Deep Detection Architecture for Security in Industrial Control Systems

Giuseppe Bernieri

Advisor: Prof. Federica Pascucci

Co-Advisor: Prof. Javier Lopez

Department of Engineering

University of Roma Tre

This dissertation is submitted for the degree of

Doctor of Philosophy

March 2018

Abstract

In recent years, the information and communications technology evolution joined the industrial control systems development, leading to new significant enhancements. After the first Supervisory Control And Data Acquisition *monolithic* systems, conceived in the 60s, the *networked* generation was born. Nowadays, the Internet of things and Industry 4.0 paradigms foresee the use of network communication for remote monitoring and control critical infrastructures. Even though the benefits are noticeable, new security challenges concerning industrial facilities arise: typical vulnerabilities of the cyber domain emerged in industrial control systems. The classic cyber-security tools are ineffective, since industrial control systems are designed to operate in standalone or isolated configurations and are characterized by hard real-time and bandwidth constraints. As a consequence, in the last decade, critical infrastructures have experienced a large number of cyber-attacks. Despite the impact of such alarms, the paramount importance of the information traveling in the control system networks and its protection is still underestimated.

In this thesis the protection of industrial control systems for critical infrastructures is addressed. It is supposed that the attackers are able to gain the access to industrial control system network, they are able to bypass the information technology security defense and exploit the system by changing control information. However, they cannot hide their target: the side-effects of cyber-attacks on physical plants reveal the malicious intents. In order to address this problem, this thesis explores attack detection mechanisms that identify attacks by monitoring both the physical system under control and the cyber layer.

This thesis considers both security issues arising at control level and supervisory level. The first one is related to the operational level of the system, the second one is related to the presence of the human in the control loop when emergencies occur. A *defense-in-depth* approach is adopted by developing a deep detection architecture able to combine information from both physical and cyber levels and reduce false alarm rates.

Table of contents

List of figures

List of tables

List of acronyms

ADU	Application Data Unit
AHP	Analytic Hierarchy Process
ARP	Address Resolution Protocol
CI	Critical Infrastructure
CIA	Confidentiality, Integrity, Availability
CPS	Cyber-Physical System
CRO	Confidentiality Restriction Detector
DDS	Deep Detection System
DFA	Deterministic Finite Automata
DoS	Denial of Service
DSS	Decision Support System
E&C	Estimator and Controller
EKF	Extended Kalman Filter
FAD	Finite Automata Detector
FDF	Fault Detection Filter
FDS	Fault Detection System
HMI	Human Machine Interface
ICMP	Internet Control Message Protocol
ICS	Industrial Control System
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
ISA	International Society of Automation
IT	Information Technology
LAN	Local Area Network
LQG	Linear Quadratic Gaussian
LTI	Linear Time-Invariant
MAC	Media Access Control
MBAP	ModBus Application Protocol header
MITM	Man In The Middle
NADE	Network Anomaly Detection Engine
NCS	Networked Control System

NIC	Network Interface Controller
OT	Operational Technology
PDU	Protocol Data Unit
PLC	Programmable Logic Controller
RTU	Remote Terminal Unit
S-IDS	Signature-based Intrusion Detection System
SCADA	Supervisory Control And Data Acquisition
SDN	Software-Defined Network
SIEM	Security Information and Event Management
SRA	Smart Replay Attack
TCP	Transmission Control Protocol
VM	Virtual Machine

Chapter 1

Introduction

1.1 Overview

Industrial Control Systems (ICSs) are computer-based systems devoted to monitor and control physical plants. Nowadays, ICSs represent a wide variety of networked Information Technology (IT) infrastructures connected to the physical domain: to this end, they can be regarded as Cyber-Physical Systems (CPS).

Modern ICSs have a multi-layer structure and are arranged as a set of network agents, such as sensors, actuators, control units (Programmable Logic Controller - PLC), remote communication devices (Remote Terminal Unit - RTU), etc. The targets of ICSs are to maintain safe operational goals by avoiding or mitigating undesirable behavior, to meet the production demands by keeping process values within certain thresholds, and to maximize production.

Some control applications are safety-critical: their failure may significantly hamper the economy, the environment or may even lead to loss of human life. For instance, ICSs for Critical Infrastructures (CIs) (e.g., electric power distribution, oil and natural gas distribution, water and waste-water treatment, and transportation systems) perform control on the main national assets. The disruption of these control systems could have a significant impact on public health, safety and lead to large economic losses.

Thus, security is crucial to guarantee safe operations in CIs: the research community has acknowledged the importance of addressing the challenge of designing secure CPS [3]. Although ICSs have not been considered as target of cyber-attacks in the past, they are now at a higher risk due to their exposure to the network and due to motivated and highly-skilled attackers.

Many computer-based incidents in control systems have been recorded in the last few years: computer accidents can be caused by any unexpected software error, non-targeted, and targeted attacks, which represent the major threats. Targeted attacks, indeed, occur when the adversary know the control system and the attack strategy aims at damaging the physical system under control. Cyber-attacks are a natural evolution of physical attacks: they have the same effects, but are cheaper, less risky for the attacker, and easier to replicate and coordinate.

A well known targeted attack to ICSs for CIs is the one to Maroochy Shire Council sewage control system in Queensland [4]. There are many other reported targeted attacks [5, 6], however, no other attack has demonstrated the threats that control systems are subject to as well as the famous Stuxnet [7]. Stuxnet disclosed to the world that there are groups with the motivation and skills to set up sophisticated computer-based attacks to CIs. The ultimate goal of Stuxnet is to sabotage the attacked asset by reprogramming PLCs to operate out of their specified thresholds. Many security companies, including Symantec and Kaspersky claimed that Stuxnet is the most sophisticated attack they have ever analyzed: Stuxnet uses four zero-day exploits, a Windows rootkit, the first known PLC rootkit, antivirus evasion techniques, peer-to-peer updates, and stolen certificates from trusted certification authorities. More recently, an extremely dangerous malware targeting Schneider Electric's Triconex safety instrumented system (SIS) has been discovered [8].

Therefore, it is not surprising that the security of ICSs for CIs has become an active research area in recent years. However, it is worth noticing that securing an ICSs is different compared to traditional IT security. Software patching and frequent updates are not well suited for control systems: upgrading a system may require months of advance in planning how to take the system offline, which it is not economically convenient. ICSs are composed by legacy systems: several research efforts have tried to provide mechanisms to protect data integrity and confidentiality, however these can be considered short-term solutions. Not all operational differences are more severe in control systems than in traditional IT systems. ICSs exhibit comparatively simpler network dynamics: they use fixed topology and user population, regular communication patterns, and a limited number of protocols. Therefore, implementing network Intrusion Detection Systems (IDSs), such as anomaly detection is easier than in traditional enterprise systems.

There are several industrial and government-led efforts to improve the security of ICSs. These efforts increase the awareness on ICS security issues, providing security design guidelines for control systems operators and IT security officers, and introducing basic security mechanism for prevention. These recommendations help in maintaining the operational level when the system is under attack, however the security of ICSs involves additional challenges in control layer.

1.2 Scope of the Work

These thesis explores and addresses new research problems that arise in an ICS under attack. The research efforts carried out are related to the analysis of threats affecting ICS specific in the context of CIs.

When adversaries bypass basic security defenses, they are able to have impact on the physical domain. Traditional computer security focuses on how to protect information. Here, a novel perspective is adopted, considering how attacks affect estimation, control and monitoring algorithms, how they affect the plant, and how they affect the decision made by the human operators.

In this thesis it is supposed that the attackers are able to gain the access to the Control Zone [2] of an ICS for CI, they are able to bypass the IT security defense and exploit the system by changing control information. However, they cannot hide their target: the side-effects of cyber-attacks on physical plants reveal the malicious intents. Corrupted sensor or controller data will not match the ones expected by the control or the supervisory system. In order to address this problem, this thesis explores attack detection mechanisms that identify attacks by monitoring both the physical system under control and the cyber layer.

This thesis considers both security issue arising at control level and supervisory level. The first one is related to the operational level of the system, the second one is related to the presence of the human in the control loop when emergencies occur. A *defense-in-depth* approach is adopted by developing a deep detection architecture able to combine information from both physical and cyber level and reduce false alarm rates.

1.3 Contributions

The proposed thesis is based on several contributions. The first one is related to the design of a security architecture for ICSs. The proposed architecture fills the gap between computer science and control theoretic approaches. The physical layers connected to ICS, indeed, are prone to disrupt when facing cyber-attacks. Considering the modules of the proposed architecture, we focus on the development of a practical framework to compare information about physical faults and cyber-attacks. Moreover, concerning the interface between the operator and the system, we proposed an effective Decision Support System (DSS). Finally, to validate the proposed approaches, several tools have been developed: the HYDRA testbed, a tool for emulating a water distribution system, has been implemented, as well as simulation tools. This contributions are further detailed in the followings.

1.3.1 Design of Deep Detection Architecture for Industrial Control System Security

The Deep Detection Architecture is a modular framework conceived to provide security in the Control Zone [2] and can be implemented in a distributed fashion. It is composed by three main levels, the *Deep Detection System* (DDS), the *Mimepot*, and the *Decision Support System* (DSS). These three levels are designed to cope with different actors.

The *Deep Detection System* represents the core of the architecture and is devoted to analyze the communication traffic and the operations of physical plant. It is able to set alerts by comparing information from both the cyber and the physical domains.

The *Decision Support System* exploits the alerts provided by the DDS and the some a priori knowledge to propose different solutions to the human operators.

The *Mimepot* is devised to define a resilient control system according to the *security-by-detection* paradigm. It is implemented as an advanced honeypot. Once an attack is on, the malicious network traffic is redirected to the Mimepot by an advanced Software Defined Network (SDN) controller. The Mimepot emulates the operation of the physical system to collect data from the attacker. It allows forensic investigations that are fundamental in CPS: when the system knowledge of the attacker, indeed, is extensive, he/she has the possibility to hide its malicious actions.

1.3.2 Development of Framework to Compare Physical and Cyber Information

The main novelty introduced by the proposed architecture is a security framework able to compare and combine the information gathered from cyber and physical domains. This is represented by the modules in the DDS, specifically, the *Fault Detection* module and the *Network Anomaly Detection Engine*.

The first one is devoted to identify the side-effects of cyber-attacks on the physical plants. To this end, it is implemented exploiting techniques borrowed from fault detection field. The main novelty of the approach with respect to the literature is represented by the use of nonlinear model to detect misbehaviors. The nonlinear fault detection filter and the extended Kalman filter are adopted to produce expected measurements that are compared with the ones provided by the plant. The residual is analyzed to detect faults or attacks.

The *Network Anomaly Detection Engine* analyzes the communication traffic. Most of the anomaly detection systems proposed in the literature are based on the analysis of communication parameters (e.g., network throughput, number of packets, etc.) with respect to a nominal behavior. Here, a different perspective is adopted: the payload of the packets is further analyzed and compared with the standard operation cycle.

1.3.3 Implementation of a Decision Support System for the Operator

The proposed DSS is based on the data forwarded by the DDS and the a priori knowledge of experts. These two elements are merged using Analytic Hierarchy Process (AHP). This methodology turns out to be more efficient with respect to the one proposed in the literature since it allows to select the most important alternative as linear combination of the absolute importances according to the different and heterogeneous criteria. Thus, the issues related to the interdependencies are addressed during the definition of the a priori knowledge, that is performed once.

1.3.4 Validation Tools

Several validation tools have been developed during the doctorate to prove the effectiveness of the proposed security architecture. The majority of the approaches presented in the literature, indeed, has not been largely validated in a real scenario. Since tests can be hardly performed directly on CIs, simulated and emulated environments have been designed and implemented. The HYDRA testbed emulates a water distribution system. At physical level, it is composed of 7 tanks; at cyber level, it is composed of the control system, the fault detection system, the signature-based IDS, and the Human Machine Interface (HMI). The HYDRA testbed can be used to emulate different scenarios (e.g., reproducing the water consumption in a small city over a day). The simulation tool Mininet has been used to test the communication system in a virtual environment: a preliminary version of the Mimepot, the SDN controller and the fault detection system have been implemented using this tool. Finally, some simulation tools have been developed in the Matlab/Simulink environment for rapid prototyping.

1.3.5 List of Publications

In this section, the publications relevant to the thesis development are listed:

J-1 G. Bernieri, E. E. Miciolino, R. Setola, F. Pascucci. “Monitoring System Reaction in Cyber-Physical Testbed under Cyber-Attacks”. Special Issue on Critical Systems Modeling and Security, Computers & Electrical Engineering, Elsevier.

J-2 G. Bernieri, E. E. Miciolino, S. Panzieri, F. Pascucci, M. M. Polycarpou, R. Setola. “Fault Diagnosis and Network Anomaly Detection in Water Infrastructures”. Special Issue on Cyber-Physical Systems Security and Privacy, Design & Test, IEEE.

C-1 G. Bernieri, E. E. Miciolino, F. Pascucci, R. Setola. “Communications Network Analysis in a SCADA System Testbed Under Cyber-Attacks”. 23st Telecommunications Forum, TELFOR 2015, November 24-26, 2015, Belgrade, Serbia.

C-2 G. Bernieri, F. Del Moro, L. Faramondi, F. Pascucci. “A Testbed for Integrated Fauld Diagnosis and Cyber Security”. 3rd International Conference on Control, Decision and Information Technologies (CoDIT), Malta, 2016.

C-3 G. Bernieri, S. Damiani, F. Del Moro, L. Faramondi, F. Pascucci, F. Tambone. “A Multiple-Criteria Decision Making Method as Support for Critical Infrastructure Protection and Intrusion Detection System”. IEEE IECON 2016, 42nd Annual Conference of the IEEE Industrial Electronics Society.

C-4 G. Bernieri, F. Pascucci, J. Lopez. “Network Anomaly Detection in Critical Infrastructure Based on Mininet Network Simulator”. First Italian Conference on Cyber Security (ITA-SEC), Venice, Italy 2017.

T-1 G. Bernieri, E. E. Miciolino, F. Pascucci, R. Setola. “Analysis of Critical Infrastructure Testbed Under Integrity Cyber-Attack” (Poster). Automatica 2016, Roma 5-7 Settembre 2016.

T-2 G. Bernieri, N. Vastarella, F. Pascucci. “Networked Control System Simulation Using Mininet For Cyber-Physical Security” (Poster). Automatica 2017, Milano 11-13 Settembre 2017.

S-1 G. Bernieri, F. Battisti, M. Carli, M. Lopardo, F. Pascucci. “Detecting integrity attacks in IoT-based Cyber Physical Systems: a case study on Hydra testbed”. Submitted to Global Internet of Things Summit (GIoTS) 2018.

1.4 Outline of the Thesis

The remainder of the thesis is organized as follows:

- Chapter 2 presents the related works on the topic covered during the doctorate. Specifically, the ICS for CI and related security issues are introduced.
- In Chapter 3, the design of the proposed architecture for the DDS is sketched. All the modules are described: the novel theoretical approaches adopted in the design of the architecture are explained.

- In Chapter 4, the tools developed during the doctorate are introduced. The modeling techniques for cyber-attacks according to control theoretic perspective are proposed. Concerning the emulated environment, the hardware and software design and the implementation of the testbed HYDRA is detailed. The simulated environment based on Mininet is also presented.

- In Chapter 5, the experimental results on the DDS combining fault detection and intrusion detection modules are reported and discussed. The experimental results on the DSS based on AHP are also presented.

- Some conclusive remarks and suggestions for future development are drawn in Chapter 6.

The interested reader can find some introductory material on the Modbus protocol and the testbed already developed during the FACIES project in the Appendix.

Chapter 2

State of the Art

This chapter introduces the literature on security for ICSs. The solutions proposed in the literature are presented and analyzed to set the work of this thesis against the state of the art.

2.1 Background

During the last two decades, technological evolution has extended classical ICS domain towards the Networked Control System (NCS) one, where monitoring and control routines are managed through communication channels. Modern Supervisory Control And Data Acquisition (SCADA) systems take advantage of communication channels to distribute monitoring and control activities over large geographical areas. This opening towards wide geographical communications has improved the control and monitoring scopes of industrial systems and, at the same time, has also expanded the spectrum of potential threats to which they are exposed. According to this perspective, the analysis carried out is based: cyber security plays a topical role in the management specifications of modern industrial systems.

According to the literature on cyber security, there are three main properties of data and IT services: *Confidentiality, Integrity, and Availability (CIA)* [9].

- *Confidentiality* is the characteristic of maintaining sensitive information or resources reserved. The need to keep certain types of information secret is absolutely necessary in sensitive sectors such as industry and government, but it is also necessary to protect the digital identities of today's Internet users. One of the access control mechanisms that allows data privacy to be preserved is encryption. For the specific industrial sector, the confidentiality refers to restricted information related to the plant that unauthorized person should not know.

- *Integrity* refers to data (cyber integrity) and to resources (physical integrity) reliability. This feature shows the need to prevent improper or unauthorized changes by malicious actors. Integrity is intended to both the content of the information and its origin: if the source is not authenticated, the integrity characteristic is missing. Considering the ICS, the integrity is related to data generated by sensors, actuators, and workstations. It is important that the data integrity remains preserved during the transmission through communication channels.

- *Availability* refers to the ability to use the information or resource of a system. If a system is not available, it cannot provide any services to users. The issue of resource availability is relevant to security: a malicious actor could intentionally organize to deny access to data or a service by making it unavailable. In the industrial scenario, the availability regards the effective operation of any industrial plant component. When a sensor deployed to the field becomes unreachable it represents an availability problem.

For the traditional IT community, the importance order of the characteristics described above follows the CIA paradigm, where confidentiality is the most important, followed by integrity and availability. On the other hand, for industrial systems, the paradigm is reversed to AIC, where availability plays the role of main feature. The reason for this is that the non-availability of resources in an industrial system could cause serious damage to people and/or the environment [10].

The specific ICS security is capturing noteworthy relevance in scientific literature and it is hard to manage: while for the IT the mission is to protect data, for the Operational Technology (OT) is to protect plants and critical assets. For this reason, it is not easy to study and implement security solutions because these should be tested in real ICS scenarios where problems can arise.

According to [11], hostile governments, terrorist groups, disgruntled employees, malicious intruders, and illicit or accidental actions by insiders represent sources of threats for control systems. When considering a ICS, a malicious event could lead to incident, for instances:

- A delay or interruption of the network data flow may downgrade the operational level of the plants;

- Unauthorized changes to licit data integrity (such as actuator commands, sensor reads, alarm thresholds) may results in damages affecting the industrial equipment, human operators, and environment;

- Incorrect information sent to human operators or system components may cause unauthorized changes or incorrect actions to provoke negative side-effects;
- Industrial control software or configuration settings illicitly modified (e.g. using malware) may lead to loss of control for the systems;
- Interference with the protection or safety systems operation may expose to danger the industrial equipment and human life.

Considering the possible issues identified above, it is possible to highlight which security objectives should be taken into account for industrial networked system scenarios:

Restricting logical access to the ICS network and network activity. This approach envisaged the use of unidirectional gateways, a demilitarized zone network architecture with firewalls to prevent network traffic from passing directly between the corporate and ICS networks. To this end, different authentication mechanisms and credentials for users of the corporate and ICS networks need to be adopted. A multiple layer can be implemented for the network topology of the ICS, to guarantee that the most critical communications occur in the most secure and reliable layer.

Restricting physical access to the ICS network and devices. Physical access control policies (e.g., locks, card readers, and guards) are used to prevent that unauthorized physical access to components could cause serious disruption of the ICS functionality.

Protecting individual ICS components from exploitation. The protection of ICS components includes some good practices: the deployment of security patches, after testing them under field conditions, the disabling of unused ports and services; the control of ICS user privileges according to person's role; the tracking and monitoring of audit trails; the use of security controls such as antivirus software and file integrity checking software where technically feasible to prevent, detect, and mitigate malware.

Restricting unauthorized modification of data. This issue includes data forwarded across the network boundaries and data at rest.

Detecting security events and incidents. The *security-by-detection* foresees the development of security events, which have not yet escalated into incidents: in this way the defenders break the attack chain before adversaries attain their objectives. This includes the capability to detect failures in ICS components, unavailable services, and exhausted resources that are important to provide proper and safe functioning of the ICS.

Maintaining functionality during adverse conditions. The business continuity needs to be granted by redundancy. Moreover, the ICS should implement degradation from fully automatic to fully super-visioned process: in this way, if the ICS operational level is downgraded, the continuity is guaranteed by the commitment of the operators.

Restoring the system after an incident. When the incidents are unavoidable, their impact needs to be reduced by applying response plans: one of the fundamental characteristic of a security program is how quickly the system can be recovered after an incident has occurred.

Attacks previously unknown can targeting ICSs more than other IT systems: cyber issues can affect the physical layer since ICSs are complex systems. As a consequence, it is not possible to apply the traditional IT security measures which are designed to react to already known attacks. Therefore, a more complex defense-in-depth approach is necessary: besides the first layer of conventional IT security tools that aim at preventing attacks, the design of advanced monitoring tools that aim at detecting and reacting to attacks that are able to breach the first layer is mandatory [12].

2.2 Cyber-Physical Attack Space and Taxonomy

As noticed in [1], cyber-attacks over CPS pose different security challenges. Thus the design of a control system resilient to cyber-attacks show different characteristics with respect to design a control system tolerant to faults. Cyber-attacks, indeed, may be executed over a certain number of attack points in a coordinated way. As opposed, faults are unexpected events that affect the physical system: in this way, it is assumed that they are not colluding and are not performed in a coordinated fashion. Moreover, cyber-attacks have specific malicious objectives to achieve, on the other hand, faults do not have objectives.

According to [9, 13] a resilient control system is designed by considering three actions: prevention, detection and mitigation. These actions need to consider both faults and cyber-attacks, addressing these two category of issues in different ways. Concerning prevention, detection, and mitigation of faults it is fulfilled by maintenance, on-line monitoring, and timely repair of the physical components of the system, respectively. The prevention, detection, and mitigation of cyber-attacks encompass both the physical and the cyber layers of a system: to this end, approaches to securing CPS consider techniques adopted in the

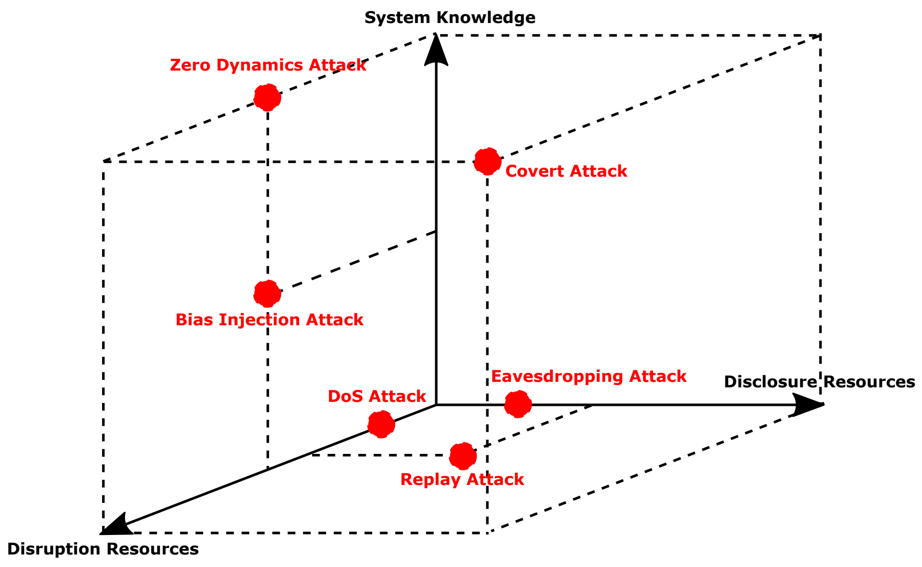


Fig. 2.1 The Cyber–Physical attack space [1].

physical domain (e.g., fault detection) and in the cyber space (e.g., encryption and improved algorithms [14]). Moreover, the design of the security system for a CPS needs to address large number of threats and requires attack impact analysis and the use of risk assessment methods [15].

2.2.1 Cyber-Physical Attack Space

As already state, both cyber and physical layers need to be considered addressing the security issue in CPS. To get insights on a cyber-attacks over a CPS, it is fundamental to evaluate the resources and the overall knowledge of the malicious actors. To this end, in [16], the authors proposed to classify the cyber threats according to the *attack space* shown in Fig. 2.1.

The attack space is characterized by three dimensions:

- *System knowledge*: represents the a priori system knowledge gained by the adversary. The a priori model knowledge can be used by the adversary to set up complex attacks, harder to be detected and arising severe consequences.

- *Disclosure resources*: are the assets discovered by the attacker. Disclosure resources operations (e.g., data sniffing) enable the adversary to obtain sensitive information about the system during the attack by violating data confidentiality. However, disclosure resources (e.g., the eavesdropping attacks) alone cannot disrupt the system operation.

- *Disruption resources*: are the assets compromised by the adversary. Disruption resources, such as data *spoofers* and *jammers*, impact the system operation. For instance, the system operation may be disrupted when data integrity or availability properties are violated. During Denial of Service (DoS) attack, the data required for correctly operating the system is made unavailable. The *Stuxnet* malware had resources to record and manipulate data in SCADA network.

The proposed three dimensions are relevant from a control theory perspective and allow to classify several attack scenarios described in the literature.

2.2.2 Cyber-Attack Taxonomy

Cyber-attacks can be further classified by considering the features compromised by the adversary, such as availability, integrity, and confidentiality. A complete taxonomy of cyber-attacks can be found in [17].

Availability. The objective of availability attacks is to disrupt services or resources of a system. The DoS attack represents the classical example for this category. When addressing ICS, a DoS attack exploitation can prevent the actuator and/or sensor data from reaching their respective destinations, thus compromising the whole system.

In [18, 19] the availability attack has been analyzed considering a resource-constrained adversary having information about the whole system. Particularly, the authors considered DoS attacks in which the adversary could tamper the communication channels preventing measurement and actuator data from reaching their destination. In [18], DoS attack without a priori model knowledge is addressed.

Integrity. Attacks targeting the integrity are conceived to manipulate network data, such as sensor readings and actuator commands. Several types of methodologies belong to this kind of attack. The deception attack, also known as data modification, is the general term for all the methodologies that intercept and modify the content of data during the transmission. For instance, replay attacks are deception attacks that record network traffic and then reuse it identically at a later stage.

Integrity attacks are discussed in [14]: the authors propose an encryption and predictive control scheme to prevent and mitigate deception attacks on control systems. Replay attacks have been deeply analyzed in [20]: the authors consider attacks against sensors and detection methodologies are proposed. In this attack scenario the adversary does not have any model knowledge but he/she is able to access and corrupt the sensor data, thus having disclosure and disruptive resources.

Another class of integrity attacks is represented by the false-data injection ones. For this type of attack, the malicious actor injects false-data in the communication channels of a networked system. Particular case of false-data injection is the bias injection attack, where the adversaries goal is to inject a constant bias in the system without being detected. In [21], the author characterizes the set of attack policies for covert (undetectable) false-data injection attacks with detailed model knowledge and full access to all sensor and actuator channels. In [22], the set of undetectable false-data injection attacks for omniscient adversaries is introduced: these threats, named zero-dynamics attacks, compromise only a subset of existing sensors and actuators. Concerning multi-agent systems, optimal adversary policies for data injection using full model knowledge and state information are proposed in [23]. In [24], the authors consider the detection and mitigation of false-data injection attacks on linear information dissemination algorithms over communication networks. In these attack scenarios confidentiality is violated, as the adversary gains access to either measurement and actuator data or full-state information. The impact of false-data injection attacks has also been considered in the literature. In [25], the reachability methods are applied to dynamic nonlinear model of power networks to analyze the impact of a false-data injection attack on a two-area power network. In [26], the authors propose methods to approximate the reachable set of states for stealthy adversaries, considering a linear networked control systems under false-data injection attacks. It is worth noticing that the hazards related to an integrity attack depends on the level of knowledge about the system gained by the malicious actor. In [27], it is analyzed a power network attacked by an adversary having complete knowledge about the model of the system. When the system knowledge is extensive, the adversary has the possibility to hide its malicious actions from the security mechanism implemented on the system under attack, so the integrity attack becomes a stealthy one. In [28], stealthy attacks with limited resources are considered and an improved detection methods is proposed. In [29], a methodology to identify the minimum number of sensors required for stealthy attacks is presented. Moreover, a corresponding measurement security metric for studying sets of vulnerable sensors is developed. Consequences of integrity attacks have also been analyzed in [16, 30, 31]. Specifically, in [30], the authors analyze attack policies with limited model knowledge, considering a power system control software: it is shown that attacks are stealthy and can induce the erroneous belief that the system is at an unsafe state. Data injection attacks on dynamic control systems are also considered. In [3], integrity ad availability attacks that compromise measurements and control input are considered: the physical impact of the attacks is further investigated. In [32], several attacks on the Tennessee-Eastman process control system are simulated and evaluated in terms of detectability.

Confidentiality. Attack class targeting confidentiality is exclusively related to the resources disclosure. As already mentioned above, confidentiality is crucial for IT security. A methodology concerning this class is the eavesdropping, also known as sniffing attack. This exploit allows the interception of data exchange between nodes on a victim network. From the point of view of networked control systems, it is possible to identify this attack as the first step towards more complex malicious activities.

The pervasive connectivity of Internet of Things (IoT)-based systems demands for new mechanisms to protect against industrial espionage and privacy of customers and employees. Hence, confidentiality of data and configuration of modern ICSs is an important security requirement to be considered [33].

Most of the recent work on cyber-security of control systems has considered scenarios where the adversary owns access to a large set of resources and knowledge, thus being placed far from the origin of the attack space. A large part of the attack space has not been explored yet. In particular, the class of detectable attacks that do not trigger conventional alarms has yet to be covered in depth.

2.3 Industrial Control System Security: the Computer Science Perspective

In this section, techniques that are commonly used to defend and secure ICSs from the Computer Science point of view are discussed. This includes analysis of tools and standards conceived for the security of industrial networks. The configurations and the protocols applied have a critical impact on the security of ICSs. Communication protocols implement specific security characteristics and concerning the old protocols (e.g., Modbus) that are still used today, the required security definitions may be missing. The standard application of IT security practices is important to secure the industrial production assets. According to the basic guidelines, the first issue to consider is the security education of industrial operators and users: the security awareness and the security policies knowledge enforce radically the defense against cyber threats. Subsequently, the use of tools and practices permit the implementation of active or passive defensive methodologies able to cope with potential malicious activities. In order to consider research and development activities for the cyber security of ICS, it is necessary to analyze and use defensive strategies, tools, and methodologies but it is also important to investigate the offensive point of view. This because the development of complex security methodologies is directly connected to the threats sophistication advance.

2.3.1 The Intrusion Detection Problem

The intrusion detection refers to the passive process of monitoring events occurring on computer networks and analyzing them for signs of possible incidents, which are violations or imminent threats to computer security policies. The main techniques can be divided into *signature-based* and *anomaly-based* intrusion detection. The signature-based detection, also called misuse detection, uses well-known attack patterns to identify intrusions. Conversely, the anomaly-based detection determines possible deviations from established patterns of normal system usage [34].

The use of IDS to protect ICSs involves the semantic level of operations by inspecting packet payloads. This highlights that IDS focusing on the depth inspection of ICS networks requires expert knowledge for the specific protocol under analysis. Consequently, the protection of ICS networks requires intrusion detection technologies that can deliver both traditional IT security measures and more specific measures tailored to the unique features of ICSs [12].

Relevant existing non-commercial IDS tools are:

- *Snort* is a well-known signature-based IDS that analyze communication traffic to search malicious activities by using a set of rules. It is able to send alerts to operators. The signature database can be extended using new rules specific for industrial network protocols [35].

- *Bro* provides a platform for network traffic analysis and intrusion detection, with a particular focus on semantic security monitoring. Respect to classical IDS, with this tool it is possible to provide a flexible framework to the cyber security community. Bro can work as a rule-based or anomaly-based IDS [36].

- *Suricata* is an open source IDS able to deal with real time network security monitoring. This tool inspects traffic using an extensive rules set [37].

In the literature, efforts in [38–41] have been made towards the use of signature-based IDS, while also several anomaly-based detection systems have been investigated [42, 43].

Concerning industrial control networks intrusion detection, specific works have been provided [44, 45]. In [46] the authors discuss the history of research in intrusion detection techniques and introduce two basic detection approaches: signature detection and anomaly detection for SCADA systems. Intrusion detection methodologies able to detect complex attacks against SCADA systems networks are presented in [47, 48]. In [49], it is proposed a distributed architecture that aims to secure CI communication networks using IDS and a customizable flow monitor. Current research directions in protecting CI and CPS intrusion using detection systems are described in [50, 51]. In [52], the authors provide details on how Snort rules can detect DoS, Command Injections, Response Injections, and System Reconnaissance vulnerabilities using a Modbus RTU/ASCII Snort implementation.

The development of new detection methodologies in the contexts of industrial scenarios, represents a captivating challenge for cyber security research community. An extensive work for the anomaly detection in SCADA systems is provided in [53]. In [54] a novel domain-aware anomaly detection system that detects irregular changes in Modbus/TCP SCADA control register values is presented. In [55], a novel approach to specification-based intrusion detection in the field of networked control systems is presented. The development of a sequence-aware IDS is proposed in [56], where a reference architecture and tests on real ICS traffic samples are provided. Processes-related threats are investigated in [57] to address detection in CI networks. In [58], an IDS for SCADA system network is presented. Here, the OCSVM (One-Class Support Vector Machine) mechanism is exploited to provide detection of anomalies in the real time system. In [59] a behavior-based IDS for industrial IEC 61850 protocol is presented.

Interesting model-based intrusion detection techniques are devoted to use Deterministic Finite Automata (DFA) for anomaly detection. In [60], an approach based on model-based intrusion detection is presented. The model proposed is built using DFA observing Modbus/TCP traffic. In [61], the authors propose a new approach that addresses how to model the network patterns of multiplexed ICS streams using DFA for anomaly detection. In [62], the authors propose to complement DFA and Discrete Time Markov Chain Modbus/TCP modeling problem for intrusion detection in SCADA systems with specification approaches. In [63], a runtime-monitoring technology that provides additional protection of the ICS against cyber-attacks and verification of the system efficiency is presented. The authors consider the specific semantic attacks threat and design a passive monitoring solution. The developed methodology has been validated using Arduino boards equipped with Ethernet modules.

More recently, the evolving landscape of threats in ICS networks is described in [64]. In [65] a distributed IDS for SCADA systems is presented. In [66], an innovative IDS has been conceived to detect anomalies that have distributed impact on the plant processes. It has been tested on the *Secure Water Treatment* testbed [67].

2.3.2 Defensive Tools

Besides IDS, more general IT security tools are also relevant for industrial network scenario [68]. Important tools for defensive cyber analysis are:

Firewalls [69]. These systems represent the perimeter defense components of IT networks that can perform connection functions between two or more network segments, thus ensuring protection in terms of network security. In the field of ICS, firewalls are most often deployed between the ICS network and the corporate network. Properly configured, they are able to restrict undesired accesses to and from control system devices. Firewalls can also improve network responsiveness by removing non-essential traffic. When properly designed, configured, and maintained, dedicated hardware firewalls can contribute significantly to increasing the ICS security.

Security Onion [70]. This is a Linux based distribution that provide a pre-configured set of open source security solutions for vulnerability assessment purposes.

Wireshark [71]. It is a free and widely used packet sniffer and analyzer. Wireshark is used by operators to inspect network traffic in real-time or offline.

Honeypots [72] are systems used to attract and lure attackers into interaction with them. Honeypots are usually more vulnerable respect to real infrastructures and cyber-attacks against these systems do not affect the normal functioning of real systems. One of the first research examples of specific ICS Honeypot developed is described in [73]. More recent works related to ICS networks honeypots are presented in [74, 75]. Moreover, within the Honeynet Project, a low interactive server side ICS honeypot, namely *Conpot*, has been designed to be easy to deploy and extend [76].

2.3.3 Offensive Tools

As already stated, in the context of ICS, it is mandatory to parallel analyze defensive and offensive cyber strategies. This because the security community needs to be ready for incoming new cyber threats. Useful tools for vulnerability assessment and penetration testing for industrial control networks are:

Kali [77]. It is an advanced Linux distribution used for penetration testing, ethical hacking and network security assessments. Kali offers to users easy access to a wide range of security tools already installed.

Nmap [78]. It is a free and open source utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime. *Nmap* can be used to maliciously map networks.

Nping [79]. It is an open source tool for network packet generation, response analysis and response time measurement. *Nping* can be used to craft network packets for a wide range of protocols, allowing users full control over protocol headers. Those packets can be used to create specific attacks such as DoS by sending an elevated number of non-authorized flows.

Scapy [80]. This is a powerful interactive python framework devoted to packet manipulation activities. It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies.

Ettercap [81]. It is an open source suite conceived for Man In The Middle (MITM) attacks. Concerning packet manipulation, specific filters have been designed and compiled with *Etterfilter*, a tool of the *Ettercap* suite, which consists of a filter compiler able to manipulate the packets on the network with a content filtering engine.

Social Engineering Toolkit [82]. It is an open-source penetration testing framework designed for social engineering attacks such as classical *phishing* or more advanced techniques like *spear phishing*.

Metasploit [83]. It is a framework devoted to developing and executing exploits to damage a remote machines.

2.3.4 Industrial Security Standardization and Guidelines

The IEC 62443 [2] series of standards are the only specifically dedicated to ICS networks security. Produced by the original International Society of Automation (ISA)-99 committee, the IEC 62443 standards provides a coherent security approach according to four general categories of documents (Fig. 2.2) addressed to the ICS security operators. The intention of the committee is to improve the confidentiality, integrity, and availability of components and

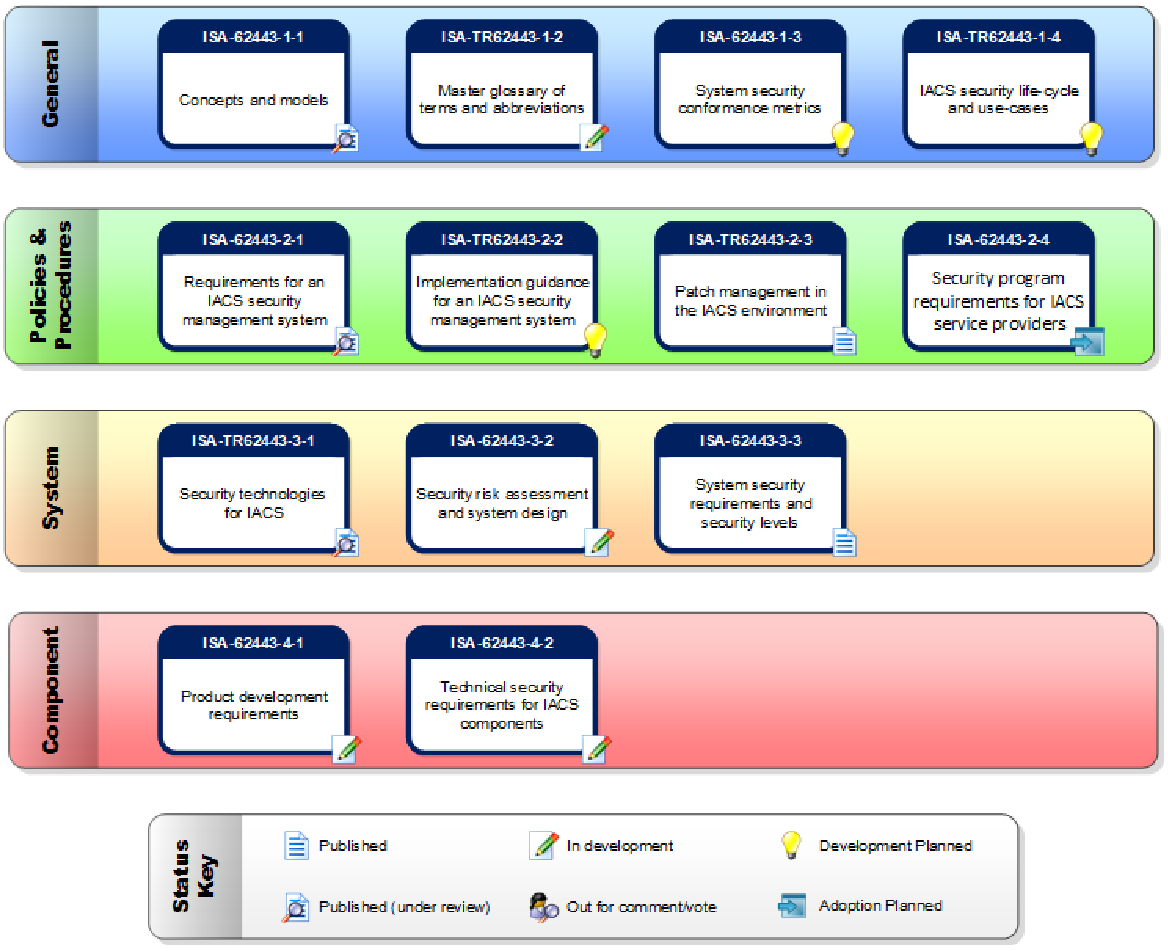

General
ISA-62443-1-1
Concepts and models
ISA-TR62443-1-2
Master glossary of terms and abbreviations
ISA-62443-1-3
System security conformance metrics
ISA-TR62443-1-4
IACS security life-cycle and use-cases
Policies & Procedures
ISA-62443-2-1
Requirements for an IACS security management system
ISA-TR62443-2-2
Implementation guidance for an IACS security management system
ISA-TR62443-2-3
Patch management in the IACS environment
ISA-62443-2-4
Security program requirements for IACS service providers
System
ISA-TR62443-3-1
Security technologies for IACS
ISA-62443-3-2
Security risk assessment and system design
ISA-62443-3-3
System security requirements and security levels
Component
ISA-62443-4-1
Product development requirements
ISA-62443-4-2
Technical security requirements for IACS components
Status Key
Published
In development
Development Planned
Published (under review)
Out for comment/vote
Adoption Planned

Fig. 2.2 IEC 62443 standard work progress [2].

systems used for automation or control and provides criteria for implementing secure control systems. The compliance with the IEC 62443 guidelines can improve ICS security, helping to identify vulnerabilities. This approach could reduce the risk associated to control systems failures. Industrial scenario benefits from the standardization of security implementations to apply specific network protocols. A global standard on the use of role based security combined to strong authentication process would support a cross-protocol interoperability.

