Towards Security-Aware Virtual Environments for Digital Twins

Matthias Eckhart
Christian Doppler Laboratory "SQI", Inst. of Information Systems Engineering, TU Wien
Vienna, Austria
matthias.eckhart@tuwien.ac.at

Andreas Ekelhart
SBA Research
Vienna, Austria
andreas.ekelhart@sba-research.org

ABSTRACT

Digital twins open up new possibilities in terms of monitoring, simulating, optimizing and predicting the state of cyber-physical systems (CPSs). Furthermore, we argue that a fully functional, virtual replica of a CPS can also play an important role in securing the system. In this work, we present a framework that allows users to create and execute digital twins, closely matching their physical counterparts. We focus on a novel approach to automatically generate the virtual environment from specification, taking advantage of engineering data exchange formats. From a security perspective, an identical (in terms of the system's specification), simulated environment can be freely explored and tested by security professionals, without risking negative impacts on live systems. Going a step further, security modules on top of the framework support security analysts in monitoring the current state of CPSs. We demonstrate the viability of the framework in a proof of concept, including the automated generation of digital twins and the monitoring of security and safety rules.

CCS CONCEPTS

• **Security and privacy** → **Intrusion detection systems**; • **Computer systems organization** → *Embedded and cyber-physical systems*;

KEYWORDS

Cyber-physical systems; industrial control systems; digital twin; simulation; security monitoring; AutomationML

1 INTRODUCTION

The terms *Industry 4.0* and *Smart Manufacturing* refer to the vision of a highly efficient, autonomous and flexible production process. At the core of both aforementioned concepts are cyber-physical systems (CPSs). A CPS is a system that combines computational (e.g., computing hardware/software) and physical (e.g., actuators) components, allowing the system to interact with the real world [4]. Moreover, CPSs can integrate networking capabilities, a feature that is a cornerstone of interconnected and autonomously operating manufacturing systems.

However, increased digitization and connectivity open up new attack vectors that may not only put an organization's assets at risk, but could also endanger human life. This is especially important for industrial control systems (ICSs) — a subset of CPSs — where safety has been the main focus. In this context, it should be noted that security issues can lead to safety implications. In fact, the lack of adequate measures to secure CPSs in critical infrastructures (e.g., sewage treatment plants [30]) could even have severe consequences for public safety [21, 22].

Security testing, monitoring, and intrusion detection, as defined in industrial standards and guidelines such as IEC 62443 [14] and NIST SP 800-82 [31], are important measures to fortify industrial environments. Due to the criticality of the running systems, testing in the production environment is not recommended. The setup and maintenance of test environments on the other hand, is expensive and time consuming, often leading to incomplete and outdated environments. A similar issue can be identified with focus on the evaluation of research results in this domain. As an example, intrusion detection in CPS got much attention over the past years, but the evaluation datasets are frequently not published and the cyber-physical setup cannot be reproduced most of the time. This hinders adoption and independent comparison [15, 24].

In this paper, we propose a novel framework named *CPS Twinning* to build and maintain fully functional digital twins of CPSs. The term "Digital Twin" was coined by Shafto et al. [29] and describes the use of holistic simulations to virtually mirror a physical system [26]. Adopting such a concept could enable operators to monitor the production process, test changes in a virtual, isolated environment, and to further strengthen the security and safety of CPSs.

A recent study by Rubio et al. [27] suggests that a virtual replica of the physical process may be leveraged for security purposes, however, leading to new issues concerning the creation and management as a consequence thereof. While the manual creation of a virtual environment for digital twins is time consuming and error-prone, we want to produce the environment completely from specification. This approach is efficient, reusable, and moreover, guarantees an identical setup. Ideally, the specification of the CPS is already defined and maintained as part of the system engineering process [19, 20] by means of standardized data formats, such as AutomationML (AML) [10].

We consider two main modes of operation of the virtual environment, either (i) in a simulation mode, operating independently of the physical environment, offering the possibility to monitor and explore a virtual clone without risk, or (ii) in a replication mode, replaying the events from the physical environment for visualization and analysis. On top of this virtual representation, multiple security features can be established. For example, security and safety rules stated as part of the specification can be automatically monitored on the basis of digital twins. In addition, new physical devices can be connected and tested in the virtual environment, without influencing production systems. Security testers also have the possibility to freely explore and attack a virtual replication of the production setup. With this approach, security can be seamlessly integrated in the entire production lifecycle, starting from the engineering phase.

The novel contributions of this paper can be summarized as follows:

- We propose a framework that provides a security-aware environment for digital twins.
- We demonstrate the feasibility of the proposed framework by providing a prototypical implementation, supporting the virtual replication of the network topology, programmable logic controllers (PLCs) and their control logic, human machine interfaces (HMIs), and physical devices (e.g., motor). To accelerate the development of this prototype, we integrated two existing open-source tools.
- We show how such virtual environments for digital twins can be automatically generated from specification (e.g., AML).
- We introduce security relevant use cases for digital twins in CPSs and show how security and safety rules can be monitored.

The remainder of the paper is organized as follows. First, in Section 2, we motivate the need for a security-aware virtual environment for digital twins by presenting four use cases. In Section 3, we outline the architecture of CPS Twinning and discuss its underlying components. Section 4 presents a proof of concept implementation of CPS Twinning, followed by Section 5 where we describe related work. Finally, in Section 6 we provide concluding remarks and discuss future work.

2 USE CASES

In this section, we describe possible use cases for digital twins to improve the security of CPSs and support security analysts in designing defenses.

Intrusion Detection. In a literature review conducted by Mitchell and Chen [23], the authors concluded that behavior-specification-based intrusion detection [33] may prove to be a promising approach to uncover intruders, while keeping the false positive rate at a minimum. Intrusion detection systems (IDSs) that use this particular technique attempt to detect malicious activity by identifying deviations from a defined model of benign behavior [23]. By using a detailed specification of the production system as a template for digital twins, they could monitor their own behavior and report deviations from their specification. This could, e.g., include the detection of unknown devices, unspecified connections, and changes in the control logic. In addition, specified security and safety rules could be monitored inside the virtual environment.

Another scenario is the use of additional, independent sensors to discover manipulations and defects [12, 18]. While the integration, monitoring, and correlation of new sensors in a production environment is not trivial, this could be handled completely inside the digital twin. After placing the additional sensor in the physical environment, the readings are replicated to its digital twin and correlated with measurements from other sensors.

System Testing & Simulation. Digital twins can be leveraged to perform system tests and simulations. Security analysts can explore a production clone, instead of relying on documentation and theoretical attack vectors. Furthermore, real devices can be tested by first connecting them in the virtual environment. For instance, if the real device is connected as a replacement of the existing digital-twin PLC, the operator could observe the behavior of the new PLC inside the virtual environment. Modules to automatically record and compare specific configurations could be a further extension. Moreover, experimenting with configurations in a virtual environment provides the possibility to detect problems and incompatibilities early, without costly setups.

Detecting Misconfigurations. Another use case is to detect mismatches of the real environment and the maintained specification. If, for instance, a physical device is added to the real environment without adapting the specification, the mismatch could be detected and reported. The same applies to the case where a physical device is not consistent with its virtual representation, due to misconfiguration or manipulation by an attacker.

Penetration Testing. Penetration testing of ICSs must be carefully designed, since network scans such as a ping sweep may cause the system to behave in an unexpected manner, potentially harming manufacturing equipment and human health [11]. Thus, preferably maintenance windows are selected for penetration tests in the live environment. However, temporarily stopping or limiting the operation of the plant is costly and often not feasible, especially for critical infrastructures. Building test environments specifically for conducting penetration testing may also not be a viable alternative, considering cost and time implications. With a virtual mirror of the production environment, security analysts could identify weaknesses and subsequently test countermeasures, before implementing them in production.

