UNIVERSITY OF VERONA

Department of COMPUTER SCIENCE

Master's Degree in
COMPUTER SCIENCE AND ENGINEERING

Master Thesis

Towards Process Comprehension of Industrial Control Systems: a Framework for Analyzing Industrial Systems

Candidate:
Marco OLIANI
VR457249

Supervisor:
Prof. **Massimo MERRO**

Co-supervisors:
Prof. **Ruggero LANOTTE**
Marco LUCCHESE

Academic Year 2022/2023

"If you spend more on coffee than on IT security, you will be hacked. What's more, you deserve to be hacked"
(Richard Clarke)

Abstract

Bla bla bla

Contents

Chapter 1

Introduction

LOREM ipsum dolor bla bla bla. Ma dove metto l'abstract? Prova di interlinea che direi posso anche andare bene, ma bisogna poi vedere il tutto come si incastra alla fine, in modo da ottenere un bel risultato alla vista.

1.1 Contribution

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

1.2 Outline

The thesis is structured as follows:

Chapter 2: provides background on the topics covered in this thesis: Industrial Control Systems (ICSs), Supervisory Control And Data Acquisition (SCADA), Programmable Logic Controllers (PLCs) and other devices, industrial communication protocols.

Chapter 3:

Chapter 4:

Chapter 5:

Chapter 6:

Chapter 7:

Chapter 2

Background

2.1 Industrial Control Systems in a nutshell

INDUSTRIAL CONTROL SYSTEMS (ICSs) are information systems used to control industrial processes such as manufacturing, product handling, production, and distribution [1].

ICSs are often found in critical infrastructure facilities such as power plants, oil and gas refineries, and chemical plants.

ICSs are different from traditional IT systems in several key ways. Firstly, ICSs are designed to control physical processes, whereas IT systems are designed to process and store data. This means that ICSs have different requirements for availability, reliability, and performance. Secondly, ICSs are typically deployed in environments that are harsh and have limited resources, such as extreme temperatures and limited power. Thirdly, the protocols and hardware used in ICSs are often proprietary and not widely used outside of the industrial sector.

ICSs are becoming increasingly connected to the internet and other networks, which has led to increased concerns about their security. Industrial systems were not originally designed with security in mind, and many of them have known vulnerabilities that could be exploited by attackers. Additionally, the use of legacy systems and equipment can make it difficult to

implement security measures. As a result, ICSs are increasingly seen as a potential target for cyber attacks, which could have serious consequences for the safe and reliable operation of critical infrastructure.

The increasing connectivity of ICSs and the associated security risks have led to a growing interest in the field of ICS security. Researchers and practitioners are working to develop new security technologies, standards, and best practices to protect ICSs from cyber attacks. This includes efforts to improve the security of ICS networks and devices, as well as the development of new monitoring and detection techniques to identify and respond to cyber attacks.

2.2 ICS components

Industrial control systems (ICSs) are composed of several different components that work together to monitor and control industrial processes.

2.2.1 SCADA systems

Supervisory Control And Data Acquisition (**SCADA**) is a system of software and hardware elements that allows industrial organizations to [2]:

- Control industrial processes locally or at remote locations
- Monitor, gather, and process real-time data
- Directly interact with devices such as sensors, valves, pumps, motors, and more through human-machine interface (HMI) software
- Record events into a log file

The SCADA software processes, distributes, and displays the data, helping operators and other employees analyze the data and make important decisions.

SCADA systems are known for their ability to monitor and control large-scale industrial processes, and for their ability to operate over long

distances. This makes them well-suited for use in remote locations or for controlling processes that are spread out over a wide area. However, the same features that make SCADA systems so useful also make them vulnerable to cyber attacks.

SCADA systems were not originally designed with security in mind, and many of them have known vulnerabilities that could be exploited by attackers. Additionally, the use of legacy systems and equipment can make it difficult to implement security measures. As a result, SCADA systems are increasingly seen as a potential target for cyber attacks, which could have serious consequences for the safe and reliable operation of critical infrastructure.

To secure SCADA systems, it is important to implement security measures such as network segmentation, secure communication protocols, and access control. Additionally, it is important to monitor SCADA systems for unusual activity and to implement incident response procedures to quickly detect and respond to any security breaches.

2.2.1.1 SCADA architecture

According to the *Purdue Enterprise Reference Architecture* (PERA), or simply **Purdue Model**, SCADA architecture consists in **six levels** each representing a functionality [3], as shown in Figure 2.1:

- Level 0 (**Processes**): contains **field devices** (2.2.2), or *sensors*.
- Level 1 (**Intlligent Devices**): includes **local or remote controllers** that sense, monitor and control the physical process, such as **PLCs** (2.2.3) and **RTUs** (2.2.4). Controllers interface directly to the field devices reading data from sensors and sending commands to actuators.
- Level 2 (**Control Systems**): contains computer systems used to supervising and monitoring the physical process: they provide a **Human-Machine Interface** (*HMI*, 2.2.5) and *Engineering Workstations* (EW) for operator control.

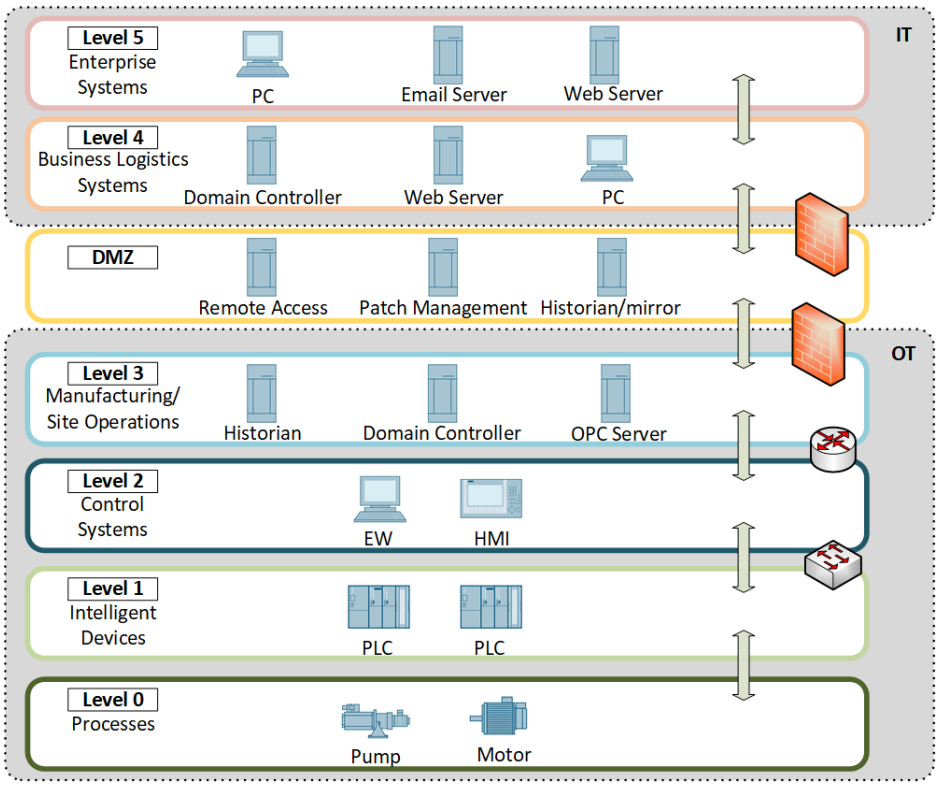


Figure 2.1: SCADA architecture schema

- Level 3 (**Manufactoring/Site Operations**): comprises systems used to manage the production workflow for plant-wide control: they collate informations from the previous levels and store them in Data Historian servers.

