Towards Formal Security Analysis of Industrial Control Systems

Marco Rocchetto
Security and Trust of Software Systems
University of Luxembourg
marco.rocchetto@uni.lu

Nils Ole Tippenhauer
Information Systems Technology and Design
Singapore University of Technology and Design
nils_tippenhauer@sutd.edu.sg

ABSTRACT

We discuss the use of formal modeling to discover potential attacks on Cyber-Physical systems, in particular Industrial Control Systems. We propose a general approach to achieve that goal considering physical-layer interactions, time and state discretization of the physical process and logic, and the use of suitable attacker profiles. We then apply the approach to model a real-world water treatment testbed using ASLan++ and analyze the resulting transition system using CL-AtSe, identifying four attack classes.

To show that the attacks identified by our formal assessment represent valid attacks, we compare them against practical attacks on the same system found independently by six teams from industry and academia. We find that 7 out of the 8 practical attacks were also identified by our formal assessment. We discuss limitations resulting from our chosen level of abstraction, and a number of modeling shortcuts to reduce the runtime of the analysis.

1. INTRODUCTION

A number of real-world examples (e.g., the ones reported in [13]) have shown the paramount importance of improving security in Cyber-Physical System (CPS). The exploitation of security flaws in CPS can potentially cause damage to nations (e.g., the attack on Maroochy Shire Council's sewage control system in Queensland, Australia, in 2000 [12]) or even the entire world (e.g., Stuxnet [42]). A CPS contains both physical and virtual components, therefore a number of new attacks that combine the two aspects can be exploited by an attacker [12]. In addition, the attacker can take advantage of the physical layer interactions with the system and then a number of new attacker properties must be taken into account (e.g., the distance between the attacker and the system) during the analysis of the security of CPS [31, 39]. Therefore, the security assessment of CPS is a challenging task, mainly due to the intrinsic *complexity* and *heterogeneity* of such systems. On the one hand, this complexity makes it difficult to efficiently apply automated or formal techniques for the security analysis of CPS [18, 24, 30]. On the other hand, the practical analysis of the security in CPS might affect the availability of the system and the exploitation may even result in expensive damage to the CPS.

We argue that a formal design-time security analysis of CPS would largely increase the security of CPS, and that a number of key elements for such an analysis have already been proposed in the security research community. In particular, formal methods for security analysis of information systems have recently been extended to cyber-physical systems in [23, 30]. In addition, generalized attacker and system models have been proposed in [9, 30, 39]. Together, those methods could allow to detect logical flaws in the system design, that may be difficult to detect by empirical security assessment [17]. One key component for such assessments is a framework that allows to model the system and attacks in sufficient detail. Using that framework, potential vulnerabilities of the system can be identified, and countermeasures can be validated. A standardized language for the definition of a CPS system model has not been defined. A definition of the system properties can be found in [12, 31] but it is far from be exhaustive. Several properties of the system has also been proposed as a result of the definition of a cyber-physical attacker, e.g., [32, 39]. However, to the best of our knowledge, there is still no general description of the properties that has to be considered when defining a system model of a CPS.

In this work, we bring together formal methods and attacker models proposed for cyber-physical systems, and use them to perform a comprehensive security assessment of a real-world water treatment testbed. We validate the results of our assessment by comparing it to attacks proposed in related work, and results from practical assessment [2].

While the definition of attacks, attackers, and system model should be a fairly standard procedure for the formal analysis of systems, the comparison between theoretical and practical results is often omitted in formal approaches. This analysis can be used to emphasize the importance of a design-time security analysis showing what it can prevent.

We summarize our main contribution as follows.

1. We propose an extensive system model of a real-world water treatment testbed (SWaT), that considers both network and physical layer interactions. Our model has been defined by following the description of the system defined during the design phase.

2. We categorize the possible attacks on SWaT, generalizing our results for water treatment plants.

3. We consider different attacker profiles and we show how our analysis results changes with respect to different profiles.

4. We provide a comparison between the attacks found by our formal analysis tool and the results of practical assessments performed by six teams from industry and academia.

Structure. In Section 2, we briefly introduce the background. We define how to model a CPS and attackers for CPS in Section 3. In Section 4, we introduce a practical use case, show how to model it, and present the results of our formal assessment. In Section 5, we compare our formal assessment results with findings from practical assessment on the modeled system. Related work is summarized in Section 6, and the paper is concluded in Section 7.

2. BACKGROUND

This work leverages related work from two publications: an attacker model framework proposed in [31], and an extension of the Dolev-Yao (DY) attacker model [16] with physical interactions proposed in [30]. We combine the two ideas and define how to formally validate a water treatment plant against different *types* of attackers (referred to as attacker profiles in [31]). In the remainder of this section, we provide more details on those two works, and introduce the formal specification language we use (ASLan++).

2.1 Attacker Model Framework

In [31], the authors define a taxonomy that they apply to review related work on attacker models for the security analysis of CPS. They use the results to define the main concepts behind the formal security analysis of such systems and then they propose a framework (implemented in a tool they called APE [28]) to encompass commonly used attacker models. We first present their terminology (relevant for our work) and then we use those definitions to informally describe their attacker model framework.

Terminology. A *System under attack* is an interacting or connected group (soft- and hardware, humans) forming a unified whole and serving a common purpose.

An *Attacker* is a group of human actors that collaborate to achieve a goal related to the system under attack.

An *Attacker Profile* describes templates or classes of attackers. These profiles are a generic description of the setting and intuition, and not an exhaustive listing of possible actions, motivations, or capabilities of the attacker.

An *Attacker Model* (together with compatible system models) ideally fully characterizes the possible interactions between the attacker and the system under attack. In particular, the model defines constraints for the attacker (e.g. finite computational resources, no access to shared keys)

A *System Model* characterizes relevant components of the system under attack, to a level of detail that allows to determine all possible interaction of the attacker with the system. We will not go into the details of the system model since our work focuses on the attacker. Therefore, we will not distinguish between system models that consider risks and threats linked to components of the system, and those that do not.

An *Attack Model* characterizes all potential interactions between the attacker and a specific configuration of the system under attack and the specification of the goal that the attacker wants to achieve with respect to the system under attack. One can consider an attack model as an instantiation of the attacker model on a specific scenario (i.e., system configuration).

Attacker Model Framework. The attacker model framework is defined as an hierarchy of *dimensions*. Each dimensions defines a relevant aspect for modeling the attacker for CPS, e.g., the distance between the attacker and the CPS. A *metric* is associated to each dimensions and, when the dimensions are instantiated, the framework produces an *attacker profile*. An attacker profile is then an instantiation of the set of dimensions defined by the attacker framework. We discuss the details of the dimensions and metrics in Section 3.2 when we define a mapping between attacker profile and attack model.
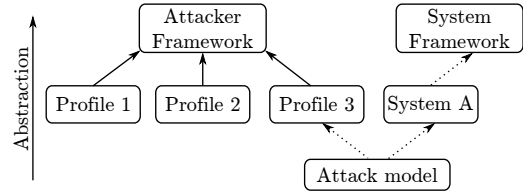

Figure 1: The extension of the Attacker Model Framework. The dotted lines are the ones formalized in this work.

As depicted in Figure 1, the attacker model framework can produce a variety of different attacker profiles. In [31], the authors define six attacker profile archetypes. Those archetypes capture common intuitions behind the related work they reviewed. Due to lack of space we cannot go into the details of those archetypes but we report their informal definitions.

A *Basic User* is someone who uses already established and potentially automated techniques to attack a system. This attacker has average access to hardware, software, and Internet connectivity, similar to what an individual can obtain through purchase with personal funds or by theft from an employer.

An *Insider*, which for example can be disgruntled employees or a social engineering victim. The employment position or the system privileges he owns (e.g., user, supervisor, administrator) are tightly related to the damage he can cause to the target.