3 FRAMEWORK

This section describes CPS Twinning, a digital-twin framework to support the presented use cases in Section 2.

As the high-level view given in Figure 1 illustrates, the architecture of the framework is composed of two main modules, viz. the generator and the virtual environment. The generator module takes engineer- and domain-specific knowledge as input to create the virtual environment. Once the digital twins and the network topology have been generated, the virtual environment can operate in two modes. First, the virtual environment provides a simulation mode, in which the digital twins run independently from the physical environment. Second, the replication mode records events such as network traffic from the physical environment and replicates them in the virtual setup. On top of both modes, the framework includes multiple modules that can be activated on demand, such as monitoring, security analysis and intrusion detection. As the framework should be extensible, we propose a multi-module architectural approach. In Section 4 we demonstrate a first prototypical implementation of CPS Twinning.

In the following, a detailed description of each component is given.

3.1 Input Knowledge

A vital part of the presented idea is to leverage the specifications of CPSs designed throughout the engineering phases. In this way, the virtual environment can be generated based on existing artifacts, instead of building it from scratch. The advantage of this

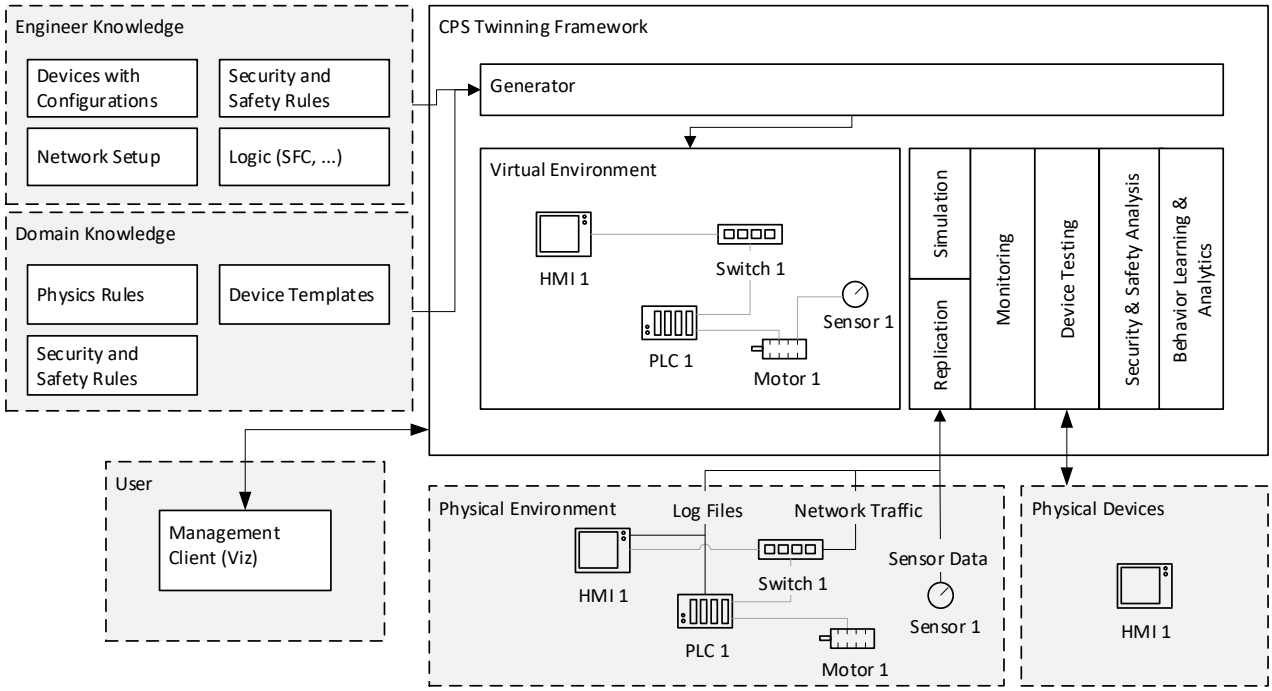

Figure 1: Architecture of CPS Twinning

approach is that the generation of the virtual environment is automatic and thereby replacing time-consuming work with an efficient and scalable solution. Furthermore, this approach yields reproducible results, meaning that the virtual environment can be rebuilt identically at any time, as long as the specification exists. Any changes to the physical environment without adapting the respective specification results in a divergence from its virtual counterpart. Consequently, the specifications of CPSs should be kept in sync, fostering a traceable and well documented environment.

Creating and maintaining a detailed specification of a CPS that enables the generation of a complete virtual replica entails additional effort. Ideally, the organization already uses standardized languages (such as AML) to design, exchange, and preserve their setups.

3.1.1 Engineer Knowledge. Engineer knowledge describes the information that is unique to the environment at hand. It comprises the design of the complete process, including system components, network information and the internal behavior of CPSs. Based on the process definition, the topology of the environment and logical connections between individual components can be derived. In essence, the topology is modeled by specifying the components (e.g., name, product type, vendor), their corresponding traits (e.g., I/O channels) and their configuration (e.g., IP and MAC address).

Furthermore, defining hierarchical relationships within components is also an important aspect to consider when modeling a system. In particular, devices can be examined during operation to check if they adhere to the specification, e.g., verifying that a system provides only those services that were defined beforehand. Moreover, this modeling approach enables CPS Twinning to enforce fine-grained policies and constraints on all hierarchical levels.

Another aspect to consider is the explicit definition of the communication path from one host to another through logical connections and endpoints [3]. Besides using these details to generate the network setup of the virtual environment, this data can also serve as a basis for implicit security rules. More specifically, the framework could monitor the traffic flow and check if network packets contain permitted addressing and protocol information. For example, a communication path between an HMI and a PLC could be defined in the specification, including the used application layer protocol (e.g., Modbus) and the underlying details of the request protocol data unit (e.g., function code). Consequently, a whitelist for network-level monitoring can be derived from the explicit mapping of the network layout.

As far as the behavior of CPSs is concerned, control logic can also be attached to device specifications. For example, a program implemented as a sequential function chart (SFC), one of the programming languages defined in IEC 61131-3 [13], can be referenced as a program block of a PLC instance. Since SFC is a standardized programming language, the referenced code can often be directly transferred to the physical PLC for execution or at least converted to vendor-specific dialects. As a result, no additional effort from an engineering perspective is incurred and realistic simulations can be

guaranteed, since the control logic deployed in both environments (physical and virtual) is identical. Including process information is also valuable for security-related modules (e.g., IDS).

Finally, we emphasize the explicit integration of safety and security rules into the specification. In the course of the process design, engineers can explicitly state safety rules that define normal operation. This knowledge enables the framework to determine whether the process is in a safe state and thereby supplementing safety instrumented systems (SIS). For example, value limits for device variables (e.g., min. temperature and max. speed) can be stated. Conditional statements, as well as comparisons between variables from different digital twins offer more complex rules.

While engineers provide the knowledge concerning safety risks, security analysts could contribute rules that indicate benign or malicious behavior. For example, a rule could be specified that restricts the network communication between an HMI and a PLC, depending on the PLC's internal state.

3.1.2 Domain Knowledge. Domain knowledge refers to information that is independent from a specific physical process, meaning that it can be defined once and then shared among several organizations. Industrial equipment producers for example, are in the position to define each of their devices and then provide these artifacts in the form of device templates. These templates could include domain-specific knowledge from experts in the field of mechanical, electrical, and control engineering. Engineers could then either reference these ready-made templates or use them as a starting point to avoid redundant work. Moreover, producers and security professionals can also provide safety and security rules and update them continuously. For example, the specification of a tank could include the maximum allowable fill level, whereas the specification of a PLC could predefine rules for protecting the integrity of the firmware.