According to the industrial security standards, the IEC-62443-3-2 [2] addresses how to define security assurance levels by using *zones* and *conduits*. The model has been developed by the ISA-99 Committee for manufacturing and control systems security and it represents the baseline for the ICS security reference architecture.

The zones are defined by dividing the ICS network into segments including systems having similar requirements or performing related functions. The key idea when using zones is to split the network into smaller, more focused environments, where security controls can

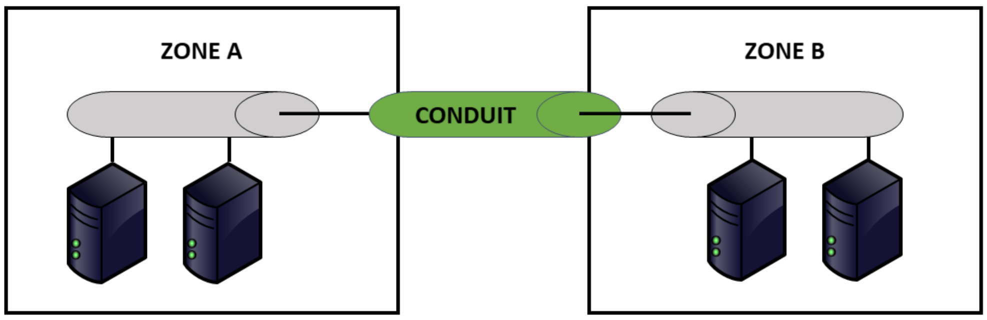


Fig. 2.3 Conduit representation.

be consistently applied. A zone is formally defined as a logical network segment within a networking environment that has a well-defined security perimeter. Applications and systems of ICS networks can be placed into different zones based on specific characteristics, such as risk levels, business criticality, processes safeguarding. Despite segmentation, each zone remains vulnerable and same threats can affect multiple zones. To this end, it is mandatory to define specific security and detection measures within each zone to grant protection from cyber threats.

According to zones paradigm, defense mechanisms are easier to apply: each zone, indeed, satisfies it own particular security requirements. Moreover, segmentation and segregation approaches can represent effective solutions for cyber security when adopted on ICSs. The network segmentation and segregation purpose, indeed, is to minimize the access to sensitive data for employees and systems that not require such information (e.g., an account manager works in the Corporate zone, he has no need to have access to specific Control zone data). The strength of segregation and segmentation is to ensure zones operation without generating contrasts between uncorrelated operations. However, there is the need of a connection bridge between different zones: this bridge is formally named as *conduit*. According to IEC-62443-3-3 [2], a conduit is defined as a logical group of communication channels, connecting two or more zones, that share common security requirements. A graphical representation of conduit is sketched in Fig. 2.3, where network nodes of two different zones communicate using a conduit.

In Fig. 2.4, zones and levels defined by the standard are described. Three zones and six operational levels are identified:

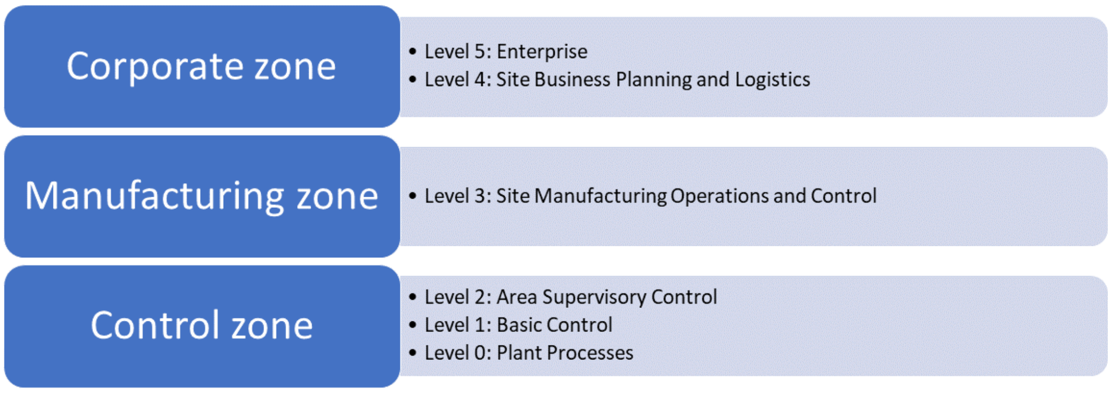

Corporate zone
• Level 5: Enterprise
• Level 4: Site Business Planning and Logistics
Manufacturing zone
• Level 3: Site Manufacturing Operations and Control
Control zone
• Level 2: Area Supervisory Control
• Level 1: Basic Control
• Level 0: Plant Processes

Fig. 2.4 ISA99 security levels and zones.

Corporate Zone: Levels 5 and 4. In Level 5 all the classic IT infrastructures are present. Corporate users have Internet access and it is possible to perform remote access through Virtual Private Network services. The Level 4 is considered an extension of Level 5. It contains the IT systems that deal with reporting, scheduling, and inventory management programs. Moreover, at this level e-mail, phone and printing services are possible. Classic IT security teams are responsible for Enterprise level. A direct communication channel between Corporate zone and Manufacturing one is strongly discouraged. Conversely, it is not uncommon, especially in medium-low dimension companies, to see high risky connections between zones without the necessary security measures implemented. According to satisfactory security practices, the inclusion of a Demilitarized zone between Corporate and Manufacturing zones is recommended. It is an intermediate zone that provides secure shared services and data between two zones.

Manufacturing Zone: Level 3. The systems in Level 3 are responsible for managing control plant operations to produce desired end products. For instance, systems and services that are present in this level include: plant historians, plant reporting systems, production scheduling systems, and engineering workstations. Level 3 systems communicate with network nodes in Corporate zone through conduits. On the other side, systems in Level 3 are able to communicate with systems in the Control zone.

Control Zone: Levels 2, 1, and 0. This zone is also known as Plant zone, the closest network to physical processes. Starting from the lower one, Level 0 basically represents the field, where sensors, actuators, and machines are connected. These systems are controlled by devices placed in Level 1. In fact, this level includes process control systems that receive input from sensors, process the data, calculate the control values and send control vectors to Level 0 actuators. Example of devices that are present in Level 1 are PLCs and RTUs. These devices run vendor-specific operating systems and are programmed and configured from engineering workstations. High-level management and control routines are handled by systems in Level 2. Supervisory control like SCADA stations, HMIs, control room workstations, and alarm systems are developed at this Level. For systems in this zone it is not possible to communicate with the Corporate zone directly. It is instead possible to communicate with Manufacturing zone systems using conduits.

As already stated, if ICS and IT business networks must be connected, it is recommended that the number of entry points into the ICS environment be kept to a minimum and implemented through security patterns like conduits. This will reduce the number of attack pathways that an intruder could use to get into the ICS network. It is well-known that direct communication between Corporate and Control zones should be prohibited for security reasons, anyway due to high cost of re-engeneerization or the impossibility of apply security-by-design [84] paradigm on pre-existent infrastructures, numerous companies, also CIs ones, use IT implementations where it is discouraged.

According to [85], the communication between Level 0, 1, and 2 are the most vulnerable points of ICS systems. Based on the history, systems placed close to field operations were traditionally isolated from the rest of the ICS network zones. Security-by-obscurity was assumed as the de facto security measure necessary to protect the infrastructure from malicious intruders. On the contrary, with the advent of the Internet, the more close is the IT connection to the field, the most is the potential damage that could cause. From this point of view, the Control zone assumed a critical role within ICS network system development. It is for this reason that this thesis has been devoted to design and implement innovative security solutions for the specific Control zone network.

2.4 Industrial Control System Security: the Control Theory Perspective

Different system model types can be used for the physical process description. The model choice depends on the examined process dynamics, characteristics, and modeling aims. The Linear Time Invariant (LTI) system is the simplest and the mostly used [86]. The standard

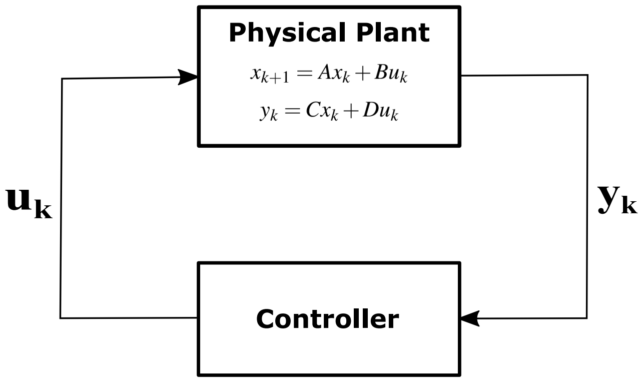


Fig. 2.5 Control system schema.

state description form of a continuous time LTI system is given by:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.1}$$

$$y(t) = Cx(t) + Du(t) \tag{2.2}$$

while for a discrete time LTI system:

$$x_{k+1} = Ax_k + Bu_k \tag{2.3}$$

$$y_k = Cx_k + Du_k \tag{2.4}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^p$ is the input (or control) vector, and $y \in \mathbb{R}^q$ is the output (or sensor) vector. Matrices A, B, C, D are real constant matrices and are defined as follows: $A \in \mathbb{R}^{n \times n}$ represents the state transition matrix, $B \in \mathbb{R}^{n \times p}$ is the input matrix, $C \in \mathbb{R}^{q \times n}$ is the output matrix, and $D \in \mathbb{R}^{q \times p}$ is the feedthrough matrix. The latter matrix will be considered as $D = 0$ because in general cases most physical systems are strictly causal. We will consider the discrete time LTI system, sketched in Fig. 2.5, as reference.

In presence of Gaussian noises, the systems described above become stochastic ones, for continuous time LTI systems:

$$\dot{x}(t) = Ax(t) + Bu(t) + w(t) \tag{2.5}$$

$$y(t) = Cx(t) + v(t) \tag{2.6}$$

while for a discrete time LTI system:

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{2.7}$$

$$y_k = Cx_k + v_k \tag{2.8}$$

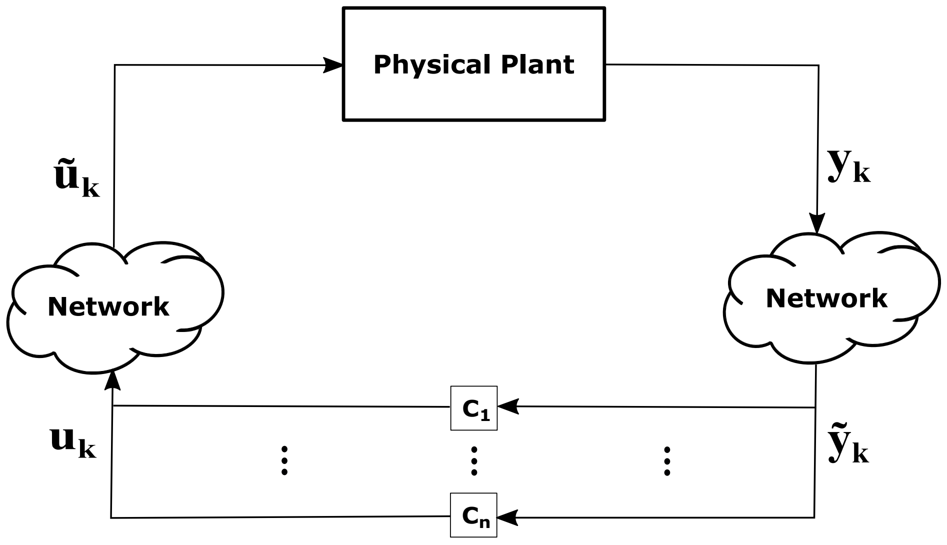


Fig. 2.6 Industrial Control System schema.

where $w(\cdot) \sim N(0,Q)$ and $v(\cdot) \sim N(0,R)$ are identical independent Gaussian noises having zero mean and variance Q and R, respectively.

In the case of ICSs, the discrete LTI system reference model changes. ICSs are spatially distributed systems whose control loops are connected through communication networks [87]. The overall system design has to deal with theoretical challenges considering loss of measurements and time-varying sampling due to the network communication channels [88]. The physical space integration with the communication network creates a new degree of interaction between these two domains [89]. Consequently, in the plant represented in Fig. 2.6, we obtain:

$$x_{k+1} = Ax_k + B\tilde{u}_k + w_k \tag{2.9}$$

$$y_k = Cx_k + v_k \tag{2.10}$$

where $\tilde{u}_k$ is the control vector passed through the communication channel:

$$\tilde{u}_k = u_k + u_k^c + u_k^a \tag{2.11}$$

where u_k^c is the quantization error and u_k^a is the attack vector. When $u_k^c = u_k^a = 0$ the ICS schema is equal to the classical control system one. Delays are not considered at this stage.

Considering the schema in Fig. 2.7, it is possible to identify three potential attack targets:

- attack against the communication channel on the system outputs y_k with y_k^a action;

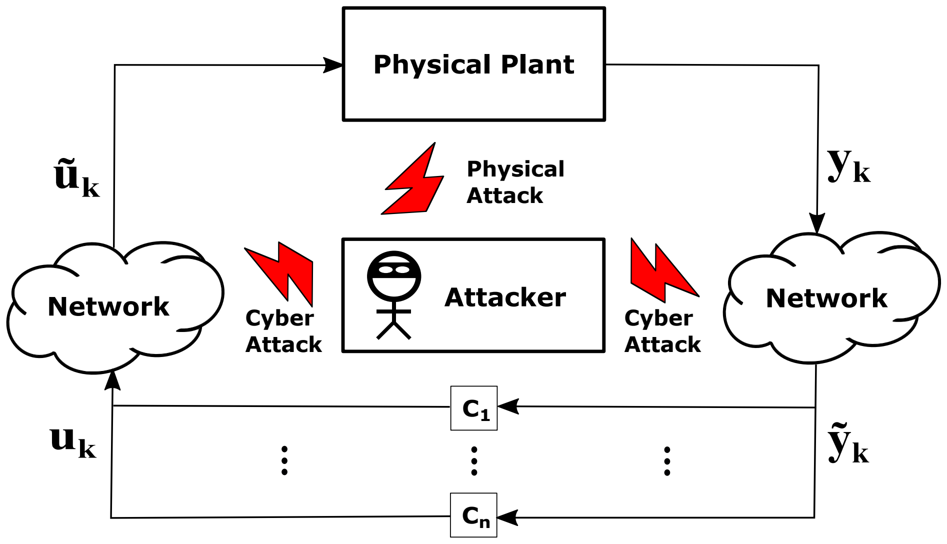


Fig. 2.7 Industrial Control System under attack schema.

- attack against the communication channel on the system control system u_k with u_k^a action;
- direct attack against the plant with malicious action on the physical system.

2.4.1 Control Techniques

Numerous control techniques can be used for LTI systems. The Linear Quadratic Gaussian (LQG) approach is presented in [87]. The aim of an LQG controller is to produce a control law u such that a quadratic cost J, that is function of both the state x and the control input u, is minimized:

$$J = \lim_{n\to\infty} E[\frac{1}{n}\sum_{j=0}^{n-1}(x_j^T\Gamma x_j + u_j^T\Omega u_j)] \tag{2.12}$$

where Γ and Ω are positive definite cost matrices [90].

Taking into account technical limitations and conditions, the LQG control approach has an optimal solution made of two components that can be modeled separately:

1 the design of a Kalman filter that produces an optimal state estimation $\hat{x}_k$ of the state x_k based on measurements with noise;

2 the design of the control action that produces the control vector u_k.

The Kalman filter [91] is an efficient recursive filter that evaluates the state of a dynamic system starting from a set of noisy measurements. For its intrinsic features, it is an optimal filter for noises and disturbances affecting Gaussian systems. The Kalman filter estimates the state as follows:

- Predict (a priori) system state $\hat{x}_k^{(-)}$ and covariance:

$$\hat{x}_k^{(-)} = A\hat{x}_{k-1} + Bu_{k-1} \tag{2.13}$$

$$P_k^{(-)} = AP_{k-1}^{(+)}A^T + Q \tag{2.14}$$

- Update parameters and (a posteriori) system state and covariance:

$$K_f = (P_k^{(-)}C^T)(CP_k^{(-)}C^T + R)^{-1} \tag{2.15}$$

$$\hat{x}_k^{(+)} = \hat{x}_k^{(-)} + K_f(\tilde{y}_k - C\hat{x}_k^{(-)}) \tag{2.16}$$

$$P_k^{(+)} = (I - K_f C)P_k^{(-)} \tag{2.17}$$

where K_f represents the Kalman gain, $P_k^{(+)}$ is the a posteriori error covariance matrix, and I is the identity matrix.

The optimal control law u_k provided by the LQG is a linear controller:

$$u_k = L\hat{x}_k^{(+)} \tag{2.18}$$

where L is the LQG feedback gain which minimizes the control cost and it is defined as follows:

$$L = -(B^T SB + \Omega)^{-1}B^T SA \tag{2.19}$$

where S is the matrix that solves the following discrete time algebraic Riccati equation:

$$S = A^T SA + \Gamma - A^T SB[B^T SB + \Omega]^{-1}B^T SA. \tag{2.20}$$

2.4.2 Detection Techniques

Detection techniques used in control theory take advantage of residues comparison with pre-designated threshold values. There are two major statistic detection techniques: stateless and stateful tests. The first one considers deviations at every time k, the second one considers the historical changes of residual values [92].

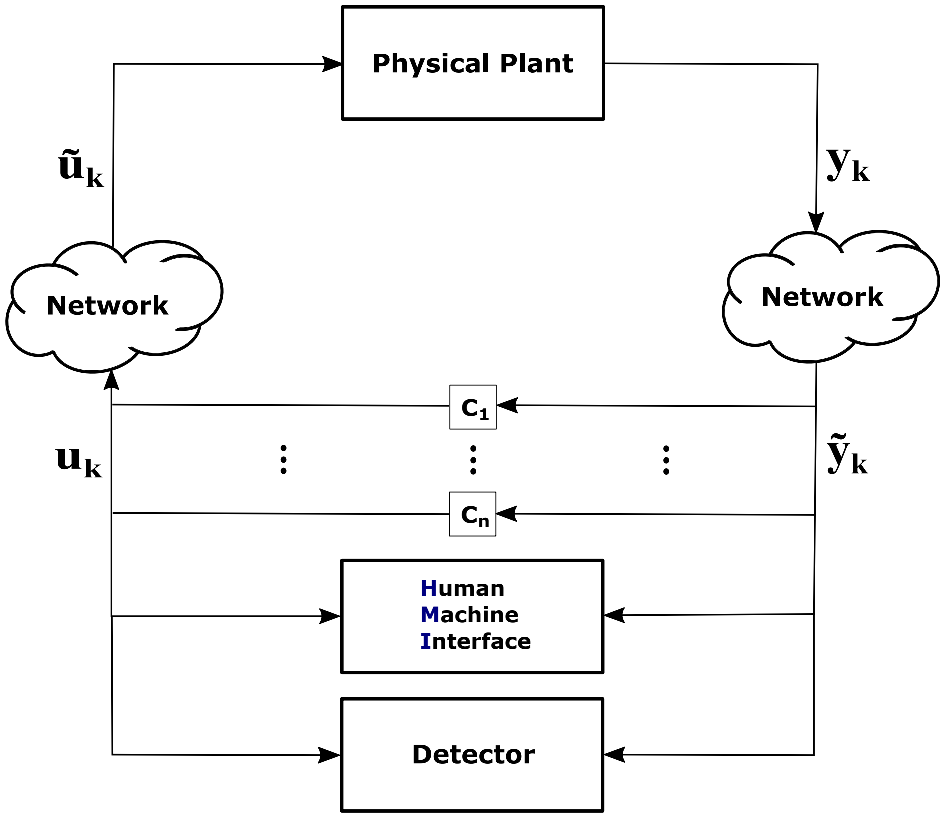


Fig. 2.8 Industrial Control System schema with detector, HMI, and controllers.

A stateless detection methodology concerns the χ^2 detection scheme proposed in [20, 93]. This detector is applicable to discrete LTI systems controlled by a LQG controller and it is classically used for fault or physical attack detection in control systems [94].

The presence of a HMI and the detector in the ICS schema is sketched in Fig. 2.8. Hypothesis for the detector operation is that the steady state condition is established for the transmission channel without delay. The residues are defined as follows:

$$r_k = \tilde{y}_k - C\hat{x}_k^{(-)} \tag{2.21}$$

It is important to notice that in this case the output vector y_k, passing through the transmission channel, can be subjected to data tampering by a malicious actor. For this reason, $\tilde{y}_k$ is used instead of y_k for the residual computation. Subsequently, an alarm signal g_k is computed based on the residues and is compared with a threshold γ to decide whether the system is in a normal state or not. The threshold is refined to minimize false alarms. The alarm signal g_k is

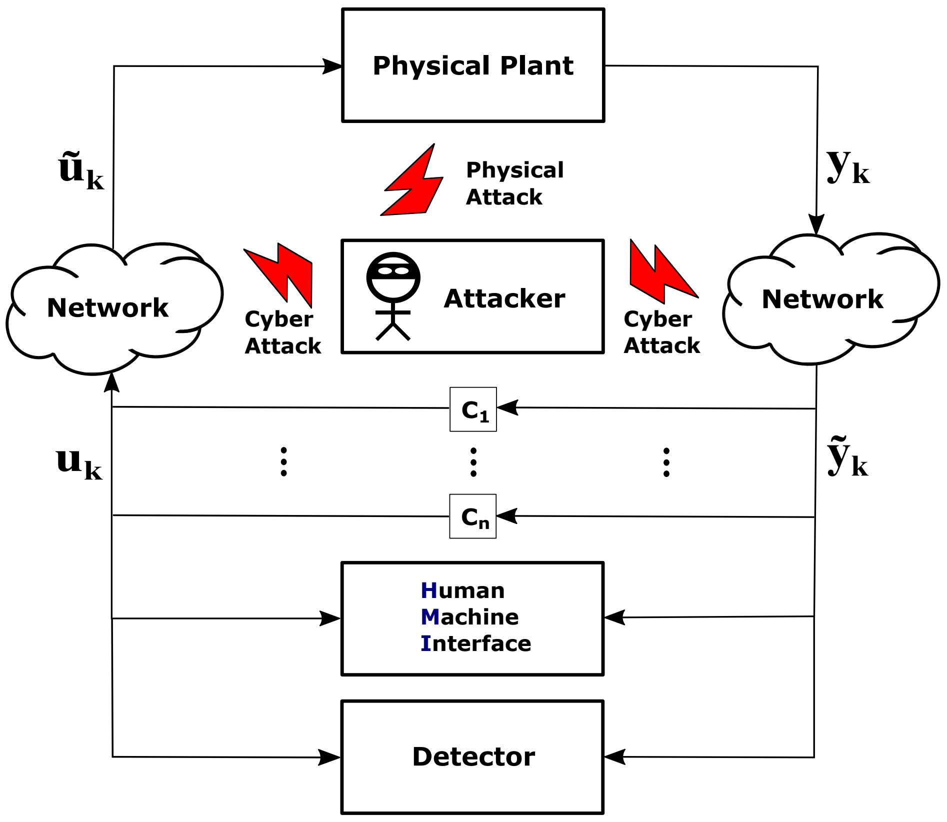


Fig. 2.9 Industrial Control System under attack schema with detector, HMI, and controllers.

computed as follows:

$$g_k = r_k^T \wp^{-1} r_k \tag{2.22}$$

$$\wp = (CPC^T + R) \tag{2.23}$$

There are two hypothesis concerning the alarm/no alarm states. If g_k is less than the threshold γ, then no attack or faults are detected on the network. Otherwise, if g_k results greater than γ, then the networked system is under faulty or attack conditions. In Fig. 2.9 the under attack situation is represented.

$$G_k = \begin{cases} \text{no alarm} & \text{if } g_k < \gamma \\ \text{alarm} & \text{if } g_k \geq \gamma \end{cases} \tag{2.24}$$

2.5 The Security Decision Problem

The DSSs are used in domains where it is not possible to directly apply optimization methods due to the presence of heterogeneities. In fact, these aim to support the decision-maker with the final aim to suggest the optimal solution. The expertise and the experience, as well as the reasoning of a human can be hardly replaced by automatic machines, however DSS can help operators to set up a reaction strategy. When an emergency or abnormal situation on the system occurs, the human operator has to take decisions in order to manage the problem in the best way. Concerning CI, such decisions can create several dire consequences.

Introduced by Saaty in the ‘70s, AHP represents a Multiple-Criteria Decision Making method. It is exploited in this work since it is an useful tool to perform decisions in CI security scenarios. To the best of our knowledge, approaches that use decision support methods for the context of endorsing the CI human security operator have never been used. In [95], the authors assess the risks in electric power SCADA systems using AHP as method useful to build an index about vulnerability risk assessment. In [96], a comparison among different IDSs is presented using the AHP multi-criteria decision making technique. In [97], a risk management strategy called Risk Management Index is proposed: it provides a set of indicators, based on predefined qualitative target and benchmarks, that allow to measure risk management performance and reduce vulnerability and losses in a given area, using the AHP method. In [98], the authors show how Safety Management can be performed through the AHP approach which allows both multi-criteria and simultaneous evaluation; the problem is presented using fuzzy numbers. In [99], both AHP and fuzzy AHP approaches are exploited in order to choose the load priority in load shedding scheme for a large pulp mill. In [100], a support framework based on AHP for the evaluation and the automation of water and water-waste treatment is presented. In [101], a review of AHP is performed in the assessment of the riskiness of constructing the Jamuana Multipurpose Bridge in Bangladesh. In [102], the authors demonstrate a quantitative approach through the AHP and decision tree analysis in order to support risk management in a case application of a cross-country petroleum pipeline project in India. In [103], an application of AHP to assist decision makers in banking industries is showed; the proposed model deals with both information security policy and information security elements.

All the approaches described above are references present in the literature regarding the use of the AHP method to support the decision maker mostly for risk management topics. The approach presented in this thesis is innovative and will be presented with the appropriate grade of details in Chapter 3.

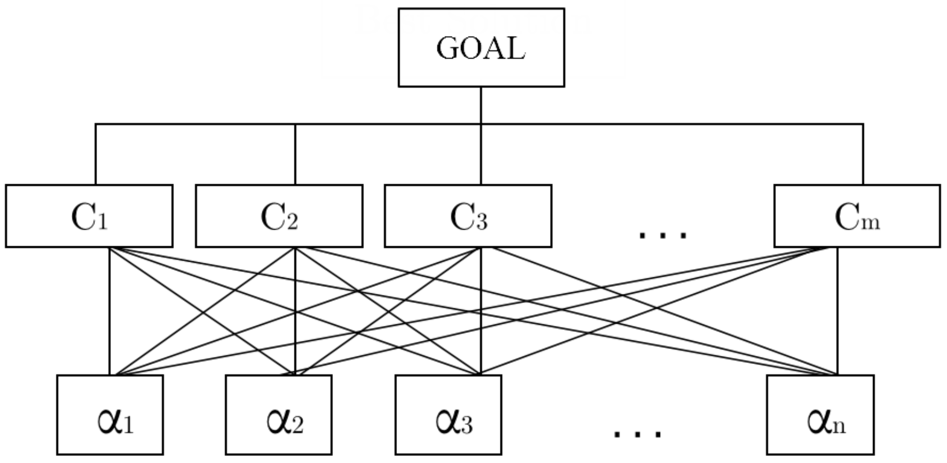


Fig. 2.10 General AHP evaluation structure for m criteria and n alternatives.

Table 2.1 Saaty's scale

Intensity of importance	*Definition*
1	Equal importance
3	Weak importance of one over another
5	Essential or strong importance
7	Demonstrated importance
9	Absolute importance

2.5.1 The Analytic Hierarchy Process

The AHP supports the decision among different alternatives based on certain criteria. Starting from n given alternatives $\alpha_1, \dots, \alpha_n$ and m criteria $c_1, \dots, c_m$, it is supposed that, for each criterion c_k and for each pair of alternatives (α_i, α_j) *relative* importance metric is given by:

$$a_{i,j,k} = \frac{w_{i,k}}{w_{j,k}}, \tag{2.25}$$

where $w_{i,k}, w_{j,k}$ are the *absolute* importance of the alternatives α_i and α_j with respect to the criterion c_k, and they are in general unknown.

The AHP method represents a tool to find the absolute importance values $w_{i,k}$ for each alternative and for each criterion; moreover, by ranking also the absolute importance of the criteria, the method allows to select the most important alternative, considering the overall importance for them. The most important alternative is a suitable linear combination of the absolute importances according to the different criteria. The AHP evaluation scheme is sketched in Fig. 2.10.

During the analysis and construction of the support model, ad hoc relative weights have been added considering the Saaty's scale (Tab. 2.1) and the pairs of possible actions. Despite large number of scales has been designed in the literature to derive priorities, the Saaty one has become the de facto standard. The rationale of this choice is the direct and easy verbal responses comparison with tangible numerical values [104]. This decision support method considers a control process of data consistency by calculation of a specific index (Consistency Index). That allows the decision maker to maintain consistency with all pairwise comparisons. If the considered ratio is not acceptable, the inconsistency is distributed to the whole pairwise comparisons matrix through the powers method [105].

Considering the analytic perspective, let stack the coefficients $a_{i,j,k}$ for a given criterion k in the matrix A_k and let $\mathbf{w}^{(k)} \in \mathbb{R}^n$ be a vector containing the absolute weights for the k-th criterion. It holds [105]:

$$A_k\mathbf{w}^{(k)} = \begin{bmatrix} \frac{w_{1,k}}{w_{1,k}} & \cdots & \frac{w_{1,k}}{w_{n,k}} \\ \vdots & \vdots & \vdots \\ \frac{w_{n,k}}{w_{1,k}} & \cdots & \frac{w_{n,k}}{w_{n,k}} \end{bmatrix} \begin{bmatrix} w_{1,k} \\ \vdots \\ w_{n,k} \end{bmatrix} =$$

$$= \begin{bmatrix} w_{1,k} & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & w_{n,k} \end{bmatrix} \begin{bmatrix} \frac{1}{w_{1,k}} & \cdots & \frac{1}{w_{n,k}} \\ \vdots & \ddots & \vdots \\ \frac{1}{w_{1,k}} & \cdots & \frac{1}{w_{n,k}} \end{bmatrix} \begin{bmatrix} w_{1,k} \\ \vdots \\ w_{n,k} \end{bmatrix} = n \begin{bmatrix} w_{1,k} \\ \vdots \\ w_{n,k} \end{bmatrix}.$$

As a consequence of the above equation, it can be noticed that A_k has rank equal to 1 and has only one non zero eigenvalue $\lambda = n$, while $\mathbf{w}^{(k)}$ is indeed the associated eigenvector. In [105], the author proposes to choose the one obtained by normalizing all the entries, dividing them by the sum of each one of them. Taking advantage of the above results, the weights $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(m)}$ associated to the different criteria can be therefore calculated. In order to rank the absolute relevance of all the criteria, it is supposed that for each pair of criteria (c_i, c_j) their relative importance q_{ij} can be expressed as:

$$q_{ij} = \frac{z_i}{z_j}, \tag{2.26}$$

where z_i, z_j are the absolute importance of the criteria and they are in general unknown. Let Q be the matrix that collects the terms q_{ij}, it turns out that this matrix has the same features as A_k, so $\mathbf{z} = [z_1, \ldots, z_n]^T$ is the normalized eigenvector of Q associated to the unique non zero eigenvalue $\lambda = m$. Knowing both the absolute importance of the alternatives, according to the specific criteria, and the relevance of the criteria, the method selects the best alternative by solving the following optimization problem:

$$\alpha_{i^*} \quad s.t. \quad i^* = \underset{i=1,\ldots n}{\arg\max} \, \{\sum_{j=1}^{m} z_j w_{i,j}\}. \tag{2.27}$$

Chapter 3

Deep Detection Architecture

This chapter is devoted to introduce the logical framework for the ICS network security designed during the doctorate. The development of new security architectures encompasses the guidelines proposed from both academy and industry. Thus, it is mandatory to consider, on the one hand, consolidated standards and directives, on the other hand, future perspectives. Some key references for the design of this architecture can be found in [2, 11, 106–109]

3.1 Global Architecture Overview

There are many tools allowing security solutions on networks. However, in the framework of ICS for CI, the classic cyber security methods, adopted in IT, do not represent an ideal solution. Isolated signature-based IDSs, for example, perform a safety check of the traffic based on static rules but are not able to identify a zero-day attacks since they do not consider a dynamic analysis of the network. The zero-days, indeed, are cyber-threats that take advantage of vulnerabilities that are not yet identified and represent the highest threat for ICS.

To overcome limits of classical detection methodologies and apply new solutions to industrial network security scenarios, an innovative architecture has been developed. The modules constituting the proposed security architecture, outlined in Fig. 3.1, have been conceived to be located in the Control zone, characterized by network traffic coming from Levels from 0 to 2. Connections between Control zone network and the Manufacturing one it is assumed to be secured by conduits. For the sake of clarity, it has been assumed that traffic from Levels from 0 to 2 is handled through a single switch, although the proposed architecture can be implemented in a distributed fashion. The architecture is composed by:

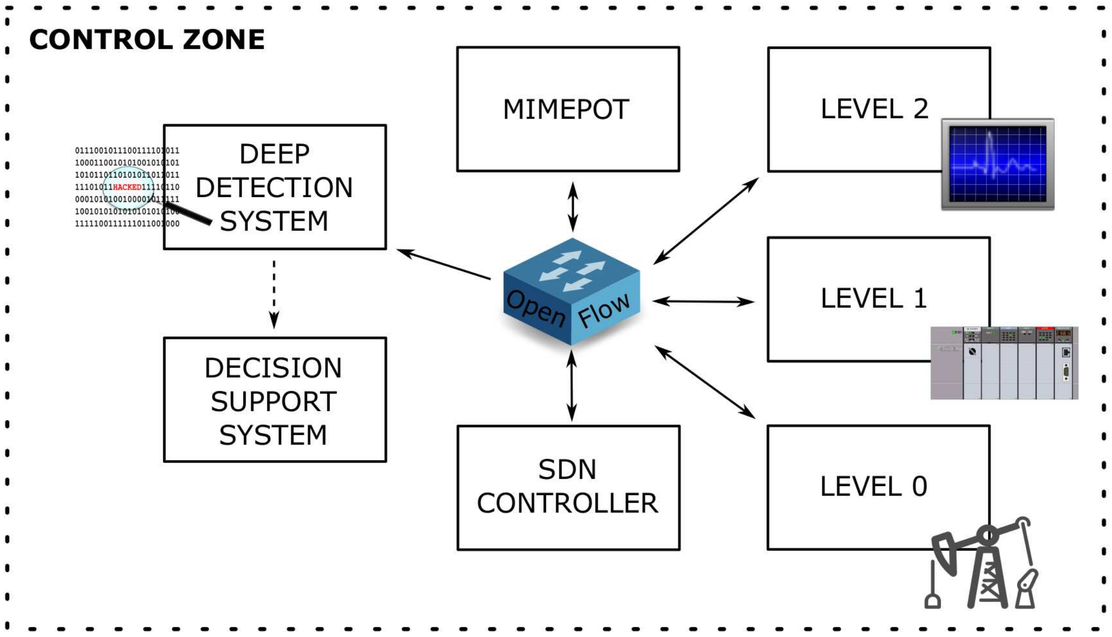


Fig. 3.1 Global architecture schema.

- **Deep Detection System (DDS)**: is the core of the architecture, it represents the advanced detection system developed. This module is composed of different submodules dedicated to different security tasks;
- **Software Defined Networking Controller (SDN Controller)**: it is the network traffic controller, able to redirect traffic according to specific laws and OpenFlow switch potentialities. This module is realized according to SDN paradigm;
- **Mimepot**: it is an advanced honeypot able to reproduce the physical processes of the plant and its control routines;
- **Decision Support System (DSS)**: it is developed to support the security operator in decision making procedures using a Multiple-Criteria Decision Making Method.

These modules will be deeply investigate in the followings. It is worth noticing that the security architecture conceived is adaptive to different industrial network deployments due to the high modularity. Moreover, it is possible to develop multiple modules to satisfy distributed control system requirements. According to this, central security stations that take in charge the activities of distributed modules can be developed.

To understand the interaction among the architecture modules, an attack situation is described in Fig. 3.2. It is assumed that an attacker has been able to unfairly obtain direct

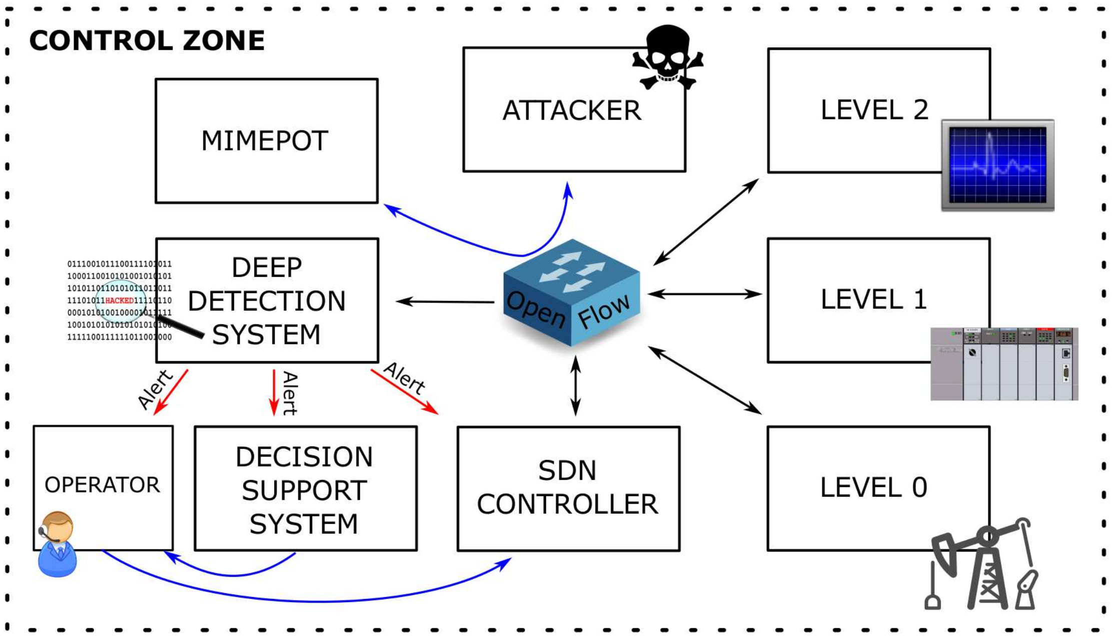


Fig. 3.2 Global architecture under attack schema.

connection to the Control zone network. In this case, the DDS detects the malicious activities on the network and it is able to send alerts to the security operator, to the DSS, and to the SDN Controller module. Then, taking into account the suggestions provided by the DSS, the operator directly applies or supervises the activity of the SDN Controller module. In a malicious situation, the SDN Controller starts to redirect in a smart way the traffic coming from the attacker node: the attacker continues to believe in a direct connection with the real plant, but on, the contrary, he is connected with the Mimepot due to the traffic redirection ability of the SDN Controller. Once trapped, the Mimepot starts to silently study the attacker dynamics without interrupting the malicious connection. This is a novel perspective, where the attacker is lured. Data provided by the Mimepot can be further exploited at forensic level to investigate the malicious intents of the attacker.

This thesis collects the efforts in designing, implementing, and evaluating the most critical modules of the proposed architecture, such as DDS and DSS.