- **Industrial Demilitarized Zone** (DMZ): intermediate level that connects the *Operational Technology* (OT) part (levels 0-3) with the *Information Technology* (IT) part of the system (levels 4 and 5). Communication takes place indirectly through services such as *proxy servers* and *remote access servers*, which act as intermediaries between the two environments.

- Level 4 (**Business Logistics Systems**): collect and aggregates data from the Manufactoring/Site Operations level overseeing the IT-related activities to generate **reporting** to the Enterprise System layer. At

this layer we can find application and e-mail servers, and *Enterprise Resource Planning* (ERP) systems.

- Level 5 (**Enterprise Systems**): represents the enterprise network, used for the business-to-business activities and for business-to-client purpose services. At Enteprise Systems level are typical IT services such as mail servers, web servers and all the systems used to manage the ongoing process.

2.2.2 Field devices

Field devices are the **sensors** and **actuators** that are used to collect data from the process and control it. Examples of field devices include temperature sensors, pressure sensors, valves and pumps.

2.2.3 Programmable Logic Controllers

A *Programmable Logic Controller* (PLC) is a **small and specialized industrial computer** having the capability of controlling complex industrial and manifacturing processes [4].

Compared to relay systems and personal computers, PLCs are optimized for control tasks and industrial environments: they are rugged and designed to withdraw harsh conditions such as dust, vibrations, humidity and temperature: they have more reliability than personal computers, which are more prone to crash, and they are more compact a require less maintenance than a relay system. Furthermore, I/O interfaces are already on the controller, so PLCs are easier to expand with additional I/O modules (if in a rack format) to manage more inputs and ouputs, without reconfiguring hardware as in relay systems when a reconfiguration occours.

PLCs are more *user-friendly*: they are not intended (only) for computer programmers, but designed for engineers with a limited knowledge in programming languages: control program can be entered with a simple

and intuitive language based on logic and switching operations instead of a general-purpose programming language (*i.e.* C, C++, ...).

2.2.3.1 PLC Architecture

The basic hardware architecture of a PLC consists of these elements [5]:

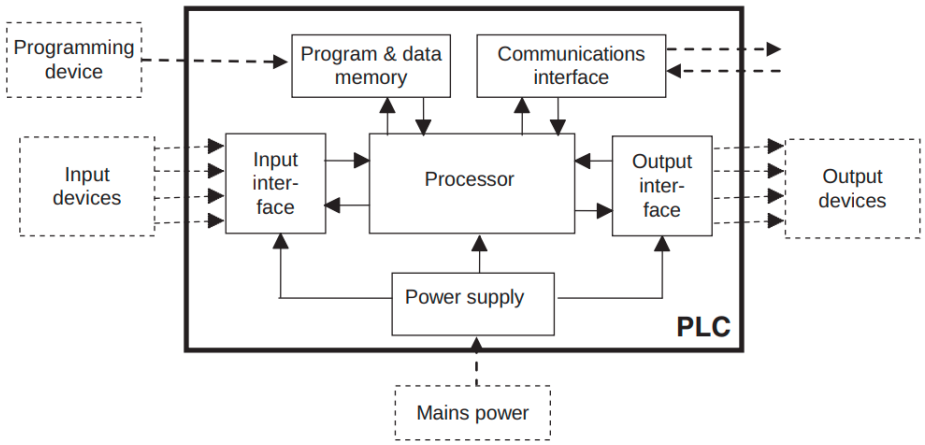


Figure 2.2: PLC architecture

- **Processor unit (CPU)**: contains the microprocessor. This unit interpretes the input signals from I/O modules, executes the control program stored in the Memory Unit and sends the output signals to the I/O Modules. The processor unit also sends data to the Communication interface, for the communication with additional devices.
- **Power supply unit:** converts AC voltage to low DC voltage.
- **Programming device:** is used to store the required program into the memory unit.
- **Memory Unit:** consists in RAM memory and ROM memory. RAM memory is used for storing data from inputs, ROM memory for storing operating system, firmware and user program to be executed by the CPU.

- **I/O modules:** provide interface between sensors and final control elements (actuators).
- **Communications interface:** used to send and receive data on a network from/to other PLCs.

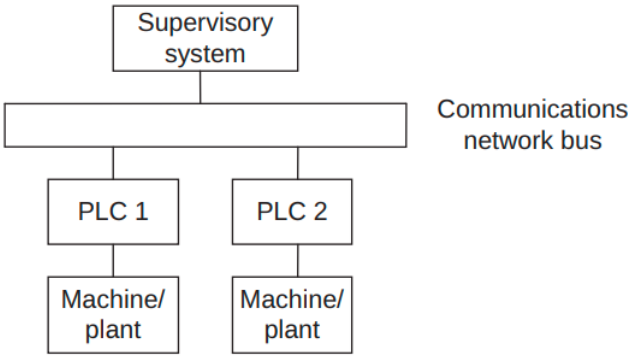


Figure 2.3: PLC communication schema

2.2.3.2 PLC Programming

Two different programs are executed in a PLC: the **operating system** and the **user program**.

The operating system tasks include executing the user program, managing memory areas and the *process image table* (memory registers where inputs from sensors and outputs for actuators are stored).

The user program needs to be uploaded on the PLC via the programming device and runs on the process image table in *scan cycles*: each scan is made up of three phases [6]:

1. reading inputs from the process images table
2. execution of the control code and computing the physical process evolution

3. writing output to the process image table to have an effect on the physical process. At the end of the cycle, the process image table is refreshed by the CPU

Standard PLCs **programming languages** are basically of two types: **textuals** and **graphicals**. Textual languages include languages such as *Instruction List* (IL) and *Structured Text* (ST), while *Ladder Diagrams* (LD), *Function Block Diagram* (FBD) and *Sequential Function Chart* (SFC) belong to the graphical languages.