Previous research on intrusion detection for CPSs focuses also on attack detection based on the physical properties of a system [34]. In this work, we propose the explicit definition of physics rules and thereby implicitly defining input that is relevant for the security and safety analysis. Consider an attack targeting a chemical process with the goal to overflow a tank. The attacker attempts to manipulate the liquid level sensor measurements in order to trick the control system into filling the tank beyond its limit. By keeping track of the previous fill and drain control activities, the framework could calculate an expected liquid level inside the tank by applying principles of fluid dynamics and compare it to the sensor data. Inconsistencies would then point to malicious behavior or defects.

3.2 CPS Twinning Framework

The CPS Twinning framework represents the main contribution of this paper. It includes a generator, the virtual environment and modules that interact with digital twins. Each component will be explained further in the following subsections.

3.2.1 Generator. The generator is responsible for transforming the specification into a virtual environment. As a first step, the specification is parsed to extract the topological structure of the network, the devices with their corresponding configuration and the security and safety rules. After that, the virtual environment

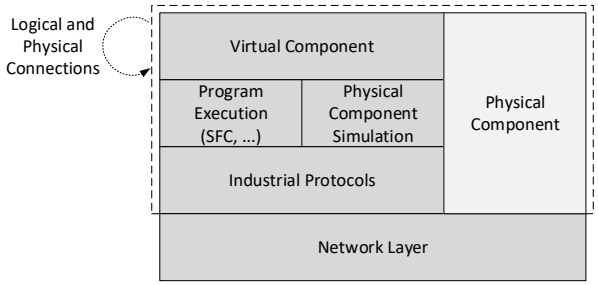

Figure 2: Layers of CPS Twinning

is built by generating virtual objects and applying their respective configuration. Finally, the parsed rules are stored in an abstract representation for later evaluation by the security and safety analysis module.

3.2.2 Virtual Environment. This component forms the core of CPS Twinning and provides the virtualized network infrastructure as well as a runtime for virtual devices. As a realistic simulation of the physical process is paramount for the use cases presented in Section 2, the virtualized CPSs should match their physical counterparts as closely as possible. In particular, this includes the execution of control logic, network protocols, device types and the physical equipment. Figure 2 depicts an overview of the framework design. The bottom layer provides a foundation for the CPSs network infrastructure. On top of it are industrial protocols that can be used by digital twins. The digital twins reside one layer above the industrial protocols and can either be emulated by executing the control logic or simulated in case the twin should replicate only a physical component.

While the "cyber" part of a CPS can be reconstructed identically in the virtual environment, physical components, such as sensors or actuators, as well as their interactions with the real world, must be simulated. As suggested by Antonioli and Tippenhauer [2], this can be realized by implementing a file-based storage for sensor and actuator values, or even by integrating hardware-in-the-loop (HIL) solutions. In Figure 2, this is represented as the physical component layer, coexisting with the virtual component layer.

3.2.3 Simulation & Replication. After the automatic generation of the virtual environment according to the specification and the configuration of digital twins, the two modes of operation become available, i.e., replication and simulation.

In simulation mode, the digital twins run independently of their physical counterparts. Similar to virtual commissioning, this mode allows users to analyze process changes, test devices or even optimize manufacturing operations. Furthermore, security professionals can use this mode to perform security tests within the virtual environment, thereby avoiding potential implications of testing on a live system.

The replication mode on the other hand, mirrors data from the physical environment. Possible data sources to mirror in the virtual environment are log files, network communication and sensor measurements from the physical environment. Furthermore, sensors

could also be directly connected to the CPS Twinning framework. By means of data diodes or unidirectional gateways, the direction of the data flow could be restricted to avoid negative effects on the physical environment. It should also be noted that a direct connection between the physical and virtual environment is not a mandatory requirement. For example, data could be collected offline for a certain period of time and then used in replication mode. Yet, to ensure that the virtual replica continuously reflects the physical environment, the use of an automated data replication method is preferred.

3.2.4 Monitoring. Monitoring the process under control is an important aspect of the framework. Hence, the monitoring module provides an interface to assess the process state. In particular, sensor values and actuator states can be collected and prepared for analysis and visualization. In this way, the monitoring functionality of CPS Twinning can assist users in gaining a deeper understanding of the environment at hand, ensure its proper operation, and facilitate troubleshooting when issues occur.

Although this module is particularly useful in replication mode, when the primary objective is to monitor the physical process, insights into the simulated environment can also be of importance. More concretely, monitoring the physical process in replication mode and then switching to simulation mode allows to investigate a certain state. This approach may provide insight regarding the root cause that led to an unexpected behavior.

3.2.5 Device Testing. Due to the realistic virtualization of the physical environment, CPS Twinning facilitates virtual commissioning. This opens the possibility to test physical devices by integrating them into the virtual environment. The process states, as well as the behavior of the system under test can be monitored to verify that the system is working as expected. While the setup of an identical physical test environment is expensive and time-consuming, this approach may be considered as an attractive alternative to test the replacement of devices and to perform integration tests.

3.2.6 Security & Safety Analysis. As mentioned in Section 3.1, security and safety rules can be stated either implicitly or explicitly in the specification. After these rules have been extracted from the specification, this module performs an analysis during operation to detect abnormal conditions of the process in the virtual environment. Note that in replication mode, the physical environment is mirrored to its virtual counterpart; thus, it can be assumed that abnormalities emerge identically. Besides detecting abnormal process states during operation, this approach also provides the possibility to run simulations in the virtual environment in order to test the setup against violations of the specified rules.

In contrast to a SIS, the framework is able to identify abnormal conditions by correlating the state of digital twins. It has access to all states and events inside the virtual environment, and thereby, can also monitor state changes over time. With this information, relationships between variables can be analyzed in order to spot violations of defined safety rules. Moreover, sensors that are independent of the process under control may be used as an additional data source to report their view on the system's state. The advantage of this approach is that the framework provides a holistic view of the physical process.

With focus on security, the digital twins and their specification can serve as a foundation for a behavior-specification-based IDS. The IDS could take the twins' state as a primary input for analysis (host-based), but also audit the network traffic (network-based). Contrary to a behavior-based IDS, this approach does not require a training phase and may yield a low false positive rate [23], provided that the specification is correct and the virtual environment is consistent with the physical. The specified process knowledge could also help to detect semantic attacks on control systems, such as discussed in [6, 25].

3.2.7 Behavior Learning & Analysis. Learning the behavior of the virtual environment serves as valuable input for process optimization and anomaly detection. It may be possible to track down process bottlenecks or other factors that negatively affect the process or the quality of manufactured products. Furthermore, typical Industry 4.0 use cases, such as predicting the manufacturing throughput or the health of systems [17], may be also implemented on top of this module. With CPS Twinning, analysts can focus on learning algorithms and data interpretation, while the setup and monitoring is handled by the framework.

3.3 Management Client

Since users need the possibility to manage and control the virtual environment, a unified interface that provides access to all modules of the framework is required. However, protecting the framework from unauthorized users is essential. If adversaries gain access to CPS Twinning, they could obtain a detailed description of the physical environment and identify weak points of systems. This information can then be used to discover attack paths to desired targets. Likewise, the security and safety rules may inform an attacker about controls that are covered by the framework.

In addition, the management client may also provide visualization features to represent digital twins graphically.

4 PROOF OF CONCEPT

In this section we demonstrate how a physical environment can be modeled, and we present a first implementation of the introduced framework. While we can only show excerpts in this section, the complete prototype and scenario files can be found on GitHub[1].