3.2 Deep Detection System

The DDS represents an evolved network detection system able to differentiate between cyber and physical issues affecting the industrial control system. The main objectives of the architecture are

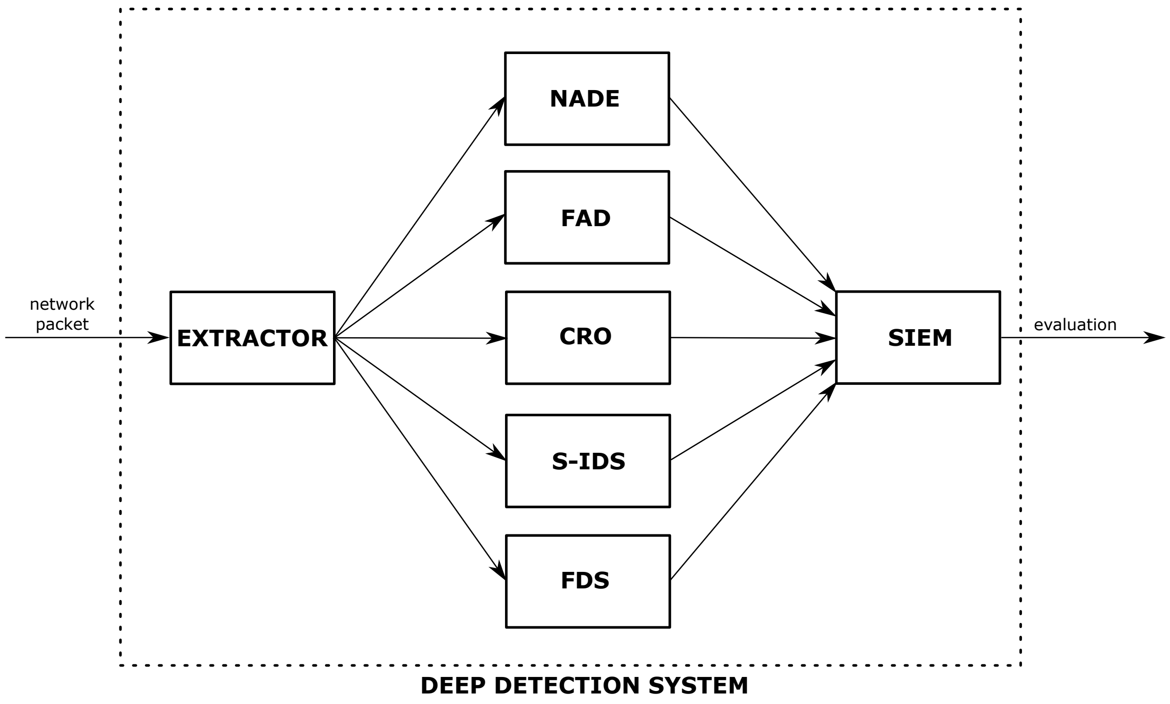


Fig. 3.3 Deep Detection System global architecture.

- to detect the cyber threats affecting the global aspects of availability, integrity, and confidentiality;
- to clearly identify and separate physical problems and cyber threats.

Unlike traditional IDS, that integrate the whole identification module in a single tool, the DDS presented in this work has been conceived to include several flexible and interchangeable modules for a better global security evaluation. It is composed by the following modules: the Extractor, the Network Anomaly Detection Engine (NADE), the Finite Automata Detector (FAD), the Confidentiality Restriction Detector (CRO), the Signature-based Intrusion Detection System (S-IDS), the Fault Detection System (FDS), and the Security Information and Event Management (SIEM) system. The detection modules are shown in Fig. 3.3.

Three modules (i.e., NADE, FAD, and CRO) of the architecture require a pre-configuration step, i.e., a learning phase. Assuming that it is possible in an implementation scenario to foresee secure initial state of the plant, a learning phase is defined as the procedure for the safe configuration and set-up of the modules that necessarily require historical database of plant status as reference for the detection. For a formal representation, this stage has been described in Fig. 3.4, where the target modules have been renamed in L-NADE, L-FAD, and L-CRO, i.e., Learning-NADE, Learning-FAD, and Learning-CRO in order to highlight the specific phase.

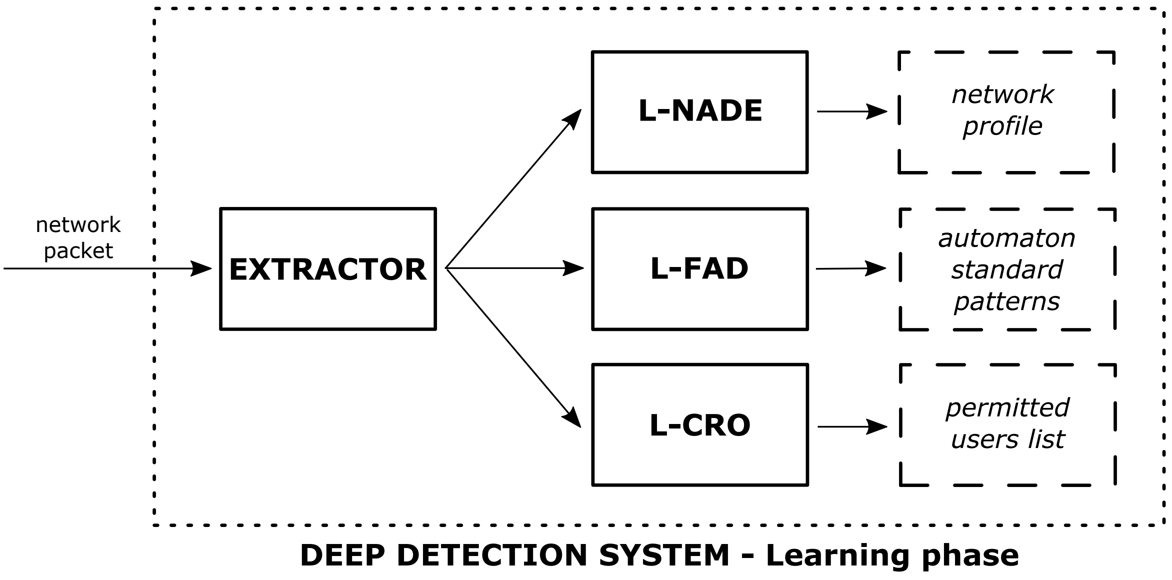


Fig. 3.4 Deep Detection System learning phase.

The module operations and the functionalities will be described in the following sections.

3.2.1 Extractor

In a detection schema, a dedicated module is mandatory to extract data and forward them to the corresponding detector module, since different network layers are analyzed and several data characteristics are involved.

The Extractor is a network module able accomplish data filtering tasks, allowing the detection phase for the modules devoted to the analysis. Exploiting different network filters, this module extracts data from the network to provide information for the subsequent modules. It receives all the network packets, selects the information and then sends these to the modules. The operation of this module is fundamental since it allows a more efficient analysis.

3.2.2 Network Anomaly Detection Engine

The NADE is devoted to the behavioral-based anomaly detection. This module for his functionalities requires two phases: a learning phase and an active detection phase. Both the phases use the same data provided from the Extractor module but have different operating mechanisms.

The two stages are described as follows:

- **Learning - Network Anomaly Detection Engine (L-NADE)**: this component, sketched in Fig. 3.4, uses the network data provided by the Extractor to generate a profile of the normal network behavior. The way the profile is generated represents the most important fulfillment of the anomaly detection. The more accurate the model, the more the possibility to identify system faults on the network traffic. This module is executed in a nominal condition without undergoing attacks or anomalous situations. Multiple data of the same operations are stored to better determine the normal behavior patterns. The time required to complete the learning phase depends on the period of the system. For example, for a water CI system, if cyclic monitoring and control operations that last one week are identified, it will be necessary to save the network traffic for a week to evaluate the normal behavior.

Comparing with the anomaly detection tools presented in the literature, the L-NADE strength is represented by configuration possibilities: this allows to easily adapt the proposed system to any network for the anomalies analysis. Moreover, this module is able to extract necessary data from the network traffic: it is possible to select any traffic characteristic of the protocol under analysis in order to provide ad hoc anomaly detection solutions. This feature represents a valuable option for CI security research scenarios due to the adaptability requirements. The L-NADE inputs are provided by the Extractor and the output is a profile file, generally a Comma Separated Values file, containing the network profile data.

The learning phase of NADE runs before the effective detection. However, it is possible to regenerate the network profile, whenever an update is necessary.

- **Network Anomaly Detection Engine (NADE)**: Once the creation of the network profile is completed, this is exploited to perform anomaly detection active tasks. The NADE analyses the traffic and compares it each sampling time with the set of parameters generated from the L-NADE. The inputs of this component are the network profile file and the up-to-date network data provided by the Extractor. The outputs are the alerts provided to the SIEM for the security evaluations and are simultaneously saved into a log file. It is important to highlight that zero-days attacks can be identified by analyzing the behavior of the network. To this end, the periodic behavior of the CI represents an advantage in terms of anomalies analysis on the network.

3.2.3 Finite-Automata Detector

The FAD system is an innovative module developed using deterministic finite automaton state machine. Starting from the work in [60], this module recognizes predefined patterns to differentiate between normal and anomalous packets chains in the transmission. Formally, a finite automaton is defined as an idealized machine that uses patterns to recognize input taken from a character set. The final objective of a finite-state machine is to accept or reject an input depending on the specific path defined by the states and transition functions. Each state represents a predefined condition that the input can reach according to the functions defined for the transitions. A deterministic finite automaton can be represented by a 5-tuple:

$$M = (Q, \Sigma, \delta, q_0, F) \tag{3.1}$$

where Q represents a finite set of states, Σ is the alphabet of input symbols, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of feasible states. This module can be used with multiple protocols: in the context of ICS networks it finds usefulness into the DDS architecture to analyze specific control network protocols. It is straightforward to understand that the development for specific protocols requires distinct states and transition functions. This is a twofold characteristic: on one side, it is possible to adapt this module to several protocols, on the other, the security operators have the chance to customize the finite-state machine in order to best fit with deployment restrictions.

For the sake of completeness, considering peculiarities of this module, a specific FAD conceived for the Modbus/TCP protocol is presented. The module is based on Modbus/TCP query and response packets analysis. Referring to [110], query packets contain requests for a specific operation to be done, characterized by Function Codes (e.g., Read Input Register, Write Single Coil, etc.), whereas response packets include the corresponding responses to the required operations. The FAD detection module, as outlined in Fig. 3.4, expects a learning phase, namely L-FAD. During this phase, performed in safe system state, the module is able to recognize the standard patterns of Modbus/TCP network traffic and specifically, the Function Codes used in the normal behavior are stored as reference. For the FAD module presented, the tuple M is composed of:

$$Q = \{S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8\} \tag{3.2}$$

$$\Sigma = \{a, b, c, d, f, g, h, i, l, m\} \tag{3.3}$$

$$q_0 = S_0 \tag{3.4}$$

$$F = \{S_5, S_6, S_7, S_8\} \tag{3.5}$$

δ is defined according to the two dimensional state transition table in Tab. 3.1. According to Table 3.2, where finite-state machine states are described, the transition functions for the schema in Fig. 3.5 are discussed below:

Table 3.1 Two dimensional state transition table.

Current state \ Next state	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
S_0	a	b	c	-	-	-	-	-	-
S_1	-	-	-	d	-	e	-	-	-
S_2	-	-	-	-	f	-	g	-	-
S_3	-	-	-	-	-	-	m	h	-
S_4	-	-	-	-	-	-	m	-	i
S_5	-	-	-	-	-	-	-	-	-
S_6	-	-	-	-	-	-	-	-	-
S_7	-	-	-	-	-	-	-	-	-
S_8	-	-	-	-	-	-	-	-	-

Table 3.2 Finite Automata Detector states description.

State	**Description**
S_0	packet check
S_1	q packet arrived
S_2	r packet arrived
S_3	$Tr_{ID}(q)$ accepted
S_4	$Tr_{ID}(r)$ accepted
S_5	q retransmission
S_6	integrity attack
S_7	store q in queue
S_8	accept r, remove q_i from queue

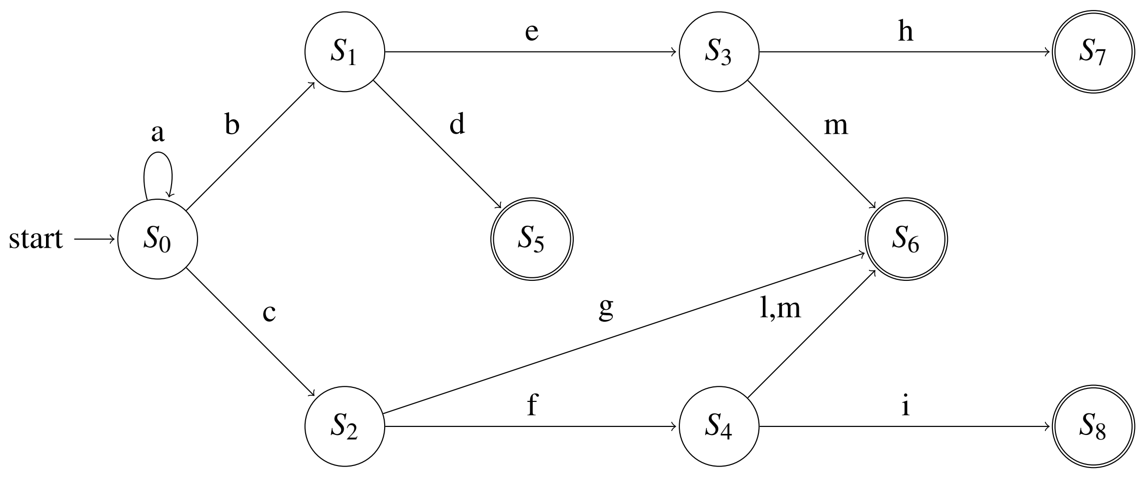


Fig. 3.5 Finite Automata Detector for Modbus/TCP.

start : specific packet data, sent by the Extractor module, arrives into the finite-state machine for analysis. First state S_0 represents the packet check analysis;

a $\rightarrow$ not *Modbus/TCP* packet : considering the analysis performed in S_0, if a packet is not attributed to the specific protocol under consideration, in this case the Modbus/TCP, then the analysis steps forward to the next packet and the finite-state machine remains into the S_0 state.

b $\rightarrow$ q packet : the packet received is a query Modbus/TCP packet, the transition function b leads the analysis to state S_1, query packet arrived. Transaction identifier of the packet, $Tr_{ID}(q)$, is stored into a queue;

c $\rightarrow$ r packet : the packet received is a response Modbus/TCP packet, the transition function c drives the analysis to state S_2, response packet arrived. Transaction identifier of the packet, $Tr_{ID}(r)$, is stored into a queue;

d $\rightarrow$ $Tr_{ID}(q) = Tr_{ID}(q_i)$, $i \in \{1,\ldots,n_q\}$: checks are made to compare Transaction identifier of current query packet, $Tr_{ID}(q)$, with the already stored Transaction identifier of previous query packets, $Tr_{ID}(q_i)$ where $i \in \{1,\ldots,n_q\}$. According to Modbus/TCP protocol specifications [111], on a TCP connection, the Transaction identifier must be unique. In the case of $Tr_{ID}(q) = Tr_{ID}(q_i)$ where $i \in \{1,\ldots,n_q\}$, a retransmission occurred and investigations about this anomaly should be carried out. For this reason, the transition function d arrives in the final state S_5 which identifies a retransmission for the packet under investigation;

e $\rightarrow Tr_{ID}(q) \neq Tr_{ID}(q_i)$**,** $i \in \{1,\ldots,n_q\}$ **:** after verifies that Transaction identifier of current query packet is not equal to Transaction identifiers previously stored, this transition function permits to reach S_3 state, namely $Tr_{ID}(q)$ accepted;

f $\rightarrow Tr_{ID}(r) = Tr_{ID}(q_i)$**,** $i \in \{1,\ldots,n_q\}$ **:** checks are performed on the Transaction identifier of the current response packet received, $Tr_{ID}(r)$. If the latter is already present in the queue of queries Transaction identifiers, $Tr_{ID}(q_i)$ where $i \in \{1,\ldots,n_q\}$, the response packet is licit and the transition function f can move the analysis to state S_4 ($Tr_{ID}(r)$ accepted);

g $\rightarrow Tr_{ID}(r) \neq Tr_{ID}(q_i)$**,** $i \in \{1,\ldots,n_q\}$ **:** if the verification of $Tr_{ID}(r)$ reveals that the Transaction identifier is not corresponding with any of the $Tr_{ID}(q_i)$ where $i \in \{1,\ldots,n_q\}$, it is possible to conclude with some degrees of certainty that the system is under *integrity attack*, represented in state S_6.

h $\rightarrow fc(\cdot)$ **admitted :** after the query packet Transaction identifier approval, a legitimacy function code verification is performed. As already stated, during the L-FAD phase, the normally used function codes of Modbus/TCP packets are saved. According to the transition function h, if the function code is licit, the state S_7 is reached and the function code relative to the query packet under investigation, $fc(q)$ is stored in a queue.

i $\rightarrow fc(r) = fc(q_i)$ **AND** $Tr_{ID}(r) = Tr_{ID}(q_i)$**,** $i \in \{1,\ldots,n_q\}$ **:** in this case, the function code admittance is checked, as it is the Transaction identifier. The function code of the packet under analysis is the same of the query packet with the same Transaction identifier. Consequently, with the transition function h, the finite-state machine can reach to the final state S_8, where the response packet is accepted and the query q_i is removed from the queue.

l $\rightarrow fc(r) \neq fc(q_i)$ **AND** $Tr_{ID}(r) = Tr_{ID}(q_i)$**,** $i \in \{1,\ldots,n_q\}$ **:** this transition function identifies a response packet that has the licit Transaction identifier, $Tr_{ID}(r) = Tr_{ID}(q_i)$, with a function code that is admitted but is not the same of the respective query, $fc(r) \neq fc(q_i)$. According to this, the transition drives to the *integrity attack* state (S_6).

m $\rightarrow fc(\cdot)$ **not admitted :** According to the transition function m, if the function code of both query or response packet under analysis is not present in the admitted function code list, generated during L-FAD phase, the FAD detects an *integrity attack*.

Once reached one of the final states, there are two possibilities: continue the analysis or generate alerts. In the first case, if final states S_7 or S_8 are reached, the module continues its analysis without generating alerts. Otherwise, if final states S_5 or S_6 are reached, the module generates alerts according to the specific issue identified and transmits these to the SIEM module.

The combination of both FAD and NADE modules allows to reduce the false positive alarms rate. This issue related to the *unusual-but-normal* situations is more relevant for the anomaly-based detection module due to the direct correlation with the normal behavior profile with well defined thresholds. Thus, a very important feature of the proposed cyber-physical security architecture is highlighted: the joint use of the modules supports the monitoring strengths of the Deep Detection Architecture conceived.

3.2.4 Confidentiality Restriction Detector

The CRO represents a security module dedicated to find issues inherent data confidentiality. Although the confidentiality for ICS networks is not considered as a priority, with the IoT advent and the ever-increasing ease of sharing confidential information, becomes important to not underestimate it.

This module requires a learning phase, sketched in Fig. 3.4, namely L-CRO, devoted to populate a database with white-listed devices. Assuming the learning phase performed in a safe ICS state, the L-CRO is able to list all the devices connected in the Control zone in order to create the white-list. The L-CRO provides an automatic learning mechanism for permitted users and devices. In addition, during L-CRO activity, there is an implementation choice which considers the insertion of an entity twice in order to apply a simple but effective security check: if a malicious actor finds a way to alter the white-list, he/she would not be aware of the necessity to insert twice the same entity. In this way, during legacy user database analysis, if there were entities with a single presence, a security alert would be generated. The authentication provided by the CRO manages users directly within the Control zone in order to prevent any intrusion into this area. CRO module receives connected devices information (e.g., IP address, MAC address) and it verifies that these devices are present into the white-list. As output, this module generates alerts directed to the SIEM module.

The operation mechanism is simple but it allows to prevent multiple potential cyber threats, such as Address Resolution Protocol (ARP) Poisoning. The CRO module integrates and not replaces the recommended security conduit between Manufacturing and Control zones. The main difference between CRO and conduit is the following: the first checks network devices and users within the Control zone, the second serves as authentication gate for who try to enter the Control network coming from a different zone.

The implementation of this module has not been directly addressed in this thesis, since it focuses on security against integrity and availability attacks, that represent the main issues in CI scenarios.

3.2.5 Signature-based Intrusion Detection System

The S-IDS is a detection tool based on predefined rules. It is fundamental in IT security applications, however its importance has been recognized also in ICS. Even if there should not be classic IT threats inside the Control zone, it is not possible to exclude a priori the possibility that these should arise. For this reason, the use of signature-based intrusion detection techniques can be useful also for the Control network. Moreover, it is possible to define ad hoc rules for network protocols that operate in the Control zone, normally different from those present in the Corporate zone. This allows to use rules for redundant security control of physical processes through the analysis of network traffic related to specific protocols.

3.2.6 Fault Detection System

Control Theory has contributed with frameworks to handle model uncertainties and disturbances as well as fault detection and mitigation respectively. In Figure 3.6, the structure of a system and the associated entities devoted to the identification of faults are depicted. The physical plant, together with actuators and sensors, are all subjected to faults and attacks. Thanks to the analysis of input and output, the FDS is able to identify problems. After the detection, the operators apply decisions considering the evaluation of the specific post filtering methodologies implemented.

The classical FDS [86] consists in the identification of anomalies on the system behavior due to the occurrence of physical faults. Since cyber-attacks on networked control systems also affect the physical behavior of the system, fault detection can be used to detect cyber-attacks and mitigate their consequences. In many works in the literature, the fault detection problem is addressed using linearized systems, as shown in Chapter 2. In this work, this problem is approached using non-linear control dynamics, in order to produce a more accurate reliable model. On the other hand, computational load is higher.

Cyber-attacks and faults have inherently distinct characteristics, which pose different challenges on the secure control and fault-tolerant approaches. Faults are considered as physical events that affect the system behavior, where simultaneous events are assumed to be non-colluding (e.g., the events do not act in a coordinated way). On the other hand, cyber-attacks may be performed over a significant number of attack points in a coordinated

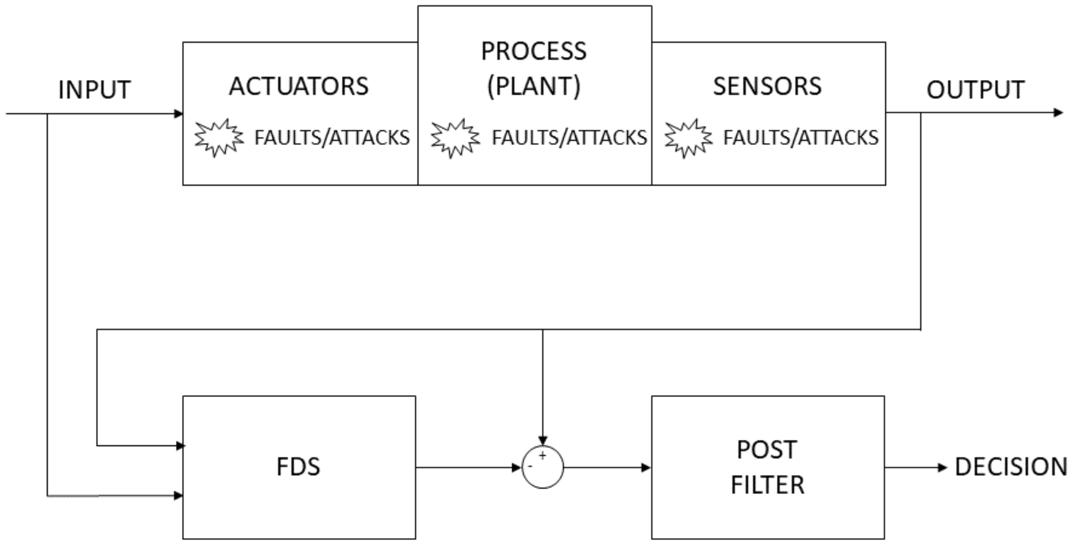


Fig. 3.6 Fault Detection System schema.

fashion [21, 30]. Moreover, faults do not have an intent or objective to fulfill, as opposed to cyber-attacks that do have a malicious intent.

FDSs are the simplest fault diagnosis systems: they trigger alarm signals to indicate the fault presence. Different approaches for fault detection using analytical models have been developed in the last 20 years. The task consists in the detection of faults in the processes, actuators, and sensors by using the dependencies between different measurable signals. Analytical process models express these dependencies. In the same way, also attacks on sensors and actuators can be modeled. In most practical cases, the process parameters are partially not known or not known at all. Then, they can be determined with parameter estimation methods by measuring input and output signals, if the basic model structure is known.

Model based fault detection methods is addressed in this thesis in two different ways. First, the concepts of analytical redundancy and residual generation under the assumption of a perfect system model are introduced. According to this assumption, the non linear Fault Detection Filter (FDF) is considered. It is a deterministic observer. Second, the Extended

Kalman Filter (EKF) is introduced as observer for system affected by disturbances and uncertainties. EKF turns to be an observer for stochastic systems.

Nonlinear FDF is based on the first FDF observer-based residual generators developed by Beard and Jones in the early 70's [112]. The FDF by Beard and Jones represents the development beginning of model-based FDI techniques. The core of the non linear FDF proposed in this thesis is a full-order state observer

$$\hat{x}_{k+1} = f(x_k, u_k) + L[y_k - h(x_k, u_k)] \tag{3.6}$$

$$\hat{y}_{k+1} = h(x_{k+1}, u_{k+1}) \tag{3.7}$$

which is built on the nominal system model described by the map $f(\cdot)$ and $h(\cdot)$, where $\hat{x}_k$ and $\hat{y}_{k+1}$ are the state estimate and the output estimate, respectively, u_k is the input vector and y_k the output vector of the real system. L is the observer matrix, able to reduce the estimate error. The selection of the observe matrix is crucial. For stochastic systems, the EKF is considered. The estimate of the expected measurement and the associate uncertainty are computed by means of prediction and correction steps. In the prediction, the state estimate is performed as

$$\hat{x}_{k|k-1} = f(k, \hat{x}_{k-1|k-1}, u_k) \tag{3.8}$$

$$P_{k|k-1} = A_k P_{k-1|k-1} {A^{\mathrm{T}}}_k + FQF^{\mathrm{T}} \tag{3.9}$$

where $f(\cdot)$ is the state transition map, $P_{k|k-1}$ is the state covariance matrix, and

$$A_k = \left[\frac{\partial f(\cdot)}{\partial x}\right]_{x=\hat{x}_{k|k}}. \tag{3.10}$$

In the correction step, the estimate is further refined according to the following equation:

$$S_k = C_k P_{k|k-1} {C_k}^{\mathrm{T}} + HRH^{\mathrm{T}} \tag{3.11}$$

$$K_k = P_{k|k-1} {C_k}^{\mathrm{T}} {S_k}^{-1} \tag{3.12}$$

$$\gamma = y_k - h(k, \hat{x}_{k|k-1}, u_k) \tag{3.13}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \gamma \tag{3.14}$$

$$P_{k|k} = (I - K_k C_k) P_{k|k-1} \tag{3.15}$$

where K_k the Kalman gain and S_k the innovation covariance matrix and:

$$C_k = \left[\frac{\partial h(\cdot)}{\partial x}\right]_{x=\hat{x}_{k|k-1}}. \tag{3.16}$$

In both cases the computation of the residual is defined as

$$r_{k+1} = V[y_{k+1} - \hat{y}_{k+1}] \tag{3.17}$$

where V is a weight matrix associated to the residual vector.

To achieve a successful fault detection based on the available residual signal, further efforts are needed. A widely-accepted way is to generate such a feature of the residual signal, by which we are able to distinguish the faults from the disturbances and uncertainties. Residual evaluation and threshold setting serve for this purpose. A decision on the possible occurrence of a fault will then be made by means of a simple comparison between the residual feature and the threshold. Depending on the type of the system under consideration (i.e., deterministic or stochastic), there exist two residual evaluation strategies: norm based residual evaluation and statistic testing. Driven by the process input signal, the value or average value or the energy of the process output may become very large.

It is assumed that for the fault detection purpose a residual vector, r is available. Next, we describe some standard evaluation functions that are a generalization of the above-mentioned evaluation functions of J:

Peak value. The peak value of residual signal is defined as r_k:

$$J_{peak} = \|r\|_{peak} := \sup_{k\geq 0} \|r_k\| \tag{3.18}$$

$$\|r_k\| = \sqrt{\sum_{i=1}^{k_r} r_{i,k}^2}. \tag{3.19}$$

Average value.

$$J_{average} = \|r\|_{average} := \sup_{k\geq 0} \|\bar{r}_k\|_{peak} \tag{3.20}$$

$$\|\bar{r}_k\| = \frac{1}{N}\sum_{j=1}^{N} r_{(k+j)}. \tag{3.21}$$

From an engineering point of view, the computation of a threshold corresponds to determine the tolerant limit for disturbances and model uncertainties under fault-free operation conditions. There are several factors that can significantly influence this procedure. Among them are:

- the dynamics of the residual generator;
- the way of evaluating the unknown inputs (disturbances) and model uncertainties;
- the bounds of the unknown inputs and model uncertainties.

A threshold can be generally defined by

$$J_{th} = \sup_{f=0,d,\Delta} J \tag{3.22}$$

where Δ denotes the model uncertainties and J the feature of the residual signal.

Concerning system with uncertainty, the most common residual evaluator is implemented using a χ^2-detector: the weighted power of residual $g_k = r_k S^{-1} r_k^T$ is compared with a threshold β defined to accept a certain error.

The following decision rule is applied:

$$R_k = \begin{cases} \mathscr{H}_0 & \text{if } s_k \leq \beta \\ \mathscr{H}_1 & \text{if } s_k > \beta \end{cases} \tag{3.23}$$

where $\mathscr{H}_0$ is the healthy and $\mathscr{H}_1$ is the faulty hypothesis, respectively. When $\mathscr{H}_1$ is accepted, the FDS triggers an alarm.

3.2.7 Security Information and Event Management

The SIEM system is considered in the global schema for the analysis of the specific alerts coming from the modules. This information is encoded using the Intrusion Detection Message Event Format.

3.3 SDN Controller

The SDN [113] is a network architectural paradigm consisting in the *control plane* separation from the *data plane*. The first one manages the network traffic forwarding, while the second one effectively forwards data to the connected nodes. This separation can lead to a large number of benefits, for example, as addressed in this work, the development of new features of cyber security. There are several implementation possibilities with SDN. For the sake of this work, a network packet flow control has been employed. With respect to conventional network traffic filtering mechanisms, the SDN technique allows a granular control of the traffic flows directly inside specific *OpenFlow* switches: with precise and defined rules with an high level of detail it is possible to carry out traffic forwarding operations acting also on the application layer. For this reason, forwarding rules related to specific protocols for ICS networks, such as the Modbus/TCP, can be implemented.

The SDN Controller, indeed, represents the crucial point of the whole architecture, since it manages communication load through switches/routers in the network. This module represents the brain of the communication system. It is important to notice that legacy switches and routers are not compatible with SDN technology. For this reason, specific *OpenFlow* devices must be used to develop the security architecture conceived.

Therefore, the integration of the SDN technology inside the Deep Detection Architecture adds value and degrees of freedom for future advanced cyber security implementations.

3.4 Mimepot

Mimepot is an evolved honeypot able to simulate physical processes and control routines of the plant. Comparing with the ICS honeypot presented in [75], the Mimepot focuses on the physical processes simulation, the related estimation and control, and the interconnection with the SDN functionalities.

A honeypot is formally defined as a monitored computing resource that can be probed, attacked, or compromised by malicious cyber actors [72]. Honeypots are by definition more vulnerable with respect to real systems, their target is to attract possible cyber-attacks replacing and safeguarding the real devices. The vulnerability is not the most relevant feature of Mimepot: it is considered a virtual system designed to replicate the cyber-physical structure of the plant. The network traffic coming from and to the Mimepot is not critical: to this end the control of the data flow performed by adopting SDN paradigm, is crucial. To this aim, the Mimepot operation is directly related to the SDN presence.

According to the Mimepot design, a point of strength respect to standard honeypots is the partitioning of physical processes reproduction from estimation and control routines (Fig. 3.7). The key idea is the attacker analysis: when an adversary is able to reach to target network, he/she will be lured easily. To this end, the Mimepot needs to be a realistic and attractive target for a potential malicious actor. According to the models presented in

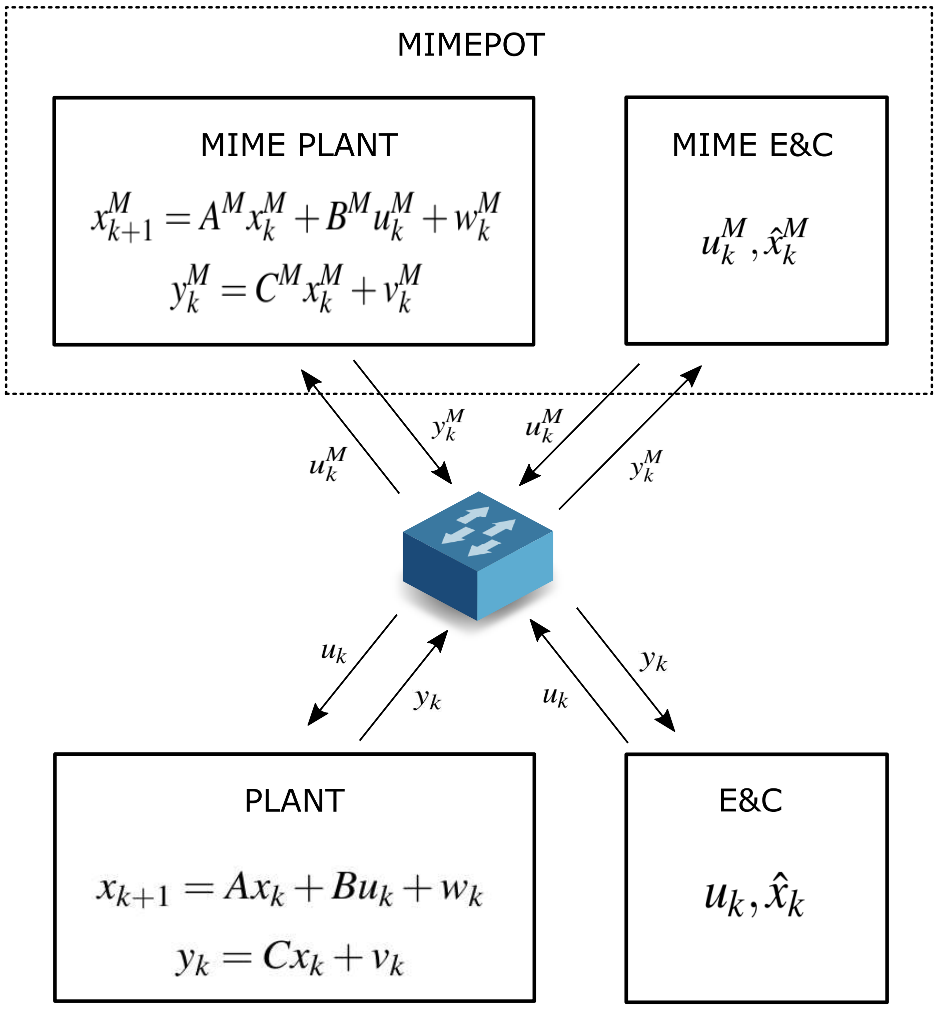


Fig. 3.7 Mimepot schema.

Chapter 2 and referring to schema in Fig. 3.7, the simulated plant state-space equations are:

$$x_{k+1}^M = A^M x_k^M + B^M u_k^M + w_k^M \tag{3.24}$$

$$y_k^M = C^M x_k^M + v_k^M \tag{3.25}$$

where $x^M \in \mathbb{R}^n$ is the mimed state vector, $u^M \in \mathbb{R}^p$ is the mimed control vector, and $y^M \in \mathbb{R}^q$ is the mimed output vector. Matrices A^M, B^M, C^M, D^M are real constant matrices, $w^M(\cdot) \sim N(0,Q)$ and $v^M(\cdot) \sim N(0,R)$ are identical independent Gaussian noises with zero mean and variance respectively Q and R. The mimed control and estimation parameters, respectively u_k^M and $\hat{x}_k^M$, are designed according to the mimed plant.

Real plant and Estimator and Controller (E&C) models have been discussed in Chapter 2. The model design depends on the complexity of the real plant, however, it is worth noticing that it is not necessary to replicate the whole real plant: the attacker have to be lured regarding his attack conditions. To this end, it is important that the Mimepot data can be attributed to a traffic flow generated by a control system. Moreover, when the attacker gain the access to the Control zone network, the target of the Mimepot is to hide the real plant topology and configurations.

The controller manages and regulates the behavior of the plant processes according to project specifications. The estimator is necessary to update the control vectors. Basic control routines and the estimation of the system state are located at Level 1. Input of this module are the sensor reads y coming from the plant and the output are the control vectors u.

From the network communications point of view, the Mimepot module can be implemented in virtualized environments where the Mime Plant and the Mime E&C can communicate through channels that make use of industrial protocols, such as the Modbus/TCP. The simulated physical values are directly inserted inside application layer of specific network packets. In this way, the attacker is deceived during the preliminary reconnaissance stage thanks to fake but plausible physical processes behavior with values of sensors and actuators managed through real industrial communication protocols.

3.5 Decision Support System

An ad hoc DSS has been developed for the proposed security architecture. This module is able to provide an enhanced passive defense approach to the human operators. It is not possible, indeed, to exclude the human decisions in the critical security scenario of ICS: the experience of human operators in applying countermeasures is always fundamental. The concept of *Human-In-The-Loop* is tightly related to both physical and cyber security operations. In the proposed architecture, there are three possible and complementary paths for the information between modules for the data coming from the DDS containing alerts (Fig. 3.8). The first one, defined as *passive defense*, directly allows alerts to reach operators. The second path, namely *enhanced passive defense*, allows the operator to receive the alerts data coming from the DDS and the countermeasures suggestions provided by the DSS. Finally, the third path regards the *operative defense*. In this case, the alerts are directly computed by the SDN Controller module that starts to redirect traffic towards the Mimepot (*autonomous operative defense*) or are evaluated with the supervision of the operator, where to the operator is given the option to explicitly activate the redirection of network traffic to the Mimepot module (*semi-autonomous operative defense*).

The DSS is developed using the AHP method introduced in 2. As explained, the AHP method allows to select among different alternatives in presence of multiple criteria. This

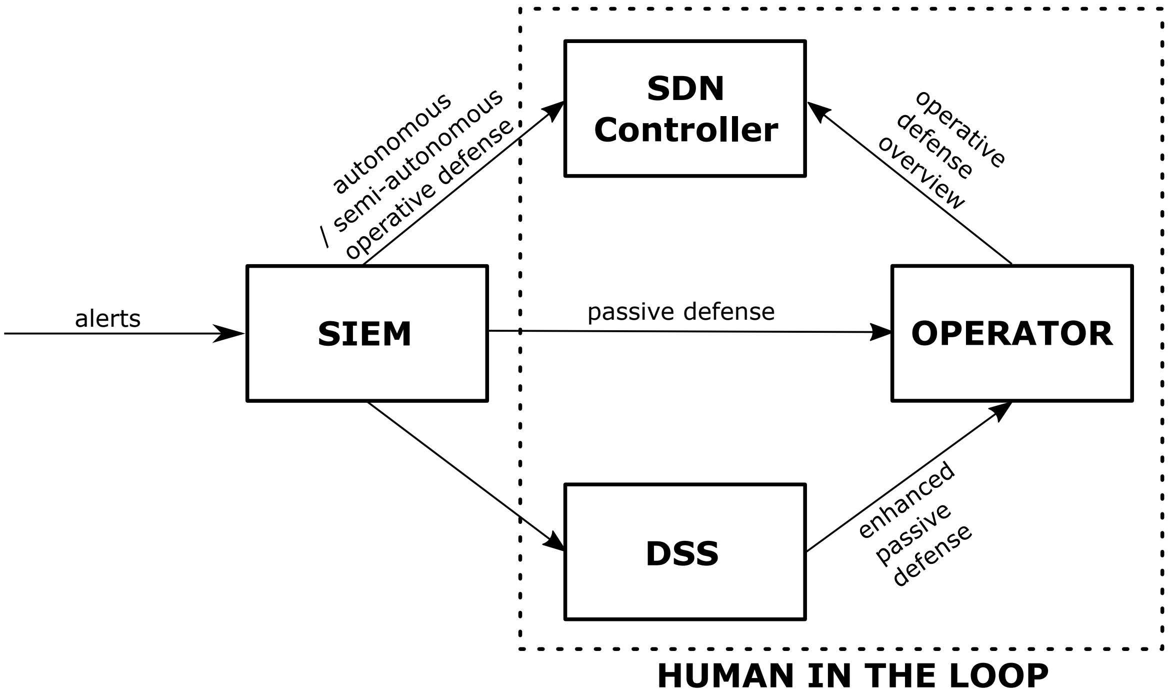


Fig. 3.8 Deep Detection with Human In The Loop.

technique allows to decompose the decision problem into its constituent elements, to structure them hierarchically considering the main objective and its sub-targets and finally to process data and opinions in order to achieve the goal. For the sake of the application, the criteria built for the assessment have been chosen as example in order to explain the methodology, considering the reliability and availability of control systems as main elements for the industrial sector. The network security operator goal is to determine the system components under cyber-attack detected by the IDS. Subsequently, he has to apply the best countermeasure to deal with the threat. In this context, an active support for the operator decision problem would be helpful. Considering the CI scenario, the final aim is to ensure the infrastructure reaction after a cyber-attack.