Graphical languages are more simple and immediate comparing to the textual ones and are preferred by programmers because of their features and simplicity, in particular the **Ladder Logic programming** (see Figure 2.4 for a comparison).

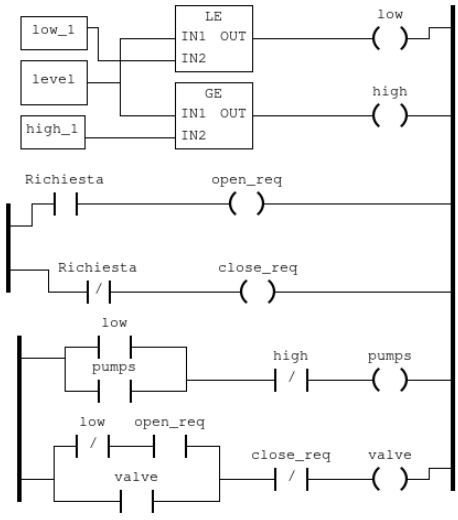
```
PROGRAM PLC1
  VAR
    level AT %IW0 : INT;
    Richiesta AT %QX0.2 : BOOL;
    request AT %IW1 : INT;
    pumps AT %QX0.0 : BOOL;
    valve AT %QX0.1 : BOOL;
    low AT %MX0.0 : BOOL;
    high AT %MX0.1 : BOOL;
    open_req AT %MX0.3 : BOOL;
    close_req AT %MX0.4 : BOOL;
    low_1 AT %MW0 : INT := 40;
    high_1 AT %MW1 : INT := 80;
  END_VAR
  VAR
    LE3_OUT : BOOL;
    GE7_OUT : BOOL;
  END_VAR

  LE3_OUT := LE(level, low_1);
  low := LE3_OUT;
  GE7_OUT := GE(level, high_1);
  high := GE7_OUT;
  open_req := Richiesta;
  close_req := NOT(Richiesta);
  pumps := NOT(high) AND (low OR pumps);
  valve := NOT(close_req) AND (open_req AND NOT(low) OR valve);
END_PROGRAM


CONFIGURATION Config0

  RESOURCE Res0 ON PLC
    TASK task0(INTERVAL := T#20ms,PRIORITY := 0);
    PROGRAM instance0 WITH task0 : PLC1;
  END_RESOURCE
END_CONFIGURATION
```

(a) Example of ST programming



(b) Example of Ladder Logic

Figure 2.4: Comparison between ST language and Ladder Logic

2.2.3.3 PLC Security

PLCs were originally designed to operate as closed systems, not connected and exposed to the outside world via communication networks:

the question of the safety of these systems, therefore, was not a primary aspect. The advent of Internet has brought undoubted advantages, but has introduced problems relating to the safety and protection of PLCs from external attacks and vulnerabilities.

Indeed, a variety of different communication protocols used in ICSs are designed to be efficient in communications, but do not provide any security measure i.e. confidentiality, authentication and data integrity, which makes these protocols vulnerable against many of the IT classic attacks such as *Replay Attack* or *Man in the Middle Attack*.

Countermeasures to enhance security in PLC systems may include [7]:

- protocol modifications implementing **data integrity**, **authentication** and **protection** against *Replay Attacks*
- use of *Intrusion Detection and Prevention Systems* (IDP)
- creation of *Demilitarized Zones* (DMZ) on the network

In addition to this, keeping the process network and Internet separated, limiting the use of USB devices among users to reduce the risks of infections, and using strong account management and maintenance policies are best practices to prevent attacks and threats and to avoid potential damages.

2.2.4 Remote Terminal Units

Remote Terminal Units (RTUs) are computers with radio interfacing similar to PLCs: they transmit telemetry data to the control center or to the PLCs and use messages from the master supervisory system to control connected objects [8].

The purpose of RTUs is to operate efficiently in remote and isolated locations by utilizing wireless connections. In contrast, PLCs are designed for local use and rely on high-speed wired connections. This key difference

allows RTUs to conserve energy by operating in low-power mode for extended periods using batteries or solar panels. As a result, RTUs consume less energy than PLCs, making them a more sustainable and cost-effective option for remote operations.

Industries that require RTUs often operate in areas without reliable access to the power grid or require monitoring and control substations in remote locations. These include telecommunications, railways, and utilities that manage critical infrastructure such as power grids, pipelines, and water treatment facilities. The advanced technology of RTUs allows these industries to maintain essential services, even in challenging environments or under adverse weather conditions.

2.2.5 Human-Machine Interface

The *Human-Machine Interface* (HMI) is the hardware and software interface that operators use to monitor the processes and interact with the ICS.
An HMI shows the operator and authorized users information about system status and history; it also allows them to configure parameters on the ICS such as set points and, send commands and make control decisions [9].

The HMI can be in the form of a physical panel, with buttons and indicator lights, or PC software.

2.2.6 Cybersecurity components

Cybersecurity components, as seen in section 2.2.3.3 about PLCs security, are used to protect ICSs from cyber threats and vulnerabilities. They can include firewalls, *Intrusion Detection and Prevention systems* (IDP), and *Security Information and Event Management* (SIEM) systems.

2.3 Communication Networks

Communication Networks are the networks that are used to connect the different components of the ICS and allow them to communicate with each other. Communication networks can include wired and wireless networks, such as Ethernet/IP, Modbus, DNP3 and others.

2.3.1 ICS Communication Protocols

As mentioned in Section 2.1, industrial systems differ from classical IT systems in the purpose for which they are designed: controlling physical processes the former, processing and storing data the latter. For this reason, ICSs require different communication protocols than traditional IT systems for real time communications and data transfer.

A wide variety of industrial protocols exists: this is because originally each vendor developed and used its own proprietary protocol. However, these protocols were often incompatible with each other, resulting in devices from different vendors being unable to communicate with each other.
To solve this problem, standards were defined with a view to allowing these otherwise incompatible device to intercommunicates.

Among all the various protocols, some have risen to prominence as widely accepted standards. These *de facto* protocols are commonly utilized in industrial systems due to their proven reliability and effectiveness. In the following sections, we will provide a brief overview of some of the most prevalent and widely used protocols in the industry.

2.3.1.1 Modbus

Modbus is a serial communication protocol developed by Modicon (now Schneider Electric) in 1979 for use with its PLCs [10] and designed expressly for industrial use: it facilitates interoperability of different devices

connected to the same network (sensors, PLCs, HMIs, ...) and it is also often used to connect RTUs to SCADA acquisition systems.

