Towards Reverse Engineering of Industrial Physical Processes*

Mariano Ceccato[1 [0000−0001−7325−0316]], Youssef Driouich[1 [0000−0003−0384−464X]], Ruggero Lanotte[2 [0000−0002−3335−234X]], Marco Lucchese[1 [0000−0002−4770−5441]] and Massimo Merro[1 [0000−0002−1712−7492]]

[1] Università degli Studi di Verona, Verona, Italy
[2] Università dell'Insubria, Como, Italy

Abstract. In the last years, Industrial Control Systems (ICSs) have been the target of an increasing number of *cyber-physical attacks*, i.e., security breaches in cyberspace that adversely alter the physical processes. The main challenge attackers face in the development of cyber-physical attacks with a precise goal is obtaining an adequate level of *process comprehension*. Process comprehension is defined as "the understanding of system characteristics and components responsible for the safe delivery of service" (Green et al. 2017). While there exist a number of tools (Nmap, PLCScan, Xprobe, etc) one can use to develop a level of process comprehension through the targeting of controllers alone, they are limited by functionality, scope, and detectability. Thus, to support the execution of realistic cyber-physical attack scenario with adequate level of physical process comprehension, we propose a *black-box dynamic analysis reverse engineering tool* to derive from scans of memory registers of exposed controllers an approximated model of the controlled physical process. Such an approximated model is developed by inferring statistical properties, business processes and, in particular, *system invariants* whose knowledge might be crucial to build up stealthy (i.e., undetectable) attacks. We test the proposed methodology on a non-trivial case study, taken from the context of industrial water treatment systems.

Keywords: Industrial process; reverse engineering; business process; invariants

1 Introduction

Industrial Control Systems (ICSs) are physical and engineered systems whose operations are monitored, coordinated, controlled, and integrated by a computing and communication core [21, 30]. They represent the backbone of *Critical Infrastructures* for safety-critical applications such as electric power distribution, nuclear power production, and water supply.

* Research partially supported by the project "Dipartimenti di Eccellenza 2018-2022", funded by the *Italian Ministry of Universities and Research* (MUR).

As industrial organizations are increasingly connecting their operational (OT) network with the corporate network to improve business and operational efficiency, ICSs are more and more exposed to sophisticated cyber attacks. Indeed, in the last years, several *cyber-physical attacks* [13, 20, 23] have targeted ICSs to take control of national critical infrastructures. Some notorious examples are: (i) the *STUXnet* worm, which reprogrammed Siemens PLCs of nuclear centrifuges in the nuclear facility of Natanz in Iran [10]; (ii) the *CRASHOVERRIDE* attack on the Ukrainian power grid, otherwise known as Industroyer [31]; (iii) the recent attack to a water treatment plant of Oldsmar, Florida, where hackers boosted the level of sodium hydroxide to 100 times higher than normal [3].

Due to the increasing number of reported cyber-physical attacks, much research work has been done to develop *intrusion detection mechanisms* (IDS) to improve the resilience of ICSs to such attacks. Standard intrusion detection techniques for ICSs rely on state estimations for detecting process anomalies (see, for instance, [14, 18, 32]). An anomaly is observed if the residual error exceeds a predefined threshold. However, since there exist various sources of noise in industrial processes, a fixed threshold between normal and abnormal sensor measurements is normally hard to find.

A more efficient approach is the *invariant rule-based method* [5, 6]. Invariant rule-based methods make use of physical conditions that are known a-priori and that must hold for all ICS states. Any observed physical process values that break these rules are classified as anomalies. Typically, these invariant rules are defined by system engineers during the design of an ICS. However, this manual process is not only costly but also error-prone [7]. Thus, new frameworks have been recently designed to systematically generate invariant rules from information contained within ICS operational data logs [11]. Here, it is important to notice a couple of important points: i) the invariants are generated under strict control of system engineers by relying on the full knowledge of the system (sensors, actuator, communication network, etc); ii) the invariants implemented and checked by the IDS are chosen by system engineers between a (possibly quite) large set of invariants to focus the detection on significant anomalies which are considered potential signals of malicious activities. Invariant-based IDSs can be found in recent versions of *Secure Water Treatment system* (SWaT) [25] at the center of a series of annual cyber-physical defense exercises, referred to as Critical Infrastructure Security Showdowns (CISS) [12].

In this paper, we take a different perspective: the attacker's perspective. As argued in depth by Green et al. [16, 17], in order to support the execution of realistic cyber-physical attack scenarios the attacker requires an adequate level of *physical process comprehension*, including: operational field (e.g., water distribution rather than power generation), controllers (e.g., PLC, RTU, etc) and their network topology, relevant measurements in the plant (e.g., pressure, temperature, etc), exposed physical devices, such as sensors and actuators which may be targeted by attackers' manipulations, etc. Last but not least, in order to bypass invariant-based IDSs, it would be important for the attacker to have an approximated knowledge of the physical invariants of the system.

In this respect, we propose a prototype reverse engineering tool based on a black-box dynamic analysis to derive an approximated model of the controlled physical process from scans of memory registers of the associated controllers. Such an approximated model is derived by adapting well-known reverse engineering techniques to infer statistical properties, business processes, and state invariants from data logs capturing the state of controller registers at discrete time steps, during a period of normal operation of the target ICS. In order to show strengths and limitations of our analysis we apply our methodology to a non-trivial case study, inspired by Lanotte et al. [22], and consisting of a network of three PLCs to control (a simplified version of) the iTrust *Secure Water Treatment system* (SWaT) [25].

Outline. In Section 2, we provide an overview of PLCs and the Modbus communication protocol. In Section 3, we describe our black-box dynamic analysis for water-tank systems. In Section 4, we define a non-trivial running example. In Section 5, we apply our black-box analysis to the running example of Section 4. Section 6 draws general guidelines to apply our methodology to reverse engineer other industrial processes, and discuss related and future work.

2 Background

We give some background on *Programmable Logic Controllers* (PLCs), used to control industrial processes, and *Modbus*, a widely diffused ICS network protocol.