A minimal setup serves as our physical environment, including a Siemens S7-1200 PLC, an HMI, a network switch to connect both components, and a conveyor belt that is driven by a motor. While the motor is operated by the PLC, the HMI is used to monitor and control the PLC via the Modbus TCP/IP protocol. The HMI (i.e., Modbus master) allows users to write holding registers on the PLC (i.e., Modbus slave) in order to start/stop the conveyor and to set the velocity respectively.

The engineer- and domain-knowledge (cf. Section 3.1) is the main input to create a virtual environment. To formulate the specification, we chose the data format AML [10]. AML aims to support the complete engineering chain of production systems by offering a standardized data exchange format for most of the artifacts (e.g., technical, topological, and control related information) within the engineering process. This XML-based data format considers both,

[1] https://github.com/sbaresearch/cps-twinning

a syntactical and a semantic level to describe the data objects, and furthermore, is flexible with respect to extensions and changes, which makes it a fitting candidate to model the required knowledge for our framework.

The core of the CPS Twinning framework has been developed in Python and is based on existing components, such as *Mininet* [16]. Mininet allows users to virtualize network environments and it is extensible, so we were able to build the CPS Twinning layers on top. Apart from Mininet, the framework also integrates the iec2c transcompiler that is included in the *MatIEC* project[2]. This compiler translates code written in a programming language of the IEC 61131-3 standard to C code. Further, we implemented a custom runtime to execute the code generated by iec2c in the context of a digital twin, enabling CPS Twinning to emulate the internal behavior of PLCs.

As a minimum requirement, the prototype depends upon the existence of an artifact that describes the characteristics of the digital twins. The prototype currently supports the generation of PLCs, HMIs, components without a network interface (e.g., motor) that only monitor the state of other digital twins, and default Mininet network nodes (e.g., switches). Moreover, the specification parser has been designed to extract data from an AML artifact describing the setup that will be presented in the following section. Thus, we expect users to either adapt the provided AML parser to their needs or write their own parser implementation for their preferred data format.

4.1 Scenario Specification

In the following, we explain the relevant parts of the scenario specification.

Figure 3 illustrates the exemplary physical process. For a better understanding, we describe the main elements of the specification and show excerpts of the artifact alongside.

When modeling the communication of hosts in AML, there are at least two views that can be defined, viz. the physical and logical network [3]. As an example for the physical network, `Wire1` connects the physical endpoint of `HMI1` with the physical endpoint of `Switch1`. The definition of the physical endpoint of `HMI1`, including a part of the network configuration of the HMI (i.e., IP address), can be seen in Listing 1. Moreover, the excerpt shows a logical device named `HMI` that contains a logical endpoint and the HMI variable `Velocity`.

```
<InternalElement Name="HMI1" RefBase="/Siemens HMI">
   <InternalElement Name="Portlist" ID="28d...">
      <ExternalInterface Name="Endpoint" ID="fb1...">
         <Attribute Name="ip">
            <Description>IP address</Description>
            <Value>192.168.0.2</Value>
         </Attribute>
         ...

   <InternalElement Name="HMI" ID="068..." RefBase="/HMI">
      <ExternalInterface Name="Endpoint" ID="068..." />
      <ExternalInterface Name="Velocity" ID="da4..."
      ↪  RefBaseClassPath="/HMIVariableInterface">
         <Attribute Name="type">
            <Value>int</Value>
         </Attribute>
         ...
```

Listing 1: Excerpt of the HMI specification

As shown in Listing 2, `Wire1` is used to connect `HMI1` to `Switch1`. In AML, a link can be modeled by using the `InternalLink` element [3]. Listing 2 depicts how the endpoints of both devices can be referenced via the `RefPartnerSideA` and `RefPartnerSideB` attributes. Note that the attribute value of `RefPartnerSideB` points to the physical endpoint of `HMI1` (cf. Listing 1).

```
<InternalElement Name="Wire1" RefBase="/PhysicalConnection"
↪  ID="652...">
   <ExternalInterface Name="EP1" ID="e5b..." />
   <ExternalInterface Name="EP2" ID="857..." />
   <InternalLink Name="Switch1 - HMI1"
   ↪  RefPartnerSideA="{a29...}:Endpoint1"
   ↪  RefPartnerSideB="{28d...}:Endpoint" />
   ...
```

Listing 2: Excerpt of a wire specification

In contrast to the physical network, the logical network models the exchange of data among hosts from an abstract perspective [3]. As Listing 3 shows, a logical connection exists between the HMI and the PLC, since both hosts exchange data via the network. In addition, the element named `LogicalConnectionA` includes protocol data units (PDU) that specify which PLC and HMI tags are exchanged. For instance, the PDU named `VelocityModbusTCPDataPacket` establishes a link between the `Velocity` tag of `PLC1` and `HMI1`.

```
<InternalElement Name="LogicalNetwork" ID="c51...">
   <InternalElement Name="LogicalConnectionA"
   ↪  RefBaseSystemUnitPath="/LogicalConnection" ID="82b...">
      <ExternalInterface Name="EP1" ID="ac0..." />
      <ExternalInterface Name="EP2" ID="cb2..." />
      <InternalElement Name="VelocityModbusTCPDataPacket"
      ↪  ID="fe0...">
         <InternalLink Name="PLC1 Velocity - HMI1 Velocity"
         ↪  RefPartnerSideA="{133...}:Velocity"
         ↪  RefPartnerSideB="{068...}:Velocity" />
         <RoleRequirements
         ↪  RefBaseRoleClassPath="/ModbusTCPDataPacket" />
      </InternalElement>
      ...
```

Listing 3: Excerpt of the logical network specification

Similar to `HMI1`, `PLC1` is specified as a physical device that includes a logical device. The PLC code has been implemented as a SFC and, as shown in Listing 4, is referenced directly within the AML artifact. A snippet of the control logic is also depicted in Figure 3, represented as a ladder diagram (LD). As can be seen in the LD, the start/stop tags control the output `Q0.0`, which in turn drives the motor control block that triggers the pulse train output (PTO).

[2] https://bitbucket.org/mjsousa/matiec

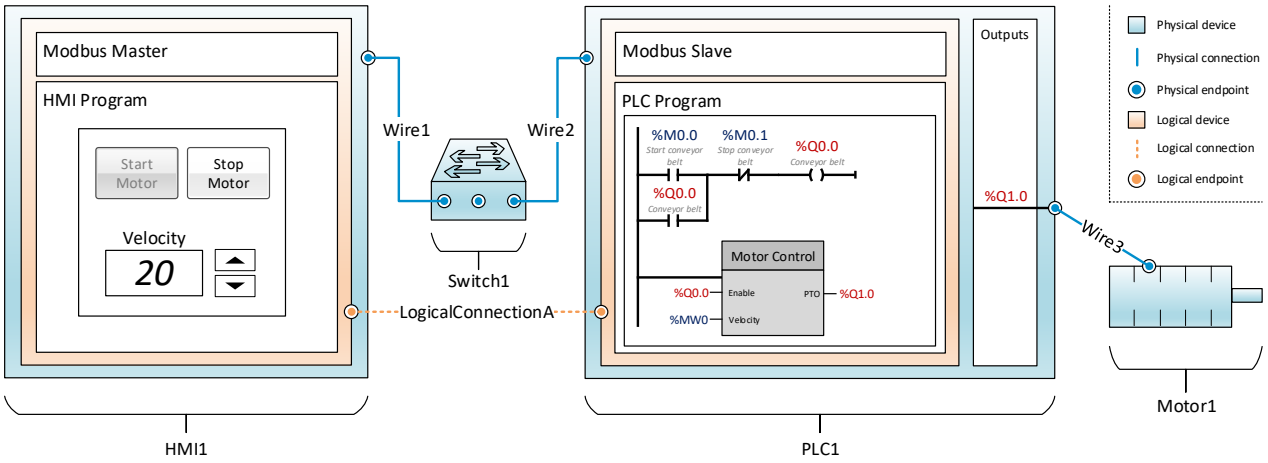

Figure 3: Illustration of the artifact that specifies the exemplary physical process

```
<InternalElement Name="Program" ID="133..." RefBase="/Main">
  <ExternalInterface Name="OB1" RefBase="/PLCopenXMLInterface"
  ↪  ID="2e2...">
    <Attribute Name="refURI" AttributeDataType="xs:anyURI">
      <Value>file:///Source/ConveyorSystem_SFC.xml</Value>
    </Attribute>
    ...
```

Listing 4: Referencing PLC code (SFC) in AML

At this point, we have introduced the engineer knowledge about the devices with their configuration, the network setup (physical and logical), as well as the control logic. When comparing it to Figure 1, it is evident that the specification of security and safety rules is still lacking. For this prototypical implementation of CPS Twinning, we want to demonstrate how a safety and a security rule can be defined.

