*Journal of the Brazilian Computer Society, 2025, 31:1,* doi: 10.5753/jbcs.2025.5358

# Containerized Testbed Architecture for Cybersecurity Data Collection on Malicious Activities in Industrial Water Systems

**Carlos Henrique Jorge** [ **Federal University of Technology – Paraná** | ***carlosjorge@alunos.utfpr.edu.br*** ]
**Luiz Nacamura Jr** [ **Federal University of Technology – Paraná** | ***nacamura@utfpr.edu.br*** ]
**Ana Cristina Barreiras Kochem Vendramin** [ **Federal University of Technology – Paraná** | ***criskochem@utfpr.edu.br*** ]

*Department of Computer Science, Federal University of Technology – Paraná (UTFPR), Av. Sete de Setembro, 3165, Curitiba, 80230-901, PR, Brazil*

**Received:** 18 December 2024 • **Accepted:** 26 May 2025 • **Published:** 08 August 2025

**Abstract** Detecting malicious activities in Information Technology (IT) is a critical component of cybersecurity and is essential for identifying threats and attacks on systems, networks, and resources. However, security of industrial control systems, an area of increasing concern due to the convergence of IT with Operational Technology (OT), requires new approaches. This work proposes a novel containerized testbed architecture for industrial water systems, implemented using open-source technologies and structured according to the Purdue reference model, which is widely used in industrial control and automation systems. The architecture aims to provide a secure and efficient environment. The experiments demonstrate that the proposed architecture enables the simulation of computational devices behavior in water systems under different scenarios, allowing data to be collected for the detection of malicious activities, such as denial-of-service attacks and command injection. The results highlight the architecture's relevance to advancing research and development efforts aimed at enhancing the security of critical water infrastructure.

**Keywords:** Industrial Control Systems; Industrial Water Systems; Cybersecurity; Containerization.

## 1 Introduction

Security testing in Information Technology (IT) systems is a well-established field, grounded in comprehensive best practices, standardized procedures, and extensive expertise. However, testing Industrial Control Systems (ICS) remains a relatively new and emerging field. The evolution of today's ICS is driven by the integration of IT capabilities into legacy Operational Technology (OT) systems, often replacing or enhancing traditional physical control operations [Bhattacharya *et al.*, 2022].

The integration of Industry 4.0 technologies into critical water infrastructure has significantly advanced the convergence of industrial networks with corporate networks and the Internet, enabling the development of smart water systems [Tuptuk *et al.*, 2021]. These systems play a vital role in meeting the current and future demands for essential public services. However, this increased interconnectivity also exposes the water treatment and sewage collection processes to new and evolving cyber risks.

In recent years, water systems have become frequent targets of cyberattacks, which can undermine the security of the water supply and pose significant threats to public health [Conti *et al.*, 2021]. Therefore, it is necessary to develop and systematically test new security techniques that can proactively mitigate cybersecurity risks.

An example of this risk is the 2021 cyberattack on the water distribution system in Oldsmar, Florida (USA). In this incident, an attacker remotely accessed a Supervisory Control and Data Acquisition (SCADA) system through a remote access application and altered the sodium hydroxide (NaOH) concentration, increasing it from 100 parts per million (ppm) to 11,000 ppm [**?**]. The effects were averted only because an employee immediately detected the abnormal change. Such attacks increases significantly as more connected components are added to industrial plants, each introducing additional vulnerabilities that expand the threat landscape [**?**]. The potential for social impact makes these infrastructures attractive targets for cybercriminals, and Internet connectivity significantly increases the likelihood of attacks [**?**].

To protect these critical infrastructures, research is being conducted on novel mechanisms to identify and prevent cyberattacks using machine learning [Vasquez *et al.*, 2017; Teixeira *et al.*, 2018]. This typically involves using a simulation environment coupled with a large dataset to train and refine algorithms [Conti *et al.*, 2021]. However, obtaining an open environment with a public and up-to-date cyberattack dataset for research purposes presents a significant challenge. Vasquez *et al.* [2017]; Teixeira *et al.* [2018]; Nicolaio *et al.* [2023] highlight that the vast majority of existing research relies on outdated public datasets or that cannot be shared with other researchers to conduct classification and data analysis studies. Given this scenario, the development of a testbed emerges as a viable alternative for the simulation and acquisition of relevant datasets.

A cybersecurity testbed is an infrastructure designed to facilitate security testing within a controlled, simulated environment [Conti *et al.*, 2021]. It is built using physical or virtual equipment, allowing the evaluation of security measures without compromising the integrity or safety of real-world systems. A customized testbed, replicating real-world solutions, allows the creation, collection and sharing of data to facilitate research in various applications, such as impact analysis, mitigation assessment, cyberphysical metrics, data and model development, security validation, cyberattack analysis, and operational training [Conti *et al.*, 2021; Bhattacharya *et al.*, 2022].

Although the literature highlights the existence of numerous testbed platforms [Conti *et al.*, 2021], only a few are specifically designed for water treatment and distribution processes. Among these, achieving reproducibility remains a significant challenge, largely due to their reliance on physical infrastructure or outdated technologies. Moreover, building an industrial testbed is a complex task that requires significant and diverse resources [Bhattacharya *et al.*, 2022].

## 2 Background

This section provides an overview of key concepts related to industrial water systems, IT/OT convergence, and Industry 4.0. It also presents the Purdue Model framework, designed for modeling and developing enterprise-wide industrial automation systems. In addition, it explores the cybersecurity of ICS, discusses the cyber risks inherent to these environments, the challenges in developing effective cybersecurity testbeds, and the role of containerized environments in simulating cybersecurity scenarios. Finally, the section concludes with a review of related work.

### 2.1 Industrial Water Systems

Industrial water systems are classified as Critical Infrastructure (CI), covering facilities, services, assets and systems whose breakdown or destruction can lead to severe social, environmental, economic, political, or international consequences, thereby threatening the security of both the state and society [**?**]. CIs are geographically distributed systems, typically controlled by ICS, where failures can significantly impact society [**?**].

Water systems are a type of Cyber–Physical System (CPS) that integrates computational and physical capabilities to control and monitor physical processes [Tuptuk *et al.*, 2021]. In the past, the security of water systems was achieved through network isolation, limiting access to their components [Knapp, 2024]. However, with the emergence of Industry 4.0, water systems, like other CI services, are increasingly adopting the philosophy of smart systems. This change promotes the incorporation of Internet-connected industrial devices and data analysis in ICS, increasing the number of sensors, control capabilities, and ensuring better integration with business processes [Tuptuk *et al.*, 2021].

More recently, the concept of smart cities has emerged with the evolution of CPS and the broader adoption of Industry 4.0, particularly in large urban centers. In smart cities, critical infrastructures play a central role in integration. These urban centers leverage technological solutions to optimize municipal operations, reduce costs, and improve the quality of life of their residents [Branquinho and Branquinho, 2021].

### 2.2 IT/OT Convergence and Industry 4.0

IT systems, encompassing devices for storing, processing, and transmitting data, have historically been distinct from OT systems, used to manage and monitor industrial processes. Traditionally, OT, including ICS, relied on isolated computing and network infrastructures to ensure secure and reliable automation, often employing mechanical devices and proprietary communication protocols [Garimella, 2018]. These systems, which include components such as SCADA systems for control, data acquisition, monitoring, and communication, as well as PLCs for the operation of sensors, pumps, and actuators, are essential for nearly all industrial processes and critical infrastructures [Hassanzadeh *et al.*, 2020].

However, the introduction of low-cost Internet-connected devices and the adoption of open communication protocols such as the Modbus Transmission Control Protocol (TCP), the Message Queuing Telemetry Transport (MQTT), and the Hypertext Transfer Protocol (HTTP) have substantially improved the integration of ICS with IT networks and the Internet [Knapp, 2024]. This convergence of IT and OT, a core feature of Industry 4.0, enables real-time data collection from sensors and actuators, facilitating remote monitoring, advanced data analysis, and more efficient decision-making [Stouffer *et al.*, 2023]. Moreover, it supports extensive data collection for Big Data analysis, optimizing industrial processes and supply chains [Garimella, 2018].

The integration of these technologies allows industries to improve capacity, efficiency and productivity, leveraging innovations such as smart factories, CPS, and interconnected value chains [Lasi *et al.*, 2014]. A key technology in this context is the Industrial Internet of Things (IIoT), which enables seamless interconnection among machines, systems, and industrial devices [Fraunholz *et al.*, 2021]. By integrating the physical and virtual worlds, the IIoT enables the development of CPS, where virtual systems exert direct influence over physical operations [Dawson, 2018].

Although this integration brings significant technological advancements, it also introduces substantial cybersecurity challenges. Increased connectivity expands the attack surface, exposing ICS to a wider range of potential cyber threats [Knapp, 2024].

### 2.3 Purdue Enterprise Reference Architecture

The Purdue Enterprise Reference Architecture (PERA), also known as the Purdue Model, was developed by the Purdue University Center for Industrial Control Systems Engineering and published in 1993. It is a widely recognized framework used in various industries to organize and structure ICS, including automation systems, production plants, manufacturing, power generation and distribution, and water treatment and distribution [Williams, 1993].

The PERA is structured into hierarchical layers, simplifying the understanding and management of ICS. This architecture includes five main layers, each offering specific functions for the different types of components [Williams, 1993]:

Level 0: Process Layer - This layer consists of physical components involved in the water treatment and distribution process, including the tangible devices that execute the actual operations. These components include sensors for measurements (e.g., pressure, flow, level), actuators to control hydraulic pumps, open and close valves, and add chemicals, and other production equipment. It is where the physical action takes place;

Level 1: Control Layer - This layer is responsible for controlling the operation of the Process layer and includes devices such as PLCs and Distributed Control Systems (DCS), and IoT systems. These devices automate the industrial process by receiving data from the Process Layer, controlling and manipulating physical devices, and interacting with components in higher layers;

Level 2: Supervisory Layer - This layer is responsible for monitoring, supervising, and coordinating operations at both the process and control levels. It includes SCADA systems, which provide functionalities such as non-compliance alerts, control panels, management systems, and operator interfaces. In addition to supervising control devices, these components also share data with higher layers;

Level 3: Industrial Perimeter - This is the highest level within the industrial network, responsible for managing the entire industrial plant. It includes production management systems, management information systems, and data historians. This level facilitates integration with the corporate network and communication with the internet, enabling the exchange of information such as reports, maintenance files, and historical analysis data;

Level 4: Business Perimeter - This layer encompasses corporate systems and their connection to the internet, integrating the industrial plant with enterprise systems. It provides core business functions, such as the orchestration of operations, controlling production schedules, managing material usage, and tracking shipping and inventory levels. This layer enables the exchange of information between the industrial processes and the broader business environment.