Chapter 4

Modeling, Hardware, and Software Tools

This chapter encompasses all the tools designed during the doctorate. Different tools for emulation and simulation have been implemented, to get insights on the behavior of CIs. These tools are crucial, since security architecture cannot be directly tested on real infrastructures. Besides that, a control theoretic perspective is adopted to model CPS, so ICS and cyber-attacks are regarded as hybrid systems. To this aim, this chapter introduced the models for CPS, the testbed HYDRA, developed to emulate a water distribution system, and a virtualization tool based on Mininet to simulate communications links. Some key references for this Chapter can be found in [107–109, 114]

4.1 Modeling Cyber-Physical Systems

Industrial Control Systems (ICS) are Cyber-Physical Systems (CPS) characterized by being geographically distributed. In fact, an ICS system can be considered composed of a physical and a cyber structure, as schematically represented in Figure 4.1. Specifically, the physical structure consists of the infrastructure/plant to be managed (e.g., all in-field devices), while the cyber structure encompasses the communication infrastructure and the elements used to supervise and manage the physical structure (e.g., the Human-Machine Interface (HMI)). The physical structure can be generally modeled as a nonlinear uncertain discrete-time system, where physical and anomalous cyber-events are represented as disturbances to the state and output functions, formalized as:

$$x_{k+1} = f(x_k, \tilde{u}_k) + \eta(x_k, d_{,k}, \tilde{u}_k) + w_{s,k} \tag{4.1}$$

$$y_k = h(x_k) + \Phi(x_k, d_k) + w_{o,k} \tag{4.2}$$

where:

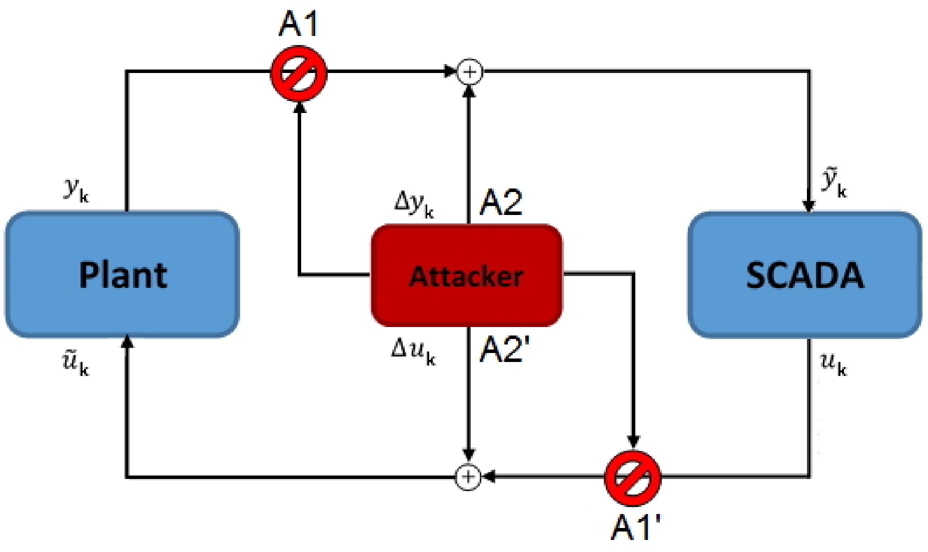


Fig. 4.1 Generic data flow between plant and SCADA system in an ICS. The attacker can interpose in the communication between plant and SCADA system, altering the data flow.

- $k \in \mathbb{N}$ is the time instant;
- $x \in \mathbb{R}^n$ represents the state vector;
- $\tilde{u} \in \mathbb{Q}^m$ is the input vector (e.g., m is the number of actuators), in which the effects of malicious inputs injected by cyber-attacks are also considered;
- $f : \mathbb{R}^n \times \mathbb{Q}^m \to \mathbb{R}^n$ models the nominal dynamics;
- $d \in \mathbb{R}^p$ are the physical faults that can affect the system;
- $\eta : \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{Q}^m \to \mathbb{R}^n$ is the cyber-physical attack/fault function affecting the state;
- $y \in \mathbb{R}^v$ is the output vector (i.e., v is the number of sensors);
- $h : \mathbb{R}^n \to \mathbb{R}^v$ is the nominal output map;
- $\Phi : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^v$ is the physical attack map affecting the output;
- $w_s \in \mathbb{R}^n$ represents the model uncertainties;
- $w_o \in \mathbb{R}^v$ corresponds to the noise vectors.

The plant, interacting with the SCADA, allows to perform control and monitoring actions. The general model of cyber actions generated by the SCADA system can be expressed as:

$$u_k = \beta(\tilde{y}_k, y_{ref,k}) \tag{4.3}$$

where:

- $\tilde{y} \in \mathbb{R}^v$ is the input vector for SCADA system, containing the measurements obtained from the field;
- $y_{ref} \in \mathbb{B}^m$ is the reference command from SCADA system;
- $\beta : \mathbb{R}^v \times \mathbb{B}^m \to \mathbb{Q}^m$ is the control function generated by the SCADA system;
- $u_k \in \mathbb{Q}^m$ is the input vector.

As depicted in Figure 4.1, the interface between the plant and the SCADA system performs the following matching:

$$\tilde{u}_k = u_k + \Delta u_k \tag{4.4}$$

$$\tilde{y}_k = y_k + \Delta y_k \tag{4.5}$$

where Δu_k and Δy_k are the injections to the input and output of the plant, respectively, resulting from cyber-physical anomalies. Under normal conditions, $\Delta u_k = 0$ and $\Delta y_k = 0$.

4.2 Modeling Attacks on Cyber-Physical Systems

The aim of a secure system is to grant data and resources availability, preventing unauthorized users access and protecting data integrity. On the contrary, the aim of an attacker is to take control or manipulate system parameters to generate and exploit threats, modify the behavior of the system, obtain and tamper data, or reduce data and resources availability. A cyber-attack may have several effects on the physical system and may downgrade its operability, performance, and efficiency. Considering the peculiarities of the frameworks in which ICSs operate, the hazards and the damages could be potentially extended to the environment or to human safety.

An attacker is assumed to be connected to the Control zone network and he/she is able to perform attacks against the CPS, creating atypical and unexpected situations within the plant. According to this assumption, all the necessary resources are available to the attacker. As shown in Figure 4.1, when a malicious agent gains access to the communication network, it becomes able to tamper the data exchanged between PLC and SCADA in different ways. Hence, the original signals u_k and y_k will turn into the attacked signals $\tilde{u}_k$ and $\tilde{y}_k$ ($A2$ and $A2'$ attacks), respectively, or will not reach the destination device at all ($A1$ and $A1'$ attacks). Two main classes of attacks have been addressed:

- ***Availability Attacks:*** conceived to reduce or disrupt the service capabilities of the target device. DoS attacks are an example for which the TCP/IP protocol vulnerabilities constitute the basis;
- ***Integrity Attacks:*** conceived to manipulate network data, such as sensor readings and actuator commands. The MITM attacks represent the practical approach, consisting in the hijacking of the data exchanged between two target hosts. The attacker places itself between them, receiving all the traffic generated by the victims and conveniently forwarding the packets to the right destination. To this end, the well-known Modbus/TCP vulnerabilities could be exploited because of the lack of authentication and encryption.

Depending on the specific goal pursued, the realized attacks can be single or non-interactive, or concurrent and/or coordinated. Single attacks consist of a set of sequential actions performed against the target for a given time interval and with a specific level of intensity, while concurrent attacks generally imply a strategic attack on several devices of the system. Attack models have been derived inspired by those proposed in [115] and by considering the definitions in Section 4.1.

4.2.1 Models for Availability Attacks

Ping Flooding. A *flooding attack* usually indicates a methodology aimed at disrupting the services of a machine connected to a network by forwarding to it a large number of packets in a short time lapse. In the case of *ping flooding*, the victim is overloaded with ICMP packets. If the attack is performed against the PLC between time instants k_s (i.e., beginning of the attack) and k_e (i.e., end of the attack), the input received by the SCADA will be:

$$\tilde{y}_k = \begin{cases} y_k & \text{for } k < k_s \text{ and } k > k_e, \\ y_{k-\tau} \vee [\,] & \text{for } k_s < k < k_e \end{cases} \tag{4.6}$$

where τ is the time delay introduced by the attack and [ ] indicates the lack of information due to the communication interruption, represented as $A1$ attack in Figure 4.1. Besides, as shown by the $A1'$ case, an attack performed against the SCADA between time instants k_s and k_e, can be modeled as:

$$\tilde{u}_k = \begin{cases} u_k & \text{for } k < k_s \text{ and } k > k_e, \\ u_{k-\tau} \vee [\,] & \text{for } k_s < k < k_e \end{cases} \tag{4.7}$$

Modbus Flooding. In this case, the goal is not to harm the network communication but to set out of order the target elaboration capabilities. Specifically, the first set of packets are received by the target device and rejected as they are not consistent with the current system operation (e.g., a corresponding request has not been previously received). After a short time interval, the target device is no longer able to cope with the wide number of Modbus packets received. This leads to the reduction of the elaboration capabilities of PLC/SCADA and to communication latency. The specific action coded on the data field is of any actual interest, as long as it is recognized as valid by the destination component. For this reason, it can be stated that there are no differences with the *ping flooding* model.

4.2.2 Models for Integrity Attacks

Data Modification. A malicious actor can manipulate the data field of the Modbus/TCP packets, modifying its content in various ways. If the attacker succeeds in gathering the proper information about the system, it may be able to perform undesirable actions against the system, such as changing the values shown on the HMI or the commands sent to the actuators. If an operator is not able to identify such illicit traffic, an escalation of threats may cause major problems, mainly on the physical layer, as the attacker would be able to perform actions on the field. To do so, the attacker needs specific knowledge about the communication protocol deployed, the structure of the data field of the packets, and accurate information about the system to be compromised. Although such hypothesis may sound quite restrictive, sufficient knowledge could be obtained by sniffing and analyzing the network traffic for an adequate time lapse.

While exploiting a MITM to perform a *Data Modification* attack the communication seems normal, as the tampered packet is syntactically correct and semantically valid. The effect depends on which type of packet is tampered and on the target of the attack. If it carries data regarding the sensors measurements at time step k_i and it is performed against the PLC, the attack model becomes:

$$\tilde{y}_k = \begin{cases} y_k + \Delta y_k & \text{for } k = k_i, \\ y_k & \text{otherwise.} \end{cases} \tag{4.8}$$

Analogously, when the attack is performed against the SCADA:

$$\tilde{u}_k = \begin{cases} u_k + \Delta u_k & \text{for } k = k_i, \\ u_k & \text{otherwise.} \end{cases} \tag{4.9}$$

Replay Attack. A particular case of *Data Modification* attack is the *Replay Attack*, whose execution is divided into two main phases. First, the Modbus data flow between target machines is recorded for an arbitrary time lapse. Then, it is repeatedly re-sent to the desired device, replacing the actual data fields. Thereby, the plant and/or the SCADA will receive and process old information, repeating the previous actions again. If this attack is properly designed, the operation parameters would not be out of the allowed range, making the attack go unnoticed by the operator. In such a case, the model for the attack that has the PLC and the SCADA as target is similar to the *Data Modification* attack:

$$\tilde{y}_k = \begin{cases} y_k + \Delta y_k & k_s < k < k_e, \\ y_k & \text{otherwise} \end{cases} \tag{4.10}$$

$$\tilde{u}_k = \begin{cases} u_k + \Delta u_k & k_s < k < k_e, \\ u_k & \text{otherwise} \end{cases} \tag{4.11}$$

where $y_{k-\tau}$ and $u_{k-\tau}$ are the sensor measurements and the control signals previously recorded, while $\Delta u_k = -u_k + u_{k-\tau}$ and $\Delta y_k = -y_k + y_{k-\tau}$.

4.3 Cyber-Physical Simulation Using Matlab/Simulink

The CI addressed during the doctorate are water distribution systems. To this end, several simulation tools have been adopted to verify and validate the security architecture. In the following, the simulation tools developed in Matlab/Simulink are detailed. These models have been used to validate the algorithms in the FDS and in the Mimepot.

Preliminary case study: water tower simulation. A water tower is a structure located in an elevated place to provide drinkable water to costumers. This infrastructure is able to provide water also in emergency situation (e.g., without electric energy), since its operations are based on gravity. The system simulation is represented by filling up and emptying the tank according to physical laws. In Fig. 4.2, a simplified water tower is shown. The continuous time relations describing the process of filling up and emptying the tank are:

$$A\dot{h} = Q_{IN} - Q_{OUT} \tag{4.12}$$

$$Q_{OUT} = a\sqrt{2gh} \tag{4.13}$$

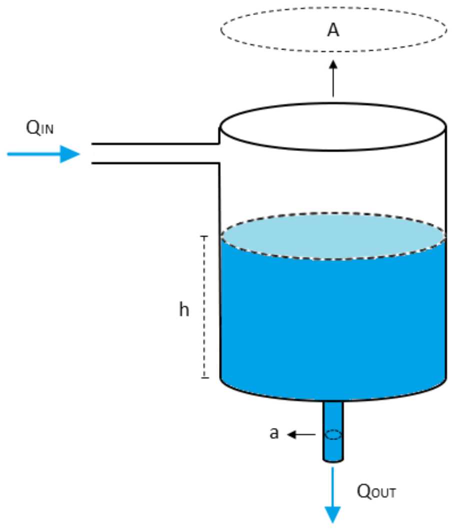


Fig. 4.2 Tank considered for the scenario.

$$\dot{h} = \frac{Q_{IN}}{A} - \frac{a\sqrt{2gh}}{A} \tag{4.14}$$

where Q_{IN} and Q_{OUT} are the incoming and outgoing flows. Q_{IN} is considered as constant, expressed in m^3/s. Q_{OUT} is calculated according to physical laws. A and a are respectively the tank and the output hole sections. h represents the water level and g is the gravitational acceleration. The model of the system has been created and validated using Simulink (see Fig. 4.3).

Single tank with proportional pump and water consumption. In this case, shown in Fig. 4.4, a single tank with proportional water pump and water consumption is shown. The system is described by the following relations:

$$Q_{IN} = \alpha \cdot P \tag{4.15}$$

$$Q_{OUT} = \beta \cdot a\sqrt{2gh} \tag{4.16}$$

$$\dot{h} = \frac{Q_{IN}}{A} - \frac{Q_{OUT}}{A} \tag{4.17}$$

where Q_{IN} is the incoming flow represented by a constant P, expressed in m^3/s, which represents the pump maximum flow, while $\alpha \in [0,1]$ is a parameter which allows to proportionally scale the pump flow value. The output flow, Q_{OUT}, is computed according to physical laws. Also Q_{OUT} is scaled according to $\beta \in [0,1]$, a parameter representing the

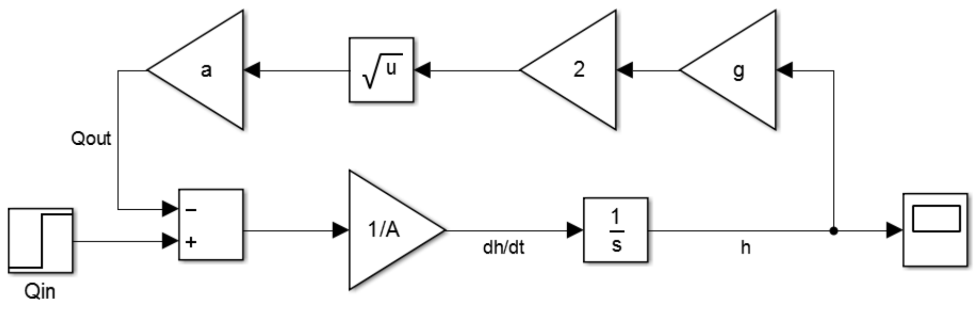


Fig. 4.3 Simulink model.

different water consumption rate. The constant value A is the tank section and the constant value a is the output hole section. The water level inside the tank is characterized by h and g is the gravitational acceleration. In Fig. 4.5, the Simulink model of this scenario is outlined.

Reservoir with two water consumers. A more complex scenario is presented in Fig. 4.7. Here, one tank is placed in a higher position with respect to the others and represents a reservoir ($T1$). It is refilled by a water pump. The remaining two tanks represent water consumers ($T2$ and $T3$), having different consumption rate. Moreover, it is possible to model different proportional water flows from the reservoir to the consumers.

The corresponding continuous time equations are:

$$Q_{IN1} = \alpha \cdot P \tag{4.18}$$

$$Q_{OUT11} = \beta_{11} \cdot a_{11}\sqrt{2gh_1} \tag{4.19}$$

$$Q_{OUT12} = \beta_{12} \cdot a_{12}\sqrt{2gh_1} \tag{4.20}$$

$$\dot{h_1} = \frac{Q_{IN}}{A_1} - \frac{Q_{OUT11}}{A_1} - \frac{Q_{OUT12}}{A_1} \tag{4.21}$$

$$Q_{IN2} = Q_{OUT11} \tag{4.22}$$

$$Q_{OUT2} = \beta_2 \cdot a_2\sqrt{2gh_2} \tag{4.23}$$

$$\dot{h_2} = \frac{Q_{IN2}}{A_2} - \frac{Q_{OUT2}}{A_2} \tag{4.24}$$

$$Q_{IN3} = Q_{OUT12} \tag{4.25}$$

$$Q_{OUT3} = \beta_3 \cdot a_3\sqrt{2gh_3} \tag{4.26}$$

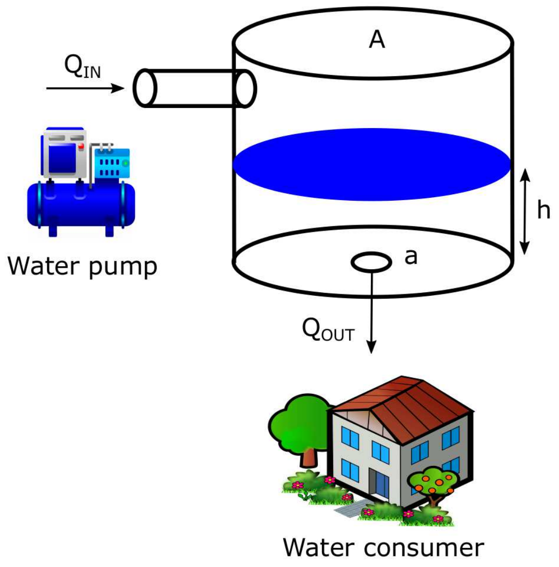


Fig. 4.4 Single tank with proportional pump and water consumer schema.

$$\dot{h}_3 = \frac{Q_{IN3}}{A_3} - \frac{Q_{OUT3}}{A_3} \tag{4.27}$$

where the gravitational acceleration is represented by g and the variables, referring to each tank, are described in Table 4.1. The Simulink model for this scenario is sketched in Fig. 4.7. An example of physical processes evolution associated with this scenario is depicted in Fig. 4.8 and lasts for 100 s. The corresponding parameters are described in Table 4.2.

4.4 HYDRA Testbed

The HYDRA testbed is a low-cost and open-source emulator for ICS. This testbed has been developed and realized starting from the scratch. The core of the HYDRA testbed is represented by a physical layer that emulates the behavior of a simple water distribution system, without considering pressure effect. The cyber layer is composed of the Control System, the HMI, and a DDS including the FDS and the S-IDS. All these modules are connected in a LAN, used to collect data from sensors and actuators, as in a real industrial control system. The HYDRA testbed can be used to emulate different scenarios: for example, it is possible to reproduce the water consumption in a small city over a day or the operation of cooling systems. Due to the modularity of the physical layer, indeed, it is possible to

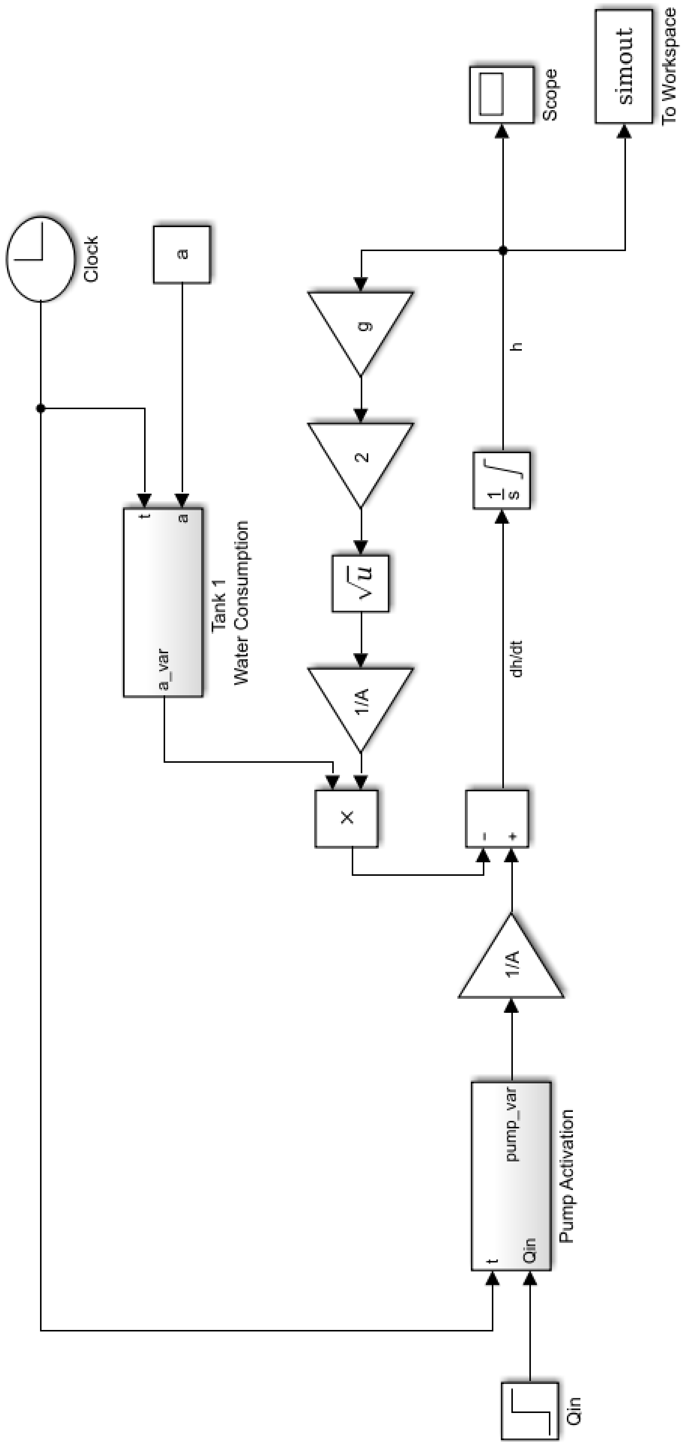


Fig. 4.5 Single tank Simulink model with water consumption and pump activation.

Table 4.1 Three tanks scenario variables description.

Reservoir – Tank 1 (T_1)	
A_1	Tank section
a_{11}	Hole to T_2 section
a_{12}	Hole to T_3 section
Q_{IN1}	Input flow
Q_{OUT1}	Output flow to T_2
Q_{OUT2}	Output flow to T_3
h_1	Tank water level
P	Pump flow constant value (m^3/s)
α	Pump proportional value ($\alpha \in [0,1]$)
β_{11}	$a_1 1$ proportional value ($\beta_{11} \in [0,1]$)
β_{12}	$a_1 2$ proportional value ($\beta_{12} \in [0,1]$)

Water consumer 1 – Tank 2 (T_2)	
A_2	Tank section
a_2	Output hole section
Q_{IN2}	Input flow T_2
Q_{OUT2}	Output flow T_2
h_2	Tank water level
β_2	a_2 proportional value ($\beta_2 \in [0,1]$)

Water consumer 2 – Tank 3 (T_3)	
A_3	Tank section
a_3	Output hole section
Q_{IN3}	Input flow T_3
Q_{OUT3}	Output flow T_3
h_3	Tank water level
β_3	a_3 proportional value ($\beta_3 \in [0,1]$)

Table 4.2 Reservoir with two water consumers simulation example parameters.

{Reservoir} – Tank 1 (T_1)	
A_1	20 m
$a_1 1$	0.5 m
$a_1 2$	0.3 m
$h_1(0)$	15 m
P	10 m^3/s
α	1
β_{11}	1
β_{12}	1

{Water consumer 1} – Tank 2 (T_2)	
A_2	10 m
a_2	0.2 m
$h_2(0)$	5 m
β_2	1

{Water consumer 2} – Tank 3 (T_3)	
A_3	5 m
a_3	0.1 m
$h_3(0)$	3 m
β_3	1

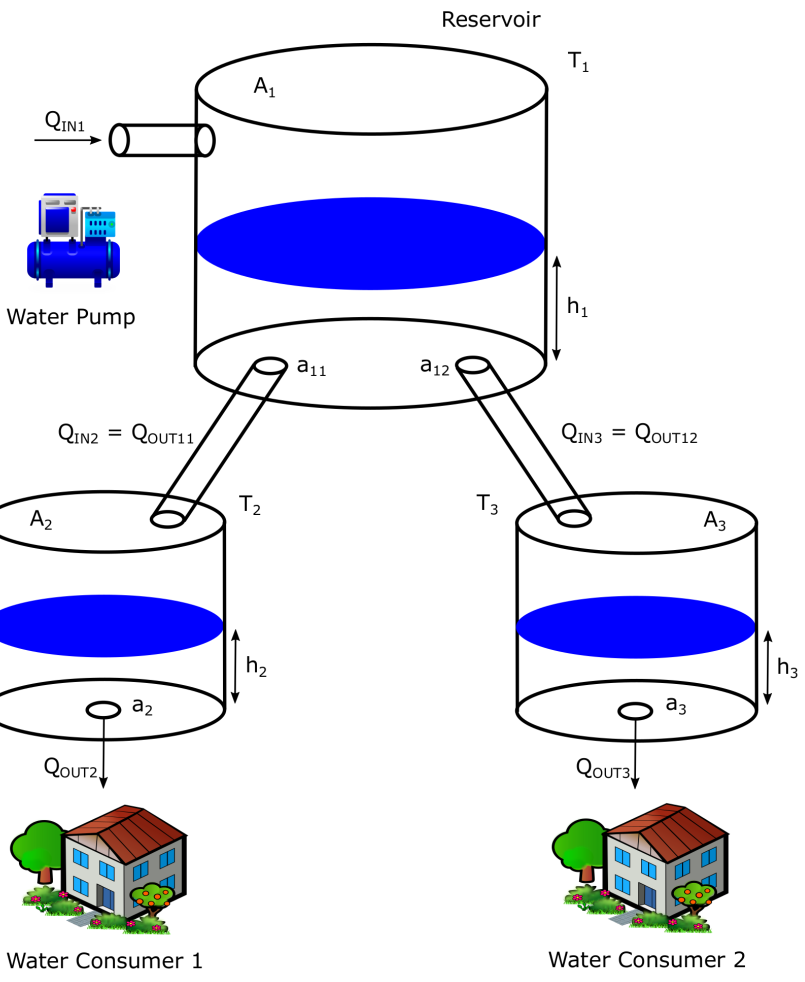


Fig. 4.6 Reservoir with two water consumer tanks schema.

set up different configurations and switch between them automatically. In the design of the testbed a low cost approach has been used, thereafter HYDRA is equipped with low cost devices that emulate expensive industrial equipment. Each component of HYDRA will be introduced later, omitting the HMI, which represents just a cockpit for the CI operator.

4.4.1 Physical Layer

The HYDRA testbed is composed by 7 tanks, deployed in a vertical fashion as depicted in Fig. 4.9. Specifically, a large tank is on the top simulating a water tank tower in urban scenario, while 6 tanks are placed on 3 different levels (see Fig. 4.10). All the tanks are equipped with pressure sensors to detect the water level. At the lowest level is placed a

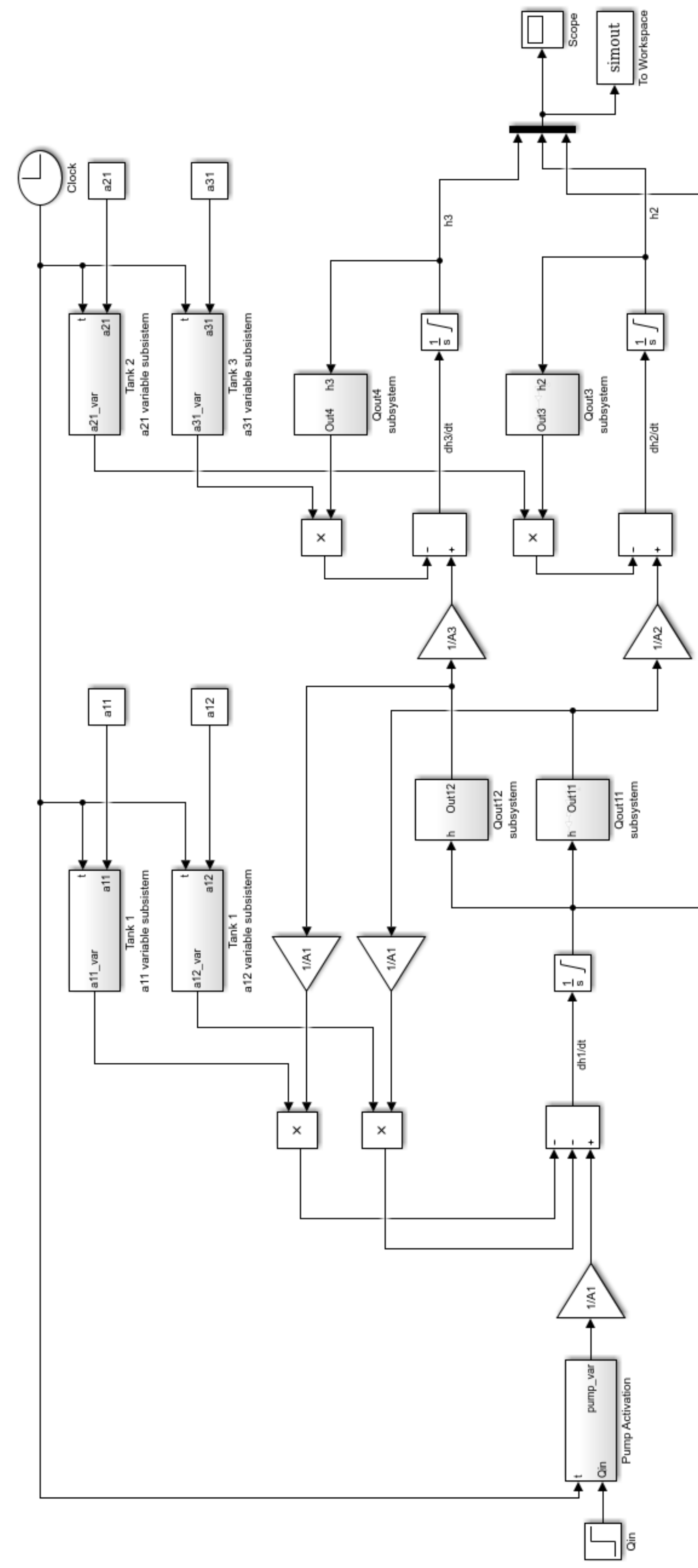


Fig. 4.7 Three tanks Simulink model with water consumption and pump activation.

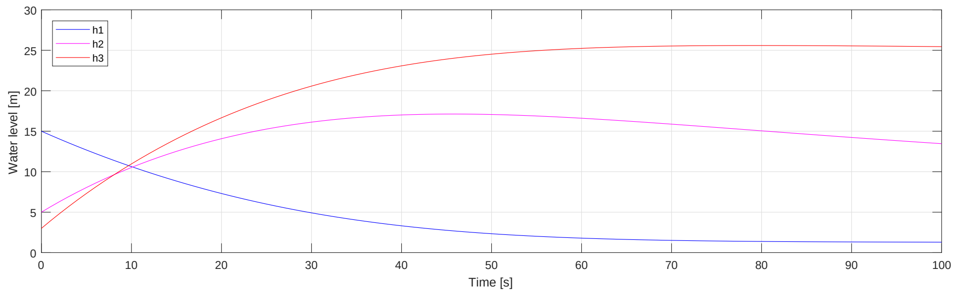


Fig. 4.8 Reservoir with two water consumers example.

reservoir, which simulates the presence of an aquifer. The tanks are connected by pipes, which are not under pressure, so the water flows from tanks exploiting pumps, gravity, and Stevin's Law. Respectively, a submersible pump supplies water from the reservoir to the water tank tower and two other external micro pumps move water from the second to the third level. Two water flow meters allow to detect and measure the presence of water in the pipes. To emulate water consumption, 8 electromechanical valves are deployed on the pipes, while 7 manual valves are used to emulate leakages. Different configurations can be achieved by closing and opening valves and moving pipes, to simulate different redundancies and reduce the complexity of the model. All the sensors and actuators are connected to the LAN by means of Galileo Gen2 and Arduino Nano board that emulate PLC.

4.4.2 Control and Fault Detection Systems

The control system designed for the HYDRA testbed is divided in two levels. The low level control maintains the water level between minimum and maximum thresholds. The testbed is configured to reproduce the behavior of a small city, simulating the water consumption cycle during a day in both residential and industrial area. To preserve the maximum efficiency of the two areas an high level control is implemented. Pumps and valves are actuated in order to refill the water reserve based on demand. As previously said, in CPS, faults caused by mechanical damaged or malicious intrusions must be monitored and identified in order to preserve the efficiency of the system and the health of the users. For this reason a fault detection module is implemented on the testbed, based on the model of the system. To this end, the functioning of the physical layer is simulated using the model revised in Section 4.1. The model is fed with the input given by the control system to the testbed and its output is compared with the real one in order to generate residuals and, eventually, alarms based on fixed threshold.

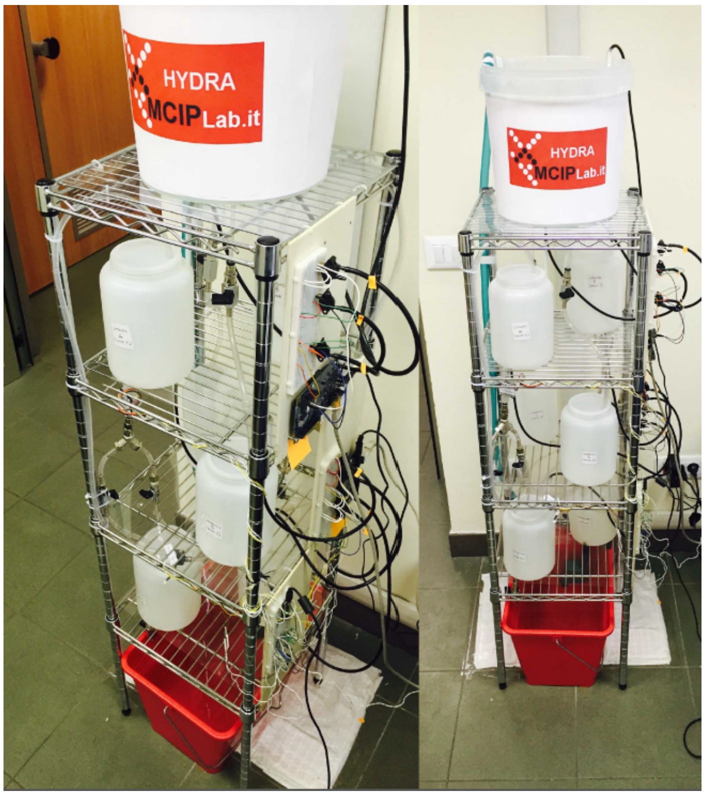


Fig. 4.9 The HYDRA water system testbed.

The overall model of the testbed is complex, here, for sake of clarity, only the subsystem shown in Fig. 4.11 is considered. It represents a building block: it is composed by three tanks deployed so to have both horizontal and vertical configuration. Thus, the overall system model can be easily retrieved by suitably composing the equations of this subsystem. The continuous time equations used to model the subsystem are based on mass balances, Bernoulli's and Stevin's laws:

$$\begin{aligned}
\dot{h}_3 = e_A [&U(h_4 - h_{con})\,U(h_{con} - h_3)\,\frac{a_f}{A}\sqrt{2g(h_4 - h_{con})} \\
&- U(h_3 - h_{con})\,U(h_{con} - h_4)\frac{a_f}{A}\sqrt{2g(h_3 - h_{con})} \\
&+ U(h_3 - h_{con})\,U(h_4 - h_{con}) sgn(h_4 - h_3)\frac{a_f}{A}\sqrt{2g\,|(h_4 - h_3)|}]
\end{aligned} \tag{4.28}$$

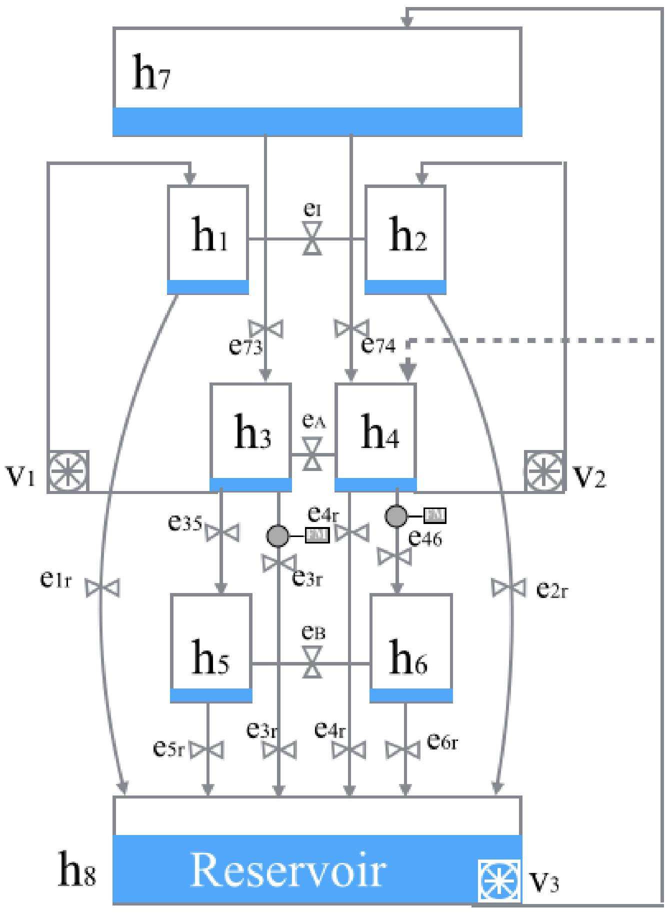


Fig. 4.10 The HYDRA testbed structure.

$$
\begin{aligned}
\dot{h}_4 = e_A[&U(h_3 - h_{con})U(h_{con} - h_4)\frac{a_f}{A}\sqrt{2g(h_3 - h_{con})} \\
&- U(h_4 - h_{con})U(h_{con} - h_3)\frac{a_f}{A}\sqrt{2g(h_4 - h_{con})} \\
&+ U(h_4 - h_{con})U(h_3 - h_{con})sgn(h_3 - h_4)\frac{a_f}{A}\sqrt{2g|(h_4 - h_3)|}] \\
&+ \frac{k_3}{A}v_3 - e_{46}\frac{a_f}{A}\sqrt{2gh_4}
\end{aligned}
\tag{4.29}
$$

$$
\dot{h}_6 = e_{46}\frac{a_f}{A}\sqrt{2gh_4} - e_{6r}\frac{a_f}{A}\sqrt{2gh_6} \tag{4.30}
$$

where h_i is the water level in the tank T_i $(i \in \{1...7\})$ over time, e_A, e_{46}, and e_{6r} are respectively the states of the horizontal valve, the valve between the tanks T_4 and T_6 and between the tank T_6 and the reservoir (1: Open, 0: Close), $A = 127\ cm^2$ is the cross-section of the tank, $a_f = 0.1256\ cm^2$ is the cross-section of the outlet hole and h_{con} is the connection height

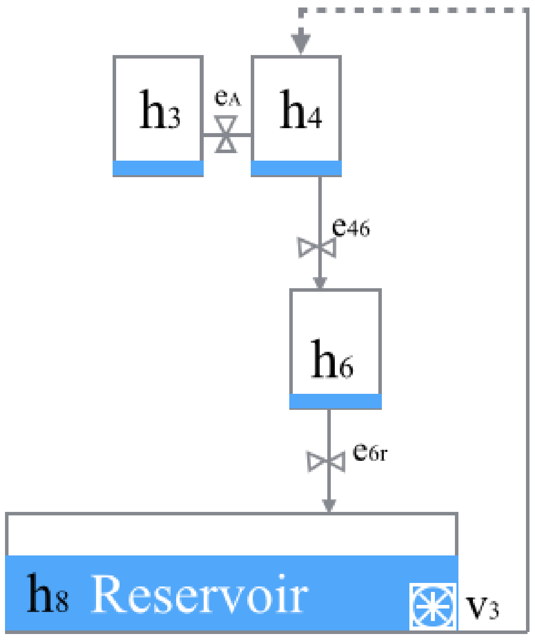


Fig. 4.11 Modeled subsystem of the testbed.

in case of communicating vessels. The acceleration of gravity is denoted with g expressed in $\frac{cm}{s^2}$. The voltage supplied to the pump is v_3 and the corresponding flow is $k_3 v_3$ where k_3 is a constant expressed in $\frac{cm^3}{Vs}$. The functions $U(\cdot)$ is defined as follows:

$$U(x) = \begin{cases} 1 & if \quad x \geq 0 \\ 0 & if \quad x < 0 \end{cases} \tag{4.31}$$

4.4.3 Signature-based Intrusion Detection System and Network Setup

On HYDRA testbed, a signature-based IDS has been enforced: a set of rules of known attacks is used to find suspicious activities in the current network traffic. It supervises the traffic of the whole network, checking locally and remotely interchanged packets. The S-IDS software developed for the testbed is based on Snort [35], an open source network Intrusion Detection System. The S-IDS performs real-time traffic and protocol analysis and packet logging on IP networks.