Safety Rule. In this example, we want to ensure that the conveyor speed does not exceed a certain threshold. This safety rule could either be stated by the vendor of the motor or by a person designing the process. In order to define this rule in the specification, we used the interface class `VariableInterface` to reference the `Velocity` variable of the PLC program and then added a `RequiredMaxValue` constraint (cf. Listing 5). Since variables of the PLC code can be stated in the specification, the person defining the rule is not required to access the code or understand the underlying logic of the PLC program.

```
<ExternalInterface Name="Velocity" RefBase="/VariableInterface">
  <Attribute Name="refURI">
    <Value>file:///Source/ConveyorSystem_SFC.xml#Velocity</Value>
    <Constraint Name="Safety Rule Motor">
      <OrdinalScaledType>
        <RequiredMaxValue>60</RequiredMaxValue>
      </OrdinalScaledType>
        ...
```

Listing 5: Specification of the safety rule, defining a maximum speed of the motor

Security Rule. Stuxnet demonstrated how susceptible ICSs are to data manipulation. Manipulated sensors or controllers reporting incorrect values can disrupt the entire process and may lead to unsafe states. Approaches to mitigate this risk include the use of independent sensors, PLC integrity checks, and consistency checks throughout the control network. In this example, we want to show how a consistency check can be modeled in AML, so that the rule can be evaluated during operation by CPS Twinning.

Recall that the HMI displays the motor status (started/stopped), according to the operator's action. An attacker who gains access to the network could launch a man-in-the-middle attack (MITM) in order to manipulate system states. For example, a successful spoofing attack could cause the HMI to display an incorrect status or trick the PLC into starting the motor, without the HMI actually sending the command.

To detect inconsistent states of the HMI and the PLC, we introduce a `VariableLinkConstraint` that expects equality between the variables of the two devices (cf. Listing 6). In addition, we could also connect an independent physical sensor (outside the process loop) to the CPS Twinning framework, and take its measurements for comparison.

```
<InternalElement Name="VelocityConstraint" ID="e0b...">
  <Attribute Name="operator" AttributeDataType="xs:string">
    <Value>equals</Value>
  </Attribute>
  <InternalLink Name="VelocityConstraint PLC1 - HMI1"
  ↪  RefPartnerSideA="{133...}:Velocity"
  ↪  RefPartnerSideB="{068...}:Velocity" />
```

```
    <RoleRequirements
      ↪  RefBaseRoleClassPath="/VariableLinkConstraintRoleClass" />
</InternalElement>
```

Listing 6: Specification of the security rule, defining that the values of both variables must be equal

4.2 Virtual Environment Generation

This section describes step-by-step how the prototype generates the virtual environment based on knowledge sources.

The creation process is initiated by executing the custom command `twinning` via Mininet's command-line interface (CLI). This particular command expects the path to the AML artifact as an argument for the parser. After invoking the AML parser, the topology is generated and the extracted rules are provided to the security and safety analysis module.

As already mentioned in Section 4, we used the API of Mininet to implement the basic network layer of the framework. Each implemented digital twin class that requires network capabilities inherits from Mininet's `Host` class, allowing a seamless integration into Mininet. As a result, the predefined Mininet commands (e.g., `nodes` to list all hosts) can also be used for the generated digital twins. Owing to Mininet's virtualization approach, each digital twin of the network topology is a process that is running in its own network namespace [16].

When instantiating a PLC node, the framework spawns another process in the twin's namespace to run the program that emulates the internal behavior of the PLC. This particular Python program starts the build process of the PLC code and subsequently manages its execution. Depending on the specification, it is also able to emulate a Modbus slave. Furthermore, the PLC emulator starts a listener to exchange data between the Mininet CLI and the process that emulates the PLC. In this way, users are able to view metadata about the PLC program (e.g., variable types) and get/set a tag's value. Listing 7 provides an overview of the supported commands.

Compared to Mininet hosts, the HMI digital twins also support the `show_tags`, `get_tag` and `set_tag` commands. If one of the commands to get/set a tag are executed, the framework spawns a Modbus master to communicate with the PLC.

In the case of virtually cloning physical devices that require no network capability (e.g., motors), instances of the corresponding class (e.g., `Motor`) are created, but not added to the virtualized network. To simulate the internal behavior of these devices, the instances monitor specific tags of connected digital twins, such as a PLC.

Finally, the security and safety analysis module is initialized with the parsed rules. This component monitors specific variables of digital twins and issues alerts in case of a detected rule violation. In this first version of CPS Twinning, alerts are logged to a file.

```
mininet> twinning /home/user/ConveyorSystem.aml
mininet> nodes
available nodes are:
HMI1 PLC1 Switch1 c0
mininet> links
Switch1-eth1<->HMI1-eth0 (OK OK)
Switch1-eth2<->PLC1-eth0 (OK OK)
```

```
mininet> show_tags PLC1
Name        |Class       |Type
--------------------------------------
ENABLE      |var         |bool
PTO         |var         |bool
Q10         |out         |bool
Q00         |out         |bool
START       |mem         |bool
STOP        |mem         |bool
VELOCITY    |mem         |int
...
mininet> get_tag PLC1 START
False
mininet> set_tag PLC1 START True
mininet> get_tag PLC1 START
True
```

Listing 7: Output of commands that can be used for a PLC digital twin

4.3 Simulation & Results

At this point, we want to test the virtual environment generated from specification in the previous section. First, we execute simple actions in the physical as well as in the virtual environment. Thereafter, we compare the outcome of both runs to see how similar the results are. Second, we execute a MITM attack inside the virtual environment, to test the detection of violated security and safety rules, based on the examples given in Section 4.1, Safety Rule and Security Rule.

4.3.1 Environment Comparison. For the comparison, we trigger all user commands via the HMI in the physical environment and through variable changes (e.g., `set_tag HMI1 Start True`) on the digital twin respectively. The simple process steps are as follows: (1) Initially, the motor is off, waiting for the `HMI1` to send a start command. Hence, this is also the first action in the simulated run. (2) Next, we adapt the velocity of the conveyor belt to a value of 19 and 21 afterwards. (3) Finally, we stop the conveyor belt after a short period.

In order to observe the behavior and events, we captured the network traffic in the virtual and physical environment. The network capture from the physical environment contains 46 packets, whereas the capture from the virtual environment contains 50. We excluded unrelated traffic, such as DHCP and DNS, from the traffic dump of the physical environment. The IP addresses are identical in both environments, as defined in our AML artifact. Furthermore, the process steps match in both environments, including our manually triggered actions, as well as the responses sent by the Modbus slave. Going into more detail, we compare the TCP stream of the velocity change inside the physical network (cf. Listing 8) to the virtual environment (cf. Listing 9).