Modbus is the most widely used communication protocol among industrial systems because it has several advantages:

- simplicity of implementation and debugging
- it moves raw bits and words, letting the individual vendor to represent the data as it prefers
- it is, nowadays, an **open** and ***royalty-free*** protocol: there is no need to sustain licensing costs for implementation and use by industrial device vendors

Modbus is a **request/response** (or *master/slave*) protocol: this makes it independent of the transport layer used.

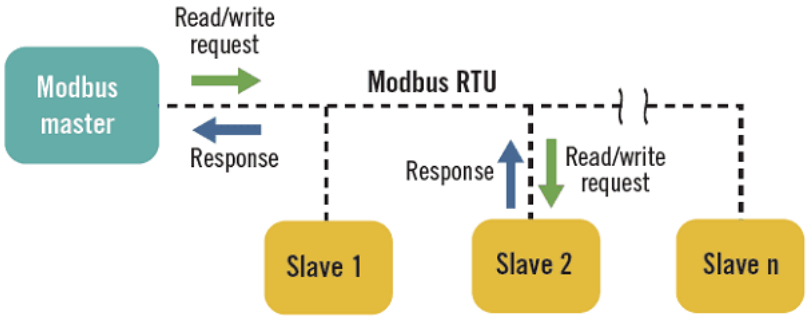


Figure 2.5: Modbus Request/Response schema

In this kind of architecture, a single device (master) can send requests to other devices (slaves), either individually or in broadcast: these slave devices (usually peripherals such as actuators) will respond to the master by providing data or performing the action requested by the master using the Modbus protocol. Slave devices cannot generate requests to the master [11].

There are several variants of Modbus, of which the most popular and widely used are Modbus RTU (used in serial port connections) and Modbus TCP (which instead uses TCP/IP as the transport layer). Modbus TCP embeds a standard Modbus frame in a TCP frame (see Figure 2.6): both masters and slaves listen and receive data via TCP port 502.

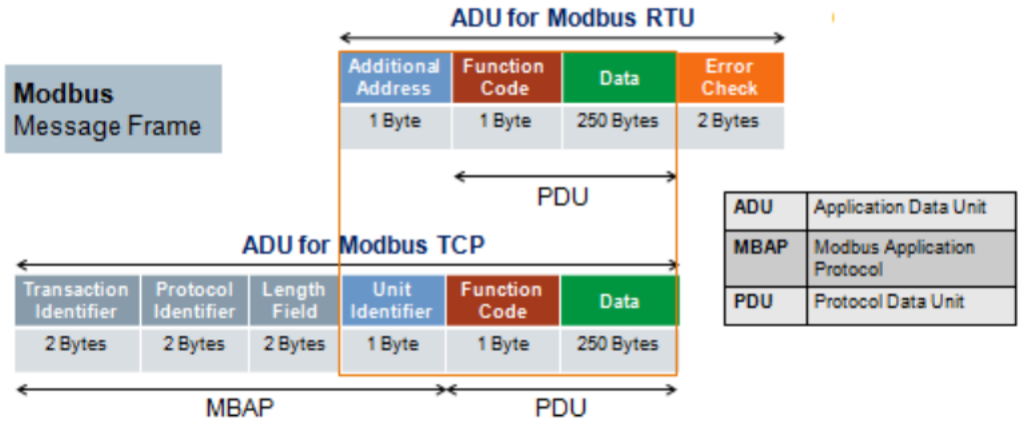


Figure 2.6: Modbus RTU frame and Modbus TCP frame

Modbus registers Modbus provides four object types, which map the data accessed by master and slave to the PLC memory:

- *Coil*: binary type, read/write accessible by both masters and slaves
- *Discrete Input*: binary type, accessible in read-only mode by masters and in read/write mode by slaves
- *Analog Input*: 16 bits in size (word), are accessible in read-only mode by masters and in read/write mode by slaves
- *Holding Register*: 16 bits in size (word), accessible in read/write mode by both masters and slaves. Holding Registers are the most commonly used registers for output and as general memory registers.

Modbus Function Codes *Modbus Function Codes* are specific codes used by the Modbus master within a request frame (see Figure 2.6) to tell the

Modbus slave device which register type to access and which action to perform on it.

Two types of Function Codes exists: for data access and for diagnostic Function Codes list for data access are listed in Table 2.1:

Function Code	Description
FC01	Read Coils
FC02	Read Discrete Input
FC03	Read Holding Registers
FC04	Read Analog Input Registers
FC05	Write/Force Single Coil
FC06	Write/Force Single Holding Register
FC15	Write/Force Multiple Coils
FC16	Write/Force Multiple Holding Registers

Table 2.1: Modbus Function Codes list

Modbus Security Issues Despite its simplicity and widespread use, the Modbus protocol does not have any security feature, which exposes it to vulnerabilities and attacks.

Data in Modbus are transmitted unencrypted (*lack of confidentiality*), with no data integrity controls (*lack of integrity*) and authentication checks (*lack of authentication*), in addition to the *lack of session*. Hence, the protocol is vulnerable to a variety of attacks, such as Denial of Services (DoS), buffer overflows and reconnaissance activities.

The easiest attack to bring to the Modbus protocol, however, is **packet sniffing**: since, as mentioned earlier, network traffic is unencrypted and the data transmitted is in cleartext, it is sufficient to use a packet sniffer to capture the network traffic, read the packets and thus gather informations about the system such as ip addresses, function codes of requests and to modify the operation of the devices.

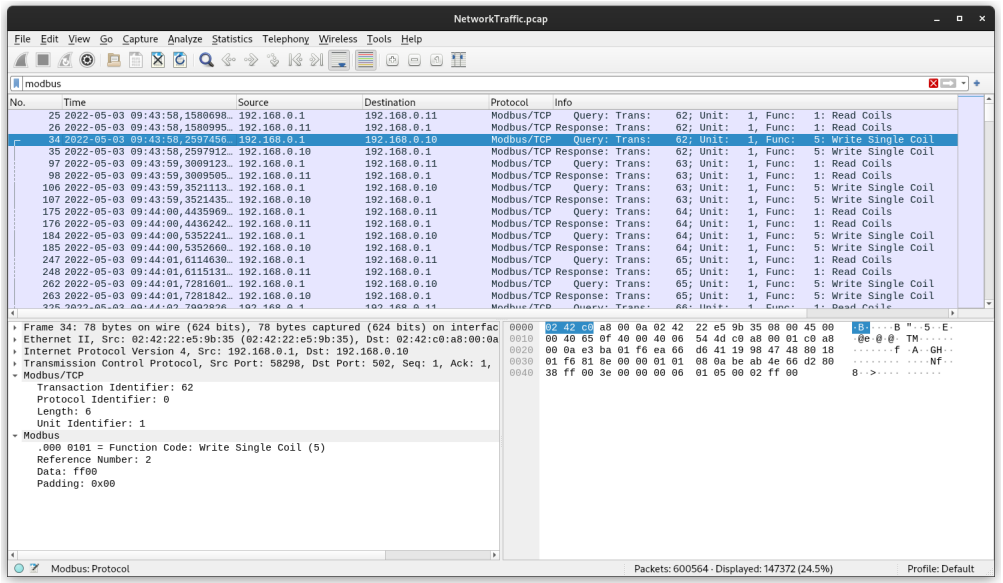


Figure 2.7: Example of packet sniffing on the Modbus protocol

To make the Modbus protocol more secure, an encapsulated version was developed within the *Transport Security Layer* (TLS) cryptographic protocol, also using mutual authentication. This version of the Modbus protocol is called **Secure Modbus** or **Modbus TLS**. In addition to this, Secure Modbus also includes X.509-type certificates to define permissions and authorisations [12].