Hacktivist. A portmanteau word which combines hacker and activist, as defined in [14]. This class of attackers uses their hacking abilities to promote a political agenda.

Terrorist, also known as cyber-terrorist. Is a politically motivated attacker who uses computers and information technology in general to cause severe disruption or widespread fear.

Cybercriminal, sometimes generally called black hat hacker or structured hacker. An attacker with extensive security knowledge and skills. This category of attackers takes advantage of known vulnerabilities, and potentially has the knowledge and intention of finding new zero-day vulnerabilities.

Nation-State, an attacker sponsored by a nation/state. Possibly belonging to (or that used to belong to) a state organization for carrying out offensive cyber operations.

2.2 Cyber-Physical Dolev-Yao

In [30], the authors define an extension of the DY attacker model, called Cyber-Physical Dolev-Yao (CPDY). Before going into the details of this extension we briefly introduce the DY attacker model.

The Dolev-Yao Model. The DY attacker model [16] is a de-facto standard for the formal analysis in the communication security domain. As such, the attacker model is commonly used for the identification of cyber-related attacks, mostly in security protocols (e.g., [4]). It also has been used for other systems, e.g., Web applications and Service-Oriented architectures as proposed in [3, 27]. Attacker models à la DY have been proposed [32, 36] to reason on CPS. In this work, we consider the standard DY [16] model of an active attacker who controls the network but cannot break cryptography.

Cyber-Physical extension. The extension (leveraging Horn logic rules) allows the standard DY to perform physical-layer interaction with the system. The authors motivate the extension by showing (on a small CPS model) that the standard DY model is not enough for modeling attackers in a

cyber-physical context. The CPDY extension is defined by two main sets of rules:

- Rules that represent physical-layer *interaction* in the system, e.g., to define that a valve can be manually operated.

- Rules that represent physical-layer *capabilities* of the attacker, e.g., to define that if an attacker has physical access to the CPS he can damage a component of the CPS.

We do not go into the details of the rules in this section since we will present our modeling technique in Section 3.

One obvious drawback of this extension is that the definition of the laws of physics related to the various processes in a CPS might not be straightforward. We discuss this in more details later in Section 3.

2.3 Formal Specification Language

We now define the terminology related to the system model specification that we use in the remainder of this paper. We use a terminology that resembles the one used in the ASLan++ language but we only informally describe the semantics of the terminology (the formal definition is reported in [7]). In our notation, a system model is formally represented by the following main concepts.

Entities, Agents and Roles. An *entity* in a system model represents a component of the system (i.e., of the CPS). Specifically, an entity describes both the *behavior* of the components and the communication with other entities. As an example, the behavior of a PLC is defined by its logic and it usually involves the interaction between the PLC and other entities in the CPS. An entity can also be considered as a set of components. In this case, the entities that have been grouped together are called sub-entities. An *agent* of the system model can play the *role* of a specific entity. There are two types of agent: honest and dishonest. When an *honest* agent plays the role of an entity, he behaves according to the behavior of the entity. A *dishonest* agent represents the attacker and then (depending on the attacker profile described later in this section) he can diverge from the honest behavior of the entity. For example, he can inject malicious payloads in the communication with other entities.

Communication and Channels. The network *communication* between different entities occur on a network *channel* can be signed, encrypted, or both (i.e., secure). An attacker model à la Dolev-Yao [16] is assumed, therefore, if the channel is unencrypted the attacker is able to read the messages exchanged through that channel. When the messages are not signed nor encrypted the attacker is able to modify the messages. In CPS, we also need to consider the analog communication and physical layer interactions between entities. For example, a sensor can communicate through an analog channel with a PLC. With the level of abstraction we have considered in this paper we do not take into account the implementation details of the different protocols involved in the analog or network communication. However, if a modeler considers an attacker who can read the analog communication then he has to explicitly model the message exchange between entities in the system model. Otherwise, he can group all the entities together and model the internal behavior without the need of specifying the internal communication.

Semantics and Transition System. Once a system model is specified in ASLan++, a translation (automatically performed by a tool) must be performed to generate a transition system (defined in the ASLan formal language [40]). The transition system can be used as input for the tools that performs the formal security analysis. The semantics of the ASLan++ model is then expressed as a transition
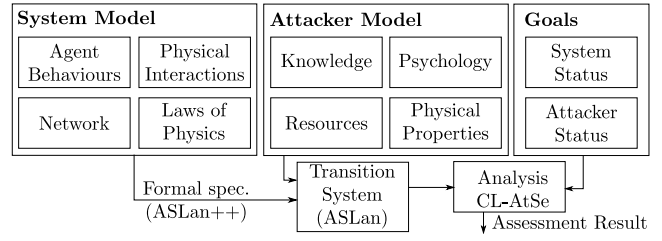

Figure 2: Graphical representation of the process of modeling and analyzing CPS. The dimensions of the attacker model are discussed in detail in [31].

system where the initial status of each entity is described by a predicate over all the variables used for the definition of the behavior of the entity itself. The transitions represents variation to the variables of the entities or on the knowledge of the attacker. The attacker knowledge is represented by a predicate as for any other entity. The final state (goal state) represents the goal that the attacker wants to achieve. We describe the goals later in Section 3 and Section 4.

3. TOOL-ASSISTED ASSESSMENT

The overall assessment process proposed is summarized in Figure 2. Following the structure of that figure, we now provide more details on system model, attacker model and goals. We conclude this section with a discussion of limitations of the proposed design choices.

3.1 System Model

As defined in Section 2.1, the system model is an abstraction of the system under attack and defines all the possible interactions of the attacker with the system. Therefore, the first step for the definition of a general system modeling technique for any CPS is the identification of the relevant aspects and components of the system. There are a number of different work in the literature for the definition of a system model for CPS, e.g., [15]. However, the problem has been often studies from an engineering point of view, with no (or little) attention for the security aspects [12].

We consider the following as main aspects of a CPS:

1. The components involved in the CPS and their behavior (e.g., defined by their logic)

2. The digital communication between the components

3. The physical interactions between components, partially defined by the related laws of physics

We now comment on the modeling of each in more detail.

3.1.1 Modeling of Components

Physical Status of the System. When modeling the status of a CPS, we have to consider the physical status of the components involved in the physical process. In order to consider the physical process we use a *database* that stores the status of all the components of the system. For example, the level of water in each tank or the level of chemicals inside the water. The term database in this context does not refer to standard DBMS (DataBase Management Systems), as ASLan++ (and other formal languages) cannot communicate with DBMS. Instead, it is possible to define a *status set* (using predicates) of tuples that uniquely defines the actual status of a property of a component in the CPS. In ASLan++ it is also possible to define the set as non-public, which means that the attacker cannot read or arbitrarily write the status of the database.

Physical Properties of Components. In a CPS, each component can have a number of physical properties which are stored in the status set. Those properties might also determine how an agent (e.g., the attacker) can interact with them. The properties of a component can take values from a continuous interval but, in our formalization, we have discretized these values into several symbolic constants. For example, when a sensor reads the level of the water inside a tank and reports them to a controller, we assume that the readings can only range in a (finite, ordered) set of symbolic constants.

As an example, we consider the valves in SWaT. There are two different type of valves: motorized valves and manual valves. The main difference is that a motorized valve can be operated remotely and manually, while a manual valve can only be manually operated. Therefore, an Insider attacker can potentially operate any type of valve while an attacker without physical access to the CPS cannot. In order to connect the physical properties of the system model to the interaction of the attacker model we have used Horn Clauses (HC) in our ASLan++ models, and other specific settings described in Section 4.2. An example of HC for the definition of a manual operation of a manual valve is the following.

$$systemStatus(Component, open) : -$$
$$attackerProp(distance, physicalAccess) \wedge$$
$$systemStatus(Component, close) \wedge$$
$$systemStatus(Component, manual)$$

where the status of the component *Component* can be changed in the *systemStatus* database if *Component* can be manually operated.