The PERA is essential for understanding the operation of industrial systems and implementing effective cybersecurity mechanisms. This architecture provides a clear, layered framework that facilitates the assessment of security risks at each level. Although the Purdue model was introduced in 1993, it remains relevant and accepted in the industry for the OT scenario, due to the clarity and definition of its levels. Its adaptability to Industry 4.0, as demonstrated in recent works by Ghadim *et al*. [2023], Ekisa *et al*. [2021], and Ozçelik *et al*. [2021].

Figure 1 illustrates the essential components of a water treatment and distribution process, according to the PERA.

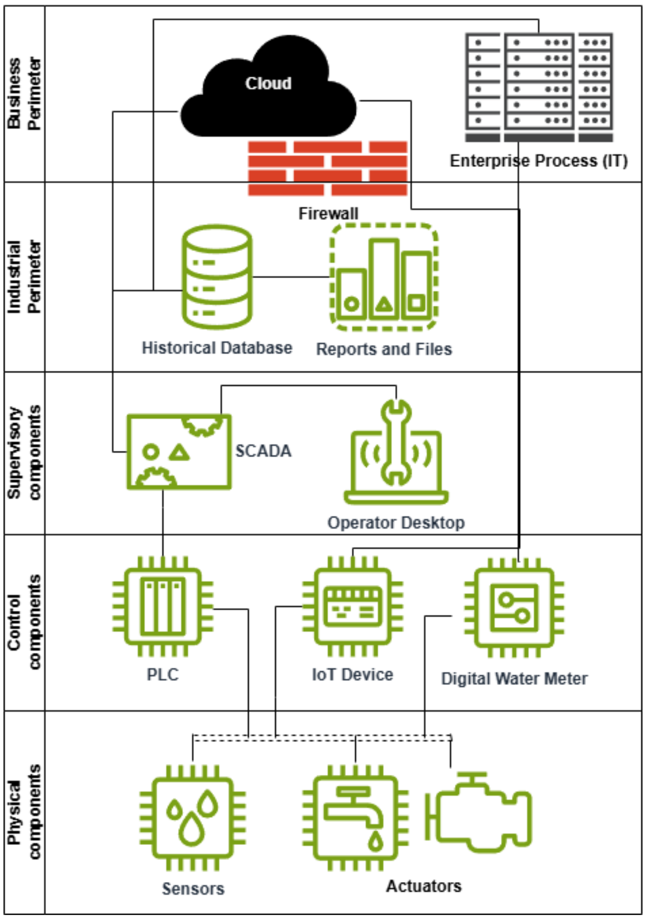


**Figure 1.** Components of a water system in a PERA representation

## 2.4 Industrial Cybersecurity

In industrial cybersecurity, the primary goal is to ensure the availability of control and monitoring systems, given the inherent risks to resources managed or affected by ICS. The traditional CIA (Confidentiality, Integrity, Availability) triad, used in information security, is adapted in industrial contexts to prioritize availability, followed by integrity and, finally, confidentiality [Knapp, 2024]. Cybersecurity, therefore, encompasses processes of risk identification and assessment, prevention and detection of threats, response to incidents, mitigation of impacts, and restoration of the digital environment post-incident [ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2023].

Historically, security concerns for ICS were primarily focused on physical attacks or the misuse of processing and maintenance facilities. However, it has become increasingly evident that ICS are also vulnerable to cyberattacks from a variety of sources, including hostile governments, terrorist groups, disgruntled employees, and other malicious actors [Radvanovsky *et al*., 2020].

Recent incidents reinforce the threats that Industry 4.0's hyper-connectivity poses if cybersecurity is not prioritized alongside efficiency and productivity. Developing new innovations, equipment, and technologies without integrating cybersecurity increases the risks associated with adopting these advancements. Consequently, the likelihood of significant disruptions or industrial sabotage from a single cyber-physical attack rises substantially [Prinsloo *et al*., 2019].

Entities in the water sector regularly face cyber incidents. These incidents can compromise their core processes of water capture, treatment, and distribution [Skiba, 2020]. Attackers use various tactics to disrupt or damage operations, with common attack types including backdoors, phishing, spoofing, adversary-in-the-middle attacks, denial-of-service, ransomware, exploitation of publicly exposed applications, and Advanced Persistent Threats (APTs) [Branquinho and Branquinho, 2021; Stouffer *et al.*, 2023]. These attacks can lead to severe consequences, such as service disruption, data loss or corruption, and reputational damage [Tuptuk *et al.*, 2021; Mullet *et al.*, 2021].

Cyberattacks on critical infrastructure can originate from a diverse set of actors with varying motivations and capabilities. These actors include individual attackers, such as lone wolves and script kiddies, as well as organized groups, like terrorist organizations and hostile nation-states [Branquinho and Branquinho, 2021]. Internal threats also pose significant risks, with malicious vendors and disgruntled employees who might exploit their system access [Ogie, 2017]. In addition, cyber-espionage actors can target water facilities to steal sensitive data and intellectual property [Mullet *et al.*, 2021].

Cyberattacks on industrial systems typically follow a four-phase process. In the reconnaissance phase, attackers gather information about the target organization [Naruoka *et al.*, 2015]. The attack phase involves gaining access to the network through techniques such as social engineering or exploiting vulnerabilities. Subsequently, attackers escalate privileges and exfiltrate sensitive data during the exfiltration phase. Finally, in the maintaining position phase, they aim to maintain persistent access for ongoing exploitation [MITRE Corporation, 2024].

Understanding these types, actors, and phases of cyberattack is essential for developing effective cybersecurity measures in the water sector. By identifying potential threats and vulnerabilities, water utilities can implement appropriate security controls and incident response plans to mitigate cyber risks, ensuring the continued safety and reliability of water services.

## 2.5 Testbed

Testbeds are controlled environments used to conduct experiments and simulate real-world systems, such as ICS [Conti *et al.*, 2021]. They offer a safe and cost-effective way to evaluate performance, security, and other aspects of these systems without risking real-world infrastructure [RNP, 2023]. In the field of cybersecurity, testbeds are essential for simulating attacks, testing security measures, and generating data to enhance attack detection algorithms [Green *et al.*, 2020].

Testbeds can be implemented using various approaches, including physical, virtual, and hybrid platforms. Physical testbeds use real hardware, offering high fidelity but often at a high cost and with greater maintenance complexity [Mathur and Tippenhauer, 2016]. Virtual testbeds utilize virtual machines and software to simulate systems, providing flexibility and scalability at lower cost, though they may lack the fidelity of physical setups [Ghadim *et al.*, 2023]. Hybrid testbeds combine elements of both, offering a balance between realism and practicality [Conti *et al.*, 2021].

In Industry 4.0, testbeds are essential for research and development of new technologies and security solutions. Effective testbeds should meet key requirements, including fidelity to real-world ICS, scalability to address evolving needs, cost-effectiveness, and compliance with industry standards like the PERA and NIST SP 800-82 [Ani and Watson, 2021]. By considering these factors, researchers can develop robust testbeds that contribute to the advancement of industrial cybersecurity [Bhattacharya *et al.*, 2022].

A cybersecurity-focused testbed for industrial water systems allows researchers to evaluate the security of water systems in a simulated environment, reducing risks to real-world infrastructure. Such a testbed enables the execution of experiments under realistic conditions, including simulated cyberattacks. It incorporates various components such as network infrastructure, SCADA systems, sensors, and actuators. These components can be configured to model water systems of different scales, enabling comprehensive security assessments across diverse scenarios. The network infrastructure may include firewalls, switches, and routers, facilitating network segmentation and the implementation of security policies.

## 2.6 Containers

Containers are a lightweight virtualization technology designed to provide a consistent and isolated environments for running applications [da Silva *et al.*, 2018]. Unlike traditional hypervisors, containers utilize kernel-level mechanisms to package application code and its dependencies, enabling efficient execution within a shared OS environment [Queiroz *et al.*, 2023]. This approach enhances scalability, optimizes resource efficiency, and allows isolated updates for individual components, making containers especially well suited for microservices architectures [Erl and Monroy, 2024].

Key advancements in operating system virtualization, such as FreeBSD *Jails* and Linux's *cgroups* and *namespaces*, enabled the development of modern container platforms such as Docker and Linux Containers (LXC) [Queiroz *et al.*, 2023]. These platforms have transformed software development and deployment by allowing applications to be packaged and executed consistently in diverse environments [Osnat, 2020].

Containerization involves packaging applications and their dependencies into isolated units, ensuring consistent execution across various environments. Its key components include the container image, which contains the application and its requirements; the container engine, responsible for managing the container lifecycle; and the container registry, which stores and distributes images [Queiroz *et al.*, 2023]. Furthermore, orchestration tools further automate container management in large-scale deployments, facilitating efficient scaling, load balancing, and self-healing [Erl and Monroy, 2024].

Docker is a popular open source platform that simplifies the creation and execution of containerized applications [Mocker and Foss, 2018]. Using a Dockerfile, developers define instructions to build a Docker image, which packages all the necessary components to run the application. This image serves as the blueprint for launch containers, runnable instances of the application [Merkel, 2014]. For multi-container applications, tools like Docker Compose enable efficient orchestration and management [Erl and Monroy, 2024].

In cybersecurity research, containerization offers an isolated and reproducible environment for simulating computational scenarios and testing security tools and techniques. This allows researchers to safely assess the effectiveness of security measures and develop new approaches to mitigate cyber threats without requiring physical hardware deployments.

## 2.7 Related work

This section examines existing research on testbeds for industrial cybersecurity, categorizing them as physical or virtual platforms. To evaluate each environment's contribution, the following criteria were considered: applicable sector, simulated industrial components, simulation software and hardware, communication protocols employed, proposed scenarios, and key findings. This analysis provides a comprehensive overview of the current state of the art and identifies gaps that the research in this paper aims to address.

### 2.7.1 Physical testbeds

Mathur and Tippenhauer [2016] introduced SWAT, a physical testbed for cybersecurity research in water treatment systems. SWAT integrates real-world components, such as PLCs, Human-Machine Interface (HMI), SCADA systems, and network devices, using Ethernet, CIP, and Modbus for communication. By capturing network traffic and sensor data during normal operation and simulated attacks, the study demonstrated the effectiveness of the testbed in evaluating threat detection and mitigation strategies.

Ahmed *et al.* [2017] introduced WADI, a physical testbed designed for security analysis in water distribution network. Building on the SWAT platform, WADI simulates the water distribution process using real-world components such as PLCs, RTUs, and sensors, which communicate via Ethernet, Modbus, and CIP protocols. This testbed enables researchers to evaluate cyberattack detection mechanisms and analyze the cascading effects of attacks on water distribution operations, utilizing a realistic scenario that encompasses water storage, distribution to consumers, and return flow.