2.3.1.2 EtherNet/IP

EtherNet/IP (where IP stands for *Industrial Protocol*) is an open industrial protocol that allows the *Common Industrial Protocol* (CIP) to run on a typical Ethernet network [13]. It is supported by ODVA [14].

EtherNet/IP uses the major Ethernet standards, such as IEEE 802.3 and the TCP/IP suite, and implements the CIP protocol stack at the upper layers of the OSI stack (see Figure 2.8). It is furthermore compatible with the main Internet standard protocols, such as SNMP, HTTP, FTP and DHCP, and other industrial protocols for data access and exchange such as *Open Platform Communication* (OPC).

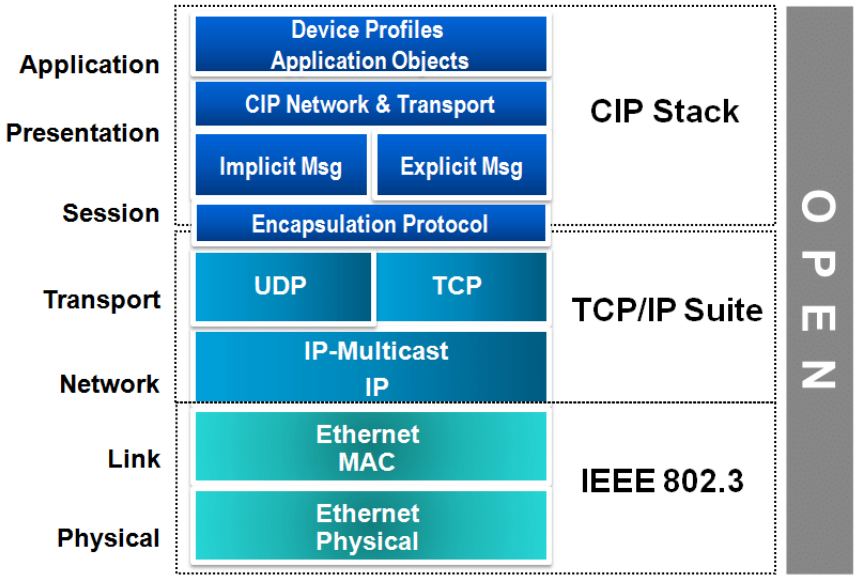


Figure 2.8: OSI model for EtherNet/IP stack

Physical and Data Link layer The use of the IEEE 802.3 standard allows EtherNet/IP to flexibly adopt different network topologies (star, linear, ring, etc.) over different connections (copper, fibre optic, wireless, etc.), as well as the possibility to choose the speed of network devices.
IEEE 802.3 in addition defines at Data Link layer the *Carrier Sense Multiple Access - Collision Detection* (CSMA/CD) protocol, which controls access to the communication channel and prevents collisions.

Transport layer At the transport level, EtherNet/IP encapsulates messages from the CIP stack into an Ethernet message, so that messages can be transmitted from one node to another on the network using the TCP/IP protocol. EtherNet/IP uses two forms of messaging, as defined by CIP standard [13][15]:

- **unconnected messaging:** used during the connection establishment phase and for infrequent, low priority, explicit messages. Unconnected messaging uses TCP/IP to transmit messages across the network asking for connection resource each time from the *Unconnected*

Message Manager (UCMM).

- **connected messaging:** used for frequent message transactions or for real-time I/O data transfers. Connection resources are reserved and configured using communications services available via the UCMM.

  EtherNet/IP has two types of message connection [13]:

  - **explicit messaging:** *point-to-point* connections to facilitate *request-response* transactions between two nodes. These connections use TCP/IP service on port 44818 to transmit messages over Ethernet.
  - **implicit messaging:** this kind of connection moves application-specific **real-time I/O data** at regular intervals. It uses multicast *producer-consumer* model in contrast to the traditional *source-destination* model and UDP/IP service (which has lower protocol overhead and smaller packet size than TCP/IP) on port 2222 to transfer data over Ethernet.

Session, Presentation and Application layer At the upper layers, EtherNet/IP implements the CIP protocol stack. We will discuss this protocol more in detail in Section 2.3.1.3.

2.3.1.3 Common Industrial Protocol (CIP)

The *Common Industrial Protocol* (CIP) is an open industrial automation protocol supported by ODVA. It is a **media independent** (or *transport independent*) protocol using a *producer-consumer* communication model and providing a **unified architecture** throughout the manufacturing enterprise [16][17].
CIP has been adapted in different types of network:

- **EtherNet/IP**, adaptation to *Transmission Control Protocol* (TCP) technologies

- **ControlNet**, adaptation to *Concurrent Time Domain Multiple Access* (CTDMA) technologies
- **DeviceNet**, adaptation to *Controller Area Network* (CAN) technologies
- **CompoNet**, adaptation to *Time Division Multiple Access* (TDMA) technologies

CIP objects CIP is a *strictly object oriented* protocol at the upper layers: each object of CIP has **attributes** (data), **services** (commands), **connections**, and **behaviors** (relationship between values and services of attributes) which are defined in the **CIP object library**. The object library supports many common automation devices and functions, such as analog and digital I/O, valves, motion systems, sensors, and actuators. So if the same object is implemented in two or more devices, it will behave the same way in each device [18].

Security [19] In EtherNet/IP implementation, security issues are the same as in traditional Ethernet, such as network traffic sniffing and spoofing. The use of the UDP protocol also exposes CIP to transmission route manipulation attacks using the *Internet Group Management Protocol* (IGMP) and malicious traffic injection.

Regardless of the implementation used, it is recommended that certain basic measures be implemented on the CIP network to ensure a high level of security, such as *integrity*, *authentication* and *authorization*.