Control Logic of the Components. The logic of industrial control system components is commonly defined by the programming of the respective PLCs and the SCADA. Such logic is usually written in languages such as *ladder logic* following the IEC 61131-3 standard [22], and represents simple state machines and similar. It should therefore be possible to model the behavior of components faithfully in languages such as ASLan++. We provide an example of such a modeling in our use case in Section 4.

3.1.2 Digital Communications

The model of the network is not new to the formal methods community. There are, in fact, a number of works that model the network (e.g., as channels where different agents are communicating with each others) and protocols over the network (e.g., [11, 40]). However, the parties communicating in a CPS and the protocols used in the communication might be different from the ones considered in security protocols or in the standard client-server paradigm, see for example [8,39]. In our case, we use the DY model and then we focus on the payloads communicated by the various components more than on the structure of the messages.

3.1.3 Physical Interactions between Components

The definition of the laws of physics related to a CPS is not a trivial task. There are however preliminary results in the direction of the automated definition of such laws, e.g., [34]. The definition of a general tool for the extraction of the laws of physics from a CPS is out of the scope of this paper. However, we illustrate the main laws of physics we consider in our examples, and the relations between them, the system, and the attacker model.

The physical interactions between components expresses the constraints that needs to be fulfilled whenever a modification of the physical part of the system take place. For example, if two water tanks are connected by a pipe, when the water level of one of the tanks is decreasing, the water level of the other tank needs to increase. In our models

we capture linear physical interactions between components but nothing prevents a modeler to consider non-linear interactions (subject to the granularity of the discretization of values). As we show in the Section 4, a modeler can use sets or a database to store those information.

We propose to model the behavior of each component in terms of input/output messages, and modification of the physical status of the CPS (i.e., of the status set). The ASLan++ language (as other formal languages) provides an AnB (Alice and Bob) notation for the definition of the message exchange between entities. For example, a sensor S that sends a message M to a PLC P is defined in ASLan++ as: $S \rightarrow P : M$. This construct is then automatically translated into a transition system where the exchange of message is defined as modification of variables in state predicates (the details of the semantics of ASLan++ can be found in [6]). Furthermore, the communication of components and the PLCs is not direct but passes through sensors and actuators. For example, a tank itself typically does not communicate directly with the PLC. Instead, a sensor (e.g., Level Indicator Transmitter or LIT) reads the level of the water and communicate the readings to the PLC. The same applies to motorized valves or pumps—the status of those components is read by sensors and changed by actuators controlled by a PLC. In our use case, examples of such sensors/actuators are: Flow, Pressure, and Level Indicator Transmitters.

Neighboring components may affect the behavior of a component C. Thus, with higher number of affecting neighbors, the complexity of the definition of the behavior of C increases. In order to avoid state explosion problems, we have limited the expressiveness of the behavior of the components in our use case in Section 4 by considering only a subset of the related components.

3.2 Attacker Profile and Attack Model

Mapping an attacker profiles to an attack model can (in general) be achieved by mapping each instantiation of each dimension (metric) of an attacker profile to a constraint for the attack model. The attack model is, by definition, composed by an attacker profile and a system model. Therefore, a constraint in the attack model results in a constraint on the interaction between the attacker and the system.

In general, the attacker framework proposed in [31] can be applied to several security techniques, e.g. from risk analysis to model checking. However, in this work we focus on model-checking based techniques. In this context, a system model is often defined by a transition system (e.g., the semantics of the ASLan++ formal language [7]) which can potentially define a tree with an infinite number of paths. Constraints over the system will then be expressed as constraints on the exploration of this tree.

The structure of the attacker model is composed by three main independent dimensions: knowledge, resources, and psychology (see Figure 2). Each dimension is structured with an hierarchy of sub-dimensions. In this work, we omit the details of the dimensions and focus on the mapping between those dimensions and an attack model (see Section 4.2).

3.3 Security Goals

We define a security goal as a malicious state of the system or in terms of knowledge of the attacker model (i.e., confidentiality violation). Due to the heterogeneous nature of CPS, defining general security goals means defining goals in an abstract way (and with a high level of abstraction). The definition of such categorization is out of the scope of this work since we aim at defining goals that can concretely be applied to a CPS. Nevertheless, commonalities should exist: for example, all water treatment plants have similar logic to

control the process of physical components. Therefore, we can categorize the security goals for a water treatment plant as follows:

1. Over/Under-flow tanks
2. Arbitrarily change the status of a component (open/-close a motorized valve)
3. Alter chemical dosing of the water
4. Drop/Increase the pressure in a pipe
5. Drop/Increase the flow of the water in a pipe

In Section 4.3, we show our coverage and examples of attacks found on SWaT.

3.4 Limitations

3.4.1 Discretization of Time and States

In our work we focus on discrete-time analysis. This simplifies the analysis and permits us to use a wide range of security analysis tools (e.g., CL-AtSe [37], OFMC [10], SATMC [5]).

In our specific application scenario (water treatment plants), we found that the system's inertia leads to time constraints that are not as strict as in other CPS, e.g., power plants. As a result, the water level in a tank can be easily discretized by considering threshold values measured at discrete time steps. In particular, such discretization of values and times is also fundamentally used for the digital control of the system used by the PLCs. Our intuition is that the discretization chosen by the system engineers to guarantee stable operations of the system will also be suitable for our formal modeling of the system. This will also allow us to directly translate the control logic implemented in the PLCs to our system model.

We acknowledge that by using a discrete time we are limiting the expressiveness of the system. To validate that we are not missing important attacks in our analysis, we will compare our formal analysis results with the results of independent manual testing in Section 5, and the related work in Section 6.

3.4.2 Abstraction Level and Laws of Physics

The level of abstraction is always a concern when it comes to the definition of a system model. When considering CPS this is particularly delicate because the processes in a CPS can be modeled only along with the laws of physics related to that process. In addition, selected attacks in a CPS model might be possible only if specific laws of physics have been considered. As an example, to increase the pressure in a tank, the direct proportionality between temperature and pressure should be considered. An attacker with physical access to the CPS could increase the temperature of the tank and generate enough pressure to make the tank burst. A low level of abstraction (high level of details) would be then preferable in case of CPS. On the other hand, when considering model checking techniques, considering a high level of details might easily lead to non-termination issues.

4. USE CASE: SWaT

We have applied our technique to a real-world water treatment plant, SWaT (see Figure 3, with general description in [25]). SWaT is an operational ICS, built for the experimental research in the design of secure ICS. As depicted in Figure 4, the HMI, the SCADA, and the Historian are connected to the PLCs of the six physical processes of SWaT. Each physical process has a specific task, and is connected to one or more other physical processes (more details in [25]).

A number of manual valves, pressure meters, and flow meters are involved in SWaT, for the sake of simplicity we only discuss the main components involved in each process.

Figure 3: Our formal model is based on the water treatment plant SWaT.

A more detailed overview of the system is provided in Appendix 7. Each component (e.g., sensors, pumps) is associated to one specific PLC, but any PLC can request to communicate to any other component (through the PLC connected to that component). An access control table is stored in each PLC and determines whether or not another PLC (which is not directly connected to the component) is allowed to perform any request to the components it controls. In our analysis, we assume that any PLC can communicate with any component according to the implementation of SWaT.