An ad-hoc network has been set and the topology is illustrated in Figure 4.12. The PLC and HMI, that emulate SCADA components, are connected to the central switch. The S-IDS has been connected to data mirror port on the switch, to analyze all the traffic present on the Control zone network. Finally, it is assumed that the attacker is connected to the same

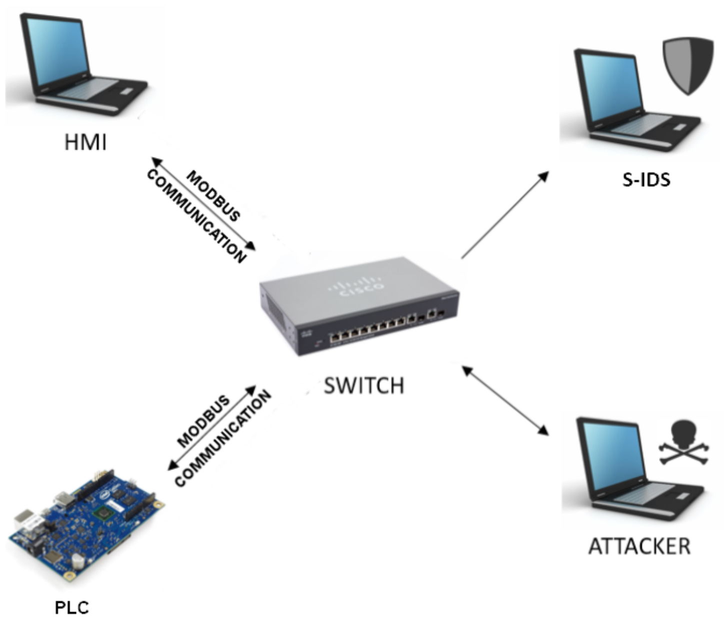


Fig. 4.12 Network topology.

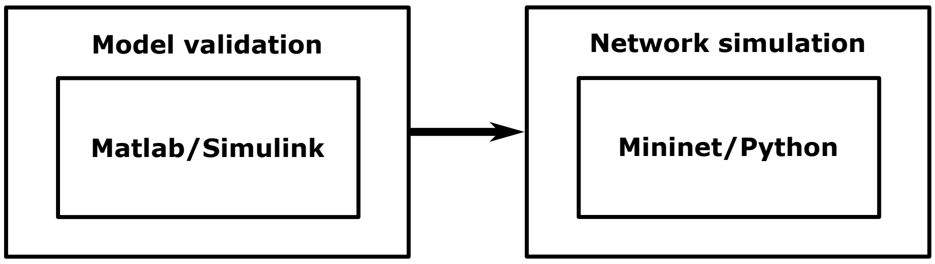


Fig. 4.13 From model validation to network simulation.

network. In order to replicate a real industrial scenario also the network setup reproduces the configuration of a real Modbus/TCP network. Using the Node.js framework [116], a runtime environment for client-server communications, the Modbus/TCP protocol has been implemented in the HYDRA testbed for the communication between the HMI and the PLC. Pumps, water level sensors, and flow meters are controlled and connected in full compliance with the Modbus/TCP protocol and the own packet structure reproducing the presence of the function codes. For further information on Modbus/TCP protocol, the reader can refer to A.

4.5 Cyber-Physical Simulation Using Mininet/Python

The purpose of the theoretical model presented in Section 4.1 is to create a solid base that allows to simulate CPS networks. However, models provide analysis of physical processes

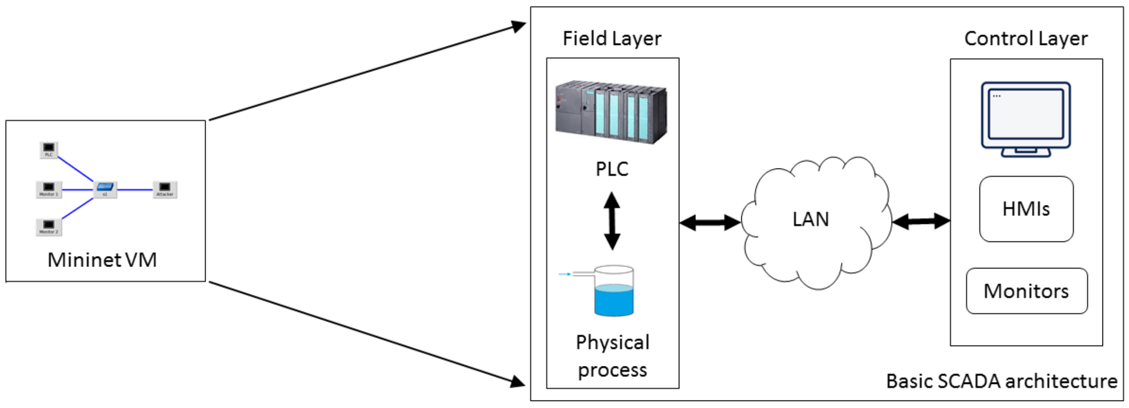
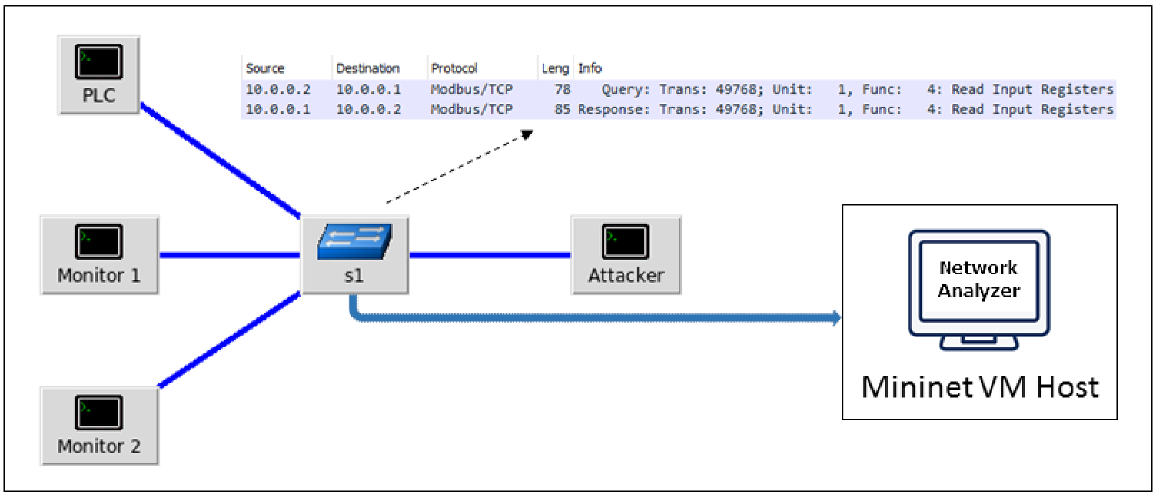


Fig. 4.14 SCADA system to Mininet VM, the basic architecture implemented.

but fails in evaluate the behavior of the network during cyber-attacks. In the literature, the research approaches consider modeling of cyber-attack to provide evaluation: here, a novel perspective, as presented also in [117], is adopted and the system is simulated using also the communication links. To this aim (see Fig. 4.13), the tools in Section 4.3 are used to create the simulation of the physical layer. Moreover, the models are imported into a virtual network able to recreate different connected nodes. In this way, it is possible to separate physical processes from network components into an integrated simulated environment. Mininet [118] and python scripting are devoted to network simulation.

Mininet is a virtual network running on a single machine used for generic communication system simulations and it represents a useful tool for research and development in the cyber domain. Using the Mininet Virtual Machine (VM), it is possible to simulate multiple nodes on a network and connect them with virtual links and switches. Every node simulates a stand-alone machine with its own network features. The versatility of Mininet grants to simulate complex network systems, using several communication protocols. The peculiar features of the nodes connected to the network are developed in python scripting and all the tools installed on the Mininet host can be used by the simulated network nodes. This is an innovative approach to evaluate cyber-threats for CPS using simulation tools.

A simple scenario, based on the water tower described in Section 4.3 is presented. In order to reproduce the behavior of a water system infrastructure, a SCADA system has been considered as monitoring and control architecture for the simulated industrial scenario. In Fig. 4.14 the implemented SCADA architecture is shown. The physical processes have been developed using Matlab/Simulink and then exported to python scripts to make possible implementation on the Mininet VM. Each component of the architecture is simulated by network nodes on the Mininet VM.



```
Source      Destination  Protocol     Leng Info
10.0.0.2    10.0.0.1     Modbus/TCP    78     Query: Trans: 49768; Unit:   1, Func:   4: Read Input Registers
10.0.0.1    10.0.0.2     Modbus/TCP    85 Response: Trans: 49768; Unit:   1, Func:   4: Read Input Registers
```

Fig. 4.15 Mininet topology.

The Field Layer is represented by the physical process and the PLC. The latter is connected to the water level sensors and it communicates with the Control layer by a LAN. The Control layer is composed of a HMI and Monitors. This simple architecture has been designed to verify effectiveness of Mininet in simulating a SCADA system. According to this approach, more complex networked ICSs can be considered. Besides the analogues reads simulated inside the PLC node, all the communications between the nodes have been implemented using the Modbus over TCP [111]. The Modbus/TCP has been selected and implemented for the communication routines to create a more realistic simulation.

The network topology, including the attacker, is shown in Fig. 4.15. The network is composed by a PLC, configured as Modbus/TCP Server, two Monitors set as Modbus/TCP Clients, and a legacy switch enabling the communications among nodes. A network analyzer has been implemented on the Mininet VM host. In this way, the host is able to inspect traffic without being part of the guest network simulating an analyzer connected to the mirroring port of a switch.

For the physical process the following parameters have been set: $Q_{IN} = 10\ m^3/s$, $A = 20\ m^2$, $a = 0.5\ m^2$. The simulation lasts 60 s and it is depicted in Fig. 4.16. Concerning the network communications, only the water level of the tank is monitored: to this end, industrial level sensors in the field layer are connected to the PLC, which controls the data. The PLC polls every second the value of the level sensor and forwards them to the Monitors by using Modbus/TCP protocol. Therefore, at each sampling time the Clients send a query to the Server in order to receive the sensor reads. The Modbus Function Code implemented for the Query/Response operations is the *Read Input Registers*.

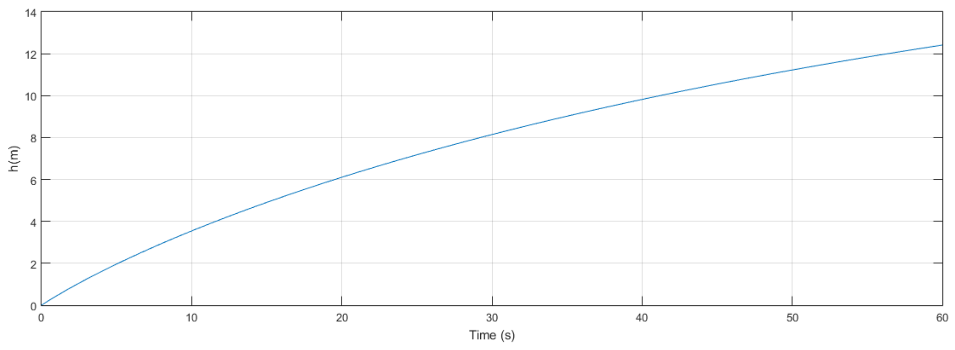


Fig. 4.16 Water level evolution simulation.

Chapter 5

Experimental Results

The challenges arising in CPS security investigation are twofold. In the one side, it is interesting to address the side effects of cyber-attacks on the physical structure of the system. On the other side, the simulation/emulation of both CPS and attacks is crucial to get insights on the CPS feedback to cyber threats and faults. Moreover, The human-in-the-Loop paradigm is fundamental in monitoring CIs: the human operator is the only decision maker when emergencies, faults or cyber-attacks occur. To this aim, in this chapter fault detection approaches are deeply analyzed to highlight pros and cons of detection schemes. Different cyber-attacks have been implemented on emulated operational scenarios: two different testbeds have been exploited to replicate in safe condition the effects of cyber threats. A simple DDS is proposed and its effectiveness proved. The main references are [108, 109, 119, 114, 120].

5.1 Fault Detection System for FACIES Testbed

Extensive experimental tests have been carried out using the testbed described in Appendix A. The 5 tanks are used to emulate a typical 24-hours water demand (i.e., the *Healthy run*). The scenario is scaled down to 6 minutes and it is implemented through the sequential activation of valves and pumps. Only few trials are reported. In most of them, the attack starts at $k_s = 100$ s. As previously mentioned, it is assumed that the attacker has already gained access to the Control zone network. Moreover, the attacker has obtained sufficient knowledge through sniffing or other malicious actions, becoming able to perform the cyber-attacks described in the followings. To validate the methodology and the results, two different attacker machines have been deployed for the tests, whose main features are compared in Table 5.1. To perform the cyber-attacks, the Kali Linux operating system and some of its tools have been exploited. Concerning the MITM attack technique, the ARP spoofing

Table 5.1 Specifications of the attacker computers.

	Attacker 1	**Attacker 2**
CPU	Intel®Core^{TM} i5-3317U @1.7GHz	Intel®Core^{TM} i7-2670QM @2.20GHz
RAM	6 GB	6 GB
NIC	Atheros AR8166 PCI-E	Controller Realtek PCIe GBE Family
Virtual Machine	VMware® Player v7.1.0	VMware® Workstation 12 Player v12.0.0
Host OS	Windows® 7	Windows® 10 Pro

methodology has been employed: the attacks exploit the ARP reply receiving mechanism to modify the ARP CACHE of the victims. Therefore, false ARP reply messages are sent to the two machines under attack and the Media Access Control (MAC) address of the attacker machine is inserted into the ARP replies. Hence, the packets supposed to travel between the victims are actually sent to the attacker. *Ettercap* and *Etterfilter*, introduced in 2, have been used to modify the packets exchanged between the targets.

During the attacks, the FDS module of the DDS is running to identify anomalies. The goal of these tests is twofold. On the one hand, it aims at validating the attack methodology on an emulated system. On the other hand, it highlights the limitations of FDS techniques in anomaly identification when a cyber-attack occurs in an ICS. Thereby, a specific S-IDS has been developed to eliminate ambiguities between cyber and physical issues, in order to fill the gap of classical FDS approaches facing cyber-threats. The joint exploitation of S-IDS and FDS represents one of the novelty of the proposed approach.

5.1.1 The Fault Detection System for FACIES Testbed

The main purpose of the FDS is to use a simulated dynamic model of the plant/infrastructure. The model is fed with the same inputs provided to the real system in order to compare the model outputs with the measurements acquired from the field. The occurrence of a significant deviation from the foreseen behavior of the model and the actual values emphasizes the presence of a fault in the physical plant. Moreover, under suitable hypothesis, the FDS may also identify the faulted component and suggest to the operator the possible mitigation strategies. Due to the nonlinearity of the system proposed as case study, the FDS module is based on the following nonlinear discrete-time model for the nominal plant operation:

$$\begin{aligned}
x_{1,k+1} &= x_{1,k}+\tfrac{T_s}{A_1}\Big(p_{1,k}-p_{5,k}-S\cdot\Big[(c_1^m(v_{1,k}^m)+c_1^s(v_{1,k}^s))\cdot\sqrt{2gx_{1,k}} \\
&\quad + c_1^c(v_{1,k}^c)\cdot\sqrt{2g\cdot|x_{1,k}-x_{2,k}|}\cdot\operatorname{sign}(x_{1,k}-x_{2,k})\Big]\Big) \\
x_{2,k+1} &= x_{2,k}+\tfrac{T_s}{A_2}\Big(p_{2,k}-S\cdot\Big[(c_2^m(v_{2,k}^m)+c_2^s(v_{2,k}^s))\cdot\sqrt{2gx_{2,k}} \\
&\quad + c_1^c(v_{1,k}^c)\cdot\sqrt{2g\cdot|x_{1,k}-x_{2,k}|}\cdot\operatorname{sign}(x_{1,k}-x_{2,k})\Big]\Big) \\
x_{3,k+1} &= x_{3,k}+\tfrac{T_sS}{A_3}\Big(c_1^s(v_{1,k}^s)\cdot\sqrt{2gx_{1,k}}-c_3^m(v_{3,k}^m)\cdot\sqrt{2gx_{3,k}} \\
&\quad + c_2^c(v_{2,k}^c)\cdot\sqrt{2g\cdot|x_{3,k}-x_{4,k}|}\cdot\operatorname{sign}(x_{3,k}-x_{4,k})\Big) \\
x_{4,k+1} &= x_{4,k}+\tfrac{T_sS}{A_4}\Big(c_2^s(v_{2,k}^s)\cdot\sqrt{2gx_{2,k}}-c_4^m(v_{4,k}^m)\cdot\sqrt{2gx_{4,k}} \\
&\quad + c_2^c(v_{2,k}^c)\cdot\sqrt{2g\cdot|x_{3,k}-x_{4,k}|}\cdot\operatorname{sign}(x_{3,k}-x_{4,k})\Big) \\
x_{5,k+1} &= x_{5,k}+\tfrac{T_s}{A_5}\Big(p_{5,k}-S\cdot c_5^m(v_{5,k}^m)\cdot\sqrt{2gx_{5,k}}\Big).
\end{aligned}$$

where A_i, $i\in\{1,\dots,5\}$ represents cross-section area of the tanks, while $S=3.167\cdot 10^{-5}m^2$ is the cross-section area of the pipes and $g=9.81\ \frac{m}{s^2}$ is the gravitational acceleration. The flow rates of the pumps supplying *Tanks 1, 2* and *5* are denoted by p_j, $j\in\{1,2,5\}$, with $p_i=\{0,1.5\cdot 10^{-4}\ \frac{m^3}{s}\}$, $i\in\{1,2\}$ and $p_5=\{0,0.85\cdot 10^{-4}\ \frac{m^3}{s}\}$, while the direct valves supplying water to *Tanks 3* and *4* are controlled by the input signals $v_i^s=\{0,1\}$, $i\in\{1,2\}$, representing their open (1) and closed (0) state. To regulate the output flow rates of each tank emulating the water demand of the consumers, the output ON/OFF valves gathered in manifolds are controlled by the digital signals $v_i^m\in\mathbb{N}$, whose values are assumed to be equal to the number of opened valves in each manifold. Specifically, $v_i^m=\{0,\dots,4\}, i\in\{1,2\}, v_j^m=\{0,\dots,3\}, j\in\{3,4\}$, and $v_5^m=\{0,\dots,2\}$. Tanks at the same height are connected by crossed-connection valves, controlled by signals $v_i^c=\{0,1\}, i\in\{1,2\}$. Hence, the nominal input vector is defined as $u=[p,\ v^s,\ v^m,\ v^c]$, with $u(k)\in\mathbb{Q}^m$, $m=12$. A discharge coefficient vector related to the output flow is associated to each valve and manifold. Their values were obtained experimentally, depending on the number of valves opened, i.e. $c_i^j(v_{i,k}^j)$. These values have been indicated as c_i^m for the valves in the manifolds, with values $c_i^m=\{0,0.68,0.34,0.23,0.17\}$, $i\in\{1,2\}$, $c_j^m=\{0,0.4,0.2,0.13\}$, $j\in\{3,4\}$, $c_5^m=\{0,0.7,0.35\}$, $c_i^s=0.62$, $i\in\{1,2\}$ for the supply valves, and $c_i^c=0.37$, $i\in\{1,2\}$ for the cross-connection valves.

For each state variable, the FD estimator dynamic is computed as:

$$\hat{x}_{k+1}=\lambda(\hat{x}_k-x_k)+f(x_k,u_k)$$

since $y_k=h(x_k,u_k)=x_k$. The observer gain was set to $\lambda=0.9$ according to empirical testing.

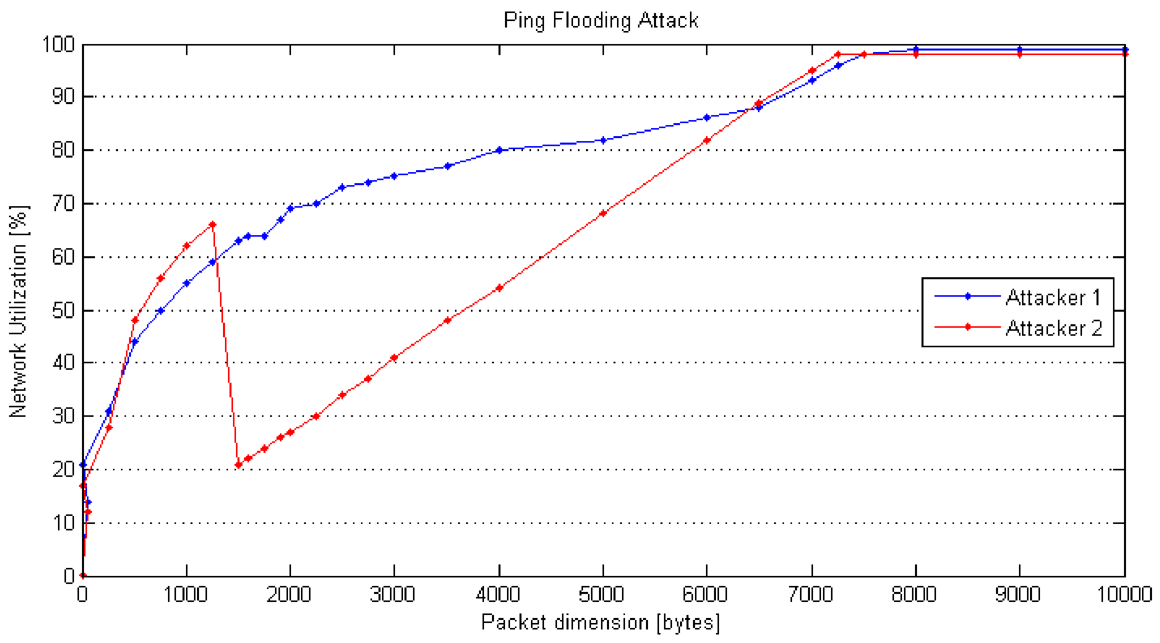


Fig. 5.1 Average network usage during ping flooding vs. SCADA with different packet sizes.

In this way, the FDS calculates an error signal for each tank, as the difference between the measured water level and its estimate at each time step. The error is compared to a fixed threshold experimentally determined. An alarm is generated when this threshold is exceeded, highlighting the forthcoming of potential physical faults. Hence, a physical fault is detected when a remarkable discrepancy between a measured water level and the related estimate occurs. Such an approach has been proven to be suitable for the early detection of different types of single and multiple physical faults.

5.1.2 Availability Attacks Analysis

Ping Flooding. This attack can be performed by means of the *ping* tool, setting the desired size of the packet to be sent repeatedly to the target machine, without latency between consecutive packets, together with other packet properties. In this way, it is possible to downgrade or disrupt the communication between the devices. To successfully perform a *packet flooding* attack in our scenario using only one attacker station, the switch ports where the target machines are plugged have been set to 10 Mbps. At first, the goal is to determine the percentage of network usage related to the size of the packets deployed for the *ping flooding* against the SCADA, considering the two different attacker machines described in Table 5.1. The results in Figure 5.1, expressed in terms of the average percentage of network utilization, show that a total disruption of the communication can be obtained by

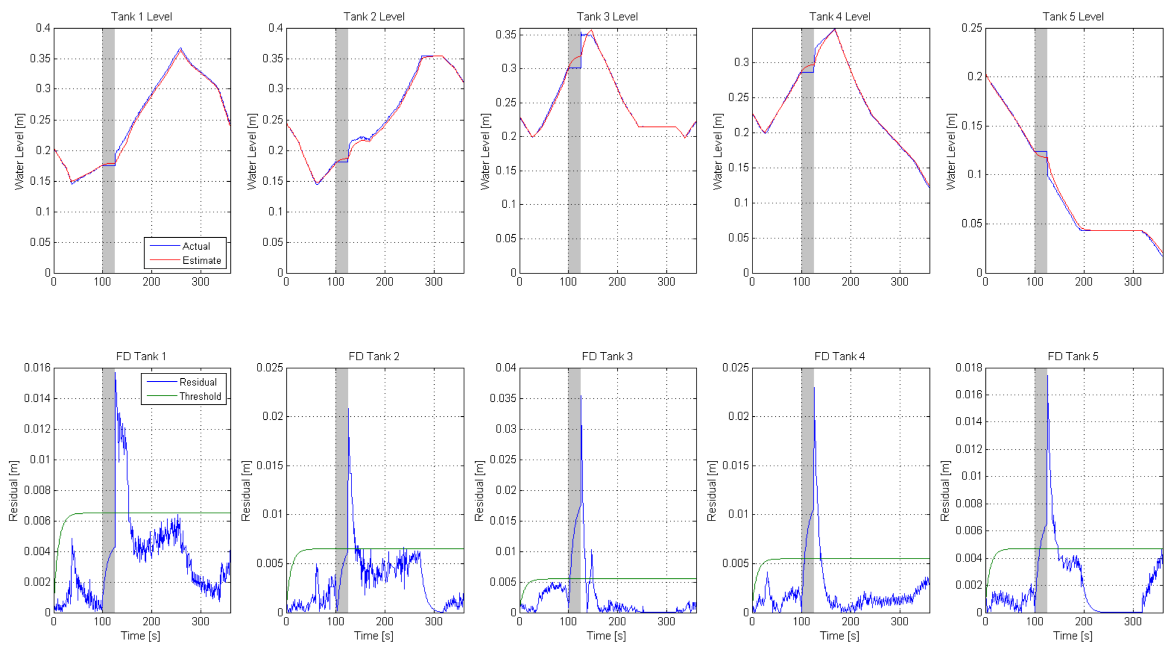


Fig. 5.2 Healthy run – Ping flooding DoS attack vs. PLC at $k_s = 100$ s for 15 s.

setting the packets dimension to at least 8 kB. For lower packet sizes a communication delay could be observed. The two different behaviors for sizes around 1 kB is due to the better performance of Attacker 2. The destination device is unable to respond to each ping, hence the number of lost packets increased and the communication is interrupted. Conversely, Attacker 1 is only able to downgrade the communication, as all the packets sent receive a ping response. Figure 5.2 shows the effects of a 15 s *ping flooding* against the PLC, that takes place at time $k_s = 100$ s during the *Healthy run*. This type of attack almost instantly triggers most of the FDS alerts since the model is not updated, due to the communication interruption. When the communication is restored, the updated sensor values are sensibly different from the expected data, thereby the error signal increases and the threshold is exceeded, highlighting the presence of an anomaly. Several tests have shown that only a subset of FDS alerts is triggered by a *ping flooding* attack, and this number largely depends on the current operating condition of the plant. Thereby, the proper identification of the cyber-attack with respect to single or multiple physical faults taking place in the system would not be guaranteed. Thus, the goal is to determine when such a cyber-attack would trigger a false alarm. This peculiar situation has been assessed carrying out a wide number of *ping flooding* attacks by varying duration between 1 s and 20 s, and at different time instants of the daily scenario ($k_i = \{50, 100, 150, 200, 250, 300\}$ s). The results of this analysis, shown in Figure 5.3, pinpoint the average number of FDS alerts triggered in each case. There is a

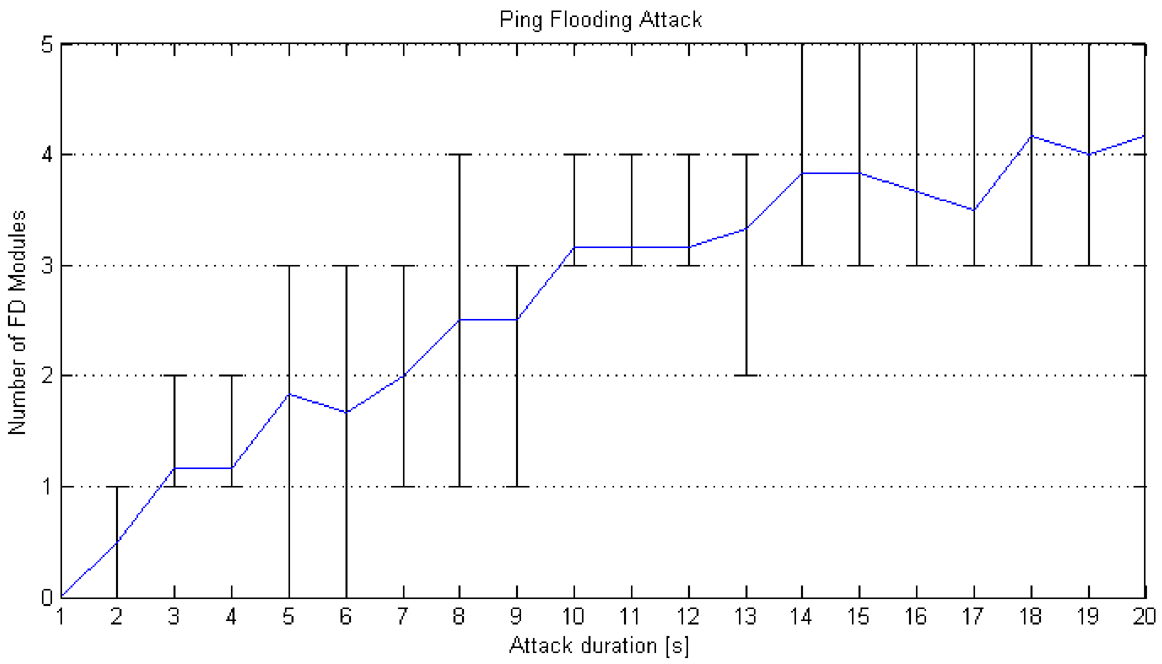


Fig. 5.3 Average number of FDS alerts triggered by ping flooding DoS attacks vs. PLC taking place at different time instants of the *Healthy run*, with varying duration. The vertical lines consider the max and min number of FDS alerts triggered for each duration.

remarkable variation in the FDS reaction, depending not only on the attack duration but also on the system behavior at that specific time.

As one may expect, the number of triggered FDS alerts increases with the attack duration and almost all of them reveal the anomaly if the attack lasts at least 14 s. Even if this result might sound obvious, it is interesting to highlight how this could be exploited by the attacker. When a large number of FDS alerts are triggered simultaneously, the operator may understand that the malfunction may not be related to the physical domain and the conclusion that the system is undergoing a communication anomaly would be straightforward. On the other hand, by exploiting short-lasting attacks, the attacker would slyly mislead the operator, which probably will not be able to distinguish between the ongoing cyber-attack and an incipient fault taking place in a single tank.

Modbus Flooding. Several tests have been performed sending consecutive Modbus packets to the SCADA system with different time delays among them. To this aim, the *nping* tool has been employed as network packet generator. A specific packet of the TCP stream is selected and deployed as model for the flood replication. Specifically, it is a response message from the PLC to the SCADA containing sensor measurements. This Modbus packet is repeatedly sent to the SCADA with time delays ranging from 0.01 ms to 1000 ms, using the IP address of the

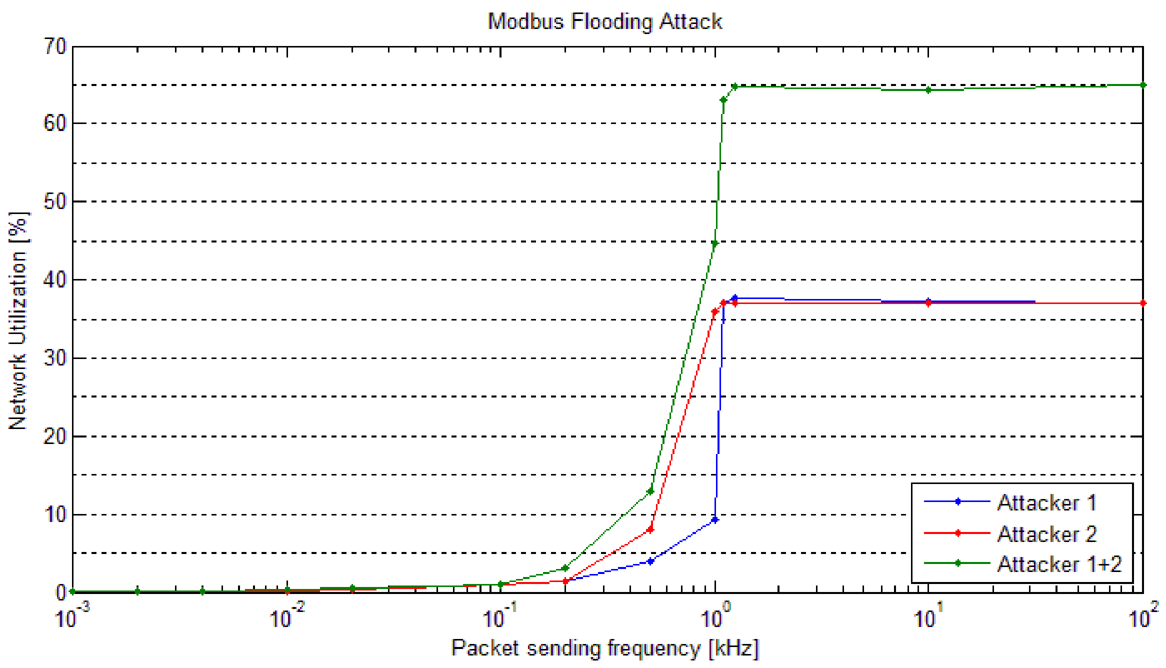


Fig. 5.4 Network usage during Modbus flooding vs. SCADA with different time delays expressed as packet rates.

PLC as fake source. As depicted in Figure 5.4, where the percentage of network utilization is plotted versus different packet rates, such attack causes a generalized communication slack for time delays lower than 1 ms and increase the network usage to $\sim 37\%$ in the single attacker worst case. This limit reaches $\sim 67\%$ using two attacker machines at the same time. After about 25 s from the beginning of the attack, for time delays beyond 0.8 ms, anomalies in the communication of sensor measurements can be observed. Such a behavior is induced because the SCADA is elaborating - at a high frequency - both actual and fake Modbus replies.

In all the aforementioned cases, the FDS is able to perceive delays or faults in the system, even if it is not able to identify the source. On the other hand, the S-IDS is not producing alarms because the Modbus/TCP traffic is allowed by the rules implemented for the test.

5.1.3 Integrity Attacks Analysis

Data Modification. During the data modification attacks, different performances are observed, depending on the characteristics of the specific machine used by the attacker. For instance, Attacker 1 is not always able to successfully perform the sensor measurements modification attacks, as a communication interruption took place, probably due to insufficient computational capabilities or poorly performing Network Interface Controller (NIC).

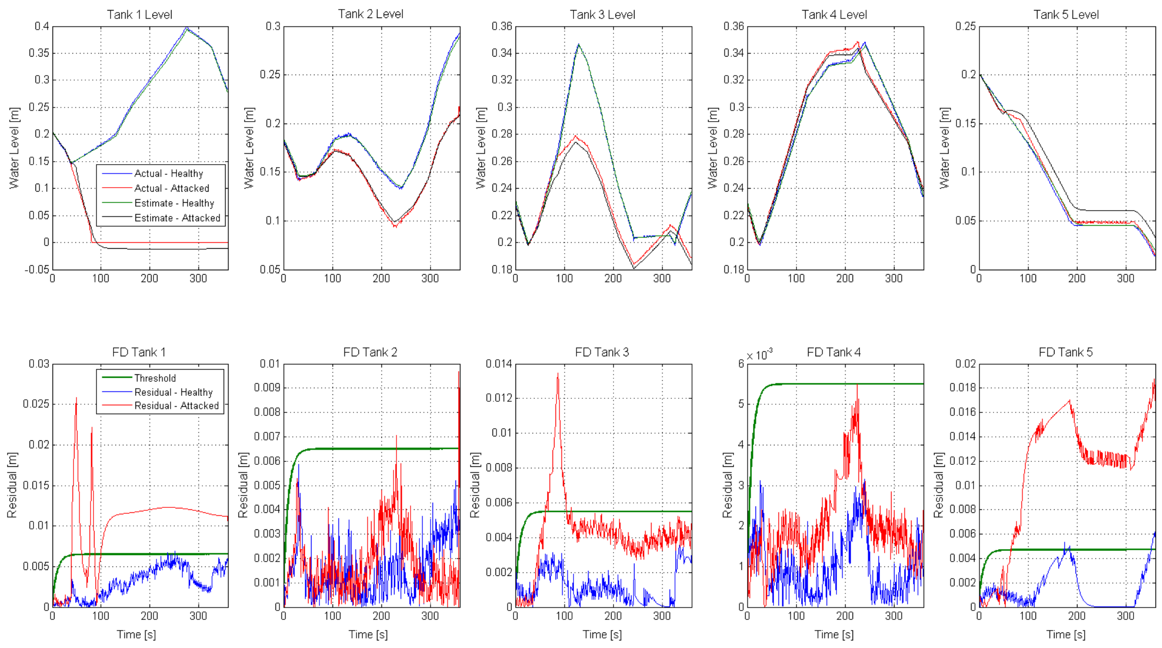


Fig. 5.5 Healthy run – Data modification attack vs. SCADA at $k_s = 35$ s for 15 s – Commands to Pump 2 performed by Pump 4. The healthy (blue) and attacked (red) conditions showed.

Thereby, the attacker should be particularly careful about the choice of the devices to exploit, in order to guarantee the achievement of his goals. Three different data modification attack cases have been studied taking place during the *Healthy run*.

A. ***Actuators state modification:*** the data field of the Modbus packets has been modified so that the commands addressed from the SCADA to Pump 2 are instead performed by Pump 4 at time $k_s = 35$ s, for 15 s. As these attacks are carried out through a MITM attack, no anomalous behavior could be observed in the operator interface. Conversely, as the FDS calculates the state estimates considering the commands sent from the SCADA, a deviation from the actual measurements of the attacked component is observed, as depicted in Figure 5.5. Specifically, the model computes the estimation considering that Pump 2 is ON, hence Tank 1 should be supplied, while actually the water level decreases. Thereby, the deviation between actual and estimated values increases, and the fault threshold is exceeded in Tank 1. On the other hand, as Pump 4 is intentionally turned ON by the attack (not expected by the model), an anomaly in Tank 5 is observed. In addition, the emptying of Tank 1 provokes a lack of water supply in Tank 3, that triggers the related detection alerts.