Line 5 of Listing 8 and Listing 9 reflects the actual Modbus request to write multiple registers, setting register number 2 to the value 19. The Modbus payload is identical in both environments, but there is a difference in the timestamps, due to the fact that we triggered the commands manually. We can also see that the response times inside the virtual environment are lower than in our physical setup. The network stack of our virtual environment and the physical devices is not identical, hence, there are slight differences in the TCP traffic.

Our digital twin for the HMI responds with an ACK packet (Listing 9, line 6) to a Modbus query before sending the Modbus response. The HMI of the physical environment only sends the Modbus response (Listing 8, line 6). Similar, the Siemens PLC sends a RST packet after FIN to close the connection (Listing 8, line 10).

This short example demonstrated, that the automatically generated digital twins behave according to the specification. The digital twins, including their network configuration match the physical environment, viz. the physical environment matches the specification. With focus on the control logic, we did not detect deviations in the control flow as the PLC's digital twin runs the same code as its physical counterpart. However, there are minor differences in the network traffic, due to varying implementations of the network stack.

```
|Time      | 192.168.0.2 <> 192.168.0.1
|12.767734 | --> | TCP: 49796 - 502 [SYN] Seq=0 MSS=1460
|12.768956 | <-- | TCP: 502 - 49796 [SYN, ACK] Seq=0 Ack=1 MSS=1460
|12.768987 | --> | TCP: 49796 - 502 [ACK] Seq=1 Ack=1
|12.769628 | --> | Modbus/TCP: Query: Func: 16 (Register 2: 19)
|12.787200 | <-- | Modbus/TCP: Response: Func: 16
|12.787291 | --> | TCP: 49796 - 502 [ACK] Seq=16 Ack=13
|12.787620 | --> | TCP: 49796 - 502 [FIN, ACK] Seq=16 Ack=13
|12.791942 | <-- | TCP: 502 - 49796 [ACK] Seq=13 Ack=17
|12.802970 | <-- | TCP: 502 - 49796 [RST, ACK] Seq=13 Ack=17
```

Listing 8: Excerpt of the Modbus TCP/IP network traffic in the physical environment

```
|Time     | 192.168.0.2  <> 192.168.0.1
|6.048040 | --> | TCP: 52606 - 502 [SYN] Seq=0 MSS=1460
|6.050265 | <-- | TCP: 502 - 52606 [SYN, ACK] Seq=0 Ack=1 MSS=1460
|6.050279 | --> | TCP: 52606 - 502 [ACK] Seq=1 Ack=1
|6.053310 | --> | Modbus/TCP: Query: Func: 16 (Register 2: 19)
|6.053515 | <-- | TCP: 502 - 52606 [ACK] Seq=1 Ack=16
|6.058982 | <-- | Modbus/TCP: Response: Func: 16
|6.058995 | --> | TCP: 52606 - 502 [ACK] Seq=16 Ack=13
|6.061786 | --> | TCP: 52606 - 502 [FIN, ACK] Seq=16 Ack=13
|6.068576 | <-- | TCP: 502 - 52606 [FIN, ACK] Seq=13 Ack=17
|6.068586 | --> | TCP: 52606 - 502 [ACK] Seq=17 Ack=14
```

Listing 9: Excerpt of the Modbus TCP/IP network traffic in the virtual environment

4.3.2 Detecting Rule Violations. We now evaluate the effectiveness of the security and safety analysis module with regards to detecting violations of the specified rules. To validate our approach, we launched an ARP spoofing attack in the simulated *virtual environment* to position the attacker between the HMI and the PLC (MITM). When the Modbus master (HMI) sends a command to the Modbus slave (PLC), the attacker intercepts the traffic and manipulates the request. In our attack scenario, the operator attempts to set the velocity of the conveyor to the value of 20 by using the HMI. The Modbus master then sends a request (FC: 16) to write the new velocity value to the respective register of the slave. However, the attacker intercepts the traffic between the two parties and modifies the packet's payload in order to set the velocity to 100. Listing 10 depicts an excerpt of the intercepted network traffic.

```
|Time     |  Flow  | Info
|0.000000 | A -> P | ARP: 192.168.0.2 is at A
|0.000387 | A -> H | ARP: 192.168.0.1 is at A
...
|29.44686 | H -> A | TCP: 34976 - 502 [SYN]
|29.44689 | A -> P | TCP Out-Of-Order 34976 - 502 [SYN]
...
|29.45431 | H -> A | Modbus/TCP: Query: Func: 16 (Register 2: 20)
|29.45432 | A -> P | TCP Retr.: Query: Func: 16 (Register 2: 100)
...
|29.47340 | P -> A | Modbus/TCP: Response: Func: 16
|29.47341 | A -> H | TCP Retr.: Response: Func: 16
...
```

Listing 10: Excerpt of the intercepted network traffic between the PLC (P) and the HMI (H) from the attacker's (A) point of view

Recall that according to the specification, the velocity must not exceed the threshold value of 60 (cf. Listing 5) and the HMI velocity tag value must be equal to the corresponding tag of the PLC (cf. Listing 6). As a result, if the attacker succeeds in tampering the velocity value, the security and safety analysis module of CPS Twinning should raise two alarms, since both constraints are violated.

```
INFO:root:'Velocity' value changed 0 -> 20 in device 'HMI1'.
INFO:root:'VELOCITY' value changed 0 -> 100 in device 'PLC1'.
WARNING:root:ALERT! 'PLC1' tag [Velocity=100] exceeds max value of
↪  60.
WARNING:root:ALERT! 'HMI1' tag [Velocity=20] does not equal 'PLC1'
↪  tag [Velocity=100].
```

Listing 11: Logging output of CPS Twinning

As can be seen in Listing 11, the framework tracks state changes and yields a warning if a violation of a specified rule occurs. The logged statements could signal plant operators or security professionals that a system is in an abnormal condition and requires investigation.

As already stated, the MITM attack has been carried out in the virtual environment, meaning that we tested the detection of rule violations when running CPS Twinning in simulation mode. However, the result in replication mode would be identical, provided that all states of the physical production system are replicated. Furthermore, the absence of the attacker's MAC address in the specification would result in a mismatch between the physical and virtual environment; hence, exposing the attack.

It should also be mentioned that the presented attack scenario can be mitigated by specifying network rules (e.g., an IP to MAC address mapping already exists in the specification). However, we chose the aforementioned rules intentionally to demonstrate how the state of digital twins can be audited.

5 RELATED WORK

Previous research can be divided into (i) the simulation of CPSs, (ii) process-aware intrusion detection systems, and (iii) the modeling of digital twins.

Several studies have attempted to simulate ICSs in order to assess their level of security [2, 7, 9, 36]. Similar to our approach,

MiniCPS [2] is also based on Mininet to emulate the network layer for simulated CPSs. However, there are several fundamental distinctions between MiniCPS and CPS Twinning. First of all, MiniCPS aims to provide researchers a virtual environment to test CPSs network setups, explore different attack scenarios, and evaluate countermeasures. While CPS Twinning also supports all of the aforementioned use cases, the focus of our work lies on generating digital twins in order to virtually replicate the physical process as close as possible. Furthermore, we propose a framework with security modules beyond network analysis, and present a first prototype, including security and safety rules in simulation mode. Second, CPS Twinning has been designed to generate the virtual environment from specification. As a result, the framework allows a seamless integration into the system engineering process and can further be maintained with ease by updating the specification. Third, in [2], the authors demonstrate how a MITM attack can be prevented by implementing an ARP spoofing detection algorithm in a custom software-defined networking (SDN) controller. In contrast, we show in Section 4.3.2 how such an attack can be detected by monitoring the states of digital twins. Finally, it is also worth highlighting that there are substantial differences in the implementation of both frameworks. While in MiniCPS the emulation of a PLC requires users to port the PLC code to Python, our presented prototype supports direct PLC code execution. As far as the implementation of industrial protocols is concerned, MiniCPS supports Modbus TCP/IP and EtherNet/IP, whereas CPS Twinning is currently limited to Modbus TCP/IP.