### 2.7.2 Virtual testbeds

Almalawi *et al.* [2013] developed SCADAVT, a virtual testbed for simulating water distribution networks. Built using the CORE network emulator and Python modules, the platform incorporates Modbus TCP communication between PLCs and a SCADA server to monitor and control water levels in three tanks. The study simulated DDoS and spoofing attacks, demonstrating the impact of these attacks on water supply availability and tank volume parameters.

Carvalho and Santos [2015] proposed HONEYSCADA, a hybrid testbed combining a Windows XP-based SCADA system and a Linux-based Siemens S7 PLC simulator, designed to generate network traffic using HTTP, Simple Network Management Protocol (SNMP), and Modbus protocols. Two scenarios were evaluated: in the first scenario, the system on a local network segment, where tools such as NMAP, PLCScan, Metasploit, and Wireshark were employed for vulnerability scanning and exploitation; in the second scenario, the system was exposed to the internet for 90 days. During this period, attack data was captured and analyzed using an Intrusion Detection System (IDS).

Prates *et al.* [2021] and Meyer *et al.* [2022] described the MENTORED TESTBED, a container-based test environment built on Software Defined Infrastructure (SDI), using the Docker engine and Kubernetes. The platform was developed for cybersecurity experimentation with IoT devices. While the testbed supports automated network experiments and offers valuable metrics, such as traffic data and resource utilization, the authors highlighted challenges related to its complex configuration and data collection limitations. Researchers relying exclusively on this environment may encounter difficulties due to restricted access to infrastructure logs and a limited dataset.

Ghadim *et al.* [2023] introduced ICSSIM, a fully containerized simulation environment for cybersecurity analysis of ICS. In the simulated water bottling plant, each device, including PLCs and a SCADA system using Modbus TCP, operates within its own isolated Docker container. The authors demonstrated the impact of reconnaissance, DDoS, and command injection attacks launched from a Kali Linux attacker station. Using Wireshark for packet capture, they analyzed the effects of these attacks on system behavior, highlighting the platform's value in evaluating ICS security.

Existing research on industrial cybersecurity testbeds reveals a wide variety of physical, virtual, and hybrid platforms. Although physical testbeds such as SWAT [Mathur and Tippenhauer, 2016] and WADI [Ahmed *et al.*, 2017] offer high fidelity, their cost and complexity limit accessibility and scalability. Virtual platforms, like those presented by Almalawi *et al.* [2013] and Carvalho and Santos [2015], face challenges with reproducibility and the accurate representation of real-world industrial environments. Recent works adopting containerized technologies [Prates *et al.*, 2021; Meyer *et al.*, 2022; Ghadim *et al.*, 2023] show promise in overcoming these limitations by employing modular, container-based architectures that offer improved reproducibility, scalability, and cost-effectiveness.

Table 1 provides a summary of the key characteristics and differences among the related works discussed.

This paper proposes a novel containerized virtual testbed for simulating an entire water system, from capture to distribution. By utilizing containerization technology, the proposed platform aims to achieve reproducibility, scalability, and cost-effectiveness while integrating both traditional ICS and Industry 4.0 components, such as IoT devices. This approach will enable researchers to conduct comprehensive security assessments across various scenarios, including realistic attack simulations and the evaluation of new security technologies. Furthermore, a centralized data collection compo-

**Table 1.** Summary of Related Works

| **Work** | **Application Sector** | **Simulated Components** | **Resources** | **Protocols** | **Scenario and Results** |
|---|---|---|---|---|---|
| SWAT Mathur and Tippenhauer [2016] | Water treatment. | Allen Bradley PLCs, SCADA, sensors, actuators, and historian system. | Physical equipment. | Ethernet, CIP, Modbus | Data collection for 11 days (7 normal, 4 under attack). Effective in simulating and analyzing attacks on a physical system. |
| WADI Ahmed *et al.* [2017] | Water distribution. | PLCs, RTUs, tanks, sensors, and actuators. | Physical equipment. | Ethernet, CIP, and Modbus. | Simulates distribution processes and evaluates the impact of cascading attacks and detection effectiveness. |
| SCADA-VT Almalawi *et al.* [2013] | Water distribution. | Sensors, PLCs, and SCADA. | CORE emulator and Python applications. | Modbus TCP. | Simulation of DDoS and spoofing attacks. Interruptions and parameter manipulation identified. |
| HoneySCADA Carvalho and Santos [2015] | Various industrial sectors. | Siemens S7 PLC and SCADA. | Ubuntu, Windows XP, Nmap, PLCScan. | Modbus, HTTP, and SNMP. | Two scenarios developed (internal network and Internet). Attacks detected and logged by IDS. |
| Mentored Testbed Prates *et al.* [2021]; Meyer *et al.* [2022] | IoT. | IoT devices and botnets. | SDI, Docker containers, Kubernetes, and scripts. | HTTP and MQTT. | Automated environment. Challenges in environment setup and data collection. |
| ICSSIM Ghadim *et al.* [2023] | Water bottling. | Simulated PLCs, conveyors, and tanks. | Docker containers, Kali Linux, and Wireshark. | Modbus TCP. | Reconnaissance, DDoS, and injection attacks. Demonstrated behavioral differences between normal and compromised scenarios. |

nent will facilitate efficient data analysis and retrieval for research purposes, addressing limitations identified in existing platforms.

## 3 Containerized testbed architecture

The need for a dedicated testbed for industrial water systems arises from significant gaps identified in the literature regarding the detection and classification of malicious activities. Researchers face challenges in accessing suitable industrial environments to simulate attacks and collect data, as highlighted by several studies [Conti *et al.*, 2021; Yu and Guo, 2019; Vasquez *et al.*, 2017; Teixeira *et al.*, 2018]. This limitation underscores the importance of developing accessible and adaptable testbed architectures to advance cybersecurity research in the water sector.

The Development of a testbed to simulate water industrial system scenarios is essential for validating new technological components, simulating the impact of cyber incidents, and generating high-quality data to assess the effectiveness of security measures implemented in these facilities. The testbed must be capable of emulating the characteristics of water industrial systems, including network topology, devices, and communication protocols such as Modbus TCP, MQTT, and HTTP.

This research presents a containerized, modular, and open-source testbed for simulating an industrial water treatment and distribution system. By integrating Industry 4.0 protocols and offering centralized data collection of network traffic, logs, and system metrics, the platform enables realistic cyberattack simulations in an isolated environment. This approach supports the evaluation of security measures and the analysis of malicious activities, contributing valuable data for understanding threat behaviors. The proposed testbed's portability and scalability facilitate easy reproduction across various settings, including individual research environments and collaborative cloud deployments.

### 3.1 Architecture Design

The proposed testbed architecture utilizes a modular and flexible framework to simulate the ICS and network infrastructure of a water treatment and distribution facility. This architecture represents integration of traditional ICS methodologies with Industry 4.0 technologies. At the core of the system is the central control center, which holds the SCADA server and operator interface, facilitating communication with both PLCs and IoT devices. PLCs are employed in the water capture and treatment stages due to their resilience and reliability in harsh environments. Meanwhile, IoT devices are used in the geographically dispersed water distribution stage, where their flexibility and cost-effectiveness offer significant advantages. This design underscores the adaptability of the testbed to different operational needs and technological advancements. Furthermore, the architecture incorporates cloud systems for data integration and orchestration, further enhancing its scalability and functionality. Figure 2 illustrates the devices distributed across the water capture, treatment, and distribution process.

Following device mapping, virtual instances of network components, operating systems, and applications were created to simulate the operational scenario. This process ensures that the modeled devices seamless interaction across the various virtual instances. The proposed topology, which illustrates the configuration and connectivity of these virtualized components, is presented in Figure 3.

The virtualization of devices was achieved using open-

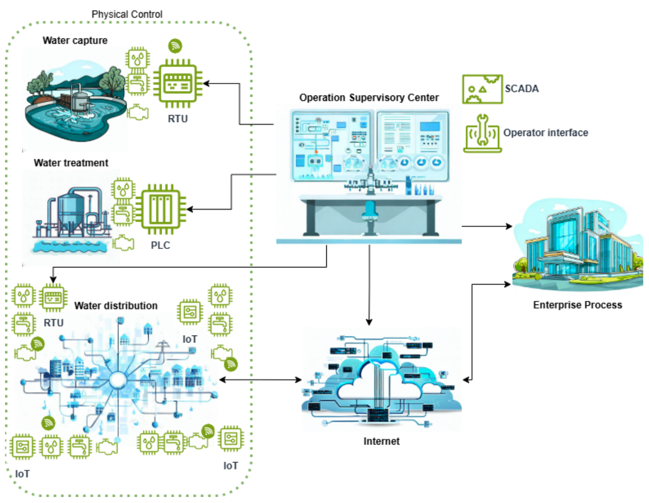


**Figure 2.** Distribution of devices within the water system

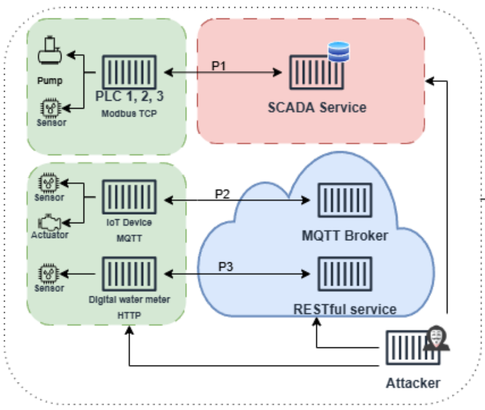


**Figure 3.** Testbed architecture

source software containers built on Docker, a containerization platform. Docker was chosen for its lightweight design, portability, and robust capability to create isolated and reproducible environments, which simplify the implementation of virtual instances. These containers represent the technological components of the water treatment and distribution system, simulating the key functionalities and characteristics of the system, such as control, supervision, data acquisition, and security devices.

The dynamic behavior of the water system is simulated by generating sensor and actuator data using specialized functions. These functions produce random yet realistic data points within predefined ranges for each device, simulating the fluctuations observed in real-world sensor readings. For example, if a previous water level reading exists, the function stores this value and then generates a small random number. This new number is added to or subtracted from the previous reading. This approach simulates natural variations in water levels over time, ensuring that the simulated data accurately reflects the dynamic behavior of the physical system and enhances the realism of the testbed environment.

The water capture, treatment, and distribution process is simulated in three subprocesses. Each subprocess utilizes a specific communication protocol for integrating data from physical control devices and supervising operations. This approach allows the simulation of different Industry 4.0 components and their interactions within the water system.

The first subprocess, SP1, simulates the initial stage of the process. It is responsible for measuring the water level of the dam for capture and pumping it to the treatment tank. At this stage, a level sensor is placed in the tank where chemicals are added, followed by pumping to the distribution network, which requires a flow sensor.