Resulting model. The attack model that we have implemented in ASLan++ is around 1000 lines long and contains more than 50 entities (4 in P1, 12 in P2, 8 in P3, 8 in P4, 14 in P5, 7 in P6) where: 11 are pumps, 16 motorized valves, 6 PLC, 9 tanks, 9 AIT, 8 FIT, and 1 PIT. The ASLan++ specification (available at [29]) generates 269 ASLan rules (i.e., transition rules). Due to the complexity of the specification, CL-AtSe (the security analysis tool we have used) takes up to one day for the analysis of several security goals (assuming that the goal can be reached). In order to speed up the analysis, for each security goal, we have (manually) reduced the attack model considering only a subset of the processes that are related to the security goal. For example, when the goal involves the level of the water in the tank in the first process, we have considered only the first, second, and third process. In this way, we have reduced the state space and accelerated the analysis procedure to few seconds.

4.1 Modeling the SWaT Processes

We now define how to model each component involved in the SWaT process. We have used the ASLan++ formal language [40] for modeling SWaT. For the sake of readability and generality, we do not give the details of the ASLan++ model (which we made publicly available at [29]). Instead, we describe the general concepts behind the modeling.

Databases. As discussed in Section 3.1, we use a *non-public* database to store the physical state of the system.

Using the same concept we also create an *Historian* set which represents the Historian component. This can be used in LTL security goals in order to understand if the attacker modified the Historian. We discuss the goal later on in this section.

Network and Message Exchange. Different network protocols are involved in the SWaT testbed. We have, however, considered an abstraction of the payloads communicated and we have abstracted away all the details related to the specific protocols involved (e.g., Ethernet/IP [20] or the Common Industrial Protocol [33]). The formal analysis of the security protocols involved in SWaT is out of the scope of this work.
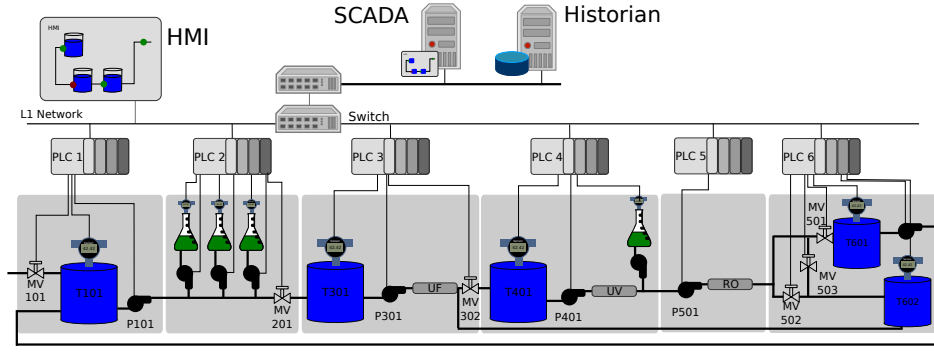
Figure 4: A simplified version of the SWaT process represented in our system model. In addition the components show, our formalized model contains details such as manual valves. The HMI, SCADA, UV and RO components are not fully represented in our system model for efficiency reasons.

Logic of the PLCs and SCADA. As proposed in Section 3.1, we defined the logic of the PLC by using the operating manual of SWaT. We note that while it should be theoretically possible to prove equivalence of the logic in SWaT and our model, our level of abstraction with respect to ASLan++ makes it difficult to provide a soundness proof between the two.

With respect to logic running in the SCADA system, we do not distinguish between the communication between the SCADA or the PLC and the various components because we abstract away the details of the protocols involved. Therefore, considering the SCADA entity would replicate the logic already defined in the PLCs. It might be interesting consider the SCADA system because it widens the attack surface, however, the complexity of the attacks remains the same since the logic of the PLC already expresses the control logic.

Example: Modeling of PLC1 and T101. As an example, we show and describe here (a part of) the model of the interactions between PCL1 and the tank T101 (where * is used as a wildcard).

PLC1 is constantly waiting for incoming communications by LIT101 (and LIT301, that we do not discuss for the sake of simplicity) and, based on the incoming values PLC1 decides how to proceed. Therefore, we have modeled the behavior of PLC1 as a switch-case over the incoming communication inside a while loop. The number of iterations of the while loop is set before the execution of CL-AtSe and variate depending on the goal. The number of iterations usually ranges between 2 to 10 since most of the attacks can be found with a low number of iterations and the analysis can take up to a day when 10 is chosen.

```
while(true){
 switch{

  %low (500mm)
  case(lit101 -> plc1: low):{
   Actor -> mv101: open;
  }

  %high (800mm)
  case(lit101 -> plc1: high):{
   plc1 -> mv101: close;
  }

  %highhigh (1200mm)
  case(lit101 -> plc1: highhigh):{
   systemStatus.remove((alarm,*));
   systemStatus.add((alarm,on));
  }
```

```
  %lowlow (250mm)
  case(lit101 -> plc1: lowlow):{
   plc1 -> mv101: open;
   plc1 -> p101: stop;

   systemStatus.remove((alarm,*));
   systemStatus.add((alarm,on));
  }
```

Example: Modeling of physical laws in T301.
In Section 3.1, we discussed the modeling of physical behavior of components based on physical laws. We now provide an example how this modeling was done in our use case. For the definition of the physical behavior of a component C we have considered the neighboring components that can affect the component C. As an example, consider the level of water inside tank T301 in process 3 (Figure 4) as a deterministic value. If the initial level is above an upper threshold, it will decrease if, e.g., the pump P101 is off, the motorized valve MV201 is closed, and the pump P301 is on.

```
if(
 (systemStatus((mv201,close)) |
   systemStatus((p101,off))) &
 systemStatus((p301,on))
){

 if(systemStatus((t301,Status))){
  systemStatus.remove((t301,Status));

  switch{
   case(Status = aboveThreshold){
     systemStatus.add((t301,highhigh));
     lit301 -> plc3: highhigh;}
   case(Status = highhigh){
     systemStatus.add((t301,high));
     lit301 -> plc3: high;}
  [...]
```

4.2 Modeling the Attacker

In this section, we describe our mapping between the different dimensions of the attacker profile and the attack model.

4.2.1 Attacker Profile

We consider two different attacker profiles: Insider and Cybercriminal. The definition of the two attacker profiles is given in Table 1 where the dimension instantiations reported reflect the descriptions in Section 2.1 and the definitions in [31]. In Table 1, we show a comparison between

Table 1: Comparison of attacker profiles in use case

Profile	System	Source Code	Distance	Strategy	Determination	Camouflage	Aim
Insider	●	●	●	●	●	●	physical
Cybercriminal	○	○	○	●	●	●	virtual

The metric associated do the dimensions is [●<○].

the two profiles, using [○<●] as a metric associated to the dimensions. For example, the symbol ●(and ○) applied to the distance dimension indicate having physical access to the CPS respectively (or not having access, respectively).

Insider. An Insider (e.g., a disgruntled employee) has physical access to the CPS. To model the high knowledge of the system we give to the attacker model the private keys of each entity. We have also performed experiments with the attacker playing the role of one or more entities in the system model. The main objective of this attacker profile is to attack the physical system.

Cybercriminal. This attacker profile is very well represented by the Dolev-Yao attacker model. The low knowledge of the system is reflected in no privileges, e.g., no private keys. The main objective of this attacker profile is to attack the virtual part of the system.

4.2.2 Dimensions

We have considered a subset of the dimensions defined in [31], since not all of the dimensions can be applied in our context (i.e., mapped into our system model).

System Knowledge. The initial knowledge of the attacker can be fine-tuned in formal languages. Specifically, in ASLan++ we define the attacker knowledge in terms of:

- Knowledge of messages exchanged.
- Encryption/decryption keys known by the attacker.
- Access to the status databases.
- Knowledge of the behavior of the entities of the CPS.