Programmable Logic Controllers. They are defined by the IEC 61131 standard [4] as "a digitally operating electronic system, designed for use in an industrial environment". The standard also states that a PLC has a *programmable memory* for the internal storage of user-oriented programs and a *temporary memory* to store the program's data during execution. PLCs have a simple ad-hoc architecture based on a central processing module (CPU) and further modules supporting physical inputs and outputs. The CPU executes the operating system of the PLC and runs a logic program defined by the user, called *user program*. Additionally the CPU is responsible for the communication with additional devices and manages the *process image*, i.e., a set of memory registers where all inputs (sensor measurements) and outputs (actuator commands) are copied. The user program operates on the process image rather than on the physical inputs and outputs, and runs in *scan cycles*. Each scan cycle consists of three phases: (i) reading inputs from the process image; (ii) execution of the controller code to compute how the physical process should evolve; (iii) writing outputs in the process image to govern the physical process as desired. The process image is refreshed by the CPU at the beginning and the end of each cycle, in particular, current physical inputs are copied in the process image and outputs are copied to the physical outputs, respectively.

The Modbus protocol. Modbus [26] is the first and most used internal point-to-point communication protocol between PLCs, and between PLCs and HMI interfaces. Modbus TCP basically embeds a Modbus frame into a TCP frame [26]. TCP/IP masters and slaves listen and receive Modbus data via port 502. Modbus communications are of two types: (a) query/response (communications between a master and a slave), or (b) broadcast (a master sends a command to all the slaves). A Modbus transaction comprises a single query or response frame, or a single broadcast frame. A Modbus frame message contains the address of the intended receiver, the command the receiver must execute and the data needed to execute the command.

Modbus maps the temporary memory of a PLC program to four different kinds of registers: (i) *discrete output coils*; (ii) *discrete input contacts*; (iii) *analog input registers*; (iv) *analog output holding registers*; the latter registers are also used as *general memory registers* of different sizes (16-32-64 bits). The commands used to manipulate these registers are called *function codes* and they can be found within a Modbus frame. The function codes allow for *reading* coils (FC01), discrete inputs (FC02), multiple holding registers (FC03), input registers (FC04), and for *writing* single coils (FC05), single holding registers (FC06), multiple coils (FC15), multiple holding registers (FC16). Last but not least, the Modbus protocol does not have security features, meaning that an attacker could forge, drop or modify Modbus frames without being noticed.

3 A black-box dynamic analysis for water tank systems

The *first objective* of our black-box analysis is to associate memory registers of the target PLCs to relevant ICS concepts, such as *measurements*, *actuator commands*, *(absolute) setpoints* (i.e., the range of measurements of physical variables), *ICS network communications*, etc. In particular, in our scans we focus on: (1) *discrete input contacts* and *analog input registers*, as they may possibly hold current *measurements* of physical variables; (2) *discrete output coils* and *analog output holding registers*, as they may possibly hold *actuator commands*; (3) *analog output holding registers*, when containing constant data such absolute setpoints; (4) *analog output holding registers*, when containing mutable data such as *Modbus-based messages* between two PLCs or one PLC and the associated HMI. For this purpose, in Section 3.1 we develop an ad-hoc scanning tool whose output is then used as input for a *graph analysis*, the first step of our black-box analysis.

The *second objective* of our analysis is to put in relation the runtime evolutions of these ICS concepts. At this regards, we go through a *business process analysis* enhanced with a *system-invariant analysis* (resp. Sections 3.2 and 3.3).

3.1 A scanning tool for graph analysis of PLC registers

Our scanning tool returns a dataset of the values associated to the registers of the target PLC, in a given time interval. Our approach was to leverage on the IP Protocol Scan offered by the Nmap python module [24], to identify the target

PLC within a range of IP addresses. Notice that we do not limit ourselves to the scanning of the standard Modbus TCP port 502, as in many ICSs the protocol Modbus runs on different ports (security through obscurity). Once both the IP address of the PLC and the Modbus port are discovered, the capture of registers values will start. We rely on the Ray module [27] to parallelize and distribute the readings of the values of the registers. Our tool reads and saves the values of all PLC registers in a given time frame with a desired time granularity. The data collected in our scans include the following (see, for instance, Listing 1.1): (1) IP addresses of the scanned PLCs, (2) port used by the Modbus protocol, (3) timestamps of the scan, (4) values saved in each PLC register.

Listing 1.1. Registers capture

```
"127.0.0.1/8502/2022-05-03 12_10_00.591": {
"DiscreteInputRegisters": {"%IX0.0": "0"},
"InputRegisters": {"%IW0": "53"},
"HoldingOutputRegisters": {"%QW0": "0"},
"MemoryRegisters": {"%MW0": "40","%MW1": "80"},
"Coils": {"%QX0.0": "0"}}
```

Our tool offers also the possibility to sniff Modbus network traffic via a MITM attack on the supervisory control network. Raw data collected during this phase are reported below (see, for instance, Listing 1.2): (1) timestamp of the involved Modbus commands, (2) (IP) addresses of communication source and destination PLCs, (3) Modbus function code, e.g., `read coil` or `write single coil`, (4) reference numbers, denoting the registers affected by the command, (5) argument of the function code, e.g., the value to write in a coil, (6) other fields: source/destination port, message length, request frame, etc.

Listing 1.2. Network captures

```
1 Time,Source,Destination,Protocol,Length,Function Code,
    ↪ Destination Port,Source Port,Data,Frame length on the
    ↪ wire,Bit Value,Request Frame,Reference Number,Info
2 2022-05-03 11:43:58.158,IP_PLC1,IP_PLC2,Modbus/TCP,76,Read
    ↪ Coils,46106,502,,76,TRUE,25,,"Response: Trans: 62; Unit
    ↪ : 1, Func: 1: Read Coils"
```

Then, our tool will make a mild *graph analysis*, based on R [2], to interpret data gathered from PLCs scans to possibly uncover patterns and trends. In particular, we first identify registers holding mutable values: this will give us an idea on which registers may contain measurements and/or actuator commands. Then, we use run charts to visualize the runtime evolution of PLC registers: this will allow to identify (relative) setpoints (bounding measurements) and information about the evolution of measurements and/or actuations (e.g., cyclic behaviors).