Dong et al. [9] investigate how SDN can be leveraged to increase the resilience of smart grids and which security risks are involved with this approach. In their work, they describe several use cases where SDN can be applied as a countermeasure against attacks, such as a distributed denial-of-service (DDoS) or packet delay attack. Further, the authors developed a smart gird test bed to validate their proposed approach. This particular test bed is based on Mininet and PowerWorld[3], a power system simulator. Contrary to [9], the paper at hand does neither focus specifically on the resilience of CPSs, nor does the prototype presented in Section 4 support the generation of digital twins that would provide a complete virtual replica of a smart grid. However, as the work by Dong et al. [9] demonstrates, Mininet is also suitable to be used as part of a SDN-based smart grid simulator; thus, enhancing the prototype to expand the concept of digital twins also to smart grids may constitute a valuable extension.

Works such as [25] and [8] propose a process-aware intrusion detection technique by utilizing the knowledge of engineers who developed the system. In [25], the authors present a network-based IDS framework for SCADA systems that is able to take process variable values into account (e.g., temperature). In their work, they propose a language that allows engineers to express normal values of process variables. While defining these constraints with their proposed language seems trivial, this requirement can be considered as an activity that does not naturally fit into the engineering workflow and may entail additional effort. In contrast, we propose the implicit or explicit definition of these constraints in engineering data formats, such as AML. While efforts have been made to automate the task of creating the rules that specify a system's correct behavior [5], manual work obviously remains if the documentation is missing or other sources provide incomplete information. As a result, we argue that security knowledge about a CPS should be defined already in early phases of the system engineering process and then maintained consistently throughout the system's lifecycle.

Other studies related to our work concentrate on modeling [1, 28] and implementing digital twins [32, 35]. It is worth mentioning that Schroeder et al. [28] also use AML to model digital twins. However, their work focuses on the data exchange between the digital twin and other systems.

6 CONCLUSIONS

In this paper, we have presented CPS Twinning, a framework to generate and execute digital twins, and furthermore, we have demonstrated the process in an industrial scenario. It covers a novel approach to automatically generate virtual environments completely from specification. As a result, this approach is reusable, consistent and guarantees a complete reflection of the specification.

With this approach, organizations that already use specification languages in their engineering process can build a digital environment without any or little additional effort. Tooling support and device templates could help in reaching a consistent specification in general. It further opens the possibility to recreate and experiment with an identical CPS environment, just by exchanging its specification.

From a security perspective, an identical (in terms of the system's specification), simulated environment can be freely explored and tested by security experts, without endangering the production environment. However, the security possibilities go beyond that. On top of the virtualization engine, security modules support experts in protecting the environment. In a proof of concept, we have demonstrated how the framework detects a MITM attack, targeting the manipulation of a motor's speed.

Current limitations of the prototype include a limited support of data types in PLC code (data types other than Boolean and integer are not available) and Modbus function codes. There are also some steps, which have been triggered manually, but could be integrated in a complete automation pipeline. For example, the translation of vendor-specific function blocks.

While the paper at hand has introduced the idea of the framework and a prototype has demonstrated the complete process, from specification to detection, other features and modules have not been implemented yet.

As far as the prototype of the CPS Twinning framework is concerned, the implementation thereof raised several issues that are worth pointing out. First, the underlying idea of this work relies on the assumption that a specification of CPSs exists in a level of detail that allows the generation of the virtual environment. Ideally, artifacts are maintained throughout the lifecycle of CPSs and are already suitable to be used as an input for CPS Twinning, which in practice may not be the case. If artifacts are missing or incomplete, users are required to manually create the specification before using the framework. Since this process is error-prone and involves manual work, integrating a specification mining approach into CPS Twinning would provide valuable support. Second, achieving an

[3] https://www.powerworld.com/

implementation of digital twins that provides an identical replication of their physical counterparts is challenging. For example, differences in the network stack implementation may manifest in the network traffic itself (cf. Section 4.3.1) and the timing of actions, causing digital twins to be out of sync with their physical counterparts. Third, implementing a framework that is capable of closely mirroring CPSs is a work-intensive task, even if some components that facilitate the development (e.g., Mininet) are already publicly available.

As future work, we want to focus on the replication mode of the framework, i.e., mirroring the state of physical systems to their corresponding digital twins. To validate our approach, we plan to launch a MITM attack in the real environment for the purpose of detecting deviations in the behavior of digital twins to spot attacks as early as possible.

Moreover, we aim to address the issue of non-existent or incomplete artifacts by mining specifications. Passive sniffing, device fingerprinting or extracting data from system logs and documentation may be possible approaches that would help users to get started with CPS Twinning if no such specification exists initially.

Further topics include, e.g., a modeling language for security and safety rules, exploring device templates with security rules, behavior learning and analysis in combination with anomaly detection, a client to visualize results and to manage the framework, as well as supporting additional industrial protocols.

ACKNOWLEDGMENTS

The financial support by the Austrian Federal Ministry for Digital, Business and Enterprise and the National Foundation for Research, Technology and Development, and COMET K1, FFG - Austrian Research Promotion Agency is gratefully acknowledged. Furthermore, this work was supported by the Austrian Science Fund (FWF) and netidee SCIENCE under grant P30437-N31.

REFERENCES

[1] K. M. Alam and A. El Saddik. 2017. C2PS: A Digital Twin Architecture Reference Model for the Cloud-Based Cyber-Physical Systems. *IEEE Access* 5 (2017), 2050–2062. https://doi.org/10.1109/ACCESS.2017.2657006

[2] Daniele Antonioli and Nils Ole Tippenhauer. 2015. MiniCPS: A Toolkit for Security Research on CPS Networks. In *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and PrivaCy (CPS-SPC '15)*. ACM, NY, 91–100. https://doi.org/10.1145/2808705.2808715

[3] AutomationML. 2014. *Whitepaper: Communication.* Technical Report V_1.0.0. AutomationML consortium.

[4] Radhakisan Baheti and Helen Gill. 2011. Cyber-physical systems. *The impact of control technology* 12 (2011), 161–166.

[5] M. Caselli, Emmanuele Zambon, Johanna Amann, Robin Sommer, and Frank Kargl. 2016. *Specification Mining for Intrusion Detection in Networked Control Systems.* USENIX Association, 791–806.

[6] Marco Caselli, Emmanuele Zambon, and Frank Kargl. 2015. Sequence-aware Intrusion Detection in Industrial Control Systems. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security (CPSS '15)*. ACM, NY, 13–24. https://doi.org/10.1145/2732198.2732200

[7] Rohan Chabukswar, Bruno Sinopoli, Gabor Karsai, Annarita Giani, Himanshu Neema, and Andrew Davis. 2010. Simulation of Network Attacks on SCADA Systems. In *First Workshop on Secure Control Systems, Cyber Physical Systems Week 2010.* http://www.truststc.org/pubs/693.html

[8] Justyna J. Chromik, Anne Remke, and Boudewijn R. Haverkort. 2016. *What's under the hood? Improving SCADA security with process awareness.* IEEE. https://doi.org/10.1109/CPSRSG.2016.7684100 eemcs-eprint-27160.