The actuators (pumps) and sensors in SP1 are controlled through the simulation of three PLCs. Each PLC is simulated using the same code and instantiated in three containers, each with specific functions to update the values of the sensors and actuators according to predefined rules:

- PLC1: responsible for monitoring the level of the capture dam (dam_level). It receives the simulated dam level from the update_level() function and updates the Modbus register associated with the dam level (DAM_LEVEL_ADDRESS).
- PLC2: monitors the treatment tank (tank_level) level and controls the activation of the capture pump. The pump activates when the tank level is below 50% and deactivated when it reaches 100%. Afterward, the PLC updates the Modbus registers related to the tank level and the status of the capture pump (TANK_LEVEL_ADDRESS and PUMP_CAPTURE_ADDRESS).
- PLC3: monitors network pressure (network_pressure) and controls the pressurizer (distribution_pump). The pressurizer activates when the pressure is below 40 psi and deactivated when it is above 80 psi. Afterward, the PLC updates the Modbus registers associated with the network pressure and the status of the distribution pump (NETWORK_PRESSURE_ADDRESS and DISTRIBUTION_PUMP_ADDRESS).

Figure 4 illustrates the interaction between the PLCs, sensors, and actuators involved in the water treatment process. Each PLC monitors specific parameters and controls corresponding actuators to maintain optimal conditions.

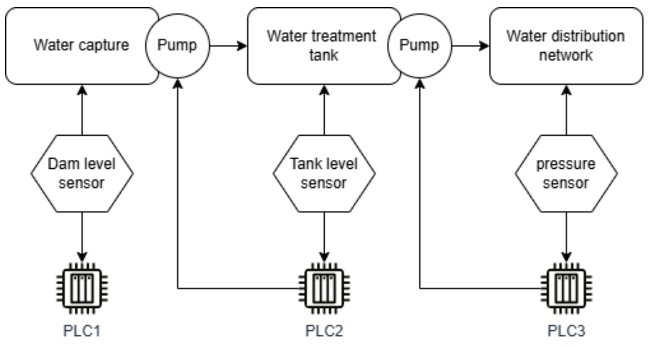


**Figure 4.** Interaction between testbed treatment process devices

In the intermediate distribution subprocess (SP2), one function simulates a sensor measuring network pressure, while another simulates the actuator pressurizing the network when necessary. These devices are controlled by IoT equipment, simulated in two containers with applications that interact as clients with an MQTT message broker. The function that simulates the sensor acts as a publisher, sending messages with the flow measurements, while the pressurizer function as a subscriber, checking the data in the queue to activate or deactivate the pressurizer based on the flow. The message broker uses the ”tp_sensor”, topic to manage communication between the sensor and actuator in SP2.

In the final distribution subprocess (SP3), a container simulates a water flow sensor in a digital water meter (IoT). This client sends information such as ID, date, time, and current water flow via an HTTP POST request to a RESTful web service. The flow metering is simulated by generating a random floating-point number between 0 and 1. The service receives this data, stores it in a database, and uses it for billing and consumption tracking.

Finally, all information is centralized in the SCADA system, which collects data via Modbus, MQTT, and HTTP protocols. It serves as the sole HMI for the operator and provides a consolidated view of the operation.

## 3.2 Attack Scenario

The attack scenario was based on the techniques and tactics represented in the MITRE ATT&CK for ICS framework[1]. The attacker, shown in Figure 3, is represented by a container running the Kali Linux operating system[2] with access to all three network segments.

Reconnaissance, an initial access technique, was performed by scanning the network or the application that is accessible to the Internet to identify open ports and vulnerabilities in the target devices. Tools like Nmap[3] were used to perform port scanning and identify services on the simulated devices.

During the attack execution phase, the goal is to compromise the services identified in reconnaissance. The following techniques were used:

- Distributed Denial of Service (DDoS) attack: aims to overwhelm the resources of target devices by consuming their resources, making them unavailable. This attack is simulated using hping3 tool, which sends a high volume of continuous TCP packets continuously, overloading the network and target devices. The use of the –rand-source parameter randomizes the source IP addresses of the packets, causing the attack to be perceived as originating from multiple distinct sources. As a result, the platform recognizes the activity as a DDoS attack.
- Web Service Attack: aims to extract data or disrupt service by exploiting vulnerabilities in the HTTP service, allowing attackers to access or corrupt data.
- Exploitation of Modbus Protocol Vulnerabilities and Remote Command Injection: targets known protocol vulnerabilities discovered during the reconnaissance to inject malicious commands, disrupting or altering their behavior of target devices.

Attack results are logged in separate files by attack type and target, and then exported to the data collector for analysis.

[1]https://attack.mitre.org/techniques/ics/
[2]https://docs.kali.org/
[3]https://nmap.org/book/man.html

## 3.3 Data collection

Platform data collection involves the capture of network packets, hardware resource utilization metrics, and containerized device logs. This proposal utilizes the Elasticsearch, Logstash, and Kibana (ELK Stack), a widely used open-source toolset for monitoring, searching, securing, and analyzing data. The ELK Stack includes four main products: Elasticsearch, Logstash, Kibana, and Beats Elasticsearch [2024].

In the data collection stage, Beats agents gather data from the host system and devices. The Packetbeat agent captures network packets, extracting relevant information such as protocols, IP addresses, ports, and payloads. Filebeat agents monitor and capture logs from specified files and directories, while the Metricbeat agent collects CPU, disk, and memory utilization.

After data collection, the information is processed and visualized using a series of interconnected tools. First, the raw data is sent to the Logstash service for extraction and transformation. The processed data are then forwarded to Elasticsearch, a centralized repository data service. Finally, Kibana is used for visualization and graph generation. The entire process is illustrated in Figure 5.

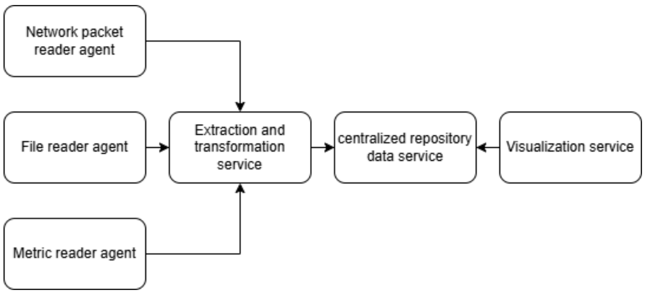

Network packet reader agent
File reader agent
Metric reader agent
Extraction and transformation service
centralized repository data service
Visualization service

**Figure 5.** Tesbed data collection process

# 4 Experiments and results

The applicability of the proposed water industrial systems testbed for cybersecurity research is demonstrated through the experiments and results presented in this section, based on the scenarios described in Subsection 3.

In accordance with PERA, the network used for the experiments was virtually segmented to separate the different simulation layers, enhancing both the security and organization of the environment. Three subnets were established for layer separation, as illustrated in Figure 6:

- Scadanet: interconnects the services of the ICS simulation, including PLC1, PLC2, PLC3, SCADAPY, and SCADABR;
- Cloud: services that simulates the IoT traffic (MQTT sensor, MQTT actuator, digital water meter) and data store components (MQTT broker and RESTful service).
- Elk: facilitates communication among ELK stack components (Elasticsearch, Logstash, and Kibana), responsible for collecting, storing, and visualizing simulation data.

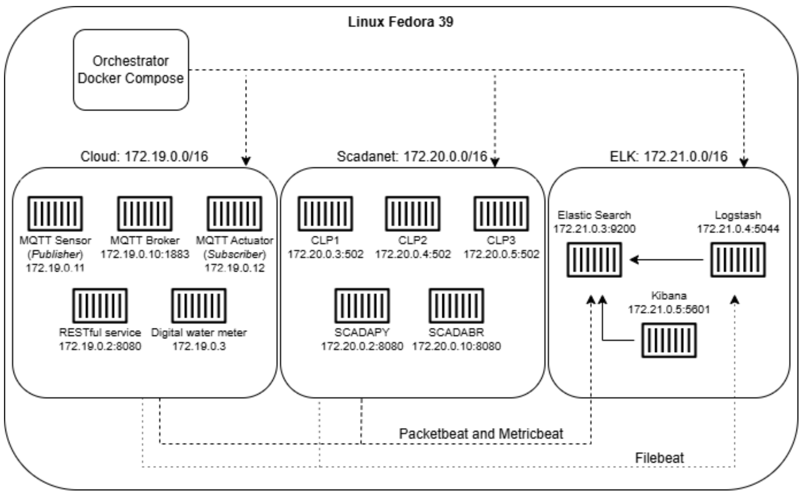

Linux Fedora 39
Orchestrator Docker Compose
Cloud: 172.19.0.0/16
Scadanet: 172.20.0.0/16
ELK: 172.21.0.0/16
MQTT Sensor (Publisher) 172.19.0.11
MQTT Broker 172.19.0.10:1883
MQTT Actuator (Subscriber) 172.19.0.12
RESTful service 172.19.0.2:8080
Digital water meter 172.19.0.3
CLP1 172.20.0.3:502
CLP2 172.20.0.4:502
CLP3 172.20.0.5:502
SCADAPY 172.20.0.2:8080
SCADABR 172.20.0.10:8080
Elastic Search 172.21.0.3:9200
Logstash 172.21.0.4:5044
Kibana 172.21.0.5:5601
Packetbeat and Metricbeat
Filebeat

**Figure 6.** Tesbed development

The testbed was implemented using Docker (version 27.1.1) as the containerization engine on a Linux Fedora 39 host server. Containers were used to simulate the computational devices of a water capture, treatment, and distribution system, as well as the data collection platform. Docker Compose (version 1.29.2) was used to automate container initialization, eliminating the need to start each container manually and reducing the occurrence of errors during this stage. The instructions for executing each container and its interaction with the host operating system were defined in the configuration file: docker-compose.yaml.

Table 2 presents the subnet configuration in the docker-compose.yml file, detailing the IP address range assigned to each subnet and the containers belonging to each segment.

Table 3 details the containerized components within the operational scenario, including the Docker image, entry point, IP address, and network port for each simulated device. This configuration supports the accurate simulation of a water treatment and distribution system.

Data collection was carried out using the Elastic Stack platform, using Filebeat, Metricbeat, and Packetbeat agents installed on the host system. These agents collect system and application logs (access logs, errors, and container events), performance metrics (CPU, memory, and disk usage), and network traffic, respectively. Container Compose simplifies orchestration using the official Elastic Search, Logstash, and Kibana images. Table 4 outlines the configuration of each container and its associated connection port.

## 4.1 Operation scenario simulation

The PLC simulation for SP1 involves three containers (PLC1, PLC2, and PLC3), each running a Python application with the pyModbusTCP.server library. This library generates random data to populate specific registers in each PLC, accessible via the Modbus TCP protocol. Each PLC manages dedicated registers: PLC1 uses register 0 to store the dam level (ranging from 0 to 99); PLC2 uses register 0 for the tank level and register 1 for capture pump status, toggling between true (activated when the tank level is below 50%) and false (deactivated when the level reaches 100%). Similarly, PLC3 uses register 0 for network pressure readings and register 1 for the distribution pump status, adhering to the same logic as the capture pump.