In security protocol analysis the knowledge of the messages exchanged between entities is a fairly standard constraint. When considering a CPS the messages might be exchanged on different network levels. For example, in SWaT, the messages exchanged between sensors and PLCs are in a different network with respect the ones exchanged between PLCs and the SCADA. In our system model, we group together different entities when they communicate on a network level non accessible to the attacker. We define an upper entity grouping together all those sub-entities and we hide from the attacker the internal communications. Therefore, only the communication between the upper-entities can be seen by the attacker. However, by giving/removing to the attacker knowledge encryption and decryption keys, it is possible to grant/restrict the attacker to a sub-part of the network of the CPS. Similarly, by giving access to the status database we can determine whether or not the attacker knows the current physical status of the system (e.g., to differentiate between Insider and Cybercriminal profiles).

Another feature of the ASLan++ language is that we can define that the attacker plays the role of one of the entities/components of the CPS. In this way we can consider scenarios in which the entity has been compromised by the attacker (assumption that we have considered during the S3 event).

Distance. Having physical access to the CPS is a great advantage of any Insider attacker. As we show later in Section 5, it is easier to perform an attack if the attacker has physical access to the CPS. In our system model, we can set the distance of the attacker by enabling/disabling his physical access to the CPS. By enabling physical access the attacker can manually operate the components of the SWaT testbed. For example, the attacker can manually open/close motorized and manual valves. While motorized valves can also be remotely operated, the manual valves are only part of the attack surface of the system model when in presence of an Insider attacker. We have modeled the distance constraints using Horn Clauses.

For each component of the SWaT system model we have also defined a constraint stating if it can be manually operated or not. Based on this and other properties of each component, a set of Horn Clauses determine the effect of manually operating a specific component, for example, the change of the state of that component in the status database.

Source Code Knowledge. In ASLan++ the attacker can play the role of any entity of the CPS, as if an attacker knew the source code running on that entity. In order, to allow the attacker to play the role of an entity, the modeler needs to take into account the feasibility of such constraint. For example, in SWaT, only when an attacker has access to the Studio5000 [35] software he can modify the logic of the PLC. Therefore, when the attacker is playing the role of the PLC, we implicitly assume that he has access to Studio5000.

Strategy and Determination. The strategy and determination dimensions express the type of analysis that the attacker wants to perform. This dimension is hard-coded in the formal analysis tools we have used (i.e., CL-AtSe [37]) and we could not change it. However, the analysis tool gives the user the possibility to tune the parameter used during the analysis. For example, the number of time each ASLan rule can be used or the number of iteration of each cycle (that is related to the number of actions each entity can perform). Leveraging those options we have been able to express different strategies and determinations.

Aim and Camouflage. We express the aim of the attacker as part of the security goals that the attacker wants to achieve. A security goal in ASLan++ can be expressed as a state or as an LTL (Linear Temporal Logic) property of the system model. Both type of security goals express an undesirable state of the system. For example, one can check that whenever a tank has been filled up then the water stops flowing inside the tank.

The stealthiness of an attack performed to achieve a security goal can be expressed by using the alarms of the system model. In SWaT, alarms can be triggered by the PLC and we use them to fine tune the stealthiness of the attacker profile considered in the analysis. For example, with the following LTL goal we express that we always want the system status database not to contain a level of the water (inside a tank) above a threshold without any alarm raised. The term *always* in the previous sentence refers to the LTL global operator (i.e., □). The operator expresses that the goal has to hold in every state reached by the security analysis tool.

$$\square \neg (systemStatus(tank, aboveThreshold) \wedge$$
$$systemStatus(alarm, off))$$

4.3 Impact of Attacker Profiles on SWaT

Security Goals. We have found a number of attacks to security goals with our attack model. CL-AtSe reports an attack trace as a message sequence chart for each attack found. We report an overview of the attacks found in Table 2. The complexity of such attacks (i.e., the time needed

Table 2: Categorization of attacks found using the formal assessment approach

Security Goal	Components affected/involved	Processes
Overflow Tank	all tanks	All
Empty Tank	all tanks	All
Changing Status Component	all pumps	Subset
Alter Chemical Dosing	P201, P202, P203, LIT201, LIT202, LIT203	Subset
Pressure Drop/Increase	PIT501	Subset
Flow Drop/Increase	all FIT	Subset

for the analysis of the security goal) mainly depend on the initial status of the system model and the attacker profile considered. To give an example, if the attack is to empty a tank and the system has just started filling up the water in the tank, a Cybercriminal waits until the water in the tank reaches a level such that the system starts emptying the tank to perform the attack. This is due to the fact that the attacker cannot manually manipulate the valves and then he cannot easily modify the status of the system. In Table 3 we have then reported the details of the attacks we have found with our technique. Specifically, we report the time of the security verification, the components involved in the attack, which attacker profile performed the attack, and the processed taken into account in the attack model. We now go into the details of the different attacks on security goals.

Tank Water Level. The first set of attacks we describe are the ones related to the level of the water inside a tank. We have considered over/under-flow of tanks with and without physical interactions from the attacker. Both the Cybercriminal and the Insider attacker can perform this type of attack on every tank, but we have considered different constraints related to the attacker profile (e.g., by requiring the stealthiness of the attacks). As an example, we consider the security constraint that the level of the water in the tank T101 (in the first process of SWaT, see Figure 4) needs to stay below a certain threshold as follows to avoid overflows:

$$\Box \neg(systemStatus(t101, aboveThreshold)$$

When we run our analysis against a cybercriminal attacker, CL-AtSe returns the following attack trace (all the traces of this section are slightly simplified for the sake of clarity).

```
plc1  -> lit101 : statusRequest
lit101 -> i(plc1): optimal
plc1  -> lit101 : statusRequest
lit101 -> i(plc1): high
plc1  -> lit101 : statusRequest
lit101 -> i(plc1): highhigh
plc1  -> lit101 : statusRequest
lit101 -> i(plc1): aboveThreshold
```

To perform this attack, the attacker waits while the water level is increasing in the tank dropping all the responses sent from the LIT101 to the PLC1 (i(plc) indicates that the attacker i is intercepting the message addressed to the PLC). The tank will then overflow because the PLC1 will not stop the inflowing of the water. We have performed our experiments with over- and underflow, and both with and without encryption of messages between PLCs and tanks/LITs but encryption does not prevent the Insider to achieve the goal.

For all the processes the attacker (either with Cybercriminal or Insider profile) can easily overflow the tank. However, the attacker with an Insider profile can directly manipulate one of the manual valve to achieve the goal. For example, we have tested our formal specification with respect to the security goal that if the MV101 is open, the level of the water in the tank T101 needs to be equal to a certain abstract

value *low* (the system fills up the tank if the level is low to prevent overflows).

$$\Box(systemStatus(mv101, open) \Rightarrow systemStatus(t101, low))$$

CL-AtSe reports the following attack trace, in which the `HC_open(Component)` is the HC (see Section 3.1.1) that allows an attacker that has physical access to the CPS to manually open a component, and `ST` are the numbered steps of the attack.

```
ST_1. systemStatus(p101,on)
ST_1. systemStatus(mv101,close)
ST_2. systemStatus(t101,low)
ST_3. HC_open[Component=mv101]
```

The goal can be trivially achieved by waiting that the level of the water increases up to a level above `low` due to the inflowing water. However, since CL-AtSe reports (only) the first attack trace found (not all the possible attack traces leading to the same attack) and given that the water was decreasing in the initial system status, CL-AtSe reports the expected attack trace. We note that in practice, this process usually reports a cyber attack (without any manual operation) even if a manual operation of a valve would be the most straightforward attack. We assume this is due to the implementation of CL-AtSe. In particular, CL-AtSe is an implementation of the DY attacker model and the manual operations are encoded in the attack model as Horn Clauses. That difference between manual and cyberattacks could explain the cyberattacks to be found first.