B. ***Sensor measurements modification:*** as illustrated in Figure 5.6, the water level of Tank 2 has been masked on the HMI between times $k_s = 100$ s and $k_e = 130$ s. Since

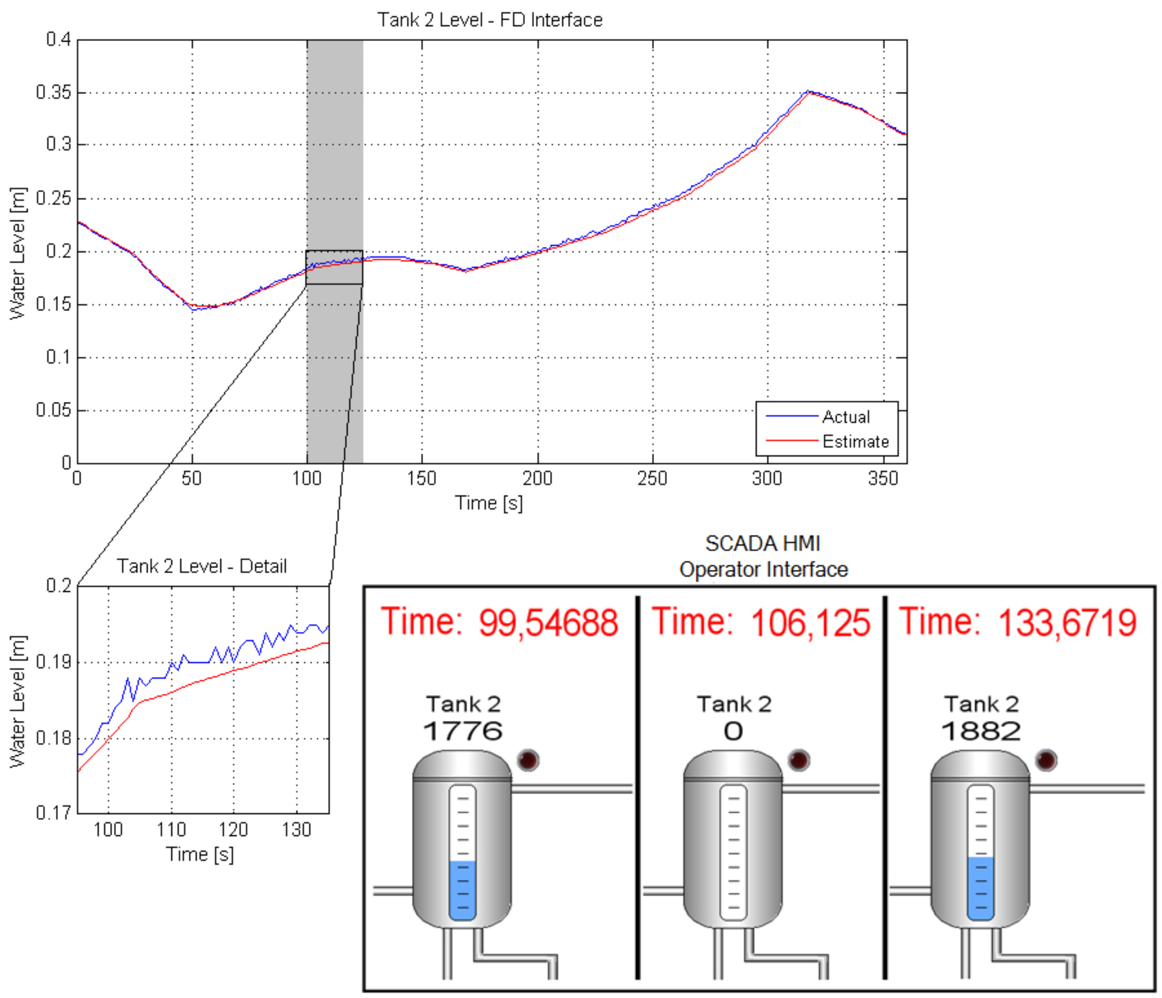


Fig. 5.6 Healthy run – Data modification attack vs. SCADA at $k_s = 100$ s for 30 s – SCADA acquired measurements replaced by 0 values, while the system was operating properly.

the actual sensor measurements are properly received, the FDS is not able to highlight any anomaly, and no alarms are triggered in this case. However, as the empty tank is considered a critical situation, an alarm is prompted on the HMI revealing the anomalous situation to the operator, who may be induced to perform unnecessary countermeasures.

C. ***Fake exception response:*** some response messages from the PLC have been intercepted and substituted by an exception response. To this aim, the data field is replaced by the chosen exception code (e.g., *Illegal function*, *Illegal data address*, *Illegal data value*, *Slave device busy*). On the plant side, no effects are observed due to the attack, as the operation requested by the SCADA is properly processed and carried out by the PLC.

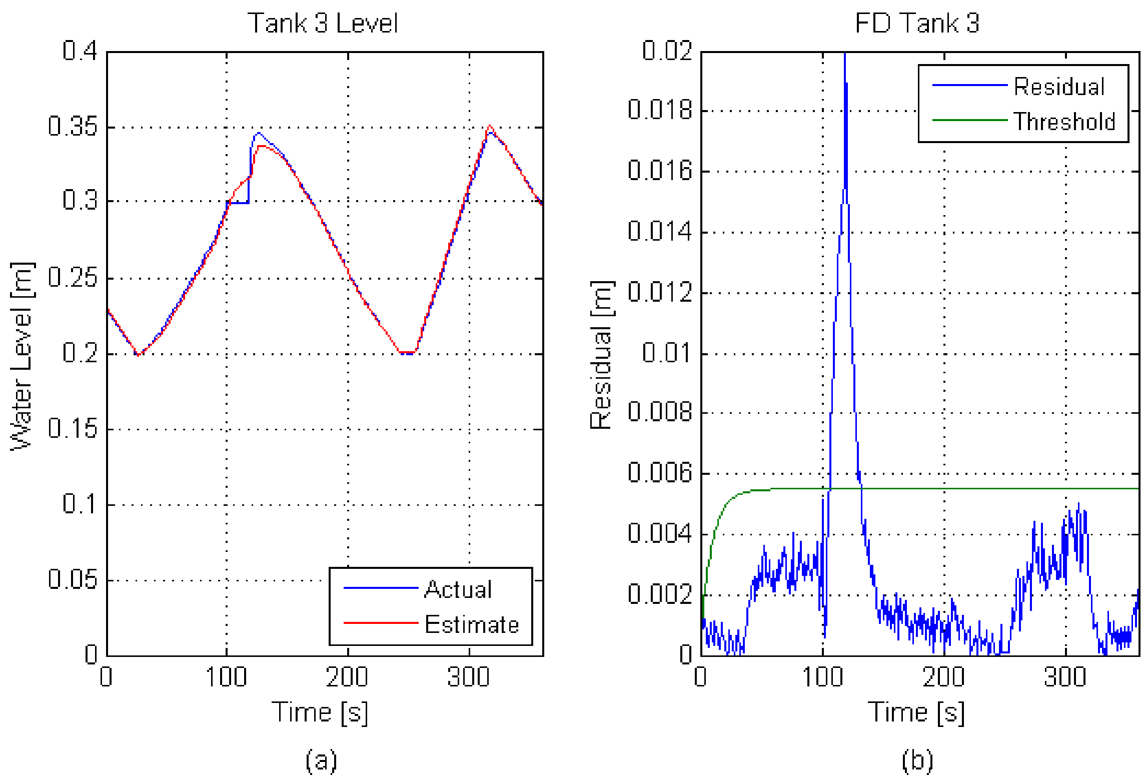


Fig. 5.7 Healthy run – Replay attack. The difference between the expected values and the tampered measurements (a) provokes peak response of the monitoring system (b).

Replay Attack. The water level measurements of Tank 3 received by the SCADA are replaced with a previously recorded constant value for 20 s, beginning at time $k_s = 100$ s, by properly changing the data field of the corresponding Modbus packets. Thereby, the data modification method is exploited to perform a replay attack. During the attack, the tampered values are displayed on both the HMI and FDS interfaces. When the attack is completed, the actual measurements returned to be visible. As shown in Figure 5.7, the attack is immediately revealed because of the remarkable difference between the fake measurements and the estimations provided by the model.

Unless integrity attacks lead to unexpected behavior in the water system, the operator cannot properly react without the S-IDS support. In this case, the S-IDS has been configured to reveal ARP spoofing attacks by using a table in which the trusted nodes are identified by static IP and MAC address entries. When a malicious actor tries to activate a MITM attack, the S-IDS alerts the security operator. In this way, the S-IDS flags an alert making the operator aware of the cyber-threat when complex cyber-attacks are able to bypass the classical FDS routines. By combining various cyber-attacks, it is possible to perform a more complex one. On the one hand, the malicious agent would send fake healthy information to

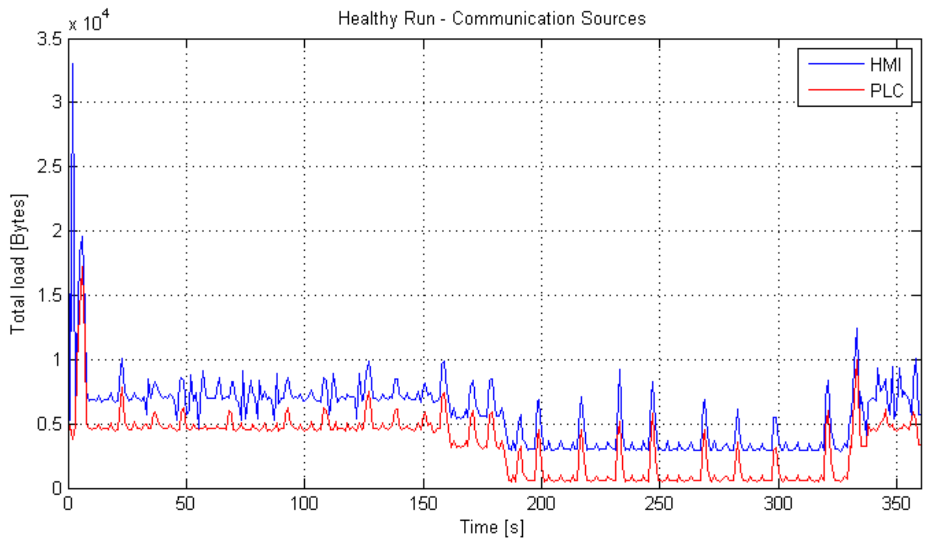


Fig. 5.8 Normal network load for SCADA and PLC during daily scenario.

the system, while attacking the physical components to move a part or the whole plant to an unstable state. In this way, the operator would not be able to easily recognize the hazard and to perform the proper recovery actions, as the situation would be reported as normal. On the other hand, it is possible for the attacker to modify the data to fake the behavior of the system and simulate the effects of an attack. In this case, the HMI would report the anomalies, while the plant is actually working normally. As a consequence, the operator would be prone to perform recovery operations from such an unstable state or, in the worst case, to interrupt the operation of the system. These undesired actions may lead to severe consequences.

5.1.4 Network Load Impact Analysis

Several experiments have been performed to highlight the consequences on the communications network of the FACIES testbed caused by cyber-attacks taking place during the *Healthy run* [119]. An analysis of the typical network flow has been carried out, evaluating the total load of packets sent from the SCADA and from the PLC, as depicted in Figure 5.8.

Considering such trend as reference, the *Modbus flooding* attack has been performed against the PLC. As described in Sec. 5.1.2, also for these experiments the *nping* tool has been exploited to create and send 100.000 fake *Read Input Registers* queries (i.e., requesting the sensor measurements) from the SCADA to the PLC at time $t = 100$ s, with a time delay of 0.01 ms between consecutive packets. In this case, the SCADA constitutes a fake source of packets, as they are actually sent by the attacker, but contain the SCADA IP as sender. As can be observed in Figure 5.9 through an external network analyzer, this provokes an

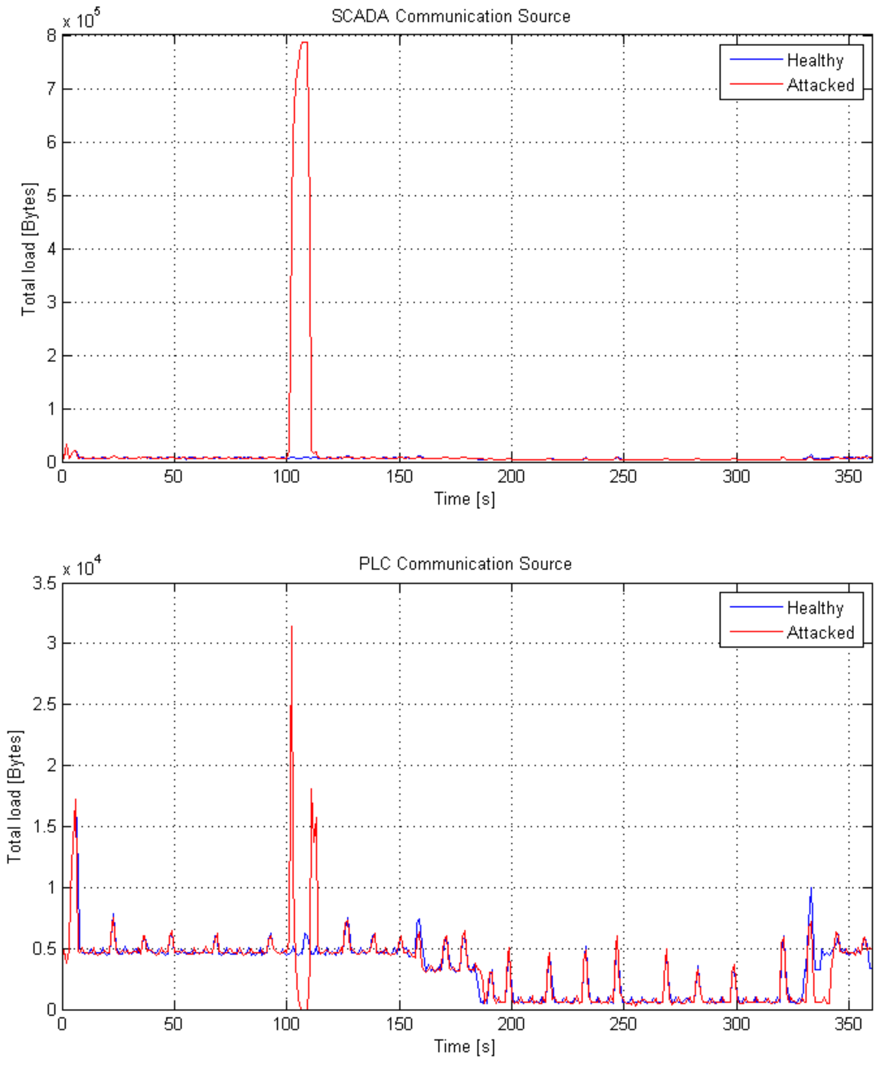


Fig. 5.9 SCADA and PLC total load (Bytes) during daily run vs. Modbus flooding at $t = 100$ s for 10 s.

instantaneous peak in the communication load both on the fake source side (SCADA), and on the PLC, which attempts to reply to every request until saturation.

After the attack, which lasts $10s$, the communication is restored, but a slight delay on the PLC replies is verified. In addition, from statistical analysis, it has been observed that the averages of the total load during the attack ($\mu_{PLC_{ATT}} = 5647$ bytes and $\mu_{SCADA_{ATT}} = 517980$ bytes) are higher than 3 times the standard deviation of the total load in normal operation ($\sigma_{1PLC} = 770$ bytes and $\sigma_{1SCADA} = 981$ bytes), i.e., such event has a probability lower than 0.3% during the *Healthy run*. Thereby, similar mean values could be associated to an anomalous behavior on the network.

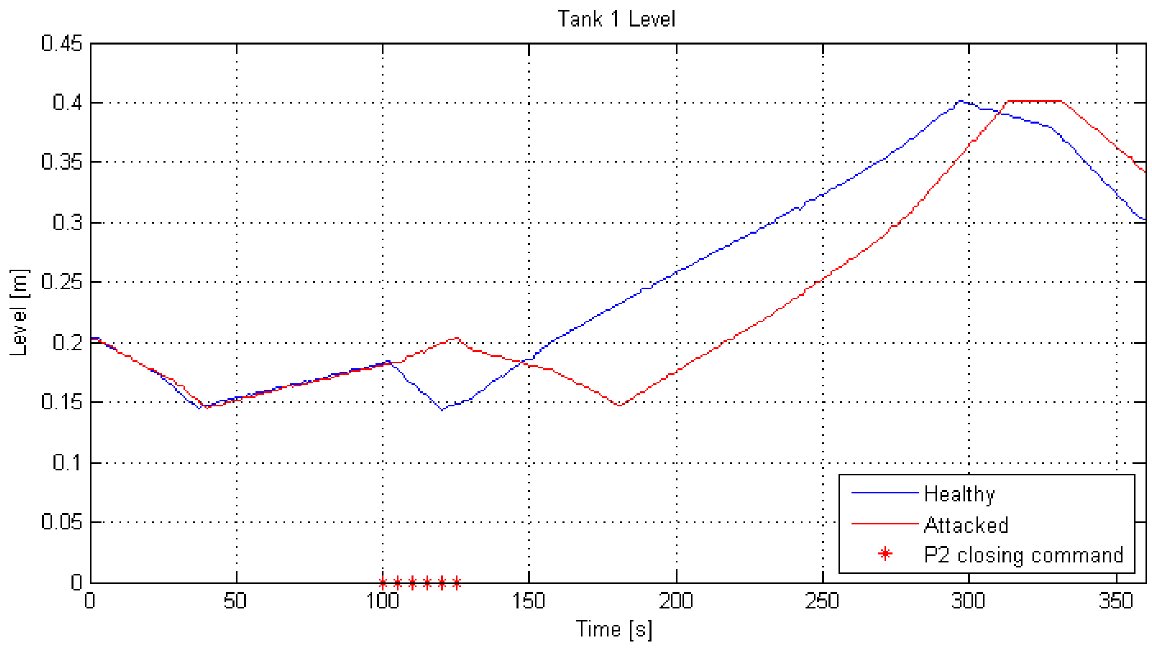
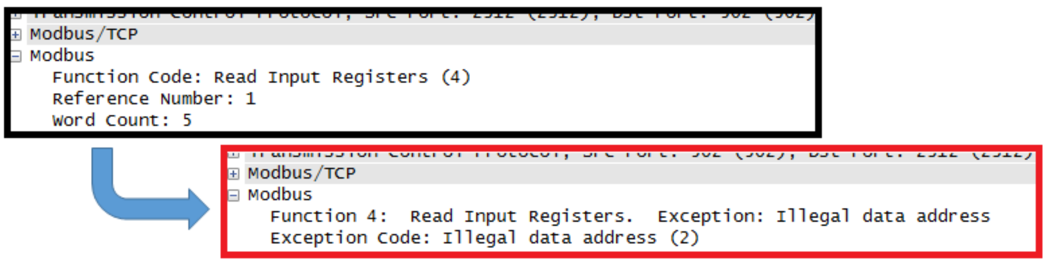


Fig. 5.10 Healthy run vs. Data modification attack vs. SCADA at $t = 100s$ - Commands to Pump 2 performed by Valve 1.

Algorithm 1 Data modification filter

```
1: while loop do
2:     C1 → IP source = target machine IP
3:     C2 → Modbus.DATA is "\x0f\x00\x15"
4:     if C1 ∧ C2 then
5:         **replace Modbus.DATA**
```

Another *Data Modification* attack has been performed in order to manipulate the data field of selected Modbus/TCP packets traveling on the network. In this case the operator was unable to deactivate Pump 2 through the HMI, as the packets were intercepted and its content was modified, so to close Valve 1. To this end, a filter has been implemented as sketched in Algorithm 1. The conditions are related to the IP source of the packets, the SCADA system in this case, and the function code and actuator to which the modification is referred. The instruction performs a substitution of the code of the target device (Pump 2) with the one corresponding to the desired one (Valve 1). Thereby, every command sent from the SCADA to Pump 2 will be performed by Valve 1. Tampering the communication in this way may lead to dangerous consequences, as the operator is unable to stop the tank filling by sending the related command from the HMI, as shown by the evolution of the water level in Tank 1 depicted in Figure 5.10, and a water overflow may occur. This has been prevented by implementing low-level security controls, which stops the operation of the corresponding



```
⊞ Modbus/TCP
⊟ Modbus
    Function Code: Read Input Registers (4)
    Reference Number: 1
    Word Count: 5
```

```
⊞ Modbus/TCP
⊟ Modbus
    Function 4:  Read Input Registers.  Exception: Illegal data address
    Exception Code: Illegal data address (2)
```

Fig. 5.11 Data modification attack - Fake exception response.

actuator when the water level measured by the sensor reaches a specific security threshold. These are performed locally on the PLC and do not depend on the communications network.

Finally, the *Data Modification* attack has been implemented to send fake exception responses from the PLC to the SCADA, following requests from SCADA. The algorithm is similar to the previous, considering the corresponding PLC IP address as source (condition *C1*) and the Read input registers function code as condition *C2*. The packet payload is modified so as to replace the request function code with the corresponding exception function code, and the data field with the desired exception code *Illegal data address* in this case). This attack has been proven to be successful by analyzing the network packets on Wireshark. As shown in Figure 5.11, the response from the PLC to the Read input request is the desired exception reply. From the plant side, no consequences are observed and the operation is normal, as the requested action is normally carried out by the PLCs and only its response to the SCADA is tampered. This is more evident when the fake response attack is performed against a Write Single/Multiple coils request, i.e., when the SCADA sends a command to the actuators. The dangerousness of such attack arises when the operator performs unnecessary countermeasures on the system to restore the plant operation, lead by the fact that it is *not* responding normally, which may derive in an unstable state or in the undesired shutdown.

5.2 Fault Detection System for HYDRA Testbed

In this Section, experimental results concerning the water testbed operation are presented: in the first part, a proper functioning example of the plant is shown, afterwards, the results of a *Data Modification* attack exploitation against the system are produced. The analysis of the residual calculations and the inconsistencies among the pressure sensor and flow meter values are performed. For the experimental stage, the proposed scenario is composed by two tanks vertically connected. As shown in Figure 5.12 the upper tank is a cold water container, whereas the lower one represents the cold water user. By means of the e_1 valve opening, tank T_2 is supplied with cold water (e.g., in order to perform a plant manufacturing) and then

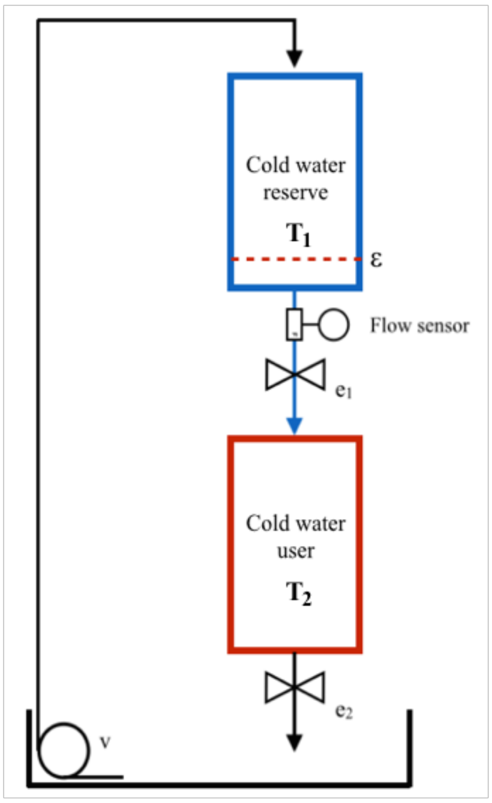


Fig. 5.12 The proposed scenario.

ejected after the use through the e_2 valve. Such system is represented by the model shown in Figure 5.12, where h_1 is the level inside the cold water reserve and h_2 the user tank T_2 level. A control system, implemented in MATLAB/Simulink, monitors the water level in the upper tank, and when necessary, it activates the pump that supplies the cold water reserve (Tank T_1) drawing water from the reservoir. The control algorithm on the HMI acquires level data from the tanks through Modbus/TCP packets coming from the Intel Galileo with a frequency of 10 Hz, while the flow meter informations are acquired via Serial port with 2 Hz frequency.

A first system test is shown: inside the cold water reserve 15 cm of water are measured; after 50 s, the e_1 valve is opened to supply tank T_2 that requires cold water to execute an industrial operation (Figure 5.13). At the end of the process (after about 210 s), valve e_1 is closed and, by the opening of the e_2 valve, all the water exploited during the user working process is expelled. Figure 5.14 shows the trend of the cold water reserve outgoing flow

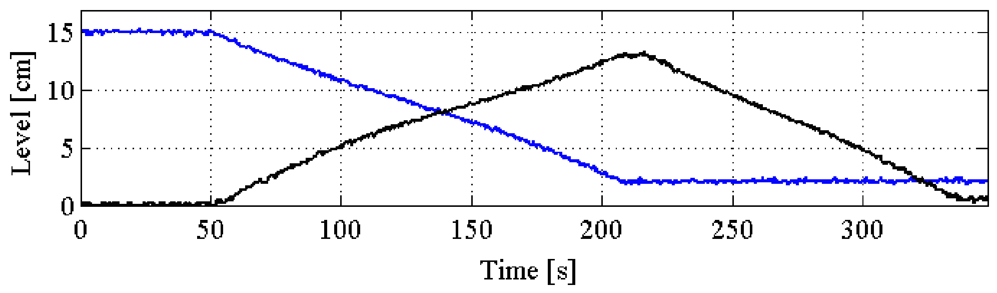


Fig. 5.13 Water levels h_1 (blue line) and h_2 (black line).

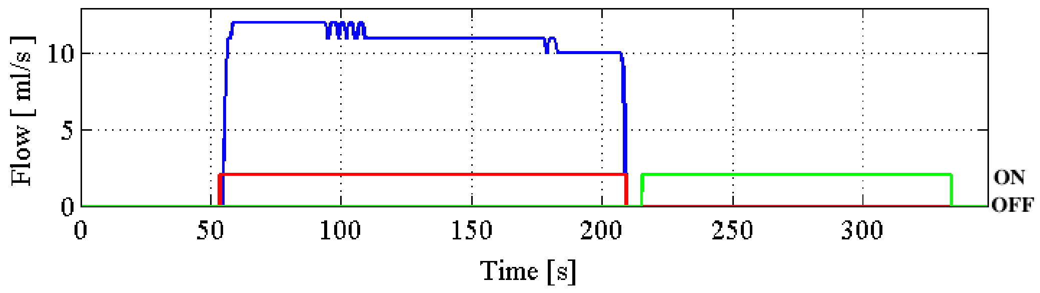


Fig. 5.14 Flow meter values (blue line) and valves state e_1(red line) and e_2 (green line).

considering the water level variation in tank T_1. As expected, the decrease of the water level in T_1 reduces the output flow from the tank.

5.2.1 Cyber-attacks evaluation

In order to carry out the experimental stage of the *Data Modification* attack, some Penetration Testing tools have been used to perform cyber-attacks against the network components. The Kali Linux and MITM attacks have been employed. In order to carry out a malicious action on the system, the attacker must be able to communicate over the network. As shown in Fig. 4.12, the network architecture has been corrupted by the inclusion of an attacker. From a methodology point of view, the ARP spoofing technique has been carried out to perform the MITM attack. Through the use of *Ettercap* the attacker has been able to sniff Modbus/TCP packets containing data of the pressure sensors and to change data field without apparently leaving any trace. Therefore, the hijacked packets have been retransmitted to the original destination with the data field modified. Also in this case, the *Ettercap* tool performs the ARP Spoofing operation and the *Etterfilter* utility compiles source filter files, created ad hoc for the experimental purposes, that can be interpreted by the *Ettercap* engine for the data manipulation.

In the second experiment, a scenario comparable to the previous one is considered, but in this case, the algorithm which controls the water level in the reserve tank (T_1) is performed.

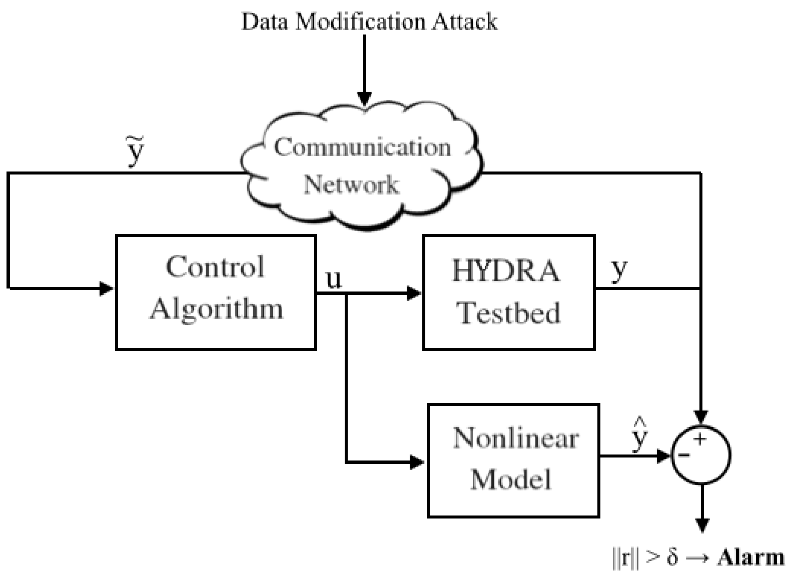


Fig. 5.15 Data Modification Attack scheme.

In order to detect any possible malfunctioning, the continuous nonlinear system model in equation (5.1) has been considered; given the same inputs represented by the state of the valves and the activation of the pumps, it provides an estimate of the real system state.

$$\begin{cases} \dot{h}_1 = -e_1 a_f \frac{1}{A}\sqrt{2gh_1} \quad + \frac{1}{A}kv \\ \dot{h}_2 = e_1 a_f \frac{1}{A}\sqrt{2gh_1} - e_2 a_f \frac{1}{A}\sqrt{2gh_2}. \end{cases} \tag{5.1}$$

It is worth mentioning that the physical model is completely observable considering the level sensors as observations. Comparing the water levels of the tanks supplied by the sensors with those generated by the model, it is possible to compute the residual values through a simple scheme of fault detection schema, as shown in Figure 5.15. As in the previous experiment, at $t = 38$ s the valve e_1 is opened and the water in the reservoir flows in the tank T_2. As a consequence, the level in tank T_1 decreases while the one in T_2 increases. At $t = 192s$, the water level in T_1 reaches the minimum threshold of 3.5 cm and the control algorithm enables the pump to move the water from the reservoir to tank T_1. As depicted in Figure 5.15, the attacker, using a MITM attack, corrupts at $t = 190s$, the data field in the Modbus/TCP packets that contains the water levels in the two tanks. Even though the pump activation, the water level in T_1 remains unchanged due to the MITM attack (Figure 5.17(a)). The signal about the pump activation is also given to the nonlinear model of the testbed (Figure 5.17(b)). The model is not affected by the attack and the computed water level

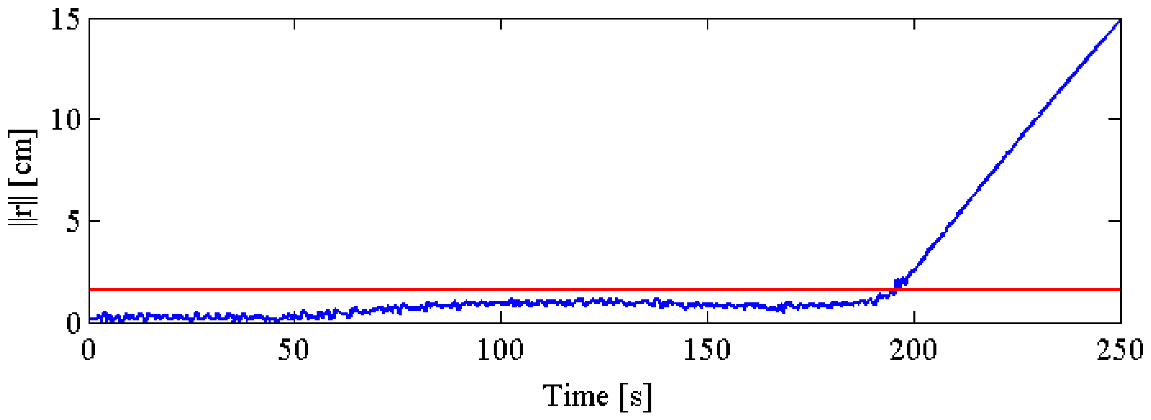


Fig. 5.16 Norm of the residual vector r in attack scenario.

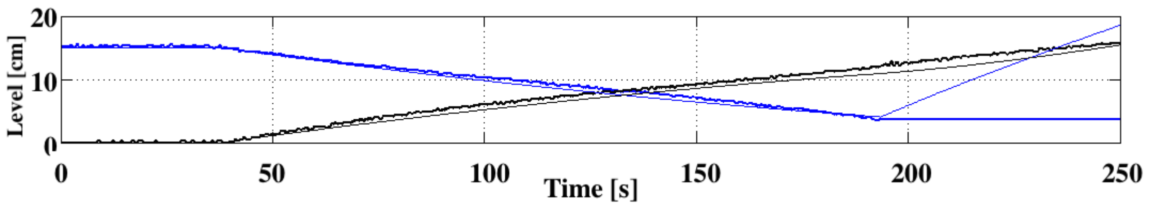
in T_1 quickly increases due to the pump activation (Figure 5.17(a)). At the same time the flow measurement increases due to the increasing water level h_1 (Figure 5.17(b)). The differences between the estimated value $\hat{y}$ and the measured value y represented by r_1 and r_2 (Figure 5.17(c)) are monitored and an alarm is triggered if $\|r\|$ is greater than a given fixed threshold δ. The δ parameter depends on the accuracy of the model. Low value of δ can result in a high false positive alarm rate, while a high value of δ can result in high false negative alarm rate. Analyzing the system under normal conditions the value of δ has been set at 1.6 *cm*. In Figure 5.16, it is depicted the value of $\|r\|$ during the test, it is defined as follows:

$$\|r\| = \sqrt{\sum_{i=1}^{N} r_i^2} \tag{5.2}$$

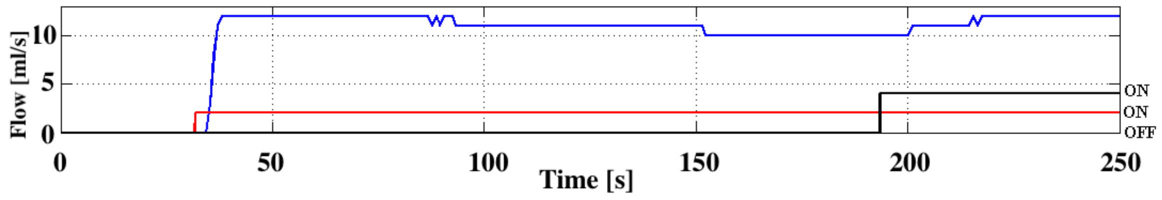
where: $r = [r_1, r_2]$ and $r_i = y_i - \hat{y}_i$.

Before the attack, the mean value of $\|r\|$ was 0.6251 *cm* and the variance σ^2 was equal to 0.0092 *cm*.

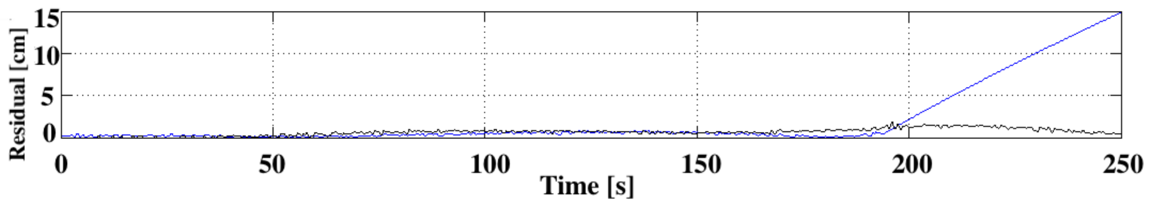
Considering the residual incrementation, the water flow incoherence with the water level measurement in T_1, and the S-IDS alarm related to an ARP spoofing attack from the water level sensor source, it has been possible to distinguish the cyber-attack instead of a physical fault. Moreover, it is possible to identify the exact measure corrupted by the cyber-attack merging the information from the S-IDS, the redundant connection to the sensors (Fig. 4.12) and the computed residual values. Single information regarding the water flow, the measurements and the S-IDS alerts, if not fused is useless; only by the data fusion it is possible to identify the typology of the attack and the threat result allowing the human operator to perform a countermeasure.



(a) Water levels h_1 (blue line) and h_2 (black line) and associated model estimation (thin line).



(b) Flow meter values (blue line), valves state e_1 (red line), and pump state v (black line).



(c) Residuals values r_1 (blue line) and r_2 (red line).

Fig. 5.17 Sensors and actuators values during Data Modification attack test.

As proof of concept, we propose another experimental trial: a basic water distribution system emulation based on the HYDRA testbed. The physical system is composed of three tanks and a reservoir (see Fig. 5.18). Tank 1 and 2 are connected by a serial pipeline: the fluid cascades due to gravity and the flow is regulated by the proportional valve $v_{1,2}$. Tank 2 and 3 are connected in a parallel configuration: the fluid moves due to Stevin's Law (communicant vessels) and the flow is regulated by the proportional valve $v_{2,3}$. The whole system is fed by a reservoir: the centrifugal pump k_1 provides water to Tank 1, while the centrifugal pump k_2 links Tank 3 and 1. Finally, three manual valves (v_1, v_2, and v_3) can be used to simulate leakages (e.g. physical faults). The system behavior is emulated by means of the following continuous time equations:

$$A\dot{h_1} = P_1 + P_2 - Q_{1,2} - Q_{1,0} \tag{5.3}$$

$$A\dot{h_2} = Q_{1,2} - Q_{2,0} - Q_{2,3} - Q_{2,3,h} + Q_{3,2,h} \tag{5.4}$$

$$A\dot{h_3} = -Q_{3,0} + Q_{2,3} + Q_{2,3,h} - Q_{3,2,h} - P_2 \tag{5.5}$$

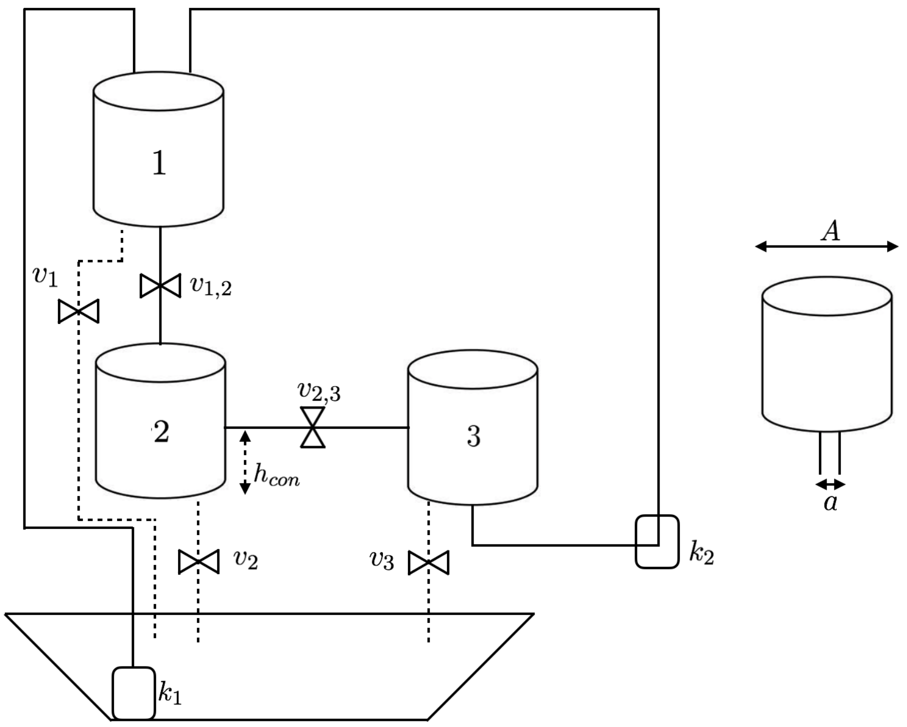


Fig. 5.18 Simulated physical system.

where:

$$Q_{1,2} = av_{1,2}\sqrt{2gh_1} \tag{5.6}$$

$$Q_{1,0} = av_1\sqrt{2gh_1} \tag{5.7}$$

$$Q_{2,0} = av_2\sqrt{2gh_2} \tag{5.8}$$

$$Q_{3,0} = av_3\sqrt{2gh_3} \tag{5.9}$$

$$\begin{aligned} Q_{2,3} = av_{2,3}U(h_2 - h_{con})U(h_3 - h_{con}) \\ Sign(h_2 - h_3)\sqrt{2g|h_2 - h_3|} \end{aligned} \tag{5.10}$$

$$Q_{2,3,h} = av_{2,3}U(h_2 - h_{con})U(h_{con} - h_3)\sqrt{2g(h_2 - h_{con})} \tag{5.11}$$

$$Q_{3,2,h} = av_{2,3}U(h_3 - h_{con})U(h_{con} - h_2)\sqrt{2g(h_3 - h_{con})} \tag{5.12}$$

$$P_2 = k_2 a\sqrt{2gh_3} \tag{5.13}$$

$$P_1 = k_1 \tag{5.14}$$

where $Q_{i,j}$ represents the flow through the tanks i and j, $U(\cdot)$ is the step signal, and g the gravitational acceleration.

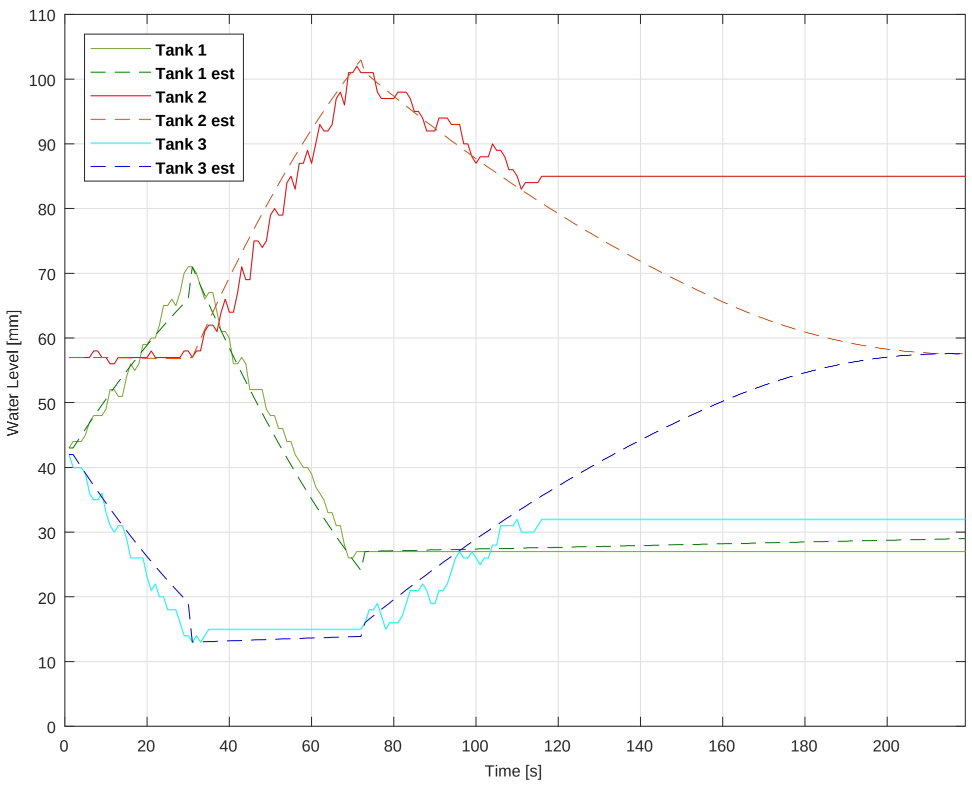


Fig. 5.19 Level of the tanks during simulation under Replay attack.