The communication between the pressure sensor and the pressurizer, controlled by the IoT devices in SP2, is simulated using a container with an MQTT message broker implemented with Eclipse Mosquitto software, exposed on port 1883. The MQTT Sensor container is a publisher that simulates an IoT pressure sensor. It runs a Java algorithm that generates a random pressure value between 0 and 60 every 30 seconds, publishing this data to the sensor/pressure topic. The MQTT Actuator acts as a subscriber, simulating an IoT pressurizer in another container. This container runs a Java algorithm that subscribes to the topic, retrieves messages from the sensor, and activates the pressurizer if the pressure is below 10 or deactivates it if the pressure exceeds 50. Figure 7 illustrates the interaction between the SP2 containers.

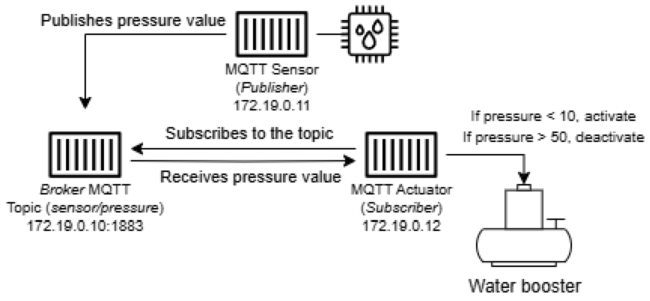

Publishes pressure value
MQTT Sensor (Publisher) 172.19.0.11
Subscribes to the topic
Receives pressure value
Broker MQTT Topic (sensor/pressure) 172.19.0.10:1883
MQTT Actuator (Subscriber) 172.19.0.12
If pressure < 10, activate
If pressure > 50, deactivate
Water booster

**Figure 7.** Interaction between the SP2 containers

The digital water meter in SP3 is simulated by a shell script application executed from a container, with the customer ID as an input parameter. The application generates a random decimal value between 0 and 1, which is added to the previous measurement, representing the water consumption. The data is sent in a JSON (JavaScript Object Notation) format, containing the following fields: customerID, currentRead, dateTime. Example of measurement submission: ”customerID”: ”123456”, ”currentRead”: 1250.78, ”dateTime”: ”2024-10-12 14:35:20”. The REST API, implemented using the Java Spring Boot framework and exposed on port 8080, receives the measurements via POST request, stores the data in the H2 database, and returns the HTTP status code 200 (OK) in case of success. To validate the re-

**Table 2.** Network Segmentation

| Network Name | IP Range | Containers |
|---|---|---|
| scadanet | 172.20.0.0/16 | PLC1, PLC2, PLC3, SCADAPY, and SCADABR |
| cloud | 172.19.0.0/16 | MQTT Broker, MQTT sensor, MQTT actuator, RESTful Service, and Water Meter |
| elk | 172.21.0.0/16 | Elastic Search, Logstash, and Kibana |

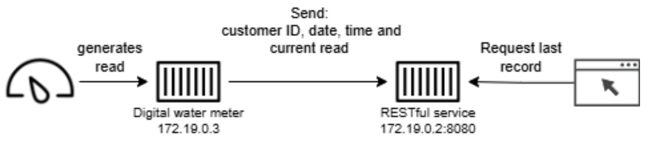
**Table 3.** Containers in the Operational Scenario

| Simulation | Image | Entry Point | IP Address | Port |
|---|---|---|---|---|
| PLC1 | python:3.8-slim | [”python”, ”/clp_simulator.py”] | 172.20.0.3 | 502 |
| PLC2 | python:3.8-slim | [”python”, ”/clp_simulator.py”] | 172.20.0.4 | 502 |
| PLC3 | python:3.8-slim | [”python”, ”/clp_simulator.py”] | 172.20.0.5 | 502 |
| SCADAPY | python:3.8-slim | [”python”, ”/scada_service.py”] | 172.20.0.2 | 8080 |
| SCADABR | eclipse-temurin:17-jdk-alpine | [”/opt/scadabr/bin/startup.sh”] | 172.20.0.10 | 8080 |
| MQTT Broker | eclipse-mosquitto | [”systemctl”,”start”,”mosquitto”] | 172.19.0.10 | 1883 |
| MQTT Sensor | eclipse-temurin:17-jdk-alpine | [”java”,”-jar”,”/app.jar”] | 172.19.0.11 | - |
| MQTT Actuator | eclipse-temurin:17-jdk-alpine | [”java”,”-jar”,”/app.jar”] | 172.19.0.12 | - |
| RESTful Service | eclipse-temurin:17-jdk-alpine | [”java”,”-jar”,”/app.jar”] | 172.19.0.2 | 8080 |
| Digital Water Meter | alpine:latest | [”./send_medicoes.sh”] | 172.19.0.3 | - |

**Table 4.** Data Collection Containers

| Service | Image | IP Address | Port |
|---|---|---|---|
| Elasticsearch | elasticsearch:8.14.2 | 172.21.0.3 | 9200 |
| Logstash | logstash:8.14.2 | 172.21.0.4 | 5044 |
| Kibana | kibana:8.14.2 | 172.21.0.5 | 5601 |

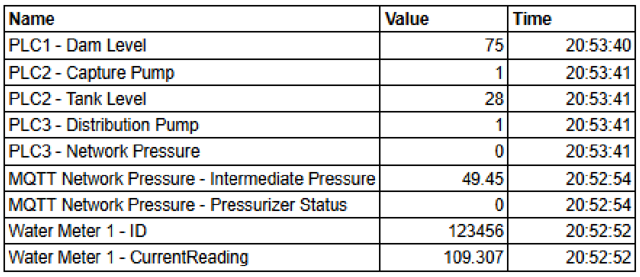
ceived information, a second REST API allows retrieval of the last record via GET request, with the customer ID as a URL parameter. Figure 8 illustrates the interaction diagram between the digital water meter and the REST API, showing the data flow and the requests.


generates read
Send: customer ID, date, time and current read
Request last record
Digital water meter 172.19.0.3
RESTful service 172.19.0.2:8080
**Figure 8.** Interaction between the digital water meter and the REST API

Initial tests were conducted with the SCADAPY container, developed to simulate a basic SCADA service. Implemented in Python using the pymodbus.client library, it initializes the scada_service.py, which generates requests every 30 seconds via the Modbus protocol on port 503 to the three PLCs. The collected data are then made accessible through a GET request to a REST API.

The SCADABR[4] project was installed in a separate container to enhance the realism of the simulation. SCADABR was chosen because it is free software that provides all the functionalities of a traditional SCADA system, supporting a wide range of industrial protocols. It is also widely used in real-world industrial facilities, making it a suitable choice to extend and adapt the simulation environment. To use SCADABR, the three PLCs and the MQTT and RESTful data concentrators were configured as data sources.

After registering all the components of the SP1, SP2, and SP3 simulations in SCADABR, the captured data is displayed in a list, as shown in Figure 9. The figure illustrates the list of registered sensors and actuators, organized by subprocess. For each entry, the name, current value, unit of measurement, and the time of the last update are displayed.



| Name | Value | Time |
|---|---|---|
| PLC1 - Dam Level | 75 | 20:53:40 |
| PLC2 - Capture Pump | 1 | 20:53:41 |
| PLC2 - Tank Level | 28 | 20:53:41 |
| PLC3 - Distribution Pump | 1 | 20:53:41 |
| PLC3 - Network Pressure | 0 | 20:53:41 |
| MQTT Network Pressure - Intermediate Pressure | 49.45 | 20:52:54 |
| MQTT Network Pressure - Pressurizer Status | 0 | 20:52:54 |
| Water Meter 1 - ID | 123456 | 20:52:52 |
| Water Meter 1 - CurrentReading | 109.307 | 20:52:52 |

**Figure 9.** Components of the SP1, SP2, and SP3 simulations listed in SCADABR

SCADABR allows the graphical representation of the configured components using the data displayed in the list shown in Figure 9. Figure 10 presents a screen displaying a graphical representation of the operational scenario, including icons for the reservoirs, pumps, sensors, and actuators of SP1, SP2, and SP3. The capture, treatment, and distribution stages are visualized through the flow of water between the elements, with colors and animations representing the status of each component. The figure also includes line graphs displaying the history of measurements from the level, pressure, and flow sensors, enabling the visualization of trends and the identification of anomalies. The graphical data visualization in a SCADA system, as observed in real-world environments, enhances the understanding of system status, enabling operators to quickly identify issues, make informed decisions, and optimize the operation of the water system.

## 4.2 Attack scenario simulation

Evaluating the platform's ability to collect relevant data during malicious activities involved conducting a series of cyberattack simulations. The attacks were carried out using a Kali Linux container, which had the Nmap, Python3, and

[4]https://www.scadabr.com.br/

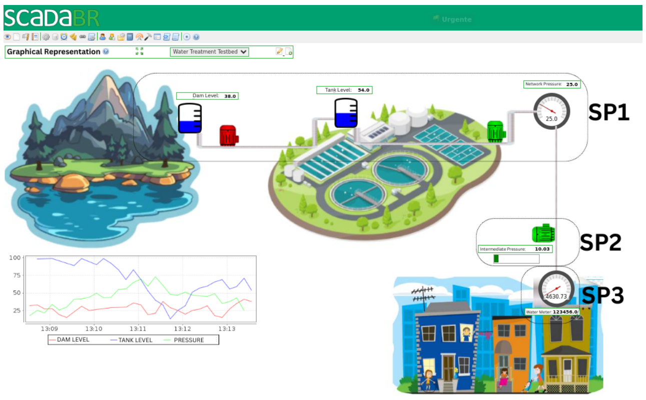
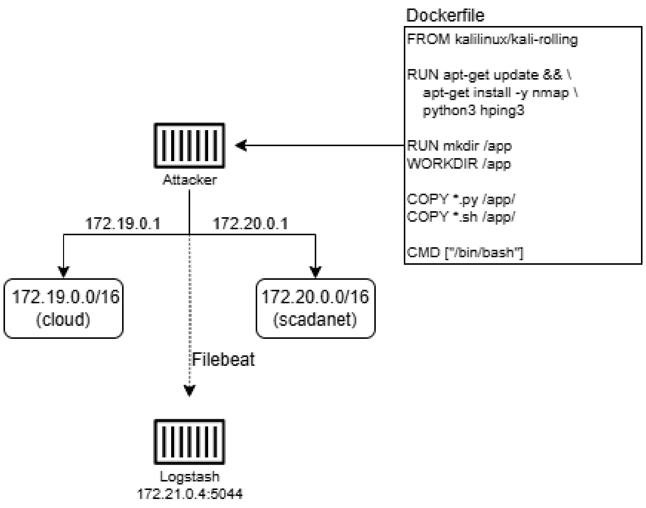