2.3.1.4 Other Protocols

Chapter 3

State of the Art

IN TRADITIONAL IT, an attacker aims to understand the behavior of a program through various techniques so as to bring attacks aimed at changing its execution flow, functionalities or bypassing limits imposed by the licensing of such software. These attack techniques include a **preliminary study** of the program: a *static analysis* (i.e., a preliminary analysis of the software without it running) and a *dynamic analysis* (i.e., an analysis performed with the program running).
The result of these two preliminary investigation techniques is a **reverse engineering** of the software, which is useful for identifying any weaknesses or bugs and therefore planning an attack.

In the OT context, however, the concept of *reverse engineering* is also associated with that of ***process comprehension***, a term coined by Green et al.'s [20] to describe the understanding of the characteristics of the system and the physical elements of within it, that are responsible for its proper functioning.

Not much knowledge exists in the literature regarding the collection and analysis of information concerning the understanding and operation of an ICS: in Section 3.1 we will look at a quick overview of some of the existing literature on the subject and in the following sections we will focus in particular on one of the papers exposed.

3.1 Literature on Process Comprehension

Keliris and Maniatikos The first approach presented in this section is by Keliris and Maniatakos [21]: they present a methodology for automating the reverse engineering of ICS binaries based on a *modular framework* (called ICSREF) that can reverse binaries compiled with CODESYS, one of the most popular and widely used PLC compilers, irrespective of the language used.

Yuan et al. Yuan et al. [22] propose a *data-driven* approach to discovering cyber-physical systems from data directly: to achieve this goal, they have implemented a framework whose purpose is to identify physical systems and transition logic inference, and to seek to understand the mechanisms underlying these cyber-physical systems, making furthermore predictions concerning their state trajectories based on the discovered models.

Feng et al. Feng et al. [23] developed a framework that can generate system *invariant rules* based on machine learning and data mining techniques from ICS operational data log. These invariants are then selected by systems engineers to derive IDS systems from them.

The experiment results on two different testbeds, the *Water Distribution system* (WaDi) and the *Secure Water Treatment system* (SWaT), both located at the iTrust - Center for Research in Cyber Security at the University of Singapore [24], show that under the same false positive rate invariant-based IDSs have a higher efficiency in detecting anomalies than IDS systems based on a residual error-based model.

Pal et al. Pal et al. [25] work is somewhat related to Feng et al.'s: this paper describes a data-driven approach to identifying invariants automatically using *association rules mining* [26] with the aim of generate invariants sometimes hidden from the design layout. The study has the same objective of Feng et al.'s and uses too the iTrust SwaT System as testbed.

Currently this technique is limited to only pair wise sensors and actuators: for more accurate invariants generation, the technique adopted must be capable of deriving valid constrains across multiple sensors and actuators.

Winnicki et al. Winnicki et al. [27] instead propose a different approach to process comprehension based on the **attacker's perspective** and not limited to mere *Denial of Service* (DoS): their approach is to discover the dynamic behavior of the system, in a semi-automated and process-aware way, through *probing*, that is, slightly perturbing the cyber physical system and observing how it reacts to changes and how it returns to its original state. The difficulty and challenge for the attacker is to perturb the system in such a way as to achieve an observable change, but at the same time avoid this change being seen as a system anomaly by the IDSs.

Green et al. Green et al. [20] also adopt an approach based on the attacker's perspective: this approach consists of two practical examples in a *Man in the Middle* (MitM) scenario to obtain, correlate, and understand all the types of information an attacker might need to plan an attack to alter the process while avoiding detection.

The paper shows *step-by-step* how to perform a ICS **reconnaissance**, which is fundamental to process comprenension and thus to the execution of MiTM attacks.

Ceccato et al. Ceccato et al. [6] propose a methodology based on a *black box dynamic analysis* of an ICS using a reverse engineering tool to derive from the scans performed on the memory registers of the exposed PLCs and network scans an approximate model of the physical process. This model is obtained by inferring statistical properties, business process and system invariants from data logs.

The proposed methodology was tested on a non-trivial case study, using a testbed inspired by an industrial water treatment plant.

In the next section I will examine this latest work in more detail, which will be the basis for my work and thus the subsequent chapters of this thesis.

3.2 Ceccato et al.'s methodology for analyzing water-tank systems

As mentioned earlier, the paper proposes a methodology based on a black box dynamic analysis of an ICS by identifying potential PLCs on the network and scanning the memory registers of the identified controllers to obtain an approximate model of the controlled physical process.

The first objective of this black box analysis is to associate the various memory registers of the target PLCs with a correspondence to the basic concepts of an ICS such as sensors (otherwise known as measurements), actuators, setpoints (range of values of a physical variable), network communications, and so on.
This is performed by analyzing the different types of memory registers associated with the Modbus protocol and trying to figure out what type of data they may contain.
The second objective is to put in relation the runtime evolution of these basic concepts.

To achieve this, Ceccato et al. developed a prototype tool [28] that performs reverse engineering of the physical system through four phases:

1. **scanning of the system and data pre-processing**: data gathering is performed to generate the data logs of PLCs registers
2. **graphs and statistical analysis**: provides information about the memory registers using graphs and statistical data derived from the gathered data
3. **invariant inference and analysis**: generates system invariants and allows user to view invariants related to a given sensor or actuator

4. **business process mining**: reconstructs, from event logs, the business process that shows how process is carried out

In Figure 3.1 we have a schematic representation of the workflow related to this work. We will cover all these phases in detail in the next sections of this chapter.