The physical system described as case study is a nonlinear system, however, an EKF and the corresponding residual evaluator, based on peak norm, the maximum value assumed by the residual norm under nominal conditions, have been implemented.

Denoting with r_k the residual signal and with α the threshold, the system is under fault or cyber-attack conditions if $r_k > \alpha$.

Experimental tests have been implemented to prove that cyber-attack produces anomalies that can be detected by the FDS. A static replay attack is performed during control routines of a cyclic simulation scenario and results are sketched in Fig. 5.19.

The residual analysis is shown in Fig. 5.20. As it can be seen, the detection system clearly identifies the anomalies. The residuals for tanks 2 and 3 in the simulated run are reported on the figure in solid blue line, while the red dotted lines are the peak norm: it is easy to understand that the thresholds are violated as soon as the attack starts.

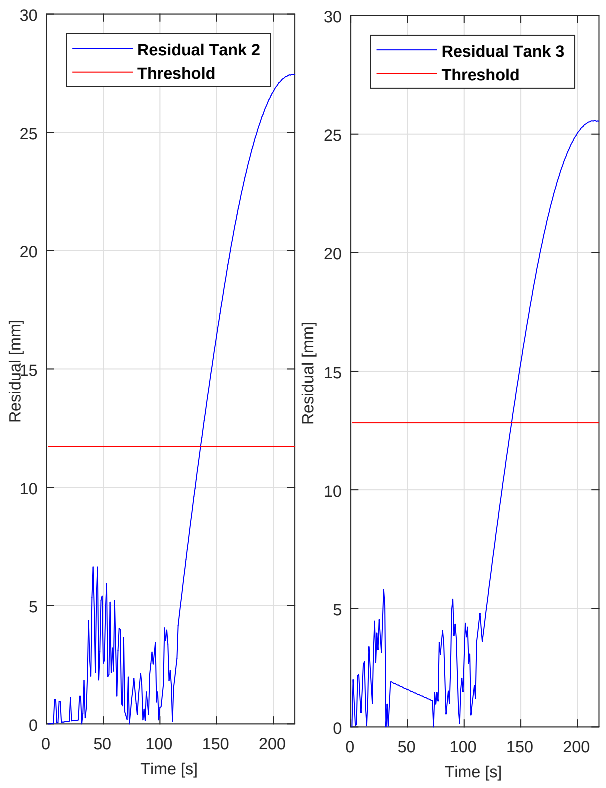


Fig. 5.20 Residuals of tank 2 and 3 during the replay attack.

5.3 Experimental Results on the Network Anomaly Detection Engine

In this section, the experimental results concerning the implementation of the NADE module in simulated and emulated scenarios are discussed. Firstly, the NADE is evaluated in Control zone virtual network. Then, using FACIES testbed, the NADE is evaluated when faults, cyber-attacks, and faults+cyber-attacks occur.

5.3.1 Network Anomaly Detection Engine in Simulated Scenario

In the preliminary scenario, the NADE module is used to evaluate cyber-physical problems on the simulated network. For this experiment, the analyzing period of the L-NADE module has been fixed to 10 times the operating period ($t = 60\ s$). The L-NADE configuration is the most critical part of the implementation: the parameters to be used for profiling the network need to be carefully chosen. For this experiment, the following parameters have

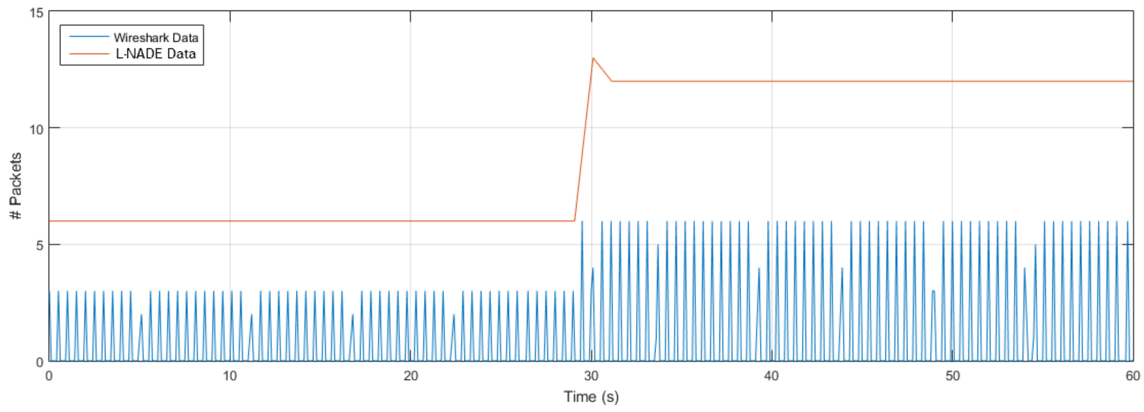


Fig. 5.21 Network normal behaviour.

been selected: *Packet Timestamp, Read Input Register Query, Read Input Register Response, Total Modbus Packets, Total Packets*. Subsequently, it has been decided to create a reference to the normal behavior of the network considering every second of traffic analyzed. In this way, $n = 60$ entries for the network traffic profile file have been generated with information on the parameters described above. Once the profile file has been created, the NADE is activated: it analyses every second of the network traffic and the parameters data taken into account are compared with those generated by the L-NADE phase. The NADE generates an alert when:

$$\eta(i) > \eta^{\star}(i) + \delta(i) \tag{5.15}$$

where: $\eta(i)$ is the i-th value of the parameter considered for anomaly detection derived from the analysis of the actual network traffic, $\eta^{\star}(i)$ represents the i-th value of the relative parameter stored in the profile file, whereas $\delta(i)$ is an uncertainty value chosen to mitigate false detection probability. For this experimental phase, a constant value $\delta = 2$ has been chosen by considering the standard deviation of 10 nominal operating periods.

In order to assess the experimental behavior of the NADE, cyber-attacks on the experimental network have been carried out to verify the effectiveness of the cyber security system. It is assumed that the attacker succeeds in gaining access to the network and he/she is connected as a normal node. The *SYN Flood Attack* has been considered: this cyber-attack represents a DoS method that exploits the TCP three-way handshake mechanism. Flooding TCP segments to a server causes the *SYN-RECEIVED* state to reach the maximum admissible value. In this way, the legitimate clients are not able to connect to the server and this provokes a DoS behaviour [121]. The periodic simulation steps are described in the Tables 5.2 and 5.3. The two Monitors start communications at different times and the *SYN Flood Attack* attempts to avoid the initialization of new connections between client and server.

Simulation Time (s)	Simulation Description
0	PLC Server starts simulating sensor values read operations
5	Monitor 1 starts querying PLC for Read Input Registers data
30	Monitor 2 starts querying PLC for Read Input Registers data
60	Simulation Ends

Table 5.2 Normal behaviour simulation routine.

Simulation Time (s)	Simulation Description
0	PLC Server starts simulating sensor values read operations
5	Monitor 1 starts querying PLC for Read Input Registers data
20	Attacker starts SYN Flood attack against the PLC
30	Monitor 2 starts querying PLC for Read Input Registers data
60	Simulation Ends

Table 5.3 Attack behaviour simulation routine.

In Fig. 5.21 the SCADA network simulated traffic in nominal conditions is depicted. The blue line represents the packet captured over the time and the red line represents the profile dynamics generated during L-NADE phase. The *Total Packets* parameter is considered for this experiment. As shown, at $t = 30\ s$, the *Monitor 2* starts to query the Server and the network detects twice the number of packets/seconds.

Once the network data acquisition and profile generation stages are completed, the NADE is activated and starts to compare the actual network traffic with the profile previously created. In Fig. 5.22, the network traffic of the system under attack is represented. As it can be seen from the figure, at $t = 20$ s the cyber-attack starts and the actual network traffic exceeds the NADE security thresholds. The NADE module, indeed, compares the traffic every second and generates alerts along the whole period of attack. When the attack ends ($t = 40$ s), the traffic analyzed drops below the NADE threshold. These preliminary results are useful to validate the proposed architecture; hence, the setup considered here is too simple to provide insights on the impact of false positive/false negatives.

5.3.2 Network Anomaly Detection Engine and Fault Detection System Analysis in Emulated Scenario

In this scenario, the interaction between and FDS and NADE modules is investigated. Specifically, the FDS is implemented as a nonlinear FDF. The NADE is implemented considering the semantic of the packet payload. In fact, it analyzes the information provided by the Modbus packet concerning the processes and sets up a nominal profile. In this way, it

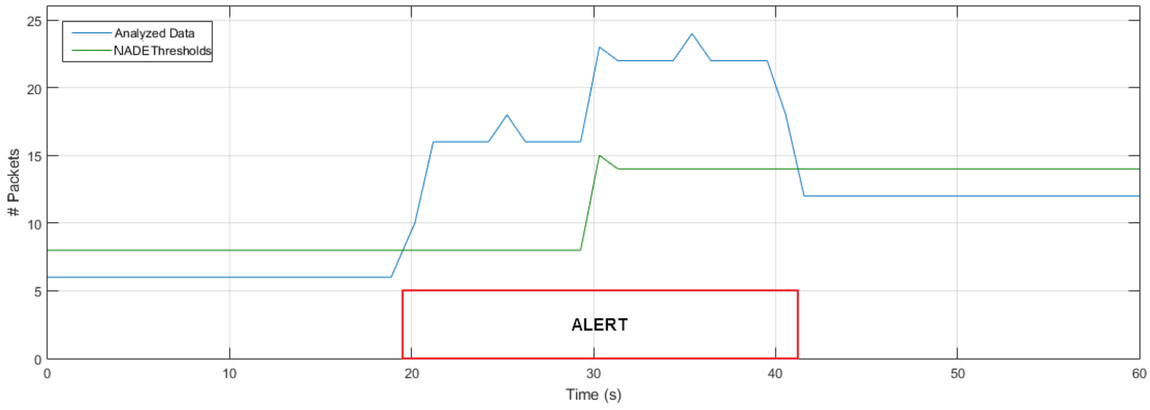


Fig. 5.22 Network under SYN Flood attack with NADE alerting.

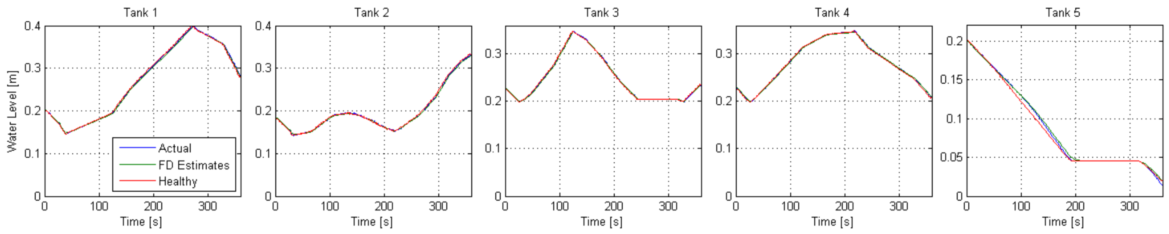


Fig. 5.23 Daily scenario - Sensor measurements (blue), FDS estimates (green) and NADE acquisitions (red).

is possible to investigate how determinate false detections or detection delays of one module can be treated by the other detection system.

Therefore, the FACIES testbed has been exploited to investigate how the monitoring modules react to different physical and cyber issues on the system, considering also the crossed effects of both domains. To this end, the typical daily water demand curve of a city, opportunely scaled down to a $360s$ interval, as shown in Fig.5.23, has been obtained through the sequential opening and closing of the different valves in the manifolds. In the studies carried out, the water system has been employed excluding Tank 5 and the actuators connected to it. This because the implemented scenario does not imply any correlation between such part of the system with the remaining, as shown in Fig. 5.23, thereby the here reported results are equally valid. Moreover, only the water levels in each tank and the water demand from the different areas have been analyzed, neglecting the pressure values in the system.

Physical Faults. The FDS module has been employed to detect physical anomalies, and a wide number of faults have been introduced to the system in order to test its effectiveness and performance. In addition, the detection has been supported by the NADE. Initially, a

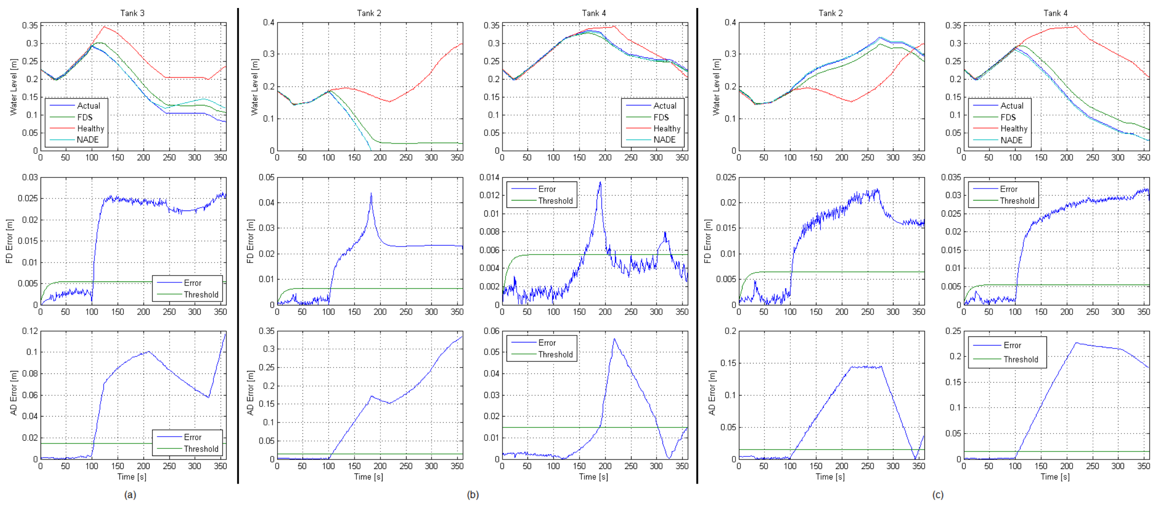


Fig. 5.24 Effects on FDS and NADE of leak fault in Tank 3 (a), fault in Pump 1 (OFF) (b) and fault in Valve D.V.17 (OFF) (c) when performed separately at time $t = 100s$.

leak in Tank 3 has been introduced at time $t = 100$ *s* during the simulation of a daily water demand, as shown in Fig. 5.24(a). It has been observed that such a fault is quickly detected by both the FDS and the NADE, less than 2 s after its occurrence. Indeed, the sharp behavior of the error signals allow a fast and easy detection of the specific fault.

Afterwards, a multiple fault has been introduced at time $t = 100$ s combining a leak in Tank 3, a fault in the supply valve connecting Tank 2 to Tank 4 (Valve D.V.17) and a fault in the pump supplying Tank 2 (Pump 1). Both the actuator faults imply the undesired switch off of the component. Fig. 5.24 illustrates some of the effects of the faults previously described when taking place as single faults.

This combined fault was first detected 2 s after the occurrence, due to the residual associated to Tank 3, which almost instantly exceeds the detection threshold as in the single fault case previously described. The residual associated to Tank 4 indicates the presence of a fault almost simultaneously, related to the Valve V.D.17 closure. To conclude, the residual related to Tank 2 highlights the presence of a fault after 14 *s* of its occurrence.

Comparing the results obtained from the single and multiple faults, it has been observed that the leak in Tank 3 for the multi-faults case is partially compensated by an additional water supply from Tank 1. This was witnessed by a lower water level in Tank 1 with respect to the nominal case, not yet sufficient to trigger a fault.

The fault in Pump 1 makes Tank 2 run out of water very quickly, degrading its supply to Tank 4. As a consequence, see Fig. 5.24(b), an alarm is first triggered by the FDS in Tank 2 at time $t = 103$ s and, after a delay of about 56 s, also in Tank 4. The NADE detection times present a slight delay with respect to the FDS. Similarly, the fault in the supply valve D.V.17

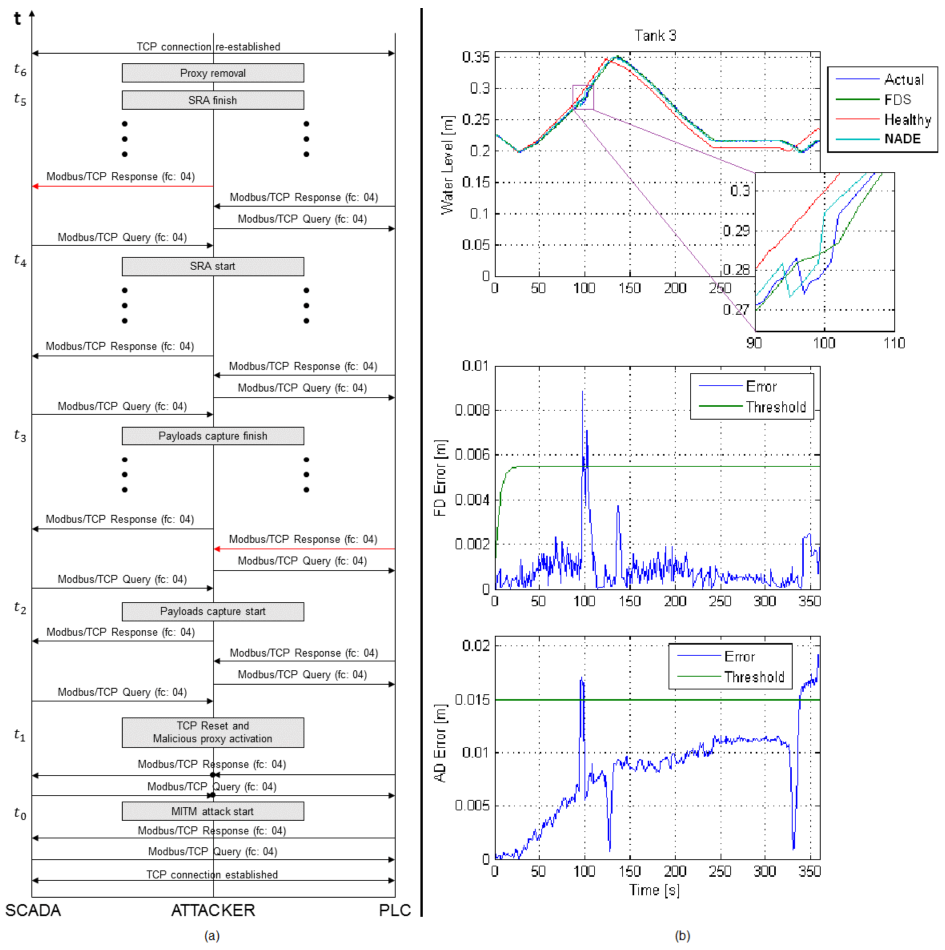

t
TCP connection re-established
t_6
Proxy removal
t_5
SRA finish
Modbus/TCP Response (fc: 04)
Modbus/TCP Response (fc: 04)
Modbus/TCP Query (fc: 04)
Modbus/TCP Query (fc: 04)
t_4
SRA start
Modbus/TCP Response (fc: 04)
Modbus/TCP Response (fc: 04)
Modbus/TCP Query (fc: 04)
Modbus/TCP Query (fc: 04)
t_3
Payloads capture finish
Modbus/TCP Response (fc: 04)
Modbus/TCP Response (fc: 04)
Modbus/TCP Query (fc: 04)
Modbus/TCP Query (fc: 04)
t_2
Payloads capture start
Modbus/TCP Response (fc: 04)
Modbus/TCP Response (fc: 04)
Modbus/TCP Query (fc: 04)
Modbus/TCP Query (fc: 04)
t_1
TCP Reset and Malicious proxy activation
Modbus/TCP Response (fc: 04)
Modbus/TCP Query (fc: 04)
t_0
MITM attack start
Modbus/TCP Response (fc: 04)
Modbus/TCP Query (fc: 04)
TCP connection established
SCADA
ATTACKER
PLC
(a)
Tank 3
Water Level [m]
Actual
FDS
Healthy
NADE
FD Error [m]
Error
Threshold
AD Error [m]
Error
Threshold
Time [s]
(b)

Fig. 5.25 Smart Replay Attack schema (a) at $t = 100$ s for 10 s during the nominal daily scenario - (b) Results for Tank 3.

triggers both FDS and NADE modules, as depicted in Fig. 5.24(c). Though, such an effect is almost simultaneous in this case.

Cyber-Attacks. An availability attack (DoS) and an integrity attack (Smart Replay Attack - SRA) have been taken into account for the experimental stage, where the considered actors were the SCADA/HMI, a PLC, and the attacker. It is assumed that the attacker has already gained access to the local network. The DoS attack has been performed by flooding the network with anomalous data, provoking a degradation of the communication between SCADA and PLCs. The packet flooding can be modulated so to introduce delays or, eventually, totally block the communication for a desired time interval.

The proposed SRA is a more sophisticated replay attack, where only the data payload is replaced into the hijacked packets and the timestamp cannot be used to identify the threat, as it is updated during the attack. In a classical replay attack all the traffic is recorded and later re-transmitted as it is. In Fig. 5.25(a), the SRA scheme is presented. For the sake of simplicity, the packet transmission times are omitted. During normal communication, at time t_0 the attacker starts the MITM attack using the ARP poisoning technique combined with the IP forwarding. In this way, all the traffic flows first through the attacker and it is later re-transmitted to the actual destination, preserving the IP of the original source. At time t_1 the attacker sends a TCP reset in order to enable the *IP-tables* rules for the prerouting/postrouting routines and activates the malicious proxy script. It is assumed that the network components are able to re-establish the connection after a disconnection. At this point, acting as a proxy, the attacker is able to operate on the packets as desired. At time t_2 the payloads capture begins and all the intercepted sensor data information carried by responses from the PLC are stored by the attacker. The payload capture lasts for a prefixed time (t_3). Thereby, all the acquired data are stored and the attacker is ready to perform the replay. The SRA is actually started at time t_4: the malicious proxy is configured to change in the TCP packets only the bytes of the Modbus payload related to the sensor readings. By doing this, the original PLC responses are modified, preserving the TCP relative sequence numbers and replacing the sensor data. At time t_5 the SRA ends, hence the attack has a total duration of $\Delta t = t_3 - t_2 = t_5 - t_4$. Later, at time t_6, the malicious proxy is removed, the *IP-tables* are flushed, the MITM ends and the original network connection is restored performing a TCP reset.

Fig. 5.25(b) shows the experimental results of the SRA for Tank 3, performed during the daily scenario. As previously described, the water level evolution of all the tanks is first recorded at $t \approx 95$ s and then re-transmitted at $t \approx 100$ s. As configured, the replaying period follows the recording one. It is possible to clearly appreciate this behavior in the zoom panel. In this case, the attack triggers almost instantaneously both the FDS and NADE modules, due to the discrepancy between the actual and expected system behavior.

Cyber-Physical Attacks. The concurrent presence of a cyber-attack and a physical fault is more cunning to detect. Fig. 5.26, illustrates a fault taking place in Tank 3 at time $t = 100s$, and a ping flooding DoS against the PLC starting at the same time instant and lasting for 15 s. As one may expect, the physical fault influences only the variable related to the component involved (Tank 3 in this case), while the lack of communication leads to an increase of the FDS and NADE residuals. After the DoS stops, the communication is restored and the FDS error signals go down to acceptable (non-faulty) values, except for the one related to Tank 3, which remains in a faulty state, as expected. Conversely, the NADE reveals a remarkable persisting difference of the water levels from the nominal values, witnessing the consequences of the cyber-attack that may have gone unnoticed with the use of the sole FDS.

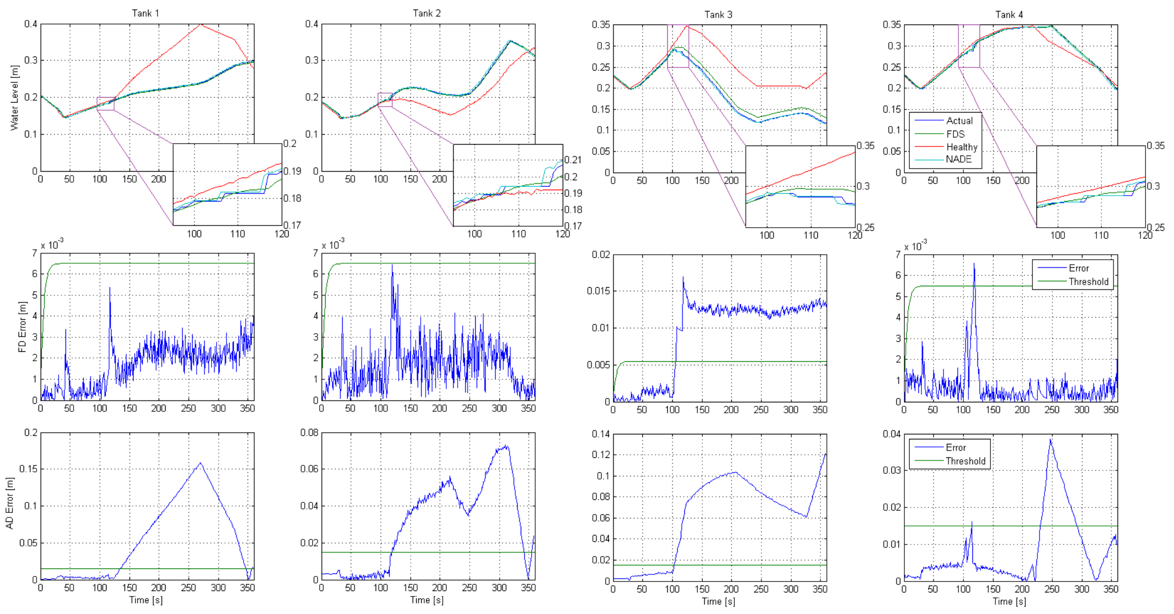


Fig. 5.26 DoS attack to the PLC at $t = 100s$ for $15s$ during the daily scenario - Leak fault Tank 3 at $t = 100s$.

Thereby, if the attacker is able to gain sufficient knowledge of the system and its operation, it would be possible to cover the effects of a cyber-physical attack by designing a proper combination of events. Moreover, in many cases it would not be possible for the operator to distinguish whether the system is undergoing a cyber-attack or a physical anomaly.

5.4 Experimental Results on Decision Support System

In order to apply, validate, and verify the efficiency of the proposed DSS in the field of CI, a simulated scenario is designed with the aim of reproducing a real environment. The proposed scenario, depicted in Fig. 5.27, reproduces the water infrastructure of a residential area divided in two sub areas (A_1, A_2). Both the residential sub areas are supplied by water from a water reserve using two identical pumping systems (p_1, p_2); the two residential areas are connected to each other, and the water flow between them is regulated by the valve v_{12}. The water infrastructure includes a water discharge pipe controlled by a valve (v_1, v_2) for each sub areas. In the daily time, the two pumping systems, p_1 and p_2 are activated to supply water to A_1 and A_2 depending on the water levels and the water demands; the valve v_{12} is normally closed.

In order to test the efficiency of the proposed decision making approach, the scenario has been implemented over the HYDRA testbed using only a part of the entire system as depicted in Fig 5.28. Tanks T_1 and T_2 are used to reproduce the simulated environment represented

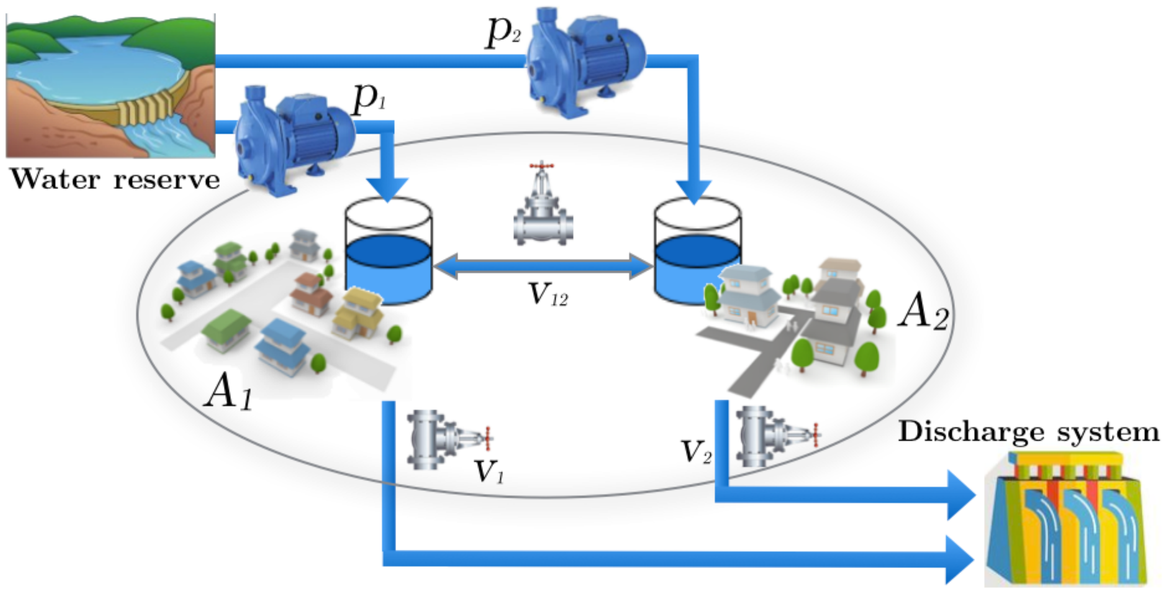


Fig. 5.27 Analyzed water distribution system.

by the sub areas A_1 and A_2. Both the tanks are refilled by pumps p_1 and p_2. The tanks are connected to each other reproducing perfectly the scenario.

In order to explain the idea behind the AHP methodology implemented for the security operator decision support, a *Data Modification* cyber-attack against the HYDRA testbed network has been exploited. In Fig. 5.29 the phases related to a cyber-attack event against a CI are described. During the preliminary stage all the specifications are studied and the different components of the system are initially designed and implemented in order to enable the system to be operative (Setup). The DSS and the DDS are conceived in this stage. In the daily operating scheme, an attack situation is modeled. During the normal operation routines the system is assumed in a safe state. Considering the classical approach, in case of cyber threats the DDS is configured to show an alert to the operator. The novelty is the introduction of the DSS: the alert coming from the DDS is exploited by the DSS which provides the alert to the human operator with a computed suggestion related to the operation to be carried out. Taking into account the DSS advice, the operator makes a decision and applies the countermeasure. The AHP methodology allows to compare different options related to a plurality of criteria, quantitative or qualitative, to assess an overall evaluation for each of them, ordering the alternatives according to specific preferences, and finally to select the best solution.

In order to apply AHP it is necessary to explain the assumptions for building and implementation of the decision model:

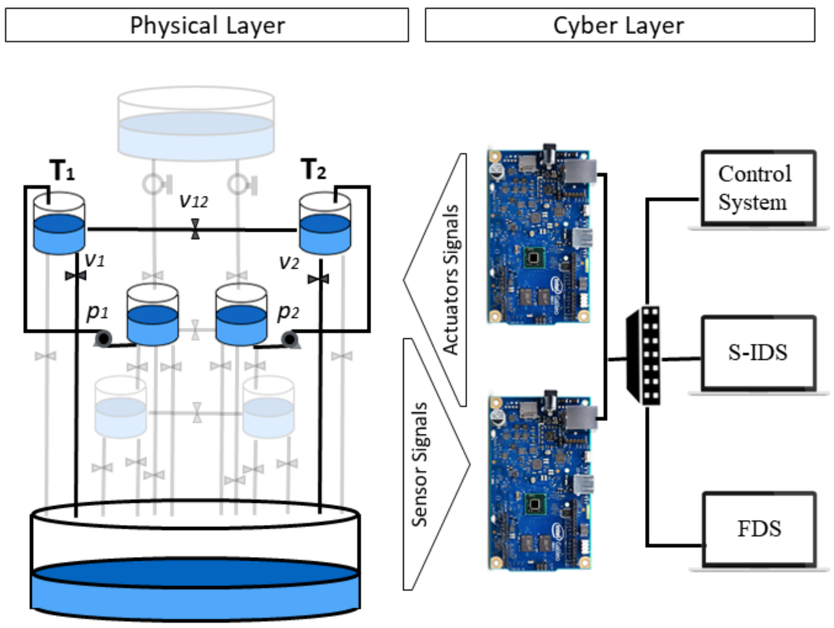


Fig. 5.28 HYDRA cyber and physical layers for the experiment.

- The presence of a SCADA system, with network components, perfectly functioning;
- The installation of a DDS component able to certainly identify malicious action on the network;
- The decision maker intelligence, which is able to check the system status through the analysis of data and indexes monitored by the operator interface;
- The dominance hierarchy and eligible alternatives identification that allow to apply the AHP method;
- The operator instantly obtains the AHP result and suggestion;
- The decision maker alternatives are all predetermined actions and valid for the system;
- The operator can not be replaced by the decision support method, he will have to make the final decision.

In case of cyber-attack against the system, the assumption definition listed above would easily allow the detection of the system section compromised by the attacker. For instance, in emergency situations, the operator could be faced with the decision to isolate the system section under attack; this action would have different consequences in terms of security of the system, economic, social and environmental impacts.

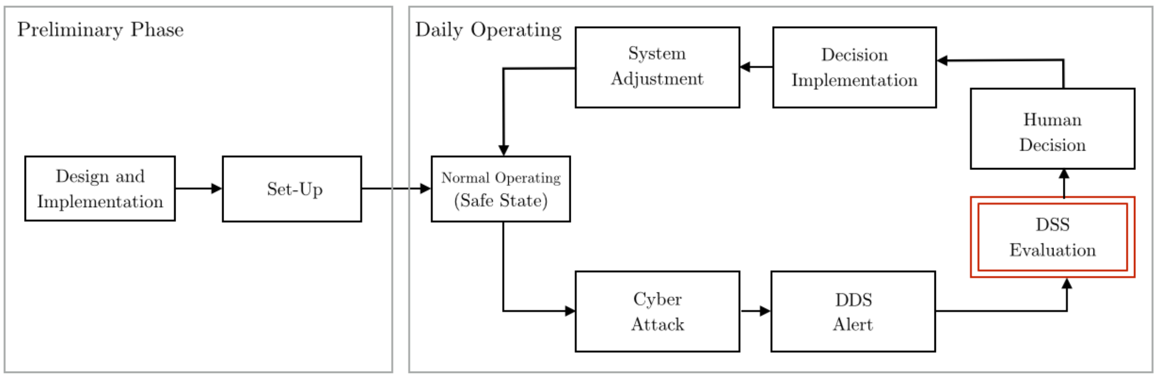


Fig. 5.29 Phases related to a cyber-attack event against a CI with the DSS.

For the case study, the hierarchy of the AHP method is built in two successive phases:

- *Criteria and sub-criteria set up*: it is made at an early stage on the basis of the issue considered (in this case the goal is the CI restoring after a cyber-attack) and on the basis of the company policy.
- *Alternatives identification*: A predefined number of alternative sets are defined considering the components of the system under possible cyber-attack. Only after the attacked component identification, the right set of alternatives is evaluated and selected for the operator suggestion. This represents a novel alternatives characterization necessary for the research.

For the sake of clarity, the *Data Modification* attack in this context allows to change the values read from the water level sensors without disrupt the overall system functionality. The DDS detects an intrusion in the cyber layer of the testbed and the misalignment, between the estimated water level in the tank T_1 and the measure h_1, highlights a *Data Modification* attack over the water level sensor in T_1. Once the intrusion is detected and the attack is identified, the DSS evaluates a set of alternatives, planned for this specific situation, and provides the operator with the best one. In order to obtain the system restoring, the DSS compares three possible alternative, α_1, α_2, and, α_3:

- Alternative α_1: Keep pump P_1 active;
- Alternative α_2: Pump P_1 deactivation;
- Alternative α_3: Pump P_1 deactivation and valve V_{12} opening.

Each alternative is evaluated on the basis of three criteria c_1, c_2, and c_3:

- Economic criteria (c_1): in terms of costs that the plant company should support;

- Social and Environmental criteria (c_2): is based on the impact of an action on the system in social and environmental terms;

- Security criteria (c_3): it is intended as the system vulnerability against external threats.

Referring to the case study, three *Pairwise Comparison Matrices* (A_1, A_2, A_3) are computed evaluating each pair of alternatives. In the following, the alternatives evaluation based on the criteria c_1, c_2, and c_3 are explained according to the Saaty's scale (Tab 2.1). As already mentioned before, the rationale behind the values selection is justified by a methodology explanation purpose:

$$A_1 = \begin{bmatrix} 1 & 1/3 & 1/3 \\ 3 & 1 & 3 \\ 3 & 1/3 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & 3 & 1/7 \\ 1/3 & 1 & 1/5 \\ 7 & 5 & 1 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} 1 & 1/7 & 1/5 \\ 7 & 1 & 5 \\ 5 & 1/5 & 1 \end{bmatrix}.$$

Observing the entries of A_1,A_2, A_3 we can make the following assumptions about the experts evaluations.

Concerning the criteria c_1 :

- The alternative α_2 is weakly important respect to the alternative α_1 and α_3;

- The alternative α_3 is weakly important respect to the alternative α_1;

Concerning the criteria c_2 :

- The alternative α_1 is weakly important respect to the alternative α_2;

- The alternative α_3 is strongly important respect to the alternative α_2, and has a proved relevance respect to the alternative α_1;

Concerning the criteria c_3 :

- The alternative α_2 is strongly important respect to the alternative α_3, and has a proved relevance respect to the alternative α_1;

- The alternative α_3 is strongly important respect to the alternative α_1.

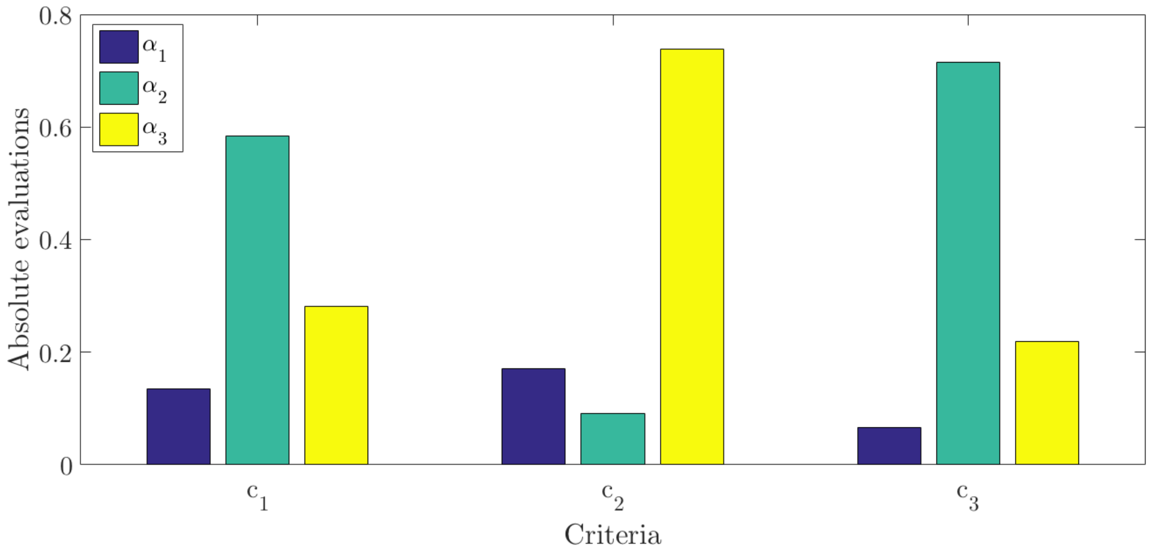


Fig. 5.30 Absolute evaluations of α_1,α_2, and α_3 on the basis of c_1,c_2, and c_3.

The normalized eigenvectors are: $\mathbf{w}^{(\mathbf{1})} = [0.13, 0.58, 0.29]^T$, $\mathbf{w}^{(\mathbf{2})} = [0.17, 0.09, 0.74]^T$, $\mathbf{w}^{(\mathbf{3})} = [0.07, 0.71, 0.21]^T$. This results shows that on the basis of the economic criteria, the best applicable alternative is α_2, respect to the social and environmental criteria (c_2), the best applicable alternative is α_3, and concerning the security criteria (c_3) the suggested alternative is α_2. The absolute weight represented by the entries of $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}$, and $\mathbf{w}^{(3)}$ are resumed in Fig. 5.30.