**Figure 10.** Graphical representation of the operational scenario

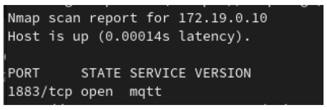
Hping3 packages installed. These tools were chosen for their effectiveness in executing the proposed attacks, including port scanning, denial-of-service attacks, and vulnerability exploitation. Python and Shell Scripts were developed to automate these attack procedures. The results were logged and sent to the data collector via the Filebeat agent. The ”attacker” container was connected to the scadanet and cloud subnets, enabling the execution of the attack. The elk subnet handled packet captures, log files, and container metrics. Figure 11 illustrates the simulation of the attack scenario, showing the interaction between the attacker container and other components of the platform.



```
FROM kalilinux/kali-rolling

RUN apt-get update && \
   apt-get install -y nmap \
   python3 hping3

RUN mkdir /app
WORKDIR /app

COPY *.py /app/
COPY *.sh /app/

CMD ["/bin/bash"]
```

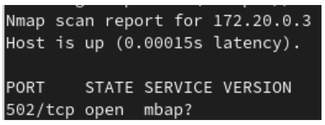
**Figure 11.** Diagram of the attack scenario simulation

Initial identification and information gathering of active devices on the scadanet (172.20.0.0/16) and cloud (172.19.0.0/16) subnets were carried out using the Nmap network scanning tool. The first scan, a quick scan was performed on both subnets to identify active devices and their IP addresses using the command nmap -sn. Results were saved in the scadanet.log and cloud.log files. The second stage involved scanning each device individually to identify open ports. For this, Nmap was used with the -sV option to identify the running services and the -O option for operating system detection. Figures 12 and 13 illustrate the results of the port scan, showing the services identified on ports 1883 (MQTT) and 502 (MBAP, Modbus Application Protocol), respectively. Figure 14 shows the packet capture during the scan, highlighting requests sent by Nmap during the tests.



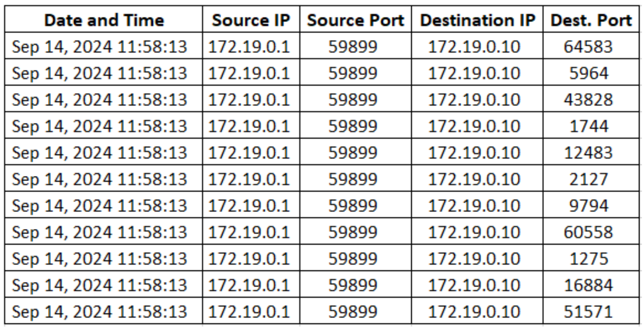
```
Nmap scan report for 172.19.0.10
Host is up (0.00014s latency).

PORT     STATE SERVICE VERSION
1883/tcp open  mqtt
```

**Figure 12.** Modbus/TCP (MBAP) port identification using Nmap



```
Nmap scan report for 172.20.0.3
Host is up (0.00015s latency).

PORT     STATE SERVICE VERSION
502/tcp open  mbap?
```

**Figure 13.** Scan results showing open ports on the device



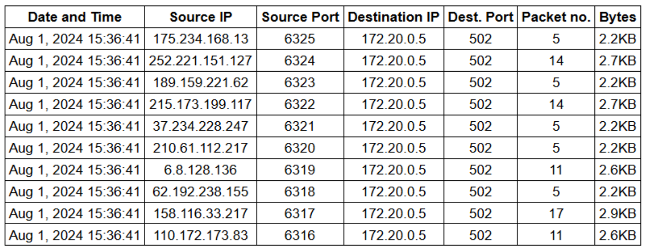
| Date and Time | Source IP | Source Port | Destination IP | Dest. Port |
|---|---|---|---|---|
| Sep 14, 2024 11:58:13 | 172.19.0.1 | 59899 | 172.19.0.10 | 64583 |
| Sep 14, 2024 11:58:13 | 172.19.0.1 | 59899 | 172.19.0.10 | 5964 |
| Sep 14, 2024 11:58:13 | 172.19.0.1 | 59899 | 172.19.0.10 | 43828 |
| Sep 14, 2024 11:58:13 | 172.19.0.1 | 59899 | 172.19.0.10 | 1744 |
| Sep 14, 2024 11:58:13 | 172.19.0.1 | 59899 | 172.19.0.10 | 12483 |
| Sep 14, 2024 11:58:13 | 172.19.0.1 | 59899 | 172.19.0.10 | 2127 |
| Sep 14, 2024 11:58:13 | 172.19.0.1 | 59899 | 172.19.0.10 | 9794 |
| Sep 14, 2024 11:58:13 | 172.19.0.1 | 59899 | 172.19.0.10 | 60558 |
| Sep 14, 2024 11:58:13 | 172.19.0.1 | 59899 | 172.19.0.10 | 1275 |
| Sep 14, 2024 11:58:13 | 172.19.0.1 | 59899 | 172.19.0.10 | 16884 |
| Sep 14, 2024 11:58:13 | 172.19.0.1 | 59899 | 172.19.0.10 | 51571 |

**Figure 14.** Packet capture during NMAP execution

### 4.2.1 Distributed Denial of Service (DDoS) attack

A DDoS attack simulation was carried out using the Hping3 tool with the following command: hping3 -S -V –rand-source IP -p PORT -d 20000, where the IP and PORT values were replaced with the IP addresses and ports of the containers in the operational scenario, as listed in Table 3. The -S parameter sends SYN packets to initiate a SYN flood attack, -V enables verbose mode to display detailed execution information, –rand-source sets random source IP addresses to hinder attacker identification, -p specifies the destination port, and -d sets the packet size, which was configured with a high value to generate a large volume of traffic. The objective of this attack simulation was to assess the container’s ability to withstand attacks aimed at service disruption, performance degradation, or excessive resource consumption.

During the DDoS attack simulation, container metrics and traffic were monitored using the ELK Stack data collection and visualization platform. Analysis of the captured packets, shown in Figure 15, revealed the presence of IP addresses external to the operational scenario. These IP addresses were simulated by the Hping3 tool using the –rand-source option, which generates random addresses to obfuscate their origin. The packets targeted port 502 on PLC3 (IP: 172.20.0.5). Figures 16, 17, and 18 show the CPU usage, memory usage, and network traffic metrics, respectively. A 300% increase in network traffic was observed during the attack simulation.



| Date and Time | Source IP | Source Port | Destination IP | Dest. Port | Packet no. | Bytes |
|---|---|---|---|---|---|---|
| Aug 1, 2024 15:36:41 | 175.234.168.13 | 6325 | 172.20.0.5 | 502 | 5 | 2.2KB |
| Aug 1, 2024 15:36:41 | 252.221.151.127 | 6324 | 172.20.0.5 | 502 | 14 | 2.7KB |
| Aug 1, 2024 15:36:41 | 189.159.221.62 | 6323 | 172.20.0.5 | 502 | 5 | 2.2KB |
| Aug 1, 2024 15:36:41 | 215.173.199.117 | 6322 | 172.20.0.5 | 502 | 14 | 2.7KB |
| Aug 1, 2024 15:36:41 | 37.234.228.247 | 6321 | 172.20.0.5 | 502 | 5 | 2.2KB |
| Aug 1, 2024 15:36:41 | 210.61.112.217 | 6320 | 172.20.0.5 | 502 | 5 | 2.2KB |
| Aug 1, 2024 15:36:41 | 6.8.128.136 | 6319 | 172.20.0.5 | 502 | 11 | 2.6KB |
| Aug 1, 2024 15:36:41 | 62.192.238.155 | 6318 | 172.20.0.5 | 502 | 5 | 2.2KB |
| Aug 1, 2024 15:36:41 | 158.116.33.217 | 6317 | 172.20.0.5 | 502 | 17 | 2.9KB |
| Aug 1, 2024 15:36:41 | 110.172.173.83 | 6316 | 172.20.0.5 | 502 | 11 | 2.6KB |

**Figure 15.** Packet capture during DDoS execution

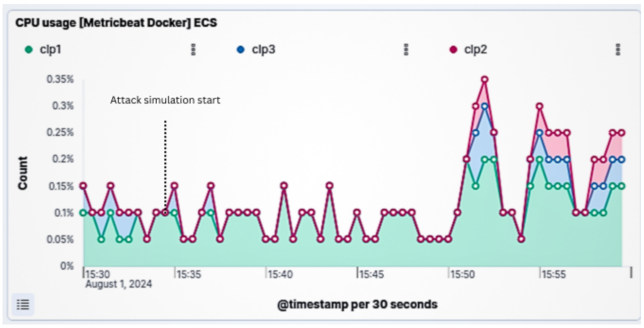


**Figure 16.** CPU metrics during the DDoS attack

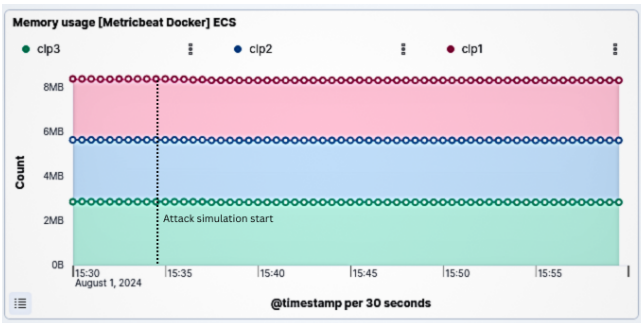


**Figure 17.** Memory metrics during the DDoS attack

#### 4.2.2 Web Service Attack

Attacks on the available HTTP services were simulated using the Nikto tool[5], an open source web server scanner that checks for various vulnerabilities. The main identified vulnerabilities are listed in Table 5:

The results of this attack reveal that the three evaluated HTTP services have vulnerabilities related to the configuration of security headers. These flaws expose the services to attacks such as cross-site scripting (XSS), clickjacking, and content injection, which could lead to information leakage, data modification, and execution of malicious code.

Additionally, the SCADABR server enables the HTTP PUT and DELETE methods, allowing an attacker to insert or remove critical information necessary for the proper functioning of the system. Furthermore, the Tomcat application service administration interface [6] is exposed, which is available at the URLs http://172.20.0.10:8080/manager and http://172.20.0.10:8080/host-manager. This exposure allows for the deployment of malicious applications and the extraction of sensitive information from the environment, such as database credentials and access to other components. These identified vulnerabilities can lead to several impacts, including the leakage of sensitive data, unauthorized data modification, and the execution of malicious code on the server.