3.2 Business Process Analysis

The business process analysis is supported by the diagrams computed by DISCO [1], a commercial tool for process mining. Starting from a (set of) event log(s) consisting of the sequences of activities taken by a system, process mining is capable of reconstructing the business process that shows how the process was actually performed. The recovered business process is represented as a directed graph (similar to a UML Activity Diagram), whose nodes represent the activities in the process and edges represent the successor relations between activities.

Data available from PLC memory scanning and Modbus commands captured from the network represent the execution trace of an industrial control system. Business process mining can be run on these data to extract structured knowledge, and build a set of graphical representations that should support human understanding of the underlying industrial process.

In fact, *Modbus commands* capture quite naturally the intuition of a new activity that has just started. In particular, Modbus *write* commands are triggered from some change in the control system (e.g., a command to turn on an actuator that was off), whereas *read* commands are not particularly interesting to us because they are not triggered from changes in the system.

As regards *memory scans*, notice that they provide an instantaneous representations of the PLC memory. Thus, in order to capture new activities we compare the values of registers in two consecutive time instants (depending on the chosen time granularity). For instance, when a *boolean* variable changes value, an auxiliary activity is started with the transition from the old to the new value, whereas when a *numeric* variable changes, an internal flag keeps track of the change direction, which can be either *ascending* or *descending*. Only when the trend changes, e.g., from *ascending* to *descending*, another auxiliary activity is started, with the observed trend change.

The business process computed in this way is a valuable support for the analyst to obtain the overall picture of the OT network of the target ICS. In particular, it allows us to conjecture whether changes in the state of some actuators correspond to a specific trend of the evolution of measurements (such as increasing or decreasing evolution), and vice versa, whether monotone behaviors of the measurements correspond to specific changes in the state of the actuators. Furthermore, the business process allows us to possibly set a causality between Modbus communications and changes in the state of the physical process.

3.3 Invariant analysis

For the invariant analysis we rely on Daikon [9], a framework allowing the dynamic detection of *linear invariants*, based on a *machine learning* technique that can be applied to arbitrary data. Daikon's invariant detection runs a program (or takes an execution run as input), observes the values that the program computes, and then reports properties holding in the observed executions.

Thus, we feed Daikon with our timestamped dataset of PLC memory scans enriched with a partial bounded history of registers, and information derived

from the previous phases, such as stable states, relative setpoints (in the absence of absolute ones), and slopes of the measured data.

In the following, we give a few examples of what kind of invariants we can derive from our enriched datasets associated to memory registers of single PLCs.

- Check whether some measurement is actually bounded by some (absolute and/or relative) setpoint (both the measurements and the setpoints have been possibly identified in the previous phases);
- Check whether state changes of actuators occur when measurements approach identified setpoints, and vice versa check if when measurements approach setpoints some actuator regularly changes its state;
- Check whether state invariants of some actuators correspond to a specific trend of the evolution of measurements (such as increasing or decreasing evolution), and vice versa, check whether monotone behaviors of the measurements correspond to specific state invariants of certain actuators.

Such invariants would allow the analyst to derive a more precise causality relation between sensor measurements and actuator commands.

Our invariant analysis could be used to derive more complex invariants involving more PLCs at the same time. In general, actuations occurring in one PLC might be related to changes of the measurements of subsystems governed by other PLCs. Furthermore, by looking at more PLCs at the same time, we could find potential communication messages derived from invariants on the states of specific memory registers of communicating PLCs. Of course, as expected, and as pointed out in [7], when considering two or more physical devices at the same time we may easily have a combinatorial explosion of the associated invariants. For this reason, Daikon is able to detect invariants over at most three variables.

4 Running example

In this section, we describe a running example, implemented by Lanotte et al. [22], consisting of a network of three PLCs to control (a simplified version of) the iTrust *Secure Water Treatment system* (SWaT) [25]. SWaT represents a scaled down version of a real-world industrial water treatment plant. The system consists of six stages, each of which deals with a different treatment, including: chemical dosing, filtration, dechlorination, and reverse osmosis. For simplicity, in our use case, depicted in Figure 1, we consider only three stages.

In the first stage, raw water is pumped in a 80 gallons tank T-201, via a pump P-101. A *valve* MV-301 connects tank T-201 with a *filtration unit* that releases the treated water in a second tank T-202 (with a capacity of 20 gallons). Here, we assume that the flow of the incoming water in T-201 is greater than the outgoing flow passing through the (motor) valve MV-301. The water in T-202 flows into a *reverse osmosis unit* to reduce inorganic impurities. In the last stage, the water coming from the reverse osmosis unit is either distributed as clean water, if required standards are met, or stored in a backwash tank T-203 and then pumped back, via a pump P-103, to the filtration unit. Here, we assume

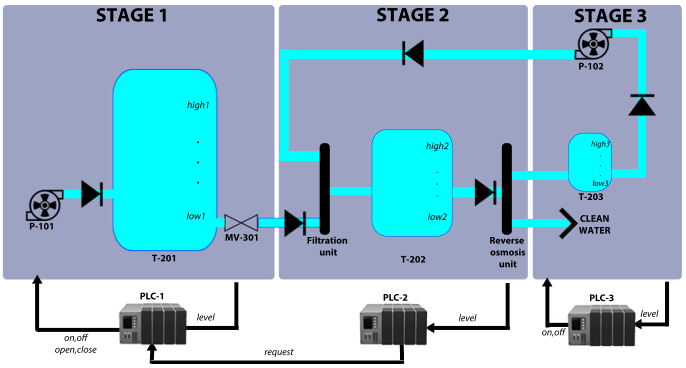


Fig. 1. A simplified Industrial Water Treatment System.

that tank T-202 is large enough to receive the whole content of tank T-203 at any moment (the capacity of T-203 is 1 gallon).

Each tank is controlled by a dedicated PLC. In the following, we give the descriptions of the user programs of the PLCs associated to each tank.