In order to evaluate the three criteria c_1, c_2, and c_3 the matrix Q and the eigenvector $\mathbf{z}$ are computed:

$$Q = \begin{bmatrix} 1 & 3 & 1/5 \\ 1/3 & 1 & 1/7 \\ 5 & 7 & 1 \end{bmatrix}, \qquad \mathbf{z} = [0.19, 0.08, 0.73]^T.$$

It follows that the criteria c_3 has significant weight respect to the other criteria.

Aiming to provide the best alternative, the equation in (2.27) is computed:

$$i^* = \underset{i=1,\dots 3}{\arg\max} \left\{ \sum_{j=1}^{3} z_j w_{i,j} \right\} = \underset{i=1,\dots 3}{\arg\max} \{0.09, 0.64, 0.27\} = 2.$$

According to Fig. 5.31, it results that the choice suggested by the AHP approach is α_2.

It is worth mentioning that the suggested choices can be retrieved by simple matrix computation that due not overload modern computers.

Performing the previously described AHP resolution, we get the hierarchy achieved at the starting stage, weighted on the opinions related to criteria and alternatives expressed in the

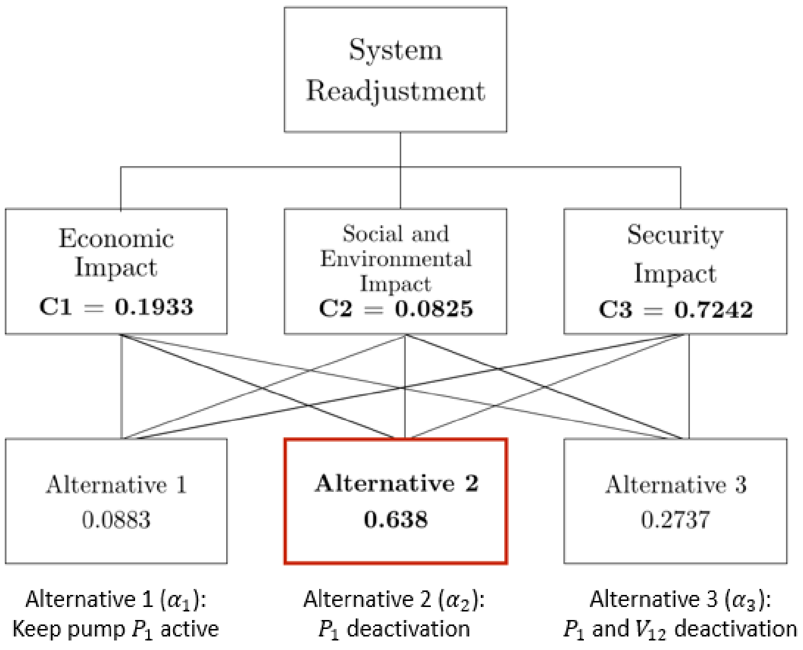

System Readjustment
Economic Impact
C1 = 0.1933
Social and Environmental Impact
C2 = 0.0825
Security Impact
C3 = 0.7242
Alternative 1
0.0883
Alternative 2
0.638
Alternative 3
0.2737
Alternative 1 (α_1): Keep pump P_1 active
Alternative 2 (α_2): P_1 deactivation
Alternative 3 (α_3): P_1 and V_{12} deactivation

Fig. 5.31 Best alternative proposal.

initial phase: that's the way it is possible to describe the priorities of the actions considered in a given scenario. A ranking of the alternatives referred to the overall weight of each alternative based on all the relevant criteria is achieved. The results justify the preference on the second alternative, which is the P_1 temporary shutdown. When a given cyber-attack occurs, the security operator will have the suggestion given by the AHP method, but the responsibility related the final decision on the specific countermeasure to carry out is up only to him.

Concluding, CIs represent the backbone of our society, even a partial compromise of their operation would bring to considerable unpleasant consequences for the population. In such systems, the human operator ability to make decisions aimed at minimize the consequences is an unavoidable necessity. As before mentioned, the responsibility of all the critical choices falls on the security operators and on their own experience. The integration of a classical decision support method with the decision-making phase of critical infrastructures' emergencies has been presented as useful tool for the security of ICSs for CIs.

Chapter 6

Conclusion and Future Works

The emerging use of information and communication technologies in ICSs raises concerns about the vulnerabilities of the CIs to cyber-attacks. Therefore, the cyber-security of ICSs for CIs became an hot topic, since they are applied to control and monitor large physical infrastructures. Moreover, the usage of commercial cyber security products makes the ICSs prone to failures due to correlated hardware and software faults. Thus, as CPS, the ICSs for CIs inherit the vulnerabilities of both the cyber and the physical components, making these systems safety and security critical.

This motivates the interest in designing a novel architecture to secure ICSs for CIs that addresses both physical and cyber threats in a unified framework. The architecture foresees the human in the loop paradigm, to further exploits the complementary ability of automatic systems and the human decision-making skills.

From the physical perspective, the architecture is based on the existing research framework in robust and fault-tolerant control system. This theoretical framework is applied to identify cyber-attacks on the communication links. From the cyber perspective, the classical intrusion detection tools are adopted and improved. The main novelty relies on the use of the semantic level in the analysis of the traffic load. In fact, the payload of the communication packets is analyzed at semantic level.

The interaction with human in considered by developing a proper DSS, able to exploit the current state of the system and a priori knowledge of experts to help the operators in making decisions when failures occur.

Furthermore, the proposed architecture envisages the use of forensic tools. The Mimepot, indeed, has been designed to collect information about the impact of unforeseeable cyber-attacks to the physical level.

The work presented in this thesis can be of particular interest to the next generation ICSs for CIs that will be extensively used to implement the Smartcity and Industry 4.0 paradigms.

Beyond the results obtained, there are still room of improvement and topics that still need to be properly addressed.

Concerning the exploitation of physical plant to identify cyber-attacks, future works can address a class of model-based detection schemes, formulated as an adversarial game between the detection system and the attacker.

Concerning the development of the Network Anomaly Detection Engine, it can use the semantic level proving the sequence of command and response, rather than the whole behavior of the system.

Simple security approaches, such as light encryption, can be adopted to securing the communication link and, at the same time, to easily identify stealth attacks as proposed in [122].

References

[1] André Teixeira, Iman Shames, Henrik Sandberg, and Karl Henrik Johansson. A secure control framework for resource-limited adversaries. *Automatica*, 51:135–148, 2015.

[2] ISA99 committee. ISA99, Industrial Automation and Control Systems Security. *International Society of Automation (ISA)*.

[3] Alvaro A Cárdenas, Saurabh Amin, and Shankar Sastry. Research challenges for the security of control systems. In *HotSec*, 2008.

[4] J. Slay and M. Miller. *Lessons Learned from the Maroochy Water Breach*, volume 253 of *IFIP*, chapter Critical Infrastructure Protection - Part II, pages 73–82. Springer, 2008.

[5] SANS. State of ICS security survey. Retrieved from: https://www.sans.org/reading-room/whitepapers/analyst/2016-state-ics-security-survey-37067, 2016.

[6] Kaspersky Lab. Threat landscape for industrial automation systems in the second half of 2016. Technical report, 2016.

[7] N. Falliere, L.O. Murchu, and E. Chien. W32. Stuxnet Dossier. Technical Report 1.4, Symantec, February 2011.

[8] TRISIS Malware - Analysis of Safety System Targeted Malware (https://dragos.com/blog/trisis/TRISIS-01.pdf). Technical report, Dragos Inc., 2017.

[9] Matthew A Bishop. *The art and science of computer security*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[10] Bonnie Zhu, Anthony Joseph, and Shankar Sastry. A taxonomy of cyber attacks on scada systems. In *Internet of things (iThings/CPSCom), 2011 international conference on and 4th international conference on cyber, physical and social computing*, pages 380–388. IEEE, 2011.

[11] Keith Stouffer, Victoria Pillitteri, Suzanne Lightman, Marshall Abrams, and Adam Hahn. Guide to industrial control systems (ics) security. Technical report, 2015.

[12] Al-Sakib Khan Pathan. *The state of the art in intrusion prevention and detection*. CRC press, 2014.

[13] Rolf Isermann. *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media, 2006.

[14] Zhong-Hua Pang and Guo-Ping Liu. Design and implementation of secure networked predictive control systems under deception attacks. *IEEE Transactions on Control Systems Technology*, 20(5):1334–1342, 2012.

[15] Siddharth Sridhar, Adam Hahn, and Manimaran Govindarasu. Cyber–physical system security for the electric power grid. *Proceedings of the IEEE*, 100(1):210–224, 2012.

[16] André Teixeira, Daniel Pérez, Henrik Sandberg, and Karl Henrik Johansson. Attack models and scenarios for networked control systems. In *Proceedings of the 1st international conference on High Confidence Networked Systems*, pages 55–64. ACM, 2012.

[17] L. Cazorla, C. Alcaraz, and J. Lopez. Cyber Stealth Attacks in Critical Information Infrastructures. *IEEE Systems Journal*, pages 1–15, 03/2016 2016.

[18] Saurabh Amin, Alvaro A Cárdenas, and Shankar Sastry. Safe and secure networked control systems under denial-of-service attacks. In *HSCC*, volume 5469, pages 31–45. Springer, 2009.

[19] Abhishek Gupta, Cédric Langbort, and Tamer Başar. Optimal control in the presence of an intelligent jammer with limited actions. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 1096–1101. IEEE, 2010.

[20] Yilin Mo and Bruno Sinopoli. Secure control against replay attacks. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pages 911–918. IEEE, 2009.

[21] Roy S Smith. A decoupled feedback structure for covertly appropriating networked control systems. *IFAC Proceedings Volumes*, 44(1):90–95, 2011.

[22] Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo. Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 2195–2201. IEEE, 2011.

[23] Ali Khanafer, Behrouz Touri, and Tamer Başar. Consensus in the presence of an adversary. *IFAC Proceedings Volumes*, 45(26):276–281, 2012.

[24] Shreyas Sundaram, Shai Revzen, and George Pappas. A control-theoretic approach to disseminating values and overcoming malicious links in wireless networks. *Automatica*, 48(11):2894–2901, 2012.

[25] Peyman Mohajerin Esfahani, Maria Vrakopoulou, Kostas Margellos, John Lygeros, and Göran Andersson. Cyber attack in a two-area power system: Impact identification using reachability. In *American Control Conference (ACC), 2010*, pages 962–967. IEEE, 2010.

[26] Yilin Mo and Bruno Sinopoli. Integrity attacks on cyber-physical systems. In *Proceedings of the 1st international conference on High Confidence Networked Systems*, pages 47–54. ACM, 2012.

[27] Yao Liu, Peng Ning, and Michael K Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):13, 2011.

[28] Oliver Kosut, Liyan Jia, Robert J Thomas, and Lang Tong. Malicious data attacks on smart grid state estimation: Attack strategies and countermeasures. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 220–225. IEEE, 2010.

[29] Henrik Sandberg, André Teixeira, and Karl H Johansson. On security indices for state estimators in power networks. In *First Workshop on Secure Control Systems (SCS), Stockholm, 2010*, 2010.

[30] André Teixeira, György Dán, Henrik Sandberg, and Karl H Johansson. A cyber security study of a scada energy management system: Stealthy deception attacks on the state estimator. *IFAC Proceedings Volumes*, 44(1):11271–11277, 2011.

[31] Le Xie, Yilin Mo, and Bruno Sinopoli. False data injection attacks in electricity markets. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 226–231. IEEE, 2010.

[32] Alvaro A Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM symposium on information, computer and communications security*, pages 355–366. ACM, 2011.

[33] Ahmad-Reza Sadeghi, Christian Wachsmann, and Michael Waidner. Security and privacy challenges in industrial internet of things. In *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, pages 1–6. IEEE, 2015.

[34] Karen Scarfone and Peter Mell. Guide to intrusion detection and prevention systems (IDPS). *NIST special publication*, 800(2007):94, 2007.

[35] Snort IDS, (https://www.snort.org/).

[36] Bro IDS, (https://www.bro.org/).

[37] Suricata IDS, (https://suricata-ids.org/).

[38] Zhimin Zhou, Chen Zhongwen, Zhou Tiecheng, and Guan Xiaohui. The study on network intrusion detection system of snort. In *Networking and Digital Society (ICNDS), 2010 2nd International Conference on*, volume 2, pages 194–196. IEEE, 2010.

[39] Michael Attig and John Lockwood. Sift: Snort intrusion filter for tcp. In *High Performance Interconnects, 2005. Proceedings. 13th Symposium on*, pages 121–127. IEEE, 2005.

[40] Hamed Salehi, Hossein Shirazi, and Reza Askari Moghadam. Increasing overall network security by integrating signature-based nids with packet filtering firewall. In *Artificial Intelligence, 2009. JCAI'09. International Joint Conference on*, pages 357–362. IEEE, 2009.

[41] Mohammed Abdul Qadeer, Arshad Iqbal, Mohammad Zahid, and Misbahur Rahman Siddiqui. Network traffic analysis and intrusion detection using packet sniffer. In *Communication Software and Networks, 2010. ICCSN'10. Second International Conference on*, pages 313–317. IEEE, 2010.

[42] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. Network anomaly detection: methods, systems and tools. *Ieee communications surveys & tutorials*, 16(1):303–336, 2014.

[43] Paulo M Mafra, Vinicius Moll, Joni da Silva Fraga, and Altair Olivo Santin. Octopus-IDS: An anomaly based intelligent intrusion detection system. In *Computers and Communications (ISCC), 2010 ieee Symposium on*, pages 405–410. IEEE, 2010.

[44] Patrick Düssel, Christian Gehl, Pavel Laskov, Jens-Uwe Bußer, Christof Störmann, and Jan Kästner. Cyber-critical infrastructure protection using real-time payload-based anomaly detection. In *CRITIS*, pages 85–97. Springer, 2009.

[45] Song Han, Miao Xie, Hsiao-Hwa Chen, and Yun Ling. Intrusion detection in cyber-physical systems: Techniques and challenges. *IEEE Systems Journal*, 8(4):1052–1062, 2014.

[46] Dayu Yang, Alexander Usynin, and J Wesley Hines. Anomaly-based intrusion detection for scada systems. In *5th intl. topical meeting on nuclear plant instrumentation, control and human machine interface technologies (npic&hmit 05)*, pages 12–16, 2006.

[47] Andrea Carcano, Igor Nai Fovino, Marcelo Masera, and Alberto Trombetta. State-based network intrusion detection systems for scada protocols: A proof of concept. Springer, 2009.

[48] Igor Nai Fovino, Andrea Carcano, Thibault De Lacheze Murel, Alberto Trombetta, and Marcelo Masera. Modbus/dnp3 state-based intrusion detection system. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 729–736. IEEE, 2010.

[49] Salvatore D'Antonio, Francesco Oliviero, and Roberto Setola. High-speed intrusion detection in support of critical infrastructure protection. In *International Workshop on Critical Information Infrastructures Security*, pages 222–234. Springer, 2006.

[50] Áine MacDermott, Qi Shi, Madjid Merabti, and Kashif Kifayat. Intrusion detection for critical infrastructure protection. In *13th Annual Postgraduate Symposium on Convergence of Telecommunications, Networking and Broadcasting (PGNet 2012)*, 2012.

[51] Robert Mitchell and Ing-Ray Chen. A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys (CSUR)*, 46(4):55, 2014.

[52] Thomas Morris, Rayford Vaughn, and Yoginder Dandass. A retrofit network intrusion detection system for modbus rtu and ascii industrial control systems. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 2338–2345. IEEE, 2012.

[53] Rafael Ramos Regis Barbosa. *Anomaly detection in scada systems-a network based approach*. PhD thesis, Centre for Telematics and Information Technology, University of Twente, 2014.

[54] Noam Erez and Avishai Wool. Control variable classification, modeling and anomaly detection in modbus/tcp scada systems. *International Journal of Critical Infrastructure Protection*, 10:59–70, 2015.

[55] Marco Caselli, Emmanuele Zambon, Johanna Amann, Robin Sommer, and Frank Kargl. Specification mining for intrusion detection in networked control systems. 2016.

[56] Marco Caselli, Emmanuele Zambon, and Frank Kargl. Sequence-aware intrusion detection in industrial control systems. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, pages 13–24. ACM, 2015.

[57] Dina Hadiosmanovic, Damiano Bolzoni, Pieter Hartel, and Sandro Etalle. Melissa: Towards automated detection of undesirable user actions in critical infrastructures. In *Computer Network Defense (EC2ND), 2011 Seventh European Conference on*, pages 41–48. IEEE, 2011.

[58] Leandros A Maglaras and Jianmin Jiang. Intrusion detection in scada systems using machine learning techniques. In *Science and Information Conference (SAI), 2014*, pages 626–631. IEEE, 2014.

[59] YooJin Kwon, Huy Kang Kim, Yong Hun Lim, and Jong In Lim. A behavior-based intrusion detection technique for smart grid infrastructure. In *PowerTech, 2015 IEEE Eindhoven*, pages 1–6. IEEE, 2015.

[60] Niv Goldenberg and Avishai Wool. Accurate modeling of modbus/tcp for intrusion detection in scada systems. *International Journal of Critical Infrastructure Protection*, 6(2):63–75, 2013.

[61] Amit Kleinmann and Avishai Wool. Automatic construction of statechart-based anomaly detection models for multi-threaded industrial control systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(4):55, 2017.

[62] Mustafa Faisal, Alvaro A Cardenas, and Avishai Wool. Modeling modbus tcp for intrusion detection. In *Communications and Network Security (CNS), 2016 IEEE Conference on*, pages 386–390. IEEE, 2016.

[63] Helge Janicke, Andrew Nicholson, Stuart Webber, and Antonio Cau. Runtime-monitoring for industrial control systems. *Electronics*, 4(4):995–1017, 2015.

[64] Mark Luchs and Christian Doerr. Last line of defense: A novel ids approach against advanced threats in industrial control systems. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 141–160. Springer, 2017.

[65] Tiago Cruz, Luis Rosa, Jorge Proença, Leandros Maglaras, Matthieu Aubigny, Leonid Lev, Jianmin Jiang, and Paulo Simões. A cybersecurity detection framework for supervisory control and data acquisition systems. *IEEE Transactions on Industrial Informatics*, 12(6):2236–2246, 2016.

[66] Hamid Reza Ghaeini and Nils Ole Tippenhauer. Hamids: Hierarchical monitoring intrusion detection system for industrial control systems. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, pages 103–111. ACM, 2016.

[67] Aditya P Mathur and Nils Ole Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *Cyber-physical Systems for Smart Water Networks (CySWater), 2016 International Workshop on*, pages 31–36. IEEE, 2016.

[68] Nazrul Hoque, Monowar H Bhuyan, Ram Charan Baishya, Dhruba K Bhattacharyya, and Jugal K Kalita. Network attacks: Taxonomy, tools and systems. *Journal of Network and Computer Applications*, 40:307–324, 2014.

[69] K Stouffer, S Lightman, V Pillitteri, M Abrams, and A Hahn. Nist special publication 800-82 revision 2, guide to industrial control systems (ics) security. *Gaithersburg, MD, USA: National Institute of Standards and Technology*, 2014.

[70] Security Onion, (https://securityonion.net/).

[71] Security Onion, (https://www.wireshark.org/).

[72] Niels Provos. A virtual honeypot framework. In *USENIX Security Symposium*, volume 173, pages 1–14, 2004.

[73] Paulo Simões, Tiago Cruz, Jorge Gomes, and Edmundo Monteiro. On the use of honeypots for detecting cyber attacks on industrial control networks. In *Proc. 12th Eur. Conf. Inform. Warfare Secur. ECIW 2013*, 2013.

[74] Paulo Simões, Tiago Cruz, Jorge Proença, and Edmundo Monteiro. Specialized honeypots for scada systems. In *Cyber Security: Analytics, Technology and Automation*, pages 251–269. Springer, 2015.

[75] Daniele Antonioli, Anand Agrawal, and Nils Ole Tippenhauer. Towards high-interaction virtual ICS honeypots-in-a-box. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*. ACM, 2016.

[76] Conpot Honeypot, (https://github.com/mushorg/conpot).

[77] Kali Linux, (https://www.kali.org/).

[78] Nmap tool, (https://nmap.org/).

[79] Nping tool, (https://nmap.org/nping/).

[80] Scapy, (http://www.secdev.org/projects/scapy/).

[81] Ettercap, (https://ettercap.github.io/ettercap/).

[82] Social Engineering Toolkit (SET), (https://github.com/trustedsec/social-engineer-toolkit).

[83] David Maynor. *Metasploit toolkit for penetration testing, exploit development, and vulnerability research.* Elsevier, 2011.

[84] Chad Dougherty, Kirk Sayre, Robert C Seacord, David Svoboda, and Kazuya Togashi. Secure design patterns. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2009.

[85] Luís Rosa, Tiago Cruz, Paulo Simões, Edmundo Monteiro, and Leonid Lev. Attacking scada systems: A practical perspective. In *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, pages 741–746. IEEE, 2017.

[86] S.X. Ding. *Model-Based Fault Diagnosis Techniques. Design Schemes, Algorithms and Tools.* Advances in Industrial Control. Springer, 2013.

[87] Jose Rubio-Hernan, Luca De Cicco, and Joaquin Garcia-Alfaro. On the use of watermark-based schemes to detect cyber-physical attacks. *EURASIP Journal on Information Security*, 2017(1):8, 2017.

[88] Fabio Pasqualetti. *Secure control systems: a control-theoretic approach to cyber-physical security.* University of California, Santa Barbara, 2012.

[89] Quanyan Zhu and Tamer Basar. 17 a hierarchical security architecture for smart grid. 2012.

[90] G.F. Franklin, J. D. Powell, and M.L. Workman. Digital control of dynamic systems. 1998.

[91] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[92] David I Urbina, Jairo A Giraldo, Alvaro A Cardenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. Limiting the impact of stealthy attacks on industrial control systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1092–1105. ACM, 2016.

[93] Yilin Mo, Sean Weerakkody, and Bruno Sinopoli. Physical authentication of control systems: Designing watermarked control inputs to detect counterfeit sensor outputs. *IEEE Control Systems*, 35(1):93–109, 2015.

[94] B Brumback and M Srinath. A chi-square test for fault-detection in kalman filters. *IEEE Transactions on Automatic Control*, 32(6):552–554, 1987.

[95] Yuancheng Li and Shengnan Chu. Construction and Reduction Methods of Vulnerability Index System in Power SCADA. *International Journal of Security and Its Applications*, 8(6), 2014.

[96] I. Ahmad, A. B. Abdullah, and A. S. Alghamdi. Comparative Analysis of Intrusion Detection Approaches. In *Computer Modelling and Simulation (UKSim), 2010 12th International Conference on*, pages 586–591. IEEE, 2010.

[97] M. L. Carreño, O. D. Cardona, and A. H. Barbat. A disaster risk management performance index. *Natural Hazards*, 41(1):1–20, 2007.

[98] M. Dağdeviren and İ. Yüksel. Developing a fuzzy analytic hierarchy process (AHP) model for behavior-based safety management. *Information Sciences*, 178(6):1717–1733, 2008.

[99] H.H. Goh, B.C. Kok, S.W. Lee, and A.A.B.M. Zin. Load Shedding Scheme in Large Pulp Mill by Using Analytic Hierarchy Process. In *AIP Conference Proceedings-American Institute of Physics*, volume 1337, page 57, 2011.

[100] C. J. Macuada, A. M. Oddershede, and R. Alarcon. Multi-criteria assessment to automate water treatment plants using the analytical hierarchy process. *Journal for Global Business Advancement*, 8(2):236–246, 2015.

[101] M. A. Mustafa and J. F. Al-Bahar. Project risk analytic assessment using the hierarchy process. *IEEE transactions on engineering management*, 38(1):46–52, 1991.

[102] P. K. Dey. Project risk management: a combined analytic hierarchy process and decision tree approach. *Cost Engineering*, 44(3):13–27, 2002.

[103] I. Syamsuddin and H. Junseok. The application of AHP model to guide decision makers: a case study of e-banking security. In *Computer Sciences and Convergence Information Technology, 2009. ICCIT'09. Fourth International Conference on*, pages 1469–1473. IEEE, 2009.

[104] A. Ishizaka and L. Ashraf. Review of the main developments in the analytic hierarchy process. *Expert systems with applications*, 38(11):14336–14345, 2011.

[105] T. L. Saaty. The analytic hierarchy and analytic network processes for the measurement of intangible criteria and for decision-making. In *Multiple criteria decision analysis: state of the art surveys*, pages 345–405. Springer, 2005.

[106] Luciana Obregon. Secure architecture for industrial control systems. *SANS Institute InfoSec Reading Room*, 2015.

[107] Giuseppe Bernieri, Federica Pascucci, and Javier Lopez. Network anomaly detection in critical infrastructure based on mininet network simulator. In *ITASEC 2017*, 2017.

[108] Giuseppe Bernieri, Estefanía Etchevés Miciolino, Federica Pascucci, and Roberto Setola. Monitoring system reaction in cyber-physical testbed under cyber-attacks. *Computers & Electrical Engineering*, 2017.

[109] Estefanía Etchevés Miciolino, Roberto Setola, Giuseppe Bernieri, Stefano Panzieri, Federica Pascucci, and Marios M Polycarpou. Fault diagnosis and network anomaly detection in water infrastructures. *IEEE Design & Test*, 34(4):44–51, 2017.

[110] IDA Modbus. Modbus Application Protocol. *v1.1b*, pages 1–51, 2006.

[111] IDA Modbus. Modbus messaging on TCP/IP implementation guide. *v1.0b*, 2004.

[112] Richard Vernon Beard. *Failure accomodation in linear systems through self-reorganization.* PhD thesis, Massachusetts Institute of Technology, 1971.

[113] Kate Greene. Tr10: Software-defined networking. *Technology Review (MIT)*, 2009.

[114] Giuseppe Bernieri, Fabio Del Moro, Luca Faramondi, and Federica Pascucci. A testbed for integrated fault diagnosis and cyber security investigation. In *Control, Decision and Information Technologies (CoDIT), 2016 International Conference on*, pages 454–459. IEEE, 2016.

[115] F. Pasqualetti, F. Dorfler, and F. Bullo. Control-Theoretic Methods for Cyberphysical Security: Geometric Principles for Optimal Cross-Layer Resilient Control Systems. *IEEE Control Systems*, 35(1):110–127, 2015.

[116] Node.js, (https://nodejs.org/it/).

[117] Daniele Antonioli and Nils Ole Tippenhauer. Minicps: A toolkit for security research on cps networks. In *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*, pages 91–100. ACM, 2015.

[118] Mininet, An Instant Virtual Network on your Laptop (or other PC), www.mininet.org.

[119] Estefanía Etchevés Miciolino, Giuseppe Bernieri, Federica Pascucci, and Roberto Setola. Communications network analysis in a SCADA system testbed under cyber-attacks. In *Telecommunications Forum (TELFOR), 23rd*, pages 341–344. IEEE, 2015.

[120] Giuseppe Bernieri, Stefano Damiani, Fabio Del Moro, Luca Faramondi, Federica Pascucci, and Francesco Tambone. A Multiple-Criteria Decision Making method as support for critical infrastructure protection and Intrusion Detection System. In *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*, pages 4871–4876. IEEE, 2016.

[121] W. M. Eddy. SYN Flood Attack. In *Encyclopedia of Cryptography and Security*, pages 1273–1274. Springer, 2011.

[122] Fei Miao, Quanyan Zhu, Miroslav Pajic, and George J Pappas. Coding schemes for securing cyber-physical systems against stealthy data injection attacks. *IEEE Transactions on Control of Network Systems*, 4(1):106–117, 2017.

[123] E. Etchevés Miciolino, F. Pascucci, J. Lopez, M.M. Polycarpou, and R. Setola. FACIES: a Testbed for Distributed Fault and Attack Identification in Interdependent Critical Infrastructures. In *2nd International SCADALab Workshop, Seville (Spain)*, 2014.

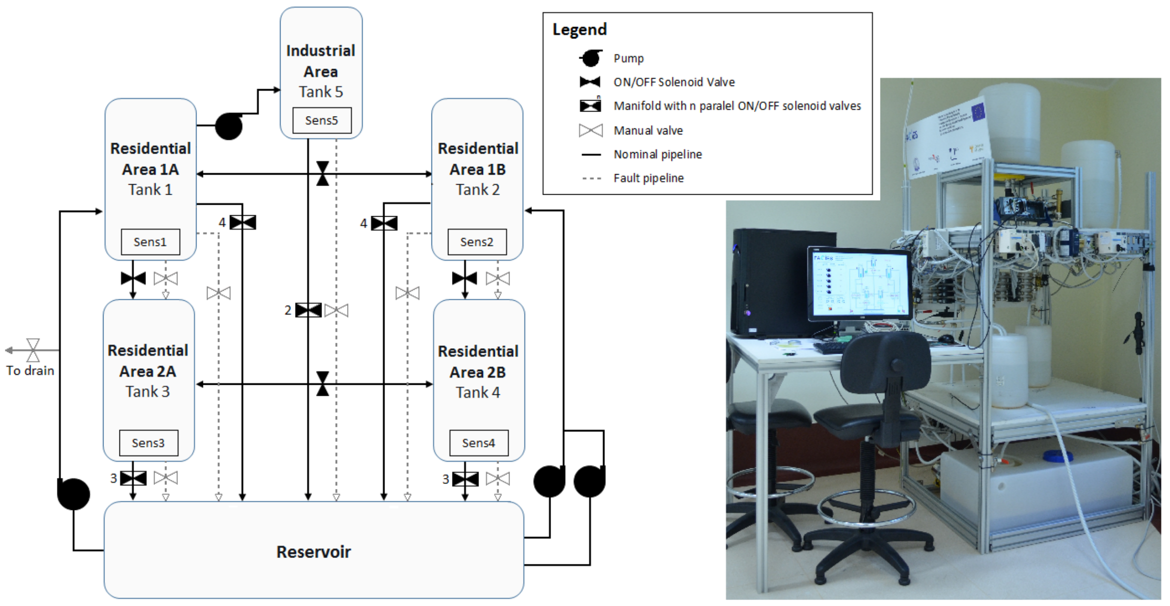
Appendix A

FACIES Testbed and Modbus protocol

A.1 Testbed Architecture

The testbed, presented in [123], has been designed as part of the EU Project "Online indentification of Failure and Attack on interdependent Critical InfrastructurES (FACIES)". It emulates a water system, encompassing the physical, control, and cyber aspects. The structure of the testbed can be divided in two interactive and interdependent layers: a physical one, consisting in the different components and main devices that reproduce the operation of a water system and a cyber one, which performs the communication, control, and data monitoring. The physical system is constituted by a number of tanks, pipelines, pumps, solenoid valves, manual valves, and level sensors, designed and disposed to better emulate the operation of the water system. On the control side, a complex architecture has been set up, able to perform a number of different tasks. It consists of two PLCs, a commercial SCADA, a network switch and other specialized modules to detect and diagnose both physical and cyber events.

Physical Structure. The system has been designed to emulate the water system of a small city composed of two residential areas, further divided into two sub-areas, and one industrial area, which is located at a different altitude. Each area is supplied by a tank, and the water demand from the costumers is reproduced by regulating the output flow from each tank by means of 20 solenoid valves connected in different manifolds, and a reservoir tank. Each tank supplies an area, and the water demand from the costumers is reproduced by regulating the output flow from each tank by means of a number of solenoid valves connected in parallel and disposed in manifolds. The FACIES testbed is depicted in Figure A.1. A typical 24-hours water demand has been scaled down to a 6 minutes scenario and implemented by the sequential activation of the valves and pumps.


Industrial Area Tank 5
Sens5
Residential Area 1A Tank 1
Sens1
Residential Area 1B Tank 2
Sens2
Residential Area 2A Tank 3
Sens3
Residential Area 2B Tank 4
Sens4
Reservoir
To drain
Legend
Pump
ON/OFF Solenoid Valve
Manifold with n paralel ON/OFF solenoid valves
Manual valve
Nominal pipeline
Fault pipeline

(a) (b)

Fig. A.1 The FACIES Testbed.

A high flexibility was desired for the system, and has been considered during the testbed design. For instance, aside the discrete modulation of the output water flow from the tanks, 14 different configurations can be obtained by exploiting the different tank connections (serial, parallel or crossed). Finally, 7 manual valves have been located at different points of the system, representing undesired output flows from the tanks or leaks along the pipelines, allowing the emulation of physical faulty conditions. In addition, different faults on the actuators (e.g., undesired disruption or activation of pumps/valves) and sensors can be introduced by properly modifying the software configuration and calibration.

Monitoring and Control System. At the lower level of the control architecture, two PLCs (Modicon M340, Schneider Electric) endowed with the proper modules interfacing to the field and for data communication, collect the real-time data from the water level sensors in each tank and control the various actuators. These execute the low-level control, programmed in Ladder Logic using Unity Pro XL v7.0 (Schneider Electric), while the high-level control has been implemented using Visual Basic, and the HMI depicted in Figure A.2 has been designed on iFIX 5 (General Electric). This interface allows the operator to monitor the state of the components of the system, to manhandle the various actuators, and to enable the different controls and scenarios that have been implemented. All the data exchanged between PLCs and SCADA is sent and stored in the local database, developed in MySQL Workbench (Oracle), which constitutes the historian. The bidirectional data transmission

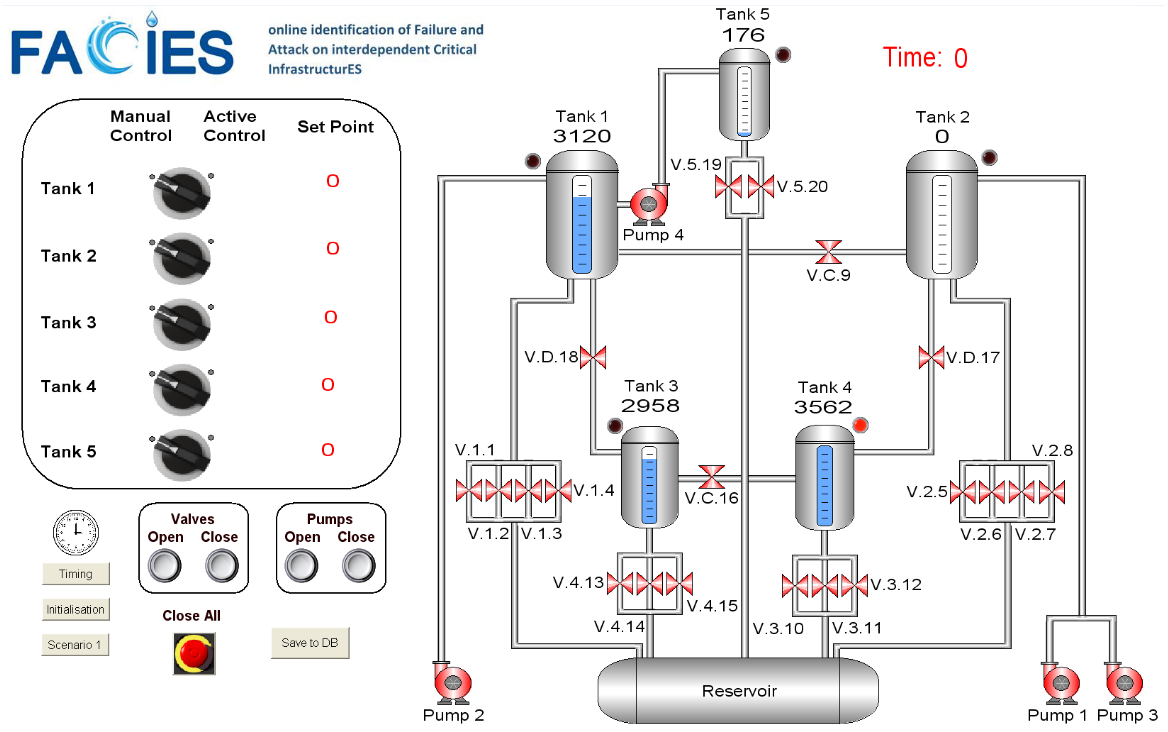


Fig. A.2 FACIES HMI designed on iFIX 5.

between PLCs, SCADA, and database takes place deploying the standardized Modbus over TCP protocol, while the remaining modules communicate through a classic TCP/IP protocol. In addition, monitoring modules performing different tasks have been included, as FDS for the early detection and identification of physical faults and a network traffic monitoring station, detecting unusual cyber events. The modules communicate through an *ad-hoc* standalone LAN. In order to perform the cyber-attacks, an additional computer (the *attacker*) has been connected to the network.

A.2 Modbus protocol

Modbus [110, 111] is an application layer messaging protocol developed by Modicon, which became the *de facto* standard since 1979 and is largely deployed for industrial applications. It provides a Client/Server communication model for devices connected on different types of networks, with a request/response fashion and offers various services specified by *Function Codes*. Generally, this protocol is used for the communication in Control zone networks of ICS. It can be implemented by asynchronous serial transmission (serial Modbus) or using TCP/IP over Ethernet (Modbus/TCP) and allows an easy communication within all types of network architectures. The Modbus/TCP protocol is implemented as a Ethernet connection:

to this end the TCP frame contains a Modbus packet as payload. The listening TCP port 502 is reserved for Modbus communications. The Modbus payload is divided into two fields: a dedicated header, called Modbus Application Protocol (MBAP), to identify the Application Data Unit (ADU) and a Protocol Data Unit (PDU). This last is further divided into two fields, the *Function Code*, which indicates in one byte the kind of action to be performed by the slave, and the *Data Field*, containing the information to perform the requested action. The *Function Codes* considered in this work are related to the functionalities implemented for the specific scenario. The read operations apply to actuator status or sensor data through the PLC, while the write commands are used to modify the state of the coils. The *Function Codes* used are:

- *Read coils (fc:01)*: used to read from 1 to 2000 contiguous status of coils (e.g., pumps and valves states). In the data field of the response message, each bit corresponds to one coil (1 ON/0 OFF).
- *Read Input Registers (fc:04)*: used to read from 1 to 125 contiguous input registers (e.g., level sensor measurements). The data field in the response message is built with two bytes per register with a binary code of the read value. In the request PDU of the read commands the starting address and the number of coils/registers to be read are specified.
- *Write Single Coil (fc:05)*: used to modify the state of a single coil. If successful, the response echoes the request after the coil requested state has been written.
- *Write Multiple Coils (fc:15)*: used to write command into a sequence of coils. The normal response echoes the function code, the starting address and the number of coils of interest.

A Modbus/TCP communication is established exploiting TCP three-way handshake mechanism and is closed when a termination request arrives or when it is locally decided by the device. Initially, a Modbus transaction is instantiated by the client. The request is then encoded by prefixing the PDU with the MBAP header with all the required information provided by the user application which is initiating the transmission demand. Finally, the IP destination address and the request ADU are passed to the TCP management module which sends it to the remote server. On the recipient side, the server has to analyze the received request, to process the required action, and to send back an appropriate response message. More specifically, the MBAP header is first parsed and the Protocol Identifier is checked, which has to be “0000” to be correct, otherwise the request is discarded. Afterwards, if a transaction is available, it is initialized and the TCP Connection Identifier, the Modbus Transaction Identifier and the Unit Identifier are stored in memory. If no more transactions are available, a Modbus exception response is sent with exception code 6 (Server busy). Then, the PDU is parsed, starting with the analysis of the function code, and the service processing activity initiates if valid, otherwise an exception response with exception code 1 (Invalid function) is built. Subsequently, the Modbus response is built and sent to the TCP management component, which returns it to the correct Modbus client. If the processing of the request was successful, the response is positive, thus the PDU is built with a function code that is the same as the request and the data field containing the results of the processing as requested by the client. If it is not, an exception response is sent providing relevant information of the error encountered during the processing. More specifically, the value “0080” is added to the request function code, and the data field is filled with an exception code indicating the reason of the error. Then the MBAP header is added as prefix of the PDU, containing the same Unit Identifier as the Modbus request, the size of the PDU and unit identifies byte indicated in the length field, the Protocol Identifier set to “0000” as given in the request, and the Transaction Identifier associated with the original request. To conclude, when the response is received by the client, it is associated with the original request by analyzing the Transaction Identifier in the MBAP header. If it corresponds to a pending transaction, it is parsed and a confirmation is sent to the user application, otherwise the message is discarded. Hence, the Protocol Identifier and Unit Identifier are first verified (the latter is discarded if the devices are directly connected on the TCP/IP network) and the function code and the response format are verified. If these are correct or an exception code is read, a positive confirmation is sent, as the command has been correctly received (but not necessarily successfully processed by the server), otherwise a negative confirmation signals the error to the user application. The timeout management relies on the TCP protocol even if it can be configured for critical applications. In the same way, on the security side, no specifications are provided or required by the standard. Currently, the data in the Modbus packets is transmitted without any type of encryption, and it is expected that the security matters are solved on the physical level, based on the specific transmission support deployed.