#### 4.2.3 Exploitation of Modbus Protocol Vulnerabilities and Remote Command Injection Attacks

The Modbus TCP protocol, commonly used in ICS, has some vulnerabilities due to the lack of security mechanisms such as authentication, encryption, and access control. These weaknesses make the protocol susceptible to remote command injection. To exploit these vulnerabilities, a Python script was developed using the pymodbus.client library, with functions to read and write data to the registers of the target PLC. The read function scans a specified number of registers, determined by the –start argument, which indicates the starting register address, and reading the number of registers specified by the –count argument. If the registers have values, these are returned. The write function operates on the register specified by the –write_address argument, and the value to be inserted is passed by the –write_value argument. This operation inserts incorrect values into the PLC, which in the operational scenario can lead to faulty monitoring by the SCADA system or the erroneous activation and deactivation of actuators.

In the simulation, attacks were carried out on the three PLCs in the operational scenario (172.20.0.3, 172.20.0.4, and 172.20.0.5). The value of register 0 in each PLC, representing the dam level, tank level, and network pressure, was modified by inserting the value 100. As shown in Figure 19, the attacks successfully altered the read value. The network traffic capture reveals that the IP address 172.20.0.1 sent packets to the PLCs, indicating manipulation of the registers, as illustrated in Figure 20.

### 4.3 Discussion

This section presents the experiments conducted and the results obtained from evaluating the proposed architecture. The developed platform enabled the execution of various containers, each simulating the components of a real water capture, treatment, and distribution system, including PLCs, IoT control devices, and SCADA. Data collection was carried out using the ELK stack, which proved effective in capturing logs, system metrics, and network traffic.

The network segmentation into different subnets, using the PERA framework, enhanced the organization and security of the environment by isolating the simulation components. This strategy facilitated the simulation of attack scenarios and allowed analysis of their impact on each network segment.

The experiments demonstrated the platform's ability to collect relevant data during attack simulations. The tools used to simulate the attacks enabled the exploitation of vulnerabilities in the protocols and services of the operational scenario, highlighting the platform's effectiveness in capturing and analyzing data during security events.

The results confirm the viability of the proposed architecture as a tool for research and development of cybersecurity solutions for critical water infrastructure. The platform facilitated the simulation of various attack scenarios and the collection of data for analysis, aiding in the identification of vulnerabilities and the development of more effective protective measures.

## 5 Conclusion

Cybersecurity for critical infrastructure, particularly in water systems, faces growing challenges due to the integration of modern technologies with legacy industrial environments. This evolution has driven the development of mechanisms, such as intrusion detection systems and anomaly detection

[5]https://www.cirt.net/Nikto2
[6]https://tomcat.apache.org/

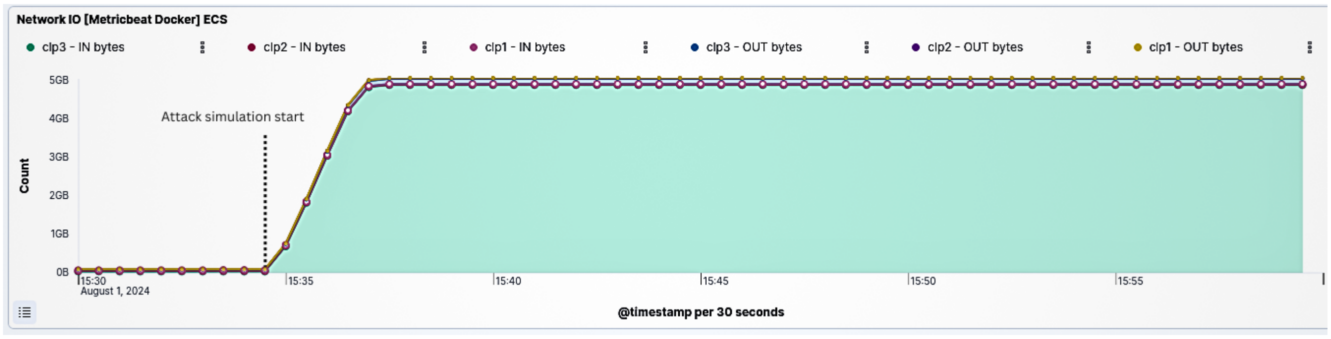
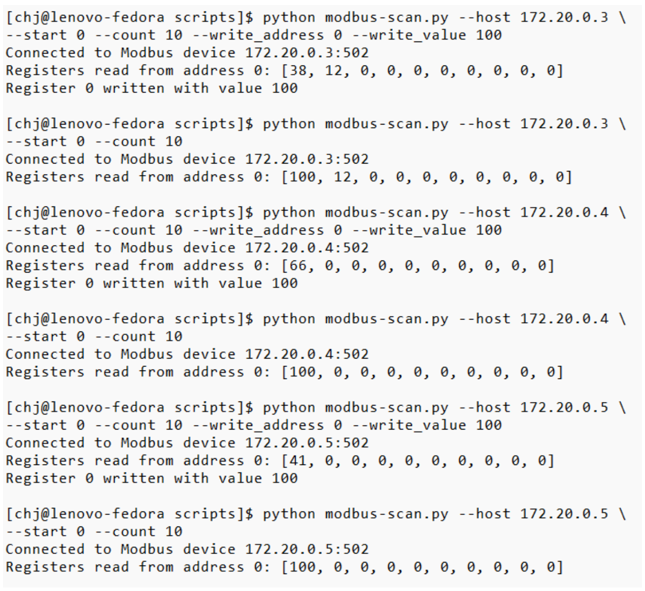


**Figure 18.** Network metrics during the DDoS attack

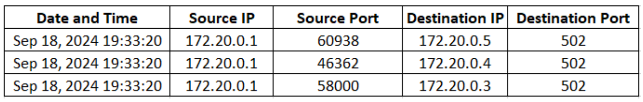
**Table 5.** Vulnerabilities found in HTTP services by the Nikto tool