Changing Status of Components. The second set of attacks is related to changing the status of a component. Most of the components are affected, since if the attacker sends a malicious message to the component on behalf of a PLC, then that component will change its status accordingly. As an example, we have specified that in the initial state of the system all the pumps are off and no water is flowing in the system. We have then defined a security goal specifying that a security violation occurs when the flow meter FIT201 (placed after the pump P101) reports no flow and the pump P101 is on (the pump should not be on when there is no water to pump out of the tank T101).

$$\Box \neg(systemStatus(fit201, noFlow) \wedge$$
$$systemStatus(p101, on))$$

The communication in SWaT is not encrypted, thus an attacker can change the payloads of messages without a key. This leads both the Cybercriminal and the Insider to achieve this goal and CL-AtSe reports the following attack trace.

```
systemStatus(fit201,noFlow)
i(plc1) -> p101 : open
```

However, several components can only be operated manually (i.e., manual valves, drains). Thus, only the Insider is able to open drains and manual valves. These manual operations can be exploited in more sophisticated attacks (for example, combining manual and network operations).

Alter Chemical Dosing. With altering chemical dosing we mean that the attacker can effectively control the pumps and LITs of process 2 (i.e., chemical dosing process). When an attacker alters the behavior of a pump by, e.g., open it when it should be closed, the chemical process is influenced. Considering the concentrations of chemicals in the water would indeed be more accurate, but our current system model does not permit such details.

We have, however, tested our formal specification with respect to goals related to the level of the chemicals in the tanks in process 2. For example, we have defined that the chemicals in tank t201 cannot be below a threshold *lowlow* as follows:

$$\Box\neg(systemStatus(t201, lowlow))$$

The attack traces reported by CL-AtSe are similar to the ones already reported in this section.

Flow/Pressure Drop. The last set of attacks are related to the pressure or flow drops/increase. One way to achieve these goals, is to trick the PLCs in reading fake values of PITs or FITs. Another way is to prevent the water to flow through a PIT/FIT while a pump (after the PIT/FIT) is on. If we specify an initial state of the system where the pump P101 is on (lowering the water in T101) and we analyze the specification with respect to the following security goal:

$$\Box\neg(systemStatus(fit201, noFlow)\wedge$$
$$systemStatus(p101, on))$$

CL-AtSe returns the following attack trace where the attacker drops all the packets that should alert the PLC1 that the level of the water in the tank T101 is decreasing until the tank is empty. The attacker then opens the pump P101.

```
lit101  -> i(plc1)  : low
lit101  -> i(plc1)  : lowlow
lit101  -> i(plc1)  : belowThreshold
i(plc1) -> p101     : open
```

5. PRACTICAL ASSESSMENT

A formal security assessment is performed over a system model that abstracts away some of the details of the real system. In order to concretely exploit on the real system the attacks (or attack traces) found with a formal assessment, one has to concretize those attacks. In other words, one has to bridge the abstraction gap between the formalized system and the real system. In CPS this is a very complicated task due to the complexity of the system. In this section, we report a comparison between the results we have obtained with our design-time analysis and the attacks found with practical assessment.

5.1 Context of practical assessment

For the practical assessment, we invited six teams from industry and academia to practically attack the real system [2]. The teams could choose between three different attacker profiles to conduct the attack. In addition to the insider and cybercriminal, a *strong attacker* profile was available that we do not discuss further in this work. Each attacker profile had different capabilities, benefits and/or constraints following our profiles as outlined in Section 2. The main goals that the teams have been challenged to achieve are the following:

- *Physical Process Goals*, control over physical actuators and processes, i.e., valve status, tank fill level, pipe pressure, chemical dosing.

- *Sensor Data Goals*, demonstrate control over sensor readings at different components, i.e., PLC, local display of sensor, PLC, HMI, SCADA, Historian.

The results of the attacks using the Cybercriminal and Insider attacker profile are discussed next.

5.1.1 Cybercriminal

This attacker profile has no physical access to the system and then the team that chose this profile could not manually interact with the system (e.g., could not manually open a valve). For the same reason they did not have access to the L0 ring. In addition, no knowledge about the system and no specific tools for the modification of PLC logic have been provided. Assuming an high level of hacking skills (according to [31]) the team with this profile have been provided with a virtual machine with access to the L1 network. This has been done to simulate a compromised machine. The offensive capabilities of this attacker have not been limited, i.e., ARP spoofing, exploits, brute force attacks have been permitted.

5.1.2 Insider

By choosing this attacker profile, the team has physical access to the testbed and then to the tools that any employee could have access to (e.g., Studio5000 to read and modify the logic of PLC). They can then physically manipulate valves and change network topology. An Insider is assumed to have password of all main components of the system, for example, SCADA, and Historian. He also can operate directly with the HMI/SCADA system. However, the offensive knowledge of an Insider is limited (with respect to a Cybercriminal) and an Insider cannot perform ARP spoofing or brute force attacks and he cannot use exploits. The hacking tools allowed with this profile are limited too and, for example, network scanners are not allowed (but the attacker has previous knowledge of the system).

5.2 Attacks Found with Practical Assessment

During the practical assessment the teams have performed the different attacks following similar attack patterns. The attacks on *tank fill level* were performed with an Insider attacker profile by all teams. The teams have used the HMI (as an Insider) to set the system (pumps and valves) in such a way that the tank either over or under-flowed. The same technique has been exploited to perform attacks on *pumps*, *valves* and *chemical dosing*. For the chemical dosing attack, the attackers have kept open the pump responsible for the chemical dosing of the water longer than the default time. This should lead to an increasing of the chemicals in the water that ultimately spoils the membrane in the reverse osmosis process (process 5). However, precautionary procedures were set up before the attacks in order to not spoil the reverse osmosis process.

A number of DoS (Denial of Service) attacks have been performed to attack PLCs or HMI/SCADA (e.g., by using Ettercap [26]). One DoS was performed by physically disconnecting the Ethernet cable from one of the PLC. This prevented the PLC to communicate with the system.

5.3 Formal vs. Practical Assessment

In this section, we compare the attacks found with our formal technique (performed independently and before the practical assessment) with the practical assessment. A general overview is given in Table 4.

The attacks found from the practical assessment are very similar to the ones we have found with our formal analysis of SWaT. In 7 cases (on 8 total cases considered) the attacks are exactly the same (on different level of abstraction). Examples for those attacks are the ones on on pumps, valves,

Table 3: Attacks found with the formal assessment

Security Goal	Target Component	Time	Cybercrim.	Insider	Components Involved	Processes Considered
Overflow Tank	T101	21.164s	●	●	LIT101, LIT301	All
Overflow Tank (no alarm)	T101	5.094h	●	●	LIT101	All
Empty Tank	T101	1.868s	●	●	LIT101	P1
Drain Water	T301	16.960s		●	ManualValve301	P3
Overflow Tank (manual operations)	T101	19.736s		●	MV101	P1
Fake Flow	FIT201	35.773m	●	●	LIT101, PLC1, PLC2, P101	P1,P2
Empty Chemical Tanks	T201,T202,T203	0.872s	●	●	P201,P202,P203 LIT201,LIT202,LIT203	P2

chemical dosing, and tanks. However, due to the different level of abstraction of the two techniques, some of the details of the attack strategies are different. For example, during the practical assessment, when an attacker wanted to send spoofed values of one component to the PLC, he had to set first the PLC in manual mode. Manual mode enables the PLC to accept messages from the HMI and then the attacker can modify the payload of those messages. If manual mode is not enabled, the messages of the attacker are overwritten with the real message of the sensors [38]. Thus, an attacker has to physically access the SWaT testbed and manually switch the PLC to manual mode. In our formal analysis, we have not considered such practical details. Therefore, some of the attacks that can be performed with a Cybercriminal profile in our attack model, are not directly feasible in the real system. While in the real system, a Cybercriminal could perform the attack whenever the PLC is in manual mode, he would potentially need to wait until someone with physical access to the PLC changes the default mode to manual mode.