Let us start with the user program of PLC1 managing the tank T-201. Intuitively, when the pump P-101 is off, the level of water in T-201 drops until it reaches its *low setpoint* (hard coded in the memory register) $low1$; when this happens the pump is turned on and it remains so until the tank is refilled, reaching its *high setpoint* (hard-coded in the memory register) $high1$. Thus, for instance, if the pump is off then PLC1 checks the water level of the tank T-201, distinguishing between three possible states. If T-201 reaches its low setpoint $low1$ then the pump is turned on and the valve is closed. Otherwise, if the tank T-201 is at some intermediate level between the low and the high setpoint then PLC1 listens for requests arriving from PLC2 to open/close the valve. Precisely, if PLC1 gets an open request then it opens the valve, letting the water flow from T-201 to T-202, otherwise, if it gets a close request then it closes the valve; in both cases the pump remains off. If the level of the tank T-201 reaches its high setpoint $high1$ then the requests of water coming from PLC2 are served as before, but the pump is eventually turned off.

PLC2 checks for the water level of tank T-202 and behaves accordingly. If the level reaches the low setpoint (hard-coded in the memory register) $low2$ then PLC2 sends a request to PLC1, via a proper Modbus channel to open the valve MV-301 that lets the water flow from T-201 to T-202, and then returns. The channel transmission is implemented by copying a boolean value stored in a memory register of PLC2 into a corresponding register of PLC1. Otherwise, if the level of tank T-202 reaches the high setpoint $high2$ then PLC2 asks PLC1 to close the valve, via the same channel, and then returns. Finally, if the tank T-202

is at some intermediate level between $low2$ and $high2$ then the valve remains open (respectively, closed) when the tank is refilling (respectively, emptying).

Finally, PLC3 checks for the water level of the backwash tank T-203 and behaves accordingly. If the level reaches the low setpoint $low3$ then PLC3 turns off a pump P-103, and then returns. Otherwise, if the level of T-203 reaches the high setpoint $high3$ then the pump P-103 is turned on until the whole content of T-203 is pumped back into the filtration unit of T-202.

5 A methodology to extrapolate process comprehension

The black-box analysis of Section 3 is now applied on the SWaT system presented in Section 4, as a concrete case study.

Our methodology supports a top-down understanding process, in which the attacker starts from the big picture of the industrial process, where all the collected data and trends are available at once, and they are computed in an automated way. Based on the analysis results at the higher level of details, the attention gradually moves to a lower and lower level of detail, by specifying where to focus and what part of the system to isolate for further analysis and comprehension. Of course, automated inference can in principle report an overwhelming number of system properties and invariants, and it could be hard to dig into them to identify the most informative ones according to the system under scrutiny. In this respect, a *business process analysis* will be helpful as it allows us to highlight relevant trends and to conjecture interesting properties. Subsequently, an *invariant analysis* will allow us to refine such partial knowledge to confirm or disprove trends and conjectures derived in the business process analysis.

5.1 Data Collection and Graph analysis

The data collection process has been conducted for six hours, getting one data point per second, as the entire cycle of the system takes around 30 minutes. Each data point consists of 168 attributes (55 registers plus 1 auxiliary slope attribute for each PLC, explained later).

As IP addresses of PLCs are hard to read, we automatically replace them by abstract names which are arbitrarily computed by concatenating the prefix `PLC` to a progressive unique integer (e.g., `PLC1` and `PLC2`).

The analysis starts by detecting the type of data contained in registers and whether they are (significant) constant or mutable data.

Property 1. The registers `PLC1_MemoryRegisters_MW0`, `PLC1_MemoryRegisters_MW1`, `PLC2_MemoryRegisters_MW0`, `PLC2_MemoryRegisters_MW1`, `PLC3_MemoryRegisters_MW0` and `PLC3_MemoryRegisters_MW1` contain constant integer values (40, 80, 10, 20, 0 and 10, respectively).

Property 2. `PLC1_Coils_QX00`, `PLC1_Coils_QX01`, `PLC1_Coils_QX02`, `PLC2_Coils_QX00`, `PLC3_Coils_QX00` and `PLC3_Coils_QX01` contain mutable Boolean values.

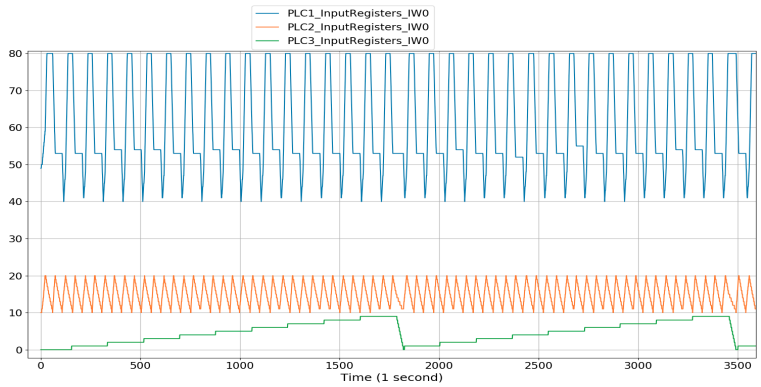


Fig. 2. Sample of the execution traces of the `InputRegisters_IW0` of the three PLCs.

Property 3. The PLC registers `PLC1_InputRegisters_IW0`, `PLC2_InputRegisters_IW0` and `PLC3_InputRegisters_IW0` contain mutable integer values.

Here, the input registers mentioned in Property 3 represent an interesting aspect to investigate further, as they might contain measurements related to the industrial process. Figure 2 shows a trace of their evolution collected during the capture by means of our tool. We can speculate that the trends of these registers are linked to a recurrent cyclic behavior, typical of tank systems.

Conjecture 1. Input registers `PLC1_InputRegisters_IW0`, `PLC2_InputRegisters_IW0` and `PLC3_InputRegisters_IW0` contain measurement values.

Moreover, from Figure 2 we can identify the range of values for the input registers of the three PLCs.

Conjecture 2 (Relative setpoints).
- The relative setpoints of `PLC1_InputRegisters_IW0` are 40 and 80.
- The relative setpoints of `PLC2_InputRegisters_IW0` are 10 and 20.
- The relative setpoints of `PLC3_InputRegisters_IW0` are 0 and 9.

5.2 Business Process Analysis

We move on to the business process analysis to highlight relevant system behaviors.