| Service | IP Address | Port | Vulnerabilities |
|---|---|---|---|
| SCADAPY | 172.20.0.2 | 8080 | Lack of security headers (X-Frame-Options, X-XSS-Protection, X-Content-Type-Options). |
| SCADABR | 172.20.0.10 | 8080 | Lack of security headers (X-Frame-Options, X-XSS-Protection, X-Content-Type-Options), version information leakage, PUT and DELETE methods are allowed, exposure of default Tomcat directories (/manager, /host-manager). |
| RESTful Service | 172.19.0.2 | 8080 | Lack of security headers (X-Frame-Options, X-XSS-Protection, X-Content-Type-Options), PUT method is allowed, an unusual header (content-disposition). |



```
[chj@lenovo-fedora scripts]$ python modbus-scan.py --host 172.20.0.3 \
--start 0 --count 10 --write_address 0 --write_value 100
Connected to Modbus device 172.20.0.3:502
Registers read from address 0: [38, 12, 0, 0, 0, 0, 0, 0, 0, 0]
Register 0 written with value 100

[chj@lenovo-fedora scripts]$ python modbus-scan.py --host 172.20.0.3 \
--start 0 --count 10
Connected to Modbus device 172.20.0.3:502
Registers read from address 0: [100, 12, 0, 0, 0, 0, 0, 0, 0, 0]

[chj@lenovo-fedora scripts]$ python modbus-scan.py --host 172.20.0.4 \
--start 0 --count 10 --write_address 0 --write_value 100
Connected to Modbus device 172.20.0.4:502
Registers read from address 0: [66, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Register 0 written with value 100

[chj@lenovo-fedora scripts]$ python modbus-scan.py --host 172.20.0.4 \
--start 0 --count 10
Connected to Modbus device 172.20.0.4:502
Registers read from address 0: [100, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[chj@lenovo-fedora scripts]$ python modbus-scan.py --host 172.20.0.5 \
--start 0 --count 10 --write_address 0 --write_value 100
Connected to Modbus device 172.20.0.5:502
Registers read from address 0: [41, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Register 0 written with value 100

[chj@lenovo-fedora scripts]$ python modbus-scan.py --host 172.20.0.5 \
--start 0 --count 10
Connected to Modbus device 172.20.0.5:502
Registers read from address 0: [100, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

**Figure 19**



| Date and Time | Source IP | Source Port | Destination IP | Destination Port |
|---|---|---|---|---|
| Sep 18, 2024 19:33:20 | 172.20.0.1 | 60938 | 172.20.0.5 | 502 |
| Sep 18, 2024 19:33:20 | 172.20.0.1 | 46362 | 172.20.0.4 | 502 |
| Sep 18, 2024 19:33:20 | 172.20.0.1 | 58000 | 172.20.0.3 | 502 |

**Figure 20**

algorithms, to enhance cybersecurity. However, significant gaps remain in accurately simulating device behaviors and extracting relevant data, which limits the optimization and advancement of these protective measures.

Testbeds designed to simulate computational processes in industrial scenarios have gained increasing attention as a means of addressing existing challenges. While physical testbeds offer high fidelity, their prohibitive costs often limit accessibility for researchers. In contrast, virtualized platforms, particularly those utilizing containerization technologies, provide a flexible and cost-effective alternative, thereby streamlining test execution.

The growing demand for accessible and efficient tools to support cybersecurity research and testing in water capture, treatment, and distribution systems motivates the development of new solutions. This study proposes a containerized architecture specifically tailored for critical water infrastructure scenarios, aiming to advance cybersecurity in critical infrastructure through a novel, low-cost testbed architecture.

The study demonstrated that a containerized computing environment, leveraging open-source technologies such as Docker, SCADA BR, and the Elastic Stack, was effective in simulating the logical operational flow of a water system, including the stages of capture, treatment, and distribution. This approach enabled the efficient collection of relevant environmental data and supported the development of a robust, flexible, and low-cost platform. It is important to note that a fully accurate representation would require comprehensive mathematical modeling based on fluid dynamics and transport phenomena. However, such elements were beyond the scope of this study.

It is important to emphasize that the attacks conducted were solely intended to validate the platform and ensure effective data collection from the simulated environment. These attacks represent only a subset of potential attack types and were not designed to explore attack techniques in depth.

The results demonstrate the effectiveness of the platform in capturing network traffic data, monitoring resource utilization metrics, and logging attack records. These capabilities help identify suspicious behavior patterns and enable real-time attack detection.

The proposed testbed architecture offers a promising and cost-effective solution for enhancing cybersecurity in critical water infrastructure, enabling efficient research and testing. The source code for implementing this architecture is available in the GitHub repository at: https://github.com/henriquechj/watercstestbed.

Future work aims to expand the architecture by:

1. integrating additional communication protocols commonly used in Industry 4.0 to enhance the platform's adaptability to more complex scenarios;
2. integrating security components, such as Security Information and Event Management (SIEM) systems, IDS, and firewalls, to improve threat identification and correlation;
3. incorporating additional components, such as sensors and chemical mixers, to expand the operational scenario and enhance the architecture's realism and comprehensiveness;
4. exploring physical simulation technologies, such as digital twins, to create a more realistic test environment. This approach will enable research that considers impacts on CPS.

## Declarations

The authors acknowledge the support from the Graduate Program in Applied Computing (PPGCA) at the Federal University of Technology – Paraná (UTFPR), and from Companhia de Saneamento do Paraná (SANEPAR).

### Funding

This research did not receive any funding.

### Authors' Contributions

All authors were involved in every stage of the work and contributed equally to its completion.

### Competing interests

The authors declare that they have no conflict of interest.

### Availability of data and materials

The source code for implementing this architecture is accessible in the GitHub repository at: `https://github.com/henriquechj/watercstestbed`.

## References

Ahmed, C. M., Palleti, V. R., and Mathur, A. P. (2017). WADI: a water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*, pages 25–28, Pittsburgh Pennsylvania. ACM. DOI: 10.1145/3055366.3055375.

Almalawi, A., Tari, Z., Khalil, I., and Fahad, A. (2013). SCADAVT-A framework for SCADA security testbed based on virtualization technology. In *38th Annual IEEE Conference on Local Computer Networks*, pages 639–646, Sydney, NSW. IEEE. DOI: 10.1109/LCN.2013.6761301.

Ani, U. and Watson, J. (2021). What makes an industrial control system security testbed credible and acceptable? towards a design consideration framework. In *11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*. DOI: 10.5220/0010170301810190.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (2023). Abnt iec/ts 62443-1-1:2023: Redes de comunicação industrial — segurança do sistema e da rede - terminologia, conceitos e modelos. Technical report, ABNT. Available at:`https://www.normas.com.br/visualizar/abnt-nbr-nm/11308/abnt-iec-ts62443-1-1-redes-de-comunicacao-industrial-seguranca-do-sistema-e-da-rede-parte-1-1-terminologia-conceitos-e-modelos`.

Bhattacharya, S., Hyder, B., and Govindarasu, M. (2022). Ics-ctm2: Industrial control system cybersecurity testbed maturity model. In *2022 Resilience Week (RWS)*, pages 1–6. DOI: 10.1109/RWS55399.2022.9984023.

Branquinho, T. and Branquinho, M. (2021). *Segurança Cibernética Industrial*. Alta Books. Book.

Carvalho, R. and Santos, A. (2015). Honeypots e sua importância na defesa cibernética das infraestruturas críticas do setor elétrico. *EletroEvolução - Sistemas de Potência*, 81:30–35. Available at:`https://cigre.org.br/eletroevolucao/`.

Conti, M., Donadel, D., and Turrin, F. (2021). A Survey on Industrial Control System Testbeds and Datasets for Security Research. *IEEE Communications Surveys & Tutorials*, 23(4):2248–2294. DOI: 10.1109/COMST.2021.3094360.

da Silva, V. G., Kirikova, M., and Alksnis, G. (2018). Containers for virtualization: An overview. *Applied Computer Systems*, 23(1):21–27. DOI: 10.2478/acss-2018-0003.

Dawson, M. (2018). Cyber Security in Industry 4.0: The Pitfalls of Having Hyperconnected Systems. *Journal of Strategic Management Studies*, 10(1):19–28. DOI: 10.24760/iasme.10.$1_1$9.

Ekisa, C., Briain, D. O., and Kavanagh, Y. (2021). An open-source testbed to visualise ics cybersecurity weaknesses and remediation strategies – a research agenda proposal. In *2021 32nd Irish Signals and Systems Conference (ISSC)*, pages 1–6. DOI: 10.1109/ISSC52156.2021.9467852.

Elasticsearch (2024). Elastic stack. `https://www.elastic.co/pt/elastic-stack`.

Erl, T. and Monroy, E. B. (2024). *Computação em Nuvem: Conceitos, Tecnologia, Segurança e Arquitetura*. Bookman, Porto Alegre, 2 edition. Book.

Fraunholz, D., Zimmermann, M., and Schotten, H. D. (2021). An adaptive honeypot configuration, deployment and maintenance strategy. *CoRR*, abs/2111.03884. DOI: 10.48550/arXiv.2111.03884.

Garimella, P. K. (2018). IT-OT Integration Challenges in Utilities. In *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, pages 199–204, Kathmandu. IEEE. DOI: 10.1109/CCCS.2018.8586807.

Ghadim, A. D., Balador, A., Moghadam, M. H., Hansson, H., and Conti, M. (2023). ICSSIM — a framework for building industrial control systems security testbeds. *Computers in Industry*, 148. DOI: 10.1016/j.compind.2023.103906.

Green, B., Derbyshire, R., Knowles, W., Boorman, J., Ciholas, P., Prince, D., and Hutchison, D. (2020). ICS testbed tetris: Practical building blocks towards a cyber security resource. In *13th USENIX Workshop on Cyber Security Experimentation and Test (CSET 20)*. USENIX Association. Available at:`https://www.usenix.org/conference/cset20/presentation/green`.

Hassanzadeh, A., Rasekh, A., Galelli, S., Aghashahi, M., Taormina, R., Ostfeld, A., and Banks, K. (2020). A Review of Cybersecurity Incidents in the Water Sector. *Journal of Environmental Engineering*, 146(5):03120003. DOI: 10.1061/(ASCE)EE.1943-7870.0001686.

Knapp, E. (2024). *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Syngress. Book.

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., and Hoffmann, M. (2014). Industry 4.0. *Business & Information Systems Engineering*, 6(4):239–242. DOI: 10.1007/s12599-014-0334-4.

Mathur, A. P. and Tippenhauer, N. O. (2016). SWaT: a water treatment testbed for research and training on ICS security. In *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*, pages 31–36, Vienna, Austria. IEEE. DOI: 10.1109/CySWater.2016.7469060.

Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, (239):2. Available at: `https://www.seltzer.com/margo/teaching/CS508.19/papers/merkel14.pdf`.

Meyer, B. H., Gemmer, D. D., Andrade, A. M., Mello, E. R. d., Nogueira, M., and Wangham, M. S. (2022). Criação de Redes Virtuais no MENTORED Testbed: Uma Análise Experimental. In *Anais do I Workshop de Testbeds (WTESTBEDS 2022)*, pages 24–35, Brasil. Sociedade Brasileira de Computação. DOI: 10.5753/wtestbeds.2022.223308.

MITRE Corporation (2024). Ics attack techniques. https://attack.mitre.org/techniques/ics/.

Mocker, K. M. and Foss, S. P. (2018). *Docker: Up and Running: Shipping Reliable Containers in Production*. O'Reilly Media. Book.

Mullet, V., Sondi, P., and Ramat, E. (2021). A Review of Cybersecurity Guidelines for Manufacturing Factories in Industry 4.0. *IEEE Access*, 9:23235–23263. DOI: 10.1109/ACCESS.2021.3056650.

Naruoka, H., Matsuta, M., Machii, W., Aoyama, T., Koike, M., Koshijima, I., and Hashimoto, Y. (2015). ICS Honeypot System (CamouflageNet) Based on Attacker's Human Factors. *Procedia Manufacturing*, 3:1074–1081. DOI: 10.1016/j.promfg.2015.07.175.

Nicolaio, I., Munaretto, A., and Fonseca, M. (2023). Uma proposta de detecção de ataques cibernéticos em sistemas de controle industrial (ics). In *Anais do XXVIII Workshop de Gerência e Operação de Redes e Serviços*, pages 153–166, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/wgrs.2023.765.

Ogie, R. I. (2017). Cyber security incidents on critical infrastructure and industrial networks. In *Proceedings of the 9th International Conference on Computer and Automation Engineering*, ICCAE '17, page 254–258, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3057039.3057076.

Osnat, R. (2020). A brief history of containers: From the 1970s till now. Available at:`https://www.aquasec.com/blog/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016/`.

Ozçelik, I., Iskefiyeli, M., Balta, M., Akpinar, K. O., and Toker, F. S. (2021). Center water: A secure testbed infrastructure proposal for waste and potable water management. In *2021 9th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–7. DOI: 10.1109/ISDFS52919.2021.9486364.

Prates, N. G., Andrade, A. M., Mello, E. R. d., Wangham, M. S., and Nogueira, M. (2021). Um Ambiente de Experimentação em Cibersegurança para Internet das Coisas. In *Anais do VI Workshop do Testbed FIBRE (WFIBRE 2021)*, pages 68–79, Brasil. Sociedade Brasileira de Computação. DOI: 10.5753/fibre.2021.15771.

Prinsloo, J., Sinha, S., and von Solms, B. (2019). A Review of Industry 4.0 Manufacturing Process Security Risks. *APPLIED SCIENCES-BASEL*, 9(23). DOI: 10.3390/app9235105.

Queiroz, R., Cruz, T., Mendes, J., Sousa, P., and Simões, P. (2023). Container-based virtualization for real-time industrial systems—a systematic review. *ACM Comput. Surv.*, 56(3). DOI: 10.1145/3617591.

Radvanovsky, R., Brodsky, J., and Look, B. G. (2020). *Handbook of Scada/Control Systems Security*. Routledge. DOI: 10.1201/b19545.

RNP (2023). Testbed. https://www.rnp.br/servicos/testbeds.

Skiba, R. (2020). Water industry cyber security human resources and training needs. *International Journal of Engineering Management*, 4:11–16. DOI: 10.11648/j.ijem.20200401.12.

Stouffer, K. A., Pease, M., Tang, C., Zimmerman, T., Pillitteri, V. Y., Lightman, S., Hahn, A., Saravia, S., Sherule, A., and Thompson, M. (2023). Nist sp 800-82r3: Guide to operational technology (ot) security. DOI: h10.6028/NIST.SP.800-82r3.

Teixeira, M., Salman, T., Zolanvari, M., Jain, R., Meskin, N., and Samaka, M. (2018). SCADA System Testbed for Cybersecurity Research Using Machine Learning Approach. *Future Internet*, 10(8):76. DOI: 10.3390/fi10080076.

Tuptuk, N., Hazell, P., Watson, J., and Hailes, S. (2021). A systematic review of the state of cyber-security in water systems. *Water (Switzerland)*, 13(1). DOI: 10.3390/w13010081.

Vasquez, G., Miani, R., and Zarpelão, B. (2017). Flow-based intrusion detection for scada networks using supervised learning. *Anais do XVII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 168–181. DOI: 10.5753/sbseg.2017.19498.

Williams, T. (1993). The purdue enterprise reference architecture. *IFAC Proceedings Volumes*, 26(2, Part 4):559–564. DOI: 10.1016/S1474-6670(17)48532-6.

Yu, X. and Guo, H. (2019). A Survey on IIoT Security. In *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, pages 1–5, Singapore. IEEE. DOI: 10.1109/VTS-APWCS.2019.8851679.