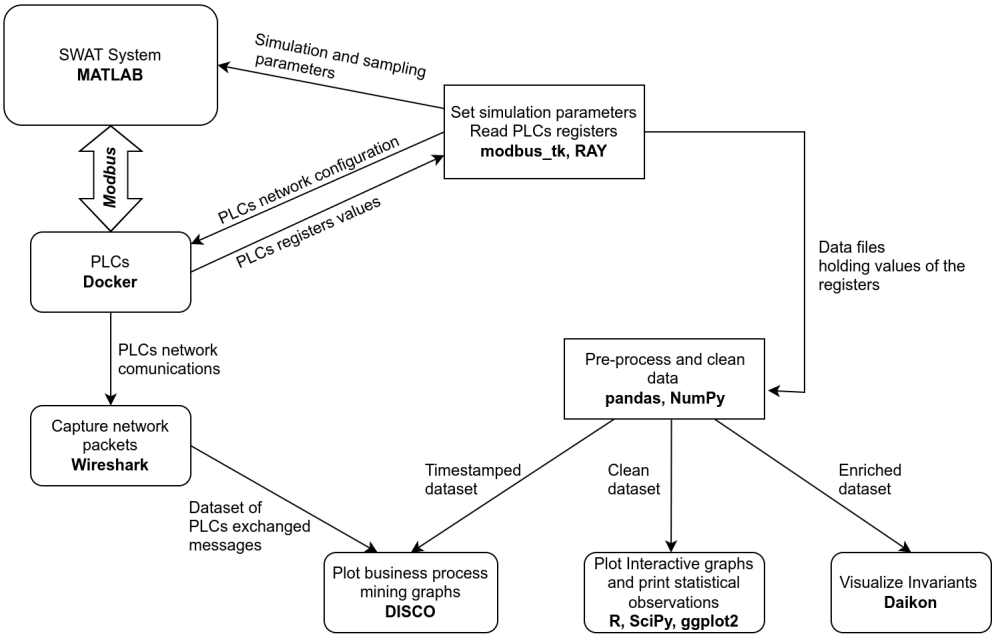


Figure 3.1: Overview

3.2.1 Testbed

Before describing the various phases of the methodology, let's take a look at the testbed on which this methodology will be tested. The testbed used to test this methodology is a (very) simplified version of the iTrust SWaT system [29] implemented by Lanotte et al. [30]: in Figure 3.2 we can see a graphical representation of the testbed. This simplified version consists of three stages, each controlled by a dedicated PLC:

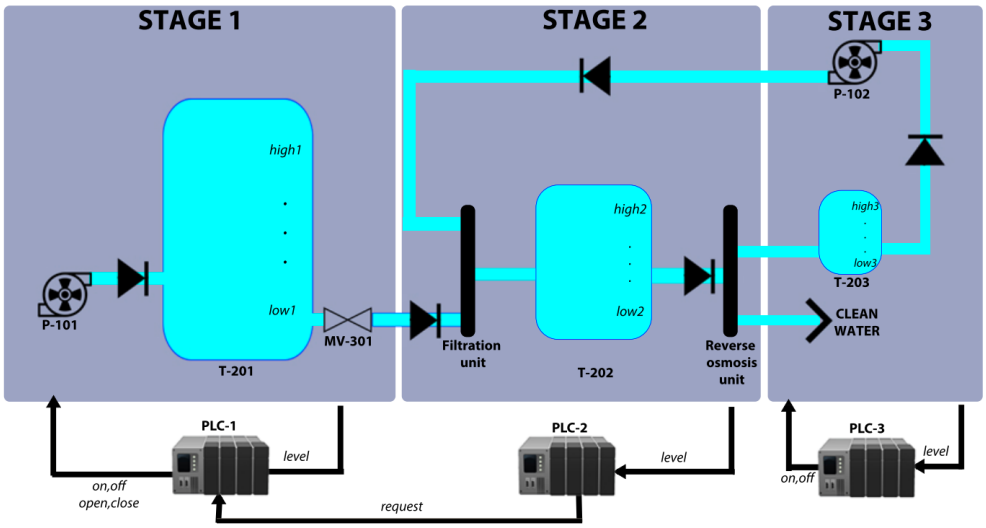


Figure 3.2: The simplified SWaT system used for running Ceccato et al. methodology

Stage 1 At the first stage, a **tank** with a capacity of 80 gallons (identified by the code T-201) is filled with raw water by the P-101 pump: the MV-301 valve (where MV stands for *motorized valve*), also connected to the T-201 tank, flushes out the water collected in the tank to send it to the second stage, first to the *filtration unit* (here not identified by any sensor) and from there to a **second tank**, identified by the code T-202 and with a capacity of 20 gallons.

Stage 2 At the second stage, water contained in T-202 flows into the *reverse osmosit unit* (RO, which in this case also acts as a valve, extracting water continuously: however, it is not identified as a pump) to reduce organic impurities in the same water. The water then flows from the *RO unit* to the third and last stage.

Stage 3 At the third stage, the water from the *RO unit* is divided according to whether standards are met: if the water is clean it will be fed into the distribution system, otherwise it will go to a *backwash tank*, identified by code T-203 and a capacity of one gallon. The water in this tank will then be pumped back to the stage 2 *filtration unit* through

pump P-102.

As mentioned, each stage corresponds to a PLC that controls it, PLC1, PLC2 and PLC3, respectively. Let us briefly see the behavior of each of them:

PLC1 PLC1 checks the level of tank T-201 distinguishing three cases:

- if T-201 reaches the *low setpoint low1* (hardcoded in memory registers), pump **P-101 is opened** and valve **MV-301 is closed**, so that the tank can be filled
- if T-201 reaches *high setpoint high1* (also hardcoded in the memory registers), pump **P-101 is closed**
- in intermediate cases, **PLC1 waits for request from PLC2** to open/close valve MV-301: if a request to open the valve MV-301 arrives, water will flow from T-201 to T-202, otherwise the valve is closed. In both situations, pump P-101 remains closed

PLC2 PLC2 monitors the level of tank T-202, behaving accordingly depending on the level of water in it. Here again there are three cases to consider:

- if the water level reaches the *low setpoint low2* (also hardcoded in the memory registers), PLC2 sends a request to PLC1 via a Modbus channel to **open valve MV-301** in order to flow water from tank T-201 to tank T-202. The transmission channel is implemented by copying a boolean value from a memory register of PLC2 to a corresponding register of PLC1
- if the water level reaches the *high setpoint high2* instead (hardcoded in the memory registers as the previous setpoints), PLC2 sends PLC1 a **close request** for valve MV-301
- In intermediate cases, the valve remains open (closed) while the tank is filling (emptying)

PLC3 PLC3 monitors the level of the T-203 backwash tank, behaving accordingly. Here there are only two cases to consider: if the tank reaches the *low setpoint low3*, pump **P103 is set to off**, so that the backwash tank can be filled: otherwise, if the *high setpoint high3* is reached, pump **P103 is opened** and the entire content of the backwash tank pumped back to the filter unit of T-202.

3.2.2 Scanning of the System and Graph Analysis

3.2.3 Invariants Analysis

3.2.4 Businness Process Analysis

3.2.5 Application

3.2.6 Limitations

Chapter 4

A framework to improve Ceccato et al.'s work.

4.1 Phase 1: Pre-processing

4.2 Phase 2: Graph Analysis

4.3 Phase 3: Invariants Analysis

4.3.1 Invariants Generation

4.4 Phase 4: Businness Process Analysis

4.5 Extra Information on the Physics

Chapter 5

Case study: the iTrust SWaT System

Chapter 6

Our framework at work: reverse engineering of the SWaT system

6.1 Pre-processing

6.2 Graph Analysis

6.2.1 Conjectures About the System

6.3 Invariants Analysis

6.3.1 Actuators Detection

6.3.2 Daikon Analysis and Results Comparing

6.4 Extra information on the Physics

6.5 Businness Process Analysis

Chapter 7

Conclusions

7.1 Discussions

7.2 Guidelines

7.3 Future work

List of Figures

List of Tables

References

[1] *ICS defintion*. URL: https://csrc.nist.gov/glossary/term/industrial_control_system (visited on 2023-04-06).