Figure 3(a) is automatically computed by business process mining, using data about PLC2 states and Modbus commands. Here, all messages originate from PLC2, and they are all directed to PLC1; the communicated messages are then written in the coils `PLC1_Coils_QX02`.

Property 4. PLC2 sends messages to PLC1 which are recorded in `PLC1_Coils_QX02`.

We also observe that PLC2_InputRegister_IW0 contains descending values for most of the time (33 seconds). Then, PLC2_Coils_QX00 switches to true and shortly after (either 1 or 0 seconds, depending on the path) the input register inverts the trend and starts an ascending trend, that lasts for 13 seconds. After this interval of time, the coils return to false and shortly afterwards (2 seconds), the input register starts a descending trend.

Conjecture 3. PLC2_Coils_QX00 determines the trend in tank T-202.

In Figure 3(a), we can also notice that shortly after a command from PLC2 is sent to PLC1 (to set PLC1_Coils_QX02 to false, but PLC1 is not show in this diagram), PLC2_InputRegister_IW0 inverts the trend that starts descending. Conversely, shortly after PLC1_Coils_QX02 moves to true, the input register starts an ascending trend.

We can now analyze the states of PLC1 in Figure 4. In the bottom-left part of the diagram, we observe that when PLC1_Coils_QX00 switches to true, an ascending trend in PLC1_InputRegister_IW0 is started. On the other hand, when PLC1_Coils_QX00 switches to false, after a while the trend becomes descending.

Conjecture 4. When PLC1_Coils_QX00 changes its state from false to true, it activates an ascending trend in tank T-201.

In Figure 3(b), a cyclic behavior is observed on PLC3, and involves two coils. When PLC3_InputRegister_IW0 starts an ascending trend, PLC3_Coils_QX02 is immediately set to false. Then, after quite a long time (27 minutes), PLC3_Coils_QX00 is set to true, and shortly after (4 seconds) the trend in the input register switches to descending for 32 seconds. Then, almost at the same time PLC3_Coils_QX00 switches to false and coils PLC3_Coils_QX02 switches to true. Then, the trends in the input register are reverted again to restart the cycle.

Conjecture 5. Coils PLC3_Coils_QX00 activates a decreasing trend in tank T-203, whereas coils PLC3_Coils_QX02 activates an increasing trend.

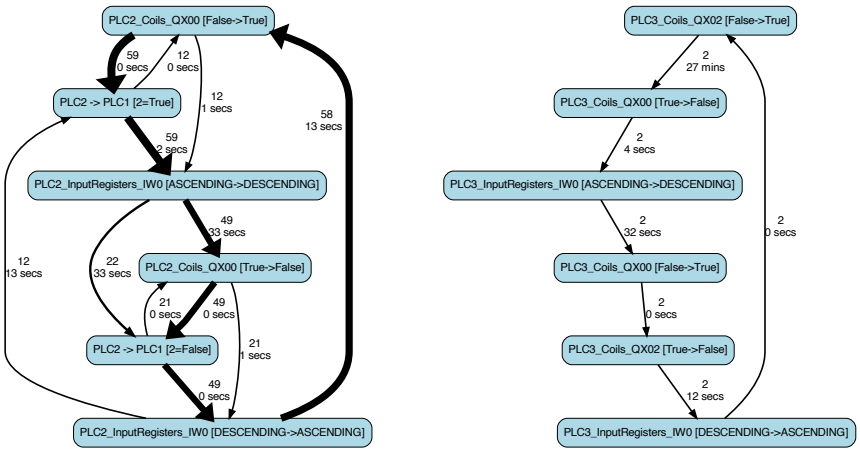
5.3 Process Invariants analysis

In this section, we confirm and refine the results obtained in the previous steps of the analysis looking for system invariants using the Daikon framework.

Setpoints. First of all, we confirm Conjecture 2 on *relative setpoints* of the water levels: six specific memory registers contain constant values coinciding with the lower and the upper relative setpoints of the three tanks. Actually, these memory registers contain the *absolute setpoints*:

```
PLC1_InputRegisters_IW0 >= PLC1_MemoryRegisters_MW0 == 40.0
PLC1_InputRegisters_IW0 <= PLC1_MemoryRegisters_MW1 == 80.0
PLC2_InputRegisters_IW0 >= PLC2_MemoryRegisters_MW0 == 10.0
PLC2_InputRegisters_IW0 <= PLC2_MemoryRegisters_MW1 == 20.0
PLC3_InputRegisters_IW0 >= PLC3_MemoryRegisters_MW0 == 10.0
PLC3_InputRegisters_IW0 <= PLC3_MemoryRegisters_MW1 == 0.0
```

Property 5. The MemoryRegisters_MW0 and MemoryRegisters_MW1, associated to each PLC, contain, respectively, the lower and the upper absolute setpoints.



(a) (b)

Fig. 3. (a) states in PLC2 and Modbus commands, (b) states in PLC3.

Communication channels. We are in position to validate and refine Property 4, stating the presence of a communication channel from PLC2 to PLC1. More precisely, from `PLC1_Coils_QX01 == PLC1_Coils_QX02 == PLC2_Coils_QX00` we derive:

Property 6. There is a *communication channel* from PLC2 to PLC1: what it is written in `PLC2_Coils_QX00` is then copied in both registers `PLC1_Coils_QX02` and `PLC1_Coils_QX01`.

Actuator changes and levels' evolution. We use Daikon to investigate the relations between the coils and the measurements.

Let us check what happens when the state of `PLC1_Coils_QX01` changes. In this case, nothing relevant arises from the analysis of PLC3. However, we get significant information from PLC1 and PLC2. As regards PLC2, when the coils `PLC1_Coils_QX01` changes from true to false, the measurements of tank T-202 reach the upper absolute setpoint as the level of water was increasing:

```
PLC2_InputRegisters_IW0 == PLC2_MemoryRegisters_MW2 == 20.0  &&   PLC2_slope > 0 .
```

Moreover, when `PLC1_Coils_QX01` moves from false to true, the measurements of T-202 reach the lower absolute setpoint as the water level was decreasing:

```
PLC2_InputRegisters_IW0 == PLC2_MemoryRegisters_MW1 == 10.0  &&  PLC2_slope < 0 .
```

Now, let us check what happens when `PLC1_Coils_QX01` remains unchanged for a significant amount of time (say 6 seconds). In this case, when the coils remains true, the level of water in T-202 increases, as `PLC2_slope > 0`.[3] Whereas

[3] The slope is an auxiliary attribute indicating the trend of the measurement.