Summarizing, our formal analysis predicted the practical attacks (apart from the ones to the HMI/SCADA that were not considered in the attack model for performance issues). Another interesting result is that with our attack model we can easily predict that most of the attacks will still be possible even when adding cryptography and signature to the communication between components. In addition, the mapping between the attacks found in the formal analysis are not easily mapped to the real system (due to the technical details abstracted away in the attack model) but they provide a quite accurate description of the outcome of the attack, the components involved, the semantics of the modification in the payload, and often about the strategy of the attacker.

6. RELATED WORK

In [31], a systematic analysis of the literature have been performed to collect and categorize the main aspects of a Cyber-Physical attacker model. The results have been implemented in a free open-source tool available in [28]. However, the authors do not focus on how to use the attacker models they defined for the formal analysis of CPS.

A first step in the direction of the formal analysis of a water treatment plant has been performed in [30]. A simple scenario of a subprocess of a water treatment plant has been defined. Our system model considers all the processes involved in SWaT and different attacker profiles.

In [1], the authors performed a security analysis of a water treatment testbed. They do not apply any formal analysis tool and a practical assessment is reported. They report a table with a number of attacks on the physical process. With our formal analysis we are able to detect all of the attacks

Table 4: Comparison between manual and formal analysis

Analysis by	Valves	Pumps	Pressure	Tank fill level	Chemical dosing	Historian	HMI/SCADA	PLC
Team 1				I			C	I
Team 2							C	
Team 3		I		I		S	C	S
Team 4	I	I		I	I		I	
Team 5					I		C	
Team 6					I	C	I	
Formal Verif.	C,I	C,I	C,I	C,I	C,I	I	–	I

The I, C, and S stands for Insider, Cybercriminal, and strong attacker profiles used to perform the attack. The attacker profile S and the HMI/SCADA have not been considered in the formal assessment

reported in [1] but with a loss of details due to the different abstraction level. The main difference between that work and our work is that in [1], the authors report the details of the real values of their attacks, while we identify the attack steps.

In [23], the authors analyze the security of the same water treatment testbed by creating a formal model in Alloy formal language and then analyzing the system model with Alloy analyzer [21]. The authors have not used any formal attacker model or profile and they have considered only three processes of the testbed. In this work, we applied several modifications of the standard Dolev-Yao attacker model and we also applied two different attacker profiles. As result, we detected a number of attacks that supersede the attacks reported in [23].

ADVISE. In [19, 24], authors formally define a framework for the identification of attacks in CPS. In this formalization, the authors define a set of abstract components to describe an attack execution graphs (AEG) and an attacker model. The AEG represents potential attack steps against the system, together with a formal definition of the attacker using a set of 6 dimensions. This formalization has been implemented in a framework called ADVISE where users can define their own attacker models, e.g., defining the knowledge of the attacker with respect to the AEG. The ADVISE tool does not provide a list of attacker profiles and, in general, the attacker has to be defined by the user. With our approach we provide two different attacker profile and we

provide rules that extend the standard DY attacker model capabilities.

CyberSAGE. In [41], the authors describe the Cyber Security Argument Graph Evaluation tool for CPS security assessment. The tool can automatically security argument graphs describing: the workflow of a CPS (i.e., how the system provides its functionalities), the security goals, and an attacker model. Note here that this graph matches with our definition of attack model in Section 2.1. For the description of the goal and workflow information CyberSAGE supports XMI formats input. The topology of the network is automatically extracted and the devices of the system are associated with properties (e.g., availability) according to heuristics.

7. CONCLUSIONS

In this work we proposed a formal modeling of ICS and applied our modeling technique to a real-world water treatment testbed called SWaT. In addition, we proposed a formalization of all the dimensions proposed in the APE [28] framework. The attack model that we implemented in ASLan++ is around 1000 lines long and contains more than 50 entities. The ASLan++ specification generates 269 ASLan rules (i.e., transition rules). Using that model, we then performed a formal security assessment of SWaT against two different attacker profiles, Cybercriminal and Insider, using CL-AtSe, a formal security analysis tool. We showed that with our formal technique we can identify attacks to the security of SWaT that take into account the logic of the ICS. To show that our results can be concretely used to reason on the security of ICS we compare our formal assessment results with the results of an independent practical assessment performed by six different teams (from industry and academia). The comparison shows that 7 (out of 8) different attacks found during the practical assessment were also discovered by our formal analysis. We did not cover the last attack type as our abstraction does not cover the related components in the system model.

8. REFERENCES

[1] S. Adepu and A. Mathur. An investigation into the response of a water treatment system into cyber attacks. In *IEEE Symposium on High Assurance Systems Engineering (HASE)*, 2015.

[2] D. Antonioli, H. R. Ghaeini, S. Adepu, M. Ochoa, and N. O. Tippenhauer. Gamifying education and research on ics security: Design, implementation and results of s3. 2017. Cornell University, ArXiv e-Prints, http://arxiv.org/abs/1702.03067v1.

[3] A. Armando, W. Arsac, T. Avanesov, M. Barletta, A. Calvi, A. Cappai, R. Carbone, Y. Chevalier, L. Compagna, J. Cuéllar, G. Erzse, S. Frau, M. Minea, S. Mödersheim, D. von Oheimb, G. Pellegrino, S. E. Ponta, M. Rocchetto, M. Rusinowitch, M. T. Dashti, M. Turuani, and L. Viganò. The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 267–282, 2012.

[4] A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In *Proceedings of Computer Aided Verification, (CAV)*, pages 281–285, 2005.

[5] A. Armando and L. Compagna. SATMC: a SAT-based model checker for security protocols. In *JELIA*, LNAI 3229. Springer, 2004.

[6] AVANTSSAR. Deliverable 5.3: AVANTSSAR Library of validated problem cases. www.avantssar.eu, 2010.

[7] AVANTSSAR. Deliverable 2.3 (update): ASLan++ specification and tutorial, 2011. Available at http://www.avantssar.eu.

[8] D. Basin, S. Capkun, P. Schaller, and B. Schmidt. Let's get physical: Models and methods for real-world security protocols. In *Proceedings of Theorem Proving in Higher Order Logics*, 2009.

[9] D. Basin, S. Capkun, P. Schaller, and B. Schmidt. Formal reasoning about physical properties of security protocols. *Transactions on Information and System Security (TISSEC)*, 14(2):16, 2011.

[10] D. Basin, S. Mödersheim, and L. Viganò. OFMC: A symbolic model checker for security protocols. *Journal of Information Security*, 4(3):181–208, 2005.

[11] M. Bugliesi, S. Calzavara, S. Mödersheim, and P. Modesti. Security protocol specification and verification with AnBx. *behaviour*, 15:16, 2015.

[12] A. A. Cárdenas, S. M. Amin, B. Sinopoli, A. Giani, A. Perrig, and S. S. Sastry. Challenges for securing cyber physical systems. In *Workshop on Future Directions in Cyber-physical Systems Security*. DHS, July 2009.

[13] A. A. Cárdenas, T. Roosta, and S. Sastry. Rethinking security properties, threat models, and the design space in sensor networks: A case study in SCADA systems. *Ad Hoc Networks*, 7(8):1434–1447, 2009.

[14] D. E. Denning. Activism, hacktivism, and cyberterrorism: The internet as a tool for influencing foreign policy. In *Networks and Netwars: The Future of Terror, Crime, and Militancy*. RAND Corporation, 2001.

[15] P. Derler, E. A. Lee, and A. S. Vincentelli. Modeling cyber-physical systems. *Proceedings of the IEEE*, 100(1):13–28, Jan 2012.