[2] *What is SCADA?* URL: https://www.inductiveautomation.com/resources/article/what-is-scada (visited on 2023-04-05).

[3] G. M. Makrakis, C. Kolias, G. Kambourakis, C. Rieger, and J. Benjamin. "Industrial and Critical Infrastructure Security: Technical Analysis of Real-Life Security Incidents". In: *IEEE Access* (2021-12-06). DOI: https://dx.doi.org/10.1109/ACCESS.2021.3133348. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9638617 (visited on 2023-04-20).

[4] *PLC defintion*. URL: https://csrc.nist.gov/glossary/term/programmable_logic_controller (visited on 2023-04-06).

[5] W. Bolton. *Programmable Logic Controllers, 6th edition*. Newnes, 2015, pp. 7–9.

[6] M. Ceccato, Y. Driouich, R. Lanotte, M. Lucchese, and M. Merro. "Towards Reverse Engineering of Industrial Physical Processes". In: CPS4CIP@ESORICSAt 2022 (Copenhagen, Denmark, Sept. 30, 2022). 2022.

[7] H. S. G. Pussewalage, P. S. Ranaweera, and V. Oleshchuk. "PLC security and critical infrastructure protection". In: 2013 IEEE 8th International Conference on Industrial and Information Systems (Dec.

2013). 2013. DOI: https://dx.doi.org/10.1109/ICIInfS.2013.6731959.

[8] *Remote Terminal Unit*. URL: https://en.wikipedia.org/wiki/Remote_terminal_unit (visited on 2023-04-08).

[9] *HMI defintion*. URL: https://csrc.nist.gov/glossary/term/human_machine_interface (visited on 2023-04-11).

[10] *What is Modbus and how does it work?* URL: https://www.se.com/us/en/faqs/FA168406/ (visited on 2023-04-13).

[11] *Hacking: Modbus - Cyber security OT/ICS*. URL: https://blog.omarmorando.com/hacking/ot-ics-hacking/hacking-modbus (visited on 2023-04-15).

[12] Modbus.org. "MODBUS/TCP Security". In: pp. 7–8. URL: https://modbus.org/docs/MB-TCP-Security-v21_2018-07-24.pdf (visited on 2023-04-16).

[13] ODVA Inc. "EtherNet/IP - CIP on Ethernet Technology". In: URL: https://www.odva.org/wp-content/uploads/2021/05/PUB00138R7_Tech-Series-EtherNetIP.pdf (visited on 2023-04-18).

[14] *Odva, Inc.* URL: https://www.odva.org (visited on 2023-04-21).

[15] *Introduction to EtherNet/IP Technology*. URL: https://scadahacker.com/library/Documents/ICS_Protocols/Intro%20to%20EthernetIP%20Technology.pdf (visited on 2023-04-19).

[16] *Common Industrial Protocol*. URL: https://www.odva.org/technology-standards/key-technologies/common-industrial-protocol-cip/ (visited on 2022-12-26).

[17] *Common Industrial Protocol*. URL: https://en.wikipedia.org/wiki/Common_Industrial_Protocol (visited on 2023-04-21).

[18] *What is the Common Industrial Protocol (CIP)?* URL: https://www.motioncontroltips.com/what-is-the-common-industrial-protocol-cip/ (visited on 2023-04-21).

[19] *CIP (Common Industrial Protocol): CIP messages, device types, implementation and security in CIP*. URL: https://resources.infosecinstitute.com/topic/cip/ (visited on 2023-04-21).

[20] B. Green, M. Krotofil, and A. Abbasi. "On the Significance of Process Comprehension for Conducting Targeted ICS Attacks". In: Workshop on Cyber-Physical Systems Security and PrivaCy (Nov. 2017). ACM, 2017, pp. 57–67. DOI: https://doi.org/10.1145/3140241.3140254.

[21] A. Keliris and M. Maniatakos. "ICSREF: A Framework for Automated Reverse Engineering of Industrial Control Systems Binaries". In: 26th Annual Network and Distributed System Security Symposium (NDSS) (Feb. 2019). 2019. DOI: https://doi.org/10.48550/arXiv.1812.03478.

[22] Y. Yuan, X. Tang, W. Zhou, W. Pan, X. Li, H. T. Zhang, H. Ding, and J. Goncalves. "Data driven discovery of cyber physical systems". In: *Nature Communications 10* (2019-10-25). URL: https://www.nature.com/articles/s41467-019-12490-1 (visited on 2023-04-20).

[23] C. Feng, V.R. Palleti, A. Mathur, and D. Chana. "A Sysematic Framework to Generate Invariants for Anomaly Detection in Industrial Control Systems". In: NDSS Symposium 2019 (San Diego, CA, USA, Feb. 24–27, 2019). 2019. DOI: https://dx.doi.org/10.14722/ndss.2019.23265. URL: https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_07A-3_Feng_paper.pdf (visited on 2023-04-20).

[24] Singapore University of Technology and Design. *iTrust - Center for Research in Cyber Security*. URL: https://itrust.sutd.edu.sg/ (visited on 2022-12-08).

[25] K. Pal, A. Adepu, and J. Goh. "Effectiveness of Association Rules Mining for Invariants Generation in Cyber-Physical Systems". In: 2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE) (Jan. 2017). 2017. DOI: https://doi.org/

10.1109/HASE.2017.21. URL: https://ieeexplore.ieee.org/document/7911883 (visited on 2023-04-20).

[26] *Association rules mining*. URL: https://en.wikipedia.org/wiki/Association_rule_learning (visited on 2023-04-21).

[27] K. Pal, A. Adepu, and J. Goh. "Cyber-Physical System Discovery: Reverse Engineering Physical Processes". In: 3rd ACM Workshop on Cyber-Physical System Security (Apr. 2017). 2017, pp. 3–14. DOI: https://doi.org/10.1145/3055186.3055195.

[28] *Ceccato et al. reverse engineering prototype tool*. URL: https://github.com/SeUniVr/PLC-RE (visited on 2023-04-23).

[29] Singapore University of Technology and Design. *Secure Water Treatment*. URL: https://itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat/ (visited on 2023-04-23).

[30] R. Lanotte, M. Merro, and A. Munteanu. "Industrial Control Systems Security via Runtime Enforcement". In: *ACM Transactions on Privacy and Security* 26.4 (2023-02), pp. 1–41. DOI: https://doi.org/10.1145/3546579.