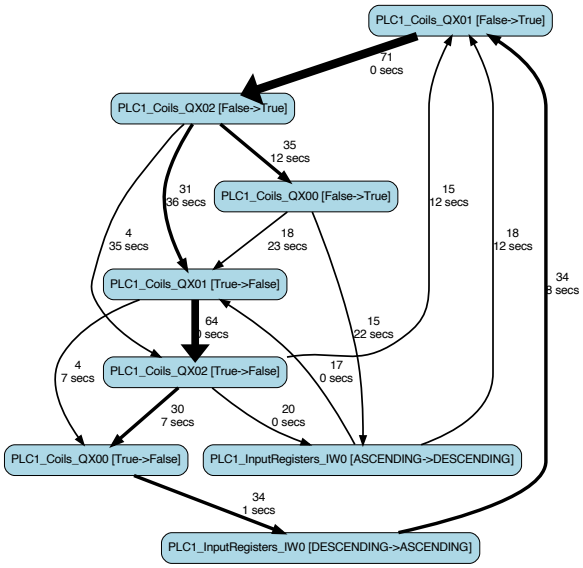


Fig. 4. Business process with states in PLC1 and Modbus commands.

when `PLC1_Coils_QX01` remains false the level of the tank is not increasing, as `PLC2_slope <= 0`. As `PLC2_Coils_QX00` is the only used coils in PLC2, from the equalities `PLC1_Coils_QX01 == PLC1_Coils_QX02 == PLC2_Coils_QX00` we can reformulate Conjecture 3 as follows:

Property 7. The level in T-202 increases iff `PLC1_Coils_QX01 == true`. The level in T-202 is non-increasing iff `PLC1_Coils_QX01 == false`.

However, `PLC1_Coils_QX01` is controlled by PLC1, and by Conjecture 4 we believe that `PLC1_Coils_QX00` determines increasing trends in tank T-201. Thus, let us check what happens when `PLC1_Coils_QX00` and `PLC1_Coils_QX01` are both false, i.e., no incoming water in both tanks T-201 and T-202. In this case, the invariant analysis says that the level in tank T-201 is stable: `PLC1_slope == 0`. On the other hand, when `PLC1_Coils_QX00` is false and `PLC1_Coils_QX01` is true the level in T-201 is decreasing while the level in T-202 is increasing: `PLC1_slope < 0` and `PLC2_slope > 0`. Recall the following invariant: `PLC1_Coils_QX01 == PLC1_Coils_QX02`. Now, we have full knowledge about all coils of PLC1 to refine Conjecture 4 on the trend of the level of water in T-201:

Property 8. The level of water in T-201 increases if and only if `PLC1_Coils_QX00 == true`. The level of water in T-201 decreases if and only if `PLC1_Coils_QX00 == false` and `PLC1_Coils_QX01 == true`.

Let us check what happens when `PLC3_Coils_QX00` remains unchanged for a significant amount of time (say 5 seconds). In this case, when the coils remains true, the level of water in T-203 decreases, as `PLC3_slope < 0`. Whereas,

when PLC3_Coils_QX00 remains false the level of the tank is not decreasing, as PLC3_slope >= 0. Thus, we are able to confirm the first part of Conjecture 5.

Property 9. The level of water in T-203 decreases iff PLC3_Coils_QX00 == true. The level of water in T-203 is non-decreasing iff PLC3_Coils_QX00 == false.

Actuator changes and setpoints. Now, we try to investigate the relation between the coils PLC1_Coils_QX00 and the setpoints in T-201. In particular, when the coils changes from true to false, the measurements have reached the upper absolute setpoint: PLC1_InputRegisters_IW0 == PLC1_MemoryRegisters_MW1 == 80.0. Conversely, when PLC1_Coils_QX00 changes from false to true, the measurements have reached the lower setpoint: PLC1_InputRegisters_IW0 == PLC1_MemoryRegisters_MW0 == 40.0. Formally,

Property 10. When PLC1_Coils_QX00 changes its state from true to false, the tank T-201 reaches the upper absolute set point; when it changes from false to true, the tank reaches the lower absolute set point.

Now, we try to investigate the relation between the coils PLC3_Coils_QX00 and the setpoints in T-203. In particular, when the coils changes from false to true, the measurements have reached the upper absolute setpoint: PLC3_InputRegisters_IW0 == PLC3_MemoryRegisters_MW1 == 10.0. Conversely, when PLC3_Coils_QX00 changes from true to false, the measurements have reached the lower setpoint: PLC3_InputRegisters_IW0 == PLC3_MemoryRegisters_MW0 == 0.0. Formally,

Property 11. When PLC3_Coils_QX00 changes its state from false to true, the tank T-203 reaches the upper absolute set point; when it changes from true to false, the tank reaches the lower absolute set point.

5.4 Discussion

At the end of our black-box analysis, we derived a quite significant comprehension on the industrial process of our case study in Section 4. Summarizing, we were able to get the following information:

- which registers contain measurements of the physical process (e.g., the water level) and their operative working ranges (setpoints);
- which registers contain actuator commands (e.g., pumps and valves)
- which conditions trigger changes in actuators (e.g., specific water levels trigger pumps and valves) and vice versa the causal relation between the state of actuators and the governed physical process;
- how PLCs communicate, and how such communications affect the underlying physical process.

On the other hand, there a number of information on the physics that we are not able to derive, such as: (i) sources and sinks of water in the system; (ii) the existence of pipelines between tanks; (iii) which actuators handle incoming or outcoming flows, for instance in Property 8 we cannot derive that PLC1_Coils_QX00 is actually associated to a pump governing the incoming flow in tank T-201.