[9] Xinshu Dong, Hui Lin, Rui Tan, Ravishankar K. Iyer, and Zbigniew Kalbarczyk. 2015. Software-Defined Networking for Smart Grid Resilience: Opportunities and Challenges. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security (CPSS '15)*. ACM, New York, NY, USA, 61–68. https://doi.org/10.1145/2732198.2732203

[10] R. Drath, A. Luder, J. Peschke, and L. Hundt. 2008. AutomationML - the glue for seamless automation engineering. In *2008 IEEE International Conference on Emerging Technologies and Factory Automation*. 616–623. https://doi.org/10.1109/ETFA.2008.4638461

[11] David Duggan, Michael Berg, John Dillinger, and Jason Stamp. 2005. Penetration testing of industrial control systems. *Sandia National Laboratories* (2005).

[12] B. Genge, C. Siaterlis, and G. Karopoulos. 2013. Data fusion-base anomaly detection in networked critical infrastructures. In *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*. 1–8. https://doi.org/10.1109/DSNW.2013.6615505

[13] IEC. 2003. 61131-3: Programmable controllers – Part 3: Programming languages. *International Standard, Second Edition, International Electrotechnical Commission, Geneva* 1 (2003).

[14] IEC. 2009. 62443: Industrial communication networks – Network and system security. *International Standard, First Edition, International Electrotechnical Commission, Geneva* 1 (2009).

[15] Mikel Iturbe, Iñaki Garitano, Urko Zurutuza, and Roberto Uribeetxeberria. 2017. Towards Large-Scale, Heterogeneous Anomaly Detection Systems in Industrial Networks: A Survey of Current Trends. *Security and Communication Networks* (2017).

[16] Bob Lantz, Brandon Heller, and Nick McKeown. 2010. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-IX)*. ACM, NY, Article 19, 6 pages. https://doi.org/10.1145/1868447.1868466

[17] Jay Lee, Edzel Lapira, Behrad Bagheri, and Hung an Kao. 2013. Recent advances and trends in predictive manufacturing systems in big data environment. *Manufacturing Letters* 1, 1 (2013), 38 – 41. https://doi.org/10.1016/j.mfglet.2013.09.005

[18] Mark Luchs and Christian Doerr. 2017. *Last Line of Defense: A Novel IDS Approach Against Advanced Threats in Industrial Control Systems.* Springer, Cham, 141–160. https://doi.org/10.1007/978-3-319-60876-1_7

[19] Arndt Lüder, Nicole Schmidt, Kristofer Hell, Hannes Röpke, and Jacek Zawisza. 2017. *Fundamentals of Artifact Reuse in CPPS.* Springer, Cham, 113–138. https://doi.org/10.1007/978-3-319-56345-9_5

[20] Arndt Lüder, Nicole Schmidt, Kristofer Hell, Hannes Röpke, and Jacek Zawisza. 2017. *Identification of Artifacts in Life Cycle Phases of CPPS.* Springer, Cham, 139–167. https://doi.org/10.1007/978-3-319-56345-9_6

[21] S. McLaughlin, C. Konstantinou, X. Wang, L. Davi, A. R. Sadeghi, M. Maniatakos, and R. Karri. 2016. The Cybersecurity Landscape in Industrial Control Systems. *Proc. IEEE* 104, 5 (May 2016), 1039–1057. https://doi.org/10.1109/JPROC.2015.2512235

[22] Bill Miller and Dale Rowe. 2012. A Survey of SCADA and Critical Infrastructure Incidents. In *Proceedings of the 1st Annual Conference on Research in Information Technology (RIIT '12)*. ACM, NY, 51–56. https://doi.org/10.1145/2380790.2380805

[23] Robert Mitchell and Ing-Ray Chen. 2014. A Survey of Intrusion Detection Techniques for Cyber-physical Systems. *ACM Comput. Surv.* 46, 4, Article 55 (March 2014), 29 pages. https://doi.org/10.1145/2542049

[24] Thomas Morris and Wei Gao. 2014. *Industrial Control System Traffic Data Sets for Intrusion Detection Research.* Springer, Berlin, 65–78. https://doi.org/10.1007/978-3-662-45355-1_5

[25] Jeyasingam Nivethan and Mauricio Papa. 2016. A SCADA Intrusion Detection Framework That Incorporates Process Semantics. In *Proceedings of the 11th Annual Cyber and Information Security Research Conference (CISRC '16)*. ACM, NY, Article 6, 5 pages. https://doi.org/10.1145/2897795.2897814

[26] Roland Rosen, Georg von Wichert, George Lo, and Kurt D. Bettenhausen. 2015. About The Importance of Autonomy and Digital Twins for the Future of Manufacturing. *IFAC-PapersOnLine* 48, 3 (2015), 567 – 572. https://doi.org/10.1016/j.ifacol.2015.06.141

[27] Juan E. Rubio, Cristina Alcaraz, Rodrigo Roman, and Javier Lopez. 2017. Analysis of Intrusion Detection Systems in Industrial Ecosystems. In *14th International Conference on Security and Cryptography (SECRYPT 2017)*.

[28] Greyce N. Schroeder, Charles Steinmetz, Carlos E. Pereira, and Danubia B. Espindola. 2016. Digital Twin Data Modeling with AutomationML and a Communication Methodology for Data Exchange. *IFAC-PapersOnLine* 49, 30 (2016), 12 – 17. https://doi.org/10.1016/j.ifacol.2016.11.115

[29] Mike Shafto, Mike Conroy, Rich Doyle, Ed Glaessgen, Chris Kemp, Jacqueline LeMoigne, and Lui Wang. 2010. Draft modeling, simulation, information technology & processing roadmap. *Technology Area* 11 (2010).

[30] Jill Slay and Michael Miller. 2008. Lessons Learned from the Maroochy Water Breach. In *Critical Infrastructure Protection*, Eric Goetz and Sujeet Shenoi (Eds.). Springer, Boston, 73–82.

[31] Keith Stouffer, Victoria Pillitteri, Suzanne Lightman, Marshall Abrams, and Adam Hahn. 2015. Guide to Industrial Control Systems (ICS) Security. *NIST special publication* 800, 82r2 (Jun 2015). https://doi.org/10.6028/nist.sp.800-82r2

[32] Thomas H.-J. Uhlemann, Christian Lehmann, and Rolf Steinhilper. 2017. The Digital Twin: Realizing the Cyber-Physical Production System for Industry 4.0. *Procedia CIRP* 61, Supplement C (2017), 335 – 340. https://doi.org/10.1016/j.

procir.2016.11.152

[33] Prem Uppuluri and R. Sekar. 2001. *Experiences with Specification-Based Intrusion Detection.* Springer Berlin Heidelberg, Berlin, Heidelberg, 172–189. https://doi.org/10.1007/3-540-45474-8_11

[34] David I. Urbina, Jairo Giraldo, Alvaro A Cardenas, Junia Valente, Mustafa Faisal, Nils Ole Tippenhauer, Justin Ruths, Richard Candell, and Henrik Sandberg. 2016. *Survey and new directions for physics-based attack detection in control systems.* Technical Report. NIST. https://doi.org/10.6028/nist.gcr.16-010

[35] Ján Vachálek, Lukás Bartalskỳ, Oliver Rovnỳ, Dana Šišmišová, Martin Morháč, and Milan Lokšík. 2017. The digital twin of an industrial production line within the industry 4.0 concept. In *2017 21st International Conference on Process Control (PC)*. 258–262. https://doi.org/10.1109/PC.2017.7976223

[36] C. Wang, L. Fang, and Y. Dai. 2010. A Simulation Environment for SCADA Security Analysis and Assessment. In *2010 International Conference on Measuring Technology and Mechatronics Automation*, Vol. 1. 342–347. https://doi.org/10.1109/ICMTMA.2010.603