[16] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[17] A. Doupé, M. Cova, and G. Vigna. *Why Johnny Can't Pentest: An Analysis of Black-Box Web Vulnerability Scanners*, pages 111–131. Springer Berlin Heidelberg, 2010.

[18] B. Feddersen, K. Keefe, W. H. Sanders, C. Muehrcke, D. Parks, A. Crapo, A. Gabaldon, and R. Palla. An ontological model for constructing mobius advise security models. In *Proceedings of Conference on Dependable Systems and Networks (DSN)*, 2015.

[19] M. D. Ford, K. Keefe, E. LeMay, W. H. Sanders, and C. Muehrcke. Implementing the ADVISE security modeling formalism in möbius. In *IEEE/IFIP Conference on Dependable Systems and Networks (DSN)*, 2013.

[20] B. Galloway and G. Hancke. Introduction to industrial control networks. *Communications Surveys Tutorials, IEEE*, 15(2):860–880, 2013.

[21] D. Jackson. *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, 2006.

[22] K. H. John and M. Tiegelkamp. *IEC 61131-3: Programming Industrial Automation Systems Concepts*
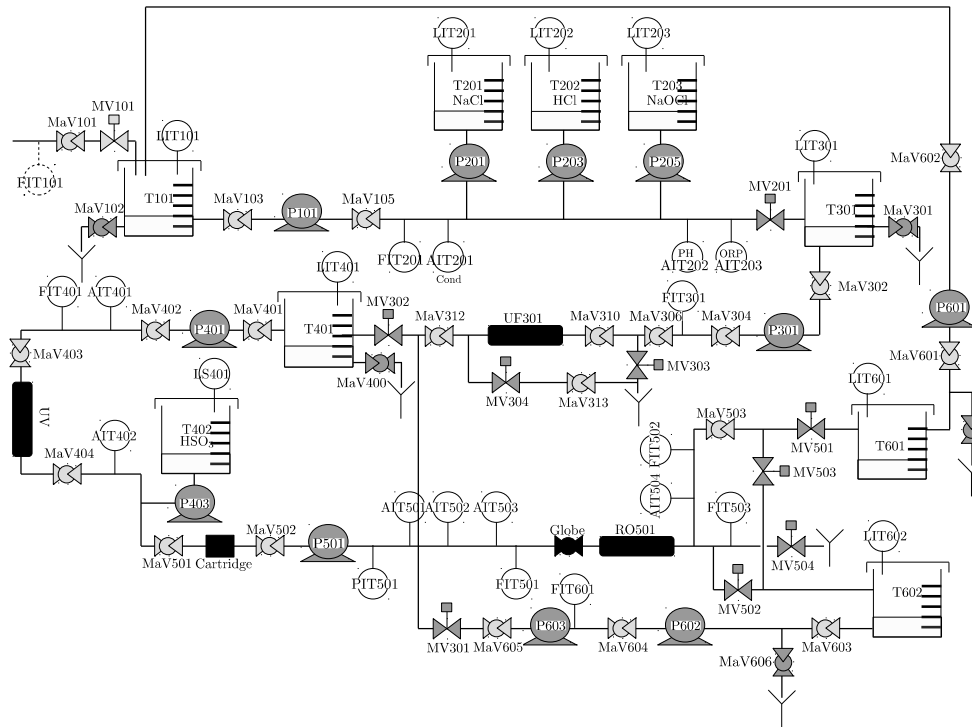
Figure 5: The details of the SWaT components and connections. The tag MaV is for manual valve, MV for motorized valve, T for tank, AIT for Analyzer Indicator Transmitter, FIT for Flow Indicator Transmitter, PIT for Pressure Indicator Transmitter

and Programming Languages, Requirements for Programming Systems, Decision-Making Aids. Springer, 2nd edition, 2010.

[23] E. Kang, S. Adepu, D. Jackson, and A. P. Mathur. Model-based security analysis of a water treatment system. In *Proceedings of the Workshop on Software Engineering for Smart Cyber-Physical Systems*, pages 22–28. ACM, 2016.

[24] E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke. Model-based security metrics using adversary view security evaluation (ADVISE). In *Proceedings of Conference on Quantitative Evaluation of Systems, QEST*, 2011.

[25] A. Mathur and N. O. Tippenhauer. A water treatment testbed for research and training on ics security. In *Proceedings of Workshop on Cyber-Physical Systems for Smart Water Networks (CySWater)*, Apr. 2016.

[26] A. Ornaghi and M. Valleri. Ettercap. https://ettercap.github.io/ettercap/, last visited August 3 2016.

[27] M. Rocchetto, M. Ochoa, and M. Torabi Dashti. Model-based detection of CSRF. In *ICT Systems Security and Privacy Protection*, volume 428 of *IFIP Advances in Information and Communication Technology*. Springer Berlin Heidelberg, 2014.

[28] M. Rocchetto and N. O. Tippenhauer. APE (Attacker Profile Examiner), 2016. Available at http://research.scy-phy.net/ape/.

[29] M. Rocchetto and N. O. Tippenhauer. ASLan++ formal model of SWaT, 2016. Available at https://research.scy-phy.net/swatmodel.

[30] M. Rocchetto and N. O. Tippenhauer. CPDY: Extending the dolev-yao attacker with physical-layer

interactions. In *Proceedings of the International Conference on Formal Engineering Methods (ICFEM)*, 2016.

[31] M. Rocchetto and N. O. Tippenhauer. On attacker models and profiles for cyber-physical systems. In *Proceedings of the European Simposium on Research in Computer Security (ESORICS)*, 2016.

[32] P. Schaller, B. Schmidt, D. A. Basin, and S. Capkun. Modeling and verifying physical properties of security protocols for wireless networks. In *Computer Security Foundations Symposium (CSF)*, pages 109–123, 2009.

[33] V. Schiffer, D. Vangompel, and R. Voss. The common industrial protocol (CIP) and the family of CIP networks. *ODVA*, 2006.

[34] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[35] R. Software. Studio 5000®. http://www.rockwellautomation.com/ rockwellsoftware/products/studio-5000.page, last visited August 3 2016.

[36] D. Steinmetzer, M. Schulz, and M. Hollick. Lockpicking physical layer key exchange: Weak adversary models invite the thief. In *Proc. ACM Conference Wireless Security (WiSeC)*, 2015.

[37] M. Turuani. The CL-Atse Protocol Analyser. In *RTA*, LNCS 4098, 2006.

[38] D. Urbina, J. Giraldo, N. O. Tippenhauer, and A. Cardenas. Attacking fieldbus communications in ICS: Applications to the SWaT testbed. In *Proceedings of Singapore Cyber Security R&D Conference (SG-CRC)*, Jan. 2016.

[39] R. Vigo. The cyber-physical attacker. In *Proceedings*

of *Workshop of Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, 2012.

[40] D. von Oheimb and S. Mödersheim. ASLan++ — a formal security specification language for distributed systems. In *FMCO*, LNCS 6957. Springer, 2010.

[41] A. H. Vu, N. O. Tippenhauer, B. Chen, D. M. Nicol, and Z. Kalbarczyk. CyberSAGE: A tool for automatic security assessment of cyber-physical systems. In *Proceeding of Quantitative Evaluation of Systems (QEST)*, pages 384–387, 2014.

[42] S. Weinberger. Computer security: Is this the start of cyberwarfare? *Nature*, 174:142–145, June 2011.

APPENDIX

A. SWaT PROCESSES

The details of the overall process of SWaT are depicted in Figure 5. The picture represents the initial status of the system model that we have considered in our formal assessment. The level of the water inside the tanks is at an optimal status (not high nor low), all the manual valves are opened, all the pumps are closed and only the first motorized valve MV101 is open (i.e., the water in T101 is filling up).