6 Conclusions, related and future work

From our black-box analysis in the previous section, we can draw *general guidelines* to apply our methodology to reverse engineer other industrial processes.

- *Two phases methodology:* Reverse engineering of an industrial process purely based on register scans and network captures requires first to acquire a general idea of the system. This consists in elaborating conjectures on how the industrial process works and how it is controlled. The subsequent, more formal, analysis is meant to provide support for these conjectures and build factual knowledge on the control system.
- *Correct estimation of sampling frequency:* The attacker should have enough information to understand if she is tackling a *slow* physical process, such as a water-tank system, or a *faster* process, involving, for instance, rotor engines. In fact, the most appropriate frequency to read PLCs register depends on the intrinsic speed of the process, as a too slow sampling rate might miss important events (e.g., the maximum level of water before it starts decreasing) while a too high sampling rate might confuse measurement error and actual trends (e.g., spurious trend inversions). If this information is not available, initial scans should be visualized (such as Figure 2) to tune sampling rate.
- *Working conditions:* The attacker should start by identifying the nominal working conditions, consisting of the operative ranges of the observed measurements. In fact, the typical responsibility of a control system is to keep the physical system within safe and secure working conditions.
- *Trigger conditions:* These are conditions on the state of physical variables triggering a reaction of the controller via actuator commands to drive the physical system to a desired operational state. Identifying these trigger conditions is the subsequent goal of the attacker.
- *Causal relation between actuator commands and physical evolution*: Once actuator commands are sent by the controller, some changes should be observed in the physical process. Understanding this causal relation would allow the attacker to explain the reaction time of the physical process.
- *Physical meaning for pieces of evidence:* When the analysis is able to deliver a large number of process invariants, the attacker should be able to understand which invariants are relevant in invariant-based IDSs of the target system.

Related work. Attackers often aim at attacking programs to alter their execution flow, for instance to subvert the outcome of a license check. Studies with professional hackers and practitioners [8] revealed that the actual attack commonly requires a preliminary investigation on the target program. Either static analysis (e.g., decompiling) and dynamic analysis (e.g., debugging) typically support a malicious reverse engineering activity, aiming at understanding the behavior of the target program, to locate interesting assets and to plan an attack strategy.

In the context of industrial control systems, Keliris and Maniatakos [19] proposed a methodology to automate the reverse engineering process of *PLC binaries*. They develop a framework whose modules are instantiated for reversing binaries compiled with CODESYS, a widely used compiler for PLCs.

Identification of hybrid dynamical systems, i.e., the automatization of the mechanistic modeling of hybrid dynamical systems from observed data, has seen a number of results spanning over two decades [29]. More recently, Yuan et al. [34] proposed a general framework for discovering cyber-physical systems directly from data. The framework involves the identification of physical systems together with their dynamics, as well as the inference of transition logics. More precisely, the proposed framework tries to understand the underlying mechanism of cyber-physical systems as well as make predictions concerning their state trajectories based on the discovered models.

Feng et al. [11] have recently designed a framework to systematically generate invariant rules from information contained within ICS operational data logs [11]. Such invariants are then selected by system engineers to generate invariant-based IDSs. Experimental results by the same authors [7,11] show that under the same false positive rate, invariant-based IDS ability to detect anomalies outperforms the residual error-based detection model by a clear margin.

Few works have addressed the need for process comprehension from an attacker's perspective; a critical precondition when seeking to achieve operational impact beyond simple denial-of-service [15,17,33].

Winnicki et al. [33] proposed an alternative approach to discover the dynamic behaviour of a cyber-physical system with *probing*. They slightly perturb the system and observe how the controls react to take the system back to the nominal status. The challenge for the attacker is to alter the system enough to make observable changes, but changes should be small enough to not be revealed as anomalies by potential intrusion detection systems.

Green et al. [17] provided two practical examples based on a Man-In-The-Middle scenario, demonstrating the types of information an attacker would need obtain, collate, and comprehend, in order to begin targeted ICS process manipulation and detection avoidance. The paper provides a step-by-step example of ICS reconnaissance, required for the successful establishment of process comprehension, and execution of network-based and host-based MITM attacks.

More recently, Green et al. [16] proposed the concept of Process Comprehension at a Distance, a novel methodological and automatable approach to the system-agnostic identification of PLC library functions, leading to the targeted exfiltration of operational data, manipulation of control-logic behavior, and establishment of covert command and control channels through unused memory.

Future work. Our preliminary results pave the way towards novel opportunity of research, meant to deepen the knowledge on how automated reverse engineering might support and facilitate attackers in conducting campaigns against ICSs.

First of all, we are interested in quantifying the gap between the invariants that dynamic analysis can infer (attacker perspective) and the full list of those invariants that can be written by an analyst with the complete knowledge of the industrial system [5,6,11] (defender perspective). This gap might effectively estimate the distance between an invariant-based intrusion detection system and an attacker, to measure the competitive advantage of the defender (if any) and, consequently, the detection potential of an IDS.

Our automatically inferred invariants can be used to understand an ICS and support manual attack to it. However, a second line of research might consist in using these invariants also to (maybe partially) automate an attack that is capable of compromising an industrial process (e.g., to delay the water cleaning process) in a way that does not violate invariants (e.g., with a nominal level decreasing rate), with the final objective to be hard to detect by a defender.

Last but not least, as Daikon [9] detects only linear relations over at most three variables, we plan to investigate other tools, such as DIG [28], to detect nonlinear equalities and inequalities of arbitrary degrees, defined over program variables.

Acknowledgements. We thank the anonymous reviewers for valuable comments.

