tools to delete data.
  To log: Functionality on host that logs changes in filesystem, and/or functionality that logs shell (if deletion happens through command-line).

- **Denial of Service *(T0814)***
  To enable: Interface that attacker IP address can communicate with.
  To log: Functionality that captures number and content of incoming requests

- **Device Restart/Shutdown *(T0816)***
  To enable: Build-in function in most devices, using web interfaces, CLI and protocols. Could also be simulated in script, by temporarily disabling access to attacker (Conpot, set counter and freeze access to all protocols).

To log: Functionality on host that logs either commands (shell) or event logs (shutdown/restart)

- **Service Stop *(T0881)***
  To enable: Services on a system that an attacker is able to terminate or disable.
  To log: Either functionality on host that registers process termination, or key/shell logger in case of standalone service (imitation)

- **Unauthorized Command Message *(T0855)***
  To enable: Functionality on device that allows an incoming command message and acts accordingly
  To log: Functionality that registers all command messages send (so, capturing network traffic)

- **Modify Alarm Settings *(T0838)***
  To enable: Functionality that acts as overall reporting system for the management of physical processes. (modifying in memory code to fixed values or tampering with assembly level instruction code)
  To log: To log: Functionality in application that handles alarm, that registers any modifications.

- **Alarm Suppression *(T0878)***
  To enable: Functionality that acts as overall reporting system for the management of physical processes
  To log: Functionality in application that handles alarm, that registers any modifications.

- **Block Command Message *(T0803)***
  To enable: Multiple devices (origin, destination of message), services that are responsible for message exchange. Access to interface in order to hinder exchange.
  To log: No capture of network traffic (since this is hindered). Functionality in application on host, that registers any modification that results in message blocking.

- **Block Reporting Message *(T0804)***
  To enable: Multiple devices (origin, destination of message), services that are responsible for message exchange. Access to interface in order to hinder exchange.
  To log: No capture of network traffic (since this is hindered). Functionality in application on host, that registers any modification that results in message blocking.

Execute ICS Attack

- **Damage to Property *(T0879)***
  To enable: Physical process, that can be controlled by the adversary.
  To log: Functionality that senses/records the change within the physical process, with respect to 'normal behaviour'.

- **Manipulation of Control *(T0813)***
  To enable: Functionality that imitates physical process, that can be controlled by the adversary.

To log: Functionality that senses/records the change within the physical process (model), with respect to 'normal behaviour'.

- **Manipulation of View *(T0815)***
  To enable: Functionality that imitates physical process, that can be view by the adversary.
  To log: Functionality that senses/records the change within graphical representation of the physical process (model).

- **Theft of Operational Information *(T0882)***
  To enable: Resource that contains information about the physical process (model) .
  To log: Functionality that senses/records access to the resource.

Excluded ICS MITRE techniques:

- Drive-by Compromise
- Supply Chain Compromise
- Screen Capture