References

1. Fluxicon disco, `https://fluxicon.com/disco/`
2. R project for statistical computing (1993), `https://www.r-project.org/`
3. A Hacker Tried to Poison a Florida City's Water Supply (2021), `https://www.wired.com/story/oldsmar-florida-water-utility-hack/`, accessed: 2022-05-14
4. 61131-3, I.S.I.: Programmable Controllers - Part 3: Programming Languages. second ed., Int'l Electrotechnical Commission (2003)
5. Adepu, S., Mathur, A.: Using Process Invariants to Detect Cyber Attacks on a Water Treatment System. In: SEC. IFIP Advances in Information and Communication Technology, vol. 471, pp. 91–104. Springer (2016)
6. Adepu, S., Mathur, A.: From Design to Invariants: Detecting Attacks on Cyber Physical Systems. In: QRS-C. pp. 533–540. IEEE (2017)
7. Adepu, S., Mathur, A.: Distributed Attack Detection in a Water Treatment Plant: Method and Case Study. IEEE Trans. Dependable Secur. Comput. **18**(1), 86–99 (2021)
8. Ceccato, M., Tonella, P., Basile, C., Falcarin, P., Torchiano, M., Coppens, B., De Sutter, B.: Understanding the behaviour of hackers while performing attack tasks in a professional setting and in a public challenge. Empirical Software Engineering **24**(1), 240–286 (Feb 2019)
9. Ernst, M.D., Perkins, J.H., Guo, P.J., McCamant, S., Pacheco, C., Tschantz, M.S., Xiao, C.: The Daikon system for dynamic detection of likely invariants. Sci. Comput. Program. **69**(1-3), 35–45 (2007)
10. Falliere, N., Murchu, L., Chien, E.: W32.Stuxnet Dossier (2011)
11. Feng, C., Palleti, V.R., Mathur, A., Chana, D.: A Systematic Framework to Generate Invariants for Anomaly Detection in Industrial Control Systems. In: NDSS. The Internet Society (2019)
12. Furtado, F., Shrivastava, S., Mathur, A., Goh, N.: The Design of Cyber-Physical Exercises (CPXs). In: CyCon. IEEE (2022)
13. Giraldo, J., Urbina, D., Cardenas, A., Valente, J., Faisal, M., Ruths, J., Tippenhauer, N.O., Sandberg, H., Candell, R.: A Survey of Physics-Based Attack Detection in Cyber-Physical Systems. ACM Computing Surveys **51**(4), 76:1–76:36 (2018)
14. Goh, J., Adepu, S., Tan, M., Lee, Z.S.: Anomaly Detection in Cyber Physical Systems Using Recurrent Neural networks. In: HASE. pp. 140–145. IEEE Computer Society (2017)

15. Gollmann, D., Gurikov, P., Isakov, A., Krotofil, M., Larsen, J., Winnicki, A.: Cyber-Physical Systems Security: Experimental Analysis of a Vinyl Acetate Monomer Plant. In: CCPS@ASIACCS. pp. 1–12. ACM (2015)
16. Green, B., Derbyshire, R., Krotofil, M., Knowles, W., Prince, D., Suri, N.: PCaaD: Towards automated determination and exploitation of industrial systems. Comput. Secur. **110**, 102424 (2021)
17. Green, B., Krotofil, M., Abbasi, A.: On the Significance of Process Comprehension for Conducting Targeted ICS attacks. In: CPS-SPC@CCS. pp. 57–67. ACM (2017)
18. Hadziosmanovic, D., Sommer, R., Zambon, E., Hartel, P.H.: Through the eye of the PLC: semantic security monitoring for industrial processes. In: ACSAC. pp. 126–135. ACM (2014)
19. Keliris, A., Maniatakos, M.: ICSREF: A Framework for Automated Reverse Engineering of Industrial Control Systems Binaries. In: NDSS. The Internet Society (2019)
20. Krotofil, M., Gollmann, D.: Industrial control systems security: What is happening? In: INDIN. pp. 670–675. IEEE (2013)
21. Lanotte, R., Merro, M.: A Calculus of Cyber-Physical Systems. In: LATA. LNCS, vol. 10168, pp. 115–127. Springer (2017)
22. Lanotte, R., Merro, M., Munteanu, A.: Industrial Control Systems Security via Runtime Enforcement. ACM TOPS (To appear). `https://doi.org/10.1145/3546579`
23. Lanotte, R., Merro, M., Munteanu, A., Viganò, L.: A Formal Approach to Physics-based Attacks in Cyber-physical Systems. ACM TOPS **23**(1), 3:1–3:41 (2020)
24. Lyon, G.: Nmap (1997), `https://nmap.org/`
25. Mathur, A.P., Tippenhauer, N.O.: SWaT: a water treatment testbed for research and training on ICS security. In: CySWater@CPSWeek. pp. 31–36. IEEE Computer Society (2016)
26. Modbus, I.: Modbus application protocol specification v1. 1a. North Grafton, Massachusetts (www. modbus. org/specs. php) (2004)
27. Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M.I., I., S.: Ray: A Distributed Framework for Emerging AI Applications. In: USENIX. pp. 561–577. USENIX Association (2018)
28. Nguyen, T., Kapur, D., Weimer, W., Forrest, S.: DIG: A Dynamic Invariant Generator for Polynomial and Array Invariants. ACM Trans. Softw. Eng. Methodol. **23**(4), 30:1–30:30 (2014)
29. Paoletti, S., Juloski, A.L., Ferrari-Trecate, G., Vidal, R.: Identification of Hybrid Systems: A tutorial. Eur. J. Control **13**(2-3), 242–260 (2007)
30. Rajkumar, R., Lee, I., Sha, L., Stankovic, J.A.: Cyber-physical systems: the next computing revolution. In: DAC. pp. 731–736. ACM (2010)
31. Slowik, J.: Anatomy of an attack: Detecting and defeating CRASHOVERRIDE. VB2018, October pp. 1–23 (2018)
32. Urbina, D.I., Giraldo, J.A., Cárdenas, A.A., Tippenhauer, N.O., Valente, J., Faisal, M.A., Ruths, J., Candell, R., Sandberg, H.: Limiting the Impact of Stealthy Attacks on Industrial Control Systems. In: CCS. pp. 1092–1105. ACM (2016)
33. Winnicki, A., Krotofil, M., Gollmann, D.: Cyber-Physical System Discovery: Reverse Engineering Physical Processes. In: CPSS@ASIACCS. pp. 3–14. ACM (2017)
34. Yuan, Y., Tang, X., Zhou, W., Pan, W., Li, X., Zhang, H.T., Ding, H., Goncalves, J.: Data driven discovery of cyber physical systems. Nature Communications **10**(1), 4894 (2019)